UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

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

Autor: M. Sc. Abel Rodríguez Morffi
Tutora: Dra. Luisa Manuela González González

Santa Clara
2007

AGRADECIMIENTOS

- A mi tutora, la Dra. Luisa Manuela González González, por tantas buenas enseñanzas ofrecidas, por ser guía insustituible y ejemplo de consagración, por dedicarme gran parte de su valioso tiempo, por su apoyo incondicional, y por su confianza.
- Al Dr. Ramiro Alberto Pérez Vázquez por sus valiosas observaciones.
- Al Dr. Rogelio Silvino Silverio Castro por sus convenientes consejos.
- A los Doctores María Matilde García Lorenzo, Zoila Zenaida García Valdivia, Ricardo del Corazón Grau Ábalo, Carlos Morell Pérez, Daniel Gálvez Lío y Vicente Fructuoso Molina Padrón, por sus preciadas sugerencias y la ayuda brindada.
- Al MSc. Waldo Pérez García por su incalculable ayuda y su esfuerzo por dar visibilidad a este trabajo.
- A la Dra. Marisela Mainegra Hing por la dedicación y el aporte incuestionable.
- A la MSc. María Elena Martínez del Busto, por ser incondicional e irremplazable.
- A la MSc. Noely González Hernández por su colaboración en la edición del texto.
- A la Dra. Yanet Rodríguez Sarabia por su indudable cooperación.
- A los estudiantes del grupo de trabajo de Bases de Datos por sus aportes a este trabajo: Norma Elisa Cabrera González, Leonel Cabrera Tamayo, Michel Artiles Pérez, Lázaro Sergio Águila Díaz, Ángel Valdés López, Darien Rosa Paz, William Abel Álvarez Martínez de la Cotera, Alain Cárdenas Castillo y Alejandro Hernández Nápoles.
- A los Maestros en Ciencia Alberto Morell Pérez, Magdiel García González, Ileana Zamora, Nelson Jiménez Martínez por sus importantes aportes.
- A la OBE de Sancti Spíritus, en especial al Ing. Raúl Fernández Álvarez y al MSc. Magdiel García González por su colaboración.
- A la Dra. Sara Catalina Hernández Gallardo, MSc. Lourdes María Ramírez Gómez y Lic. Gilda Martínez Guelmes por estar siempre conmigo.
- A mis compañeros del Departamento de Ciencia de la Computación por el apoyo brindado en estos años de bregar.
- Al Consejo de Universidades Flamencas por su apoyo financiero a través del programa IUC VLIR-UCLV.

DEDICATORIA

A mi familia

SÍNTESIS

La posibilidad de distribuir datos sobre diferentes sitios de una red de computadoras ha incrementado la difusión de los sistemas de Bases de Datos Distribuidas (BDD). No obstante los beneficios que reporta la distribución de datos, y de los resultados teóricos que permiten su modelación, aún las técnicas, metodologías y herramientas asociadas no alcanzan la madurez necesaria para ayudar a los desarrolladores de software básico ni de aplicaciones.

Como resultado de este trabajo se obtuvo el Sistema Integrado de Ayuda al Diseño de BDD (SIADBDD) constituido por las herramientas: ERECASE que sistematiza la resolución de problemas de modelación de esquemas conceptuales globales, NETWIZARD que capta la información necesaria para la caracterización de la red y los sitios de procesamiento, APPWIZARD que captura información sobre las aplicaciones del universo de discurso, FRAGMENTER que realiza la fragmentación de esquemas globales en esquemas lógicos, y ALLOCATOR que lleva acabo la asignación lógica y física de los fragmentos a los correspondientes sitios de procesamiento.

Este sistema ha sido validado en el diseño de una BDD para el control de los transformadores de las Organizaciones Básicas Eléctricas (OBE). Ha sido utilizado además, con fines docentes en la carrera de Ciencia de la Computación.

TABLA DE CONTENIDOS

Pág.

INTRODUCCIÓN

La cantidad de innovaciones tecnológicas que han surgido en los últimos años ha promovido un cambio en la forma de enfocar las aplicaciones computacionales. También el continuo descenso de los costos del hardware, en contraposición al incremento de la complejidad y costos del software a gran escala, han estimulado el interés en el desarrollo e implementación de los Sistemas de Bases de Datos Distribuidas (SBDD), los cuales han resuelto de manera sutil, la aparente dicotomía existente entre dos puntos de vista del procesamiento de datos: los Sistemas de Bases de Datos Centralizadas (SBDC) y la tecnología de redes de computadoras, representativas de integración y distribución respectivamente. El objetivo fundamental de los SBDD es integrar la manipulación de datos para que sean presentados al usuario como una única colección de datos global y coherente [182].

Según Özsu y Valduriez [182], una Base de Datos Distribuida (BDD) es una colección de múltiples Bases de Datos (BD), lógicamente interrelacionadas distribuidas sobre una red de computadoras. La distribución involucra el hecho de que los datos no residen necesariamente en el mismo sitio, pero poseen propiedades comunes que los vinculan, y se facilita su acceso a través de una interfaz común. Es necesario destacar que los enlaces entre los datos se llevan a cabo en una red de comunicación, lo que implica generalmente que los sitios estén localizados en diferentes áreas geográficas y con capacidad de procesamiento autónomo.

Existen varias razones técnicas para distribuir datos, en particular, 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. 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 [182], pero al mismo tiempo añaden nuevas complejidades a su diseño e implementación.

Las diferentes técnicas de distribución de datos generalmente se basan en la semántica de los datos, y se rigen por principios idénticos que las BD centralizadas, incorporando otros detalles particulares, como la fragmentación de las entidades y su posterior localización en los diferentes sitios de la red. La fragmentación es muy útil a los fines de mejorar los tiempos de respuesta y garantizar el paralelismo de un sistema [15, 45, 118, 124, 137, 182]. 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 sobre 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.

Con el propósito de mostrar la problemática expuesta y la contribución de este trabajo para su solución, se presenta el problema del control de los transformadores y las operaciones relacionadas en la explotación de los mismos en las diferentes unidades de la Organización Básica Eléctrica (OBE) de Sancti Spíritus, que es típica de las OBEs en general. Desde el punto de vista tecnológico se necesita reflejar la estructura distribuida de las unidades y mantener la autonomía local ofreciendo buen rendimiento. La información que se maneja es de vital importancia para la operación de la red electroenergética nacional y se realiza siguiendo los procedimientos aprobados por el área técnica de la Unión Eléctrica.

Antecedentes y actualidad del tema

Los primeros modelos de distribución de datos fueron presentados en la década de 1970 [53, 88, 108, 109, 142, 155]; aunque un verdadero intento de formalización del problema de distribución de datos se reporta en la década de 1980 con los trabajos de Stefano Ceri [34-39], en que se caracteriza el problema y se brindan soluciones mediante complejos modelos matemáticos que constituyen el núcleo de los desarrollos posteriores en el área de las BDD. A pesar del indudable valor de dichos modelos, la

mayoría tenían una complejidad inherentemente exponencial y no presentaban un enfoque integrado, pues su propósito no estaba dirigido hacia su modelación computacional.

No obstante los beneficios que reporta la distribución de datos, y de los resultados teóricos que permiten su modelación, aún las técnicas, metodologías y herramientas asociadas no alcanzan la madurez necesaria para ayudar a los desarrolladores de software básico ni de aplicaciones. Desde el punto de vista comercial, los Sistemas Manejadores de Bases de Datos (SMBD) soportan la replicación de información para mantener copias redundantes de datos y así lograr mayor disponibilidad y tolerancia ante fallos; aunque la replicación de información no se realiza aún con la transparencia requerida.

La mayoría de los SMBD comerciales utilizan el modelo cliente-servidor con múltiples clientes y un solo servidor, donde el soporte para replicación de información o no se ofrece o se hace a través del protocolo sincrónico (eager) lee-uno-escriben-todos (ROWA, acrónimo del inglés Read-One-Write-All), en el cual una transacción que actualiza una réplica no termina hasta que todas las réplicas hayan sido actualizadas [182, 186, 187] brindando una fuerte consistencia. La replicación sincrónica no es un enfoque viable en la mayoría de los ambientes de procesamiento de datos actuales [4]. Por otra parte, la replicación asíncrona (lazy) tratada en ocasiones como optimista o preventiva [3, 4, 30, 47, 68, 69, 76, 110, 120, 121, 183-188] sacrifica consistencia por lograr mejor desempeño ya que los tiempos de respuesta son menores al no existir ninguna comunicación intratransacción.

La ubicación de los datos próximos a donde se usan incide en la eficiencia del acceso a los mismos; para esto es necesario tener un buen soporte para fragmentación y replicación de información. Otro aspecto donde se puede incrementar la eficiencia es 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 que un sistema evolucione sin que se vea degradado su desempeño, es decir, la obtención de sistemas escalables. La creación de tales sistemas de manera rápida y económica se ha logrado por el desarrollo de la tecnología de microprocesadores y estaciones de trabajo. Sin embargo, respecto de

la escalabilidad, la comunicación de la información tiene un costo que no se ha estudiado con suficiente profundidad.

Aunque una gran cantidad de investigadores han propuesto modelos y han diseñado algoritmos para ubicar fragmentos en BDD [94, 96, 104, 134, 138, 140, 148, 149, 153, 201, 224, 244, 264], la mayoría de los modelos son muy complicados y difíciles de entender. Así, es difícil usarlos en un ambiente real.

Formulación del problema de distribución

El problema de distribución de datos que enfrentan los diseñadores de BDD consiste en definir cómo distribuir la BD entre todos los sitios de una red en aras de lograr máxima localidad, que los datos se encuentren en el sitio donde más se necesiten, aprovechar la distribución para paralelizar ciertas operaciones o distribuir la carga de trabajo. El procedimiento a seguir le exige resolver dos problemas fundamentales: cómo fragmentar los esquemas que conforman la BD (problema de fragmentación) y dónde ubicar cada uno de estos fragmentos (problema de asignación), de tal manera que se asignen según el objetivo de optimización definido. Probablemente será necesario tener réplicas de estos fragmentos en varias localidades para facilitar tanto los accesos como su disponibilidad. Lo anterior exige decisiones de diseño difíciles y procesamientos complejos que han sido abordados anteriormente de forma incompleta y no integrada.

La modelación de dicho problema tratada en la literatura se refiere al nivel lógico. Si bien es cierto que la modelación hasta el nivel físico no puede ofrecer soluciones generales por su fuerte dependencia de los motores de BD, al menos pueden ofrecerse determinadas soluciones propias de este nivel, así como soluciones particulares a los motores más populares. Por otro lado, las soluciones que se mencionan a nivel lógico, en su mayoría son formulaciones matemáticas sin los correspondientes desarrollos algorítmicos. La fragmentación horizontal primaria ha sido tratada con más profundidad y se han propuesto algoritmos para llevarla a cabo, aunque se usan criterios subjetivos que dependen del juicio del diseñador [19, 21, 36, 83, 85, 145, 174, 234]. Para la fragmentación horizontal derivada, sólo se han dado algunos criterios a tener en cuenta cuando existe más de una posible fragmentación [182]. La fragmentación vertical ha sido tratada ampliamente y se han propuesto varios algoritmos para llevarla a cabo [22, 43, 48, 50, 64, 74, 84, 96, 135, 139, 167, 168, 170, 198, 240]. El menos formalizado es el problema de asignación, para el cual se han propuesto soluciones muy particulares, aplicables sólo baja ciertas condiciones [9, 18, 56, 62, 104-106, 126, 134, 135, 138, 140,

149, 191, 200, 201, 203, 207, 216, 232, 242, 244, 260]. El problema de diseño de distribución de datos consta de varios subproblemas que son NP-Hard. Por ello se trabaja en la aplicación de métodos metaheurísticos, métodos heurísticos de más alto nivel diseñados para la obtención de soluciones razonablemente buenas en un tiempo de cómputo aceptable, evitando el estancamiento en óptimos locales. Estos métodos han sido usados con éxito en otras esferas.

Por todo lo anterior se formula el siguiente **problema de investigación**: No existen herramientas computacionales comerciales de apoyo al diseño de distribución de datos que permitan ayudar a los diseñadores en cada una de sus etapas de manera armónica.

El **objetivo general** de esta investigación es contribuir a sistematizar el proceso de diseño de BDD desde el análisis de requerimientos hasta su soporte por un SMBD, y brindar soluciones y herramientas computacionales acordes a cada nivel de abstracción que ayuden a realizar este proceso.

Para lograr este objetivo se plantean los siguientes **objetivos específicos** de la investigación:

1. Formalizar la captación de la información necesaria para el diseño de la distribución de datos en asistentes independientes, teniendo en cuenta la naturaleza de la información a captar sobre: redes, sitios, aplicaciones y datos.
2. Crear herramientas para abordar los tres tipos de fragmentación; así como la asignación mediante métodos heurísticos que consideren la replicación, y contemplar la posibilidad de obtención de esquemas hasta el nivel físico.
3. Diseñar una arquitectura que considere un catálogo como elemento de integración de las diferentes componentes (asistentes y herramientas).

Como resultado de la revisión bibliográfica y de la factibilidad del trabajo en función de los objetivos, se define la siguiente **hipótesis de investigación**: La estrategia "divide y vencerás" permitirá abordar el proceso de distribución de datos mediante el desarrollo de soluciones a los problemas acordes a su naturaleza matemática a través de herramientas que se integren en una arquitectura de referencia para ofrecer ayuda a los diseñadores de BDD.

Las **tareas de investigación** trazadas son:

1. Analizar a profundidad el modelo teórico que da soporte al diseño de BDD y su evolución, así como la implementación de métodos existentes para realizar el diseño de BDD.

2. Determinar los parámetros que caracterizan la BD, las aplicaciones, los sitios de procesamiento y la red de comunicación que son útiles para el diseño de BDD; y describir un catálogo para almacenar tales parámetros como una BD relacional.
3. Desarrollar un método para la obtención de fragmentos horizontales derivados.
4. Dividir el proceso de asignación en dos etapas: lógica y física; y desarrollar algoritmos para la asignación redundante de fragmentos a sitios que consideren el modelo de réplicas.
5. Desarrollar herramientas que se integren al proceso de diseño de BDD y que disminuyan el tiempo de ejecución, así como el esfuerzo requerido en este proceso.
6. Validar la herramienta creada.

En la realización de este trabajo se aplicaron diferentes métodos de trabajo científico. Concretamente se empleó el método de análisis-síntesis para procesar, integrar, interpretar y valorar la información consultada; el método sistémico para presentar el objeto de investigación en su integridad, reflejando sus componentes y los nexos de funcionalidad existente entre ellos; y el método de modelación y síntesis para la concepción del sistema que integra las herramientas obtenidas como contribuciones de este trabajo. La observación fue empleada para apreciar los resultados publicados en la literatura y determinar las deficiencias que existen en estos. El criterio de usuario es utilizado finalmente para evaluar la validez de las herramientas desarrolladas en la solución a un problema real.

La **novedad científica** del presente trabajo consiste en:

1. Brindar consideraciones generales matemáticamente fundamentadas para sistematizar el proceso de diseño de BDD hasta el nivel físico.
2. Fundamentar un criterio para la fragmentación horizontal derivada basado en la dependencia de existencia, presente en las asociaciones entre tipos de entidades.
3. Ofrecer una solución computacional al proceso de asignación que considera la replicación, mediante dos métodos metaheurísticos: uno basado en Algoritmos Genéticos Generacionales, y otro basado en el método Q-Learning de Aprendizaje Reforzado.

Por su parte, el **valor práctico** consiste en la obtención de un sistema integrado de herramientas de ayuda a los diseñadores de BDD. Este sistema se empleó satisfactoriamente en la solución al problema del control de transformadores en las OBEs, en particular de Sancti Spíritus.

Por último, el **valor metodológico** está dado por la sistematización en el proceso de diseño de BDD con un enfoque metodológico basado en los tres niveles de modelación de la arquitectura de referencia de las BDD: conceptual, lógico y físico, lo cual tiene repercusiones docentes.

La **tesis** está **estructurada** en tres capítulos:

En el capítulo 1 se expone una panorámica del estado del arte y los aspectos que son particularmente interesantes en el diseño de BDD. Se hace énfasis en el modelo teórico y los diferentes enfoques que se le ha dado a este problema. Se realiza un análisis crítico de los principales trabajos reportados en la bibliografía y se orienta el enfoque particular del presente trabajo.

En el capítulo 2 se proponen soluciones desde el punto de vista teórico o práctico a las problemáticas de diseño identificadas. Se comentan algunas investigaciones recientes que permitieron su perfeccionamiento y su implementación computacional. Los métodos seleccionados se analizan y se implementan con resultados confiables en un tiempo de respuesta aceptable. Desde el punto de vista metodológico se exponen los resultados siguiendo los tres niveles de modelación de la arquitectura de referencia planteada en el capítulo 1.

En el capítulo 3 se validan las herramientas obtenidas como resultado de esta investigación, en la gestión del control de los transformadores de una OBE típica, concretamente de Sancti Spíritus. Se sigue el ciclo de los transformadores desde que se instalan en un banco hasta que son retirados de la línea, almacenando todas las actividades que se realizan sobre los mismos.

La tesis culmina con las conclusiones, recomendaciones, bibliografía y anexos.

1. ESTADO DEL ARTE EN EL DISEÑO DE BASES DE DATOS EN AMBIENTES DISTRIBUIDOS

En este capítulo se expone una panorámica del estado del arte y los aspectos que son particularmente interesantes en el diseño de BDD. Se hace énfasis en el modelo teórico y los diferentes enfoques que se le ha dado a este problema, realizando un análisis crítico de los principales trabajos reportados en la bibliografía.

1.1. Introducción a las bases de datos en ambientes distribuidos

El término BDD tiene diferentes definiciones, para el propósito de este trabajo es suficiente la definición dada por Özsu y Valduriez en [182], mencionada en la introducción. Esta definición enfatiza los principales aspectos de las BDD: la distribución, o sea, el hecho de que los datos no residan en el mismo sitio; y la interrelación lógica, lo cual significa que los datos poseen algunas propiedades comunes que los vinculan y son accedidos por una interfaz común. Es necesario destacar que el enlace entre los datos se realiza a través de una red de comunicación, lo que no implica necesariamente que los sitios estén localizados en diferentes áreas geográficas, pero cada uno tiene capacidad de procesamiento autónomo y puede ejecutar aplicaciones locales. Se pudiera precisar el concepto de BDD 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.

Por su parte, un Sistema Manejador de Bases de Datos Distribuidas (SMBDD) es el software que permite la administración de la BDD y hace la distribución transparente al usuario [182]. Cuando se realiza un diseño inicial de una BDD es recomendable construir el diseño global de la BD utilizando un sistema homogéneo, en el cual todos los SMBDD locales poseen características idénticas. Los SBDD heredan el grado de dificultad de los sistemas de BDC e incorporan un nuevo nivel de complejidad, que está asociado a la distribución de datos en diferentes sitios de la red. Es por esta razón que se necesita aclarar que las BDD incluyen los mismos elementos que las Bases de Datos

Centralizadas (BDC) pero no son simples implementaciones distribuidas de las BDC; Ceri y Pelagatti [37] hacen un estudio de las diferencias entre ambas.

Respecto a la distribución de datos en un sistema de computadoras distribuidas, es necesario aclarar qué se pretende distribuir, pues existen situaciones para las cuales este asunto es objeto de confusión. Al ser los sistemas distribuidos elementos autónomos de procesamiento interconectados por una red de computadoras se deduce claramente que es necesario distribuir el procesamiento lógico. Otra posibilidad es distribuir las funciones de un sitio, que pueden ser delegadas a varias piezas de hardware o de software. Otro modo de distribución es 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 [182], existen varias alternativas para distribuir la BD acorde con la fragmentación y/o replicación que se aplique a la misma:

1. Fragmentada y no replicada,
2. No fragmentada y replicada,
3. 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 y 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 almacenadas en varios sitios.

Existen diversos factores que complican el proceso de diseño de BDD [182] como son:

1. Transparencia de la distribución: Se refiere a ocultar la distribución a los usuarios e incluso a los programas de aplicación.
2. Diseño de la BD: Se trata de diseñar la BD buscando un mejor desempeño. Muchos de los problemas que se presentan tienen una modelación matemática compleja y otros no tienen solución computacional con respuesta en tiempo aceptable, por tanto se usan métodos heurísticos para ellos.
3. Procesamiento distribuido de solicitudes: Se recurre a métodos que evitan soluciones desfavorables pero no se buscan soluciones óptimas.
4. Manejo de directorio distribuido: Se presentan los mismos problemas de decisión que con la BDD relativas a fragmentación, replicación, etc.

5. Control de concurrencia distribuido: Es mucho más complejo que en las BDC.

Por su importancia dentro de este trabajo, a continuación se hace un análisis de los dos primeros factores enumerados anteriormente.

La tecnología de BD intenta extender el concepto de independencia de datos a ambientes en los que los datos son distribuidos y replicados en determinado número de máquinas conectadas por medio de una red. La independencia de los datos está dada por una serie de formas de transparencia: transparencia de la fragmentación, transparencia de la red (por tanto de la distribución), y transparencia de la réplica. Así, los usuarios de la BDD verían una sola imagen, lógicamente integrada, permitiéndoles el acceso como si se tratara de una BDC. El grado de sofisticación de los SMBDD se mide a menudo por el grado de la transparencia de distribución otorgado a los usuarios. En una situación ideal, el usuario no necesita estar pendiente de la distribución de los datos, y el sistema toma la responsabilidad de distribuir las operaciones de acceso a las BD en diversos sitios. Sin embargo, la distribución real de los datos afecta todo el desempeño respecto al tiempo y al costo requerido para acceder a los mismos, dependiendo de si todos los datos están almacenados en el mismo lugar o si se encuentran en varios sitios. La mayoría de los SMBDD comerciales no proveen un nivel suficiente de transparencia y una parte está dada por la falta de apoyo del manejo de datos replicados. Algunos SMBDD intentan establecer sus propios esquemas de nombramiento transparentes, usualmente con resultados insatisfactorios. El Sistema Operativo puede a su vez ayudar con la transparencia de replicación, dejando la tarea de la transparencia de fragmentación al SMBDD.

Las BDD como sistemas integrados, asumen que la distribución sea transparente a los usuarios [182]. 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, aún cuando el sistema tenga un alto grado de transparencia. La replicación es un tema muy discutido que se demanda cuando se desea incrementar el rendimiento de un sistema global de información o incrementar la fiabilidad de algún

medio de almacenamiento [171]. 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 enfrenta un problema difícil: ¿Cómo distribuir los datos y programas en diferentes sitios para obtener el desempeño deseado, la confiabilidad y la disponibilidad? De esta manera el diseño de la distribución se convierte en un problema que requiere su propia teoría, metodología de diseño, y herramientas de apoyo. La comunidad de BD todavía no tiene un entendimiento completo sobre las implicaciones del desempeño de todas las alternativas de diseño distribuido. Diseñar una BDD es una tarea realmente difícil, a los problemas que se presentan para BDC se añaden otros nuevos tales como la distribución óptima de datos y de las aplicaciones en los diferentes sitios. Los dos principios básicos de las BDD [12, 182] son reducir el intercambio de datos entre los sitios y eliminar datos irrelevantes en la ejecución de solicitudes. Estos sirven de guía al proceso de diseño de BDD, el cual ha sido dividido en cuatro pasos fundamentales [12, 15, 39, 54, 149, 182, 224, 228]:

1. Diseño del esquema conceptual donde se describe la BD integrada.
2. Diseño de la fragmentación.
3. Diseño de la asignación o ubicación de los fragmentos.
4. Diseño de la BD física, transformando los esquemas en las áreas de almacenamiento y determinando los métodos de acceso apropiados.

Los problemas 1 y 4 son comunes a las BDC y a las BDD. Los problemas 2 y 3 caracterizan el diseño de BDD. La fragmentación puede decirse que se ocupa fundamentalmente de los criterios lógicos que motivan la descomposición de esquemas globales en fragmentos mientras que la asignación se ocupa de los aspectos físicos de su ubicación y réplica en sitios; aunque hay una diferencia entre ambos procesos, su interrelación es importante para obtener un diseño óptimo. Como se expresó anteriormente, la replicación incrementa la disponibilidad y la eficiencia de las solicitudes de sólo lectura al poder ser ejecutadas en paralelo. Por otra parte, la ejecución de solicitudes de actualización causa problemas, ya que el sistema tiene que asegurar que todas las copias de los datos sean adecuadamente actualizadas. De aquí

que la decisión con relación a la replicación es una cuestión que depende del peso de las solicitudes de sólo lectura con respecto a las solicitudes de actualización. En este capítulo se hace un análisis de aspectos teóricos relacionados con los problemas 2 y 3, que particularmente son de interés en esta investigación.

1.2. Necesidad de la distribución de datos

El uso y generalización de los SBDD está condicionado por dos motivos fundamentales; por una parte el continuo descenso del costo del hardware, en contraposición al incremento del costo del software de aplicación a gran escala. Su objetivo fundamental es integrar la manipulación de datos para presentarlos como una única colección global y coherente.

La posibilidad que ofrecen las BDD de difundir datos sobre diferentes sitios de una red de computadoras y ejecutar aplicaciones que requieran el acceso a datos ubicados en más de un sitio, es muy codiciada por numerosas organizaciones que poseen una estructura distribuida, ya sea lógica o físicamente. En muchas organizaciones geográficamente distribuidas las vías centralizadas no representan una alternativa factible, y la migración hacia SBDD resulta natural, ya que esta alude de manera natural la estructura descentralizada de la organización. En estos casos, cada unidad dentro de la empresa 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 [67].

1.3. Modelo lógico de distribución de datos

El modelo lógico de las BDD fue formulado inicialmente por Ceri *et al.* en 1983 [35], 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. En un sistema bien diseñado al menos el 90 por ciento de los datos deben ser ubicados localmente y hasta el 10 por ciento pueden ser accedidos remotamente [39].

El diseño de la fragmentación refleja 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. Un esquema no es una unidad de asignación por un gran número de razones. Primero, las aplicaciones usualmente operan sobre subconjuntos de esquemas. Por tanto, la localidad de accesos de aplicaciones no es definida sobre esquemas completos sino sobre sus subconjuntos. De aquí que sea natural considerar estos subconjuntos como las unidades de distribución. Segundo, 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. O bien el esquema no está replicado y es asignado a un sólo sitio, o éste se encuentra replicado en todos o en algunos de los sitios donde las aplicaciones residen.

Finalmente, se puede argumentar a favor de la fragmentación que al tratar la descomposición de un esquema en fragmentos como unidades de asignación, se está facilitando la ejecución concurrente de un gran número de transacciones y por lo tanto mejora el desempeño de un sistema. También, la fragmentación facilita que la ejecución de una sola solicitud se pueda dividir en un conjunto de subsolicitudes que operen sobre los fragmentos, posibilitándose el paralelismo [25, 97, 124].

