Universidad Central "Marta Abreu" de Las Villas
Facultad de Matemática – Física – Computación
Departamento de Ciencia de la Computación

Un modelo híbrido para la asignación redundante de fragmentos en el diseño de bases de datos distribuidas.

Tesis presentada en opción al grado científico de
Máster en Ciencia de la Computación

Autor: Darién Rosa Paz

Tutores: Dra. Luisa M. González González
Dr. Abel Rodríguez Morffi

Santa Clara
2008

RESUMEN

El auge alcanzado por los Sistemas de Información distribuidos y el desarrollo de los manejadores de bases de datos comerciales han incrementado la difusión de los Sistemas de Bases de Datos Distribuidas (SBDD). Debido a la complejidad del proceso de diseño de la distribución de los datos, las diferentes propuestas de solución presentadas hasta la fecha han coincidido en dividirlo en dos fases seriadas: la fragmentación y la asignación de los fragmentos en los sitios de la red.
En la presente investigación se hace un estudio del estado del arte en el diseño de la asignación de fragmentos, y se propone un modelo híbrido que integra una técnica de Aprendizaje Reforzado con Algoritmos Genéticos para dar solución al problema de asignación. Este modelo es implementado en un módulo de una herramienta de ayuda a la asignación lógica y física de los fragmentos a los correspondientes sitios de procesamiento.

SUMMARY

The boom reached by the distributed information systems and the development of the commercial data base managers have greatly increased the diffusion of Distributed Data Base Systems (DDBS). Due to the complexity of the data distribution design process, the different approaches to solve the allocation problem, presented so far, have coincided in dividing it in two serial stages: fragmenting and allocating the resulting fragments in the network. Herein, we carry out a study about the state of the art of designing fragment allocation, and then proposing a hybrid model which incorporates a Reinforcement Learning Technique with Genetic Algorithms in order to solve the allocation problem. This model is implemented in a module of a tool for logical and physical fragment allocation to the corresponding processing sites.

TABLA DE CONTENIDOS

Introducción 1

Capítulo I. La asignación de fragmentos en el diseño de bases de datos distribuidas 6

1.1 Acerca de las bases de datos distribuidas 6

1.2 Diseño de la distribución 9

1.3 Problema de asignación 10

1.3.1 Modelo de asignación 11

1.3.2 Soluciones ofrecidas al problema de asignación 16

1.4 Conclusiones parciales 26

Capítulo II. Módulo de asignación 28

2.1 Consideraciones realizadas al modelo de asignación 28

2.1.1 Condición de réplica 28

2.1.2 Tratamiento de restricciones 28

2.1.3 Informaciones adicionales 28

2.2 Aspectos generales del diseño y la implementación del asistente 29

2.2.1 Diagrama de clases 29

2.2.2 Definición de las clases 30

2.3 Análisis de la complejidad temporal del modelo 42

2.4 Generación de scripts para SQL Server 43

2.5 Descripción del asistente 45

2.6 Conclusiones parciales 46

Capítulo III. Validación del modelo 47

3.1 Caso de estudio: Sistema de reservación para una aerolínea 47

3.1.1 Descripción del caso de estudio 47

3.1.2 Modelación conceptual 47

3.1.3 Modelación lógica 51

3.1.4 Modelación física 56

3.2 Casos de prueba 56

3.3 Resultados experimentales 57

3.4 Conclusiones parciales 58

Conclusiones 59

Recomendaciones 60

Bibliografía 61

Anexos

INTRODUCCION

En los últimos tiempos se ha producido un auge notable de los Sistemas de Información distribuidos, debido, entre otros factores, a la gran cantidad de información que cada día se va generando, a los avances en la tecnología de las comunicaciones y al decremento en los costos de las computadoras.

Los sistemas distribuidos se adaptan mejor a las necesidades de las organizaciones descentralizadas ya que reflejan más adecuadamente su estructura y tienen importantes ventajas con respecto a los centralizados (Őzsu y Valduriez, 1991). La descentralización se justifica desde el punto de vista tecnológico, ya que permite autonomía local y promueve la evolución de los sistemas y los cambios en los requerimientos de los usuarios, proporciona una arquitectura de sistemas simple, flexible y tolerante a fallos; además de ofrecer buenos rendimientos. Otra razón a considerar a favor de la descentralización es la distribución de los accesos a la memoria, tanto de entrada como de salida, donde se almacena finalmente la información. Igualmente, las redes de computadoras trabajan cada vez a mayores velocidades, abriendo una puerta a la distribución del trabajo y la información (Őzsu y Valduriez, 1999). Al mismo tiempo añaden nuevas complejidades a su diseño e implementación.

De este modo los Sistemas de Gestión de Base de Datos (SGBD) han avanzado de centralizados a distribuidos, donde tanto los datos como las aplicaciones se encuentran físicamente distribuidos en los diferentes nodos de una red de computadoras, aunque lógicamente pertenecen a una base de datos única; de manera que la distribución sea transparente al usuario. El objetivo fundamental de los Sistemas de Bases de Datos Distribuidas es integrar la manipulación de datos para que sean presentados al usuario como una única colección de datos global y coherente (Őzsu y Valduriez, 1999).

Según Ceri y Pelagatti (Ceri y Pelagatti, 1984), una Base de Datos Distribuida (BDD) es una colección de múltiples Bases de Datos (BD), lógicamente interrelacionadas, distribuidas sobre una red de computadoras.

El diseñador de una BDD tiene la tarea de definir la distribución de los datos entre los sitios de la red. Las diferentes técnicas de distribución se basan generalmente en la semántica de los datos, y se rigen por principios idénticos que las Bases de Datos Centralizadas (BDC), incorporando otros detalles particulares, como la

fragmentación de los esquemas y su posterior localización en los diferentes sitios de la red; a este problema se le conoce como diseño de la distribución. Para garantizar estas metas es necesario desarrollar soluciones óptimas o con un grado aceptable de desempeño que suponen la implementación de algoritmos sumamente engorrosos de una elevada complejidad computacional. Dado que los datos pueden estar replicados, el control de concurrencia y los mecanismos de recuperación son mucho más difíciles que en un sistema centralizado.

En el diseño de la distribución influyen muchos factores relacionados con la BD, las aplicaciones que acceden a la misma, la comunicación de la red, y el sistema de computadoras que la soporta; lo que hace que sea muy complicada la formulación de un problema de distribución. Por tanto, es necesario decidir cómo fragmentar y distribuir los datos sobre los diferentes sitios y cuáles de estos datos deben ser replicados. Debido a la complejidad de este proceso, existe un gran déficit de metodologías y herramientas de apoyo al mismo.

La ubicación de los datos afecta notablemente el desempeño de los sistemas distribuidos debido a que el tiempo y el costo requeridos para el procesamiento de las solicitudes dependen en gran parte del lugar donde se encuentren almacenados, ya sea en un solo nodo o que estén distribuidos en varios sitios de la red. El logro de una máxima localidad, o sea, que los datos se encuentren en el sitio donde más se necesiten, incide en la eficiencia del acceso a los mismos; para esto es necesario tener un buen soporte para la fragmentación y replicación de información. También se puede incrementar la eficiencia mediante la explotación del paralelismo entre operaciones, especialmente en el caso de varias consultas independientes que se pueden procesar por sitios diferentes. Otra alternativa para mejorar la eficiencia es que el procesamiento de una sola consulta puede consistir en un plan de ejecución que involucre varios sitios y procesarse de manera más rápida. La distribución también facilita la escalabilidad, es decir, la evolución del sistema sin que se vea degradado su desempeño.

La modelación del problema de asignación o ubicación tratada en la bibliografía llega hasta el nivel lógico porque el nivel físico depende fuertemente de los motores de BD; no obstante pueden ofrecerse algunas soluciones para el nivel físico teniendo en cuenta los motores más populares.

Aunque el problema de asignación de fragmentos en BDD ha sido estudiado ampliamente por una gran cantidad de investigadores y muchos de ellos han propuesto modelos y diseñado algoritmos para su solución (Yoshida et al., 1985; Lin et al., 1993; Lin y Orlowska, 1995; Fung et al., 1997; Tamhankar y Ram, 1998; Fung et al., 2003; Maskarinec y Neumann, 2003; Lee y Baik, 2004; Ma et al., 2005; Pérez et al., 2005; Roosta, 2005; Gu et al., 2006; Ma et al., 2006), aun no se ha resuelto totalmente debido a su dificultad tanto en la formulación del modelo matemático como en la fase de solución del mismo; ya que todos hacen uso de consideraciones y modelan diferentes parámetros, por lo que son aplicables solo bajo ciertas especificaciones (Mei y Sheng, 1992). En muchos de ellos se impone la condición de que cada fragmento se encuentre almacenado en un solo sitio de la red, de esta forma se desaprovechan las ventajas de la replicación a cambio de que el espacio de soluciones disminuya considerablemente.

Para dar solución a estos complejos modelos de optimización se emplean enfoques exactos y heurísticos, los cuales han resuelto parte del espectro de problemas reales. Los métodos exactos resultan factibles para solucionar problemas pequeños, no así para problemas mayores que son tratados mediante técnicas heurísticas.

En el pasado reciente se han desarrollado trabajos de investigación que tratan el problema de la asignación de fragmentos en bases de datos distribuidas. En (González, 2001) y (Hernández, 2005) se aplican las técnicas SARSA y Q-Learning respectivamente, obteniéndose resultados significativos, aunque no se tuvo en cuenta la replicación de fragmentos.

En (Rosa, 2006) se aplican los métodos de Algoritmos Genéticos y Q-Learning para dar solución al modelo de ubicación planteado por Özsu (Őzsu y Valduriez, 1991), teniendo en cuenta la replicación de fragmentos. Se pudo comprobar que ambos métodos son computacionalmente tratables y son factibles de aplicar en la solución del problema de ubicación. Sin embargo, a medida que aumenta el tamaño de la instancia del problema, las soluciones obtenidas se alejan de la solución óptima y evidencian problemas de convergencia. El resultado final de dicha investigación fue la implementación de la herramienta ALLOCATOR para llevar a cabo el diseño de la asignación, haciendo uso de las técnicas mencionadas anteriormente. Otra limitación de esta herramienta es que obtiene soluciones hasta

el nivel lógico solamente.

Dando continuidad a las investigaciones anteriores y ante las limitaciones mencionadas, se justifica la elaboración e implementación de un nuevo método que integre técnicas heurísticas para obtener soluciones al problema de la asignación de fragmentos, donde se tenga en cuenta la replicación, y se mantenga la complejidad computacional de los métodos anteriores, obteniéndose soluciones hasta el nivel físico.

Por todo lo anterior se formula el siguiente **problema de investigación**: La solución del problema de asignación de fragmentos en el diseño de bases de datos distribuidas utilizando Algoritmos Genéticos y Q-Learning ha mostrado numerosas ventajas respecto a soluciones precedentes, pero aun puede ser mejorado si se combinan ambos métodos adecuadamente.

En aras de solucionar la situación planteada, se establece la siguiente **hipótesis**: Un modelo híbrido que integre el método Q-Learning con Algoritmos Genéticos es más propicio que el empleo individual de cualquiera de ellos para resolver el problema de asignación de fragmentos en el diseño de bases de datos distribuidas.

Para dar cumplimiento a la hipótesis planteada anteriormente, se define el siguiente sistema de objetivos:

Objetivo general

Resolver el problema de asignación de fragmentos en el diseño de bases de datos distribuidas mediante un modelo híbrido que integre un método de Aprendizaje Reforzado con Algoritmos Genéticos.

Objetivos específicos

1. Crear un modelo para la asignación de fragmentos que considere la replicación, basado en un algoritmo genético cuya población inicial se obtenga mediante el método Q-Learning.
2. Incluir en la herramienta ALLOCATOR, el módulo que lleve a cabo el proceso de asignación lógica de fragmentos.
3. Obtener los esquemas físicos acordes con la asignación lógica propuesta por la herramienta ALLOCATOR.

La **novedad científica** del presente trabajo consiste en el empleo del método Q-Learning para la generación de la población inicial de un Algoritmo Genético que

realiza el proceso de asignación de fragmentos en el diseño de BDD considerando la replicación.

El **valor práctico** de los resultados de este trabajo está dado por la incorporación de un nuevo método a la herramienta ALLOCATOR que lleva a cabo el proceso de asignación de fragmentos hasta el nivel físico y que se integra al sistema SIADBDD (Rodríguez, 2007) de ayuda al diseño de bases de datos distribuidas.

Esta tesis se encuentra estructurada en tres capítulos: en el capítulo I se expone una panorámica del estado del arte y los aspectos más relevantes en el proceso de asignación de fragmentos en el diseño de BDD. Se presenta el modelo de asignación y se analizan los diferentes enfoques propuestos para su solución. En el capítulo II se describe la integración de las técnicas de Algoritmos Genéticos y Q-Learning para dar una solución al modelo de asignación planteado en el capítulo I; además se aborda la obtención del modelo físico de asignación mediante scripts que serán ejecutados en servidores de Microsoft SQL Server. En el capítulo III se valida el modelo obtenido como resultado de este trabajo mediante su aplicación en la solución de un caso de estudio y a través de pruebas realizadas al mismo. Se comparan los resultados alcanzados con los obtenidos mediante otros métodos heurísticos que solucionan el problema de asignación y con los de otros modelos reportados en la bibliografía. La tesis culmina con conclusiones, recomendaciones, bibliografía y anexos.

CAPITULO I. LA ASIGNACIÓN DE FRAGMENTOS EN EL DISEÑO DE BASES DE DATOS DISTRIBUIDAS

En este capítulo se expone una panorámica del estado del arte y los aspectos más relevantes en el proceso de asignación de fragmentos en el diseño de BDD. Se analizan los diferentes enfoques que se le ha dado al problema de asignación, realizando un análisis crítico de los principales trabajos reportados en la bibliografía.

1.1 Acerca de las bases de datos distribuidas

Se pudiera precisar el concepto de BDD dado en la Introducción como un conjunto de esquemas y ocurrencias de datos fragmentados, replicados o interrelacionados de alguna manera, lo que obviamente depende de las características de las aplicaciones que se ejecuten sobre esos datos, asumiendo que existe al menos una aplicación global que accede a los mismos.

Para el propósito de esta investigación, la distribución de datos en un sistema de computadoras distribuidas pretende implementar un modo de distribución de acuerdo a los datos usados por varias aplicaciones que pueden estar distribuidas en diferentes sitios de procesamiento. Este modo de distribución pondera la localidad de procesamiento y una reducción del tráfico en la red.

Según Özsu y Valduriez (Őzsu y Valduriez, 1991), existen varias alternativas para distribuir la BD acorde con la fragmentación y/o replicación que se aplique a la misma: fragmentada y no replicada, no fragmentada y replicada, o fragmentada y replicada.

El término fragmentación se emplea para referirse al proceso de división de un esquema de una BD en varias porciones de acuerdo con un criterio de fragmentación específico determinado por las aplicaciones que acceden a la BD, mientras que la replicación se utiliza en relación con la existencia de copias de esquemas o fragmentos de esquemas almacenados en varios sitios.

Las BDD como sistemas integrados, asumen que la distribución sea transparente a los usuarios (Őzsu y Valduriez, 1999). La transparencia de la distribución se refiere a ocultar la distribución a los usuarios e incluso a los programas; donde no será necesario indicar el lugar de asignación de los recursos, ni cambiar nombres durante la migración. También se aplica a la replicación, donde lograr mayor disponibilidad y confiabilidad incluye aspectos tan complicados como decidir cuánto

replicar, dónde ubicar las copias, decidir qué copia usar ante una solicitud, y cómo asegurar la propagación de las actualizaciones. El diseñador de la BD debe considerar cuidadosamente la distribución de los datos incluyendo su replicación, aun cuando el sistema tenga un alto grado de transparencia.

La replicación es particularmente útil cuando se desea incrementar el rendimiento de un sistema global de información o incrementar la fiabilidad de algún medio de almacenamiento (Nejdl et al., 2002). La misma incrementa la disponibilidad de la BD, ya que las copias de los datos almacenados en el sitio de un fallo, o inaccesibles por alguna razón, existen en otros sitios operacionales; también contribuyen a la confiabilidad y disponibilidad de un sistema. A pesar de las ventajas anteriores, la replicación perturba el funcionamiento de un sistema debido al costo de mantener la consistencia entre copias.

El diseñador de aplicaciones de BDD debe determinar cómo distribuir los datos y programas en diferentes sitios para obtener el desempeño deseado, la confiabilidad y la disponibilidad. Así, el diseño de la distribución se convierte en un problema que requiere su propia teoría, metodología de diseño, y herramientas de apoyo. Los dos principios básicos de las BDD (Őzsu y Valduriez, 1999; Baião et al., 2003) son reducir el intercambio de datos entre los sitios y eliminar datos irrelevantes en la ejecución de solicitudes. El diseño de la asignación o ubicación de los fragmentos hasta el nivel físico se ocupa de los aspectos físicos de la ubicación y réplica de fragmentos en sitios. Como se expresó anteriormente, la replicación incrementa la disponibilidad y la eficiencia de las solicitudes de solo lectura al poder ser ejecutadas en paralelo, aunque en la ejecución de solicitudes de actualización es complicado mantener la consistencia. Las BDD representan mejor la estructura descentralizada de muchas organizaciones en la actualidad, donde se requiere distribuir los datos sobre diferentes sitios de una red de computadoras y ejecutar aplicaciones que requieran el acceso a datos ubicados en más de un sitio. Así, cada unidad dentro de la organización necesita mantener los datos que son relevantes para sus operaciones. En tal sentido, muchos autores recalcan el principio de que un sistema distribuido muestra la estructura de la BD como un espejo de la estructura de la empresa con lo cual se incrementa la localidad de referencia y se reduce drásticamente el tráfico en la red (Date, 2000).

El modelo lógico de las BDD fue formulado inicialmente por Ceri *et al.* en 1983 (Ceri et al., 1983), actualmente considerado como un modelo clásico consolidado, incluye el problema de cómo distribuir los datos óptimamente en los sitios de una red. Este problema tiene como principio clave alcanzar máxima localidad de los datos, ubicándolos tan cerca como sea posible de las aplicaciones que los utilizan, lo que permite reducir el tráfico de comunicaciones en la red.

El diseño de la fragmentación precede al proceso de asignación de fragmentos a sitios de procesamiento reflejando el criterio de mantener los datos locales en el sitio donde frecuentemente son accedidos por las aplicaciones, es por ello que los fragmentos constituyen una unidad apropiada de asignación. Si las aplicaciones que manipulan un esquema están ubicadas en sitios diferentes, se pueden seguir dos alternativas con todo el esquema como unidad de distribución: replicado o no replicado (Johansson et al., 2000; Gançarski et al., 2002a; Bhalla y Hasegawa, 2005).

Muchos de los problemas relacionados con la fragmentación y asignación en el diseño de BDD tienen una modelación matemática compleja y son considerados de la clase NP-Completo (Őzsu y Valduriez, 1999; Baião et al., 2003; Lee y Baik, 2004; Pérez et al., 2005), donde no hay garantía de encontrar una solución óptima con algoritmos determinísticos en un tiempo polinomial. Así, su complejidad es tal que cualquier algoritmo que resuelve óptimamente cada uno de sus casos requiere un esfuerzo computacional que crece exponencialmente en función del tamaño del problema; en dependencia de la cantidad de fragmentos y de sitios. Intentando dar solución a estos problemas se han presentado propuestas (Ceri y Pernici, 1985; Ceri et al., 1987; Mei y Sheng, 1992; Navathe et al., 1995; Tamhankar y Ram, 1998; Savonnet et al., 1999; Baião et al., 2004; Pérez et al., 2005) que sugieren el uso de heurísticas para algunos de ellos. Usualmente hay dos enfoques alternativos para el diseño de una BDD, uno es el descendente y el otro es el ascendente. La estrategia descendente es adecuada para el desarrollo inicial de un sistema de BDD sin tener restricciones de otros sistemas ya instalados y que deban ser integrados de alguna manera al sistema distribuido como corresponde a la estrategia ascendente. Bajo el enfoque descendente, en una primera fase los esquemas globales se dividen en subconjuntos llamados fragmentos; y en una segunda fase los fragmentos son ubicados en los diferentes sitios.

1.2 Diseño de la distribución

El diseño de distribución consta de dos pasos: el diseño lógico, seguido del diseño físico. Durante el diseño lógico se distribuyen los fragmentos en los sitios. Según Date (Date, 2000), es posible tratar las relaciones como una unidades de distribución, aunque también hay esfuerzos dirigidos a considerar los objetos como unidades de distribución (Malinowski y Chakravarthy, 1997; Baião et al., 1998; Ezeife y Barker, 1998; Bellatreche et al., 2000; Cruz et al., 2002; Fung et al., 2002; Schewe, 2002; Ezeife y Dey, 2003; Fan y Lynch, 2003b; Fan y Lynch, 2003a; Fung et al., 2003; Ma, 2003; Pérez et al., 2003a; Pérez et al., 2003b; Pérez et al., 2003c; Baião et al., 2004; Hababeh et al., 2004; Ma y Schewe, 2004; Pérez et al., 2004b; Ma et al., 2005; Lin y Veeravalli, 2006). En la práctica, el problema de distribución de objetos se ha abordado insuficientemente.

Sobre el problema de distribución se han publicado numerosos trabajos, los primeros se enfocaron al problema de ubicación de archivos FAP (del inglés File Allocation Problem). El objetivo planteado para este problema era la optimización de la ubicación de archivos a sitios. Dowdy y Foster (Dowdy y Foster, 1982) contemplan la revisión de esos trabajos definiendo la optimalidad con respecto a dos medidas: costo mínimo y eficiencia máxima. Hoffer (Hoffer, 1976) desarrolló un modelo no lineal binario que minimiza una combinación lineal de los costos de almacenamiento, consultas de actualización y de solo lectura con restricciones de capacidad. Otro de los trabajos sobre el FAP aplicado a la ubicación de archivos en Internet con autocertificación para un sistema seguro de distribución de contenido es el de Fu *et al.* (Fu et al., 2002). Otras publicaciones recientes sobre modificaciones a los algoritmos y estructuras aplicados al mismo problema pueden encontrase en (Chang et al., 2001; Fornaciari et al., 2001; Tsai et al., 2002; Awerbuch et al., 2003; Mei et al., 2003; Sun et al., 2004).

Los modelos obtenidos del problema de ubicación de archivos FAP no son aplicables directamente en las BDD, pues son demasiado simples y no abarcan otros aspectos de la distribución de datos (Apers, 1988). Tomando en cuenta el problema anterior, se genera una nueva forma de solucionar el problema, mediante la ubicación de fragmentos DAP (del inglés Data Allocation Problem). Esta forma divide el diseño de la distribución en dos: fragmentación y asignación de los fragmentos a los sitios de procesamiento en una red de computadoras (Őzsu y

Valduriez, 1999). Como el diseño de la distribución de datos abarca una gran cantidad de variables y diversidad de relaciones entre estas, el problema del diseño se hace complejo (Apers, 1988; Lin y Orlowska, 1995).

Según Huang (Huang y Chen, 2001), el diseño de una BDD involucra varios factores interrelacionados que complican el proceso de diseño de BDD aun más; incluso cuando cada factor sea considerado individualmente, el problema sigue siendo intratable. Para simplificarlo se suelen tratar por separado la fragmentación y asignación de los fragmentos, ya que existe una relativa independencia entre ellos (Saccà y Wiederhold, 1985; Őzsu y Valduriez, 1999). Otras realizan el proceso de fragmentación y ubicación simultáneamente usando diferentes enfoques (Tamhankar y Ram, 1998; Pérez, 1999; Ma et al., 2005; Pérez et al., 2005; Ma et al., 2006).

En realidad, el proceso de distribución es lineal, donde la salida de la fragmentación es la entrada de la asignación. A primera vista, este aislamiento parece simplificar la formulación del problema al reducir el espacio de decisión. Si se analiza con más profundidad, se aumenta la complejidad de los modelos de asignación. Las entradas de ambas etapas son semejantes, difiriendo apenas en que la entrada de la fragmentación son los esquemas globales y las de la asignación son los fragmentos obtenidos en la etapa previa. Las dos fases necesitan de información sobre las aplicaciones, pero al realizarse por separado, en la segunda fase se tienen que incluir, nuevamente, mucha de la especificación detallada en la primera fase. Una combinación adecuada de estas fases a través de metodologías que usen determinadas heurísticas y se apoyen en herramientas que ayuden a los diseñadores en lugar de intentar sustituirlos es una forma apropiada de enfrentar el problema de diseño (Őzsu y Valduriez, 1999).

1.3 Problema de asignación

La segunda parte del diseño de distribución, la asignación o ubicación, típicamente es considerada independiente de la fragmentación. El problema de la asignación de datos, según (Ceri et al., 1982) se define de la siguiente manera: Se asume que hay un conjunto de fragmentos $F = \{f_1, f_2, ..., f_n\}$ y una red constituida por los sitios $S = \{s_1, s_2, ..., s_m\}$ en los cuales se están ejecutando un conjunto de

aplicaciones $Q = \{q_1, q_2, ..., q_q\}$. El problema de ubicación consiste en encontrar la distribución "óptima" de F sobre S.

La optimalidad puede ser definida con respecto a costos o desempeño (Őzsu y Valduriez, 1999). Si se define en función de los costos, se intenta encontrar un esquema de ubicación que minimice una función de costo, en la cual se combinan los costos de almacenamiento, consultas, actualizaciones y el de comunicación de datos. Si se define en función del desempeño, se debe mantener alguna métrica de desempeño, por ejemplo: minimizar el tiempo de respuesta.

La aproximación básica e intuitiva para el problema de asignación de datos consiste en generar exhaustivamente todas las posibles asignaciones de datos usando el conjunto de fragmentos, y calcular el costo de cada posible asignación para seleccionar la asignación que produzca el costo mínimo. Esto garantiza que la asignación escogida es el óptimo global, pero no es factible en la práctica, debido a que el número de posibles asignaciones es sumamente grande (exponencial respecto al número de sitios y fragmentos). En general, la ubicación de los fragmentos tiene como objetivo principal minimizar el número de accesos remotos ejecutados por las aplicaciones. Existen dos alternativas básicas para realizar la ubicación de datos: no replicada y replicada.