El control semántico de datos es también un problema a valorar; específicamente el chequeo de integridad es importante y tiene una complejidad elevada, ya que como resultado de la fragmentación, los atributos que participan en una dependencia pueden ser descompuestos en fragmentos diferentes y por tanto pudieran ser ubicados en sitios diferentes.

Muchos de los problemas relacionados con la fragmentación y asignación en el diseño de BDD tienen una modelación matemática compleja y los métodos que les dan solución son considerados del tipo NP-Completo [12, 134, 182, 201], 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 [15, 38, 39, 157, 169, 201, 228, 244] que sugieren el uso de heurísticas para algunos de ellos.

El modelo lógico de distribución de datos no ha sido tratado totalmente en ninguna metodología integradora y tampoco se tienen herramientas comerciales que ayuden a resolver los problemas propios de modelación de BDD.

1.4. Evolución del modelo lógico

A continuación se analiza el desarrollo del modelo básico formulado por Ceri y Pelagatti [37] en 1983.

1.4.1. Modelación de datos

Un modelo de datos es una colección de conceptos y notaciones para describir datos, sus interrelaciones, su semántica, restricciones de consistencia, y operaciones sobre los mismos. Un modelo de datos ofrece una manera de describir una BD para un determinado nivel de abstracción [235].

Un modelo de diseño también proporciona diversas interpretaciones o visiones con diferentes niveles de abstracción y describe las expectativas de los usuarios sobre las aplicaciones o el comportamiento de un sistema [236]. La metodología del diseño de BDD varía de acuerdo con la estructura de un sistema concreto; usualmente hay dos enfoques alternativos para el diseño de una BDD, uno es el descendente (top-down) y el otro es el ascendente (bottom-up). La modelación de un problema real se presenta usualmente de una manera no pura, lo que requiere del empleo de ambas alternativas. La estrategia descendente (véase la figura 1.1) es adecuada para el desarrollo inicial de un SBDD sin tener restricciones de otros sistemas ya instalados y que deban ser integrados de alguna manera al sistema distribuido.

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. La estrategia ascendente podría aplicarse donde haya que proceder a un diseño a partir de un determinado número de BD existentes, con el fin de integrarlas en una sola. Aunque este caso se pueda presentar con facilidad en la realidad, se prefiere partir de cero y avanzar en el desarrollo del trabajo siguiendo la estrategia descendente, ya que es la que presenta la problemática con toda su amplitud y complejidad. La estrategia descendente comienza con el análisis de los requerimientos y termina con el diseño físico. Es bien conocido que la actividad de diseño y explotación es un proceso en continuo movimiento que requiere un constante monitoreo, ajustes periódicos y puesta a punto. Por esto se incluyen la observación y el monitoreo como una actividad importante en este proceso; monitoreándose no solamente el comportamiento de la BD, sino también la adecuación de las vistas de los usuarios.

En [39, 182] se aborda el proceso de diseño ascendente que no tiene relevancia en esta investigación.

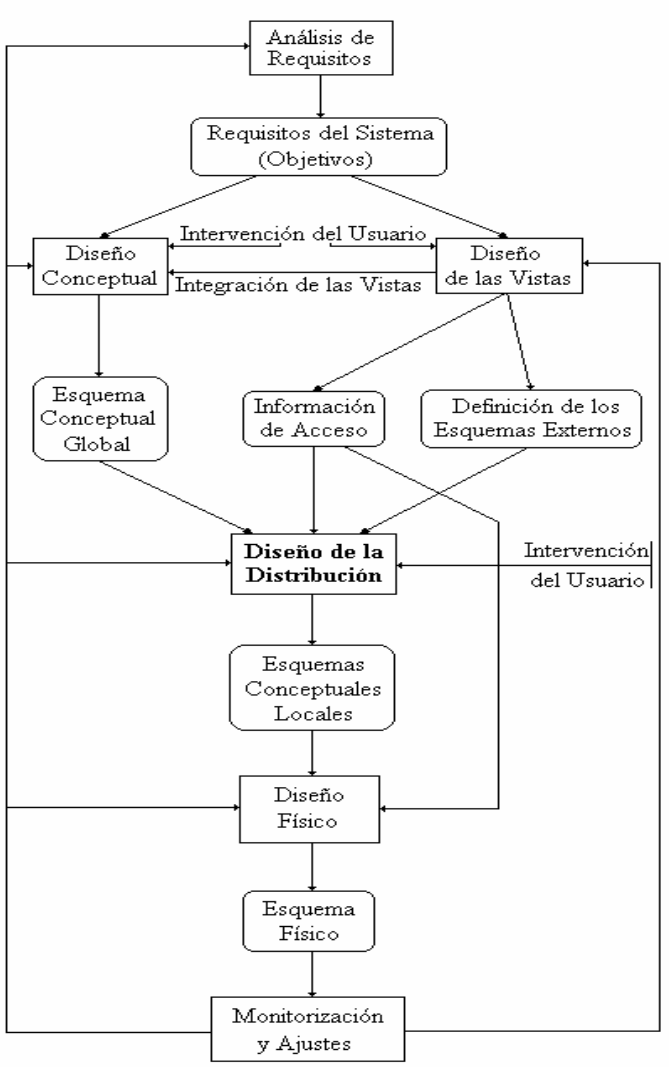

Figura 1.1. Estrategia descendente de diseño de BDD. Tomado de [182].

1.4.2. Modelación conceptual

La modelación conceptual [172, 177-179, 212, 213] es el proceso de creación de representaciones abstractas de un dominio de aplicación en términos de conceptos familiares a los actores de ese dominio y no en términos técnicos. Esta requiere de notaciones, herramientas y técnicas para representar datos y procesos. Las investigaciones actuales tratan de estrechar la barrera entre los conceptos del mundo real y su representación en modelos conceptuales.

Se puede dividir el proceso de modelación conceptual en dos grupos de actividades relacionadas entre sí: análisis de entidades y análisis funcional. El análisis de entidades está relacionado con el análisis de un problema a resolver para determinar los tipos de entidades, sus atributos, y las interrelaciones entre ellos. El análisis funcional, por otra parte, se corresponde con la determinación de las funciones fundamentales con las

cuales se relaciona el problema modelado. Los resultados de estos dos pasos deben hacerse en paralelo para obtener una mejor comprensión sobre cuáles funciones se corresponden con tales entidades.

En las actividades de diseño conceptual y diseño de vistas en BDD, el usuario necesita especificar las entidades y determinar qué aplicaciones van a ser soportadas sobre la BD; también precisa colectar información estadística sobre el acceso de dichas aplicaciones, que incluye la especificación de la frecuencia de las aplicaciones del usuario, el volumen de información, entre otros patrones de acceso y uso que más adelante se tratan. Además, debe obtener información que caracterice la red de comunicación y los sitios de procesamiento donde residirá la BDD, como son los costos de acceso de lectura y escritura a los medios de almacenamiento y costos de comunicación, entre otros factores. Obtener o estimar toda la información requerida es una tarea difícil que requiere de esfuerzo y experiencia.

Los modelos semánticos de datos son beneficiosos en las actividades de diseño conceptual debido a la economía de expresión, ya que las operaciones se definen explícitamente en las interrelaciones, y el modelo de datos incluye la semántica en sí; el mantenimiento de la integridad se da a través de la capacidad de definición de restricciones de integridad y a su vez permite un nivel más alto de abstracción, dotado de capacidades para modelar situaciones del mundo real; con estos modelos los diseñadores usan operaciones y restricciones predefinidas sin tener que implementarlas a bajo nivel [226].

Es ampliamente aceptado el uso del modelo semántico de datos conocido como modelo Entidad-Relación (ER) introducido por Chen [57] y sus extensiones para el diseño del Esquema Conceptual Global (ECG). El modelo ER extendido (ERE) usa los conceptos de entidad, propiedad, interrelación y subtipo para incorporar información semántica importante acerca del mundo real [75, 226, 235, 246]. La mayoría de la información semántica de estos modelos está basada en las interrelaciones entre entidades.

El modelo ERE a veces es impreciso para transformar un diagrama ERE a los formalismos de un SMBDD específico y deja algunos detalles sin concretar (por ejemplo la creación de índices), y es incapaz de tratar las restricciones de integridad relativas al problema (del inglés business rules), excepto en muy pocos casos especiales [67, 226]. No obstante, este modelo ha trascendido como la técnica de modelación conceptual reconocida como ayuda al diseño de BD relacionales. Por su parte, el

Lenguaje de Modelación Unificado (UML) [241] también puede ser útil para la modelación conceptual previa a un diseño relacional, pero su desarrollo ha estado asociado fundamentalmente con el modelo orientado a objetos.

Cualquier herramienta de ayuda a la caracterización del ECG que se integre al proceso de diseño de BDD debe ofrecer una amplia variedad de construcciones del modelo ERE que permitan capturar mejor la semántica del universo de discurso, de forma que se facilite el diseño de la distribución y realicen validaciones a los diagramas para que el resultado sea la obtención de esquemas libres de errores.

Existen varias herramientas de ayuda a la modelación conceptual de datos mediante la creación de diagramas ERE, entre las que se pueden citar las siguientes:

1. Embarcadero ER/Studio v. 6.0.1 [78].
2. Computer Associates AllFusion ERwin Data Modeler v. 4.1.4.3643 [42].
3. CharonWare Case Studio 2 [55].
4. theKompany Data Architect v. 3.0.0 [247].
5. Evergreen Software Tools EasyCase Professional v. 4.21.016 [81].
6. Chilli Source Database Design Studio Lite v. 1.09.0 [60].
7. Datanamic DeZign for databases v. 2.5.2 [66].
8. XTG Systems XTG Data Modeller v. 2.3.4 [263].
9. Microsoft Visio 2003 [159].
10. SmartDraw Suite Edition v. 7.01 [237].
11. Dia v. 0.91 [130].

Un análisis más detallado sobre las herramientas citadas anteriormente muestra que las mismas poseen las siguientes desventajas:

1. Hay poca variedad de construcciones para construir diagramas comprometiendo la expresividad del ECG.

2. Se incluyen elementos de los modelos lógicos y físicos en el diseño, como la migración de llaves foráneas en tiempo de diseño para interrelaciones con cardinalidad uno-muchos que pueden confundir a un diseñador inexperto que quizás sólo tenga conocimientos básicos del modelo ER y no le sea sencillo enfrentarse a conceptos como el de migración de llaves o índices.

3. No permiten colocar atributos a las interrelaciones (DeZign).

4. No permiten interrelaciones ternarias (plantean usar un tipo de entidad débil asociada a tres tipos de entidad) y de grado mayor que tres.

5. Sólo se validan los diagramas siguiendo criterios simples como detectar ausencia o repetición de nombres de entidades, atributos e interrelaciones, entidades sin atributos, atributos sin tipo de datos, entidades sin interrelacionar, entidades sin identificadores. Para diseñadores expertos, algunas de ellas son herramientas para dibujar que no realizan validaciones (SmartDraw, Dia).

6. No se captura información adicional requerida para el diseño de BDD; tal es el caso de las dependencias de existencia presentes en las asociaciones [238], para las cuales no existe una construcción específica, por lo que puede quedar sin modelar si no se identifica.

Según Olivé [178, 179] se ha identificado la calidad como un tema principal en las investigaciones modernas sobre modelación conceptual, por tanto el mejoramiento de la calidad de representaciones conceptuales proporciona mayores oportunidades de mejorar la productividad del desarrollo de los sistemas de información [162]. Aunque se ha hecho énfasis en la necesidad de más guías para el aseguramiento de la calidad de los procesos de modelación, la mayoría de las investigaciones actuales fijan su atención en la calidad del producto [72, 172, 212, 213] y definen criterios y medidas para evaluar la calidad pero no cómo elaborar modelos con alta calidad [162].

La propuesta realizada por Teorey [245] emplea un método heurístico cuya entrada es un diagrama ERE de la BD para obtener un ECG en Tercera Forma Normal, y posteriormente realizar el diseño de la distribución. Gran parte de las decisiones de diseño están fundamentadas en el buen juicio del diseñador de la BD. El trabajo de van Brommel [252] trata sobre la transformación de esquemas durante el desarrollo de un sistema de información, describiendo la transformación de los modelos conceptuales de datos y su representación interna, incluyendo una amplia variedad de parámetros de control a través de herramientas computacionales de ayuda. Ma *et al.* [149] también emplean un modelo ER de alto nivel para el cual describieron un álgebra general para las solicitudes, basada en el trabajo realizado por Schewe [230] aplicado a BD no relacional y plantean la necesidad de analizar las aplicaciones globales para hacer más eficiente y efectivo sus accesos a la BD.

En estas propuestas se emplea el modelo ER para la caracterización de los ECG, obteniéndose diagramas ER que son transformados a esquemas relacionales que posteriormente serán objeto de distribución. Aunque éstas abordan parcialmente el problema de modelación conceptual en BDD, el proceso seguido por las mismas ha

servido de motivación para la presente investigación. Además, se puede concluir que las herramientas consultadas no realizan validaciones estructurales a los diagramas ER y tampoco se integran al proceso de diseño de BDD.

1.4.3. Diseño de la distribución

Las entradas para el diseño de la distribución son las salidas de la modelación conceptual. Estas comprenden información sobre los esquemas de la BD, las aplicaciones, la red de computadoras, y los sitios de procesamiento. Las dos últimas son de carácter cuantitativo y sirven, principalmente para desarrollar el proceso de asignación.

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 obteniéndose así los Esquemas Conceptuales Locales (ECL). Según Date [67], es posible tratar las relaciones como una unidades de distribución, aunque también hay esfuerzos dirigidos a considerar los objetos como unidades de distribución [13, 15, 19, 48, 84-87, 95, 96, 106, 137, 143, 145, 148, 150, 200, 203-205, 231]. Baião *et al.* [12, 15] tratan el problema de elegir las técnicas más apropiadas de fragmentación para aplicar a cada clase del esquema de la BD orientada a objetos, algo que no había sido planteado por los trabajos precedentes en [19, 20, 22, 83, 84, 125, 227, 228]. Por su parte, Pérez *et al.* [201-203, 205] han enfrentado el problema de distribución aplicado a una variedad de clases y ambientes, mientras que Ezeife *et al.* [82-85] han presentado un enfoque de distribución de clases de objetos. En la práctica, la problemática de distribución de objetos se ha abordado insuficientemente. En el diseño físico se llevan los ECL a los dispositivos de almacenamiento físico. La entrada para este proceso es el ECL y la información sobre los patrones de acceso a los fragmentos de éste.

Sobre el problema de distribución se han publicado numerosos trabajos, los primeros se enfocaron al problema de ubicación de archivos FAP (acrónimo del inglés File Allocation Problem). El objetivo planteado para este problema era la optimización de la ubicación de archivos a sitios. Dowdy [73] contempla la revisión de esos trabajos definiendo la optimalidad con respecto a dos medidas: costo mínimo y eficiencia máxima. Hoffer [108] desarrolló un modelo no lineal binario que minimiza una combinación lineal de los costos de almacenamiento, consultas de actualización y de sólo lectura con restricciones de capacidad. Otro de los trabajos sobre 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.* [92]. Otras publicaciones recientes sobre modificaciones a los algoritmos y estructuras aplicados al mismo problema pueden encontrase en [10, 52, 89, 156, 242, 248].

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 [9]. Tomando en cuenta el problema anterior, se genera una nueva forma de solucionar el problema, mediante la ubicación de fragmentos DAP (acrónimo del inglés Data Allocation Problem). Esta nueva 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 [182]. Como el diseño de la distribución de datos abarca una gran cantidad de variables y diversidad de relaciones entre éstas, el problema del diseño se hace complejo [9, 138].

Según Huang [114], el diseño de una BDD involucra varios factores interrelacionados acerca de cómo debe ser fragmentado un esquema, 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 distribución. Estos factores complican en proceso de diseño de BDD aún 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 [182, 225]. Otras realizan el proceso de fragmentación y ubicación simultáneamente usando diferentes enfoques [148, 149, 198, 201, 244].

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 quizás el abordaje más

apropiado para el problema de diseño [182] y motivación fundamental de la presente tesis.

Fragmentación

La mayoría de los autores coinciden en establecer dos vías básicas para fragmentar: horizontal y vertical [12, 15, 39, 114, 149, 151, 182]. Hay variados trabajos reportados en la bibliografía que tratan la fragmentación horizontal (FH) [19-21, 36, 38, 82, 83, 85, 128, 134, 145, 146, 174, 234] o la fragmentación vertical (FV) [22, 43, 48-50, 61, 64, 74, 84, 93-96, 135, 139, 167, 168, 170, 198, 207, 240] pero muy pocos han dado enfoques integrados [105, 134, 148, 149, 169, 198, 244]. La granularidad en la cual la BD debe ser fragmentada es una decisión importante que afectará la ejecución de las solicitudes en tiempo de explotación (véase la figura 1.2).

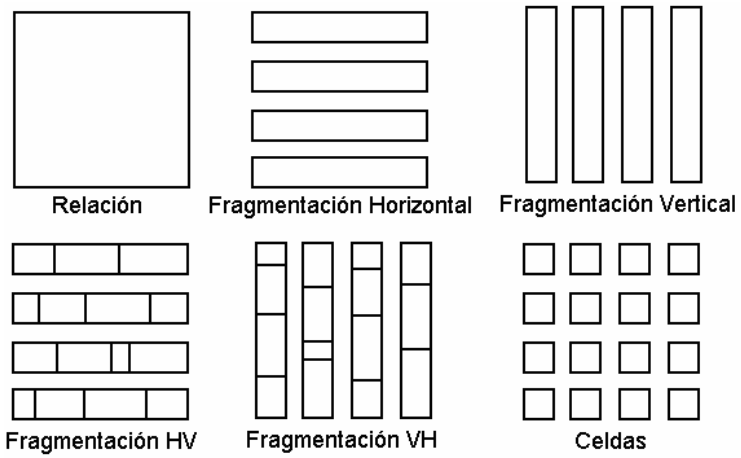

Figura 1.2. Alternativas de fragmentación. Tomado de [169].

El grado de fragmentación va de un extremo, que es la no fragmentación, a otro extremo, que es fragmentar hasta el nivel de tuplas individuales (en el caso de la FH) o hasta el nivel de atributos individuales (en el caso de FV). Tener muy pocas, o demasiadas unidades de fragmentación puede tener efectos adversos. Es necesario encontrar un nivel apropiado de fragmentación que sea un término medio entre los dos extremos. Tal nivel sólo puede ser definido con respecto a las aplicaciones que deben ser soportadas sobre la BD, pero requiere decidir cómo hacerlo.

Existen tres reglas que debe cumplir la fragmentación, las cuales asegurarán la ausencia de cambios semánticos en la BD durante el proceso, logrando así una fragmentación correcta [182].

1. Completitud: Dada un relación R y su descomposición en una serie de fragmentos $F_R = \{f_1, f_2, \cdots, f_n\}$, se considera completa si y solamente si cada elemento de datos en R

se encuentra en uno o varios fragmentos f_i. Esta propiedad asegura que los datos de la relación global se proyectan sobre los fragmentos sin pérdida alguna (idéntica a la propiedad de descomposición sin pérdida de información [75]). En el caso de la FH el elemento de dato, normalmente, es una tupla, mientras que en el caso vertical es un atributo.

2. Reconstrucción: Si una relación R se descompone en una serie de fragmentos $F_R = \{f_1, f_2, \cdots, f_n\}$, la relación R debe poderse reconstruir a partir de sus fragmentos, es decir se puede definir un operador relacional ∇ tal que: $R = \nabla_{1 \le i \le n} f_i$, $f_i \in F_R$. El operador ∇ será diferente dependiendo de las diferentes formas de fragmentación. La reconstrucción de la relación a partir de sus fragmentos asegura que las restricciones definidas sobre los datos en forma de dependencias sean preservadas.

3. Disyunción. Si una relación R se descompone horizontalmente en una serie de fragmentos $F_R = \{f_1, f_2, \cdots, f_n\}$, y un elemento de datos d_i se encuentra en algún fragmento f_j, entonces no se encuentra en otro fragmento f_k ($k \neq j$). Esta regla asegura que los fragmentos horizontales sean disjuntos. Si una relación R se descompone verticalmente, sus atributos de la llave primaria normalmente se repiten en todos sus fragmentos, y la disyunción se define sólo sobre los atributos que no forman parte de la llave primaria.

Diversos investigadores se han dado a la tarea de solucionar los problemas de fragmentación y ubicación; pero a causa de la dificultad que representa el formular el modelo matemático y la fase de solución del modelo, la optimalidad global no ha sido del todo resuelta. Se han publicado diversos modelos matemáticos que utilizan algunas consideraciones y modelan diferentes parámetros, pero estos son ejecutables sólo bajo ciertas especificaciones [182]. Los mejores algoritmos que existen para realizar la fragmentación son iterativos. En general, las aplicaciones necesitan ser caracterizadas con respecto a determinados parámetros, y de acuerdo a sus valores es posible identificar los fragmentos individuales.

Navathe *et al.* [169] propusieron los componentes necesarios para un prototipo de herramienta para el diseño de BDD siguiendo la metodología obtenida por ellos para la fragmentación mixta (FM). Dicho prototipo recibió el nombre de D^3T (acrónimo del inglés Distributed Database Design Tool) y formaba parte de un proyecto de la University of Florida y de la Hong Kong University of Science and Technology. Este prototipo sólo permitía realizar el proceso de FH, FV y FM sin considerar la asignación.

En las implementaciones para solucionar el problema del diseño de la distribución se han utilizado métodos exactos y heurísticos como se observa a continuación. Papadimitriou [189] destacó que el desarrollo de metodologías matemáticas para explicar y predecir el desempeño de los métodos heurísticos es uno de los retos más importantes a los que hoy en día se enfrentan en el área de optimización y algoritmos. Pérez *et al*. [206] hacen una comparación entre los métodos exactos y heurísticos, y proponen criterios de selección para estos, aunque en general no se les brinda ayuda a los diseñadores para tomar la decisión de usar un algoritmo para la FH o FV. Baião *et al.* [12] proponen heurísticas que guían la elección de la técnica de fragmentación a aplicar en cada clase de esquema de BD. Estas heurísticas se implementan en un algoritmo de análisis que es incorporado a una metodología que toma algoritmos de FH y FV de la bibliografía y los adapta. Los resultados experimentales son reportados en [13, 14] aplicados al sistema FORTE [12], y muestran una ejecución más de tres veces más rápida que las alternativas propuestas por otros trabajos reportados en la bibliografía, pero todos estos se refieren a BD orientadas a objetos, no soportadas por la mayoría de los SMBD más populares hoy en día, que soportan ampliamente el modelo relacional. El algoritmo de análisis decide la técnica de fragmentación a usar en cada clase de la BD; y se ofrece su implementación en Prolog. En [147] se hace un análisis del impacto de la fragmentación en el costo de ejecución de las aplicaciones.

Muchos autores han enfocado la generalización de las técnicas de fragmentación a datos semiestructurados, valores complejos y modelos orientados a objetos [11, 13, 15, 18, 19, 21, 22, 48, 61, 83-87, 90, 94-96, 104, 106, 125, 137, 141, 143-150, 200, 202-205, 224, 231], lo cual queda fuera del alcance de esta investigación.

Fragmentación horizontal

La FH, bajo un enfoque relacional, trabaja sobre las tuplas, dividiendo la relación en conjuntos disjuntos mediante el operador de selección σ, que pueden ser unidos para recomponer la relación original. La FH de una relación R produce una serie de fragmentos $F_R = \{f_1, f_2, \cdots, f_n\}$, cada uno de los cuales contiene un subconjunto de las tuplas de R que cumplen determinadas propiedades [182].

El trabajo realizado por Ceri y Pernici [38] plantea parcialmente el problema de distribución de datos con una metodología llamada DATAID-D para el diseño de BDD de forma descendente. DATAID-D sólo considera la FH y ubicación de fragmentos, y obvia la FV. Esta metodología presume que el diseñador tiene la capacidad de

pronosticar las características de las aplicaciones con respecto a los datos y la frecuencia de ejecución de estas aplicaciones desde cada uno de los sitios. Para integrar las BD existentes se ofrece una metodología de diseño ascendente.

Teorey [245] emplea un método heurístico que usa como entrada un esquema ER, elabora un esquema lógico global normalizado, y realiza la FH y la ubicación de estos fragmentos. Pramanik *et al.* [214] proponen dos estrategias para la FH de relaciones recursivas, basado en el concepto de conjuntos de orden parcial. Zhou *et al.* [265] describen un algoritmo de clausura transitiva paralela para fragmentar datos en BD deductivas.

Ezeife y Barker [83, 85] describen un conjunto de algoritmos para fragmentar horizontalmente cuatro modelos de clases diferentes en un sistema distribuido basado en objetos. Posteriormente, Ezeife y Dey [85] proponen un método de refragmentación incremental que usa una actualización de la información de entrada para la obtención de los fragmentos anteriores, y así definir fragmentos más rápidamente.

Navathe y Ra [170] describen un algoritmo no iterativo, basado en teoría de grafos. El mismo utiliza una matriz de afinidad de los predicados simples y los fragmentos finales son definidos procesando la clusterización (clasificación no supervisada) de los predicados simples. Özsu y Valduriez [182] describen un algoritmo iterativo que realiza la FH de una relación usando los predicados mintérminos (minterms), formados a partir de predicados simples extraídos de las aplicaciones de usuarios, como se muestra más adelante.

Como se puede observar, existen múltiples enfoques para realizar esta tarea. En general, son consideradas dos variantes de FH: primaria (FHP) y derivada (FHD). La FHP favorece las operaciones de selección, y la FHD facilita el tratamiento de las llaves foráneas. En esta investigación se le presta especial atención a los trabajos desarrollados por Özsu y Valduriez en [182] puesto que presentan un buen nivel de formalización.

Fragmentación horizontal primaria

Comúnmente, bajo un enfoque relacional, la FHP de una relación se desarrolla empleando el método clásico de uso de predicados definidos en esa relación [19, 149, 182]. Esta variante está orientada a maximizar localidad de procesamiento, y trata de lograr que los datos estén lo más cerca posible de las aplicaciones que las usan. Para realizar la FH es necesario proporcionar información acerca de las relaciones que

componen la BD y de las aplicaciones que las utilizan. No siempre es posible investigar y caracterizar todas las aplicaciones que actuarán sobre la BD, pero al menos se pueden caracterizar las más importantes. Para esto se utiliza la regla 20/80, o sea, se caracteriza el 20 por ciento de las aplicaciones más frecuentes o que realizan las transacciones más críticas que son las que realizan la mayor parte de los accesos a datos [35, 149, 182]. En la caracterización de las aplicaciones es necesario determinar los predicados simples [182] que favorecen la localidad de distribución.

Las aplicaciones emplean predicados simples y predicados más complejos, resultado de combinaciones lógicas de los predicados simples; una combinación especialmente importante es la conjunción de predicados simples denominada predicado minterm [182]. Partiendo de que siempre es posible transformar una expresión lógica en su forma normal conjuntiva, se usan los predicados minterms en los algoritmos para no causar ninguna pérdida de generalidad. Cada predicado minterm puede contener la forma natural o a la forma negada del predicado simple. Es importante señalar que la referencia a la negación de un predicado es significativa para predicados de igualdad de la forma Atributo=Valor. Para predicados de desigualdad, la negación debería tratarse como su complemento [182].

La información cuantitativa relativa a las aplicaciones que es necesaria para la FHP involucra la selectividad de cada predicado minterm y la frecuencia con la que la misma accede a los datos.

La FHP se define sobre un esquema de la BD aplicando una fórmula de selección para obtener los fragmentos minterms que se corresponden con cada predicado minterm. El algoritmo HORIZONTALP expuesto en [182] asume lo antes mencionado y hace uso del método COM_MIN que establece un conjunto de predicados con las propiedades de completitud y minimalidad. Para la completitud es necesario y suficiente que exista una probabilidad idéntica de acceder por cada aplicación a cualquier fragmento minterm. La minimalidad es un estado que indica el grado de influencia de un predicado en el desarrollo de la fragmentación; es decir, el predicado simple debe ser relevante para provocar la fragmentación. Si todos los predicados de un conjunto de predicados simples son relevantes; el conjunto es mínimo. Se puede apreciar como la definición de completitud de un conjunto de predicados simples difiere de la regla de completitud de la fragmentación. Una definición formal de la relevancia se da en [35], aunque siempre es aconsejable acudir a la habilidad y experiencia de los diseñadores antes de utilizar tal

definición formal. En [182] se hace un análisis de la corrección de la FHP realizada con HORIZONTALP de acuerdo a las reglas de fragmentación antes planteadas.

Fragmentación horizontal derivada

Se realiza la FHD a un esquema con al menos una llave foránea que haga referencia a otro esquema fragmentado mediante una FHP. Ceri y Pernici [38] identifican aquellos esquemas que se deben priorizar en el proceso de fragmentación, ya sea por su llave primaria (esquemas propietarios) o por su llave foránea (esquemas miembros). La FHD se define sobre un esquema miembro de acuerdo a una operación de semiacople con su esquema propietario y es por ello que el fragmento resultante se define únicamente sobre los atributos del esquema miembro. Por tanto, si un esquema R tiene una llave foránea (miembro) que referencia al esquema S (propietario), los fragmentos horizontales derivados de R se definen como [182]: $R_i = R \ltimes S_i$, $1 \leq i \leq w$, donde w es el número de fragmentos definidos sobre R, y $S_i = \sigma_{Fi}(S)$, F_i es la fórmula según la cual se define el fragmento horizontal primario S_i.

El algoritmo de FHD es similar a HORIZONTALP tomando como entradas el conjunto de fragmentos del esquema propietario, el esquema miembro y el conjunto de predicados de semiacople $\ltimes$ entre el propietario y el miembro. En un esquema de BD resulta frecuente que existan más de dos enlaces sobre un esquema R. En este caso existe más de una posibilidad de FHD. Para esto se elige la FHD con mejores características de acople, o la empleada en más aplicaciones para facilitar el acceso a los usuarios que hagan mayor uso de los datos, minimizando el impacto total en el rendimiento de un sistema [182]. No es tan sencillo elegir el criterio relacionado con las mejores características de acople, que beneficia a las aplicaciones que hagan operaciones de acople sobre dos esquemas. Al poder realizarlo sobre fragmentos más pequeños y posibilitar la realización de acoples de manera distribuida, se enfatiza la esencia de las BDD. Un sistema puede mejorar sus tiempos de respuesta si, además de estar ejecutando un número de aplicaciones en diferentes sitios, puede ejecutar una aplicación en paralelo.

Fragmentación vertical

La FV no es más que la división de una relación en fragmentos mediante una operación de proyección, cada uno de los cuales contiene un subconjunto de los atributos de la relación original. Su principal objetivo es mejorar el rendimiento de las transacciones;

para esto los fragmentos deben ser diseñados de tal manera que las aplicaciones accedan al menor número de atributos irrelevantes o innecesarios que incrementen los costos de almacenamiento y procesamiento, especialmente cuando el número de tuplas involucradas es muy grande. En una BDD, cuando los atributos relevantes están en diferentes fragmentos de datos y localizados en diferentes sitios, se genera un costo adicional debido al acceso de datos remotos. Por esto, una de las características deseables de una BDD que se debe favorecer con la FV, es la accesibilidad local en cualquier sitio.

Si cada aplicación accediera a un subconjunto diferente y disjunto de atributos, el diseño de la fragmentación vertical sería obvio; sin embargo, en la práctica, lo más común es que las aplicaciones accedan a conjuntos de atributos diferentes y solapados. Un aspecto a decidir es si los fragmentos van a ser solapados o disjuntos. En el primer caso, los fragmentos se adaptan mejor a los requerimientos de las aplicaciones al aumentar las posibilidades de encontrar en un solo fragmento los datos requeridos. Esto aumenta la complejidad de un sistema y los esfuerzos por mantener la consistencia de la BD. En el segundo caso no hay redundancia, pero una determinada fragmentación puede afectar aquellas aplicaciones que al ejecutarse usen atributos de más de un fragmento, siendo necesaria la realización de un acople, operación que es costosa, en particular cuando los fragmentos se encuentran en distintos sitios de la red.

Por tanto, no basta con maximizar el número de aplicaciones que acceden a un solo fragmento y minimizar las que acceden a más de uno, sino que es necesario evaluar cómo influye una determinada fragmentación en el rendimiento total de un sistema. Para esto se debería tener en cuenta toda una serie de aspectos como son: la frecuencia de activación de las transacciones, los métodos de acceso empleados, las estrategias de procesamiento de transacciones, costos de transmisión y posibilidad de replicación, entre otros.

Una solución teórica simple al problema de la FV sería escoger un criterio o función objetivo, evaluarlo para todas las posibles particiones y seleccionar aquella que optimiza el criterio. Esta solución tiene dos grandes dificultades: Primero, la obtención de una función objetivo que convierta todos los aspectos enumerados anteriormente en un modelo matemático razonable; segundo, el número posible de particiones es muy grande, incluso para un número moderado de atributos, de manera que incluso evaluar el criterio más simple para todas las particiones no resulta factible. El número total de

particiones en una relación con n atributos está dado por el n-ésimo número de Bell B(n) [182], donde n es el número de atributos de la relación. Para valores grandes de n, $B(n) \approx n^n$. Esto conduce a la idea, bastante aceptada, de que es inútil buscar soluciones óptimas exactas al problema de la FV, y es necesario recurrir a heurísticas que disminuyan el espacio de solución. Cuando éstas son usadas, se obtiene una solución cercana a la óptima para la partición de los atributos, usando un proceso de minimización paso a paso. En este caso se comienza con una partición y se intenta derivar de ella una nueva partición que sea superior a la original por el hecho de que la BD fragmentada de acuerdo a la nueva partición tendrá un menor costo de procesamiento. Cuando esto es logrado, la heurística trata de obtener mejoras a partir de la nueva partición derivada. Cada vez que tiene éxito mejorar una partición, mejora el rendimiento de la BD. Este proceso continúa hasta que no se puedan hacer mejoras a la última fragmentación obtenida; por tanto, ésta será el resultado de la heurística de fragmentación de atributos. Desde este punto de vista existen dos alternativas: agrupamiento y particionamiento. La primera considera inicialmente cada atributo como un fragmento, y a partir de ahí va uniendo los fragmentos entre sí para obtener las particiones candidatas. La segunda se inicia de un solo fragmento que abarca toda la relación y va dividiéndolo para ir obteniendo los nuevos fragmentos candidatos. Esta última se considera la más apropiada pues la solución óptima debe estar más cerca de la relación original y no de fragmentos compuestos por cada atributo independiente.

El concepto de usar la FV con el objetivo de mejorar el rendimiento de los SMBD ha aparecido con frecuencia en la literatura, a continuación se mencionan algunos de los trabajos que más sobresalen. Estos pueden dividirse en dos grupos: los que buscan una solución óptima del problema y los que usan heurísticas para hallar la solución. En general, la FV proyecta una relación en subconjuntos de sus atributos conservando la llave en cada proyección, que puede ser recompuesta posteriormente mediante el operador de acople (join).

Hoffer y Severance [109] miden la afinidad entre cada par de atributos construyendo una Matriz de Afinidad de Atributos (AA) que sirve de base para agrupar los atributos en clusters usando el Algoritmo de Energía de Enlace (BEA, acrónimo del inglés Bond Energy Algorithm) desarrollado en [155]. Esto permite reducir grandemente el número de fragmentos a evaluar en las fases posteriores del diseño. Este algoritmo se considera apropiado ya que está diseñado específicamente para determinar grupos de elementos

similares y no una ordenación lineal de estos, el agrupamiento final no depende del orden en que los elementos fueron presentados al algoritmo, y la complejidad del algoritmo es razonable, del orden de O(n) donde n es el número de atributos. Se puede decir que la simetría de la matriz AA permite la permutación en parejas de filas y columnas, lo que reduce su complejidad. En dicho trabajo se deja al criterio subjetivo del diseñador la selección de los grupos de atributos que formarán los fragmentos. Además, la similitud entre un par de atributos puede ser inadecuada si no se tiene en cuenta la similitud entre grupos mayores de atributos. Este trabajo sirvió de motivación para la mayoría de los trabajos siguientes sobre fragmentación vertical.

Navathe *et al.* [168] dividen el problema en dos etapas. Primeramente efectúan la fragmentación de las relaciones aplicando diferentes funciones objetivo empíricas que agrupan los atributos extendiendo los trabajos de Hoffer y Severance [109] mediante el algoritmo BEA. Estos algoritmos determinan grupos de atributos en fragmentos solapados y no solapados aplicando FV a tres tipos de BD (distribuidas, sin jerarquía de memoria, y con jerarquía de memoria) con el objetivo de minimizar el número de fragmentos que usa una transacción, y refinar los fragmentos usando factores de costo que reflejan el ambiente físico donde se almacena la BD. Posteriormente se realiza la ubicación replicada o no de fragmentos a sitios, de forma que se maximice el procesamiento local de transacciones aplicando un algoritmo heurístico de tipo goloso (greedy). La metodología empleada es de particionamiento en lugar de agrupamiento.

Cornell y Yu [43] optimizaron el trabajo de Ceri [38] desarrollando un algoritmo para la FV, que obtiene una partición binaria óptima para BD relacionales usando información de factores físicos para disminuir el número de accesos a disco. Posteriores refinamientos son logrados aplicando un algoritmo de partición binaria iterativamente.

Navathe y Ra [170] desarrollaron un algoritmo que usa una técnica gráfica donde se construye un grafo completo denominado "grafo de afinidad" a partir de la generación de un árbol linealmente conectado a partir de la matriz AA, en el cual los nodos representan los atributos y los vértices la afinidad entre ellos. Entonces, formando un árbol linealmente conectado, el algoritmo genera todos los fragmentos en una iteración considerando los ciclos como fragmentos. En este algoritmo no hay necesidad de partición binaria iterativa. La mayor debilidad de esta última es que en cada paso se generan dos nuevos problemas lo que incrementa la complejidad. Además, no se requieren algoritmos complementarios como el encargado del corrimiento de las filas y

las columnas. La complejidad del algoritmo es $O(n^2)$, que mejora el $O(n^2 \log n)$ de los algoritmos anteriores. La principal desventaja de este algoritmo es que usa como heurística una función objetivo intuitiva la cual no está explícitamente cuantificada.

Öszu y Valduriez [182] discuten este trabajo previo sobre FV en BDD usando información de las frecuencias de acceso y aplican el algoritmo BEA. Los grupos de atributos son clusterizados y se usan ecuaciones de costo para definir la mejor posición a lo largo de la diagonal de la matriz clusterizada para dividir la relación en fragmentos. Los factores de costo reflejan el ambiente físico en el que los fragmentos van a estar almacenados.

Muthuraj *et al.* [167] presentan un trabajo donde se argumenta que los primeros algoritmos para la FV son *ad hoc*, por eso se propone una función objetivo llamada Evaluador de Particiones para determinar la calidad de las particiones generadas por varios algoritmos. Dicho evaluador consta de dos términos: los costos por accesos a atributos locales irrelevantes y los costos por acceder a atributos remotos relevantes. El primero mide los costos del procesamiento local de las transacciones debido a accesos a atributos irrelevantes, y el segundo mide los costos del procesamiento remoto debido a las transacciones remotas que acceden a atributos relevantes. Se estudia explícitamente el problema referente a la FV n-aria. Esta fragmentación se efectúa con distintos propósitos, como por ejemplo en los SBDD, en la cual se crean fragmentos para ubicarlos en los diferentes sitios de la red, o también en los SBD centralizadas a fin de ubicar los fragmentos dentro de diferentes jerarquías de memoria.

Du *et al.* [74] proponen un enfoque para la FV basado en algoritmos genéticos como opción para dar solución a este problema de optimización. Se aplican restricciones en la manipulación de los cromosomas para excluir aquellos redundantes durante el procesamiento del algoritmo genético, de forma que sea factible el cruzamiento y mutación orientado a grupo. Se reportan mejorías en la velocidad de fragmentación y en los resultados, especialmente en problemas de fragmentación vertical de gran tamaño respecto al enfoque de algoritmo genético clásico.

En resumen, la FV se usa durante el diseño de la BDD para mejorar el rendimiento de las transacciones, incrementar el procesamiento local y minimizar el acceso a datos remotos, aumentar el paralelismo durante la ejecución de las consultas, la concurrencia y el rendimiento. La fragmentación vertical no sólo es beneficiosa en el caso de las BDD, también es aplicable en otros contextos como son: BDC con jerarquía de

memoria, BDD orientadas a objetos, BD deductivas distribuidas, entre otras. Las propuestas presentadas por Özsu y Valduriez [182] han servido de inspiración a esta investigación por el grado de formalización e integración que presentan y la aplicabilidad computacional de las mismas.

Fragmentación mixta

Existe otra estrategia de fragmentación denominada mixta (FM), donde simultáneamente se aplican ambos tipos (FH y FV) al mismo esquema. Esta estrategia no es considerada básica, pero es obvio que muchas particiones de la vida real pueden ser mixtas. La FM es generada a través de la aplicación recursiva de operadores del álgebra relacional en los fragmentos.

La investigación desarrollada en 1988 por Apers [9] determina la FM de los esquemas con base en el ECG, las consultas y sus frecuencias de uso. Este tipo de fragmentación demanda experiencia del diseñador, pues produce fragmentos de granularidad muy fina y se requiere realizar agrupaciones.

El trabajo de Lim y Ng [136] reporta un enfoque mixto para la fragmentación de BD deductivas distribuidas, mientras que Navathe *et al.* [169] publican una metodología para llevar a cabo la FM de diseños iniciales de BD basada en la teoría expresada en [39, 43, 168, 170] y mediante el uso de grafos implementa un algoritmo que obtiene una solución a la FV con mayor eficiencia, con una complejidad del orden de $O(n^2)$, siendo n el número de atributos. Para la FH se hace uso de los predicados de las transacciones más importantes (según la regla 20/80) en la creación de una matriz de afinidad de predicados que sirve para conformar un grafo, que con el apoyo de heurísticas permite obtener los FH o celdas que se deriven del uso de esos recursos. La complejidad de este algoritmo también es del orden de $O(n^2)$, siendo n el número de predicados. El proceso concluye con la mezcla de celdas (véase la figura 1.2) en un sentido ascendente, opuesto al descendente que se siguió para la obtención de las celdas de particiones.

Como se ha podido observar, para el diseño de la fragmentación se necesita una gran variedad de información difícil de obtener, y demanda experiencia en los diseñadores; así como los métodos para dar solución a los problemas planteados son de una elevada complejidad computacional, por lo que se sugiere el uso de heurísticas que permitan obtener soluciones con menor esfuerzo.

Asignación

La segunda parte del diseño de distribución, la asignación o ubicación, típicamente es considerada independiente de la fragmentación. La ubicación de fragmentos en una red de comunicación es un tema importante, ya que tiene que ver con el desempeño global de los SBDD. El problema de la asignación de datos, según [182] se define de la siguiente manera: Suponiendo que existe un conjunto de fragmentos $F = \{f_1, f_2, ..., f_n\}$ y una red de computadoras compuesta por sitios $S = \{s_1, s_2, ..., s_m\}$ sobre los cuales se ejecuta un conjunto de aplicaciones $A = \{a_1, a_2, ..., a_q\}$. El problema de la ubicación consiste entonces, en encontrar la distribución óptima de F en S.

La optimalidad puede ser definida con respecto a diversas medidas, entre las principales se encuentran las dos siguientes [182]:

1. Costo Mínimo: El problema de la ubicación 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.
2. Desempeño: La estrategia de ubicación es diseñada para mantener una 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.

Sobre el problema de asignación en BD se han publicado numerosos trabajos que intentan reducir la complejidad del problema [9, 10, 18, 28, 52, 56, 62, 70, 89, 104-106, 114, 116, 126, 134, 135, 138, 140, 148, 149, 151, 156, 191, 200-204, 207, 215, 216, 224, 229, 232, 233, 242, 244, 248, 260]. 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 el cual aplica un algoritmo que 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 [166] y asignación de BD [264]. Posteriormente el problema de asignación fue formulado como problema de programación en enteros binaria [62] e incluso se ha integrado con mecanismos específicos de control de concurrencia [216, 244]. 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 [232].

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 [53] desarrolló una teoría de asignación de fragmentos y diseñó un algoritmo de flujo de red para resolver el problema de ubicación en BDD. Ya que es conocido que la complejidad de este problema es NP-Completo [12, 182, 201, 203], varios autores han propuesto soluciones que reducen la complejidad mediante algoritmos heurísticos como los presentados en [34] para dar solución al problema de la mochila y en [88] usando las técnicas de ramificación y acotación.

Ceri *et al.* [35] 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 [38] propusieron un método simple que primeramente ignora la replicación buscando una solución óptima no replicada, y luego aplica un algoritmo

goloso 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 [44] 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 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.* [215] 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 el problema de tener copias replicadas.

Chaturvedi *et al.* [56] 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.* [140] presentan algoritmos para la asignación de datos para obtener costos totales de comunicación mínimos. Después, Lin y Orlowska [138] 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 [151] 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 [9]) y cada aplicación es descompuesta en operaciones de 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 de un sistema, de comunicación, de operaciones entrada y salida a disco, de procesamiento de CPU y de almacenamiento; tomando en cuenta las restricciones de capacidad de cada nodo. Park y Baik [191] 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.* [258-261] se presentan las estrategias de asignación de datos considerando que las condiciones cambian

dinámicamente. Daudpota *et al.* [70] 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 (LAN/WAN acrónimo del inglés Wide Area Network). Bellatreche *et al.* [18] 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.* [114] 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. Considera que 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 compleja 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. Todos los fragmentos accedidos por una aplicación son considerados independientes, lo cual no es realista. Además, ignora información sobre los sitios como son la capacidad de almacenamiento y procesamiento, y sólo es aplicable a redes LAN. También desvaloriza los costos de procesamiento de CPU y entrada/salida al minimizar el costo total en un ambiente de redes WAN.

Chang [54] presenta métodos de solución heurísticos que permitan 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 usa la técnica de búsqueda "divide y vencerás" para asignar clusters a los sitios en la red tal 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. Para cada una de las fases, se van presentando los resultados a través de un ejemplo práctico.

Hababeh *et al.* [106] 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 [224] plantea un nuevo 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 binaria no lineal, aunque puede ser convertido a un modelo de programación lineal binario cuya solución es de igual complejidad al planteado en [182].

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.

Asignación no redundante

El trabajo realizado por Ceri en [38] plantea el problema de ubicación de fragmentos con la metodología DATAID-D donde ubica los fragmentos en el sitio donde se presente una frecuencia de acceso mayor, de esta forma obtiene una solución no replicada. Cuenta con un algoritmo goloso para obtener la ubicación replicada de los fragmentos de acuerdo a la solución no replicada.

Saccà y Wiederhold [225] 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 (acrónimo del inglés Greedy First Fit). Inicialmente este algoritmo selecciona los fragmentos mediante el algoritmo goloso, 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.* [133], 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 sólo 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.

Asignación redundante

La replicación de datos es un tema cada vez más importante para la comunidad de investigadores en BD y sistemas distribuidos. Hace pocos años las organizaciones no consideraban el uso de la replicación para la distribución de datos, pero ha ido creciendo el interés en este tema debido a que los negocios cada vez son más descentralizados y distribuidos geográficamente, y la gerencia sigue necesitando acceder de forma coherente a los mismos datos como si aún estuvieran centralizados, y se trabaja con grandes volúmenes de datos que no pueden ser fácilmente recuperados o restituidos en caso de fallas. 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 abordan las posibilidades de materialización del diseño de BDD mediante mecanismos de 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 [171]. 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 [87], que los agentes de replicación registren las transacciones, supervisen las actualizaciones, garantizando coherencia y consistencia [113]. Los algoritmos que garantizan una fuerte consistencia se basan en protocolos de bloqueos y confirmación (commit) [24]. Respecto al soporte para replicación de información, o no se ofrece o se hace a través del protocolo sincrónico (eager) [121, 127, 193] ROWA, que brinda una fuerte consistencia. Este protocolo sincrónico no es apropiado para cluster de BDD 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) [182], y es poco escalable [187]. Por tanto, la replicación sincrónica no es un enfoque viable en la mayoría de los ambientes de procesamiento de datos actuales [4]. Por otra parte, la replicación asíncrona (lazy) [3, 4, 30, 68, 69, 76, 110, 120, 121, 183-185, 188] sacrifica consistencia por lograr mejor desempeño, tratada en ocasiones como optimista o preventiva [47, 186, 187]. Pacitti *et al.* [185] proponen un algoritmo que asegura corrección para una configuración asíncrona pero no consideran la replicación parcial, como sí lo hacen en [187], aunque tanto estos dos trabajos como [186] incrementan el tráfico en la red. El algoritmo reportado en [123] ofrece una fuerte consistencia y preserva la autonomía de los SMBD locales, mas requiere que las transacciones actualicen una copia primaria fija por lo que cada tipo de transacción está asociada a un solo sitio y sólo puede ser ejecutada en ese sitio. Esta es una desventaja para las aplicaciones que hacen muchas actualizaciones. Plattner y Alonso [211] presentan un algoritmo de copia primaria que separa las transacciones de actualización de las de sólo-lectura donde las de actualización se enrutan al sitio de la copia primaria mientras que las otras se enrutan a cualquier copia disponible. Wu y Kemme [262] proponen una buena solución que no necesita declarar por adelantado ninguna propiedad para las transacciones. Akal *et al.* [3, 4] presentan un protocolo que asume la existencia de copia primaria conjuntamente con planificaciones globales serializables, lo cual es difícil de lograr en configuraciones prácticas de BD.

Fan y Lynch [86, 87] presentan un algoritmo de replicación de datos distribuidos ajustado especialmente a datos de lectura 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 [114] 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.* [32] 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. Se demuestra que un sistema clusterizado funciona mejor que uno replicado si se usa una gran cantidad de servidores de solicitudes, principalmente debido a la reducción de carga de la red. En [27, 33, 45-47, 97, 98, 102, 122, 123, 131, 132, 186, 187, 192, 251] 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 SMBD). Esto tiene la ventaja de preservar la autonomía de las BD y evitar costosas migraciones de datos a BD paralelas [97, 100, 101].

El reto de la replicación es el control de réplicas para mantener las copias consistentes. Muchas de estas soluciones realizan el control de réplicas en una capa intermedia por encima de las réplicas de la BD. Adicionalmente, algunos sistemas tienen requisitos rigurosos en las transacciones (por ejemplo: declarar por adelantado todos los datos que serán accedidos).

El artículo de Lin *et al.* [141] presenta un esquema de replicación basado en capa intermedia (middleware) que provee el conocido nivel de aislamiento instantáneo (snapshot isolation) [69, 77, 141, 262] al mismo nivel de granularidad de tupla que los SMBD como el PostgreSQL y ORACLE sin tener que declarar por adelantado ninguna propiedad de las transacciones. Tanto las transacciones de sólo-lectura como las de actualización pueden ser ejecutadas en cualquier réplica garantizando la consistencia de los datos en cualquier momento. Este enfoque asegura lo que llaman aislamiento instantáneo de copia primaria (1-copy-snapshot-isolation) si las réplicas de la BD subyacente ofrecen el aislamiento instantáneo. Este enfoque fue implementado como una capa intermedia encima de las réplicas de PostgreSQL. Mediante una interfaz ODBC, esta capa intermedia es completamente transparente a los programas de usuario. En este trabajo [141] se usan las réplicas para tener copias redundantes como estrategia proactiva y lograr mayor disponibilidad ante fallos. La mayoría de los SMBD

comerciales ofrecen soluciones a través del enfoque sincrónico, que para lograr mayor disponibilidad, actualiza todas las copias disponibles antes de realizar el commit (lo cual demora la ejecución de las transacciones). También se usa el enfoque asíncrono, que para lograr accesos locales más rápidos y escalabilidad, actualiza todas las copias disponibles después de realizar el commit (por lo que pueden aparecer problemas de consistencia si algún sitio falla antes de que se propaguen todos los cambios). Muchos autores han vuelto a analizar el tema de replicación y han propuesto soluciones para eliminar los problemas asociados con el control de réplica [6, 7, 23, 26, 27, 30, 31, 33, 65, 68, 69, 102, 110-112, 123, 127, 141, 183-186, 188, 192, 193, 195, 196, 223]. Un enfoque común es la replicación basada en clusters con una propagación híbrida que es sincrónica y asíncrona [121]. Por otra parte, muchas implementaciones tanto de la industria como de la investigación basan su control de réplica en una capa intermedia [17, 33, 118, 141, 161, 193, 223] entre los clientes y las réplicas de la BD.

Por su parte, Lin y Veeravalli [137] 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. El diseño de BDD para sistemas de tiempo real no es de interés en esta tesis.

Lee y Baik [134] proponen un algoritmo heurístico que resuelve el problema de asignación replicada de datos así como de transacciones en BDD. 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 se ha observado, los algoritmos de fragmentación deciden cómo dividir los esquemas teniendo en cuenta los accesos de las aplicaciones a los mismos, pero los modelos de asignación ignoran esto. Por lo tanto, los modelos de asignación tienen que incluir una vez más la especificación detallada de las interrelaciones entre los fragmentos y cómo las aplicaciones acceden a ellos.

Soluciones que combinan la fragmentación y la asignación

Se reportan pocas investigaciones donde se traten las tareas de fragmentación y ubicación conjuntamente, ya que las metodologías que lo soportan tienden a ser muy

complejas, predominantemente teóricas, y de alcance limitado desde el punto de vista práctico [244]. Algunas de estas son referidas en [114, 198, 199, 253, 254] sobre el modelo de Fragmentación, Ubicación y Reubicación Dinámica de Datos (FURD [199]), que es un modelo matemático de programación lineal entera binaria (no replicada) que se encarga de combinar los procesos de FV y ubicación de atributos a sitios. Para resolver este modelo se utilizó, en primer término, 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 [198, 201, 206-208, 254] donde reportan métodos aceptables de solución heurística aplicados al problema de distribución en BDD [198], el cual ha sido resuelto parcialmente con diferentes modelos y técnicas [15, 35, 38, 125, 149, 151, 182, 227, 228, 258] mediante dos pasos: primero la obtención de los fragmentos y luego la ubicación de los mismos en los sitios. Este modelo 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, aún 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 [158] 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, no el de diseño inicial de una BDD; por lo que no es de interés en este trabajo.