1.3.1 Modelo de asignación

A continuación se plantea una forma general del modelo de asignación de fragmentos cuyo objetivo es minimizar el costo total de procesamiento y almacenamiento, sujeto a ciertas restricciones (Őzsu y Valduriez, 1991).

min. (Costo Total)

Sujeto a:

Restricciones de tiempo de respuesta.

Restricciones de almacenamiento.

Restricciones de procesamiento.

La variable de decisión es x_{jk} y se define como:

$$x_{jk} = \begin{cases} 1, & \text{fragmento } f_j \text{ almacenado en el sitio } s_k \\ 0, & \text{e.o.c} \end{cases}$$

1.3.1.1 Información requerida

Para el proceso de asignación se necesitan datos cuantitativos sobre la base de datos, las aplicaciones que se ejecutan sobre esta, la red de comunicación y sobre cada sitio de la red. A continuación se detalla cada aspecto.

Información sobre la base de datos

- $sel_i(f_j)$: Se define como selectividad de un fragmento f_j con respecto a una solicitud q_i, como el número de tuplas de f_j que tienen que ser accedidas para procesar q_i.
- $size(f_j) = card(f_j) \cdot length(f_j)$. El tamaño de un fragmento f_j dado por la cantidad de tuplas del fragmento y su longitud en bytes.

Información sobre las aplicaciones

La mayoría de la información relacionada con las aplicaciones ya ha sido recopilada durante el proceso de fragmentación, pero para el modelo de ubicación se requiere alguna información adicional:

- RR_{ij}: Número de accesos de lectura que una consulta q_i realiza sobre un fragmento f_j durante su ejecución.
- UR_{ij}: De forma similar representa el número de accesos de actualización.

Se necesitan también dos matrices: UM y RM con elementos u_{ij} y r_{ij} respectivamente, especificados como sigue:

$$u_{ij} = \begin{cases} 1, & q_i \text{ actualiza el fragmento } f_j \\ 0, & \text{e.o.c} \end{cases} \qquad r_{ij} = \begin{cases} 1, & q_i \text{ lee el fragmento } f_j \\ 0, & \text{e.o.c} \end{cases}$$

- $o(i)$: Sitio donde se origina la consulta q_i.
- $MaxTime(i)$: Tiempo de respuesta máximo admisible para cada aplicación.

Información sobre los sitios

Para cada sitio se necesita conocer su capacidad de almacenamiento y su velocidad de procesamiento. Para captar estos valores se pueden elaborar funciones o simplemente pueden ser estimados.

- USC_k: Costo unitario de almacenar un bloque de datos en el sitio s_k.

- SPC_k: Costo de procesar una unidad de trabajo en el sitio s_k. La unidad de trabajo pudiera ser igual a los valores de RR y UR.

Información sobre la red

En este modelo se asume la existencia de una red simple, donde el costo de comunicación pueda ser definido en términos de un frame de datos.

- $g_{kk'}$: Costo de comunicación por frame entre los sitios s_k y $s_{k'}$.
- fsize: Tamaño (en bytes) de un frame.

En un ambiente distribuido los fragmentos de datos son ubicados en sitios diferentes, de manera que en el procesamiento de las consultas se combinan datos de diferentes sitios para obtener la respuesta final. Para hacer esto de una forma económica se requiere de una buena función de costo que justifique la fragmentación de los datos, contemple las frecuencias de uso de todas las consultas y actualizaciones, y los sitios a los que los resultados deben ser enviados. En la mayoría de los sistemas de la vida real, toda la información requerida para la solución del modelo es prácticamente imposible de recopilar, ya que el sistema no tiene un comportamiento estático en el tiempo y la cantidad de datos que se requieren para el modelo podría ser abrumadora.

1.3.1.2 Función de costo

La función de costo total tiene dos componentes: el procesamiento de solicitudes y el almacenamiento. Se expresa de la siguiente forma:

$$TOC = \sum_{\forall q_i \in Q} QPC_i + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} STC_{jk}$$

donde QPC_i es el costo de procesamiento de las solicitudes de la aplicación q_i, y STC_{jk} es el costo de almacenamiento del fragmento f_j en el sitio s_k.

El costo de almacenamiento está dado por:

$$STC_{jk} = USC_k \cdot size(f_j) \cdot x_{jk}$$

Las dos sumatorias calculan el costo de almacenamiento total en todos los sitios, para todos los fragmentos.

El costo de procesamiento de las solicitudes es un poco más difícil de especificar. La mayoría de los modelos para resolver el clásico problema de ubicación de ficheros en sistemas distribuidos lo separan en dos componentes: el costo de procesamiento para solo lectura, y el costo de procesamiento para la actualización.

En este caso, que se trata de un modelo para el problema de ubicación de base de datos, se especificará como el costo de procesamiento (PC) y el costo de transmisión (TC). Así, el costo de procesamiento de solicitudes (QPC) para la aplicación q_i sería:

$$QPC_i = PC_i + TC_i$$

El costo de procesamiento, contiene tres factores, el costo de acceso (AC), el costo del chequeo de restricciones de integridad (IE), y el costo del control de concurrencia (CC):

$$PC_i = AC_i + IE_i + CC_i$$

La especificación de cada uno de esos factores de costo depende de los algoritmos usados para realizar estas tareas.

La especificación detallada de AC sería:

$$AC_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} \left(u_{ij} \times UR_{ij} + r_{ij} \times RR_{ij}\right) \times x_{jk} \times SPC_k$$

Los dos primeros términos de la fórmula anterior calculan el número de accesos de solicitudes de usuarios q_i al fragmento f_j.

Nótese que $(UR_{ij} + RR_{ij})$ refleja el número total de accesos de solo lectura y de actualización. Se asume que el costo total de procesar las solicitudes es el mismo. La sumatoria calcula el número total de accesos a todos los fragmentos referenciados por q_i. La multiplicación por SPC_k da como resultado el costo del acceso al sitio. Y se usa de nuevo x_{jk} para seleccionar solo los valores de costo de los sitios donde los fragmentos han sido almacenados.

Hay una cuestión muy importante que destacar aquí. La función de costo de acceso asume que procesar una consulta incluye su descomposición en un conjunto de subconsultas, cada una de las cuales se ejecuta sobre un fragmento almacenado en un sitio, y la transmisión del resultado hacia el sitio donde fue originada la solicitud. Esta es una visión muy reducida del problema, la cual no tiene en consideración la complejidad del procesamiento de la base de datos. Por ejemplo la función de costo no tiene en cuenta el costo de realizar acoples (si fuese necesario), lo cual puede ser ejecutado de varias maneras. En un modelo que sea más real que el modelo genérico, estos aspectos no deben ser omitidos.

El costo del chequeo de restricciones de integridad puede ser especificado muchas veces como la componente de procesamiento, excepto en caso de que el costo de la unidad local de procesamiento pudiera cambiar y alterar el verdadero costo de chequeo de restricciones de integridad.

Los costos operativos de la transmisión de datos para las consultas de actualización y de solo lectura son bien diferentes. En las consultas de actualización es necesario actualizar tantos sitios como réplicas existan, mientras que en las consultas de solo lectura, es suficiente con acceder a solo una de las copias. Además, al final de una solicitud de actualización no hay transmisión de datos de regreso hacia el sitio que originó la solicitud, solo un mensaje de confirmación, mientras que en las consultas de solo lectura se puede producir una enorme transmisión de datos.

La componente de actualización de la función de transmisión es:

$$TCU_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} u_{ij} \times x_{jk} \times g_{o(i),k} + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} u_{ij} \times x_{jk} \times g_{k,o(i)}$$

El primer término representa el envío del mensaje de actualización desde el sitio $o(i)$ que origina q_i, a todas las réplicas del fragmento que requieren ser actualizadas. El segundo término es para la confirmación.

El costo de las consultas de solo lectura es:

$$TCR_i = \sum_{\forall F_j \in F} \min_{S_k \in S} (r_{ij} \times x_{jk} \times g_{o(i),k} + r_{ij} \times x_{jk} \times \frac{sel_i(F_j)}{fsize} \times length(F_j) \times g_{k,o(i)})$$

El primer término representa el costo de transmitir la consulta de solo lectura hacia los sitios que tienen copias de fragmentos que necesitan ser accedidos.

El segundo término representa el costo para la transmisión de los resultados desde esos sitios hasta el sitio que originó la solicitud.

La ecuación establece que entre todos los sitios que contienen copias de un mismo fragmento, solo el sitio que logre el mínimo costo de transmisión total es el que debe ser elegido para la ejecución de la operación.

Resumiendo, la función de costo de transmisión para la solicitud q_i puede plantearse como:

$$TC_i = TCU_i + TCR_i$$

la cual define la función de costo total.

Restricciones

La restricción de tiempo de respuesta:

$$q_i \leq \text{MaxTime}(i),\ \forall q_i \in Q$$

La restricción de almacenamiento:

$$\sum_{\forall F_j \in F} \text{STC}_{jk} \leq \text{capacidad de almacenamiento en el sitio } s_k, \forall s_k \in S$$

La restricción de procesamiento:

$$\sum_{\forall q_i \in Q} \text{procesar la ejecución de } q_i \text{ en el sitio } s_k \leq \text{capacidad de procesamiento de } s_k, \forall s_k \in S$$

1.3.2 Soluciones ofrecidas al problema de asignación

Sobre el problema de asignación en BD se han publicado numerosos trabajos que intentan reducir la complejidad del problema (Apers, 1988; Raghuram et al., 1989; Sheng, 1989; Chiu y Raghavendra, 1990; Ram y Marsten, 1991; Brahmadathan y Ramarao, 1992; Lin et al., 1993; Chaturvedi y Roan, 1994; Lin y Orlowska, 1995; March y Rho, 1995; Shepherd et al., 1995; Wolfson y Jajodia, 1995; Karlapalem y Pun, 1997; Lim y Ng, 1997; Park y Baik, 1997; Bellatreche et al., 1998; Daudpota, 1998; Tamhankar y Ram, 1998; Pérez et al., 2000; Chang et al., 2001; Fornaciari et al., 2001; Huang y Chen, 2001; Tsai et al., 2002; Awerbuch et al., 2003; Hababeh et al., 2003; Mei et al., 2003; Pérez et al., 2003a; Pérez et al., 2003b; Hababeh et al., 2004; Iida et al., 2004; Lee y Baik, 2004; Pérez et al., 2004a; Pérez et al., 2004b; Sun et al., 2004; Ma et al., 2005; Pérez et al., 2005; Roosta, 2005; Gu et al., 2006; Ma et al., 2006; Savsar y Al-Anzi, 2006). Una estrategia ha sido asumir que todos los fragmentos posibles han sido determinados junto con sus costos asociados y sus beneficios en términos del procesamiento de consultas. Así, el problema es modelado como la elección de la fragmentación y asignación óptimas para cada relación. Otra simplificación frecuentemente empleada es ignorar inicialmente la replicación de datos y encontrar una solución óptima para el caso no replicado. La replicación se incorpora en un segundo paso aplicando un algoritmo que se inicia a partir de la solución no replicada y trata de mejorarla iterativamente.

Se han realizado trabajos basados en el desempeño de la BD, tales como análisis de redundancia de archivos (Muro et al., 1985) y asignación de BD (Yoshida et al., 1985). Posteriormente el problema de asignación fue formulado como un problema

de programación en enteros binaria (Chiu y Raghavendra, 1990) e incluso se ha integrado con mecanismos específicos de control de concurrencia (Ram y Marsten, 1991; Tamhankar y Ram, 1998). Se han usado algunos enfoques analíticos para redes LAN basadas en Ethernet para determinar el tiempo de respuesta de las transacciones en el problema de asignación en BD (Sheng, 1989).

Como puede entenderse, el problema de la asignación de los datos no es un problema sencillo de resolver algorítmicamente, por lo que es necesario buscar métodos de solución heurísticos que permitan obtener una buena solución. Para aplicaciones que hagan solicitudes de lectura se pueden ubicar los fragmentos en los sitios donde se originan estas, o puede ser un poco más complicado obtenerlos de sitios remotos. Una solicitud de escritura puede ser más complicada, pues se debe ejecutar la propagación de escritura a todos los sitios donde haya copias de los fragmentos involucrados, si se tienen, para mantener la consistencia. También debe considerarse la frecuencia de cada solicitud emitida en los sitios, ya que el comportamiento de las aplicaciones puede afectar la ubicación óptima de los fragmentos. Así, las fórmulas de costos debieran obtenerse como resultado de la información sobre las aplicaciones para minimizar el costo de procesamiento de las mismas.

Chang (Chang y Liu, 1982) desarrolló una teoría de asignación de fragmentos y diseñó un algoritmo de flujo de red para resolver el problema de ubicación en BDD. Como se conoce que este problema es de la clase NP-Completo (Őzsu y Valduriez, 1999; Baião et al., 2003; Pérez et al., 2004b; Pérez et al., 2005), varios autores han propuesto soluciones que reducen la complejidad mediante algoritmos heurísticos como los presentados en (Ceri et al., 1982) para dar solución al problema de la mochila y en (Fisher y Hochbaum, 1980) usando las técnicas de ramificación y acotación.

Ceri *et al.* (Ceri et al., 1983) hacen uso del modelo de programación entera con el fin de minimizar el costo de procesamiento de transacciones y ubicar los objetos sin replicarlos en los sitios. Además, Ceri y Pernici (Ceri y Pernici, 1985) propusieron un método simple que primeramente ignora la replicación buscando una solución óptima no replicada, y luego aplica un algoritmo greedy que trata la replicación, intentando mejorar la solución factible inicial. Los problemas resueltos eran de complejidad sencilla contando de 60 a 100 variables.

Cornell y Yu (Cornell y Yu, 1989) modelan el problema como un problema de programación lineal, proponiendo dos fases: la primera fase descompone cada consulta en una secuencia de operaciones del Álgebra Relacional conforme a la estrategia del optimizador de consultas, mientras que en la fase dos se usa cada operación del Álgebra Relacional como entrada del algoritmo que optimiza la ubicación de relaciones y el sitio donde serán efectuados las operaciones de acople. De una manera alternativa emplea también una técnica heurística para minimizar el tiempo de respuesta promedio.

El modelo propuesto por Raghuram *et al.* (Raghuram et al., 1989) es muy abarcador, involucra pocas restricciones, tales como la capacidad de cómputo de cada sitio y el tiempo máximo de respuesta deseado de cada solicitud, pero no considera la replicación.

Chaturvedi *et al.* (Chaturvedi y Roan, 1994) reportan un método de fragmentación basado en machine learning que adquiere conocimiento acerca de los patrones de acceso a datos para cada nodo y demuestran su eficacia mediante simulación. Lin *et al.* (Lin et al., 1993) presentan algoritmos para la asignación de datos para obtener costos totales de comunicación mínimos. Después, Lin y Orlowska (Lin y Orlowska, 1995) reportan un modelo de programación lineal con el objetivo de minimizar los costos totales de comunicación a consecuencia del procesamiento de las consultas.

Por otra parte, March y Rho (March y Rho, 1995) desarrollan un modelo matemático y un algoritmo genético que ubica datos y operaciones a los nodos. Primero determina la unidad de datos a ubicar (basado en el trabajo de Apers (Apers, 1988) y cada aplicación es descompuesta en operaciones del Álgebra Relacional. Como resultado obtiene los fragmentos y operaciones, y se ubican en sitios mediante un modelo matemático en el cual su función objetivo es minimizar los costos de operación del sistema tomando en cuenta las restricciones de capacidad de cada nodo. Park y Baik (Park y Baik, 1997) ofrecen como alternativa un modelo probabilístico para la ejecución de transacciones y un algoritmo genético que minimiza el costo de procesamiento, aplicado a sistemas que requieran alta disponibilidad.

En las investigaciones realizadas por Wolfson *et al.* (Wolfson y Jajodia, 1992b; Wolfson y Jajodia, 1992a; Wolfson y Jajodia, 1995; Wolfson et al., 1997) se

presentan las estrategias de asignación de datos considerando que las condiciones cambian dinámicamente. Daudpota *et al.* (Daudpota, 1998) crearon un modelo formal para la asignación de datos y obtuvieron un algoritmo para fragmentar relaciones y ubicar los fragmentos resultantes. Este trabajo no es apropiado para aplicaciones distribuidas para redes con diferente conectividad. Bellatreche *et al.* (Bellatreche et al., 1998) desarrollaron un modelo para calcular costos totales de transferencia de datos y su algoritmo de ubicación generaba soluciones al problema cercanas a la óptima.

Huang *et al.* (Huang y Chen, 2001) proponen un algoritmo heurístico que refleja el comportamiento de las transacciones en BDD. Su modelo determina la cantidad de réplicas de cada fragmento y luego busca una ubicación cercana a la óptima de todos los fragmentos, replicados o no, de modo que el costo total de comunicación sea mínimo. Este trabajo considera independientes a todos los fragmentos accedidos por una aplicación, lo cual es un enfoque simplista no realista. Además, ignora información sobre los sitios como son la capacidad de almacenamiento y procesamiento, los costos de procesamiento de CPU y entrada/salida.

Chang (Chang, 2004) presenta métodos de solución heurísticos que permiten obtener una buena solución al problema de distribución. En este sentido se describe el método de dos fases para el problema de asignación óptima de fragmentos sobre una red en un SBDD. En la primera fase se realiza la agrupación de fragmentos, en la que se forman grupos de fragmentos que tienden a ser accedidos por una misma consulta en el SBDD basándose en un conjunto de consultas dadas y sus frecuencias de acceso. En la segunda fase se asignan clusters a los sitios en la red de modo que el costo de procesamiento de todas las consultas (combinación de costo de transmisión y costo de procesamiento determinado por un optimizador de consultas distribuidas) sea minimizado.

Hababeh *et al.* (Hababeh et al., 2004) proponen un método para agrupar los sitios de la BDD creando cluster de sitios de acuerdo a los costos de comunicación para determinar la asignación de fragmentos a los clusters, en lugar de asignarlos por sitio. Este trabajo optimiza los costos de las funciones de ubicación para reducir los tiempos de procesamiento de las solicitudes y determina los fragmentos que serán asignados a los sitios de la BDD. Este trabajo es aplicado a BDD orientadas a objeto sin perder generalidad.

Roosta (Roosta, 2005) plantea un modelo para ubicar tablas en un SBDD. Este modelo considera el tamaño de las tablas, las frecuencias de acceso y actualizaciones, los costos de acceso a memoria y los de transmisión, así como las capacidades de memoria de cada sitio y los tiempos esperados de acceso permitidos. Como la mayoría de los autores, aquí también la función objetivo está expresada para lograr la optimalidad en los costos totales de operación. El trabajo de Roosta se diferencia de los demás en que el modelo está formulado como un problema de programación en enteros binario no lineal, aunque puede ser convertido a un modelo de programación lineal binario cuya solución es de igual complejidad al planteado en (Őzsu y Valduriez, 1999).

Se puede observar que para dar solución al problema de asignación se requiere de una gran cantidad de información de diversa naturaleza y los métodos que lo resuelven tienen una alta complejidad, por lo que se requiere de heurísticas que permitan obtener buenas soluciones en un tiempo aceptable.

1.3.2.1 Asignación no redundante

El trabajo realizado por Ceri en (Ceri y Pernici, 1985) plantea el problema de ubicación de fragmentos con la metodología DATAID-D donde se ubican los fragmentos en los sitios que presenten una frecuencia de acceso mayor, de esta forma obtiene una solución no replicada. Cuenta con un algoritmo greedy para obtener la ubicación replicada de los fragmentos de acuerdo a la solución no replicada.

Saccà y Wiederhold (Saccà y Wiederhold, 1985) realizan la fragmentación en un primer paso y luego pasan a la ubicación de los fragmentos obtenidos a través de la implementación del algoritmo heurístico GFF (del inglés Greedy First Fit). Inicialmente este algoritmo selecciona los fragmentos mediante el algoritmo greedy, lo combina con el algoritmo first fit bin packing para ubicar los fragmentos que fueron seleccionados en los procesadores. Como resultado se obtiene un esquema no redundante de ubicación de fragmentos. Lee *et al.* (Lee et al., 2000), proponen una metodología heurística para determinar la asignación simultánea de archivos y carga de trabajo en una red LAN. Este método minimiza el tiempo de respuesta de las transacciones pero solo serán enrutadas a un mismo servidor aquellas transacciones con iguales propiedades, lo cual no garantiza una minimización en los costos de comunicación. Asumen una asignación no

redundante lo cual disminuye la fiabilidad de un sistema, y el impacto de almacenar copias de fragmentos en los sitios de la red LAN no es muy significativo.

1.3.2.2 Asignación redundante

La replicación de datos es un tema cada vez más importante para la comunidad de investigadores en BD y sistemas distribuidos. La replicación permite compartir información entre BD y plataformas heterogéneas, y modificar y reconciliar esa información; también garantiza la disponibilidad de los datos correctos cuándo y dónde se precisen. En la presente investigación se aborda el diseño de BDD mediante mecanismos que consideran la replicación. A menudo se necesita replicación cuando se desea incrementar el rendimiento de un sistema global de información o incrementar la fiabilidad de algún medio de almacenamiento (Nejdl et al., 2002). Esta es más compleja en el caso de las BD que en archivos simples, ya que se requiere que se tenga en cuenta la naturaleza de los datos, sus interrelaciones, el control de concurrencia (Fan y Lynch, 2003b), que los agentes de replicación registren las transacciones, supervisen las actualizaciones, garantizando coherencia y consistencia (Hotek, 2002). Los algoritmos que garantizan una fuerte consistencia se basan en protocolos de bloqueos y confirmación (commit) (Bernstein et al., 1987). Respecto al soporte para replicación de información, o no se ofrece o se hace a través del protocolo sincrónico (eager) (Kemme y Alonso, 2000; Irún-Briz et al., 2003; Patiño-Martínez et al., 2005) ROWA, que brinda una fuerte consistencia. Este protocolo sincrónico no es apropiado porque viola la autonomía de los sistemas locales, puede conducir a bloqueos (deadlock) en caso de fallos en algún sitio, ya que normalmente la atomicidad de las transacciones distribuidas se basan en el protocolo de compromiso en dos fases (2PC) (Őzsu y Valduriez, 1999), y es poco escalable (Pacitti et al., 2005). Por tanto, la replicación sincrónica no es un enfoque viable en la mayoría de los ambientes de procesamiento de datos actuales (Akal et al., 2004). Por otra parte, la replicación asíncrona (lazy), tratada en ocasiones como optimista o preventiva (Breitbart y Korth, 1997; Holliday et al., 1999a; Pacitti et al., 1999b; Pacitti et al., 1999a; Pacitti y Simon, 2000; Pacitti et al., 2001; Irún-Briz et al., 2003; Akal et al., 2004; Daudjee y Salem, 2004; Elmeleegy et al., 2004; Irún-Briz et al., 2004; Akal et al., 2005; Daudjee y Salem, 2006), sacrifica consistencia en aras de lograr mejor desempeño (Pacitti et al., 2003; Coulon et al., 2005c; Pacitti et al., 2005).

Fan y Lynch (Fan y Lynch, 2003b; Fan y Lynch, 2003a) presentan un algoritmo de replicación de datos distribuidos ajustado especialmente a datos de lectura y escritura a gran escala, o sea, sistemas de archivos. Este algoritmo asegura la consistencia de los datos y un bajo costo en cuanto al tiempo requerido para su recuperación, pero no tiene en cuenta la naturaleza de los mismos en una BD, ni las interrelaciones entre ellos.

Huang y Chen (Huang y Chen, 2001) proponen un modelo global que analiza el funcionamiento de aplicaciones en BDD. Analizan cómo debe ser fragmentada una relación global, cuántas copias de un fragmento deben ser replicadas, cómo deben ser asignados a los sitios de la red de comunicaciones, y cuál es la información requerida para la fragmentación y asignación, evidenciándose que estos factores complican el proceso de diseño de BDD. Incluso cuando cada factor sea considerado individualmente, el problema sigue siendo intratable. Se desarrollan dos algoritmos heurísticos basados en el modelo y en información sobre las aplicaciones para encontrar un esquema de ubicación cercano al óptimo, de forma que se minimice el costo total de comunicaciones tanto como sea posible.

Se han propuesto múltiples soluciones de replicación basadas en cluster que proveen escalabilidad y tolerancia a fallos. Cacheda *et al.* (Cacheda et al., 2004) muestran que los cuellos de botella en una arquitectura distribuida pura son los agentes (brokers) que reciben numerosas solicitudes, mientras que en un sistema replicado es la red la que los provoca debido al gran número de servidores de solicitudes y al continuo intercambio de datos con los agentes. En (Böhm et al., 2000; Patiño-Martínez et al., 2000; Gançarski et al., 2002a; Gançarski et al., 2002b; Jiménez-Peris et al., 2002; Jiménez-Peris et al., 2003; Le Pape et al., 2003; Pacitti et al., 2003; Cecchet et al., 2004; Grabs et al., 2004; Le Pape et al., 2004; Coulon et al., 2005c; Coulon et al., 2005a; Coulon et al., 2005b; Pacitti et al., 2005; Valduriez et al., 2005) se trata el tema de clusters de BD que no son más que clusters de sitios o sitios autónomos (con discos y procesadores propios que ejecutan su propio SGBD). Esto tiene la ventaja de preservar la autonomía de las BD y evitar costosas migraciones de datos a BD paralelas (Gançarski et al., 2002a; Goel et al., 2003b; Goel et al., 2003a).

El trabajo de Lin *et al.* (Lin et al., 2005) presenta un esquema de replicación basado en capa intermedia (middleware) que provee el conocido nivel de

aislamiento instantáneo (snapshot isolation) (Elnikety et al., 2005; Lin et al., 2005; Wu y Kemme, 2005; Daudjee y Salem, 2006) al mismo nivel de granularidad de tupla que los motores como el PostgreSQL y ORACLE sin tener que declarar por adelantado ninguna propiedad de las transacciones. Tanto las transacciones de solo lectura como las de actualización pueden ser ejecutadas en cualquier réplica garantizando la consistencia de los datos en cualquier momento. En el trabajo de Lin et al. (Lin et al., 2005) se usan las réplicas para tener copias redundantes como estrategia proactiva y lograr mayor disponibilidad ante fallos.