Tamhankar y Ram [244] 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 SMBDD que no son satisfechas totalmente por los SMBDD 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 SMBDD respecto a capacidades de almacenamiento, creación de índices, es decir, sólo 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. Según los autores, trabajarán en extender la metodología a otras arquitecturas como la cliente-servidor, aunque no se encontró ninguna publicación al respecto.

Hababeh *et al.* [105] 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 [106].

Otro trabajo más reciente es el de Ma *et al.* [149], 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 [182] 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. Desde un punto de vista pragmático se considera la regla 20/80. El enfoque del trabajo de Ma *et al.* [149] no es apropiado para el problema enfocado en esta investigación puesto que es aplicado a BD no relacionales.

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 que ayuden a los diseñadores humanos, en lugar de intentar reemplazarlos, como una forma muy apropiada de resolver el problema del diseño de BDD. Estos métodos deben ser computacionalmente tratables.

1.5. Arquitectura para las bases de datos distribuidas

Todo el desarrollo de la tecnología de BD ha estado caracterizado por lograr altos grados de independencia entre los programas de aplicación y los aspectos internos de manipulación y representación. Los niveles de transparencia asociados con capas en la arquitectura de las BDD son importantes para comprender las peculiaridades de los procesos distribuidos. Los niveles básicos de la arquitectura de referencia son: conceptual, lógico y físico.

Para comprender los niveles de transparencia se tiene una arquitectura básica muy general organizada en esquemas (véase la figura 1.3):

1. Esquema global, que comprende el nivel conceptual y describe los datos como si la BD no estuviera distribuida, pudiéndose usar cualquier modelo conceptual o lógico global de los conocidos para BD centralizadas, por ejemplo el modelo ERE. Sin embargo, es conveniente tener en cuenta que debe ser adecuado para las transformaciones hacia otros niveles.

2. Esquema de fragmentación, que abarca el nivel lógico y divide los esquemas globales en fragmentos.

3. Esquema de asignación, que define en qué sitios se ubican los fragmentos, lo cual puede hacerse con réplicas o no. Todos los fragmentos correspondientes a una misma relación global que son asignados al mismo sitio constituyen la imagen física de la relación en el sitio que cubre el nivel físico.

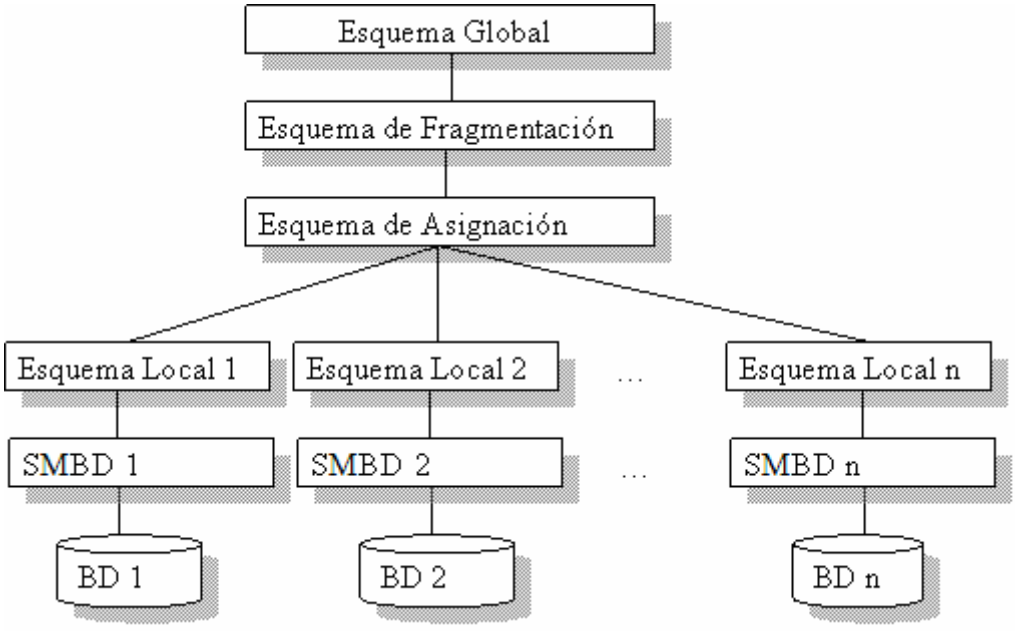

Figura 1.3. Arquitectura de referencia para las BDD. Tomado de [182].

Los primeros 3 niveles de la arquitectura no dependen de los sitios y por tanto, tampoco de los SMBD locales a los sitios. En el nivel más bajo, sin embargo, es necesario transformar las imágenes físicas en objetos manipulables por los SMBD locales. A este proceso se le llama proceso de transformación local.

Tanto en las BDC como en las BDD, el diseño de las aplicaciones se hace después que el diseño de los esquemas, no obstante en BDD es necesario tener información acerca de sus requerimientos, pues los esquemas deben servir de soporte a las mismas. Los requerimientos de las aplicaciones sólo se refieren a las más importantes e incluyen: sitio desde el cual se emite una aplicación (conocido como sitio de origen de la aplicación), frecuencia de activación de la aplicación en cada sitio (cantidad de veces que se activa en una unidad de tiempo); y cantidad, tipo y distribución estadística de los accesos de cada aplicación a cada dato requerido. Más adelante, en el capítulo 2, se trata en detalle lo relativo a todos los requerimientos de información para el proceso de diseño de distribución.

La arquitectura mostrada proporciona un lineamiento muy general para comprender las BDD. Los tres objetivos que motivan las facilidades de esta arquitectura son:

1. Separación entre la fragmentación y la asignación: Esta separación permite distinguir dos niveles diferentes de transparencia: transparencia de fragmentación y de asignación. La transparencia de fragmentación es el grado más alto de transparencia y posibilita que los usuarios o programadores de aplicación trabajen sobre relaciones globales. La transparencia de asignación es un nivel más bajo de transparencia y requiere que los usuarios trabajen sobre fragmentos en lugar de relaciones globales, sin embargo no tienen por qué saber donde los fragmentos son ubicados.

2. Control de redundancia: En la descripción de las imágenes físicas puede haber solapamiento de fragmentos (o réplicas). Esta redundancia, como se expresó anteriormente, debe ser controlada y es útil para varios aspectos de un manejo de BDD.

3. Independencia de los SMBD locales: También se conoce como transparencia de transformación local. Permite estudiar diversos problemas de la BDD sin tener en cuenta los modelos específicos a los SMBD locales, se reduce la complejidad si el sistema es homogéneo.

Otro tipo de transparencia que se deriva de la transparencia de asignación es la transparencia de réplicas y aunque puede considerarse como consecuencia una de la otra, en ocasiones puede ser necesaria su diferenciación.

La transparencia de distribución no constituye un problema, pues se puede tener instalado el SMBDD en cada servidor donde se encuentren almacenados los datos. El diseño de la BD representa de manera implícita que un sistema distribuido responde a la estructura de la organización de forma que los programas de aplicación se ubican en los lugares donde son necesarios. Es un enfoque común asumir que la red ha sido diseñada y por tanto sólo se enfatiza en la distribución de los datos. De los aspectos anteriormente mencionados, son objeto de análisis en la presente tesis aquellos relacionados con el diseño de la BD, en particular, el diseño de esquemas globales, su transformación a esquemas de fragmentación, de ahí a los esquemas de asignación y por último la obtención de esquemas locales con replicación.

Durante el diseño deben observarse los siguientes objetivos o aspectos que se desean priorizar:

1. Localidad del procesamiento.
2. Disponibilidad y confiabilidad de los datos distribuidos.
3. Distribución de la carga de trabajo.
4. Disponibilidad y costo de almacenamiento.

El diseño de BDD persigue distribuir los datos atendiendo a la maximización de la localidad del procesamiento, que responde al principio de colocar los datos tan cerca como sea posible de las aplicaciones que los usan. La forma más simple de caracterizar la localidad es considerar dos tipos de referencias: locales y remotas. Diseñar una distribución que maximice localidad o lo que es lo mismo, minimice los accesos remotos del procesamiento, puede hacerse añadiendo la cantidad de referencias locales y remotas correspondientes a cada fragmentación candidata, y asignar los fragmentos eligiendo la mejor solución. Este criterio de optimización no sólo es ventajoso para reducir los accesos remotos, sino que además coopera con el principio de autonomía local, ya que cuando una aplicación es completamente local se simplifica su control.

A menudo, el diseñador se enfrenta con el problema de que varias aplicaciones necesitan acceder a los mismos datos desde diferentes lugares. En este caso, el diseño más efectivo es el que asegura la localidad al número más grande de aplicaciones. Una suposición básica de este trabajo es que el diseñador de la BD es capaz de predecir ambas propiedades lógicas que caracterizan la localidad de los datos con respecto a las aplicaciones e información cuantitativa que mide la carga de trabajo de aplicaciones, en términos de frecuencia de la ejecución de las peticiones en cada sitio. De hecho, la

dificultad del diseño de la distribución de la BD reside en la interferencia de consideraciones lógicas y cuantitativas.

La disponibilidad y confiabilidad de datos distribuidos son ventajas de los sistemas distribuidos. Un alto grado de disponibilidad se logra para aplicaciones de sólo-lectura si se almacenan múltiples copias o réplicas de la misma información, de forma que un sistema tenga alternativas de solución si alguna de ellas no está disponible. La confiabilidad también se logra mediante la existencia de múltiples copias, pues es posible recuperar copias dañadas o destruidas a partir de otra. Por supuesto que como los daños pueden obedecer a catástrofes físicas, deben tenerse copias en lugares geográficamente separados. De aquí se puede observar la necesidad del uso de réplicas. El tema de replicación ha sido abordado desde múltiples enfoques [3, 4, 6, 7, 16, 27, 29-31, 45, 47, 68, 69, 71, 77, 79, 80, 86, 87, 91, 110-113, 117-121, 127, 131, 137, 141, 156, 161, 164, 165, 173, 183-188, 192, 193, 195, 196, 211, 222, 257, 259, 261, 262]. Estos trabajos se destacan en el análisis, diseño y optimización de algoritmos para el control de réplicas. Ninguno de ellos trata la posibilidad de usar la replicación (por ejemplo, con opciones de filtrado) para materializar diseños distribuidos de datos, sólo para mantener copias en aras de lograr mayor disponibilidad y tolerancia ante fallos.

La distribución de la carga de trabajo sobre los sitios se hace sobre la base de utilizar la potencia de los computadores de cada sitio y maximizar paralelismo en la ejecución de las aplicaciones. Como que la distribución de la carga puede afectar la localidad del procesamiento, es necesario correlacionar ambos objetivos.

En cuanto a la disponibilidad y costo de almacenamiento, la capacidad de almacenamiento de cada sitio debe tenerse en cuenta; usualmente el costo de almacenamiento no es importante comparado con otros costos, no obstante las limitaciones de almacenamiento deben ser considerados.

Atender a todos los criterios simultáneamente es difícil pues conduce a modelos de optimización muy complejos, y con frecuencia se consideran como restricciones algunos objetivos y también se priorizan otros.

1.6. Conclusiones parciales

Aunque muchos investigadores han propuesto modelos y han diseñado algoritmos para el diseño de 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. Sin embargo, las dimensiones de las aplicaciones actuales exceden normalmente las capacidades del diseñador para realizar un diseño de distribución adecuado en forma manual. También se ha detectado una aparente ausencia de métodos para llevar los esquemas lógicos locales a esquemas físicos locales, así como la toma de decisiones respecto a cuánto replicar y cuáles relaciones fragmentar mediante FHD. Es por esta razón que se justifica el desarrollo de nuevas metodologías y herramientas de ayuda al proceso de diseño.

El aislamiento de la fragmentación y los pasos de la ubicación simplifican la formulación del problema de distribución, reduciendo el espacio de decisión, aunque un análisis cuidadoso revela que aislar los dos pasos contribuye a la complejidad de los modelos de ubicación.

Tanto la fragmentación como la asignación requieren variada información acerca de los esquemas de la BD, las aplicaciones, los sitios y las redes de comunicación, pero cada cual ignora cómo la otra captura y usa esta información. Cualquier metodología que lo intente puede ser muy compleja.

Como resultado de varios años de investigación y desarrollo, en la actualidad, los SMBDD comerciales presentan nuevos y más sofisticados servicios que facilitan la distribución de los datos. Tal es el caso del Microsoft SQL Server [113] y Oracle [180] que desde hace casi una década ofrecen una variedad de tecnologías de replicación que permiten el intercambio de información entre servidores manteniendo la consistencia.

2. HERRAMIENTAS DE AYUDA AL DISEÑO DE BASES DE DATOS DISTRIBUIDAS.

Cualquier ayuda al diseño de BDD debe integrar metodologías prácticas de diseño que sean correctas y completas; y la implementación de herramientas que cooperen entre sí. En este capítulo se proponen soluciones dadas desde el punto de vista teórico o práctico a las problemáticas de diseño identificadas. Se comentan algunas investigaciones recientes que permitieron su perfeccionamiento y su implementación computacional. Los métodos seleccionados se analizan y se implementan con resultados confiables en un tiempo de respuesta aceptable. Desde el punto de vista metodológico se exponen los resultados siguiendo los tres niveles de modelación de la arquitectura de referencia: conceptual, lógico y físico.

2.1. Arquitectura de las herramientas de ayuda al diseño de BDD

La obtención de la gran cantidad de información requerida para el diseño de BDD es una tarea difícil que requiere de tiempo, experiencia y esfuerzo. Esto ha sido reducido mediante la integración a nivel de procesos, donde los datos apropiados para cada proceso del flujo de trabajo (representado de modo abstracto en la figura 2.1), son obtenidos del catálogo del diseño, y a su vez cada herramienta coloca sus salidas en éste, para que sirvan de entrada a los procesos siguientes. Esta integración puede ser negativa porque provoca dependencias que incrementan la complejidad y pueden cometerse errores. Sin embargo, estas dependencias son positivas porque ahorran una gran cantidad de tiempo y esfuerzo.

Como se pudo apreciar en el capítulo 1, el diseño de BDD involucra varias tareas integradas en un proceso para obtener diseños correctos con el menor tiempo posible. El Sistema Integrado de Ayuda al Diseño de BDD, denominado SIADBDD (véase el anexo 10), fue desarrollado con este fin, integrando un conjunto de herramientas que se explican en detalle más adelante en este capítulo. La herramienta ERECASE permite la caracterización de los ECG; APPWIZARD se encarga de la caracterización de las aplicaciones; NETWIZARD recopila información sobre la red de comunicación y los

sitios de procesamiento donde residirá la BDD objeto de diseño; FRAGMENTER realiza la fragmentación y ALLOCATOR se encarga del diseño lógico y la materialización del diseño físico. El último paso de diseño es la optimización local seguido de monitoreos y ajustes que no son objetivo de esta investigación.

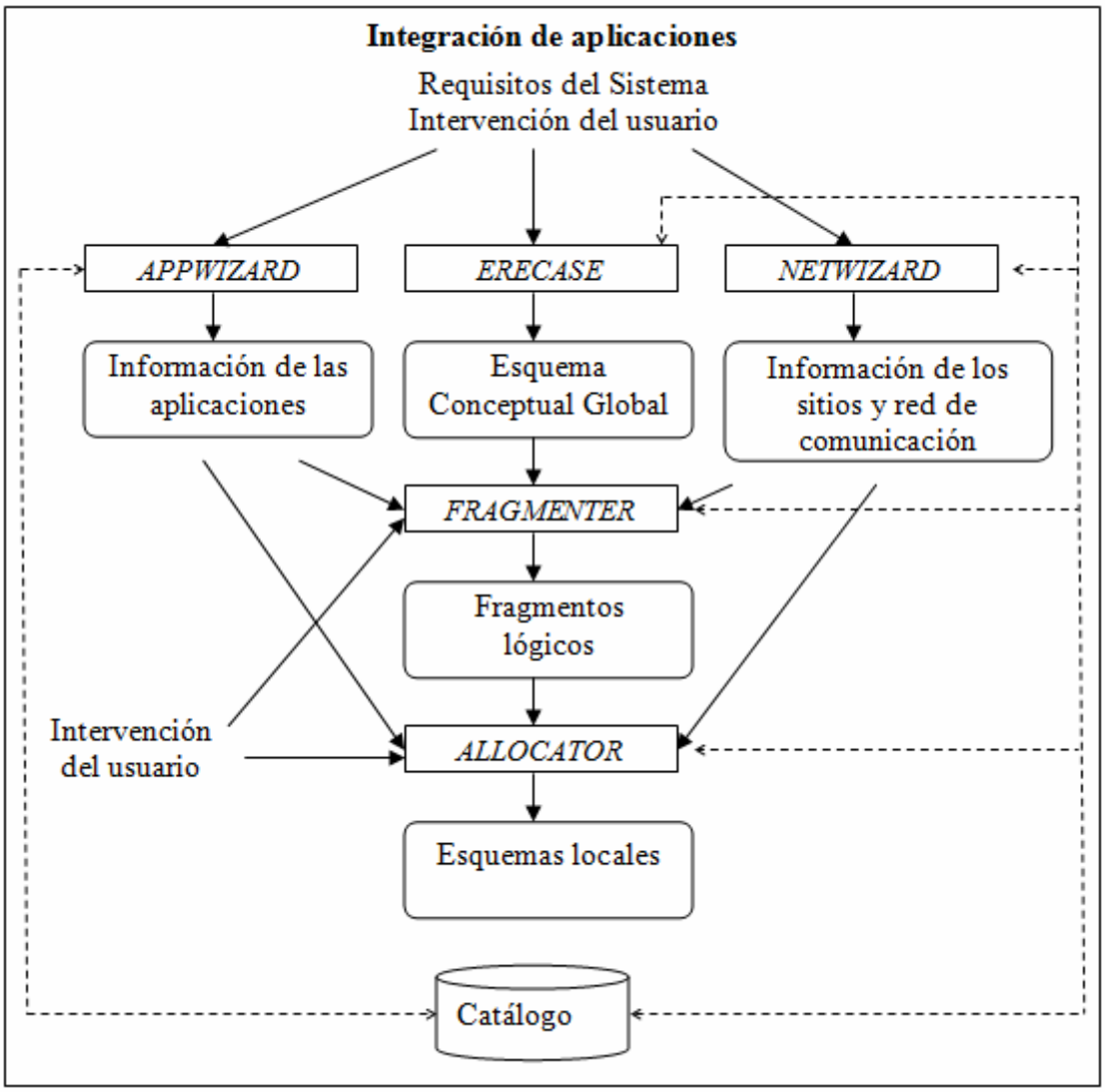

Figura 2.1. Integración de herramientas a través del flujo de trabajo de diseño de BDD.

Una herramienta de ayuda al diseño de BDD debe mantener una gran cantidad de información estructurada referente a los esquemas que conforman cada BD objeto de diseño, sobre las aplicaciones que dan solución a los casos de uso o solicitudes, sobre los sitios donde residirán los datos y el procesamiento de las solicitudes, y sobre la red que soporta la conexión entre sitios y el intercambio entre estos a fin de efectuar sus funciones. Toda esta variedad de información es almacenada usualmente en un catálogo que mantiene SIADBDD para sus propios fines internos. A través de este catálogo se establece la interoperabilidad entre las herramientas, permitiendo las funciones de

búsqueda, obtención, transferencia y evaluación de la información almacenada para llevar a término un proyecto de diseño de BDD.

Para la descripción arquitectónica de SIADBDD se toma en consideración la práctica recomendada por la IEEE para sistemas intensivos de software, estándar 1471-2000 [115]. Según este estándar, el paradigma arquitectónico debe ser flexible como para contemplar las funciones relevantes y la especificación de cada punto de vista seleccionado para organizar la vista arquitectónica y las motivaciones para ello. El mejor modelo que se amolda a este requisito es el que integre los distintos niveles de la arquitectura de referencia de las BDD integrada al proceso de diseño (véase la figura 2.2). La implementación de las herramientas fue realizada en Visual C# como parte de Visual Studio 2005 [217, 255].

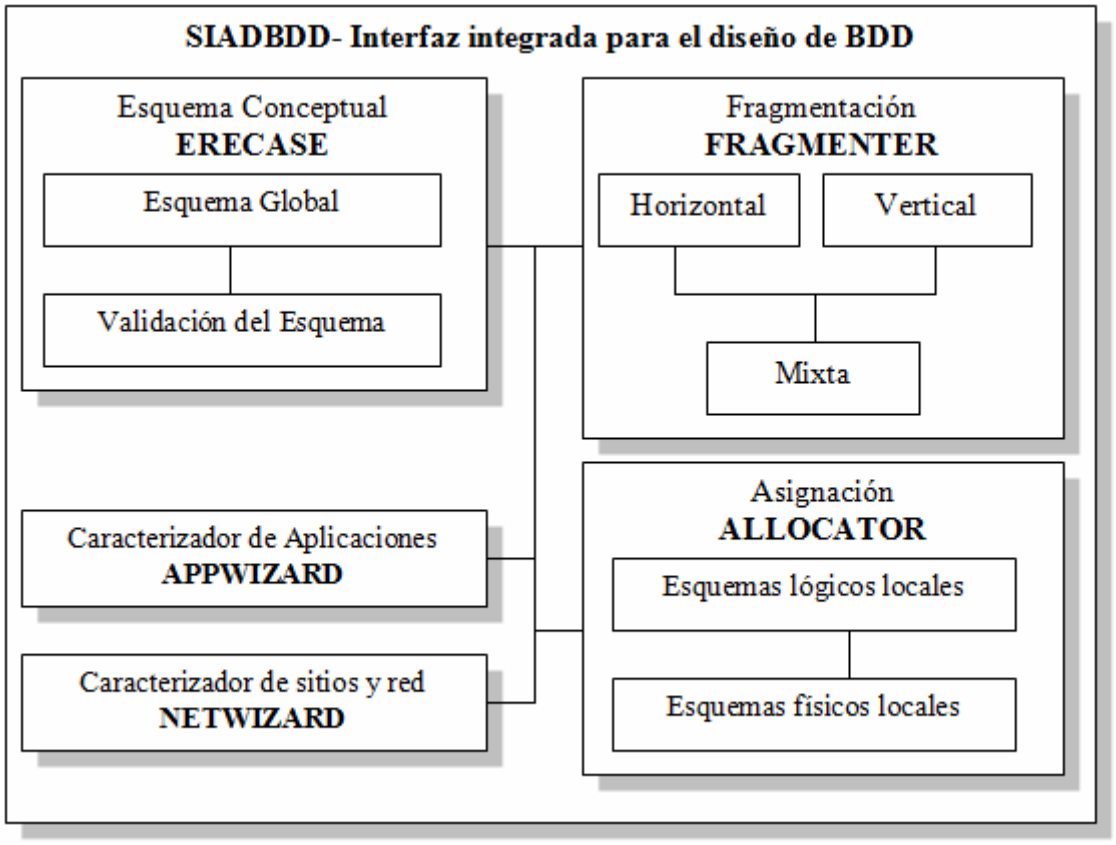

Figura 2.2. Vista arquitectónica general de la herramienta de ayuda al diseño de BDD.

2.2. Herramientas para la modelación de BDD

Desde el punto de vista metodológico se siguen los tres niveles de modelación de la arquitectura de referencia planteada en el capítulo 1. El nivel conceptual es abordado por los asistentes ERECASE, APPWIZARD y NETWIZARD introducidos en la sección 2.1; el nivel lógico es emprendido por FRAGMENTER y ALLOCATOR,

mientras que el nivel físico se logra mediante réplicas que se materializan al ejecutar cada una de las secuencias de comandos (scripts) generados por ALLOCATOR en cada uno de los SMBDD de cada uno de los sitios registrados por NETWIZARD.

Una suposición básica de este trabajo es que el diseñador de la BD es capaz de predecir las propiedades lógicas que caracterizan la localidad de los datos con respecto a las aplicaciones y la información cuantitativa que mide la carga de aplicaciones en términos de frecuencia de ejecución de las peticiones en cada sitio. De hecho, la dificultad del diseño de la distribución de la BD reside en la interferencia de consideraciones lógicas (cualitativas) y cuantitativas.

2.2.1. Modelación conceptual

Tomando como base la vista arquitectónica de la figura 2.2, se deduce que para la obtención de esquemas globales, una tarea importante es la modelación conceptual que, a partir del análisis de requisitos, siguiendo una metodología de diseño, genera el esquema global que más tarde será objeto de distribución, obteniendo los esquemas lógicos locales que posteriormente serán ubicados físicamente en los sitios de procesamiento. Aquí se observa la necesidad de abarcar varios niveles de abstracción, que van desde el lógico global hasta el físico local. De lo anterior se puede observar la necesidad de abordar los diferentes niveles de abstracción ofreciendo ayudas al diseño en cada uno de estos.

La mayoría de los SMBD comerciales están enfocados al modelo relacional, debido en gran parte a que cuenta con sólidas bases matemáticas, de hecho, el modelo relacional sigue representando la tendencia dominante en el mercado actual de estos sistemas. La importancia del modelo relacional también está dada por las técnicas y herramientas que se han desarrollado para ayudar a la fase de diseño, existe un conjunto de reglas de transformación bien definidas para obtener esquemas relacionales a partir de diagramas ERE [75].

En la sección 1.4.2 se presentaron algunas limitaciones de las herramientas analizadas para la modelación conceptual, lo que estimula la creación de una nueva herramienta. Para dar solución a la problemática de obtención del esquema conceptual global se desarrolló la herramienta de diseño ERECASE [99, 220], que cumple con los requisitos que debe tener una herramienta para esta función; y se integra coherentemente al proceso general de diseño de BDD emprendido conjuntamente por las herramientas consideradas en la visión arquitectónica de la figura 2.2, y que han sido desarrolladas

como parte de esta investigación. Esta integración se lleva a cabo a través del catálogo. A continuación se especifican las características distintivas de ERECASE.

ERECASE

La herramienta ERECASE posee una interfaz gráfica de usuario que apoya la creación de diagramas ERE y la transformación automática a esquemas del modelo relacional; no sin antes someter el diagrama ERE a una exhaustiva validación estructural. A través del uso de cómodas interfaces gráficas, la herramienta permite editar las propiedades de los conjuntos de entidades, interrelaciones y otras construcciones del diagrama ERE. Para la creación del esquema conceptual, se ofrece el siguiente conjunto de construcciones del diagrama ERE: entidades fuertes y débiles; asociaciones recursivas, binarias y ternarias; interrelaciones del tipo ISA e ID; generalizaciones y agregaciones. La representación de agregaciones es algo poco común en las herramientas para la creación de diagramas ERE, lo que hace que ERECASE sea singular en este sentido.