Muchos autores han vuelto a analizar el tema de replicación y han propuesto soluciones para eliminar los problemas asociados con el control de réplica (Berenson et al., 1995; Chundi et al., 1996; Birman, 1997; Breitbart y Korth, 1997; Anderson et al., 1998; Pedone et al., 1998; Breitbart y Korth, 1999; Holliday et al., 1999b; Holliday et al., 1999a; Pacitti et al., 1999b; Pacitti et al., 1999a; Böhm et al., 2000; Kemme y Alonso, 2000; Pacitti y Simon, 2000; Patiño-Martínez et al., 2000; Pedone et al., 2000; Pacitti et al., 2001; Jiménez-Peris et al., 2002; Röhm et al., 2002; Amza et al., 2003; Holliday et al., 2003; Pacitti et al., 2003; Cecchet et al., 2004; Daudjee y Salem, 2004; Grabs et al., 2004; Lin et al., 2005; Patiño-Martínez et al., 2005; Daudjee y Salem, 2006). Por su parte, Lin y Veeravalli (Lin y Veeravalli, 2006) presentan un algoritmo de replicación de datos para BDD de tiempo real que mantiene la consistencia y garantiza la satisfacción de las restricciones impuestas por las solicitudes que arriban a un sistema. Se presenta un modelo matemático cuyo objetivo es minimizar el costo total de servicio sujeto a las restricciones de atender todas las solicitudes que lleguen en el tiempo esperado. Los autores diseñan un algoritmo de replicación dinámica de datos que se adapta a los patrones aleatorios de acceso de las solicitudes en tiempo real.

Lee y Baik (Lee y Baik, 2004) proponen un algoritmo heurístico que resuelve el problema de asignación replicada de datos así como de transacciones en BDD, y hacen uso de tablas para almacenar balances de carga y tráfico en la red, en cada sitio y así dar solución al problema planteado.

Como trabajo previo del autor de esta tesis (Rosa, 2006) se desarrollaron dos métodos que fueron incorporados a una herramienta en forma de asistente denominada ALLOCATOR. Estos métodos tienen forma de agentes y usan la información requerida para el modelo de asignación lógica, la cual ha sido

colocada en un catálogo en etapas previas del diseño de la BDD. Uno de los agentes se basa en un algoritmo genético generacional (Cheng et al., 2002; Pérez et al., 2004a; Pérez et al., 2004b; Du et al., 2006). El otro agente se basa en el método Q-Learning (Cichosz, 1995; Sutton y Barto, 1998; Mariano y Morales, 2001; Pettinger y Everson, 2002; Abe et al., 2003; Paul y Iba, 2003; Choi et al., 2004; Iida et al., 2004; Morales y Sammut, 2004; O et al., 2004). En los métodos desarrollados se observa que a medida que aumenta el tamaño de la instancia del problema las soluciones obtenidas se alejan de la solución óptima y evidencian problemas de convergencia. La herramienta ALLOCATOR brinda al diseñador la facilidad de elegir el método mediante el cual pretende dar solución al problema de asignación, se lleva a cabo el proceso de optimización y finalmente se muestra la asignación lógica alcanzada o un mensaje en caso de que no se haya encontrado una solución que cumpla con las restricciones del modelo de asignación. Como se puede observar, solo se llega hasta el nivel lógico, sin tener en cuenta el nivel físico.

1.3.2.3 Soluciones que combinan la fragmentación y la asignación

Se reportan algunas investigaciones donde se tratan las tareas de fragmentación y ubicación conjuntamente, aunque las metodologías que lo soportan tienden a ser muy complejas, predominantemente teóricas, y de alcance limitado desde el punto de vista práctico (Tamhankar y Ram, 1998). Algunas de estas son referidas en (Pérez y Pazos, 1997; Vélez, 1997; Pérez, 1999; Vélez, 2000; Huang y Chen, 2001) sobre el modelo de Fragmentación, Ubicación y Reubicación Dinámica de Datos FURD (Pérez y Pazos, 1997), que es un modelo matemático de programación lineal entera binaria (no replicada) que se encarga de combinar los procesos de fragmentación vertical y ubicación de atributos a sitios. Para resolver este modelo se utilizó el método de solución exacto de Ramas y Cotas, para los casos de prueba pequeños. Debido a la complejidad del problema modelado, posteriormente se trabajó con los métodos aproximados de Recocido Simulado y Aceptación por Umbral, para dar solución a los casos de prueba grandes.

Otros trabajos sobre integración de soluciones al modelo FURD han sido publicados en (Pérez, 1999; Pérez et al., 2000; Vélez, 2000; Pérez et al., 2002; Pérez et al., 2004c; Pérez et al., 2005) donde reportan métodos aceptables de solución heurística aplicados al problema de distribución en BDD (Pérez, 1999), el

cual ha sido resuelto parcialmente con diferentes modelos y técnicas (Ceri et al., 1983; Ceri y Pernici, 1985; Wolfson y Jajodia, 1992a; Karlapalem et al., 1994; March y Rho, 1995; Őzsu y Valduriez, 1999; Savonnet et al., 1999; Savonnet y Terrasse, 2001; Baião et al., 2004; Ma et al., 2006) mediante dos pasos: primero la obtención de los fragmentos y luego la ubicación de los mismos en los sitios. Este problema es NP-Hard (problema de optimización cuyo problema de decisión asociado es NP-Completo), integra los dos pasos en uno. La solución para este problema es muy compleja, su espacio de solución es extremadamente grande, aun para problemas pequeños. Esto justifica que se usen métodos heurísticos tales como el presentado en este artículo, que es una variante del algoritmo de Recocido Simulado que a su vez implementa el algoritmo de Metropolis para simular el recocido de metales. Se refiere que es posible resolver problemas reales de gran magnitud y se observa la potencialidad del algoritmo para resolver estos problemas en un tiempo razonable. Se debe observar que el problema al que se da solución es de rediseño de BDD, por lo que no es de interés en este trabajo.

Tamhankar y Ram (Tamhankar y Ram, 1998) proponen una metodología integrada para la fragmentación, ubicación y replicación en un solo paso de distribución de BD relacionales, en lugar de tratar independientemente cada una de estas tareas, y muestran la aplicación la combinación mediante un caso de estudio, mostrando resultados positivos. Una de las fortalezas de esta metodología es que en la primera distribución, donde la fragmentación y ubicación se realizan simultáneamente, no se generan fragmentos innecesarios y se reflejan claramente las interdependencias existentes entre las decisiones de fragmentación y localización. Comparado con los enfoques más comunes, los cuales son muy complejos computacionalmente, este enfoque es más simple aunque no se sacrifica la complejidad del ambiente de BDD. La aplicación de esta metodología presupone el cumplimiento de un conjunto de restricciones relacionadas con las capacidades de los motores de BD que no son satisfechas totalmente por los motores existentes, la redes que interconectan los sitios no siempre tienen un solo tipo de conectividad (LAN o WAN) ni tienen la alta fidelidad esperada, y de la metodología en sí que no incluye la ubicación de los programas, lo cual no es difícil de satisfacer, y no considera la sobrecarga impuesta por los Sistemas Operativos y los motores de BD respecto a capacidades de almacenamiento, creación de

índices, es decir, solo tiene en cuenta el almacenamiento básico de los datos. También incluye procesos de decisión respecto a volúmenes de datos, costos, etc. que deben ser estimados, los cuales pueden ser valorados subjetivamente, y excluye procesos de ajuste o desnormalización típico en las BDC que también son aplicables a las BDD.

Hababeh *et al.* (Hababeh et al., 2003) proponen una estrategia integrada para fragmentar y ubicar en el diseño de BDD que usa un método de clusterización. Un trabajo más avanzado de estos autores sobre este tema se puede apreciar en (Hababeh et al., 2004).

Otro trabajo más reciente es el de Ma *et al.* (Ma et al., 2006), donde se realiza el proceso de fragmentación y ubicación simultáneamente usando un enfoque heurístico mediante la aplicación de un modelo basado en costos. La fragmentación se realiza siguiendo los mismos pasos expresados en (Őzsu y Valduriez, 1999) y las decisiones de fragmentación se basan en la eficiencia de las solicitudes más frecuentes. Por esto se hace un análisis de las aplicaciones para obtener una distribución adecuada de los datos. El enfoque del trabajo de Ma *et al.* (Ma et al., 2006) no es apropiado para el problema enfocado en esta investigación puesto que es aplicado a BD no relacionales.

1.4 Conclusiones parciales

A partir del análisis de la evolución del modelo teórico de las BDD y de los trabajos referidos anteriormente, se ha reconocido la necesidad del desarrollo de métodos y herramientas computacionalmente tratables que ayuden a los diseñadores, en lugar de intentar reemplazarlos, como una forma muy apropiada de resolver el problema del diseño de distribución en BD.

Aunque muchos investigadores han propuesto modelos y han diseñado algoritmos para el diseño de la distribución, la mayoría de los modelos tienen una alta complejidad computacional y es difícil usarlos en un ambiente real.

Tradicionalmente, el trabajo de diseño de la distribución ha sido realizado manualmente por el diseñador de la BD basado en su experiencia y en heurísticas.

La aplicación de los métodos de Algoritmos Genéticos y Q-Learning para dar solución al modelo de asignación (Rosa, 2006) con replicación de fragmentos, aportó buenas soluciones para problemas de pequeña y mediana escala, sin embargo a medida que aumenta el tamaño de la instancia del problema, las

soluciones obtenidas se alejan de la solución óptima y evidencian problemas de convergencia. Ante las limitaciones de las soluciones planteadas en el marco teórico desarrollado en este capítulo, y dando continuidad a estas investigaciones, se justifica el desarrollo de un nuevo método como puede ser la propuesta de integración de técnicas heurísticas para obtener soluciones al problema de la asignación de fragmentos, donde se tenga en cuenta la replicación, se mantenga la complejidad computacional de los métodos anteriores, y se obtengan soluciones hasta el nivel físico.

CAPITULO II. MODULO DE ASIGNACION

En este capítulo se describe la integración del algoritmo Q-Learning a un Algoritmo Genético para dar una solución al problema de asignación planteado en el capítulo I; así como las consideraciones y variaciones impuestas al modelo de asignación para una implementación factible. Además se aborda la obtención del modelo físico de asignación mediante scripts que serán ejecutados en servidores de Microsoft SQL Server.

2.1 Consideraciones realizadas al modelo de asignación

El modelo descrito en el capítulo I sobre la asignación de datos en la red es muy general, para poder lograr una implementación computacional del mismo hay que hacer un conjunto de consideraciones que se describirán a continuación.

2.1.1 Condición de réplica

El modelo de la forma que está planteado siempre tiene como solución óptima la matriz nula, o sea, la no asignación de fragmentos en la red, lo que no tiene sentido desde el punto de vista práctico. Teniendo esto en cuenta se agrega una restricción de réplica, la cual establece que cada fragmento es ubicado en al menos un sitio. La condición de réplica queda modelada de la siguiente manera:

$$\sum_{\forall F_j \in F} x_{jk} \geq 1, \quad \forall s_k \in S$$

donde j y k son los índices de los fragmentos y los sitios respectivamente.

2.1.2 Tratamiento de restricciones

Con el objetivo de lograr un mayor desempeño y por ser el espacio libre para el almacenamiento una característica muy variable de los sitios, se asume que habrá espacio suficiente para el almacenamiento de la base de datos, por tanto no se tiene en cuenta la restricción de almacenamiento. En cuanto a la restricción de tiempo de respuesta de las aplicaciones, también es desechada por no contar con suficiente información en un diseño partiendo de cero, esta será considerada en futuras implementaciones de un modelo de reubicación, teniendo en cuenta parámetros de desempeño de las aplicaciones.

2.1.3 Informaciones adicionales

El asistente al cual se le incorporará el módulo implementado será insertado en una herramienta de ayuda al diseño de bases de datos distribuidas. Esta

herramienta debe mantener una gran cantidad de información detallada, descriptores o metadatos referentes a los datos que conforman cada base de datos objeto de diseño; sobre las aplicaciones que usan los datos para dar solución a los casos de uso o solicitudes de usuarios; sobre los sitios que intervendrán en el almacenamiento de los datos y el procesamiento de las solicitudes; y sobre la red que soporta la conexión entre sitios y el intercambio entre ellos a fin de efectuar sus funciones. Toda esta variedad de información es almacenada en el catálogo del sistema, que no es más que una base de datos relacional, autodescriptiva por naturaleza, que mantiene la herramienta para sus propios fines internos.

En el trabajo con el catálogo se usan métodos y lógica de acceso propios para almacenar y recuperar rápida y eficientemente la información que se necesita en la realización de las tareas para las cuales han sido concebidas las herramientas. Las tablas que componen dicho catálogo solo son usadas internamente por las herramientas, compartiendo por esta vía toda la información relevante para el proceso de diseño; debido a esto, no son de interés para los diseñadores como usuarios finales del sistema.

Se asume que parte de la información necesaria para dar solución al modelo de asignación ha sido captada en etapas anteriores del diseño y debidamente almacenada en el catálogo; la información relacionada con la cantidad y tipo de accesos de las aplicaciones a los fragmentos se capta con el asistente. Para el correcto funcionamiento del mismo, el catálogo debe ubicarse en el mismo directorio de su archivo ejecutable. El anexo 1 muestra una descripción más detallada del catálogo de la herramienta.

2.2 Aspectos generales del diseño y la implementación del asistente

El asistente se programó en el lenguaje de programación C Sharp, que está incluido en el paquete Microsoft Visual Studio 2005. Este lenguaje de programación está diseñado para construir una amplia variedad de aplicaciones que se ejecutan en el entorno .NET.

2.2.1 Diagrama de clases

La figura 2.1 muestra el diagrama estático de clases del módulo desarrollado en este trabajo.

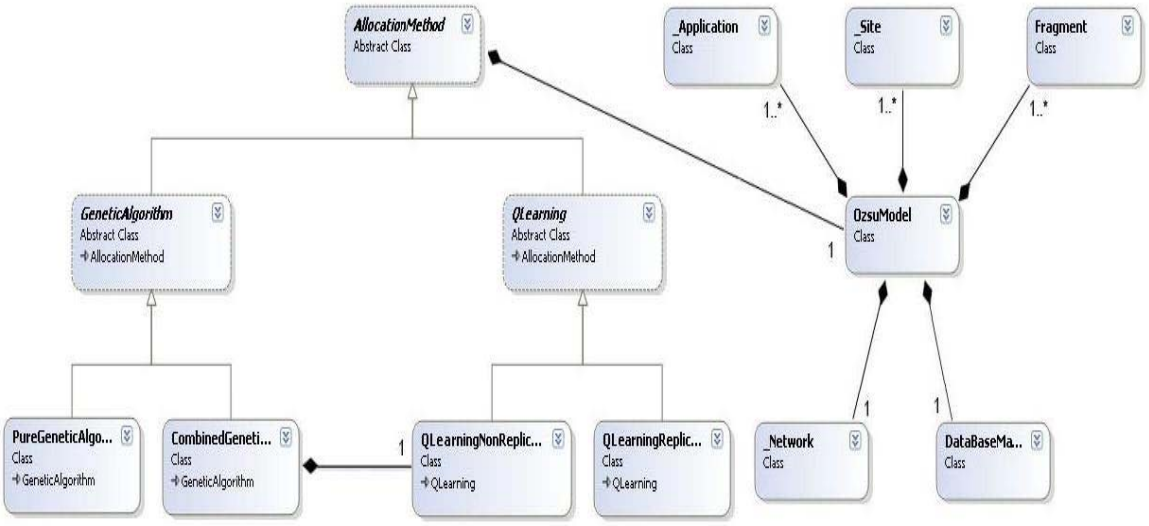


Figura 2.1. Diagrama de clases de la herramienta.

2.2.2 Definición de las clases

Las principales clases de este proyecto son: *GeneticAlgorithm*, *QLearning* y *OzsuModel*. Las dos primeras son clases abstractas que imitan el comportamiento de los Algoritmos Genéticos y el algoritmo Q-Learning respectivamente. De *GeneticAlgorithm* heredan las clases *PureGeneticAlgorithm* y *CombinedGeneticAlgorithm* y de la clase *QLearning* heredan las clases *QLearningNonReplicatedAlternative* y *QLearningReplicatedAlternative*; todas ellas se encargan del proceso de optimización del modelo de asignación, representado por *OzsuModel*.

Las demás clases sirven de complemento a las primeras, cumpliendo determinadas funciones dentro del proyecto. A continuación se describen las más importantes.

La clase DataBaseManagement

Esta clase es la encargada de interactuar con la base de datos que representa al catálogo. Lee toda la información que necesitan los algoritmos para resolver el modelo del problema y salva el resultado de la asignación en el catálogo de la aplicación. Para ello se vale de una variedad de clases que usan la interfaz de programación OLE DB y que pertenecen al namespace System.Data.OleDb. Esta interfaz llevó a la práctica los conceptos teóricos de la estrategia Universal de Acceso a Datos (UDA) de Microsoft, permitiendo el acceso a cualquier tipo datos tanto relacional como no relacional.

La clase OzsuModel

Esta clase representa al modelo de asignación presentado en el capítulo I, el cual se desea optimizar. Cuenta con arreglos de instancias de las clases *Site*, *Fragment* y *Application* y con una instancia de la clase *Network*. Cada una de estas clases contiene la información que le corresponde dentro del modelo.

Otro atributo importante perteneciente a esta clase es una matriz que toma valores binarios, declarada de tipo *char*, que representa la asignación de los fragmentos en los sitios.

Por último, cuenta con una instancia de la clase *DataBaseManagement* para interactuar con el catálogo.

La clase AllocationMethod

Esta es una clase abstracta que cuenta con una instancia de la clase *OzsuModel*. De ella heredan las clases *GeneticAlgorithm* y *QLearning*. Una instancia de esta clase dirige el funcionamiento del problema. En tiempo de ejecución y en dependencia del método seleccionado por el usuario, se inicializa con cualquiera de los constructores de las clases derivadas y comienza el proceso de optimización, invocando al método *Execute* correspondiente al algoritmo seleccionado, haciendo uso del polimorfismo.

2.2.2.1 Implementación del Algoritmo Genético

Los Algoritmos Genéticos (AG) surgen como herramientas para la solución de complejos problemas de búsqueda y optimización, producto del análisis de los sistemas adaptativos en la naturaleza, y como resultado de abstraer la esencia de su funcionamiento. Son métodos de búsqueda de propósito general basados en los mecanismos de la selección natural y la genética.

En el uso de los AG es necesario encontrar una posible estructura (cromosoma) para representar las soluciones; una instancia de esta estructura representa un punto o un estado en el espacio de búsqueda de todas las posibles soluciones (Gálvez, 1998).

Al optimizar una estructura usando un AG se necesita una medida de su calidad en el espacio de búsqueda. La función de adaptabilidad es la encargada de esta tarea. Esta función se obtiene directamente a partir de la función de evaluación del problema correspondiente.

Un AG simple genera una población inicial de n individuos que se van modificando repetidamente mediante la aplicación de los siguientes operadores genéticos: el operador de selección o Darwiniano realiza la selección de las cadenas de acuerdo a su adaptabilidad para el posterior apareamiento, incrementa geométricamente la presencia de los individuos aventajados y reduce la presencia de los retrasados, aunque no introduce nuevos individuos; el operador de cruzamiento o Mendeliano realiza la recombinación del material genético de dos cadenas padres; el operador de mutación produce una alteración de un gen dentro de un cromosoma o cadena a sus diferentes formas alelomorfas, introduce variedad en el juego de esquemas de la población y proporciona un mecanismo de seguridad frente a posibles pérdidas de información valiosa (Pérez, 1996).

Para cada uno de estos operadores está asociado el uso de probabilidades y la generación de números aleatorios. Una vez completada la acción de los tres operadores se dice que ha transcurrido un ciclo generacional. Luego se repite el mismo proceso mientras no se garantice el criterio de parada.

A continuación se explica como se modelaron los principales elementos y operadores del Algoritmo Genético implementado para resolver el modelo de asignación presentado en el capítulo I.

La clase *GeneticAlgorithm*

Esta clase está compuesta por un arreglo de tamaño dos de instancias de la clase *Population* el cual representa dos poblaciones: la población actual y la población de la generación anterior. El mejor individuo de la población actual está representado por una instancia de la clase *Individual*.

Además cuenta con atributos que definen los parámetros de control del algoritmo, tales como el tamaño de la población, el número máximo de generaciones, la tolerancia, y las probabilidades de cruzamiento y mutación.

Todos los atributos y métodos de esta clase se transmiten automáticamente a las clases *PureGeneticAlgorithm* y *CombinedGeneticAlgorithm* mediante el mecanismo de la herencia. Estas clases tienen básicamente el mismo funcionamiento y se diferencian en la selección de la población inicial, que se hace de manera parcialmente aleatoria en la primera y en la segunda se utiliza las técnica de sembrado de individuos. En (Rosa, 2006) se hace un análisis más detallado de la clase *PureGeneticAlgorithm*.

El Algoritmo Genético usado es de tipo generacional, donde la nueva población reemplaza totalmente a la antigua población, excepto el mejor individuo que pasa de una generación a la siguiente, garantizando que si en alguna de ellas se alcanza el óptimo, no se pierda en los procesos de cruzamiento o mutación.

Definición del cromosoma

El cromosoma usado es una matriz de tipo *char* que toma valores binarios. Con esto se logra que su representación esté lo más cerca posible de la solución real. Las columnas del cromosoma representan el índice de los sitios y las filas representan el índice de los fragmentos; la intersección de la fila i con la columna j significa que el fragmento i está almacenado en el sitio j.

Generación de la población inicial

Debido a que el sembrado en una población con soluciones de alta calidad obtenidas de otra técnica heurística, ayuda al AG a encontrar mejores soluciones más rápido que con un comienzo aleatorio (Gálvez, 1998), este método es utilizado para obtener una población inicial en la clase *CombinedGeneticAlgorithm*.

Conocidos los buenos resultados de técnicas de Aprendizaje Reforzado en cuanto a rapidez y convergencia al óptimo para solucionar el problema de asignación de fragmentos sin tener en cuenta la replicación (Pérez, 1999), y al hecho de que en varios trabajos desarrollados anteriormente (Ceri et al., 1987) se parte de una solución inicial sin réplica y luego, siguiendo un modelo de reubicación de fragmentos se obtiene una con réplica; se utiliza el algoritmo Q-Learning para obtener una solución inicial sin réplica la cual se utiliza para sembrar el 10% de la población inicial. Esta variante del Q-Learning se implementa en la clase *QLearningNonReplicatedAlternative* la cual se explica más detalladamente en los epígrafes siguientes.

El resto de la población se obtiene de forma parcialmente aleatoria. Se inicializa la matriz que representa el cromosoma de cada individuo de modo que los fragmentos se ubiquen en un solo sitio, escogido aleatoriamente. De esta forma se garantiza que la asignación inicial cumpla con la condición de réplica.

Debido a que el sembrado de individuos iguales en la población inicial contradice en parte la teoría de que en términos generales esta debe ser lo más variada posible y la necesidad de que la distribución de aptitudes sea uniforme para evitar la convergencia temprana (Pérez, 1996); es conveniente introducir en la misma un

mayor grado de diversidad, esto se logra fijando el tamaño de la población en 120 individuos.

Función de evaluación

Para asignar la adaptabilidad de cada individuo al medio, se utiliza la función de costo del modelo planteado anteriormente. Si el individuo no cumple con alguna de las restricciones se penaliza, asignándole un valor de adaptabilidad cualitativamente muy bajo, (como se está minimizando este valor sería muy alto cuantitativamente).

Selección

En las generaciones iniciales de un AG los valores de adaptabilidad promedio son bajos. La presencia de algunas cadenas con un valor de adaptabilidad relativamente alto provoca que un mecanismo de selección proporcional asigne un número grande de copias a estas supercadenas provocando así la convergencia temprana. Lo más seguro es que el individuo más destacado de la población, aun cuando es mediocre, logre apropiarse de la próxima generación y por lo tanto no hay posibilidades de continuar la evolución. De esta forma se ha logrado la convergencia sin asegurarse que el valor a donde se converge es en realidad un valor bueno (Goldberg, 1989).

Por otra parte, en las postrimerías de la búsqueda todos los individuos son demasiado “buenos”, entonces la varianza entre los valores de adaptabilidad de las cadenas es pequeña y por tanto el esquema de selección proporcional asigna más o menos igual número de copias para todas las cadenas, la población se mantendrá por ese orden de bondad y será muy difícil buscar individuos aun mejores, salvo por un golpe de “suerte” logrado mediante la mutación o el cruzamiento.

Para evitar los inconvenientes de la selección proporcional antes mencionados se usa para este problema el mecanismo de selección estocástico de torneo, en este caso de tamaño dos. Esta alternativa combina la idea del ranqueo de la población con el método de selección de la ruleta, evitando la convergencia temprana. Consiste en seleccionar de manera aleatoria dos individuos de la población y escoger el que tenga mayor adaptabilidad.

Cruzamiento

El cruzamiento se lleva a cabo si al generar un número aleatorio es menor o igual que la probabilidad de cruce. El cruce se realiza de manera análoga al método tradicional, se selecciona una fila de cruce de forma aleatoria, las filas que ocupan posiciones inferiores o igual a la fila de cruce se mantienen igual y las filas que ocupan posiciones superiores se intercambian entre ambos individuos. Se tomó 0.90 como probabilidad de cruzamiento.

Mutación

La mutación se lleva a cabo si al generar un número aleatorio es menor o igual que la probabilidad de mutación. Semejante al método tradicional, se selecciona de forma aleatoria un elemento de la matriz del cromosoma del individuo y su valor se sustituye por su complemento. Se tomó 0.1 como probabilidad de mutación.

Condición de parada

Para detener el algoritmo se usa un criterio de terminación mixto que consta de un mecanismo de tipo MAX y cierta tolerancia de convergencia. El mecanismo de tipo MAX fija un número máximo de generaciones y es usado dado que la función de evaluación es complicada. Paralelamente se usa un criterio de tipo “TOL“ enfocado al contenido, el cual hace parar al algoritmo una vez que se hayan superado ciertos requisitos de convergencia, independientemente de la iteración en que se encuentre. Esto es: si la adaptabilidad del mejor individuo dividida entre el promedio de adaptabilidad de los individuos no penalizados es mayor que la tolerancia, se detiene el algoritmo. En el caso particular de este problema se fijó la tolerancia en 0.995 y el número máximo de generaciones en 60.

El método OneEpoch

Este método lleva a cabo una generación completa del algoritmo. En él se realizan los procesos de selección, cruce y mutación. La figura 2.2 muestra el seudocódigo asociado al mismo.

El método Execute

Este método inicia el proceso de optimización del Algoritmo Genético. Luego de completarse el ciclo generacional se copia la matriz que representa el cromosoma del mejor individuo de la población correspondiente a la última generación, en la matriz de solución del modelo; y se determina la calidad de la solución en dependencia si el individuo está penalizado o no.

```
NuevaPoblacion[0]←MejorIndividuo   // Elitismo
i←0
Mientras i < TamañoPoblacion hacer
   hijo1←NuevaPoblacion[i]
   hijo2←NuevaPoblacion[i+1]
   padre1←SeleccionarIndividuo()
   padre2←SeleccionarIndividuo()
   si AplicarOperador(ProbabilidadCruce) entonces
      Cruzar(padre1, padre2, hijo1, hijo2)
   si no
      hijo1←padre1
      hijo2←padre2
   si AplicarOperador(ProbabilidadMutacion) entonces
      Mutar(hijo1)
   si AplicarOperador(ProbabilidadMutacion) entonces
      Mutar(hijo2)
   i←i+2
generacion ← generacion + 1
EvaluarNuevaPoblacion
```

Figura 2.2. Seudocódigo del método *OneEpoch* de la clase *GeneticAlgorithm*

Durante la evaluación de la población se le calcula a cada individuo su adaptabilidad por medio de la función de evaluación; así mismo se obtiene el mejor individuo de la población y se verifica la condición de parada del algoritmo que utiliza el criterio de la tolerancia de convergencia.

La figura 2.3 muestra el seudocódigo asociado al método.

```
Obtener PoblacionInicial
Evaluar PoblacionInicial
generacion←0
Mientras generacion < NumMaxGeneraciones
   OneEpoch()
si MejorIndividuo penalizado entonces
   SolucionFactible←falso
si no
   SolucionFactible←verdadero
```

Figura 2.3. Seudocódigo del método *Execute* de la clase *GeneticAlgorithm*

Asignación de los parámetros de control

No existen criterios definitivos para dar valores a los parámetros de un Algoritmo Genético. En principio, están reputados como técnicas de búsqueda paramétricamente robustas, es decir, funcionan, mejor o peor, pero funcionan en un amplio rango de valores de sus parámetros. Sin embargo, las diferencias de eficiencia en la práctica son lo suficientemente espectaculares como para solicitar una mayor atención en dicha elección (Pérez, 1996).

Los parámetros de un Algoritmo Genético pueden ser tratados independientemente, pero el total desempeño del algoritmo no depende exclusivamente de un único parámetro sino de la combinación de todos ellos. Ya que los Algoritmos Genéticos están abiertos en su forma de implementación, muchos investigadores le prestan más atención a unos parámetros que a otros, pero la mayoría coincide en que los parámetros a controlar son los esquemas de selección, el tamaño de la población, la cantidad de generaciones y la variación de los operadores genéticos y sus probabilidades (Eiben, 1999).

Para llevar a cabo la asignación de los parámetros de control en el Algoritmo Genético implementado en esta tesis se realizó una serie de experimentos usando los casos de prueba presentados en el anexo 4. Los parámetros que fueron objeto de experimentación son la cantidad de generaciones, y las probabilidades de cruce y mutación; el método de selección de los demás parámetros fue expuesto en epígrafes anteriores. Los experimentos consisten en ejecutar 20 veces el algoritmo para los diferentes casos de prueba usando combinaciones de los valores de las probabilidades de cruce y mutación, variándolos dentro de los rangos de buen funcionamiento reportados en la bibliografía para cada parámetro. En cada ejecución se reporta la cantidad de generaciones que necesitó el algoritmo para alcanzar la tolerancia de convergencia exigida para este problema, nunca sobrepasando las 80 generaciones dado que la función de adaptabilidad tiene una complejidad alta. Luego de una secuencia de ejecuciones se calcula el error relativo del costo promedio de las soluciones alcanzadas respecto al óptimo y el promedio de la cantidad de generaciones consumadas para cada caso de prueba.

En la figura 2.4 se muestran los resultados obtenidos para los distintos casos de prueba y valores de los parámetros objeto de experimentación.

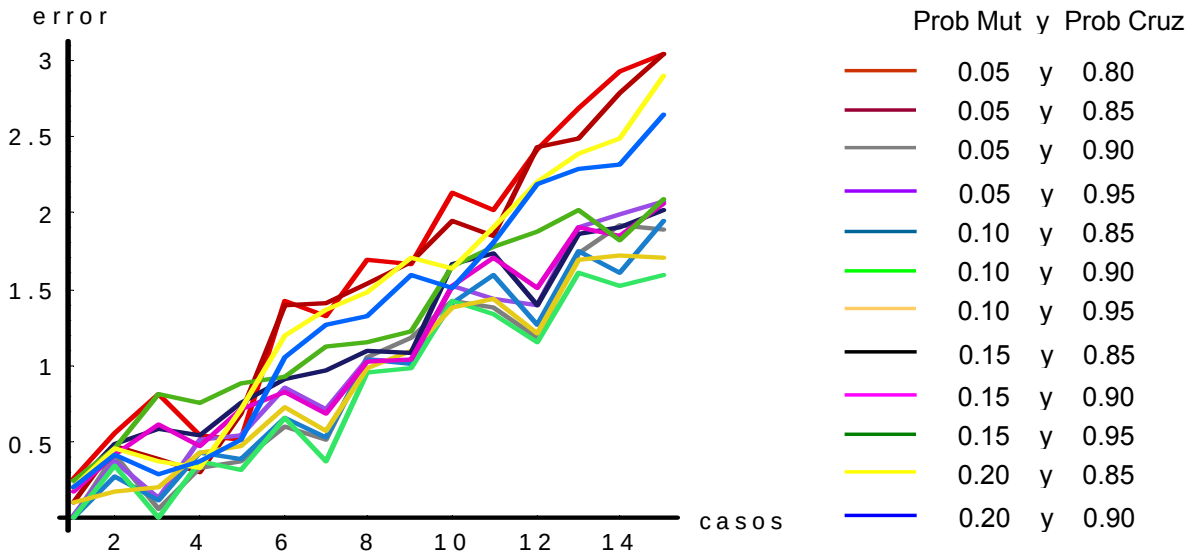


Figura 2.4. Resultados del proceso de ajuste de las probabilidades de cruzamiento y mutación.

Aunque los resultados obtenidos son dependientes en mayor o menor medida del problema en particular, se puede concluir que el algoritmo alcanza su mejor comportamiento asignándole una probabilidad de mutación de 0.1 y una probabilidad de cruzamiento de 0.90. Para estos valores el promedio de la cantidad de generaciones varía desde 45 para los casos pequeños hasta 80 para los casos mayores, estableciendo una media aproximada entre estos valores se fija la cantidad de generaciones en 60, véase la figura 2.5.

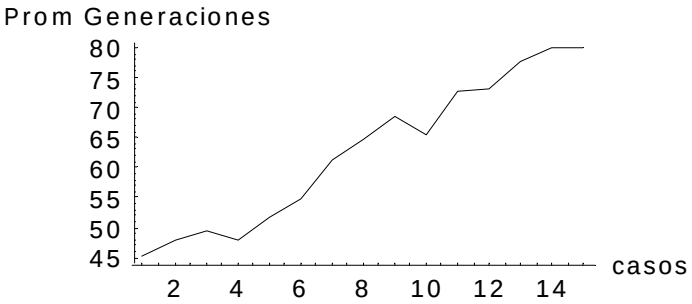


Figura 2.5. Promedio de la cantidad de generaciones consumadas para los casos de prueba.

2.2.2.2 Implementación del algoritmo Q-Learning

El Aprendizaje Reforzado (AR) es un conjunto de técnicas diseñadas para dar solución a problemas de decisión secuencial o control óptimo, cuya base son los procesos de decisión markovianos (MDP). La clave del AR está en la interacción continua entre un agente y su ambiente. El agente ejercita acciones de control que influyen en el estado del proceso y recibe una recompensa o penalización, dependiendo del estado resultante de cada acción tomada. De este modo adquiere

la experiencia necesaria para mejorar su comportamiento en el futuro.

Uno de los avances más importantes en el Aprendizaje Reforzado fue el desarrollo de un algoritmo off-policy de control de Diferencias Temporales conocido como Q-Learning (Cichosz, 1995; Watkins, 1999).

El aprendizaje de Diferencias Temporales es una combinación de los métodos de Monte Carlo y Programación Dinámica, donde se enfrenta la necesidad de intercambiar exploración y explotación, y este enfoque se divide en dos clases principales: política on (on-policy) y política off (off-policy) (Cichosz, 1995; Sutton y Barto, 1998; Mariano y Morales, 2001). La característica distintiva de una política on de una política off es que ellas estiman el valor de una política mientras son usadas para el control. La política para generar conducta (política de conducta) no debe estar relacionada a la política de evaluación y mejora (política de estimación). En un método de política off estas dos funciones son separadas, mientras que en la política on no lo están. Una ventaja de esta separación es que la política de estimación debe ser determinista (greedy), mientras que la política de conducta puede continuar muestreando todas las posibles acciones (ε-greedy). O sea, la política de comportamiento está separada de la política que se quiere mejorar.

Para obtener una solución del modelo de asignación sin réplicas que luego será usada para sembrar individuos en la población inicial del Algoritmo Genético, se emplea un algoritmo Q-Learning que posee las características que distinguen a un sistema de Aprendizaje Reforzado, contando con: agente, ambiente, estados, acciones, política, función de recompensa inmediata y función de valor. Este algoritmo es implementado en la clase *QLearningNonReplicatedAlternative*. La otra variante del algoritmo Q-Learning utilizada en el asistente ALLOCATOR para llevar a cabo la asignación de fragmentos es implementada en la clase *QLearningReplicatedAlternative*. Ambas clases heredan de la clase abstracta *Q-Learning*.

Entre otros atributos, la clase *QLearning* cuenta con dos matrices. La primera es una matriz declarada de tipo *char* que toma valores binarios, esta matriz tiene la misma estructura que la matriz de solución del modelo de asignación y precisamente representa la mejor solución que se ha obtenido hasta el momento. La segunda matriz representa la tabla Q que almacena el conocimiento adquirido por el agente, las filas de esta matriz representan los estados del algoritmo y las

columnas las acciones. Además cuenta con atributos que definen los parámetros de control del método, tales como la cantidad de episodios, la velocidad de aprendizaje, el factor de descuento y el factor de exploración.

El agente, el ambiente, la función de valor, la política y la inicialización del algoritmo son puntos comunes en ambas implementaciones del método Q-Learning. A continuación se explica detalladamente como se modelaron estos y otros elementos del método para la versión no replicada del algoritmo. En (Rosa, 2006) se explican los detalles para la versión replicada.

El agente

Su función es aprender a ubicar los fragmentos de la base de datos en los sitios de la red, de manera tal que se minimicen los costos de procesamiento y almacenamiento en un espacio de tiempo razonable.

Los estados

Están representados por los fragmentos e indican la condición actual del ambiente en que se ubica el problema.

Las acciones

Se encuentran representadas por los sitios, constituyen el conjunto de alternativas para ubicar los fragmentos. Estando en un estado determinado, la selección de una acción en particular indica que el fragmento correspondiente a dicho estado será asignado al sitio correspondiente a la acción escogida. La selección de las acciones se hace siguiendo la política *ε-greedy.*

El ambiente

El ambiente incluye toda la información que utiliza el modelo del MDP para definir la dinámica del sistema y está constituido por el catálogo de la aplicación. Allí se almacenan los datos que se necesitan para la selección de estados, acciones, recompensas inmediatas y transiciones del sistema, dentro del marco de las restricciones del problema.

La función de valor

Está representada por la matriz Q [estados, acciones], en la cual se acumulan los valores de recompensa que se han obtenido durante la interacción del agente con el ambiente. Luego de acumular suficiente conocimiento, su contenido le permite determinar a un agente si es factible o no ubicar un fragmento en un sitio determinado.

Inicialización del algoritmo

Antes de comenzar a ejecutar el ciclo de episodios del algoritmo, se necesita inicializar la matriz solución de la clase *OzsuModel* y la matriz Q. En el primer caso se tiene acceso a la matriz a través de una instancia de la clase *OzsuModel* que pertenece a la clase *AllocationMethod* y que es heredada por la clase *QLearning*. Para inicializarla se escoge una solución donde se ubica cada fragmento en un solo sitio, escogido de manera aleatoria. De esta forma se obtiene una solución inicial que cumpla con la condición de réplica. La segunda matriz se inicializa con todos sus valores en cero.

La política

Al encontrarse en un estado se debe seleccionar la acción que se va a tomar de acuerdo a la política seguida por el agente, basándose en la información almacenada en la tabla Q. En este caso se hace con una política ***ε-greedy***; es decir, en la tabla Q [Estados] [Acciones] se busca con una probabilidad (1- ε) el menor valor de las posibles acciones para este estado, donde ε es la constante de exploración. Así mismo, se elige una acción aleatoria de las posibles acciones con probabilidad ε. En los primeros episodios este índice es alto para que se exploren bastantes acciones y va disminuyendo a medida que transcurren los episodios, para poder explotar el conocimiento adquirido.

Actualización de la matriz de solución

La matriz de solución del modelo se actualiza luego de seleccionar una acción y no es más que asignar el fragmento correspondiente al estado actual, al sitio correspondiente a la acción tomada.

La función de recompensa inmediata

Luego de tomar una acción, la recompensa inmediata es el costo de ubicar el fragmento correspondiente al estado actual en el sitio representado por la acción escogida, para todas las aplicaciones, utilizando la función de costo del modelo de asignación.

El método Execute

Este método inicia el proceso de optimización del algoritmo Q-Learning. En cada episodio se recorren todos los estados actualizando la matriz de solución del modelo de asignación y la matriz Q. Luego se actualizan los parámetros de control del algoritmo. Si la solución actual cumple con la restricción de procesamiento se

evalúa en la función de costo total del modelo, si es menor que la obtenida hasta el momento, se almacena como la mejor solución.

Luego de completarse el ciclo de episodios se copia la matriz que almacena la mejor asignación en la matriz de solución del modelo; y se determina la calidad de la solución. La figura 2.6 muestra el seudocódigo asociado al método. La cantidad de episodios para este problema se fijó en 100.

```
Inicializar Q(s,a) en cero
Inicializar MatrizSolucion aleatorio
Inicializar α←0.9, ε←0.99, γ←0.6
btSol←MatrizSolucion
Repetir (para todos los episodios)
    Repetir (para cada estado s del episodio)
        a ←SeleccionarAccion usando una política ε - greedy
        ActualizarMatrizSolucion(s,a)
        r←RecompensaInmediata(s,a)
        s'←ProximoEstado(s)
        Actualizar Q
        s←s'
    si ε > 0.01 entonces
          ε ← ε – 0.01
    si MatrizSolucion factible entonces
        si cost(MatrizSolucion)< cost(btSol) entonces
            btSol←SolutionMatrix
si btSol factible entonces
    SolucionFactible←verdadero
si no
    SolucionFactible←falso
```

Figura 2.6. Seudocódigo del método *Execute* de la clase *QLearningNonReplicatedAlternative*

2.3 Análisis de la complejidad temporal del modelo

Las operaciones del modelo que requieren un gasto significativo de tiempo son obtener la población inicial del Algoritmo Genético y hallar en cada generación la adaptabilidad de todos los individuos de la población.

Sean $f : \mathrm{N} \to \Re^*$ y $g : \mathrm{N} \to \Re^*$ dos funciones que definen respectivamente el orden de la complejidad temporal de las dos operaciones antes mencionadas. Luego la complejidad temporal del modelo propuesto viene dada por el máximo de las complejidades de f(n) y g(n), véase la figura 2.7.

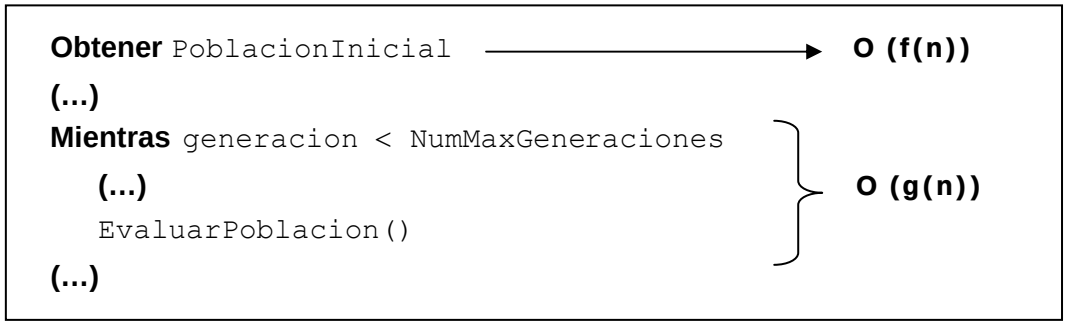


Figura 2.7. Operaciones del modelo propuesto que requieren un gasto significativo de tiempo.

La complejidad temporal de obtener la población inicial viene dada por la complejidad temporal del algoritmo Q-Learning usado para hallar el 10% de sus individuos. En este algoritmo la operación que implica un gasto de tiempo considerable es obtener el valor de la recompensa inmediata para todos los estados en cada uno de sus episodios, véase la figura 2.6. Dado que el cálculo de la recompensa inmediata involucra todas las aplicaciones, la complejidad temporal usando la notación asintótica es un $O(100 * n * q) = O(n * q)$, donde n y q son la cantidad de fragmentos y aplicaciones respectivamente y 100 es el número de episodios del algoritmo.

La complejidad temporal de la segunda operación se obtiene multiplicando la complejidad de hallar la adaptabilidad de un individuo de la población por el tamaño de la población y en el peor de los casos por el número de generaciones. La adaptabilidad de un individuo se calcula utilizando la función de costo total del modelo de asignación planteado en el capítulo I que involucra todos los sitios, fragmentos y aplicaciones. De esta forma la complejidad temporal es un $O(120 * 60 * m * n * q) = O(m * n * q)$, donde 120 y 60 son el tamaño de la población y el número de generaciones el Algoritmo Genético respectivamente y m es la cantidad de sitios.

De esta forma se concluye que la complejidad temporal del modelo híbrido propuesto en esta tesis es un $O(m * n * q)$.

2.4 Generación de scripts para SQL Server

La modelación hasta el nivel físico no puede ofrecer soluciones generales por su fuerte dependencia de los motores de BD. Por esta razón se ofrece una solución propia de este nivel para Microsoft SQL Server. La replicación de datos permite tratar algunos de los problemas que se presentan en sistemas distribuidos,

permitiendo a los usuarios que manejen localmente los datos, en lugar de acceder a grandes BDC a través de redes. También permite duplicar un servidor local de BD a un servidor remoto, y si uno falla las aplicaciones pueden continuar accediendo a copias en el otro.

Por su parte, el campo de la réplica de datos y la experimentación de métodos de replicación y su aplicación sistemática, también necesita de futuras investigaciones. La experimentación es requerida para evaluar los argumentos de los diseñadores de algoritmos y sistemas. Una de las dificultades de la evaluación cuantitativa de las técnicas de replicación es la ausencia de modelos comúnmente aceptados de fallos.

La replicación de datos de SQL Server es un conjunto de tecnologías destinadas a la copia y distribución de datos a diferentes ubicaciones, y para mejorar el rendimiento de las aplicaciones, separando físicamente los datos en función de cómo se utilicen estos o cómo se distribuye el procesamiento entre varios servidores. Este motor utiliza la metáfora de publicación/suscripción (Ulusoy, 2001a; Ulusoy, 2001b; Miklós, 2002; Schewe, 2002; Belokosztolszki et al., 2003; Chen et al., 2003; Eugster et al., 2003; Jiménez-Peris et al., 2003; Chakravarthy y Vontella, 2004; Gray, 2004; Aekaterinidis y Triantafillou, 2005) en su modelo de replicación. Las fases de la replicación son (Hotek, 2002): configuración, generación y aplicación de la instantánea inicial, modificación de los datos replicados y sincronización, y propagación de los datos.

SQL Server tiene tres métodos de replicación: instantánea, transaccional y mezcla. La tercera permite obtener copias diferidas de los datos y es la de más potencialidades para materializar un diseño distribuido de una BD. Entre sus características más relevantes están el permitir que varios sitios funcionen en línea o desconectados de manera autónoma, y mezclar más adelante las modificaciones de datos realizadas en un resultado único y uniforme. Este tipo de replicación es especialmente fuerte en el filtrado de opciones, lo que adquiere relevancia en el proceso de materialización de la distribución, permitiendo la creación de los fragmentos que se ubican en los sitios de procesamiento según los esquemas de asignación obtenidos por los algoritmos implementados. Los filtros horizontales permiten materializar la fragmentación horizontal primaria; los verticales, a su vez, la fragmentación vertical; y los combinados, por su parte, la fragmentación mixta.

Para implementar el diseño de la distribución mediante replicación, se emplea un método que utiliza la creación de los scripts con llamadas a procedimientos almacenados de Transact-SQL como vía para la materialización de la distribución. En estos scripts se recrea el entorno de replicación mediante la determinación de la directiva asíncrona de resolución de conflictos y la activación de la variable XACT_ABORT para una correcta ejecución concurrente de las transacciones (Breitbart y Korth, 1997; Holliday et al., 1999a; Pacitti et al., 1999a; Pacitti et al., 1999b; Pacitti y Simon, 2000; Pacitti et al., 2001; Irún-Briz et al., 2003; Akal et al., 2004; Daudjee y Salem, 2004; Elmeleegy et al., 2004; Irún-Briz et al., 2004; Akal et al., 2005; Daudjee y Salem, 2006) antes de implementar la replicación; se reconocen los servidores involucrados en la distribución; se crea la BD fragmentada y las vistas asociadas; se configuran las publicaciones, distribuciones y suscripciones de tipo Pull porque dan mayor autonomía a los suscriptores. El tipo de replicación elegido es el de mezcla, por ser el que da mayor autonomía a los sitios. Los datos en el suscriptor son automáticamente actualizables cuando se usa replicación de mezcla. Si el diseñador lo desea, puede modificar la secuencia de comandos una vez generada. Por otra parte, la configuración de los permisos de acceso a publicaciones forma parte de la administración y no de la modelación, por lo que se deja diferida al actor Administrador.

2.5 Descripción del asistente

Para ejecutar los algoritmos el asistente recibe como entrada la información almacenada en el catálogo. Se selecciona el algoritmo mediante el cual se pretende dar solución al problema y se ingresa la cantidad de accesos de lectura, de actualización y la selectividad de las aplicaciones sobre los fragmentos. Seguidamente se lleva a cabo el proceso de optimización. En caso de que el algoritmo seleccionado alcance una solución que cumpla con las restricciones del modelo de asignación, se muestra la asignación obtenida. El asistente le da la oportunidad al diseñador de la base de datos de escoger el nombre de los fragmentos físicos y si desea generar los scripts que servirán de ayuda para llevar a cabo el diseño físico de la base de datos sobre servidores de Microsoft SQL Server que se ejecutan en los sitios de procesamiento; en tal caso el diseñador debe escoger el nombre de los servidores. Luego el asistente mostrará los scripts

obtenidos para cada sitio a partir de los resultados de la asignación. El diseñador puede editar los scripts en la caja de texto correspondiente y guardarlos.

Es importante advertir que por razones de seguridad todos los parámetros que involucran contraseñas son encriptados con NULL o una cadena vacía en los scripts. Para que el script funcione correctamente los parámetros `@job login` y `@job password` de los procedimientos almacenados `sp addpublication snapshot` y `sp addmergepullsubscription agent` deben contener un nombre de usuario y contraseña válidos para el sistema de autentificación de Windows o para el del servidor de Microsoft SQL Server correspondiente.

En los sitios donde se está configurando el servidor como distribuidor, los parámetros `@data folder` y `@log folder` del procedimiento almacenado `sp adddistributiondb` deben contener una directorio válido del sistema. Además se debe crear una carpeta compartida que será usada para que todas las publicaciones del servidor almacenen los archivos snapshot. Esta carpeta debe tener permisos de “lectura y ejecución”, “listar contenido”, “lectura” y “escritura” para aquellas cuentas bajo las cuales publica un usuario del propio servidor o se subscribe un usuario de algún otro servidor; se sugiere que se comparta la carpeta `repldata` que se encuentra en el directorio de instalación del servidor, de lo contrario se deberá sustituir en el script la palabra `repldata` por el nombre de la carpeta compartida correspondiente.

2.6 Conclusiones parciales

Se especificó un algoritmo Q-Learning que soluciona el problema de asignación de fragmentos sin tener en cuenta la replicación. Esta solución se emplea para formar parte de la población inicial de un Algoritmo Genético que se utiliza para resolver el mismo problema, con replicación de fragmentos. La integración de ambas técnicas no aumenta la complejidad temporal del modelo propuesto. Se obtuvo el modelo físico de asignación mediante la generación de scripts que recrean el entorno de replicación usando secuencias de comandos en Transact-SQL.

CAPITULO III. VALIDACIÓN DEL MODELO

En este capítulo se valida el modelo obtenido como resultado de este trabajo mediante su aplicación en la solución de un caso de estudio y a través de pruebas realizadas al mismo. Se comparan los resultados alcanzados con los obtenidos mediante otros métodos heurísticos que solucionan el problema de asignación y con los de otros modelos reportados en la bibliografía.