La arquitectura interna de ERECASE consta de tres módulos fundamentales (véase la figura 2.4): el editor gráfico, el módulo de validación y el de generación de esquemas. Los dos primeros módulos interactúan entre si durante todo el proceso de diseño y el último sólo interactúa con los demás en el proceso de búsqueda de los esquemas para la generación del código SQL asociado al modelo ERE planteado en el editor gráfico activo.

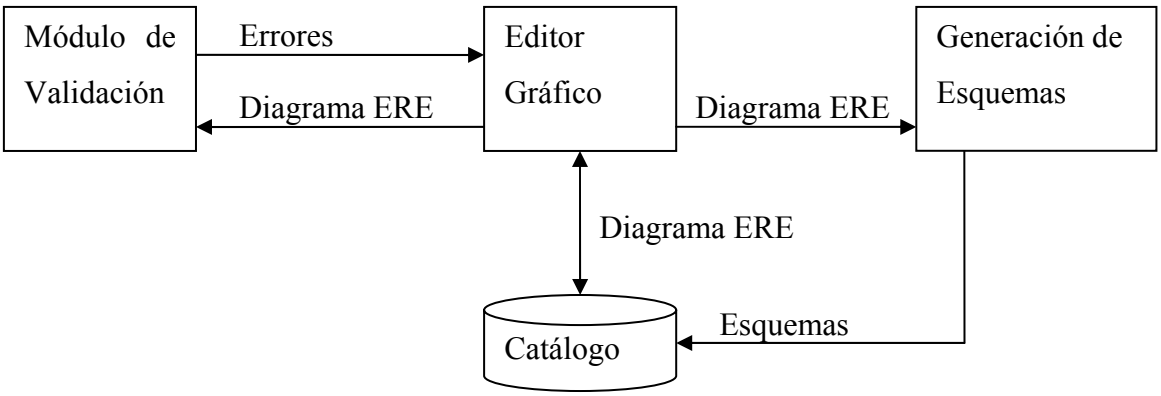

Figura 2.4. Vista arquitectónica de ERECASE.

El módulo Editor Gráfico tiene tres componentes fundamentales:

1. Contenedor del diagrama ERE: Su función es contener las construcciones gráficas del diagrama ERE y brindar el espacio necesario para su creación (véase la figura A2.1 en el anexo 2). Para el caso de las asociaciones, el diseñador puede elegir si hay dependencia de existencia [238] (véase la figura A2.3 en el anexo 2), lo cual será de

gran utilidad en el momento de la transformación a esquemas relacionales y el posterior diseño de distribución, particularmente en la FHD de relaciones miembros de asociaciones, como se verá más adelante. La semántica de la interrelación de dependencia de existencia es clara y precisa, no posee ambigüedad y permite el chequeo de consistencia y de integridad en el ECG.

2. Editor del contenedor: Es el encargado de manipular las funciones generales del diseño como la creación de nuevas construcciones, copiar, cortar, pegar y eliminar entidades; también es el encargado de notificar los eventos a los editores de las construcciones y actualizar el explorador del modelo; así como notificar los errores del modelo en tiempo de edición para su rectificación inmediata.

3. Manipulador de los objetos gráficos: Es el que realiza las funciones gráficas específicas para cada construcción como mover, redimensionar, invocar el editor de propiedades correspondiente (véase la figura A2.2 en el anexo 2), etc.

El módulo de Validación es el encargado de identificar todos los errores que tiene un diagrama ERE, darle respuesta al diseñador del tipo de error cometido, y la localización exacta del error, así como la explicación de la causa del error (véase la zona inferior de figura A2.1 en el anexo 2). Este módulo realiza validaciones en busca de errores en cuanto a nombres repetidos, conjuntos de entidades sin atributos de identificación, etc.; también analiza estructuralmente el diagrama ERE chequeando los esquemas que se generarán a partir de las relaciones recursivas, los ciclos de relaciones binarias, las relaciones ternarias y los ciclos de relaciones binarias donde intervienen relaciones ternarias.

ERECASE tiene la capacidad de determinar si un diagrama ERE hecho con la herramienta es válido o no, aplicando varios tipos de chequeos al diagrama. Estos son:

1. Exclusividad de nombres: no permite que exista un nombre repetido dentro de las construcciones del diagrama ERE.

2. Chequeo de identificación: todo conjunto de entidades que no sea débil y no tenga una interrelación de identificación con otro conjunto de entidades, debe tener al menos un atributo llave.

3. Chequeo de identificación de conjuntos de entidades débiles: todo conjunto de entidades débiles debe tener al menos una interrelación con una entidad fuerte que la identifique.

4. Chequeo de validez estructural de asociaciones recursivas y ternarias; de ciclos de asociaciones binarias, de asociaciones binarias con participación de asociaciones ternarias, y identificación de conjuntos de entidades débiles [99]. Aunque se soportan estos chequeos, no es objetivo de esta investigación profundizar en la validación estructural o semántica de diagramas ERE.

El módulo de Generación de Esquemas cuenta con las funciones necesarias para la transformación a esquemas relacionales de todos los diagramas ERE creados. Para realizar esta transformación se siguen los métodos planteadas en [5], a los cuales se añade el empleo de la regla RTDE obtenida como resultado de esta investigación. Esta regla se concreta más adelante.

La regla RTDE es aplicable al caso de asociaciones con dependencia de existencia [238, 239] identificadas por el diseñador durante la caracterización del ECG. Una definición informal de dependencia de existencia es la siguiente: Si una entidad de tipo P se asocia a una y sólo una, y siempre la misma entidad de tipo Q, entonces la existencia de P depende de Q. De lo anterior se deduce que cuando un lado de la asociación con cardinalidad 1 no es actualizable, una actualización indica la modelación de otro hecho. Los hechos dependen de la realidad y siempre se identifican con llaves, por tanto se requiere una llave que lo identifique. Por tanto, la dependencia de existencia modela la duración de la asociación basada en la noción de la "vida" de un objeto que comprende desde su creación hasta su finalización. Las cardinalidades del lado de la entidad dependiente indican si puede haber solapamientos (m) o no (1) en el ciclo de vida de las entidades que tienen dependencia de existencia. Las asociaciones con dependencia de existencia son siempre obligatorias para el tipo de objeto dependiente. La clasificación de tipos de entidad de acuerdo a dependencia de existencia es la mejor alternativa para los conceptos, a veces confusos, de agregación y composición; además, permite el control de la integridad semántica.

Una definición formal de dependencia de existencia es la siguiente [238]: "Sean P y Q tipos de entidades. P tiene dependencia de existencia de Q si y sólo si una instancia p de P está contenida en la vida de una y solo una ocurrencia q de Q".

RTDE: Para asociaciones con dependencia de existencia se crea un nuevo esquema al cual se le genera un subrogado o identificador de entidad que será su llave primaria, y se toman como descriptores las llaves primarias de los esquemas participantes, y los atributos propios de la asociación si los tiene definidos. Las llaves primarias de los

esquemas obtenidos para las entidades participantes en la asociación constituirán, por supuesto, llaves foráneas en este nuevo esquema.

Es de interés realizar algunas consideraciones sobre la regla RTDE: El concepto de subrogado ya apareció en el modelo RM/T de Codd [41] y representa un identificador que distinguirá cada entidad al margen de sus atributos o propiedades durante toda su vida. ERECASE ofrece la posibilidad de editar los subrogados para reflejar su semántica. El concepto de llave de usuario no es redundante con el de subrogado. Pueden existir propiedades identificadoras asociadas a un esquema que lo distingan a nivel de usuario, pero en ningún caso estas propiedades jugarán el papel de unicidad que tienen las llaves primarias en el modelo relacional.

Las construcciones del diagrama ERE están estrechamente relacionadas desde el punto de vista interno y visual, ya que muchos de los errores que se comenten al diseñar se pueden detectar en tiempo de diseño gráfico, y así no es necesario aplicar algoritmos de alta complejidad para detectar dichos errores. Por ejemplo, cuando se inserta una entidad fuerte sin atributos identificadores, se notifica automáticamente al diseñador que la entidad no tiene ningún atributo de identificación.

En el anexo 2 se muestran otras vistas de esta herramienta durante la caracterización del ECG para el control de transformadores y en el anexo 3 se pueden observar los esquemas relacionales obtenidos por ERECASE

La generación de esquemas relacionales se efectúa hacia el catálogo y también se hace mediante código en el lenguaje SQL a través de llamadas sucesivas a operaciones CREATE/ALTER TABLE, que podrá ser ejecutado en el SMBD específico en caso de diagramas libres de errores. Cuando se generan los esquemas hacia el catálogo (véase la figura A2.4 del anexo 2), se identifican los esquemas propietarios y miembros de acuerdo a la integridad referencial entre esquemas y se almacena en la tabla LINKS (véase el anexo 1).

Toda la información de entrada y de salida del ERECASE es manejada través del catálogo, que es creado cuando se inicia un proyecto de diseño, y se actualiza en los pasos ulteriores (véase la figura A2.4 en el anexo 2). Otra ventaja de ERECASE es la facilidad que da para la modelación de los problemas siguientes:

1. Caracterización de las generalizaciones de acuerdo a su tipo (total o parcial) y participación (exclusiva o solapada).

2. Especificación de la dependencia de existencia presente en asociaciones entre tipos de entidades.

3. Elección de la forma en que migran las llaves en la generación de esquemas cuando puede haber más de una posibilidad.

APPWIZARD

En la literatura aparecen dispersos los requerimientos del universo de discurso en varios momentos del proceso de modelación. En la presente investigación se sistematiza este proceso mediante la herramienta APPWIZARD, que da solución a la necesidad de caracterizar las aplicaciones. Esta herramienta capta la información sobre los patrones de acceso y uso de las aplicaciones sobre la BD, de utilidad en el diseño de la distribución. APPWIZARD es un asistente capaz de introducir en el catálogo los datos que el diseñador, bien por su experiencia o estudios realizados, determina que caracterizan a las aplicaciones activadas con mayor frecuencia. Como se expresó en el capítulo 1, no siempre se pueden caracterizar todas las aplicaciones que operan sobre la BD, pero al menos se caracteriza el 20 por ciento de las aplicaciones más frecuentes o que realizan las transacciones más críticas en cuanto a volúmenes de accesos a datos.

En el proceso de caracterización de las aplicaciones, esta herramienta solicita al diseñador la entrada de la siguiente información:

- Nombre o descriptor que identifique cada aplicación.
- Esquemas que usa.

Para una potencial FV y su posterior asignación se hace necesario capturar:

- Atributos de cada esquema usados por la aplicación así como el tipo de acceso de lectura y/o escritura sobre cada atributo.
- Selectividad de cada esquema que significa la cantidad estimada de tuplas a las que accede la aplicación.
- Sitios donde se activa cada aplicación y la frecuencia con que lo hace en cada sitio.

Para una potencial FHP y su posterior asignación se hace necesario capturar:

- Predicados simples. Para sistematizar la modelación con la intención de favorecer la localidad de procesamiento se tomó la decisión de adquirir los principios de localidad desde la perspectiva de las aplicaciones mediante predicados simples y brindar al diseñador sólo aquellos esquemas que sean usados por alguna aplicación. La captura de estos predicados simples se realiza mediante un constructor de expresiones que, a la vez que permite su edición, realiza la validación de los predicados (véase la figura 2.5).

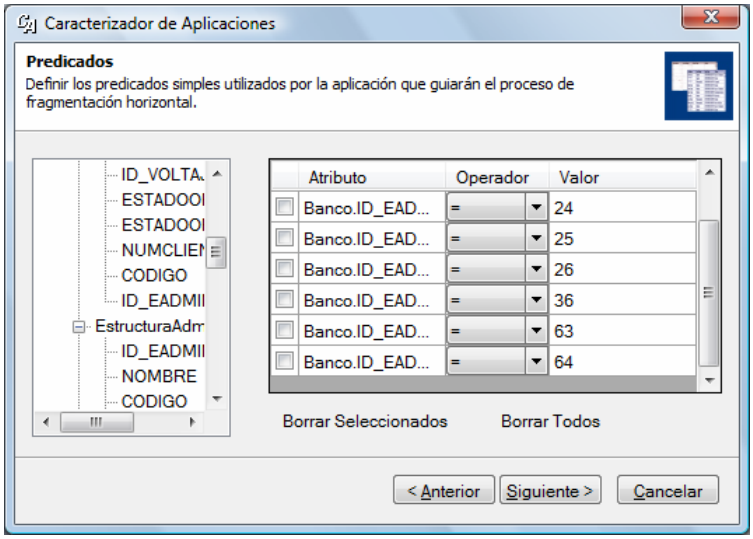

Figura 2.5. Captura de los predicados simples en APPWIZARD.

Los datos de entrada al asistente son obtenidos por ERECASE y NETWIZARD y depositados en el catálogo. Las salidas se guardan en el mismo, quedando disponibles para las herramientas que continúan con el proceso de diseño, o ejecuciones posteriores de este asistente. En el anexo 5 pueden observarse un conjunto de vistas capturadas durante la caracterización de una aplicación.

NETWIZARD

Otro aspecto sin solucionar es la caracterización de las redes que dan soporte a las BDD. Se han reportado trabajos que analizan parcialmente el problema de la comunicación en la red y proponen soluciones con limitaciones [10, 18, 52, 70, 89, 105, 106, 114, 134, 135, 138, 149, 151, 156, 191, 207, 224, 242, 244, 248, 260]. En este sentido es necesario analizar las propiedades de los sistemas de computación en cada sitio y la red de comunicación.

La gran cantidad de aplicaciones distribuidas en la actualidad sólo han sido posibles debido a la evolución constante de tecnologías remotas. La realización de aplicaciones comerciales de alguna manera distribuida, fue posible sólo después que se solucionaran algunos problemas técnicos de estas tecnologías. CORBA, COM+, y EJB comenzaron este proceso hace varios años, el cual se vio simplificado con la aparición de .Net Remoto, que tiene la ventaja de utilizar los estándares opensource muy bien establecidos, como SOAP para mensajería, y HTTP y TCP como protocolo de comunicación, eliminando así las dificultades que tiene DCOM para atravesar firewalls. Además, es más flexible y más personalizable, permitiendo definir nuevos formateadores y canales de comunicación, no necesita definir las interfaces en un

lenguaje abstracto, y oculta todos los detalles de implementación del trabajo con sockets, entregando al programador final una interfaz de programación que se distingue por las facilidades de uso y su potente alcance. Con .Net el concepto de facilidad de implementación ha sido ampliado al desarrollo de aplicaciones distribuidas. No hay ciclos de compilaciones como en Java RMI. No se tienen que definir las interfaces en un lenguaje abstracto como en CORBA o DCOM. Una característica única es que se tiene que decidir el formato de codificación de las peticiones remotas; en cambio, se puede cambiar de un formato binario rápido a SOAP, cambiando solo una palabra en un archivo de configuración. Se puede proporcionar incluso, ambos canales de comunicación para los mismos objetos añadiendo otra línea a la configuración. No se está atado a una plataforma ni a un lenguaje de programación como con DCOM, COM+, o Java EJB. La configuración y el desarrollo es mucho más fácil que como era en DCOM.

NETWIZARD presenta una arquitectura cliente-servidor como se muestra en la figura 2.6, usando la tecnología de comunicación .Net Remoto [217, 218, 255]. En cada sitio reside un servidor y el asistente actuará como un cliente que guarda los datos recopilados sobre las características físicas de cada uno de los sitios y los tiempos de comunicación entre ellos, en el catálogo de la aplicación. La realización de estas actividades no tiene un orden predefinido, se puede comenzar por una u otra indistintamente.

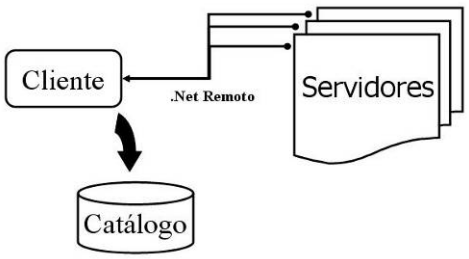

Figura 2.6. Vista arquitectónica de NETWIZARD.

En este proceso de NETWIZARD se buscan los sitios mediante difusión, usando NetServerEnum de netapi32.dll de Windows® [154]. Después se encarga de manejar las peticiones directas del diseñador para buscar los dominios, los tipos de máquinas, así como de chequear la disponibilidad de los sitios manualmente añadidos. Se recopilan los tiempos de comunicación entre sitios, comprobando la existencia de los objetos remotos en los sitios escogidos previamente. Estos objetos remotos se encargan de las comunicaciones para obtener una referencia al proxy que atenderá las peticiones hechas

por ellos y recuperar la información necesaria sobre los sitios de procesamiento. El objeto servidor del asistente reside en un servicio Windows y es el encargado de escuchar las peticiones de los objetos clientes. Este objeto remoto usa un canal HTTP para la comunicación con los clientes, logrando pasar a través de firewalls. Es necesario instalar el servicio en cada uno de los sitios que se desee tomar como servidor para la BDD (se brinda el instalador del servicio junto con NETWIZARD). El objeto remoto en sí es el encargado de resolver la información asociada a los sitios.

La herramienta NETWIZARD realiza la caracterización de los sitios de procesamiento mediante la captura del nombre que identifica a cada sitio, su dirección IP, la velocidad del procesador dado en gigahertz, el espacio total en disco y el espacio libre medidos en gigabytes, los tiempo de lectura de disco y de escritura hacia disco, dados en milisegundos. En cuanto a las redes, la herramienta almacena en el catálogo las mediciones de tiempos de respuesta (en milisegundos) al enviar un frame de un sitio a otro. Concretamente, almacena la dirección IP de origen, la dirección IP de destino, y el tiempo de respuesta como medida de costo de comunicación entre sitios. Cada vez que se mide un tiempo de respuesta entre dos sitios, éste es promediado con el que ya se encuentra almacenado para esos dos sitios en la tabla del catálogo que almacena los tiempos de respuesta, y es reemplazado. Estos parámetros determinan, y en cierta medida caracterizan, los sitios de procesamiento y la red de computadoras sobre la cual, en definitiva, va a estar implementado el SBDD. Según la combinación de estos factores, un sitio es más conveniente que otro para recibir determinado fragmento. Posteriormente todos estos parámetros capturados por NETWIZARD sirven de entrada para el modelo de asignación que se implementa en este trabajo. En el anexo 4 pueden observarse algunas vistas de NETWIZARD.

2.2.2. Modelación lógica

Los sistemas distribuidos más recientes son construidos en redes LAN en las que cada sitio es una sola computadora que mantiene una sola BD local. La siguiente generación será diseñada de manera distinta por el desarrollo de la tecnología, especialmente el surgimiento de multiprocesadores de bajo costo y redes de alta velocidad. El diseño de un sistema también se verá afectado por el aumento del uso de la tecnología de BD en la aplicación dentro de dominios que son más complejos que el procesamiento de datos comerciales y por la adopción amplia del modelo cliente-servidor, junto con la estandarización de la interfaz cliente-servidor. Así, los SMBDD incluirán servidores de

BD con multiprocesadores conectados a redes de alta velocidad que los una a las máquinas de los clientes que ejecutan aplicaciones y participan en la ejecución de los requerimientos de las BD. Por tanto, en su contexto más moderno, el modelo cliente-servidor del que se habla aquí se refiere a máquinas reales, no a procesos. Así, bajo el modelo cliente-servidor se pueden tener arquitecturas como son la de "un servidor-varios clientes" o la más sofisticada de "varios clientes-varios servidores". Bajo la perspectiva de la lógica de los datos, los SMBD cliente-servidor proporcionan la misma visión de una BD lógicamente única que dan los sistemas no jerárquicos o también conocidos como "punto a punto" (peer to peer) [8, 197], la distribución real se da a nivel físico. Las diferencias sólo se dan a nivel del paradigma de arquitectura empleado para llegar al nivel de transparencia, no en la transparencia a nivel de usuario.

FRAGMENTER

En la creación de la herramienta FRAGMENTER, que guía el proceso de diseño de la fragmentación de los esquemas globales, fueron implementados algoritmos usando los referentes de [182] tanto para la FHP como para la FV. A pesar de que la literatura recomienda comenzar con la fragmentación horizontal, por la experiencia del diseñador y las características de la BD que se esté fragmentando, se puede realizar sólo la FV o iniciar el proceso con ella cuando sea mejor para aumentar el rendimiento de las transacciones, incrementar el procesamiento local y minimizar el acceso a datos remotos, aumentar el paralelismo durante la ejecución de las consultas, la concurrencia y el rendimiento.

La herramienta FRAGMENTER permite que el diseñador elija qué tipos de fragmentación aplicará a cada esquema. Después de concluida una alternativa de fragmentación, se pueda realizar otra alternativa obteniendo una FM.

Para la FHP se usan los predicados minterms [182], donde una de las tareas más importantes es encontrar el conjunto de predicados simples mínimo y completo, ya que es la base para determinar los predicados minterms de forma tal que la fragmentación se realice correctamente. Un algoritmo clásico para encontrar el conjunto de predicados simples mínimo y completo es el COM_MIN [182], el cual elimina los predicados redundantes y produce un conjunto mínimo, luego usa los complementos para obtener el conjunto completo que serán usados posteriormente para crear los predicados minterms. Como se planteó en el capítulo 1, determinar si un conjunto de predicados es mínimo y completo no es tan sencillo.

Como resultado de esta investigación se propone el método M-COM_MIN* como alternativa al algoritmo COM_MIN planteado en [182] para encontrar el conjunto de predicados simples mínimo y completo. El valor de M-COM_MIN* radica en que mejora la eficiencia de COM_MIN (véase paso 2 de M-COM_MIN*).

Método M-COM_MIN*

Entrada: R: Esquema propietario de la BD, Pr: Conjunto de predicados simples

Salida: Pr': Conjunto de predicados simples mínimo y completo

Pr'← Pr

Paso 1: Agrupar Pr' por atributo

Para cada grupo obtenido hacer los pasos del 2 al 6:

Paso 2: Ordenar los grupos obtenidos en Pr' de acuerdo al valor del atributo involucrado. Este paso es para lograr mayor eficiencia que la de COM_MIN a la hora de eliminar los predicados redundantes. Si existen varios predicados con igual significado, el valor debe ser el mismo. Por tanto después del proceso de ordenamiento, ellos deben estar adyacentes.

Paso 3: Estandarizar los predicados en Pr' y eliminar redundantes. En la estandarización los operadores > y ≥ se cambian a su forma complementaria ya que un predicado simple es esencialmente el mismo que su forma complementaria. Seguidamente se eliminan los redundantes.

Paso 4: Obtener un conjunto en el cual el mismo valor aparezca a lo sumo tres veces. Bajo esta condición, basta con solo tener los operadores < e =. Luego, si existen tres valores adyacentes iguales, se deben cambiar por dos predicados con igual atributo y valor pero con los operadores < e =, eliminando el tercer predicado, ya que en el paso siguiente se añaden los complementos.

Paso 5: Adicionar los complementos al conjunto Pr'. Para aquellos predicados con operador = y que no tengan el mismo valor que su predecesor (se esta asumiendo que siempre el < y ≤ estarán delante de = para un mismo atributo e igual valor, obtenido en el paso 2) y no todos los predicados tengan como operador el =, insertar el operador < delante.

Paso 6: Colocar el complemento en la posición final de Pr'. Cuando todos los predicados sean de operadores =, insertar una condición al final que signifique la negación de todos los predicados anteriores a esta posición; en caso contrario, insertar en la posición final una condición complemento, en dependencia del

predicado predecesor. Al concluir este paso se tiene un conjunto de predicados mínimo y completo.

Paso 7: Unir los grupos obtenidos en Pr'. Al final se obtienen varios conjuntos de predicados basados en el mismo atributo. Si unen todos esos conjuntos se obtiene el conjunto Pr' de predicados simples mínimo y completo.

FinMétodo M-COM_MIN*

Algunas consideraciones sobre el método M-COM_MIN* son las siguientes: Se han ordenado los predicados según su valor y se han eliminado los redundantes basado en sus atributos. Si existiesen predicados que no dividieran más un fragmento, al cambiarlos a forma estándar no sufrirán cambios. Por tanto, no existen predicados no relevantes. Además, por la forma en que han sido captados los predicados mediante un constructor de expresiones en APPWIZARD, se tiene el predicado si no existe redundancia y si hay alguna aplicación que lo utiliza. De esta forma nunca existirán dos o más fragmentos que sean siempre accedidos simultáneamente por cualquier aplicación. En otras palabras, al menos una aplicación accede a los dos fragmentos de forma diferente. Luego el conjunto de predicados que se obtuvo es mínimo. Si existiese algún predicado usado por una aplicación, será insertado en este conjunto de predicados. Luego de la fragmentación, se asegura que todas las condiciones han sido incluidas en los predicados minterms. Si alguna aplicación accediera a los fragmentos, podría acceder a todas las tuplas en el fragmento. Es decir, las tuplas en los fragmentos son accedidos uniformemente y tienen igual probabilidad. Entonces la completitud está garantizada.

En la obtención del conjunto de predicados minterms para realizar la FHP, se combinan los predicados simples obtenidos mediante el método M-COM_MIN* de cada conjunto usando la operación *AND* sin repetir predicados del mismo conjunto en cada combinación. En este proceso se puede observar que no se obtendrán predicados minterms contradictorios, debido a que se selecciona una sola condición de cada atributo cada vez. Además se puede saber cuántos predicados minterms serán obtenidos, es decir cuántos fragmentos se obtendrán con solo multiplicar la cantidad de elementos de cada conjunto intermedio. La complejidad del algoritmo es de $O(n^4)$. Una vez obtenidos los predicados minterms, la herramienta FRAGMENTER ofrece la posibilidad de eliminar aquellos que sean contradictorios de acuerdo a la semántica del

problema, lo que demanda un conocimiento profundo del problema por parte del diseñador.

La FHD se aplica a relaciones miembros a través de llaves foráneas que hacen referencia a esquemas con FHP. Para esto se usaron dos criterios CFHD1 y CFHD2 siguiendo el método Hacer_FHD.

Método Hacer_FHD.

Si es aplicable CFHD1 Entonces

Aplicar CFHD1

En otro caso

Aplicar CFHD2

FinMétodo Hacer_FHD

Los criterios en cuestión son:

CFHD1: Si el esquema a fragmentar fue obtenido como resultado de la transformación realizada por ERECASE a una asociación con dependencia de existencia mediante la regla de transformación RTDE vista anteriormente, se efectúa la FHD a partir del esquema de mayor cardinalidad que tenga realizada una FHP.

CFHD2: Usar el criterio de "la fragmentación usada en más aplicaciones" referido en el capítulo 1 en la sección relativa a *Fragmentación Horizontal Derivada*. Su implementación fue muy directa, considerando la frecuencia con la cual las aplicaciones acceden a los datos y que es capturada por APPWIZARD.