3.1 Caso de estudio: Sistema de reservación para una aerolínea

En este epígrafe se presenta un ejemplo de aplicación de la herramienta en un caso de estudio propuesto en (Ceri et al., 1987), donde se describe el funcionamiento de un sistema distribuido de reservaciones de vuelos en varias localidades.

3.1.1 Descripción del caso de estudio

La compañía aérea que se presenta en este caso de estudio mantiene una base de datos distribuida sobre tres sitios: aeropuertos 1, 2 y 3; cada uno de los cuales se ubica en el centro de un área geográfica. Para ayudar a la visualización del sistema se toma una compañía que opera en Estados Unidos, cuyos principales aeropuertos se ubican en Denver, CO (oeste); Nueva York, NY (norte) y Atlanta, GA (sur). La base de datos almacena información relacionada con las regulaciones de los aeropuertos, vuelos disponibles, cronograma de vuelos y reservaciones de pasajeros.

3.1.2 Modelación conceptual

La modelación conceptual del problema constituye la etapa inicial en el diseño de la BDD e incluye la caracterización del Esquema Conceptual Global (ECG), la caracterización de la red y los sitios y la caracterización de las aplicaciones. Con el objetivo de simplificar la modelación fueron omitidos algunos atributos y características de los sitios que no son usados en el diseño de la distribución.

3.1.2.1 Caracterización del esquema conceptual global

La herramienta ERECASE (Rodríguez et al., 2002; García et al., 2005; Álvarez et al., 2006) permitió modelar con éxito el ECG para el problema planteado (véase la figura 3.1). Se asume que cada vuelo va directo desde un aeropuerto de salida a uno de destino, sin escalas intermedias. Todas las entidades, atributos e interrelaciones son explicativos por sí mismo.

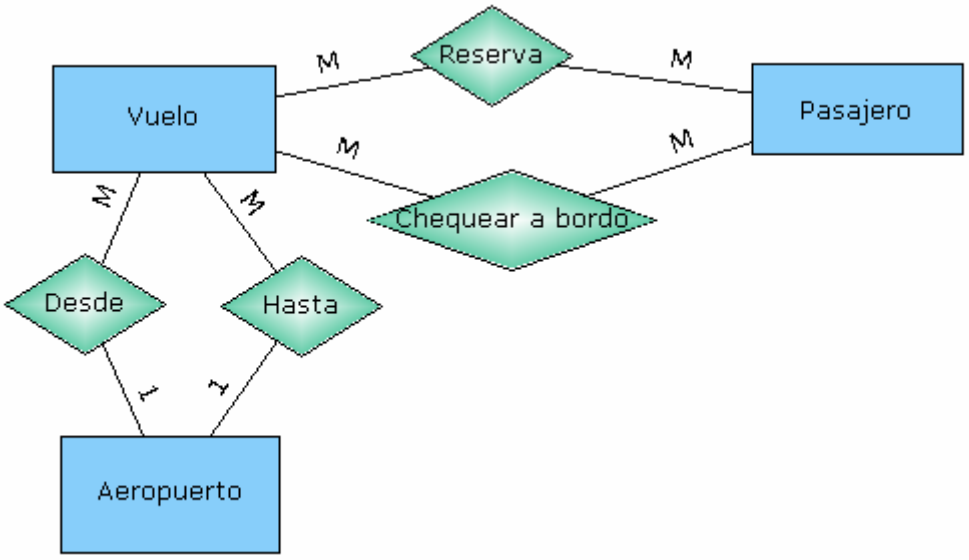


Figura 3.1. Diagrama Entidad-Relación para el caso de estudio Sistema de reservación para una aerolínea.

Los esquemas lógicos obtenidos a partir de la caracterización del ECG son:

Aeropuerto (Emblema, Área, Ciudad)

Vuelo (Número, Fecha, AeropuertoSalida, AeropuertoLlegada, HoraSalida, HoraLlegada, AsientosDisponibles, MapaAsientos, Puerta, Retraso)

Pasajero (Nombre, Teléfono)

Reservación (Nombre, Teléfono, NumVuelo, FechaVuelo, Clase)

Chequeo (Nombre, Teléfono, NumVuelo, FechaVuelo, NúmeroAsiento, EquipajesDespachados)

3.1.2.2 Caracterización de la red y los sitios

La información referente a los sitios se muestra en la tabla 3.1. La columna *SiteName* indica el nombre del sitio, los tiempos de lectura (RAT) y escritura (WAT) en disco y el tiempo máximo de respuesta permitido (MAT) se miden en milisegundos.

Tabla 3.1. Parámetros de los sitios.

SiteName	*RAT*	*WAT*	*MAT*
Denver	0,03098	0,06196	10000
Nueva York	0,05236	0,08237	10000
Atlanta	0,04031	0,07467	10000

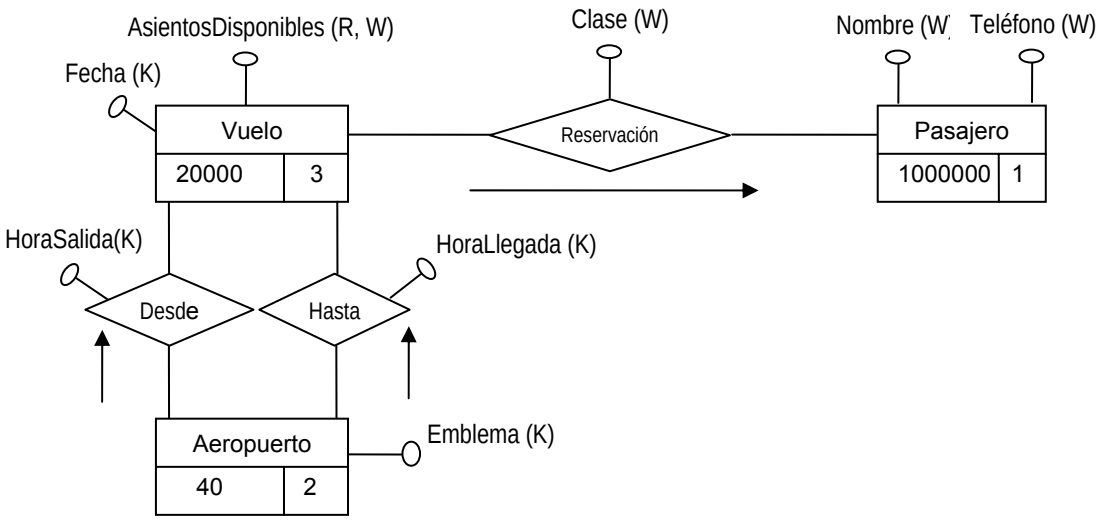
Por otra parte, los detalles que caracterizan la red de comunicación aparecen en la tabla 3.2, registrándose el promedio del costo de envío (AvgT) de un frame de un sitio a otro (FromStation y ToStation) medido en milisegundos.

Tabla 3.2. Costos de comunicación por frame entre los sitios de la red.

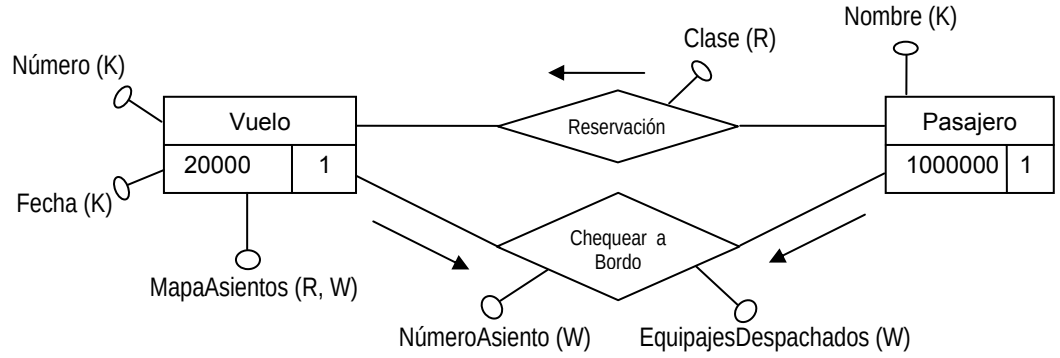
FromStation	*ToStation*	*AvgT*
Denver	Atlanta	0,7612
Denver	Nueva York	0,5373
Nueva York	Denver	0,5373
Nueva York	Atlanta	0,6
Atlanta	Denver	0,7612
Atlanta	Nueva York	0,6

3.1.2.3 Caracterización de las aplicaciones

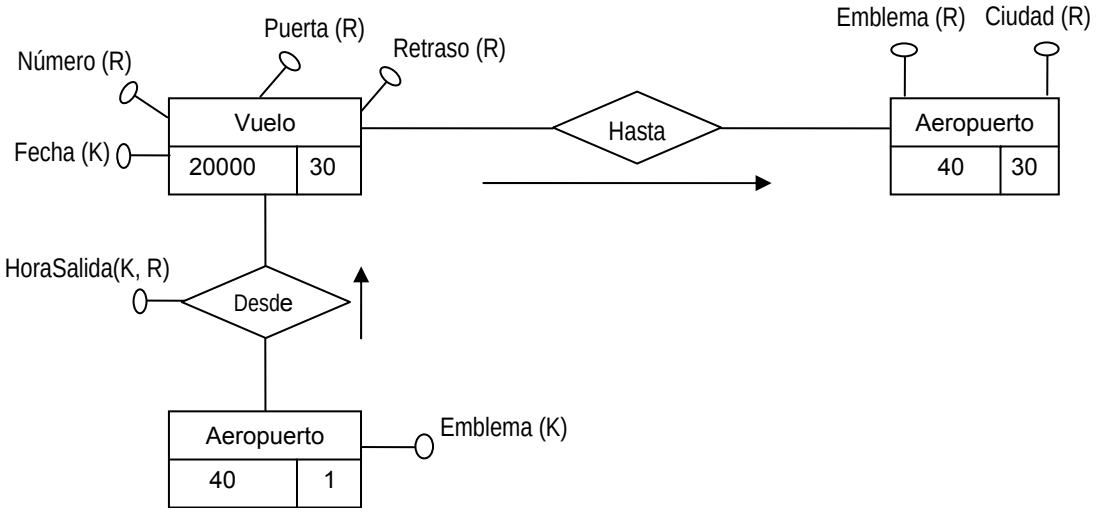
El diseño de la distribución también requiere recolectar información sobre las aplicaciones más importantes que se ejecutan sobre la base de datos. La heurística 80/20 asegura que se obtendrá el 80 por ciento de los accesos a la base de datos analizando el 20 por ciento de las operaciones más frecuentes (Ceri et al., 1987). La figura 3.2 muestra las aplicaciones más importantes identificadas para este problema.



(a)



(b)



(c)

Figura 3.2. Descripción de las aplicaciones. (a) Hacer Reservación.
(b) Chequear-a-Bordo. (c) Salidas-Aeropuerto.

La aplicación "*Hacer Reservación*" (véase la figura 3.2a) se activa cada vez que un pasajero quiere hacer una nueva reservación para un vuelo. En este caso el acceso a la base de datos es a través de los aeropuertos de salida y destino, la hora de salida y llegada y de la fecha del vuelo. Estos atributos son marcados con un *K* en el diagrama, lo cual indica que son usados como llaves para el acceso a los datos. Las flechas indican que el acceso procede desde la entidad *Aeropuerto* hasta la entidad *Vuelo* mediante las interrelaciones *Desde* y *Hasta*. Los números en la esquina inferior izquierda y derecha de los tipos de entidad indican la cardinalidad del esquema y la selectividad de la aplicación respectivamente. Una vez que el vuelo haya sido identificado, se crea una nueva instancia de la entidad *Pasajero* y una nueva instancia de la interrelación *Reservación*. Los atributos

Nombre, *Teléfono* y *Clase* son actualizados (*W*) en la base de datos y el atributo *AsientosDisponibles* es primero leído (*R*) y luego actualizado (*W*).

Una aplicación similar, llamada "*Continuar-Reservación*" permite tomar las reservaciones para aquellos pasajeros cuyos datos ya han sido recogidos y almacenados en la entidad *Pasajero*. Esta aplicación no se considera entre las más relevantes.

La aplicación "*Chequear-a-Bordo*" (véase la figura 3.2b) se ejecuta en el momento que un pasajero aborda el avión. Teniendo en cuenta el nombre del pasajero y la fecha del vuelo (*K*), se detectan las instancias de *Pasajero* y *Vuelo* involucradas en la consulta; luego se lee la *Clase* (*R*), y dado el valor del atributo *MapaAsientos* (*R*) de la entidad *Vuelo* se le asigna un número de asiento al pasajero. Los atributos *MapaAsientos*, *NúmeroAsiento* y cantidad de *EquipajesDespachados* por el pasajero son actualizados en la base de datos.

La aplicación "*Salidas-Aeropuerto*" (véase la figura 3.2c) hace un reporte donde se describe la información relativa a la salida de los próximos 30 vuelos que parten de un aeropuerto para ser mostrados en los monitores de las salas de los aeropuertos. Para identificar las instancias de las entidades *Aeropuerto* y *Vuelo* involucradas en la consulta se usa el emblema del aeropuerto y la fecha y hora vigentes. Para cada vuelo se leen los atributos *Número*, *Hora de salida*, *Puerta*, *Retraso*, y del aeropuerto de destino el *Emblema* y la *Ciudad*. La información relacionada con el aeropuerto de destino se obtiene usando la relación *Hasta*.

La tabla 3.3 muestra las frecuencias con se activan las aplicaciones a, b y c en los sitios 1 (Denver), 2 (Nueva York), 3 (Atlanta) en una unidad de tiempo.

Tabla 3.3.Tabla de frecuencia de activación de las aplicaciones en los sitios.

Tabla de Frecuencia	*Sitios*		
	1	2	3
a	10000	20000	10000
b	8000	12000	8000
c	100	120	100

3.1.3 Modelación lógica

La primera etapa de la modelación lógica se corresponde con la fragmentación. En este proceso se observa la elección del atributo *Área* para establecer un criterio de fragmentación horizontal primaria para la entidad *Aeropuerto*, obteniéndose tres

fragmentos en dependencia del área a la que pertenece el aeropuerto (véase la figura 3.3a). Luego se realiza una fragmentación horizontal derivada a las entidades *Vuelo* y *Pasajero*, inducidas por la fragmentación primaria de los aeropuertos en áreas (véase la figura 3.3b). Una vez que los aeropuertos fueron particionados por áreas, la entidad *Vuelo* se fragmenta por el aeropuerto de salida de cada vuelo; obteniéndose de esta forma tres fragmentos.

El esquema *Pasajero* también es objeto de una fragmentación horizontal derivada mediante el número del vuelo, basándose en el aeropuerto de salida de todos los vuelos para los cuales ha reservado. En este caso se obtienen siete nuevos fragmentos: los pasajeros pueden tener reservaciones sobre vuelos que parten de un área solamente (1, 2 ó 3); o de dos áreas (1-2, 2-3, ó 1-3) o finalmente de todas las áreas (1-2-3). Nótese que la asignación de las instancias de esta entidad es dinámica, pues cada vez que el pasajero hace una nueva reservación la correspondiente instancia de la entidad podría moverse de un sitio a otro.

Entidad	*Atributo de fragmentación primaria*	*Fragmentos*	*Predicados*	*Cardinalidad*
Aeropuerto	Área	AirportCO	Área = ‘W’	12
		AirportNY	Área = ‘N’	18
		AirportGA	Área = ‘S’	10

(a)

Entidad	*Atributo de frag. derivada*	*Entidad base*	*Fragmentos*	*Cardinalidad*
Vuelo	Aeropuerto de salida	Aeropuerto	FlightCO	6000
			FlightNY	9000
			FlightGA	5000
Pasajero	Número de vuelo	Vuelo	PassengerCO	200000
			PassengerNY	200000
			PassengerGA	200000
			PassengerCO&NY	100000
			PassengerCO&GA	100000
			PassengerNY&GA	100000
			PassengerCO&NY&GA	100000

(b)

Figura 3.3. (a) Tabla de la fragmentación horizontal primaria.
(b) Tabla de la fragmentación horizontal derivada.

A continuación se muestran los resultados de la asignación de los fragmentos obtenidos durante la modelación lógica del problema descrito en el presente caso de estudio. Las variantes de asignación fueron obtenidas siguiendo una estrategia greedy, y aplicando el algoritmo híbrido propuesto en el capítulo II.

3.1.3.1 Asignación obtenida mediante una estrategia greedy

El algoritmo de asignación de fragmentos siguiendo una estrategia greedy se divide en dos fases: primero una asignación no redundante y luego una asignación redundante de los fragmentos. La asignación no redundante se lleva a cabo siguiendo el criterio de ubicar cada fragmento en el sitio donde más se usa. La frecuencia estimada del uso de un fragmento en un sitio viene dada por la sumatoria, para todas las aplicaciones que se originan en el sitio, del producto entre las polarizaciones y las frecuencias de activación de cada aplicación en ese sitio. De esta forma es posible identificar el sitio desde el cual un fragmento tiene una mayor cantidad de accesos.

Dados:

f_{ij}: frecuencia de activación de la aplicación i en el sitio j.

p_{ijk}: polarización del fragmento k, dada una aplicación i que se origina en el sitio j.

La tabla 3.4 muestra la tabla de polarización. Las columnas están asociadas con las activaciones de las aplicaciones en cada sitio, las filas con los fragmentos obtenidos durante la fragmentación. Cada entrada indica el por ciento de accesos que una aplicación, que se activa en un sitio dado, realiza para un determinado fragmento. Solo algunas entradas han sido especificadas, las restantes pueden ser calculadas asumiendo una distribución uniforme de las demás instancias. Las aplicaciones que aparecen tachadas no son relevantes debido a que no usan el esquema asociado.

Tabla 3.4. Polarización

		a			b			c		
		1	2	3	1	2	3	1	2	3
Aeropuerto por área	F1	80						100		
	F2		75						100	
	F3			80						100
Vuelo por aeropuerto de salida	F4	70			100			80		
	F5		75			100			80	
	F6			70			100			80
Pasajero por aeropuerto de salida	F7	40			40					
	F8		40			40				
	F9			30			30			
	F10	30	20		30	20				
	F11	10		30	10		30			
	F12		20	10		20	10			
	F13	20	20	20	20	20	20			

El número de accesos al fragmento k desde el sitio j viene dado por:

$$n_{kj} = \sum_{i:\, i\ usa\ k} f_{ij} \cdot p_{ijk} \qquad (1)$$

Luego el fragmento k es asignado al sitio j tal que:

$$n_{kj} = \max_{\forall j} n_{kj}$$

La asignación redundante se realiza seleccionando sitios adicionales para ubicar cada fragmento teniendo en cuenta que el beneficio obtenido por los accesos de solo lectura debe exceder al costo que implica la actualización de las copias redundantes. Una medida aproximada de este beneficio se puede obtener evaluando la diferencia entre el número de accesos de lectura, que se convierten en locales debido a la adición de una nueva copia, y los accesos remotos de actualización que se requieren para mantener esta copia. Este número debe ser considerablemente mayor que cero para justificar la redundancia. La diferencia es evaluada para un fragmento k y una posible nueva copia almacenada en el sitio j como sigue:

$$d_{kj} = \sum_{i:\, i\ lee\ k} f_{ij} \cdot p_{ijk} - \sum_{i:\, i\ actualiza\ k} \sum_{h \neq j} f_{ih} \cdot p_{ihk} \qquad (2)$$

Siguiendo esta estrategia, los fragmentos *AirportCO*, *FlightCO* y *PassengerCO* son ubicados en el sitio 1, los fragmentos *AirportNY*, *FlightNY* y *PassengerNY* son

ubicados en el sitio 2, y los fragmentos *AirportGA*, *FlightGA* y *PassengerGA* son ubicados en el sitio 3; de esta misma manera se asignan los fragmentos *PassengerCO&NY*, *PassengerNY&GA* y *PassengerCO&NY&GA* al sitio 2 y finalmente *PassengerCO&GA* al sitio 3.

En la segunda fase de la asignación se calculan los beneficios de asignar copias de los fragmentos a los demás sitios utilizando la ecuación (2), resultando factible la replicación de los fragmentos obtenidos del esquema *Pasajero*. Los fragmentos *PassengerCO&NY*, *PassengerCO&GA* y *PassengerNY&GA* corresponden a aquellos pasajeros que habían reservado vuelos que partían de dos áreas; estos fueron ubicados en los dos sitios correspondientes. Así mismo, el fragmento *PassengerCO&NY&GA*, que incluye pasajeros con vuelos que parten de las tres áreas fue asignado a todos los sitios.

Tomando la función de costo del modelo de asignación abordado en el capítulo I, la asignación final obtenida, siguiendo la estrategia greedy implica un costo de 14215,89 milisegundos. Este resultado será comparado más adelante con el obtenido por el método que propone esta tesis.

3.1.3.2 Asignación obtenida mediante el algoritmo híbrido

La solución del problema no replicado obtenido mediante el algoritmo Q-Learning es igual a la obtenida por el método greedy e implica un costo de 16237,28 milisegundos. Esta solución es usada para formar 12 individuos que pasan a formar parte de la población inicial del Algoritmo Genético. Durante su ejecución se pudo comprobar la rapidez en la convergencia del algoritmo que solo necesita 42 generaciones para satisfacer el criterio de parada de tipo “TOL“, con una tolerancia de 0.9991 mayor que 0.995 exigida por el algoritmo. La solución obtenida ubica los fragmentos *AirportCO*, *AirportNY* y *AirportGA* en el sitio 1 y los replica en los restantes sitios; los fragmentos, *FlightCO* y *PassengerCO* son ubicados en el sitio 1, los fragmentos, *FlightNY* y *PassengerNY* son ubicados en el sitio 2, y los fragmentos, *FlightGA* y *PassengerGA* son ubicados en el sitio 3; de esta misma manera se asignan los fragmentos *PassengerCO&NY* a los sitios 1 y 2, *PassengerNY&GA* a los sitios 2 y 3, *PassengerCO&GA* a los sitios 1 y 3 y finalmente *PassengerCO&NY&GA* es ubicado en los tres sitios. El costo total de la ubicación según el modelo de asignación es de 13645,18 milisegundos, inferior al obtenido por el método que sigue la estrategia greedy, descrito en la sección

anterior; esto se explica por la replicación de los fragmentos *AirportCO*, *AirportNY* y *AirportGA* en los restantes sitios de la red comoquiera que ninguna aplicación hace accesos de actualización sobre los mismos.

3.1.4 Modelación física

Se generó un conjunto de scripts que recrean el entorno de replicación usando secuencias de comandos en Transact-SQL, los cuales deberán ser ejecutados en los servidores de los sitios correspondientes.

3.2 Casos de prueba

Con el propósito de hacer un análisis comparativo entre las soluciones obtenidas por el modelo implementado y los resultados de trabajos de investigación previos (Pérez, 2002; González, 2001) se escogió un banco de datos con 10 casos de prueba y las soluciones obtenidas aplicando los métodos de Redes Neuronales, el algoritmo SARSA de Aprendizaje Reforzado y Búsqueda Tabú, sin tener en cuenta la replicación de fragmentos. Para obtener estos casos de prueba se generaron problemas de solución conocida que fácilmente pueden hacerse crecer adicionando componentes desacoplados en la diagonal de una matriz. A este método se le conoce como Método de Componentes Desacoplados (Cruz, 1999). Para el problema de asignación, cada componente desacoplado es una copia de un problema básico generado cuya solución se puede obtener mediante cualquier software comercial de optimización combinatoria. Para formar un nuevo problema, se concatenan dos problemas básicos en forma disjunta. Así, por la propiedad de independencia, el valor de la solución del nuevo problema es la sumatoria de los valores de las soluciones de cada problema básico, véase el anexo 3.

Además se utilizaron los casos de prueba aleatorios presentados en (Rosa, 2006) que dados los rangos de valores entre los cuales se desea que se encuentren el número de fragmentos, de sitios y de aplicaciones, obtenga los datos necesarios para caracterizar una instancia del problema. En los tres primeros casos se encontró el óptimo (en milisegundos) por un método exacto de enumeración total. En los casos restantes se muestra la mejor solución obtenida en las etapas de prueba del software. El anexo 4 muestra estos casos de prueba.

3.3 Resultados experimentales

En esta sección se presentan los resultados obtenidos en los experimentos realizados al software con el objetivo de evaluar la calidad de la solución.

Se comparó las soluciones obtenidas por el algoritmo Q-Learning implementado con las soluciones brindadas por el método SARSA en los casos de prueba expuestos en el anexo 3. Se ejecutó el algoritmo Q-Learning 20 veces para los diferentes casos de prueba, tomando en cada caso el promedio del costo de las soluciones y el error relativo del promedio de las soluciones respecto al óptimo. El anexo 5 muestra los resultados de los experimentos realizados.

La figura 3.4 muestra el comportamiento de ambos algoritmos teniendo en cuenta el error relativo para los diferentes casos de prueba.

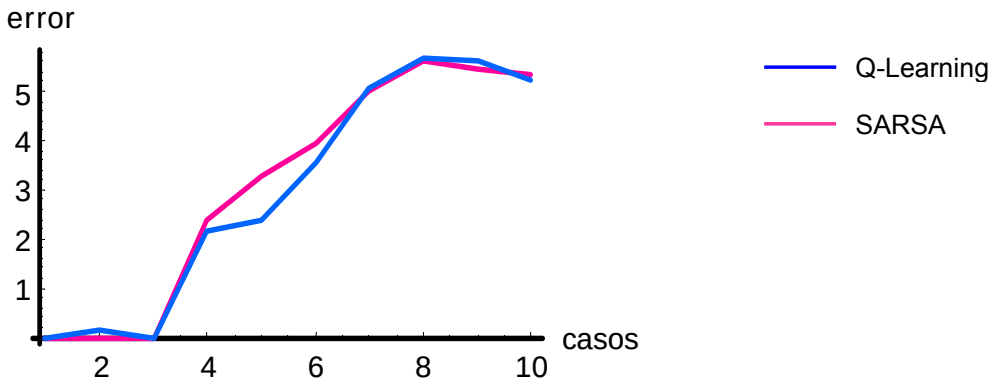


Figura 3.4. Comportamiento de los algoritmos SARSA y Q-Learning teniendo en cuenta el error relativo del promedio de soluciones respecto a la solución óptima.

Por otra parte se realizó una comparación entre los métodos propuestos en (Rosa, 2006), el algoritmo greedy analizado en este capítulo y el método híbrido obtenido como resultado de esta tesis. Se ejecutó 20 veces cada algoritmo para los diferentes casos de prueba. Se determinó para cada caso el promedio del costo de las soluciones, las veces que alcanzó el óptimo y el error relativo del promedio de soluciones de los algoritmos respecto al óptimo. El anexo 6 muestra los resultados de los experimentos realizados.

La figura 3.5 muestra el comportamiento de los algoritmos teniendo en cuenta el error relativo para los diferentes casos de prueba.

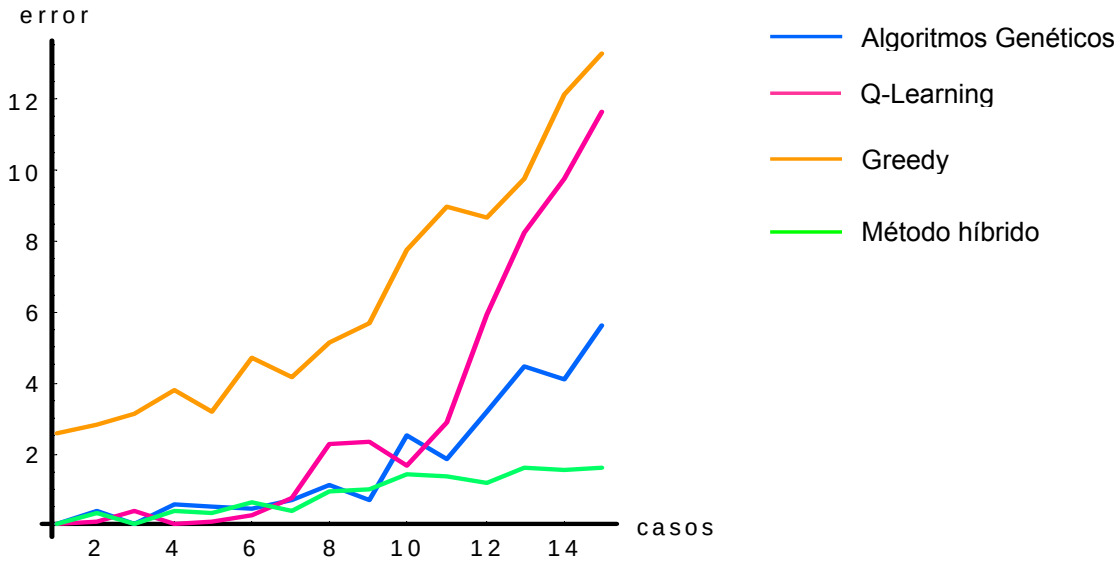


Figura 3.5. Comportamiento de los algoritmos teniendo en cuenta el error relativo del promedio de soluciones respecto al óptimo.

3.4 Conclusiones parciales

El modelo propuesto en esta tesis ha sido aplicado en la solución del caso de estudio "Sistema de reservación para una aerolínea" presentado en (Ceri et al., 1987). El costo de la asignación obtenida es inferior al reportado para el caso de estudio.

Además, las soluciones obtenidas por el algoritmo implementado son superiores en términos de costo del modelo de asignación a las obtenidas por el Algoritmo Genético y el método Q-Learning propuestos en (Rosa, 2006) y a las obtenidas siguiendo la estrategia greedy expuesta en este capítulo.

CONCLUSIONES

A continuación se resumen las principales resultados de la presente investigación:

1. Se obtuvo un modelo híbrido mediante el empleo del método Q-Learning para obtener parte de la población inicial de un Algoritmo Genético que da solución al problema de asignación de fragmentos en el diseño de bases de datos distribuidas, considerando la replicación. Se comprobó a través de los experimentos realizados que las soluciones obtenidas por el algoritmo implementado son superiores en términos de costo del modelo de asignación a las obtenidas mediante otros métodos heurísticos que solucionan el problema de asignación y a las de otros modelos reportados en la bibliografía.
2. Se le adiciona a la herramienta ALLOCATOR el método metaheurístico híbrido propuesto que considera la replicación de una forma computacionalmente tratable para dar solución a un problema de decisión de la clase NP-Completo.
3. La herramienta elaborada permite materializar diseños de distribución hasta el nivel físico a través de la generación de secuencias de comandos correspondientes al modelo de réplicas para el manejo distribuido de datos soportado por Microsoft SQL Server.

RECOMENDACIONES

1. Continuar la validación de la herramienta mediante la solución de problemas reales y otros reportados en la bibliografía.
2. Obtener un mecanismo automático que ajuste la cantidad de generaciones y el tamaño de la población en el Algoritmo Genético, de acuerdo a las características particulares de la instancia del problema. Estos parámetros influyen en gran medida en el desempeño y la fortaleza del Algoritmo Genético. La asignación de un valor independientemente del tamaño de la instancia podría ocasionar convergencia prematura o representar pobremente el espacio de soluciones si son valores demasiado pequeños; por otro lado, si son demasiado grandes, seguramente se desperdiciará recursos computacionales sin obtener mejoras apreciables. Por eso es conveniente obtener un mecanismo que ajuste tales valores a las circunstancias particulares del problema.

BIBLIOGRAFÍA

Abe, N., Biermann, A.W. y Long, P.M. (2003) Reinforcement Learning with Immediate Rewards and Linear Hypotheses. *Algorithmica,* vol. 37**,** num. 4, 263-293.

Aekaterinidis, I. y Triantafillou, P. (2005) Internet Scale String Attribute Publish/Subscribe Data Networks. *Proceedings of the International Conference on Information and Knowledge Management (CIKM'05).* Bremen, Germany, ACM.

Akal, F., Türker, C., Schek, H.-J., Breitbart, Y., Grabs, T. y Veen, L. (2005) Fine-grained lazy replication and scheduling with freshness and correctness guarantees. *Proceedings of the 31st VLDB Conference.* Trondheim, Norway, Swiss Federal Institute of Technology Zurich.

Akal, F., Türker, C., Schek, H.-J., Grabs, T. y Breitbart, Y. (2004) Fine-grained lazy replication with strict freshness and correctness guarantees. Zurich, Swiss Federal Institute of Technology.

Álvarez, W.A., Rodriguez, A. y García, C.E. (2006) ERECASE v.2.0. Una herramienta para el diseño conceptual de bases de datos con validación estructural. *Departamento de Ciencia de la Computación.* Santa Clara, Cuba, Universidad Central "Marta Abreu" de Las Villas.

Amza, C., Cox, A.L. y Zwaenepoel, W. (2003) Distributed Versioning: Consistent Replication for Scaling Back-End Databases of Dynamic Content Web Sites. *Middleware 2003, ACM/IFIP/USENIX International Middleware Conference.* Rio de Janeiro, Brazil, Springer.

Anderson, T.A., Breitbart, Y., Korth, H.F. y Wool, A. (1998) Replication, Consistency, and Practicality: Are These Mutually Exclusive? En: L.M. Haas y A. Tiwary (Eds.) *Proceedings of the ACM SIGMOD International Conference on Management of Data 1998.* Seattle, WA, USA, ACM Press.

Apers, P.M.G. (1988) Data Allocation in Distributed Database Systems. *ACM Trans. Database Syst.,* vol. 13**,** num. 3, 263-304.

Awerbuch, B., Bartal, Y. y Fiat, A. (2003) Competitive distributed file allocation. *Inf. Comput.,* vol. 185**,** num. 1, 1-40.

Baião, F.A., Mattoso, M., Shavlik, J.W. y Zaverucha, G. (2003) Applying Theory Revision to the Design of Distributed Databases. En: T. Horváth y A.

Yamamoto (Eds.) *Proceedings of the 13th International Conference on Inductive Logic Programming ILP 2003, LNAI 2835.* Szeged, Hungary, Springer.

Baião, F.A., Mattoso, M. y Zaverucha, G. (1998) Towards an Inductive Design of Distributed Object Oriented Databases. *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems.* New York City, NY, USA, IEEE Computer Society.

Baião, F.A., Mattoso, M. y Zaverucha, G. (2004) A Distribution Design Methodology for Object DBMS. *Distributed and Parallel Databases,* vol. 16**,** num. 1, 45-90.

Belokosztolszki, A., Eyers, D.M., Pietzuch, P.R., Bacon, J. y Moody, K. (2003) Role-based access control for publish/subscribe middleware architectures. En: H.-A. Jacobsen (Ed.) *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003 (in conjunction with SIGMOD/PODS).* San Diego, CA, USA, ACM.

Bellatreche, L., Karlapalem, K. y Li, Q. (1998) Complex Methods and Class Allocation in Distributed Object-Oriented Database Systems. *International Conference on Object Oriented Information Systems (OOIS 1998).* Paris, France.

Bellatreche, L., Karlapalem, K. y Simonet, A. (2000) Algorithms and support for horizontal class partitioning in object-oriented databases. *Distributed and Parallel Databases,* vol. 8**,** num. 2, 155-179.

Berenson, H., Bernstein, P.A., Gray, J., Melton, J., O'Neil, E.J. y O'Neil, P.E. (1995) A Critique of ANSI SQL Isolation Levels. En: M.J. Carey y D.A. Schneider (Eds.) *Proceedings of the ACM SIGMOD International Conference on Management of Data 1995* San Jose, CA, USA, ACM Press.

Bernstein, P.A., Hadzilacos, V. y Goodman, N. (1987) *Concurrency Control and Recovery in Database Systems*, Addison-Wesley.

Bhalla, S. y Hasegawa, M. (2005) Parallelizing serializable transactions within distributed real-time database systems *Proceedings of the International Conference on Embedded and Ubiquitous Computing (EUC 2005). Lecture Notes in Computer Science.* Springer.

Birman, K.P. (1997) Building Secure and Reliable Network Applications. En: T. Masuda, Y. Masunaga y M. Tsukamoto (Eds.) *Worldwide Computing and Its*

Applications, International Conference, WWCA '97. Tsukuba, Japan, Springer.

Böhm, K., Grabs, T., Röhm, U. y Schek, H.-J. (2000) Evaluating the Coordination Overhead of Replica Maintenance in a Cluster of Databases. En: A. Bode, T. Ludwig II, W. Karl y R. Wismüller (Eds.) *Proceedings of the 6th International Conference on Parallel Processing (EuroPar 2000)* Munich, Germany, Springer.

Brahmadathan, K. y Ramarao, K.V.S. (1992) On the design of replicated databases. *Inf. Sci.,* vol. 65**,** num. 1-2, 173-187.

Breitbart, Y. y Korth, H.F. (1997) Replication and Consistency: Being Lazy Helps Sometimes. *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* Tucson, Arizona, USA, ACM Press.

Breitbart, Y. y Korth, H.F. (1999) Replication and Consistency in a Distributed Environment. *J. Comput. Syst. Sci.,* vol. 59**,** num. 1, 29-69.

Cacheda, F., Plachouras, V. y Ounis, I. (2004) Performance Analysis of Distributed Architectures to Index One Terabyte of Text. En: S. McDonald y J. Tait (Eds.) *Proceedings of the 26th European Conference on IR Research, ECIR 2004.* Sunderland, UK, Springer.

Cecchet, E., Marguerite, J. y Zwaenepoel, W. (2004) C-JDBC: Flexible Database Clustering Middleware. *Proceedings of the FREENIX Track: 2004 USENIX Annual Technical Conference.* Boston, Marriott Copley Place, Boston, MA, USA, USENIX.

Ceri, S., Martella, G. y Pelagatti, G. (1982) Optimal File Allocation in a Computer Network: a Solution Method Based on the Knapsack Problem. *Computer Networks,* vol. 6**,** num. 5, 345-357.

Ceri, S., Navathe, S.B. y Wiederhold, G. (1983) Distribution Design of Logical Database Schemas. *IEEE Trans. Software Eng.,* vol. 9**,** num. 4, 487-504.

Ceri, S. y Pelagatti, G. (1984) *Distributed Databases: Principles and Systems*, McGraw-Hill Book Company.

Ceri, S. y Pernici, B. (1985) *DATAID-D: Methodology for Distributed Database Design,* North-Holland.

Ceri, S., Pernici, B. y Wiederhold, G. (1987) Distributed Database Design Methodologies. *IEEE Database Eng. Bull.,* vol. 75**,** num. 5, 533-546.

Cichosz, P. (1995) Truncating Temporal Differences: On the Efficient Implementation of TD(lambda) for Reinforcement Learning. *J. Artif. Intell. Res. (JAIR),* vol. 2**,** num. 1, 287-318.

Colunga, A. A., Muñoz, L.C., Ramírez, M.L. (2001) *Agente Inteligente de Aprendizaje Reforzado Aplicado al Diseño de Bases de Datos Distribuidas*". Tesis de licenciatura dirigida por JJ. González. Instituto Tecnológico de Ciudad Madero, México.

Cornell, D.W. y Yu, P.S. (1989) On Optimal Site Assignment for Relations in the Distributed Database Environment. *IEEE Trans. Software Eng.,* vol. 15**,** num. 8, 1004-1009.

Coulon, C., Pacitti, E. y Valduriez, P. (2005a) Consistency Management for Partial Replication in a High Performance Database Cluster. *IEEE International Conference on Parallel and Distributed Systems (ICPADS2005).* Fukuoka, Japan.

Coulon, C., Pacitti, E. y Valduriez, P. (2005b) Optimistic Preventive Replication in a Database Cluster. *Proceedings of the 21èmes Journées Bases de Données Avancées (BDA).* France, INRIA.

Coulon, C., Pacitti, E. y Valduriez, P. (2005c) Scaling up the preventive replication of autonomous databases in cluster systems. *International Conference on High Performance Computing of Computational Science (VecPar2004), Lecture Notes in Computer Science 3402.* Valencia, Spain, Springer.

Cruz, F., Baião, F.A., Mattoso, M. y Zaverucha, G. (2002) Towards a Theory Revision Approach for the Vertical Fragmentation of Object Oriented Databases. En: G. Bittencourt y G. Ramalho (Eds.) *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence SBIA 2002.* Porto de Galinhas, Recife, Brazil, Springer.

Cruz, L. (1999) *Automatización del Diseño de la Fragmentación Vertical y Ubicación en Bases de Datos Distribuidas Usando Métodos Heurísticos y Exactos*. Tesis de maestría. Instituto Tecnológico y de Estudios Superiores de Monterrey, pp. 116.

Chakravarthy, S. y Vontella, N. (2004) A Publish/Subscribe Based Architecture of an Alert Server to Support Prioritized and Persistent Alerts. En: R.K. Ghosh y M. Hrushikesha (Eds.) *Proceedings of the First International Conference on Distributed Computing and Internet Technology ICDCIT 2004.* Bhubaneswar, India, Springer.

Chang, P.-Y., Chen, D.-J. y Kavi, K.M. (2001) File Allocation Algorithms to Minimize Data Transmission Time in Distributed Computing Systems. *J. Inf. Sci. Eng.,* vol. 17**,** num. 4, 633-649.

Chang, S.K. y Liu, A.C. (1982) File allocation in a distributed database. *International Journal of Computer Information Sciences,* vol. 11**,** num. 5, 325-340.

Chang, V.N. (2004) Método de Dos Fases para la Asignación de Datos en una Base de Datos Distribuida.

Chaturvedi, A.R. y Roan, J. (1994) Scheduling the allocation of data fragments in a distributed database environment: a machine learning approach. *IEEE Transactions on Engineering Management,* vol. 41**,** num. 2, 194-207.

Chen, X., Chen, Y. y Rao, F. (2003) An efficient spatial publish/subscribe system for intelligent location-based services. En: H.-A. Jacobsen (Ed.) *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003 (in conjunction with SIGMOD/PODS).* San Diego, CA, USA, ACM.

Cheng, C.H., Lee, W.-K. y Wong, K.-F. (2002) A genetic algorithm-based clustering approach for database partitioning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C,* vol. 32**,** num. 3, 215-230.

Chiu, G.-M. y Raghavendra, C.S. (1990) A Model for Optimal Database Allocation in Distributed Computing Systems. *Proceedings of the Conference on Computer Communications, Ninth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM '90.* San Francisco, CA, USA, IEEE.

Choi, S.P.M., Zhang, N.L. y Yeung, D.-Y. (2004) Reinforcement Learning in Episodic Non-stationary Markovian Environments. En: H.R. Arabnia y Y. Mun (Eds.) *Proceedings of the International Conference on Artificial Intelligence, IC-AI '04. Proceedings of the International Conference on*

Machine Learning; Models, Technologies and Applications, MLMTA '04. Las Vegas, Nevada, USA, CSREA Press.

Chundi, P., Rosenkrantz, D.J. y Ravi, S.S. (1996) Deferred Updates and Data Placement in Distributed Databases. En: S.Y.W. Su (Ed.) *Proceedings of the Twelfth International Conference on Data Engineering.* New Orleans, Louisiana, IEEE Computer Society.

Date, C.J. (2000) *Introducción a los Sistemas de Bases de Datos, Séptima edición,* México, Addison-Wesley.

Daudjee, K. y Salem, K. (2004) Lazy Database Replication with Ordering Guarantees. *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004.* Boston, MA, USA, IEEE Computer Society.

Daudjee, K. y Salem, K. (2006) Lazy Database Replication with snapshot isolation. *Proceedings of the 32nd International Conference on Very Large Databases* Seoul, Korea, ACM VLDB Endowment.

Daudpota, N.H. (1998) Five Steps to Construct a Model of Data Allocation for Distributed Database Systems. *J. Intell. Inf. Syst.,* vol. 11**,** num. 2, 153-168.

Dowdy, L.W. y Foster, D.V. (1982) Comparative Models of the File Assignment Problem. *ACM Comput. Surv.,* vol. 14**,** num. 2, 287-313.

Du, J., Alhajj, R. y Barker, K. (2006) Genetic algorithms based approach to database vertical partition. *Journal of Intelligent Information Systems.,* vol. 26**,** num. 2, 167-183.

Eiben, A., Hinterding, R. y Michalewicz, Z. (1999) Parameter control in evolutionary algorithms Evolutionary Computation, *IEEE Transactions*, 124-141.

Elmeleegy, K., Chanda, A., Cox, A.L. y Zwaenepoel, W. (2004) Lazy Asynchronous I/O for Event-Driven Servers. *Proceedings of the General Track: 2004 USENIX Annual Technical Conference.* Boston Marriott Copley Place, Boston, MA, USA, USENIX.

Elnikety, S., Zwaenepoel, W. y Pedone, F. (2005) Database Replication Using Generalized Snapshot Isolation. *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05).* IEEE Computer Society.

Eugster, P.T., Felber, P.A., Guerraoui, R. y Kermarrec, A.-M. (2003) The many faces of publish/subscribe. *ACM Computing Surveys,* vol. 35**,** num. 2, 114-131.

Ezeife, C.I. y Barker, K. (1998) Distributed Object Based Design: Vertical Fragmentation of Classes. *Distributed and Parallel Databases,* vol. 6**,** num. 4, 317-350.

Ezeife, C.I. y Dey, P. (2003) Incremental Horizontal Fragmentation of Database Class Objects. *Proceedings of the 5th International Conference on Enterprise Information Systems ICEIS (1) 2003.* Angers, France.

Fan, R. y Lynch, N.A. (2003a) Brief announcement: efficient replication of large data objects. *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003).* Boston, Massachusetts, USA, ACM.

Fan, R. y Lynch, N.A. (2003b) Efficient Replication of Large Data Objects. En: F.E. Fich (Ed.) *Proceedings of the 17th. International Conference on Distributed Computing DISC 2003.* Sorrento, Italy, Springer.

Fisher, M.L. y Hochbaum, D.S. (1980) Database Location in Computer Networks. *Journal of the ACM,* vol. 27**,** num. 4, 718-735.

Fornaciari, W., Piuri, V., Prestileo, A. y Zaccaria, V. (2001) An Agent-Based Approach to Full Interoperability and Allocation Transparency in Distributed File Systems. En: S. Pierre y R.H. Glitho (Eds.) *Proceedings of the Third International Workshop on Mobile Agents for Telecommunication Applications, MATA 2001.* Montreal, Canada, Springer.

Fu, K., Kaashoek, M.F. y Mazières, D. (2002) Fast and secure distributed read-only file system. *ACM Transactions on Computer Systems,* vol. 20**,** num. 1, 1-24.

Fung, C.-W., Karlapalem, K. y Li, Q. (1997) Cost-Driven Evaluation of Vertical Class Partitioning in Object-Oriented Databases. En: R.W. Topor y K. Tanaka (Eds.) *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA'97).* Melbourne, Australia, World Scientific.

Fung, C.-W., Karlapalem, K. y Li, Q. (2002) An Evaluation of Vertical Class Partitioning for Query Processing in Object-Oriented Databases. *IEEE Trans. Knowl. Data Eng.,* vol. 14**,** num. 5, 1095-1118.

Fung, C.-W., Karlapalem, K. y Li, Q. (2003) Cost-driven vertical class partitioning for methods in object oriented databases. *VLDB Journal,* vol. 12**,** num. 3, 187-210.

Gálvez, D. (1998) Algoritmos Genéticos. Monografía para la especialización de Inteligencia Artificial. Universidad Cooperativa de Colombia.

Gançarski, S., Naacke, H., Pacitti, E. y Valduriez, P. (2002a) Parallel Processing with Autonomous Databases in a Cluster System. En: R. Meersman y Z. Tari (Eds.) *Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE.* Irvine, California, USA, Springer.

Gançarski, S., Naacke, H. y Valduriez, P. (2002b) Load Balancing of Autonomous Applications and Databases in a Cluster System. En: W. Litwin y G. Lévy (Eds.) *Records of the 4th International Meeting on Distributed Data and Structures, (WDAS 2002).* Paris, France, Carleton Scientific.

García, C.E., Rodríguez, A., González, L.M. y Álvarez, W.A. (2005) ERECASE, una herramienta con validación de diagramas entidad relación. *6to. Simposium Iberoamericano de Computación e Informática SICI 2005.* Instituto Tecnológico de Nuevo León, Monterrey, NL, México, Instituto Tecnológico de Nuevo León.

Goel, S., Sharda, H. y Taniar, D. (2003a) Message-Oriented-Middleware in a Distributed Environment. En: B. Thomas, H. Gerhard y U. Herwig (Eds.) *Revised Papers of the Third International Workshop on Innovative Internet Community Systems (IICS 2003).* Leipzig, Germany, Springer.

Goel, S., Sharda, H. y Taniar, D. (2003b) Transaction Management in Distributed Scheduling Environment for High Performance Database Applications. *Proceedings of the 5th International Workshop on Distributed Computing - IWDC 2003.* Kolkata, India, Springer.

Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company.

González, J.J. (2001) Agente Inteligente de Aprendizaje Reforzado aplicado al diseño de bases de datos distribuidas. Departamento de Sistemas y Computación, Instituto Tecnológico Ciudad Madero.

Grabs, T., Böhm, K. y Schek, H.-J. (2004) PowerDB-IR - Scalable Information Retrieval and Storage with a Cluster of Databases. *Knowl. Inf. Syst.,* vol. 6**,** num. 4, 465-505.

Gray, J. (2004) The Revolution in Database System Architecture. En: G. Gottlob, A.A. Benczúr y J. Demetrovics (Eds.) *Proceedings of the 8th East European Conference on Advances in Databases and Information Systems (ADBIS 2004)* Budapest, Hungary, Springer.

Gu, X., Lin, W.J. y Veeravalli, B. (2006) Practically realizable efficient data allocation and replication strategies for distributed databases with buffer constraints. *IEEE Transactions on Parallel and Distributed Systems,* vol. 17**,** num. 9, 1001-1013.

Hababeh, I.O., Bowring, N. y Ramachandran, M. (2003) An integrated strategy for data fragmentation and allocation in a Distributed Database Design. *Proceedings of the International Conference on Information Technology and Natural Science (ICITNS).* Amman, Jordan.

Hababeh, I.O., Bowring, N. y Ramachandran, M. (2004) A Method for Fragment Allocation in Distributed Object Oriented Database Systems. *Proceedings of the 5th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking & Broadcasting (PGNet).* Liverpool, UK.

Hernández, A. (2005) Agente de aprendizaje reforzado aplicado al problema de asignación en bases de datos distribuidas. *Ciencia de la Computación.* Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

Hernández, I., López, R. y Nieto, A. (2000) Automatización de la Evaluación de Algoritmos Genéticos. Tesis de Ing, ITCM, Noviembre 2000.

Hoffer, J.A. (1976) An integer programming formulation of computer data base design problems. *Inf. Sci.,* vol. 11**,** num. 1, 29-48.

Holland, J. (1975) Adaptation in natural and Artificial Systems. The University of Michigan Press.

Holliday, J., Agrawal, D. y Abbadi, A.E. (1999a) Database Replication: If You Must be Lazy, be Consistent. *Proceedings of the Eighteenth Symposium on Reliable Distributed Systems.* Lausanne, Switzerland, IEEE Computer Society.

Holliday, J., Agrawal, D. y Abbadi, A.E. (1999b) The Performance of Database Replication with Group Multicast. *Proccedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing, Digest of Papers: FTCS-29.* Madison, Wisconsin, USA, IEEE Computer Society.

Holliday, J., Steinke, R.C., Agrawal, D. y Abbadi, A.E. (2003) Epidemic Algorithms for Replicated Databases. *IEEE Trans. Knowl. Data Eng.,* vol. 15**,** num. 5, 1218-1238.

Hotek, M. (2002) Database Replication. *SQL Server Magazine. WindowsITPro. Penton Media Inc.*

Huang, Y.-F. y Chen, J.-H. (2001) Fragment Allocation in Distributed Database Design. *Journal of Information Science and Engineering,* vol. 17**,** num. 3, 491-506.