Para diseñar la FM es más conveniente comenzar con una FH lo que se justifica considerando los beneficios que guían la naturaleza de la FH y dentro de los cuales es necesario resaltar la localidad de referencia, con lo que se garantiza menor tiempo de respuesta y menor costo de procesamiento para solicitudes locales. Otra ventaja adicional es que el esquema de fragmentación es, a menudo, consistente con la estructura de la organización. Teniendo en cuenta además, que la fórmula $\Pi_{Az}\,(\sigma_{Fj}\,(R_i))$ es más usada en los modelos de optimización de consulta porque al aplicar primeramente el operador de selección se disminuyen más cardinalidades y por tanto el tiempo de respuesta de las solicitudes es menor, y que la fórmula $\sigma_{Fj}\,(\Pi_{Az}\,(R_i))$ implica que en el momento de realizar la proyección hay que garantizar que no se eliminen los atributos necesarios para la selección, lo que le añade complejidad al diseño de FM ya que habría que comprobar en cada paso que el atributo a eliminar no esté presente en el predicado de selección, y esto incrementaría los tiempos de respuesta de las solicitudes.

No obstante, el diseñador de la BD tiene la posibilidad de decidir cuál de los dos tipos de fragmentación aplicar primero, en base al conocimiento que tenga de las aplicaciones. FRAGMENTER coloca en el catálogo todos los esquemas relacionales obtenidos en ERECASE como fragmentos, y luego procede a la fragmentación; así cada esquema será objeto de asignación aunque no sea fragmentado.

ALLOCATOR

La asignación de los datos a los sitios en un ambiente distribuido afecta notablemente el desempeño de un sistema, ya 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. Como se planteó en el capítulo 1, el problema de asignación ha sido estudiado ampliamente por diferentes autores [9, 10, 18, 52, 70, 104, 106, 114, 126, 134, 138, 149, 151, 156, 176, 200, 202, 203, 215, 216, 229, 233, 242, 244, 248, 260] aunque aún 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 sólo bajo ciertas especificaciones; por ejemplo, se impone la condición de que cada fragmento se encuentre almacenado en un solo sitio de la red, desaprovechando las ventajas de la replicación a cambio de que el espacio de soluciones disminuya considerablemente. Esta investigación propone dos métodos, uno basado en algoritmos genéticos y el otro en aprendizaje reforzado, para solucionar el problema de asignación de fragmentos en el diseño de BDD según el modelo matemático general planteado por Özsu y Valduriez [182], en lo adelante denominado modelo de referencia, que es NP-Hard, cuyo problema de decisión asociado es NP-Completo. Posteriormente se realizan consideraciones para la implementación de estos métodos.

El problema de asignación enunciado en el capítulo 1 tiene como objetivo minimizar el costo total de procesamiento y almacenamiento, sujeto a restricciones de tiempo de respuesta, de capacidad de almacenamiento y de tiempo de procesamiento. La variable de decisión es x_{ij} que toma el valor 1 si f_i es almacenado en s_j y 0 en otro caso, lo que implica una asignación no replicada (no redundante). El modelo de referencia es muy general, por lo que para lograr una implementación computacional del mismo hay que

hacer un conjunto de consideraciones sobre la condición de réplica y el tratamiento a las restricciones.

Condición de réplica: El modelo de referencia 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. Sean i y j los índices de los fragmentos y los sitios respectivamente. La condición de réplica queda modelada de la siguiente manera:

$$\sum_{\forall f_i \in F} x_{ij} \geq 1, \ \forall s_j \in S$$

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 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, también es desechada por no contar con suficiente información en un diseño descendente, esta será considerada en futuras implementaciones de un modelo de reubicación, teniendo en cuenta parámetros reales de desempeño de las aplicaciones.

Los métodos propuestos en ALLOCATOR son desarrollados en forma de agentes que van a usar los datos almacenados en el catálogo para obtener esquemas lógicos locales que serán objeto de ubicación. Uno de los agentes se basa en un algoritmo genético generacional [59, 74, 202, 203] cuya primera versión fue programada en Borland Delphi versión 6 y publicada en [219], que luego fue reprogramado en Visual C# para integrarse a la herramienta ALLOCATOR, desarrollada para dar solución al problema de la asignación. El otro agente también se integra a la herramienta y se basa en el algoritmo Q-Learning del Aprendizaje Reforzado [1, 40, 63, 116, 152, 163, 176, 194, 209, 243]. 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 alcanzada o un mensaje en caso de que no se haya encontrado una solución que cumpla con las restricciones del modelo matemático.

Algoritmo genético

Para dar solución al problema se implementó un algoritmo genético generacional que aparece más adelante como método M-AG, donde la población nueva reemplaza a la

antigua población manteniendo el mejor individuo mediante elitismo. El cromosoma usado es una matriz de valores binarios. Con esto se logra que su representación esté lo más cerca posible de la solución real. Las columnas del cromosoma (véase la figura 2.7) 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 toma valor 1 si el fragmento i está almacenado en el sitio j, 0 en otro caso.

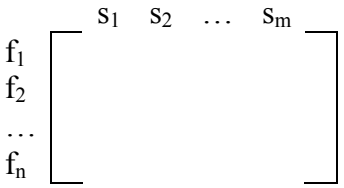

Figura 2.7. Estructura del cromosoma.

Se generó la población inicial 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. Los parámetros usados en la implementación fueron obtenidos mediante un periodo de experimentación y ajuste de los mismos. Mediante las pruebas realizadas al algoritmo se determinó usar 99 individuos como tamaño de la población. Como función de evaluación que asigna la adaptabilidad de cada individuo al medio ha sido elegida la función de costo del modelo de referencia. Si el individuo no cumple con alguna de las restricciones se penaliza, asignándole un valor de adaptabilidad cualitativamente muy bajo, lo que significa que como se está minimizando este valor sería muy alto cuantitativamente. Debido a su simplicidad y a la esencia no determinista del algoritmo genético, se usa el mecanismo de selección estocástico de torneo, en este caso de tamaño dos, que consiste en seleccionar de manera aleatoria dos individuos de la población y escoger el que tenga mayor adaptabilidad. Esta alternativa combina la idea del ranqueo de la población con el método de selección de la ruleta, evitando la convergencia temprana. El cruzamiento se lleva a cabo si al generar un número aleatorio, este 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 iguales, y las filas que ocupan posiciones superiores se intercambian entre ambos individuos. Se tomó 0,98 como probabilidad de cruzamiento.

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 0,2 como probabilidad de mutación. Para detener el algoritmo se usa un criterio de terminación híbrido que consta de un mecanismo de tipo MAX y cierta tolerancia de convergencia. El mecanismo de tipo MAX fija un número máximo de generaciones, en este caso se escogió 80.

En el proceso de evaluación de la población, se calcula la suma de las adaptaciones de los individuos no penalizados. 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. Para este problema se fijó la tolerancia en 0,995. En el algoritmo se realizan los procesos de selección, cruce y mutación. Para evitar la convergencia temprana se pone en práctica la variante del elitismo, donde el mejor individuo se copia para la siguiente generación, garantizando que si en alguna de ellas se alcanza el óptimo, no se pierda en los procesos de cruzamiento o mutación.

Para llevar a cabo 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 de si el individuo está penalizado o no. El método implementado es el M-AG.

Método M-AG

```
Inicializar PoblaciónActual aleatorio
Evaluar (PoblaciónActual)
Mientras Generación < maxGeneraciones Hacer //Una época
  NuevaPoblación[0] ← MejorIndividuo( ) // Elitismo
  i ← 1
  Mientras i < TamañoPoblación Hacer
    Hijo1 ← NuevaPoblación[i] // selección estocástica (torneo de tamaño 2)
    Hijo2 ← NuevaPoblación[i+1]
    Padre1 ← SeleccionarIndividuo()
    Mientras Padre1 = Padre2 Hacer
        Padre2 ← SeleccionarIndividuo()
    Si AplicaOperador(ProbabilidadCruce) Entonces
        Cruzar(padre1, padre2, hijo1, hijo2)
    En otro caso
```

```
            Hijo1 ← Padre1
            Hijo2 ← Padre2
        Si AplicaOperador(ProbabilidadMutación) Entonces
            Mutar(hijo1)
        Si AplicaOperador(ProbabilidadMutación) Entonces
            Mutar(hijo2)
        i ← i+2
    Reemplazar PoblaciónActual
    generación ← generación+1
    Evaluar (PoblaciónActual)
    Si MejorIndividuo penalizado Entonces
      buenaSolución ← false
    En otro caso
      buenaSolución ← true
FinMétodo M-AG
```

Algoritmo Q-Learning

Uno de los avances más importantes en el Aprendizaje Reforzado fue el desarrollo del Q-Learning [40, 256], que es un algoritmo de política off de control de Diferencias Temporales. 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) [40, 152, 243]. 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 (golosa), mientras que la política de conducta puede continuar muestreando todas las posibles acciones (ε-goloso). O sea, la política de comportamiento está separada de la política que se quiere mejorar.

El algoritmo implementado para dar solución al problema de asignación posee las características que distinguen a un sistema de Aprendizaje Reforzado, que cuenta con:

agente, ambiente, estados, acciones y función de valor. El algoritmo aparece más adelante como método M-QL.

El agente tiene la función de aprender a ubicar los fragmentos en los sitios, de manera tal que se minimicen los costos de procesamiento y almacenamiento en un espacio de tiempo razonable. El ambiente está constituido por el catálogo de la aplicación. Allí se almacena toda la información que se necesita en el modelo de referencia, la cual se utiliza para estimar la función de costo y así ubicar los fragmentos de forma óptima, cumpliendo con las restricciones del modelo. El espacio de estados es el conjunto $\{(f_i, s_j) \mid i = 1...n, j = 1...m\}$ de estados, donde cada estado representa un par (fragmento, sitio) e indican la condición actual del ambiente en que se ubica el problema, es decir, si el fragmento se ubica o no en el sitio correspondiente.

Las posibles acciones a tomar por el agente son dos: "sí ubicar", en caso de que el fragmento se ubique en el sitio correspondiente al par (fragmento, sitio) según el estado en cuestión; o "no ubicar" en caso contrario. La función de valor está representada por la matriz Q(estados, acciones) [243], 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 (exploración), su contenido le permite determinar a un agente si es factible o no ubicar un fragmento en el sitio correspondiente (explotación).

El algoritmo Q-Learning cuenta con dos matrices. La primera es una matriz de valores binarios que tiene la misma estructura que la matriz de solución del modelo y precisamente representa la mejor solución que se ha obtenido hasta el momento. La segunda matriz representa la matriz Q que almacena el conocimiento adquirido por el agente, las filas de esta matriz representan los estados (combinación de fragmentos con sitios) del algoritmo y las dos columnas representan el valor de costo para las dos acciones posibles (costo de ubicar y de no ubicar). Además, cuenta con atributos que definen los parámetros de control del algoritmo, tales como la cantidad de episodios, la velocidad de aprendizaje α, el factor de descuento γ, y el factor de exploración ε.

Antes de comenzar a ejecutar el ciclo de episodios del algoritmo, se necesita inicializar la matriz solución y la matriz Q. La matriz solución se inicializa colocando aleatoriamente cada fragmento en un sitio. 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.

A cada par estado-acción (*s*,*a*) se le asigna un valor de Q o acción de utilidad, que representa un elemento en la matriz cuya entrada es el correspondiente valor aproximado de Q(*s*,*a*). Este valor es un estimado de la suma de los futuros valores de refuerzo recibidos a partir del estado *s*, tomando la acción *a* y siguiendo una política golosa con respecto a la función Q. En su forma más simple un paso del algoritmo se define por la regla RQL:

RQL: $Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha[r_{t+1}+\gamma \max_a Q(s_{t+1},a)-Q(s_t,a_t)]$

Esta regla actualiza cada par estado-acción maximizando todas las posibles acciones en el estado siguiente. En este caso, la función de acción-valor observada Q se aproxima directamente a Q^*. La función de acción-valor óptima es independiente de la política seguida. La política mantiene un efecto en la determinación de qué par estado-acción será visitado y actualizado. Sin embargo, todo lo que se requiere es que cada par continúe siendo actualizado. Este es un requerimiento mínimo para cualquier método interesado en encontrar un comportamiento óptimo en el caso general [243].

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-golosa; es decir, en la tabla Q(*s*,*a*) 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 (alrededor de un 95%) para que se exploren bastantes acciones, y va disminuyendo a medida que transcurren los episodios, para poder explotar el conocimiento adquirido.

Si la acción seleccionada es cero, (indica que el fragmento no se ubica en el sitio correspondiente) y como consecuencia, el fragmento no queda ubicado en ningún sitio, se fuerza a tomar la acción uno; de esta forma se logra que en cada paso del algoritmo la matriz de solución cumpla con la condición de réplica. La recompensa inmediata r_{ija} es el costo de ubicar el fragmento correspondiente al estado actual (f_i, s_j) en los sitios donde se encuentra asignado, para todas las aplicaciones, utilizando la función de costo del modelo de referencia. Si producto de la acción tomada se excede la capacidad de procesamiento del sitio, se penaliza el hecho de haberla escogido, tomando como recompensa inmediata el doble del costo de la asignación del fragmento en todos los

sitios donde se encuentra. Las transiciones de estado son deterministas (véase la tabla 2.1), donde el próximo estado sólo depende del anterior.

Tabla 2.1. Transiciones de estado en M-QL.

Estado actual	(f_i, s_j)	
Acción	0	1
Recompensa	r_{ij0}	r_{ij1}
Próximo estado	(f_i, s_{j+1}), si $j < m$	
	(f_{i+1}, s_1), si $j = m$	

La matriz de solución del modelo se actualiza tomando la acción correspondiente al menor valor en la matriz Q para cada estado (comparando los valores de las dos columnas, la de "sí ubicar" y la de "no ubicar"). Si la ubicación final de un fragmento incumple con la condición de réplica, se le asigna el sitio correspondiente a la menor diferencia entre sus acciones en la tabla Q.

También puede ocurrir que un fragmento quede ubicado en más de un sitio (replicado) porque resultó un valor menor en la columna "sí ubicar" que en la de "no ubicar" para un mismo fragmento en diferentes sitios.

Método M-QL

Inicializar Q como matriz cero
Inicializar MatrizSolución aleatoriamente
mejorSol←MatrizSolución
buenaSolución←false
Inicializar α←0,5, ε←0,99, γ←0,8
Repetir para cada episodio
  Repetir para cada i←1..n, j←1..m
    s←(f_i, s_j)
    a←SeleccionarAcción usando política ε-golosa
    Actualizar MatrizSolución(*a*)
    r←Recompensa(*s*,*a*)
    s'←PróximoEstado(*s*)
    Actualizar Q usando la regla RQL
    s←*s*'
  Actualizar MatrizSolución(Q)

Si $\alpha > 0{,}15$ Entonces $\alpha \leftarrow \alpha - 0{,}015$

Si $\varepsilon > 0{,}5$ Entonces $\varepsilon \leftarrow \varepsilon - 0{,}005$

Si MatrizSolución factible Entonces

Si costo(MatrizSolución) < costo(mejorSol) Entonces

mejorSol←MatrizSolución

Si mejorSol factible Entonces

buenaSolución←true

FinMétodo M-QL

En cada episodio se recorren todos los estados actualizando la matriz Q siguiendo la regla RQL. Luego se actualiza la matriz de ubicación del modelo matemático, y si 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. El método implementado es el M-QL [221]. Se puede garantizar que la mejor solución obtenida por M-QL converge a la óptima debido a la utilización de la tabla de valores Q como función de aproximación. En la programación dinámica clásica normalmente se usan tablas como función de aproximación, aunque esto puede limitar el tamaño y la complejidad de los problemas a resolver. Para problemas reales con espacios de estados continuos o extremadamente grandes, no es posible representar la función de valor usando tablas; para esto se recomienda una función de aproximación que generalice e interpole los valores de estado no visitados como extensión a la iteración de valor clásica usando, por ejemplo, redes neuronales para la aproximación.

Para la experimentación de los algoritmos se generaron casos de prueba aleatorios (véase la tabla 2.3), 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 tiempo dado en milisegundos) por un método exacto de enumeración total. En los casos restantes el valor es el más pequeño que se ha obtenido en las etapas de prueba del software. Para evaluar la calidad de la solución y el tiempo de procesamiento, se midieron por tiempos de ejecución. Está claro que no se trata de tiempo efectivo de procesos, que es difícil de medir en el ambiente Windows. Se ejecutó 20 veces cada algoritmo para los diferentes casos de

prueba. Se determinó para cada caso el promedio de las soluciones, las veces que alcanzó el óptimo, la peor solución, el tiempo que tardó en obtener la solución (en segundos) y el error relativo del promedio con respecto al óptimo. Los anexos 8 y 9 muestran los resultados de los experimentos realizados.

Tabla 2.3. Casos de experimentación.

Caso	*Fragmentos*	*Sitios*	*Aplicaciones*
1	6	2	22
2	7	4	5
3	3	10	6
4	17	3	8
5	24	2	10
6	16	3	22
7	8	6	16
8	7	11	5
9	11	8	7
10	25	7	5
11	14	9	29
12	24	16	16
13	27	18	18
14	30	20	20
15	36	24	24

Para instancias pequeñas y medianas del problema ambos algoritmos se comportan de manera similar en cuanto a la calidad de la solución. El algoritmo basado en Q-Learning consume un menor tiempo de ejecución independientemente del tamaño del problema, pero a medida que este aumenta, va perdiendo calidad en las soluciones, no así el Algoritmo Genético que mantiene cierta estabilidad en el error relativo del promedio respecto al óptimo. Una vista arquitectónica de ALLOCATOR puede observarse en la figura 2.8, mientras que en el anexo 7 se muestran más detalles de la misma.

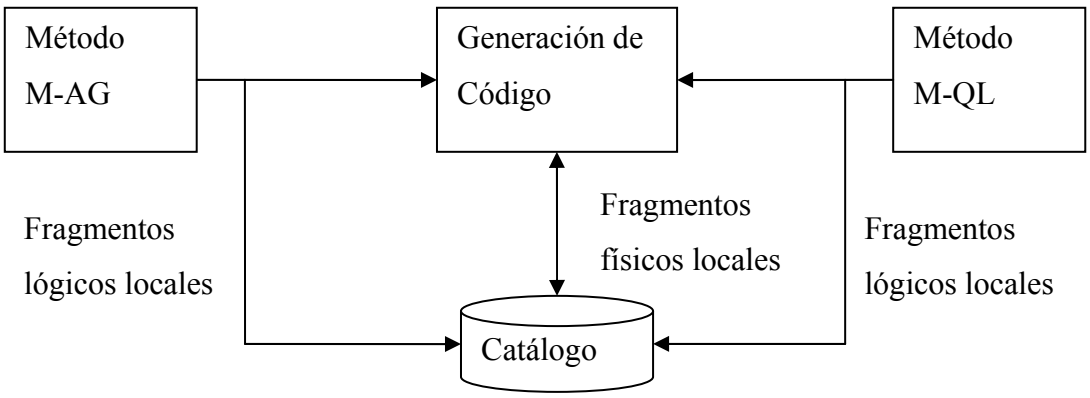

Figura 2.8. Vista arquitectónica de ALLOCATOR.

La aplicación de los métodos heurísticos, basados en Algoritmos Genéticos y Aprendizaje Reforzado, para dar solución al problema de asignación, se justifica dada la limitación de los métodos exactos en obtener la solución óptima en un tiempo de cómputo razonable cuando crece el tamaño del espacio de solución. Mediante las pruebas realizadas y los resultados obtenidos, se demostró que ambas técnicas de Inteligencia Artificial son factibles de aplicar, y a medida que aumenta el tamaño del problema, la solución del mismo mediante los Algoritmos Genéticos tiende a ser mejor que la solución mediante el algoritmo Q-Learning.

El módulo de generación de código se encarga de construir un conjunto de scripts, uno para cada sitio caracterizado por NETWIZARD elegido previamente en la caracterización de las aplicaciones por APPWIZARD.

2.2.3. Modelación física

La modelación hasta el nivel físico no puede ofrecer soluciones generales por su fuerte dependencia de los SMBDD. Por esta razón se ofrece una solución propia de este nivel para el motor de BD Microsoft SQL Server, de gran popularidad en Cuba. 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 bases de datos centralizadas a través de redes. También permite duplicar un servidor local de bases de datos 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 SMBD utiliza la metáfora de publicación/suscripción [2, 17, 51, 58, 80, 103, 122, 160, 231, 249, 250] en su modelo de replicación. Las fases de la replicación son [113]: 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 probablemente 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 FHP; los verticales, a su vez, la FV; y los combinados, por su parte, la FM.

Para implementar el diseño de la distribución mediante replicación, se emplea el método M-GC, 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 de resolución de conflictos (replicación asíncrona, activación de la variable XACT_ABORT para una correcta ejecución concurrente de las transacciones) [3, 4, 30, 68, 69, 76, 110, 120, 121, 183-185, 188] 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 (siempre se elige la suscripción de tipo Pull porque da mayor autonomía a los suscriptores).

El tipo de replicación elegido es el de mezcla, por ser el que da mayor autonomía a los sitios. Si el diseñador lo desea, puede modificar la secuencia de comandos una vez generada. Los datos en el suscriptor son automáticamente actualizables cuando se usa replicación de mezcla. Por otra parte, la configuración de los permisos de acceso a publicaciones se deja diferida al actor Administrador de la BD.

Método M-GC

Repetir para cada sitio s_k , k←1..m

Repetir para cada sitio s_l , l←1..m

Si k ≠ l Entonces

ReconocerServidor s_l //sp_addlinkedserver

SeleccionarDirectivaAsíncrona //sp_serveroption lazy validation

SET XACT_ABORT ON

CrearBDLocal(<nombre del proyecto>)

ConfigurarOpciónReplicaciónBD(merge publish)

ConfigurarPublicaciónMergeBD(<nombre del proyecto>)

Repetir para cada f_j, j←1..n

Si ALLOCATOR ubicó f_j en s_k Entonces

CrearFragmentoLocal(<fragmento f_j según FRAGMENTER>)

Repetir para cada f_j, j←1..n

Si ALLOCATOR ubicó f_j en s_k Entonces

CrearIntegridadReferencial(f_j según FRAGMENTER)

ConfigurarArtículoMerge(f_j)

Repetir para cada columna c_{ij} de f_j

ConfigurarColumnaArtículoMerge(c_{ij})

Si f_j es replicado Entonces

ConfigurarSuscripciónPullMerge(f_j)

ActivarPublicaciónMergeBD(<nombre del proyecto>)

FinMétodo M-GC

2.3. Catálogo

El catálogo es una BD relacional, autodescriptiva por naturaleza [107, 118, 129, 175, 181, 190, 210], a la cual se accede indirectamente a través de las herramientas. El mismo almacena toda la información que interviene en el proceso de diseño de BDD emprendido por las herramientas presentadas anteriormente en este capítulo. En el uso del catálogo se emplean 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 sólo son usadas internamente por las herramientas, tanto para guardar sus datos como para obtener sus valores de entrada, compartiendo por esta vía toda la información relevante para el proceso de diseño de BDD, de esta forma no son de interés para los diseñadores de BDD; como usuarios de las herramientas. En la figura

2.9 aparece el diagrama ER del catálogo. En el anexo 1 se muestra el esquema relacional asociado.

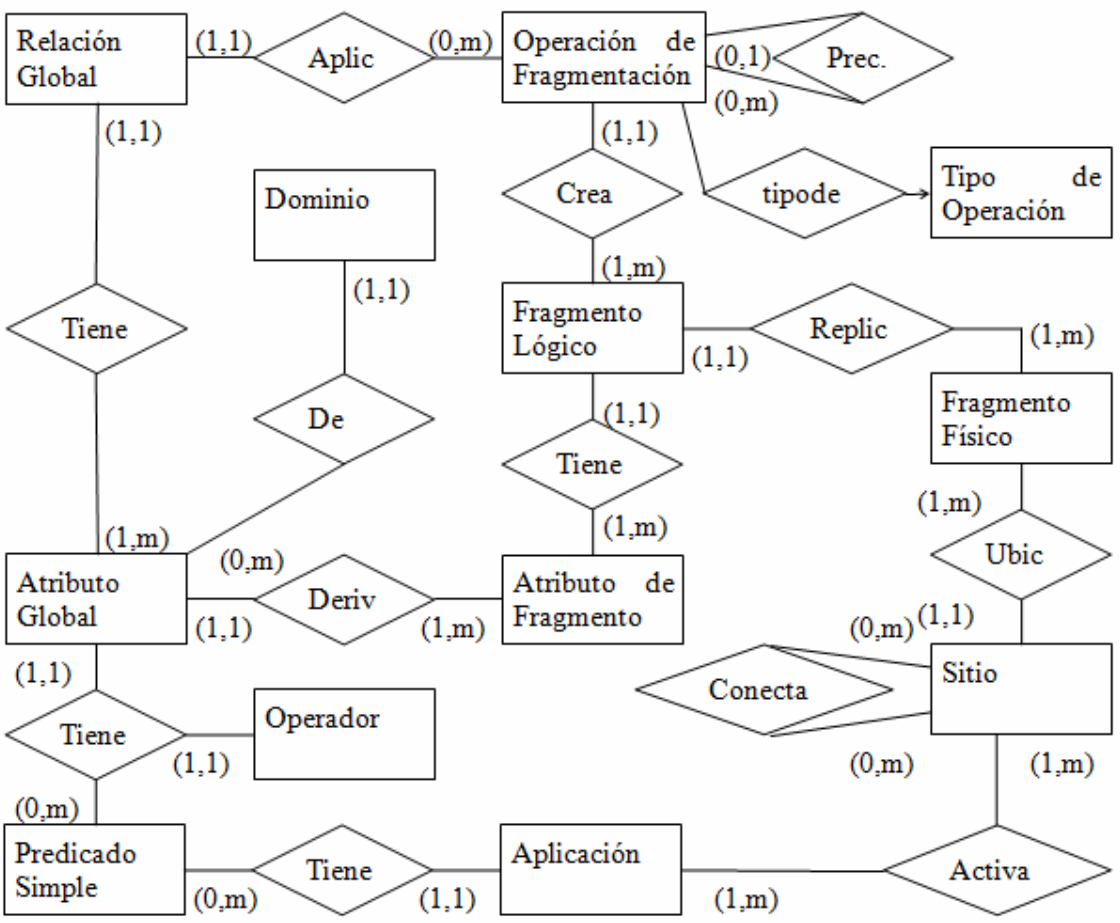

Figura 2.9. Esquema conceptual del catálogo.

2.4. Conclusiones parciales

Se han obtenido cinco herramientas de ayuda al diseño de BDD que se integran a través de un catálogo y tienen valor práctico, a la vez que son novedosas en la sistematización de la teoría del diseño de BDD y añaden nuevas soluciones a problemas planteados en el capítulo 1, como son:

- Creación de una regla de transformación (RTDE) para asociaciones con dependencia de existencia, y un método específico para su uso en la FHD.
- Mejoramiento de la eficiencia del método COM_MIN empleado en la FHP mediante la implementación del método M-COM_MIN*.
- Empleo de dos métodos heurísticos (M-AG y M-QL) para dar solución al problema de asignación con posibilidad de replicación de una forma computacionalmente tratable.
- Construcción de un método que aproveche las características de la replicación en Microsoft SQL Server para la generación de esquemas físicos.