Iida, S., Kuwayama, K., Kanoh, M., Kato, S. y Itoh, H. (2004) A Dynamic Allocation Method of Basis Functions in Reinforcement Learning. En: G.I. Webb y X. Yu (Eds.) *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence AI 2004: Advances in Artificial Intelligence.* Cairns, Australia, Springer.

Irún-Briz, L., Muñoz-Escoí, F.D. y Bernabéu-Aubán, J.M. (2004) The Abortion Rate of Lazy Replication Protocols for Distributed Databases. *Proceedings of the 6th International Conference on Enterprise Information Systems ICEIS 2004.* Porto, Portugal.

Irún-Briz, L., Muñoz-Escoí, F.D., Decker, H. y Bernabéu-Aubán, J.M. (2003) COPLA: A Platform for Eager and Lazy Replication in Networed Databases. *Proceedings of the 5th International Conference on Enterprise Information Systems ICEIS 2003.* Angers, France.

Jiménez-Peris, R., Patiño-Martínez, M., Alonso, G. y Kemme, B. (2003) Are Quorums an Alternative for Data Replication? *ACM Trans. Database Syst.,* vol. 28**,** num. 3, 257-294.

Jiménez-Peris, R., Patiño-Martínez, M., Kemme, B. y Alonso, G. (2002) Improving the Scalability of Fault-Tolerant Database Clusters: Early results. *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02).* Vienna, Austria, IEEE Computer Society.

Johansson, J.M., March, S.T. y Naumann, J.D. (2000) The effects of parallel processing on update response time in distributed database design. En: S. Ang, H. Krcmar, W.J. Orlikowski, P. Weill y J.I. DeGross (Eds.) *Proceedings of the Twenty-First International Conference on Information Systems ICIS.* Brisbane, Australia, ACM.

Karlapalem, K., Navathe, S.B. y Morsi, M.M.A. (1994) Issues in distribution design of object-oriented databases. En: M.T. Özsu, U. Dayal y P. Valduriez (Eds.) *Distributed Object Management.* Morgan Kaufmann.

Karlapalem, K. y Pun, N.M. (1997) Query-Driven Data Allocation Algorithms for Distributed Database Systems. En: A. Hameurlain y A.M. Tjoa (Eds.) *Proceedings of the 8th International Conference on Database and Expert Systems Applications, DEXA '97.* Toulouse, France, Springer.

Kemme, B. y Alonso, G. (2000) A new approach to developing and implementing eager database replication protocols. *ACM Trans. Database Syst.,* vol. 25**,** num. 3, 333-379.

Le Pape, C., Gançarski, S. y Valduriez, P. (2003) Trading Freshness for Performance in a Cluster of Replicated Databases. En: R. Meersman y Z. Tari (Eds.) *Proceedings of the OTM Confederated International Workshops On The Move to Meaningful Internet Systems 2003: HCI-SWWA, IPW, JTRES, WORM, WMS, and WRSM 2003.* Catania, Sicily, Italy, Springer.

Le Pape, C., Gançarski, S. y Valduriez, P. (2004) Refresco: Improving Query Performance Through Freshness Control in a Database Cluster. En: R. Meersman y Z. Tari (Eds.) *Proceedings of the OTM Confederated International Conference On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE.* Agia Napa, Cyprus, Springer.

Lee, H., Park, Y.-K., Jang, G. y Huh, S.-Y. (2000) Designing a distributed database on a local area network: a methodology and decision support system. *Information & Software Technology,* vol. 42**,** num. 3, 171-184.

Lee, J.W. y Baik, D.K. (2004) Database allocation modeling for optimal design of distributed systems. *IEICE Transactions on Information and Systems,* vol. E87D**,** num. 7, 1795-1804.

Lim, S.J. y Ng, Y.-K. (1997) Vertical Fragmentation and Allocation in Distributed Deductive Database Systems. *Inf. Syst.,* vol. 22**,** num. 1, 1-24.

Lin, W.J. y Veeravalli, B. (2006) An object replication algorithm for real-time distributed databases. *Distributed and Parallel Databases.,* vol. 19**,** num. 2-3, 125-146.

Lin, X. y Orlowska, M.E. (1995) An Integer Linear Programming Approach to Data Allocation with the Minimum Total Communication Cost in Distributed Database Systems. *Information Sciences,* vol. 85**,** num. 1-3, 1-10.

Lin, X., Orlowska, M.E. y Zhang, Y. (1993) On Data Allocation with the Minimum Overall Communication Costs in Distributed Database Design. En: O. Abou-Rabia, C.K. Chang y W.W. Koczkodaj (Eds.) *Proceedings of the Fifth International Conference on Computing and Information - ICCI'93.* Sudbury, Ontario, Canada, IEEE Computer Society.

Lin, Y., Kemme, B., Patiño-Martínez, M. y Jiménez-Peris, R. (2005) Middleware based Data Replication providing Snapshot Isolation. *Proceedings of the ACM SIGMOD International Conference on Management of Data.* Baltimore, Maryland, USA, ACM Press.

Ma, H. (2003) Distribution design in object oriented databases. Master's Thesis. Massey, Massey University.

Ma, H. y Schewe, K.-D. (2004) A heuristic approach to horizontal fragmentation in object oriented databases. En: J. Barzdins (Ed.) *Proceedings of the 2004 Baltic Conference on Databases and Information Systems.* Riga, Latvia, Faculty of Computer Science and Information Technology, Riga Technical University.

Ma, H., Schewe, K.-D. y Wang, Q. (2005) Distribution design for higher-order data models. Palmerston North, New Zealand, Massey University, Department of Information Systems.

Ma, H., Schewe, K.-D. y Wang, Q. (2006) A Heuristic Approach to Cost-Efficient Fragmentation and Allocation of Complex Value Databases. En: G. Dobbie y J. Bailey (Eds.) *The 17th Australasian Database Conference (ADC2006)* Hobart, Australia, ACM International Conference Archive Proceeding Series vol. 170. Australian Computer Society Inc.

Malinowski, E. y Chakravarthy, S. (1997) Fragmentation techniques for distributing object-oriented databases. En: D.W. Embley y R.C. Goldstein (Eds.) *Lecture Notes in Computer Science. Conceptual Modeling - ER '97.* Springer.

March, S.T. y Rho, S. (1995) Allocating Data and Operations to Nodes in Distributed Database Design. *IEEE Trans. Knowl. Data Eng.,* vol. 7**,** num. 2, 305-317.

Mariano, C. y Morales, E.F. (2001) DQL: A New Updating Strategy for Reinforcement Learning Based on Q-Learning. En: L. De Raedt y P.A. Flach (Eds.) *Proceedings of the 12th European Conference on Machine Learning - EMCL 2001.* Freiburg, Germany, Springer.

Maskarinec, M. y Neumann, K. (2003) Heterogeneous Cost Functions and a Partitioning Algorithm for a Distributed Deductive Database. En: H.R. Arabnia, R. Joshua y Y. Mun (Eds.) *Proceedings of the International Conference on Artificial Intelligence IC-AI '03.* Las Vegas, Nevada, USA, CSREA Press.

Mei, A., Mancini, L.V. y Jajodia, S. (2003) Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems. *IEEE Trans. Parallel Distrib. Syst.,* vol. 14**,** num. 9, 885-896.

Mei, H. y Sheng, O. (1992) A Semantic Based Methodology for Integrated Computer-Aided Distributed Database Design. *The 25th Hawaii International Conference on System Sciences.*

Miklós, Z. (2002) Towards an Access Control Mechanism for Wide-Area Publish/Subscribe Systems. *Proceedings of the 22nd. International Conference on Distributed Computing Systems, Workshops (ICDCSW '02)* Vienna, Austria, IEEE Computer Society.

Morales, E.F. y Sammut, C. (2004) Learning to fly by combining reinforcement learning with behavioural cloning. En: C.E. Brodley (Ed.) *Proceedings of the Twenty-first International Conference on Machine Learning (ICML 2004).* Banff, Alberta, Canada, ACM.

Muro, S., Ibaraki, T., Miyajima, H. y Hasegawa, T. (1985) Evaluation of the File Redundancy in Distributed Database Systems. *IEEE Trans. Software Eng.,* vol. 11**,** num. 2, 199-205.

Navathe, S.B., Karlapalem, K. y Ra, M. (1995) A mixed fragmentation methodology for initial distributed database design. Atlanta, GA, College of Computing, Georgia Institute of Technology.

Nejdl, W., Siberski, W., Simon, B. y Tane, J. (2002) Towards a Modification Exchange Language for Distributed RDF Repositories. En: I. Horrocks y J.A. Hendler (Eds.) *Proceedings of the First International Semantic Web Conference ISWC 2002.* Sardinia, Italy, Springer.

O, J., Lee, J.W., Lee, J. y Zhang, B.-T. (2004) Dynamic Asset Allocation Exploiting Predictors in Reinforcement Learning Framework. En: J.-F. Boulicaut, F. Esposito, F. Giannotti y D. Pedreschi (Eds.) *Proceedings of the 15th European Conference on Machine Learning: ECML 2004.* Pisa, Italy, Springer.

Őzsu, M.T. y Valduriez, P. (1991) *Principles of Distributed Database Systems*, Prentice-Hall.

Őzsu, M.T. y Valduriez, P. (1999) *Principles of Distributed Database Systems, Second Edition*, Prentice-Hall.

Pacitti, E., Minet, P. y Simon, E. (1999a) Fast Algorithms for Maintaining Replica Consistency in Lazy Master Replicated Databases. En: M.P. Atkinson, M.E. Orlowska, P. Valduriez, S.B. Zdonik y M.L. Brodie (Eds.) *Proceedings of 25th International Conference on Very Large Data Bases VLDB'99.* Edinburgh, Scotland, UK, Morgan Kaufmann.

Pacitti, E., Minet, P. y Simon, E. (1999b) Maintaining Replica Consistency in Lazy Master Replicated Databases. En: C. Collet (Ed.) *Proceedings of the 15èmes Journées Bases de Données Avancées BDA'99.* Bordeaux, France.

Pacitti, E., Minet, P. y Simon, E. (2001) Replica Consistency in Lazy Master Replicated Databases. *Distributed and Parallel Databases,* vol. 9**,** num. 3, 237-267.

Pacitti, E., Őzsu, M.T. y Coulon, C. (2003) Preventive Multi-master Replication in a Cluster of Autonomous Databases. *Proceedings of the 9th International Conference on Parallel Processing (EuroPar 2003)* Springer.

Pacitti, E., Őzsu, M.T., Coulon, C. y Özsu, M.T. (2005) Preventive Replication in a Database Cluster. *Distributed and Parallel Databases,* vol. 17**,** num. 3, 36.

Pacitti, E. y Simon, E. (2000) Update Propagation Strategies to Improve Freshness in Lazy Master Replicated Databases. *VLDB J.,* vol. 8**,** num. 3-4, 305-318.

Park, S.-J. y Baik, D.-K. (1997) A data allocation considering data availability in distributed database systems. *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS '97).* Seoul, Korea, IEEE Computer Society.

Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B. y Alonso, G. (2000) Scalable Replication in Database Clusters. En: M. Herlihy (Ed.) *Distributed*

Computing, 14th International Conference DISC 2000. Toledo, Spain, Springer.

Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B. y Alonso, G. (2005) MIDDLE-R: Consistent Database Replication at the Middleware Level. *ACM Transactions on Computer Systems,* vol. 23**,** num. 4, 375–423.

Paul, T.K. y Iba, H. (2003) Reinforcement Learning Estimation of Distribution Algorithm. En: E. Cantú-Paz, J.A. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R.K. Standish, G. Kendall, S.W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M.A. Potter, A.C. Schultz, K.A. Dowsland, N. Jonoska y J.F. Miller (Eds.) *Proceedings of the Genetic and Evolutionary Computation Conference- GECCO 2003.* Chicago, IL, USA, Springer.

Pedone, F., Guerraoui, R. y Schiper, A. (1998) Exploiting Atomic Broadcast in Replicated Databases. En: D.J. Pritchard y J. Reeve (Eds.) *Proceedings of the 4th International Conference on Parallel Processing (EuroPar 1998).* Southampton, UK Springer.

Pedone, F., Wiesmann, M., Schiper, A., Kemme, B. y Alonso, G. (2000) Understanding Replication in Databases and Distributed Systems. *Proceedings of the 20th International Conference on Distributed Computing Systems* Taipei, Taiwan, IEEE Computer Society.

Pérez, A. (1996) Una introducción a la computación evolutiva.

Pérez, J. (1999) Integración de la Fragmentación Vertical y Ubicación en el Diseño Adaptativo de Bases de Datos Distribuidas. Morelos, Mexico, Instituto Tecnológico de Estudios Superiores de Monterrey ITESM.

Pérez, J. y Pazos, R.A. (1997) Fragmentación, Ubicación y Reubicación Dinámica de Datos en Bases de Datos Distribuidas. *Actas de I Jornadas de Investigación y Docencia en Bases de Datos.* Getafe, Madrid, Spain, Universidad Carlos III de Madrid.

Pérez, J., Pazos, R.A., Fraire, H.J., Cruz, L. y Pecero, J.E. (2003a) Adaptive Allocation of Data-Objects in the Web Using Neural Networks. En: C. Amedeo y T. Franco (Eds.) *Proceedings of the 8th Congress of the Italian Association for Artificial Intelligence, AI*IA 2003: Advances in Artificial Intelligence* Pisa, Italy, Springer.

Pérez, J., Pazos, R.A., Frausto-Solis, J., Reyes, G., Santaolaya, R., Fraire, H.J. y Cruz, L. (2005) An approach for solving very large scale instances of the design distribution problem for distributed database systems. *Proceedings of the 4th International School and Symposium on Advanced Distributed Systems (ISSADS2005). Lecture Notes in Computer Science.* Springer.

Pérez, J., Pazos, R.A., Frausto-Solís, J., Rodríguez, G., Cruz, L., Mora, G. y Fraire, H.J. (2004a) Self-Tuning Mechanism for Genetic Algorithms Parameters, an Application to Data-Object Allocation in the Web. En: L. Antonio, L.G. Marina, K. Vipin y M. Youngsong (Eds.) *Proceedings of the International Conference on Computational Science and Its Applications - ICCSA 2004, Part IV.* Assisi, Italy, Springer.

Pérez, J., Pazos, R.A., Mora, G., Castilla, G., Martínez, J.A., Landero, V., Fraire, H.J. y González, J.J. (2004b) Dynamic Allocation of Data-Objects in the Web, Using Self-tuning Genetic Algorithms. En: A.L.C. Bazzan y L. Sofiane (Eds.) *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence-SBIA 2004, Advances in Artificial Intelligence* São Luis, Maranhão, Brazil, Springer.

Pérez, J., Pazos, R.A., Romero, D., Santaolaya-Salgado, R., Rodríguez, G. y Sosa-Sosa, V.J. (2003b) Adaptive and Scalable Allocation of Data-Objects in the Web. En: K. Vipin, M.L. Gavrilova, T. Chih Jeng Kenneth y P. L'Ecuyer (Eds.) *Proceedings of the International Conference on Computational Science and Its Applications - ICCSA 2003, Part I.* Montreal, Canada, Springer.

Pérez, J., Pazos, R.A., Santaolaya-Salgado, R., Frausto-Solís, J., Rodríguez, G., Cruz, L. y Bravo, M. (2003c) Data-Object Replication, Distribution, and Mobility in Network Environments. En: B. Manfred y A.V. Zamulin (Eds.) *Revised Papers of the 5th International Andrei Ershov Memorial Conference, Perspectives of Systems Informatics, PSI 2003.* Akademgorodok, Novosibirsk, Rusia, Springer.

Pérez, J., Pazos, R.A., Solís, J.F., Rodríguez, G., Cruz, L. y Fraire, H.J. (2004c) Comparison and Selection of Exact and Heuristic Algorithms. En: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.J.K. Tan y O. Gervasi (Eds.)

International Conference on Computational Science and Its Applications - ICCSA 2004. Assisi, Italy, Springer.

Pérez, J., Pazos, R.A., Fraire, H.J, Johnatan E. (2002) Fragmentación Vertical, Ubicación y Reubicación Dinámica en Bases de Datos Distribuidas Aplicando Redes Neuronales Artificiales. Centro Nacional de Investigación y Desarrollo Tecnológico, Mexico.

Pérez, J., Pazos, R.A., Solís, J.F., Romero, D. y Cruz, L. (2000) Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using the Threshold Accepting Algorithm. En: O. Cairó Battistutti, L.E. Sucar y F.J. Cantu (Eds.) *MICAI 2000: Advances in Artificial Intelligence, Mexican International Conference on Artificial Intelligence.* Acapulco, Mexico, Springer.

Pérez, J., Pazos, R.A., Velez, L. y Rodríguez, G. (2002) Automatic Generation of Control Parameters for the Threshold Accepting Algorithm. En: C.A. Coello Coello, A. de Albornoz, L.E. Sucar y O. Cairó Battistutti (Eds.) *MICAI 2002: Advances in Artificial Intelligence, Second Mexican International Conference on Artificial Intelligence.* Merida, Yucatan, Mexico, Springer.

Pettinger, J.E. y Everson, R.M. (2002) Controlling Genetic Algorithms With Reinforcement Learning. En: W.B. Langdon, E. Cantú-Paz, K.E. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke y N. Jonoska (Eds.) *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference.* New York, USA, Morgan Kaufmann.

Raghuram, A., Morgan, T.W. y Julien-Laferriere, P. (1989) A model for determining the optimal topology and database allocation in computer network. *Proceedings of the Eighth Annual International Phoenix Conference on Computers and Communications.* Arizona, USA.

Ram, S. y Marsten, R.E. (1991) A Model for Database Allocation Incorporating a Concurrency Control Mechanism. *IEEE Trans. Knowl. Data Eng.,* vol. 3**,** num. 3, 389-395.

Rodríguez, A., González, L.M., Cabrera, L. y Morell, A. (2002) ERECASE:Una herramienta de ayuda a la modelación de esquemas conceptuales globales. *Actas I Workshop de Bases de Datos, Jornadas Chilenas de Computación*

JCC 2002. 1 ed. Universidad de Atacama, Copiapó, Chile, Cámara Chilena del Libro A.G.

Rodríguez, A. (2007) *Sistema integrado de herramientas de ayuda al diseño de bases de datos en ambientes distribuidos*. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. Santa Clara, Cuba.

Röhm, U., Böhm, K., Schek, H.-J. y Schuldt, H. (2002) FAS - A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components. *Proceedings of the Thirtieth International Conference on VLDB 2002.* Hong Kong, China, Morgan Kaufmann Publishers.

Roosta, S.H. (2005) A new model for distributed database systems. *International Journal of Computer Mathematics,* vol. 82**,** num. 12, 1447-1454.

Rosa, D. (2006) Aplicación de técnicas de Inteligencia Artificial en la solución del problema de ubicación en el diseño de bases de datos distribuidas. *Departamento de Ciencia de la Computación.* Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

Saccà, D. y Wiederhold, G. (1985) Database Partitioning in a Cluster of Processors. *ACM Trans. Database Syst.,* vol. 10**,** num. 1, 29-56.

Savonnet, M. y Terrasse, M.-N. (2001) Fragtique: Applying an OO Database Distribution Strategy to Data Warehouses. En: Y. Kambayashi, W. Winiwarter y M. Arikawa (Eds.) *Proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery DaWaK 2001.* Munich, Germany, Springer.

Savonnet, M., Terrasse, M.-N. y Yétongnon, K. (1999) FRAGTIQUE: An OO Distribution Design Methodology. En: A.L.P. Chen y F.H. Lochovsky (Eds.) *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications (DASFAA).* Hsinchu, Taiwan, IEEE Computer Society.

Savsar, M. y Al-Anzi, F.S. (2006) Reliability of data allocation on a centralized service configuration with distributed servers *Computer Journal,* vol. 49**,** num. 3, 258-267.

Schewe, K.-D. (2002) Fragmentation of object oriented and semi-structured data. En: H.-M. Haav y A. Kalja (Eds.) *Databases and Information Systems II.* Kluwer Academic Publishers.

Sheng, O.R.L. (1989) Database allocation in Ethernet-based local area networks: a queuing analytic approach. *The Twenty-second Annual Hawaii International Conference on System Sciences (HICSS'89).* Maui, Hawaii, USA, IEEE Computer Society.

Shepherd, J., Harangsri, B., Chen, H.L. y Ngu, A.H.H. (1995) A Two-Phase Approach to Data Allocation in Distributed Databases. En: T.W. Ling y Y. Masunaga (Eds.) *Proceedings of the 4th International Conference on Database Systems for Advanced Applications (DASFAA).* Singapore, World Scientific.

Sun, W., Shu, J. y Zheng, W. (2004) Dynamic File Allocation in Storage Area Networks with Neural Network Prediction. En: F. Yin, J. Wang y C. Guo (Eds.) *Proceedings of the International Symposium on Neural Networks (ISNN 2004).* Dalian, China, Springer.

Sutton, R. y Barto, A. (1998) *Reinforcement Learning: An Introduction,* Cambridge, MA, USA, MIT Press.

Tamhankar, A.M. y Ram, S. (1998) Database fragmentation and allocation: an integrated methodology and case study. *IEEE Transactions on Systems, Man, and Cybernetic Part A,* vol. 28**,** num. 3, 288-305.

Tsai, H.-P., Huang, J.-L., Chao, C.-M., Chen, M.-S. y Liao, C. (2002) SIFA: A Scalable File System with Intelligent File Allocation. *Proceedings of the 26th International Computer Software and Applications Conference (COMPSAC 2002).* Oxford, England, IEEE Computer Society.

Ulusoy, Ö. (2001a) Data Replication and Availability. En: K.-y. Lam y T.-W. Kuo (Eds.) *Real-Time Database Systems: Architecture and Techniques.*, Kluwer Academic Publishers.

Ulusoy, Ö. (2001b) Distributed Concurrency Control. En: K.-y. Lam y T.-W. Kuo (Eds.) *Real-Time Database Systems: Architecture and Techniques.* Kluwer Academic Publishers.

Valduriez, P., Pacitti, E. y Coulon, C. (2005) Large-scale Experimentation with Preventive Replication in a Database Cluster. *DDIR 2005.* France, INRIA.

Vélez, A.G. (1997) Ubicación Óptima de Datos en Aplicaciones de Base de Datos Distribuidas. Cuernavaca, México, CENIDET.

Vélez, L.P. (2000) Esquema de Enfriamiento Adaptativo para el Algoritmo de Aceptación por Umbral Aplicado al Diseño de Base de Datos Distribuidas. León, Mexico, Instituto Tecnológico de León.

Watkins, C. (1999) Learning from Delayed Rewards. Cambridge, UK, University of Cambridge, King's College.

Wolfson, O. y Jajodia, S. (1992a) An Algorithm for Dynamic Data Distribution. En: J.-F. Pâris y H. Garcia-Molina (Eds.) *Proceedings of the Second Workshop on the Management of Replicated Data.* Monterey, CA, USA, IEEE Computer Society Press.

Wolfson, O. y Jajodia, S. (1992b) Distributed Algorithms for Dynamic Replication of Data. *Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* San Diego, CA, USA, ACM Press.

Wolfson, O. y Jajodia, S. (1995) An Algorithm for Dynamic Data Allocation in Distributed Systems. *Inf. Process. Lett.,* vol. 53**,** num. 2, 113-119.

Wolfson, O., Jajodia, S. y Huang, Y. (1997) An Adaptive Data Replication Algorithm. *ACM Trans. Database Syst.,* vol. 22**,** num. 2, 255-314.

Wu, S. y Kemme, B. (2005) Postgres-R(SI): Combining Replica Control with Concurrency Control based on Snapshot Isolation. *IEEE International Conference on Data Engineering (ICDE 2005).* Tokyo, Japan.

Yoshida, M., Mizumachi, K., Wakino, A., Oyake, I. y Matsushita, Y. (1985) Time and Cost Evaluation Schemes of Multiple Copies of Data in Distributet Database Systems. *IEEE Trans. Software Eng.,* vol. 11**,** num. 9, 954-959.

Anexo 1. Descripción del catálogo del asistente

Tabla: LogicalFragments — Fragmentos lógicos de la BD

Atributo	Tipo	Descripción
IdLogFragm	Auto numérico	Identificador del fragmento lógico
IdOperation	Integer	Identificador de la operación de fragmentación
FragmentName	Text[25]	Nombre del fragmento lógico
Cardinality	Integer	Cardinalidad del fragmento lógico

Tabla: PhysicalFragments — Fragmentos físicos de la BD

Atributo	Tipo	Descripción
IdPhysicalFragment	Auto numérico	Identificador del fragmento físico
IdLogFragm	Integer	Identificador del fragmento lógico
SiteName	Text[25]	Nombre del sitio donde fue ubicado
PhysicalFragmName	Text[25]	Nombre del fragmento físico

Tabla: Sites — Sitos de la red

Atributo	Tipo	Descripción
SiteName	Text[25]	Nombre del sitio
IP	Text[20]	IP del sitio
TotalDiskSpace	Double	Capacidad de disco duro
FreeDiskSpace	Double	Espacio libre de disco duro
ReadAccessTime	Double	Costo de procesar una unidad de trabajo
WriteAccessTime	Double	Costo de almacenar un bloque de datos
MaxAnswerTime	Double	Tiempo de respuesta máximo admisible

Tabla: GlobalAttributes — Atributos globales de las relaciones

Atributo	Tipo	Descripción
IdAtt	Auto numérico	Identificador del atributo
IdType	Integer	Identificador del tipo del atributo
SchemaName	Text[20]	Nombre del esquema al que pertenece
AttName	Text[25]	Nombre del atributo
Pkey	Yes/No	Si es llave primaria
Fkey	Yes/No	Si es llave foránea
AttSize	Integer	Longitud del atributo

Tabla: Applications — Aplicaciones

Atributo	Tipo	Descripción
IdApp	Auto numérico	Identificador de la aplicación
MaxAnswerTime	Double	Tiempo de respuesta máximo admisible
ExecuteTime	Double	Tiempo de ejecución

Tabla: FragmentAttributes — Atributos de los fragmentos

Atributo	Tipo	Descripción
IdLogFragm	Integer	Identificador fragmento lógico
IdGlobalAtt	Integer	Identificador del atributo global

Tabla: App_LogicalFragment — Caracterización de las aplicaciones

Atributo	Tipo	Descripción
IdApp	Integer	Identificador de la aplicación
IdLogfrag	Integer	Identificador del fragmento lógico
ReadAccessCount	Integer	Accesos de lectura que la aplicación realiza sobre el fragmento durante su ejecución
WriteAccessCount	Integer	Accesos de actualización que la aplicación realiza sobre el fragmento durante su ejecución
Selectivity	Integer	Selectividad del fragmento lógico respecto a la aplicación

Tabla: AccessTime — Costo de comunicación entre los sitios

Atributo	Tipo	Descripción
FromStation	Text[25]	Nombre del primer sitio
ToStation	Text[25]	Nombre del segundo sitio
AverageTime	Double	Tiempo de acceso entre los sitios

Tabla: App_Site — Muestra los sitios donde se originan las aplicaciones

Atributo	Tipo	Descripción
IdApp	Integer	Identificador de la aplicación
SiteName	Text[25]	Nombre del sitio de origen de la aplicación
ActivationFreq	Integer	Frecuencia de activación de la aplicación en el sitio

Tabla: DataTypes — Tipos de datos de SQL Server

Atributo	Tipo	Descripción
IdType	Auto numérico	Identificador del tipo de dato
Type	Text[30]	Nombre del tipo de dato

Tabla: Schemata — Esquemas lógicos

Atributo	Tipo	Descripción
SchemataName	Text[20]	Nombre del esquema
Cardinality	Integer	Cardinalidad del esquema

Tabla: App_Attributes — Tipos de acceso de las aplicaciones sobre los atributos

Atributo	Tipo	Descripción
IdApp	Integer	Identificador de la aplicación
IdAtt	Integer	Identificador del atributo global
OpRead	boolean	Si la aplicación lee el atributo global
OpWrite	bolean	Si la aplicación actualiza el atributo global

Tabla: Links — Relaciones entre llave foránea y primaria

Atributo	Tipo	Descripción
IdAtt	Integer	Identificador del atributo que es llave foránea
IdAttOwner	Integer	Identificador del atributo que es llave primaria
ExistDep	boolean	Si existe dependencia de existencia

Tabla: Operations — Operaciones que se realizan sobre los esquemas

Atributo	Tipo	Descripción
IdOperation	Integer	Identificador del atributo que es llave foránea
SchemaName	Text[20]	Nombre del esquema al que se le aplica la operación de fragmentación
IdOpType	Integer	Identificador del tipo de operación
Expresión	Text[20]	Expresión asociada a la operación

Tabla: OperationType — Tipos de operaciones

Atributo	Tipo	Descripción
IdOpType	Integer	Identificador del tipo de operación
OperationType	Text[20]	Tipo de operación

Anexo 2. Interfaz visual del asistente

La figura 1 muestra la presentación del asistente.

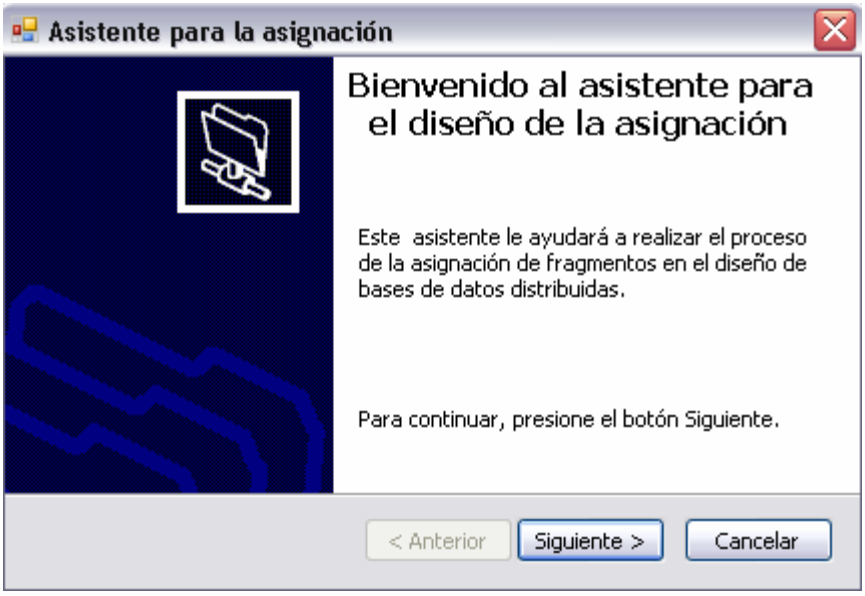


Figura 1. Presentación del asistente.

En la figura 2 se muestra la ventana donde el diseñador debe escoger el algoritmo que se utilizará para resolver el problema de asignación de fragmentos. Una vez seleccionado el algoritmo se pasa a la ventana donde se ingresa la cantidad de accesos de lectura, de actualización y la selectividad de las aplicaciones sobre los fragmentos, figura 3.

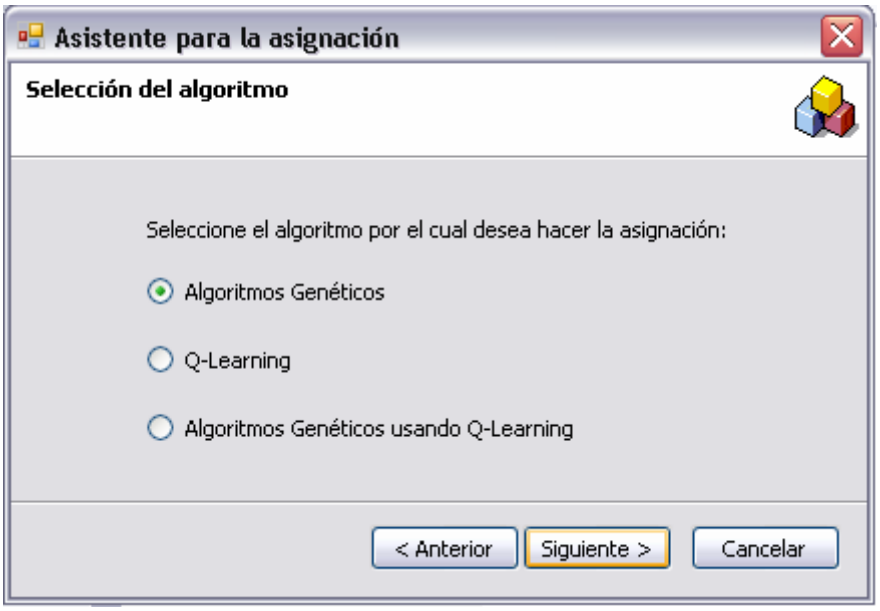


Figura 2. Selección del algoritmo.

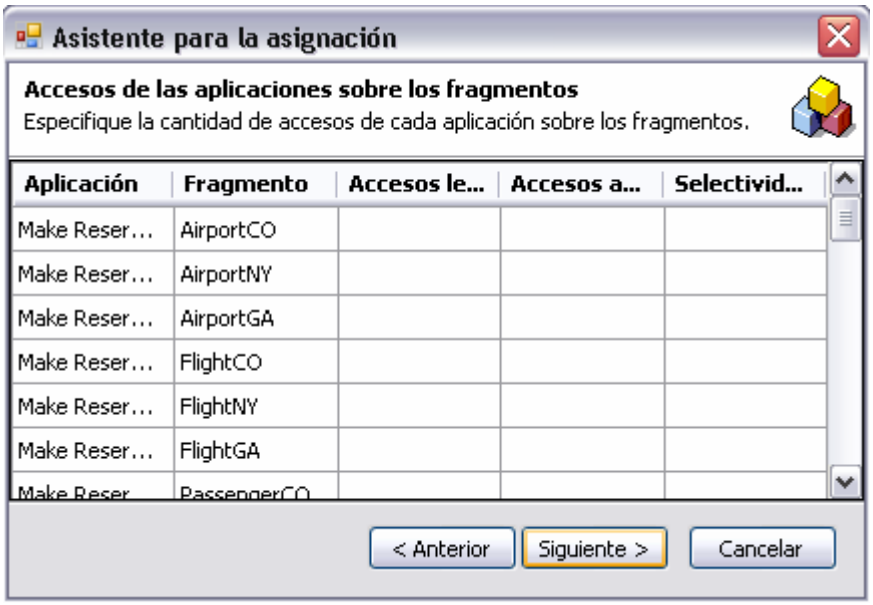


Figura 3. Accesos de lectura, de actualización y la selectividad de las aplicaciones sobre los fragmentos.

Al hacer clic en el botón “Siguiente”, el asistente llevará a cabo el proceso de asignación utilizando el algoritmo seleccionado. En la ventana siguiente se mostrará un texto que indica si el algoritmo encontró (figura 4), o no (figura 5), una solución al problema de asignación que cumpla con las restricciones del modelo.

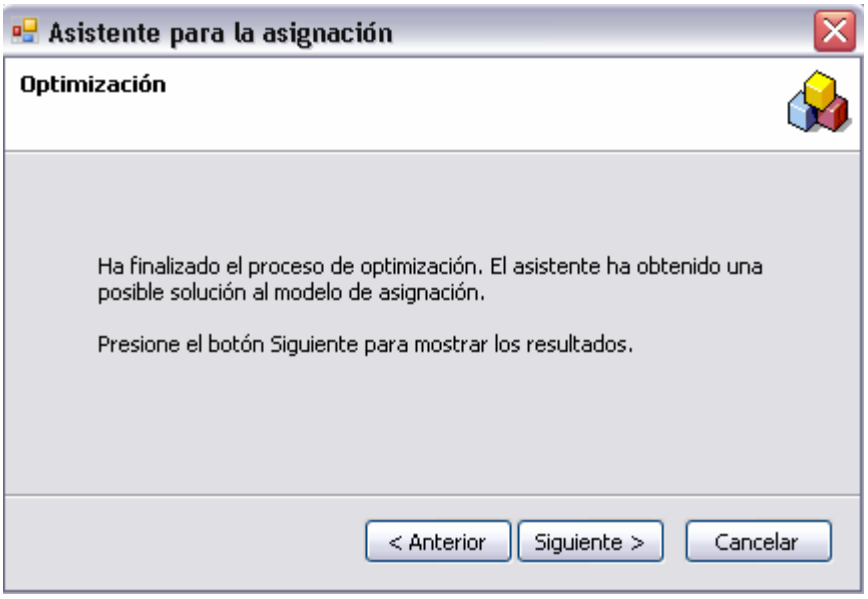


Figura 4. Solución encontrada.

La ventana de la figura 4 se muestra en caso de que se haya encontrado una posible solución al modelo de asignación. Al hacer clic en el botón “Siguiente” se mostrará una ventana con los resultados de la asignación (figura 6).

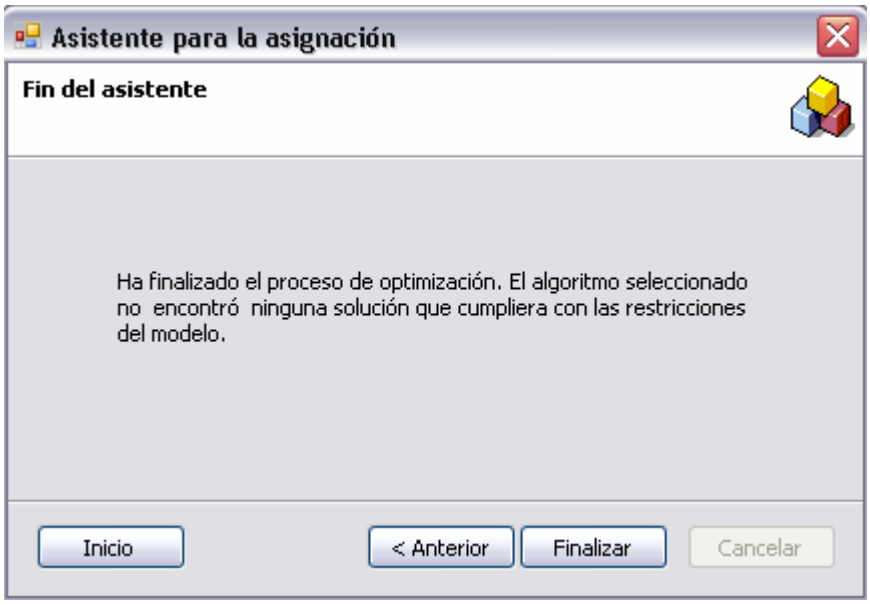


Figura 5. Solución no encontrada.

La ventana de la figura 5 se muestra en caso de no hallarse una solución que cumpla con las restricciones del modelo.

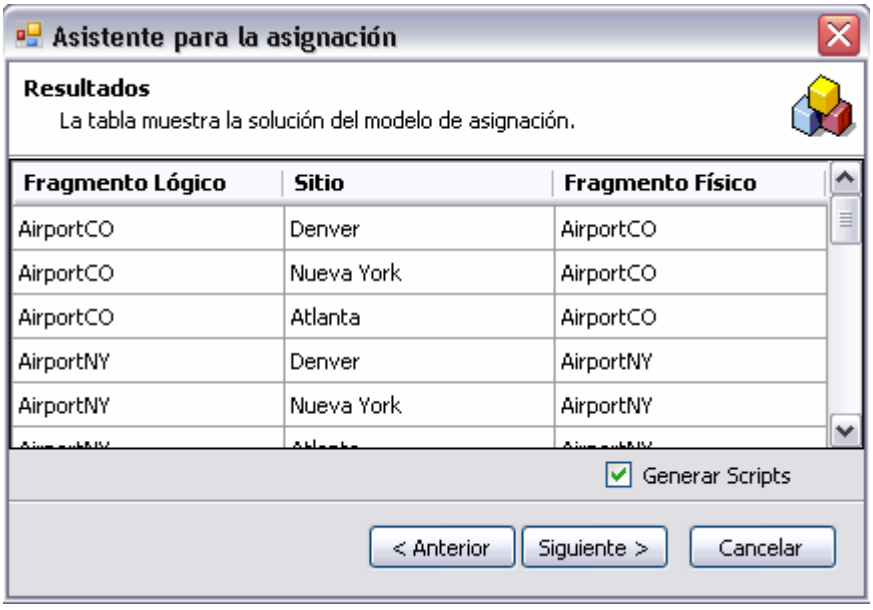


Figura 6. Resultados de la asignación.

En la ventana de la figura 6 se muestra el resultado de la asignación. El asistente le da la oportunidad al diseñador de la base de datos de escoger el nombre de los fragmentos físicos y si desea generar los scripts que servirán de ayuda para llevar a cabo el diseño físico de la base de datos sobre servidores de Microsoft SQL Server que se ejecutan en los sitios de procesamiento. Si el checkbox permanece marcado, al hacer clic en el botón “Siguiente” el asistente mostrará una ventana

para que el diseñador asigne los nombres de los servidores de SQL Server que se ejecutan en los sitios correspondientes (figura 7). En caso contrario se mostrará la ventana final del asistente (figura 9).

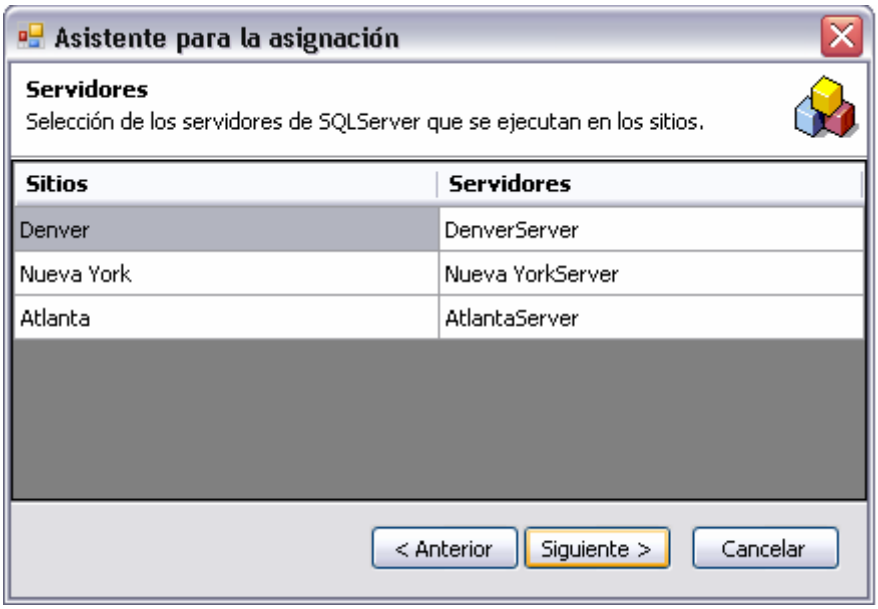


Figura 7. Selección de servidores de SQLServer.

En la figura 7 se muestra una ventana donde aparecen los sitios y los nombres de los servidores de SQL Server que se ejecutan en los mismos. El diseñador tiene la posibilidad de cambiar los nombres de los servidores que aparecen por defecto. Al hacer clic en el botón “Siguiente”, el asistente mostrará en la próxima ventana los scripts generados para cada sitio (figura 8).

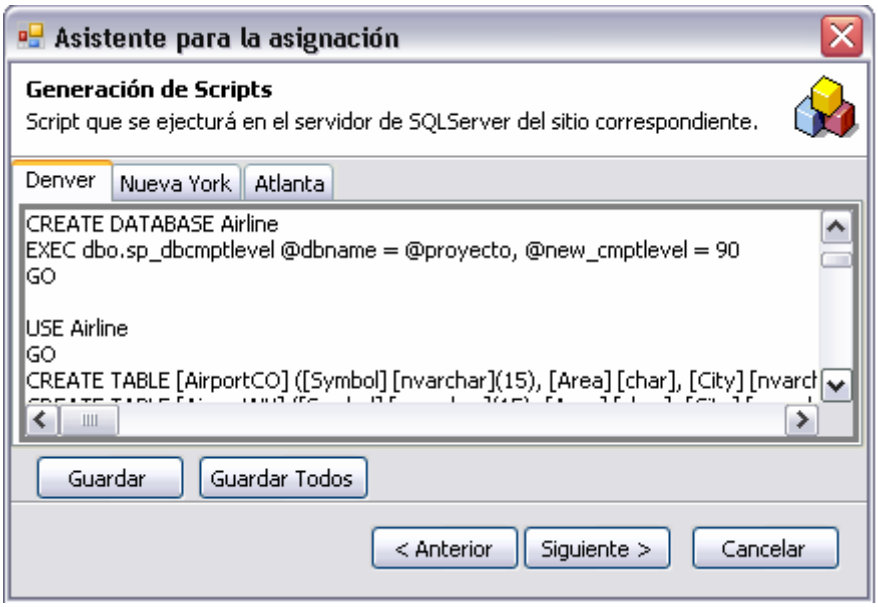


Figura 8. Scripts para SQL Server.

En la ventana de la figura 8 se muestran los scripts obtenidos para cada sitio a partir de los resultados de la asignación. El diseñador puede editarlos en la caja de texto correspondiente y guardarlos, para posteriormente ejecutarlos en los servidores de SQL Server que se ejecutan en los sitios. Al hacer clic en el botón “Siguiente” se mostrará la ventana final del asistente (figura 9).

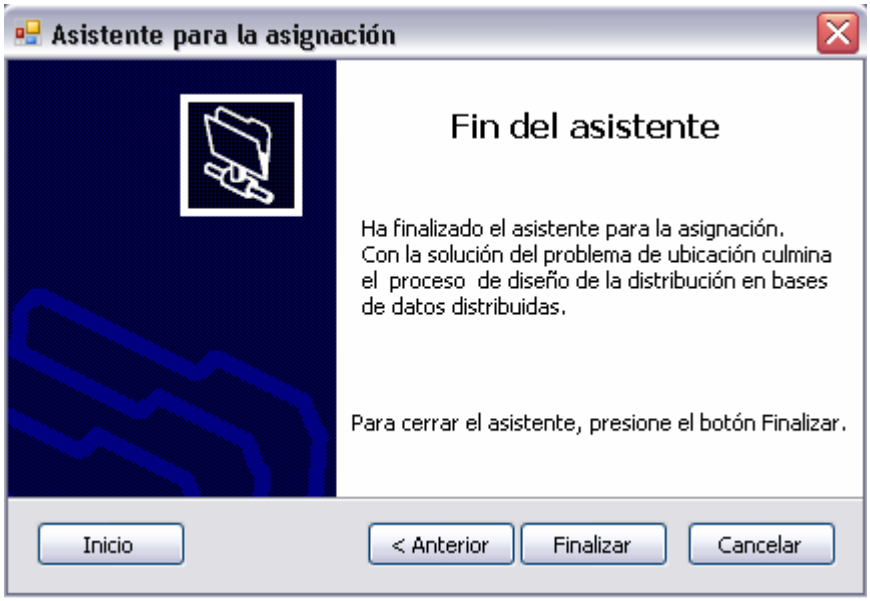


Figura 9. Ventana final del asistente.

Anexo 3. Casos de prueba para el problema de asignación, sin considerar replicación.

Caso	Fragmentos	Sitios	Aplicaciones	Óptimo(ms)
1	10	4	8	5377.4
2	20	8	16	10754.8
3	40	16	32	21509.6
4	80	32	64	43019.2
5	90	36	72	48396.6
6	100	40	80	53774
7	120	48	96	64528.8
8	160	64	128	86038.4
9	170	68	136	91415.8
10	180	72	144	96793.2

Anexo 4. Casos de prueba para el problema de asignación, considerando replicación.

Caso	Fragmentos	Sitios	Aplicaciones	Óptimo (ms)
1	6	2	22	61104,85
2	7	4	5	13549,53
3	3	10	6	4479,58*
4	17	3	8	52659,82*
5	24	2	10	102502,55*
6	16	3	22	54498,22*
7	8	6	16	71426,56*
8	7	11	5	26087*
9	11	8	7	40975,05*
10	25	7	5	51282,82*
11	14	9	29	252440,2*
12	24	16	16	198615,14*
13	27	18	18	369647,17*
14	30	20	20	339398,45*
15	36	24	24	549275,72*

* No se conoce el óptimo con exactitud.

Anexo 5. Resultados cuantitativos de los experimentos realizados a los algoritmos SARSA y Q-Learning para los casos de prueba expuestos en el anexo 3.

		SARSA		Q-Learning	
Caso	**Óptimo**	**AVE**	**Error**	**AVE**	**Error**
1	5377.4	5377,4	0	5377,4	0
2	10754.8	10754,8	0	10768,78	0,13
3	21509.6	21509,6	0	21509,6	0
4	43019.2	44046,6	2,39	43952,71	2,17
5	48396.6	49975	3,26	49558,11	2,40
6	53774	55903,4	3,96	55672,73	3,53
7	64528.8	67760,2	5,01	67800,41	5,07
8	86038.4	90861,2	5,61	90933,98	5,69
9	91415.8	96396,7	5,45	96553,36	5,62
10	96793.2	101932,1	5,31	101855,4	5,23

AG: Algoritmo SARSA.

QL: Q-Learning.

Óptimo: Costo de la solución óptima según el modelo de asignación, dado en milisegundos.

AVE: Costo promedio de las soluciones alcanzadas en cada ejecución, dado en milisegundos.

$$\textbf{Error} = \frac{|\text{AVE - Optimo}|}{\text{Optimo}} \cdot 100$$

Anexo 6. Resultados cuantitativos de los experimentos realizados a los diferentes algoritmos para los casos de prueba expuestos en el anexo 4.

		AG			QL			Greedy			AG-QL		
Cas	**Óptimo**	**AVE**	**VO**	Error	AVE	VO	Error	AVE	VO	Error	AVE	VO	Error
1	61104,85	61104,85	20	0	61106,27	19	0	62669,134	17	2,56	61104,85	20	0
2	13549,53	13599,85	16	0,37	13559,16	17	0,07	13930,271	14	2,81	13595,59	19	0,34
3	4479,58*	4479,58	20	0	4496,48	19	0,38	4619,3428	13	3,12	4479,58	20	0
4	52659,82*	52951,11	11	0,55	52659,82	20	0	54645,095	9	3,77	52854,66	15	0,37
5	102502,5*	103040,35	10	0,52	102571,99	5	0,07	105772,32	7	3,19	102830,5	13	0,32
6	54498,22*	54726,56	13	0,42	54651,85	15	0,28	57059,636	4	4,70	54857,90	15	0,66
7	71426,56*	71918,28	6	0,69	71962,17	4	0,75	74405,047	2	4,17	71690,83	12	0,37
8	26087*	26371,27	2	1,09	26672,93	1	2,25	27420,045	1	5,11	26337,43	9	0,96
9	40975,05*	41251,02	4	0,67	41922,69	1	2,31	43306,530	0	5,69	41376,60	11	0,98
10	51282,82*	52540,99	1	2,53	52080,81	3	1,64	55241,853	0	7,72	52011,03	6	1,42
11	252440,2*	257096,76	2	1,84	259728,43	1	2,89	275084,08	0	8,97	255822,8	6	1,34
12	198615,1*	204943,25	2	3,19	210416,34	1	5,94	215815,16	0	8,66	200919,0	4	1,16
13	369647,1*	386241,44	1	4,49	400167,30	1	8,26	405724,65	0	9,76	375598,4	2	1,61
14	339398,4*	353381,61	1	4,12	372421,86	0	9,73	380499,54	0	12,11	344557,2	5	1,52
15	549275,7*	580199,92	1	5,63	613211,39	0	11,64	622274,44	0	13,29	558009,1	4	1,59

AG: Algoritmos Genéticos.

QL: Q-Learning.

Greedy: Método que sigue una estrategia greedy.

AG-QL: Método híbrido.

Óptimo: Costo de la solución óptima o la mejor conocida, según el modelo de asignación, dado en milisegundos.

AVE: Costo promedio de las soluciones alcanzadas en cada ejecución, dado en milisegundos.

VO: Número de veces que se encontró la solución óptima o la mejor solución conocida.

$$\textbf{Error} = \frac{|\text{AVE - Optimo}|}{\text{Optimo}} \cdot 100$$

* No se conoce el óptimo con exactitud.