3. VALIDACIÓN DE LAS HERRAMIENTAS EN LA MODELACIÓN DE DATOS PARA EL CONTROL DE TRANSFORMADORES ELÉCTRICOS

En este capítulo se validan las herramientas obtenidas como resultado de este trabajo, en la gestión del control de los transformadores en las OBEs, concretamente de Sancti Spíritus, siguiendo el ciclo de los transformadores desde que se instalan en un banco hasta que son retirados de la línea, almacenando todas las actividades que se realizan sobre los mismos.

3.1. Problema del control de los transformadores eléctricos

La OBE de Sancti Spíritus se dio a la tarea de implementar un módulo para el control de los transformadores instalados y las operaciones relacionadas en la explotación de los mismos, reflejando la estructura inherentemente distribuida de las diferentes unidades de la Empresa Eléctrica, teniendo autonomía local y ofreciendo buen rendimiento. En Cuba, durante los últimos años, la distribución ha sido el área de trabajo menos atendida de la Unión Eléctrica presentándose altos índices de pérdidas e interrupciones, un creciente número de transformadores dañados, y un escaso nivel de informatización y automatización.

Como respuesta a los problemas informáticos detectados, se identifica como necesidad el desarrollo de un sistema que permita controlar los transformadores instalados con una reducción de los gastos totales de explotación de la distribución, influyendo en un mejor servicio a los clientes. El transformador es el equipo más cercano al cliente que más abunda en las redes eléctricas cubanas, con una distribución espacial muy variada y el mayor índice de fallas; de ahí la importancia de minimizar sus averías. Esta es otra razón por la que se necesita desarrollar un sistema capaz de automatizar toda la información relacionada con ellos, poder seguir sus ciclos de mantenimiento, su estado de carga; así como ser capaz de prevenir las fallas.

Para diseñar el sistema hay que tener en cuenta la estructura geográficamente distribuida de las Empresas Eléctricas y sus propias características de comunicaciones; reconociendo los cuatro niveles bien definidos: nacional, provincial, territorial y de

sucursal, además de un quinto nivel que es el del taller, al mismo nivel de la provincia. La comunicación entre la OBE Provincial y las OBEs territoriales se hace a través de líneas telefónicas de baja velocidad; aunque en los últimos tiempos se han introducido gradualmente las redes inalámbricas, aún no se cuenta con este sistema en todas las provincias, ni es política de la Unión Eléctrica hacerlo extensivo en poco tiempo. Como la topología de las redes informáticas no es la más adecuada para contar con una BDC provincial donde accedan todos los usuarios, ya sean de la provincia, taller o territorio, es necesario diseñar una BDD, para poder seguir el ciclo de vida completo de cada transformador manteniendo datos históricos de todas las operaciones que se realizan. Esto le da la facilidad de hacer numerosos estudios comparativos del comportamiento de las diferentes instalaciones, equipos, así como prever posibles fallas en ellos. El transformador tiene un ciclo de vida muy difícil de seguir ya que pasa por diferentes estados en diferentes ubicaciones.

El control de los transformadores se hace en el ámbito provincial, por lo que sólo intervienen la provincia, el territorio y el taller como sitios del sistema. El ciclo de vida del transformador comienza en el taller mediante la verificación de un transformador nuevo (véase el diagrama de estados de un transformador en la figura 3.1).

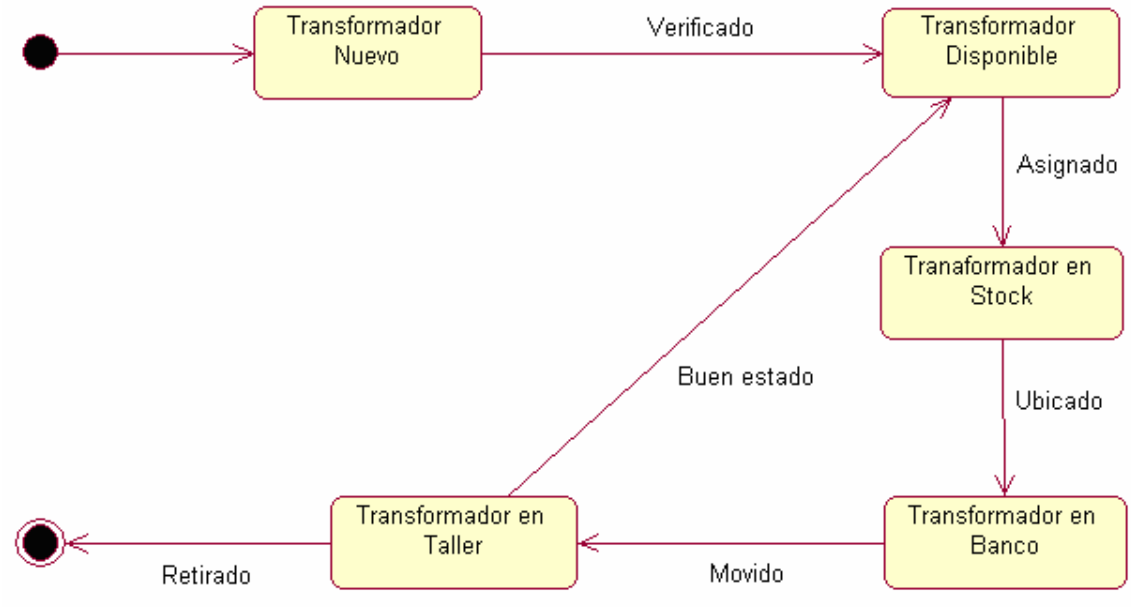

Figura 3.1. Diagrama de estados de un transformador.

Después pasa a formar parte de los transformadores disponibles en la OBE Provincial donde aparecerán todos los datos de su ficha. Allí, dependiendo de sus características y las necesidades de las diferentes OBEs territoriales, es asignado a una de estas, y de ahí

hacia el banco que lo necesita. En el municipio sólo se encuentran los datos pertenecientes a las instalaciones y equipos que atiende este territorio, mientras que en la provincia está centralizada toda la información de las OBEs territoriales.

En la OBE territorial se realizan todas las operaciones sobre el transformador que van conformando una historia. Existen dos formas para que un transformador retorne hacia la provincia: moviéndolo directo al taller o mediante un evento de *Reporte de inspección al transformador dañado* que lo retira automáticamente del Banco y lo envía al taller si se comprueba que está dañado, a la vez que se crea un evento de *Necesidad de transformador* para sustituir este averiado. Luego de llegar el transformador y sus datos al taller, se realiza la *Defectación o Diagnóstico en el Taller*. Esta operación es la que dice si el transformador se retira definitivamente por su daño o si su problema es solucionable. En este caso, el transformador no pierde su historia, se da como disponible, pasa a la provincia y vuelve a fluir, dependiendo de sus características técnicas y las necesidades de las OBEs territoriales. Nótese que en todo este ir y venir no pierde los datos de todas las pruebas que se le hayan realizado, aunque sea situado en una OBE territorial diferente.

3.2. Modelación conceptual

En la modelación conceptual del problema del control de los transformadores se tomaron en cuenta las necesidades de información identificadas por la OBE Provincial.

3.2.1. Caracterización del ECG

La herramienta ERECASE permitió modelar con éxito el ECG para el problema planteado (véase la figura 3.2). Algunas ventajas presentes en esta herramienta son:

1. Poder caracterizar el tipo de entidad Ubicación como generalización para los tipos de entidades Banco y Taller, de acuerdo a su tipo y participación (véase la figura 3.3a).
2. Lograr especificar la dependencia de existencia presente en la asociación entre los tipos de entidades Banco y EstructuraAdministrativa (véase la figura 3.3b).
3. Poder elegir la forma en que migran de llaves en la generación de esquemas para los tipos de entidades ReporteMovimiento y MovimientoItem (véase la figura 3.3c)

En el anexo 2 se muestran otras vistas de esta herramienta durante la caracterización del ECG para el control de transformadores de la red electroenergética, y en el anexo 3 se pueden observar los esquemas relacionales obtenidos por ERECASE.

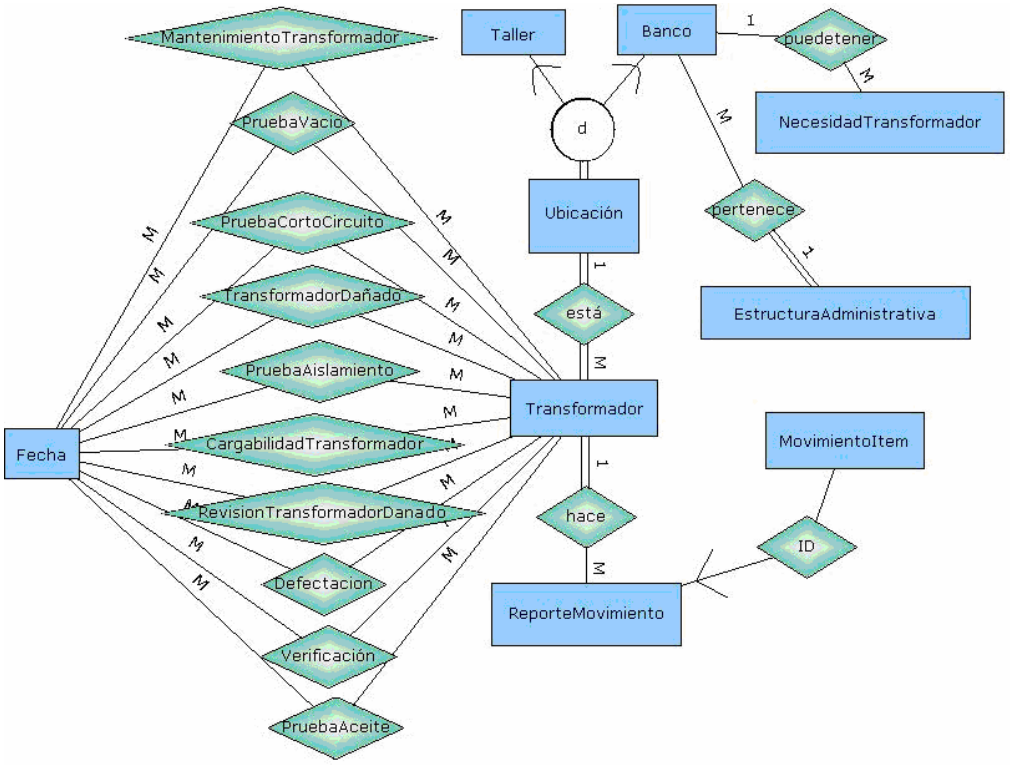

Figura 3.2. Diagrama ERE para el control de transformadores obtenido en ERECASE.

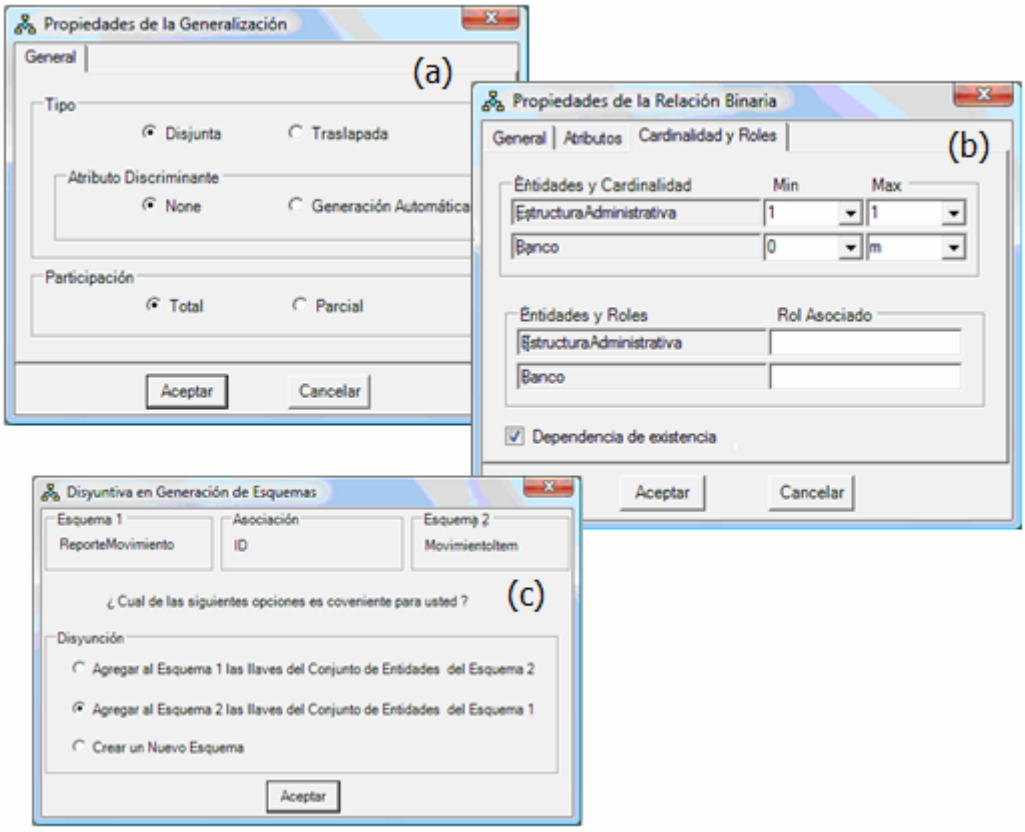

Figura 3.3. Solución a problemas de modelación en ERECASE.

Los nombres de los esquemas obtenidos durante la caracterización del ECG son los siguientes:

- Transformador
- Defectación
- TransformadorDañado
- PruebaAceite
- PruebaVacío
- PruebaCortocircuito
- NecesidadTransformador
- Banco
- Taller
- PruebaAislamiento
- Verificación
- MantenimientoTransformador
- RevisiónTransformadorDañado
- CargabilidadTransformador
- MovimientoItem
- ReporteMovimiento
- EstructuraAdministrativa
- Pertenece

3.2.2. Caracterización de la red y los sitios

El proceso de caracterización de la red y los sitios de procesamiento tiene un alto nivel de automatización mediante NETWIZARD, donde sólo se necesita indicar los sitios donde residirá la BDD, y los agentes cliente y servidor de esta herramienta se encargan de recopilar la información necesaria.

Los sitios identificados son:

- UEB Subcentro Fomento (Sigere_Fto)
- UEB OBE Sancti Spíritus (Tecn_ssp)
- UEB Subcentro La Sierpe (Tecn_Sierpe)
- UEB Subcentro Taguasco (Tecn_Taguasco)
- UEB OBE Trinidad (Tecn_Obetdad)
- UEB OBE Yaguajay (Tecn_Yag)
- UEB OBE Cabaiguán (Tecn_Cab)
- UEB OBE Jatibonico (Tecn_Jco)
- Taller (Transformadores)

Los resultados obtenidos sobre la caracterización de los sitios aparecen en la tabla 3.1, donde la velocidad del procesador (CPU) se mide en gigahertz, la capacidad de disco ocupada (DS) y libre (FS) en gigabytes, y los tiempos de lectura (RAT) y escritura (WAT) en disco en milisegundos. Por otra parte, los resultados obtenidos sobre la caracterización de 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 segundos. La columna SiteName en la tabla 3.1 indica el nombre de cada sitio.

Tabla 3.1. Parámetros de los sitios obtenidos por NETWIZARD.

SiteName	*IP*	*CPU*	*DS*	*FS*	*RAT*	*WAT*
Transformadores	172.19.231.12	2,33	200	67	0,05435	0,07552
Tecn_Cab	172.19.235.16	2,53	120	31	0,08568	0,10454
Tecn_Jco	172.19.229.19	3,00	250	47	0,04322	0,06456
Tecn_ssp	172.19.229.46	2,93	100	32	0,07012	0,09514
Tecn_Obetdad	172.19.233.23	2,33	150	29	0,06322	0,08434
Tecn_Yag	172.19.234.8	1,50	200	35	0,04284	0,06765
Sigere_Fto	172.19.236.10	2,53	149	54	0,05255	0,07367
Tecn_Sierpe	172.19.237.5	1,65	250	24	0,04222	0,06677
Tecn_Taguasco	172.19.238.10	2,83	120	17	0,06522	0,09045

Tabla 3.2. Parámetros de la red obtenidos por NETWIZARD.

FromStation	*ToStation*	*AvgT*	*FromStation*	*ToStation*	*AvgT*
Sigere_Fto	Tecn_Jco	2,36	Tecn_Sierpe	Tecn_Taguasco	3,70
Sigere_Fto	Tecn_Sierpe	4,56	Tecn_Sierpe	Tecn_Obetdad	4,15
Sigere_Fto	Tecn_Taguasco	3,56	Tecn_Sierpe	Tecn_Yag	2,37
Sigere_Fto	Tecn_Obetdad	2,78	Tecn_Sierpe	Tecn_Cab	2,52
Sigere_Fto	Tecn_Yag	3,69	Tecn_Sierpe	Tecn_ssp	2,95
Sigere_Fto	Tecn_Cab	6,20	Tecn_Sierpe	Transformadores	3,60
Sigere_Fto	Tecn_ssp	4,50	Tecn_Taguasco	Tecn_Obetdad	5,80
Sigere_Fto	Transformadores	2,60	Tecn_Taguasco	Tecn_Yag	6,10
Tecn_Jco	Tecn_Taguasco	3,90	Tecn_Taguasco	Tecn_Cab	3,01
Tecn_Jco	Tecn_Obetdad	2,60	Tecn_Taguasco	Tecn_ssp	7,10
Tecn_Jco	Tecn_Yag	2,80	Tecn_Taguasco	Transformadores	2,30
Tecn_Jco	Tecn_Cab	4,93	Tecn_Obetdad	Tecn_Yag	5,40
Tecn_Jco	Tecn_ssp	5,35	Tecn_Obetdad	Tecn_Cab	3,10
Tecn_Jco	Transformadores	2,82	Tecn_Obetdad	Tecn_ssp	6,50
Tecn_Yag	Tecn_Cab	3,45	Tecn_Obetdad	Transformadores	4,50
Tecn_Yag	Tecn_ssp	3,80	Tecn_ssp	Transformadores	2,78
Tecn_Yag	Transformadores	3,70	Tecn_Taguasco	Tecn_Sierpe	3,80
Tecn_Cab	Tecn_ssp	5,23	Tecn_Cab	Transformadores	2,56

3.2.3. Caracterización de aplicaciones

Las aplicaciones más importantes identificadas son las tres siguientes:

1. Movimiento de transformador.

2. Búsqueda de transformador.

3. Verificación del transformador.

El intervalo de tiempo tomado como referencia para la determinación de los parámetros que caracterizan las aplicaciones es la semana.

Movimiento de transformador

La aplicación "Movimiento de transformador" registra la información relacionada con los traslados de transformadores entre instalaciones. Esta aplicación se ejecuta en las OBEs Territoriales 17 veces y en la OBE Provincial 46 veces. En la tabla 3.3 se pueden observar los esquemas y atributos que usa esta aplicación, mientras que en la figura A5.1 del anexo 5 puede observarse la secuencia seguida en APPWIZARD para caracterizar esta aplicación.

Tabla 3.3. Parámetros de la aplicación "Movimiento de transformador" proporcionados a APPWIZARD.

Esquema	*Atributos usados*
REPORTEMOVIMIENTO Cardinalidad: 5000 Selectividad: 58	Atributos de lectura/escritura: Id_Movimiento, FechaEjecución, Descripción, NoMovMedioBásico, H19, EjecutadaPor, Observaciones, OrdenTrabajo, Folio, Año
TRANSFORMADOR Cardinalidad: 7200 Selectividad: 65	Atributos de solo lectura: Id_EAdministrativa, Id_Transformador, Código, NumEmp, Fase, NoSerie Atributos de lectura/escritura: TapEncontrado, TapDejado, UltAcciónVer
MOVIMIENTOITEM Cardinalidad: 12500 Selectividad: 100	Atributos de lectura/escritura: Id_Movimiento, Id_Transformador, Id_EAEquipo, Causamos, Desde, Hacia, FaseConectada

Búsqueda de transformador

La aplicación "Búsqueda de transformador" obtiene información sobre uno o más transformadores teniendo en cuenta una o más características elegidas por el usuario. Esta es una aplicación que sólo realiza lecturas de los datos almacenados y se ejecuta en el taller 18 veces, mientras que en la OBE Territorial se ejecuta 45 veces y en la OBE Provincial 90 veces. En la tabla 3.4 se pueden observar los esquemas y atributos de solo lectura que usa esta aplicación. Una vez obtenidos estos datos, se procede a capturar los predicados simples que indican localidad; que en este caso se corresponden con las estructuras administrativas de la OBE de Sancti Spíritus, las cuales pueden observarse

en la tabla 3.5. En la figura A5.2 del anexo 5 puede observarse la secuencia seguida en APPWIZARD para caracterizar esta aplicación.

Tabla 3.4. Parámetros de la aplicación "Búsqueda de transformador" proporcionados a APPWIZARD.

Esquema	*Atributos usados*
BANCO Cardinalidad: 4500 Selectividad: 45	Código, CódigoAntiguo, Id_EAdirección, Codirección, Seccionalizador, Circuito, Conexión, TipoAlimentación, NSección, TipoTerreno, Id_VoltajeSalida, TipoSalida, Id_VoltajePrimario, Id_EAdministrativa, EstadoOperativo, Id_TipoCarga, NumClientes
TRANSFORMADOR Cardinalidad: 7200 Selectividad: 65	Id_EAdministrativa, Id_Transformador, Código, Numemp, Id_Capacidad, Id_VoltajePrim, NoSerie, NecesidadEmitida, CE, PosiciónBanco, EstadoOperativo, TapEncontrado, TapDejado, NumFase, UltAcciónVer, Id_Voltaje_Secun, NSección, Número, Tipoalimentación, NVerificado, Marca, AñoFabricación, Frecuencia, Fase, Id_CorrienteN, TipoEnfriamiento, PolaridadGrupo, Impedancia
ESTRUCTURA-ADMINISTRATIVA Cardinalidad: 9 Selectividad: 3	Id_EAdministrativa, Nombre, N_Consumidores

Tabla 3.5. Predicados simples asociados con el esquema EstructuraAdministrativa proporcionados a APPWIZARD.

UEB Subcentro Fomento	Id_EAdministrativa = '22'
UEB OBE Jatibonico	Id_EAdministrativa = '23'
UEB Subcentro La Sierpe	Id_EAdministrativa = '24'
UEB Subcentro Taguasco	Id_EAdministrativa = '25'
UEB OBE Trinidad	Id_EAdministrativa = '26'
UEB OBE Yaguajay	Id_EAdministrativa = '36'
UEB OBE Cabaiguán	Id_EAdministrativa = '63'
UEB OBE Sancti Spíritus	Id_EAdministrativa = '64'

Verificación del transformador

La aplicación "Verificación del transformador" capta la información sobre una serie de pruebas que se realizan en el taller a todo transformador nuevo que entra para saber si el

mismo está apto para su uso. Esta aplicación se ejecuta 25 veces en el taller. En la tabla 3.6 se pueden observar los esquemas y atributos (de lectura/escritura) que usa esta aplicación y en la figura A5.3 del anexo 5 puede observarse la secuencia seguida en APPWIZARD para caracterizar esta aplicación.

Tabla 3.6. Parámetros de la aplicación "Verificación del transformador" proporcionados a APPWIZARD.

Esquema	*Atributos usados*
TRANSFORMADOR Cardinalidad: 7200 Selectividad: 65	Id_EAdministrativa, Id_Transformador, Numemp, Número, Id_Capacidad, Id_VoltajePrim, NoSerie, EstadoOperativo, NumFase, Id_Voltaje_Secun, NSección, Tipoalimentación, Marca, AñoFabricación, Frecuencia, Id_CorrienteN, TipoEnfriamiento, PolaridadGrupo, Impedancia
PRUEBAAISLAMIENTO Cardinalidad: 5000 Selectividad: 325	Id_Transformador, Id_EAEquipo, Fecha, Temperatura, BajaAltaTierra1, AltaBajaTierra1, BajaAltaTierra2, AltaBajaTierra2, VoltMegger
PRUEBAACEITE Cardinalidad: 5000 Selectividad: 325	Id_Transformador, Id_EAEquipo, Fecha, NivelAceite, NivelPromedio, ColoraciónAceite, AcidezAceite,
PRUEBAVACÍO Cardinalidad: 5000 Selectividad: 325	Id_Transformador, Id_EAEquipo, Fecha, KVacío, KNúcleo, CorrVacío100, CorrVacío110, PerdNúcleo100, PerdNúcleo110
VERIFICACIÓN Cardinalidad: 15000 Selectividad: 125	Id_Transformador , Id_EAEquipo, FechaEjecución, TipoTrabajo, CartaTécnica, Hermeticidad, Polaridad, Aceptado, EjecutadaPor, Observaciones, OrdenTrabajo, AltoVoltajeFrecIndustrial, PruebaAltoVoltaje, Folio, Name, Año, Id_FabricanteActual, UltAccionVer, Id_FabricanteAnterior, Impedancia, Peso, PesoAislante, Código, Temperatura
PRUEBACORTOCIRCUITO Cardinalidad: 5000 Selectividad: 325	Id_Transformador, Id_EAEquipo, Fecha, Voltaje, PérdidaCobre, Impedancia

3.3. Modelación lógica

Durante el análisis se reconoció que el esquema TRANFORMADOR tiene atributos que sólo se usan en el TALLER por lo que se realiza primeramente una FV utilizando la herramienta FRAGMENTER, quedando en el fragmento Transformador0 los atributos usados por la aplicación 1 "Verificación del Transformador"; y el resto de los atributos en el fragmento Transformador1 (véase la tabla 3.7).

Posteriormente se realiza una fragmentación horizontal, obteniéndose una FHP sobre ESTRUCTURAADMINISTRATIVA (véase la tabla 3.8); y seguidamente son fragmentados mediante FHD todos los esquemas con llave foránea referenciando la llave primaria del esquema ESTRUCTURAADMINISTRATIVA (véase la tabla 3.9).

La información sobre esquemas propietarios y miembros es obtenida de la tabla LINKS del catálogo, generada durante la modelación conceptual con la herramienta ERECASE.

Tabla 3.7. Resultados de la FV obtenidos por FRAGMENTER.

Esquema	*TRANFORMADOR*
Fragmento:	Transformadores0
Atributos:	Id_EAdministrativa, Id_Transformador, NumEmp, Id_Capacidad, Marca, Id_VoltajePrim, NoSerie, EstadoOperativo, NumFase, Id_Voltaje_Secun, NSección, Número, TipoAlimentación, AñoFabricación, Frecuencia, Id_CorrienteN, TipoEnfriamiento, PolaridadGrupo, Impedancia
Fragmento:	Transformadores1
Atributos:	Id_Transformador, Código, Fase, NecesidadEmitida, CE, PosiciónBanco, TapEncontrado, TapDejado, Id_EAdministrativa, UltAcciónVer, NVerificado

A continuación se realizó una FHD sobre el fragmento Transformador1 según la ESTRUCTURAADMINISTRATIVA a la que pertenece el BANCO donde se encuentra instalado el transformador (a través del atributo Código del Banco); asimismo el esquema NECESIDADTRANSFORMADOR es sometido a FHD por el atributo Código del Banco. También se identificó que los esquemas de las pruebas, obtenidos a partir de las asociaciones entre FECHA y TRANSFORMADOR en el ECG, tienen dependencia de existencia de TRANSFORMADOR, motivando a realizar FHD para cada prueba por el atributo Id_Transformador. Efectivamente, esta FHD obtenida considerando la dependencia de existencia, facilita el acople local entre los fragmentos de TRANSFORMADOR y sus respectivas pruebas.

Tabla 3.8. Resultados de la FHP obtenidos por FRAGMENTER.

Esquema	*Fragmento*	*Expresión*
EstructuraAdministrativa	EstructuraAdministrativa0	Id_EAdministrativa = ‘22’
	EstructuraAdministrativa1	Id_EAdministrativa = ‘23’
	EstructuraAdministrativa2	Id_EAdministrativa = ‘24’
	EstructuraAdministrativa3	Id_EAdministrativa = ‘25’
	EstructuraAdministrativa4	Id_EAdministrativa = ‘26’
	EstructuraAdministrativa5	Id_EAdministrativa = ‘36’
	EstructuraAdministrativa6	Id_EAdministrativa = ‘63’
	EstructuraAdministrativa7	Id_EAdministrativa = ‘64’
	EstructuraAdministrativa8	Id_EAdministrativa = ‘65’

La posibilidad de eliminar predicados minterms en el proceso de fragmentación llevado a cabo por FRAGMENTER es de gran utilidad, ya que permite excluir aquellos predicados que constituyen hechos incorrectos o contradictorios de acuerdo al dominio del problema, evitándose la generación innecesaria de algunos fragmentos.

Tabla 3.9. Resultados de la FHD obtenidos por FRAGMENTER.

Esquema: PERTENECE

Fragmento: Pertenece0 (1)

Expresión: Pertenece0.EstructuraAdministrativa__Id_EAdministrativa=
EstructuraAdministrativa0.Id_EAdministrativa

Esquema: BANCO

Fragmento: Banco0 (2)

Expresión: Pertenece0.Banco__Código=Banco0.Código

(1) De igual forma, se obtuvieron los fragmentos desde Pertenece1 hasta Pertenece7 para cada ESTRUCTURAADMINISTRATIVA.

(2) Asimismo, se obtuvieron los fragmentos desde Banco1 hasta Banco7 para cada Banco según los fragmentos obtenidos para el esquema PERTENECE.

Los resultados de la asignación lógica realizada por ALLOCATOR se encuentran en la tabla 3.10. De aquí se puede observar que la integración entre las herramientas presentadas en este trabajo ahorran una gran cantidad de tiempo y esfuerzo en el proceso de diseño.

3.4. Modelación física

En la tabla 3.10 se muestran los fragmentos asignados a cada sitio por la herramienta ALLOCATOR.

Tabla 3.10. Resultados de la asignación obtenida por ALLOCATOR.

Sitio	*Fragmento*
Tecn_Yag	Pertenece5.1, PruebaCortocircuito5.1, Transformador1.5, Banco5.1, EstructuraAdministrativa5.1, PruebaAislamiento5.1, PruebaVacío5.1, TransformadorDañado5.1, PruebaAceite5.1, ReporteMovimiento.5, RevisiónTransformadorDañado5.1, MovimientoItem.5, Verificación5.1, MantenimientoTransformador5.1, CargabilidadTransformador5.1, NecesidadTransformador5.1,
Tecn_Cab	Pertenece6.1, PruebaCortocircuito6.1, Transformador1.6, Banco6.1, EstructuraAdministrativa6.1, PruebaAislamiento6.1, PruebaVacío6.1, TransformadorDañado6.1, PruebaAceite6.1, ReporteMovimiento.6, RevisiónTransformadorDañado6.1, MovimientoItem.6, Verificación6.1, MantenimientoTransformador6.1, CargabilidadTransformador6.1, NecesidadTransformador6.1,
Tecn_Jco	Pertenece1.1, PruebaCortocircuito1.1, Transformador1.1, Banco1.1, EstructuraAdministrativa1.1, PruebaAislamiento1.1, PruebaVacío1.1, TransformadorDañado1.1, PruebaAceite1.1, ReporteMovimiento.1, RevisiónTransformadorDañado1.1, MovimientoItem.1, Verificación1.1, MantenimientoTransformador1.1, CargabilidadTransformador1.1, NecesidadTransformador1.1
Sigere_Fto	Pertenece0.1, PruebaCortocircuito0.1, Transformador1.0, Banco0.1, EstructuraAdministrativa0.1, PruebaAislamiento0.1, PruebaVacío0.1, TransformadorDañado0.1, PruebaAceite0.1, ReporteMovimiento.0, RevisiónTransformadorDañado0.1, MovimientoItem.0, Verificación0.1, MantenimientoTransformador0.1, CargabilidadTransformador0.1, NecesidadTransformador0.1
Tecn_ssp	Pertenece7.1, PruebaCortocircuito7.1, Transformador1.7, Banco7.1, EstructuraAdministrativa7.1, PruebaAislamiento7.1, PruebaVacío7.1, TransformadorDañado7.1, PruebaAceite7.1, ReporteMovimiento.7, RevisiónTransformadorDañado7.1, MovimientoItem.7, Verificación7.1, MantenimientoTransformador7.1, CargabilidadTransformador7.1, NecesidadTransformador7.1

Sitio	*Fragmento*
Tecn_Sierpe	Pertenece2.1, PruebaCortocircuito2.1, Transformador1.2, EstructuraAdministrativa2.1, PruebaAislamiento2.1, TransformadorDañado2.1, PruebaAceite2.1, Banco2.1, RevisiónTransformadorDañado2.1, MovimientoItem.2, PruebaVacío2.1, MantenimientoTransformador2.1, Verificación2.1, CargabilidadTransformador2.1, NecesidadTransformador2.1, ReporteMovimiento.2
Transformadores	Transformador0.1, PruebaCortocircuito8.1, ReporteMovimiento.8, EstructuraAdministrativa8.1, PruebaAislamiento8.1, Taller.0, TransformadorDañado8.1, Defectación.8, PruebaAceite8.1, RevisiónTransformadorDañado8.1, PruebaVacío8.1, MantenimientoTransformador8.1, Verificación8.1, CargabilidadTransformador8.1, MovimientoItem.8, NecesidadTransformador8.1
Tecn_Taguasco	Pertenece3.1, PruebaCortocircuito3.1, Transformador1.3, EstructuraAdministrativa3.1, PruebaAislamiento3.1, TransformadorDañado3.1, PruebaAceite3.1, Banco3.1, RevisiónTransformadorDañado3.1, MovimientoItem.3, PruebaVacío3.1, MantenimientoTransformador3.1, Verificación3.1, CargabilidadTransformador3.1, NecesidadTransformador3.1, ReporteMovimiento.3
Tecn_Obetdad	Pertenece4.1, PruebaCortocircuito4.1, Transformador1.4, EstructuraAdministrativa4.1, PruebaAislamiento4.1, TransformadorDañado4.1, PruebaAceite4.1, Banco4.1, RevisiónTransformadorDañado4.1, MovimientoItem.4, PruebaVacío4.1, MantenimientoTransformador4.1, Verificación4.1, CargabilidadTransformador4.1, NecesidadTransformador4.1, ReporteMovimiento.4

La creación de los esquemas físicos es realizada por la herramienta ALLOCATOR, que posee un módulo de generación de código encargado de construir los scripts que

deberán ser ejecutados en el servidor de BD Microsoft SQL Server instalado en cada sitio elegido. Estos scripts recrean el entorno de replicación usando secuencias de comando en Transact-SQL.

El primer paso de cada script lo constituye el reconocimiento de los demás servidores a través de la sentencia sp_addlinkedserver, donde la variable @server toma como valor el nombre del servidor SQL encargado del almacenamiento de la BD en cada sitio. Por ejemplo, para el sitio SIGERE_FTO reconocer el servidor del sitio TECN_JCO se generó la sentencia: EXECUTE sp_addlinkedserver @server = 'TECN_JCO\ Tecn_Jco'. Asimismo, se generaron sentencias imilares para reconocer el resto de los servidores TECN_CAB, TECN_SSP, TECN_TAGUASCO, TECN_OBETDAD, TECN_YAG, TECN_SIERPE y TRANSFORMADORES.

Posteriormente se configuró el parámetro que especifica cómo se realizará la propagación de los cambios entre réplicas en cada servidor. En todos los casos se elige la propagación lazy como se expresó en el capítulo 2 para lograr mejor desempeño ya que los tiempos de respuesta son menores al no existir ninguna comunicación intratransacción: EXECUTE sp_serveroption @server='SIGERE_FTO\Sigere_Fto', @optname='lazy schema validation', @optvalue='true'.

Se activó la variable XACT_ABORT con el valor ON para asegurar que cualquier acción realizada por una transacción sea deshecha en caso de error. Seguidamente se procedió a la creación de las BD locales.

A continuación, el servidor de UEB OBE Sancti Spíritus (OBE Provincial) se configuró como publicador. Todas las publicaciones realizadas son de tipo Mezcla con filtros horizontales para los fragmentos obtenidos mediante FHP, filtros verticales para la FV y filtros dinámicos que permiten realizar la FHD. Las suscripciones en los nodos de las OBEs territoriales se configuraron del tipo PULL que es el tipo de suscripción elegido por el método M-GC presentado en el capítulo 2. El primer paso en la configuración de la suscripción Pull fue la activación de la propiedad allow_pull con el valor true a través del procedimiento sp_changemergepublication, seguido de las ejecuciones de los procedimientos sp_addmergepullsubscription y sp_addmergepullsubscription_agent. La elección del tipo de suscripción Pull es la más apropiada en ambientes distribuidos donde hay posibilidades de desconexión temporal, aunque también puede usarse en redes más fiables como LAN y WAN. Esta solución resultó ser factible atendiendo a las propias características de las redes de comunicaciones de la OBE de Sancti Spíritus,

donde las conexiones se realizan por MODEM y se tiene una programación de una conexión diaria, en horarios preferentemente de madrugada para aprovechar el poco tráfico existente por las líneas.

3.5. Conclusiones parciales

La aplicación de SIADBDD al problema de control de los transformadores simplificó la modelación conceptual, lógica y física de la base de datos distribuida de este sistema, y disminuyó el tiempo de ejecución y el esfuerzo requerido en las tareas de diseño.

En la solución a este problema se verificaron algunas características importantes de las herramientas utilizadas, en particular:

- ERECASE posee construcciones adecuadas para la modelación del mismo.
- APPWIZARD permitió obtener los datos necesarios para la caracterización de las aplicaciones; sin embargo requiere que el diseñador posea suficiente conocimiento del problema para poder estimar los parámetros de cardinalidad, selectividad y frecuencia de accesos solicitados por este asistente.
- Con el empleo de NETWIZARD se pudieron obtener los parámetros de la red y de los sitios de procesamiento, lo cual sería muy difícil de estimar o realizar manualmente. Este asistente usa el canal HTTP, por lo que no presenta dificultad para pasar por firewalls.
- La herramienta FRAGMENTER incluye los tres tipos de fragmentación, y los resultados obtenidos son los esperados.
- La herramienta ALLOCATOR permitió realizar la asignación lógica y física para el problema planteado, el cual fue abordado mediante los dos métodos proporcionados, y los resultados obtenidos por estos métodos coincidieron. Aunque esta herramienta es de gran ayuda, aún es necesario seguir perfeccionándola para ofrecer otras variantes de solución tanto desde el punto de vista lógico como físico.

Desde el punto de vista metodológico, la solución para la OBE de Sancti Spíritus puede ser de utilidad para la OBE de cualquier provincia, ya que se corresponden tanto la estructura organizativa, como los datos y aplicaciones que manejan estos datos; aunque será necesario recaracterizar los sitios de procesamiento y la red de comunicación para posteriormente realizar una distribución apropiada a cada provincia.

Por su grado de generalidad, estos sistemas han sido empleados para fines docentes en la carrera de Ciencia de la Computación en la Universidad Central “Marta Abreu” de Las Villas. Los resultados de esta investigación forman parte del proyecto de

investigación y desarrollo “Tecnologías de Bases de Datos aplicadas a problemas”, Número: 01700031 del Programa Nacional de Tecnologías de la Información, apoyado por el Ministerio de Ciencia Tecnología y Medio Ambiente (CITMA) de Cuba.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

El conjunto de herramientas obtenidas colaboran en un ambiente integrado como ayuda al diseño de BDD, y además pueden usarse de forma independiente. Estas herramientas les permiten a los diseñadores obtener esquemas distribuidos de datos hasta el nivel físico; aunque en la literatura consultada sólo se ofrecen soluciones hasta el nivel lógico.

1. ERECASE brinda un amplio conjunto de construcciones del modelo ERE y realiza diferentes validaciones estructurales a éste. A esta herramienta se le adicionó una nueva construcción reportada en la literatura pero no incluida en ninguna de las herramientas consultadas; precisamente esta construcción sirve de ayuda a la FHD.

2. NETWIZARD incluye dos agentes, un cliente y un servidor, que usan la tecnología de comunicación .Net Remoto para la obtención automatizada de la información necesaria para la caracterización de la red y los sitios de procesamiento, la cual es muy difícil de estimar; y emplea el canal HTTP, por lo que no presenta dificultad para pasar por firewalls. En la literatura no se reportan ayudas en este sentido.

3. APPWIZARD es un asistente fácil de usar para capturar de una gran variedad de información sobre las aplicaciones más representativas en el universo de discurso objeto de modelación.

4. FRAGMENTER realiza la fragmentación de esquemas globales para obtener un conjunto de fragmentos lógicos que son objeto de ubicación en los diferentes sitios de procesamiento donde residirá la BDD. Esta herramienta integra tres módulos independientes para los diferentes tipos de fragmentación: FHP, FHD y FV. La FHP se basa en el método clásico reportado en [182] sustituyendo el método COM_MIN por M-COM-MIN* de mayor eficiencia, que también asegura completitud y minimalidad. La FHD emplea como criterios para seleccionar las llaves foráneas, que se deben priorizar para favorecer los acoples con las llaves primarias respectivas, aquellos que aparecen en la literatura, y se adiciona un nuevo criterio que tiene en cuenta el concepto de dependencia de existencia presente en algunas asociaciones. La FV se realiza siguiendo el enfoque clásico recogido en la literatura [182].

5. ALLOCATOR implementa dos métodos metaheurísticos que consideran la replicación, uno basado en Algoritmos Genéticos Generacionales y otro basado en el

método Q-Learning de Aprendizaje Reforzado, para dar solución a un problema de decisión de la clase NP-Completo. Estos métodos pueden ser elegidos libremente por los diseñadores para materializar diseños de distribución. Esta herramienta incluye la creación de esquemas físicos 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, motor más popular en Cuba.

Recomendaciones

1. Evaluar otras alternativas de solución al problema de asignación tanto lógica como física.
2. Probar las herramientas con otros casos, similares y diferentes, para continuar la validación de las mismas.

REFERENCIAS BIBLIOGRÁFICAS

[1] Abe, N.; Biermann, A.W. y Long, P.M.: Reinforcement Learning with Immediate Rewards and Linear Hypotheses. Algorithmica, 37(4) 263-293 (2003)

[2] Aekaterinidis, I. y Triantafillou, P.: Internet Scale String Attribute Publish/Subscribe Data Networks. Proceedings of the International Conference on Information and Knowledge Management (CIKM'05), ACM, Bremen, Germany (2005) 44-51

[3] Akal, F.; Türker, C.; Schek, H.-J.; Breitbart, Y.; Grabs, T. y Veen, L.: Fine-grained lazy replication and scheduling with freshness and correctness guarantees. Proceedings of the 31st VLDB Conference, Swiss Federal Institute of Technology Zurich, Trondheim, Norway (2005) 565-576

[4] Akal, F.; Türker, C.; Schek, H.-J.; Grabs, T. y Breitbart, Y.: Fine-grained lazy replication with strict freshness and correctness guarantees, Swiss Federal Institute of Technology, TR 457-2004 (2004)

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

[6] Amza, C.; Cox, A.L. y Zwaenepoel, W.: Distributed Versioning: Consistent Replication for Scaling Back-End Databases of Dynamic Content Web Sites. En: M. Endler y D.C. Schmidt (eds.): Proceedings of the International Middleware Conference 2003 ACM/IFIP/USENIX, Springer, Rio de Janeiro, Brazil (2003) 282-304

[7] Anderson, T.A.; Breitbart, Y.; Korth, H.F. y Wool, A.: 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, ACM Press, Seattle, WA, USA (1998) 484-495

[8] Androutsellis-Theotokis, S. y Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv., 36(4) 335-371 (2004)

[9] Apers, P.M.G.: Data Allocation in Distributed Database Systems. ACM Trans. Database Syst., 13(3) 263-304 (1988)

[10] Awerbuch, B.; Bartal, Y. y Fiat, A.: Competitive distributed file allocation. Inf. Comput., 185(1) 1-40 (2003)

[11] Aygun, R.S. y Patil, A.S.: PressBase: A presentation synchronization database for distributed multimedia systems IEEE Transactions on Multimedia, 8(2) 289-296 (2006)

[12] Baião, F.A.; Mattoso, M.; Shavlik, J.W. y Zaverucha, G.: 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, Springer, Szeged, Hungary (2003) 57-74

[13] Baião, F.A.; Mattoso, M. y Zaverucha, G.: Towards an Inductive Design of Distributed Object Oriented Databases. Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems, IEEE Computer Society, New York City, NY, USA (1998) 188-197

[14] Baião, F.A.; Mattoso, M. y Zaverucha, G.: A Framework for the Design of Distributed Databases. En: W. Litwin y G. Lévy (eds.): Records of the 4th International Meeting on Distributed Data & Structures 4 (WDAS 2002), Carleton Scientific, Paris, France (2002) 29-36

[15] Baião, F.A.; Mattoso, M. y Zaverucha, G.: A Distribution Design Methodology for Object DBMS. Distributed and Parallel Databases, 16(1) 45-90 (2004)

[16] Bataller, J.; Decker, H.; Irún-Briz, L. y Muñoz-Escoí, F.D.: Replication for Web-Based Collaboration. Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA 2004). IEEE Computer Society, Zaragoza, Spain (2004) 247-253

[17] Belokosztolszki, A.; Eyers, D.M.; Pietzuch, P.R.; Bacon, J. y Moody, K.: 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), ACM, San Diego, CA, USA (2003) 1-8

[18] Bellatreche, L.; Karlapalem, K. y Li, Q.: Complex Methods and Class Allocation in Distributed Object-Oriented Database Systems. International Conference on Object Oriented Information Systems (OOIS 1998), Paris, France (1998) 239-256

[19] Bellatreche, L.; Karlapalem, K. y Simonet, A.: Algorithms and support for horizontal class partitioning in object-oriented databases. Distributed and Parallel Databases, 8(2) 155-179 (2000)

[20] Bellatreche, L. y Simonet, A.: Horizontal Fragmentation in Distributed Object Database Systems. En: L. Böszörményi (ed.): Proceedings of the Third International ACPC Conference with Special Emphasis on Parallel Databases and Parallel I/O, Springer, Klagenfurt, Austria (1996) 223-226

[21] Bellatreche, L.; Karlapalem, K. y Simonet, A.: Horizontal Fragmentation in Distributed Object Database Systems. Proceedings of the 8th International

Conference on Database and Expert Systems Applications DEXA '97, Springer, Toulouse, France (1997) 58-67

[22] Bellatreche, L.; Simonet, A. y Simonet, M.: Vertical Fragmentation in Distributed Object Database Systems with Complex Attributes and Methods. En: R. Wagner y H. Thomas (eds.): Proceedings of the Seventh International Workshop on Database and Expert Systems Applications DEXA '96, IEEE-CS Press, Zurich, Switzerland (1996) 15-21

[23] Berenson, H.; Bernstein, P.A.; Gray, J.; Melton, J.; O'Neil, E.J. y O'Neil, P.E.: 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 ACM Press, San Jose, CA, USA (1995) 1-10

[24] Bernstein, P.A.; Hadzilacos, V. y Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley (1987)

[25] Bhalla, S. y Hasegawa, M.: 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, Vol. 3824, Springer (2005) 203-213

[26] Birman, K.P.: Building Secure and Reliable Network Applications. En: T. Masuda; Y. Masunaga y M. Tsukamoto (eds.): Worldwide Computing and Its Applications, International Conference, WWCA '97, Springer, Tsukuba, Japan (1997) 15-28

[27] Böhm, K.; Grabs, T.; Röhm, U. y Schek, H.-J.: 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) Springer, Munich, Germany (2000) 435-444

[28] Brahmadathan, K. y Ramarao, K.V.S.: On the design of replicated databases. Inf. Sci., 65(1-2) 173-187 (1992)

[29] Breitbart, Y.; Komondoor, R.; Rastogi, R.; Seshadri, S. y Silberschatz, A.: Update Propagation Protocols for Replicated Databases. En: A. Delis; C. Faloutsos y S. Ghandeharizadeh (eds.): Proceedings ACM SIGMOD International Conference on Management of Data 1999, ACM Press, Philadelphia, Pennsylvania, USA (1999) 97-108

[30] Breitbart, Y. y Korth, H.F.: Replication and Consistency: Being Lazy Helps Sometimes. Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM Press, Tucson, Arizona, USA (1997) 173-184

[31] Breitbart, Y. y Korth, H.F.: Replication and Consistency in a Distributed Environment. J. Comput. Syst. Sci., 59(1) 29-69 (1999)

[32] Cacheda, F.; Plachouras, V. y Ounis, I.: 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, Springer, Sunderland, UK (2004) 394-408

[33] Cecchet, E.; Marguerite, J. y Zwaenepoel, W.: C-JDBC: Flexible Database Clustering Middleware. Proceedings of the FREENIX Track: 2004 USENIX Annual Technical Conference, USENIX, Boston, Marriott Copley Place, Boston, MA, USA (2004) 9-18

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

[35] Ceri, S.; Navathe, S.B. y Wiederhold, G.: Distribution Design of Logical Database Schemas. IEEE Trans. Software Eng., 9(4) 487-504 (1983)

[36] Ceri, S.; Negri, M. y Pelagatti, G.: Horizontal Data Partitioning in Database Design. Proceedings of the ACM SIGMOD International Conference on Management of Data 1982. ACM Press, Orlando, Florida (1982) 128-136

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

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

[39] Ceri, S.; Pernici, B. y Wiederhold, G.: Distributed Database Design Methodologies. IEEE Database Eng. Bull., 75(5) 533-546 (1987)

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

[41] Codd, E.F.: The Relational Model for Database Management, Version 2. Addison-Wesley (1990)

[42] Computer-Associates: AllFusion ERwin Data Modeler, http://www.ca.com.

[43] Cornell, D.W. y Yu, P.S.: A Vertical Partitioning Algorithm for Relational Databases. Proceedings of the Third International Conference on Data Engineering, IEEE Computer Society, Los Angeles, CA, USA (1987) 30-35

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

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

[46] Coulon, C.; Pacitti, E. y Valduriez, P.: Optimistic Preventive Replication in a Database Cluster. Proceedings of the 21èmes Journées Bases de Données Avancées (BDA), INRIA, France (2005) 1-19

[47] Coulon, C.; Pacitti, E. y Valduriez, P.: 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, Springer, Valencia, Spain (2005) 74-188

[48] Cruz, F.; Baião, F.A.; Mattoso, M. y Zaverucha, G.: 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, Springer, Porto de Galinhas, Recife, Brazil (2002) 216-226

[49] Cruz, L.: Automatización del Diseño de la Fragmentación Vertical y Ubicación en Bases de Datos Distribuidas usando Métodos Heurísticos y Exactos. Universidad Virtual del ITESM, Monterrey, NL, México (1999)

[50] Chakravarthy, S.; Muthuraj, J.; Varadarajan, R. y Navathe, S.B.: An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis. Distributed and Parallel Databases, 2(2) 183-207 (1994)

[51] Chakravarthy, S. y Vontella, N.: 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, Springer, Bhubaneswar, India (2004) 106-116

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

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

[54] Chang, V.N.: Método de Dos Fases para la Asignación de Datos en una Base de Datos Distribuida. Departamento Académico de Informática, Universidad Nacional de Trujillo, Perú, http://www.informatizate.net, Acceso: 27 de octubre de 2004 (2004)

[55] CharonWare: Case Studio 2, http://www.casestudio.com.

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

[57] Chen, P.P.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Trans. Database Syst., 1(1) 9-36 (1976)

[58] Chen, X.; Chen, Y. y Rao, F.: 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), ACM, San Diego, CA, USA (2003) 6

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

[60] ChilliSource: Database Design Studio Lite, http://www.chillisource.com/

[61] Chinchwadkar, G.S. y Goh, A.: An Overview of Vertical Partitioning in Object-Oriented Databases. Comput. J., 42(1) 39-50 (1999)

[62] Chiu, G.-M. y Raghavendra, C.S.: 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, IEEE, San Francisco, CA, USA (1990) 827-833

[63] Choi, S.P.M.; Zhang, N.L. y Yeung, D.-Y.: 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, Vol. 2, CSREA Press, Las Vegas, Nevada, USA (2004) 752-758

[64] Chu, W.W. y Ieong, I.T.: A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems. IEEE Trans. Software Eng., 19(8) 804-812 (1993)

[65] Chundi, P.; Rosenkrantz, D.J. y Ravi, S.S.: Deferred Updates and Data Placement in Distributed Databases. En: S.Y.W. Su (ed.): Proceedings of the Twelfth International Conference on Data Engineering, IEEE Computer Society, New Orleans, Louisiana (1996) 469-476

[66] Datanamic: DeZign for databases, http://www.datanamic.com.

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

[68] Daudjee, K. y Salem, K.: Lazy Database Replication with Ordering Guarantees. Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, IEEE Computer Society, Boston, MA, USA (2004) 424-435

[69] Daudjee, K. y Salem, K.: Lazy Database Replication with snapshot isolation. Proceedings of the 32nd International Conference on Very Large Databases ACM VLDB Endowment, Seoul, Korea (2006) 715 - 726

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

[71] Decker, H.; Muñoz-Escoí, F.D.; Irún-Briz, L.; Castro, P.; Bernabéu-Aubán, J.M.; Calero-Monteagudo, A.; Esparza-Peidro, J.; Bataller, J. y Galdámez, P.: Enhancing the Availability of Networked Database Services by Replication and Consistency Maintenance. Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03), IEEE Computer Society, Prague, Czech Republic (2003) 531-535

[72] Delcambre, L.; Khatri, V.; Wand, Y.; Williams, B.; Woo, C. y Zozulia, M.: Eliciting Data Semantics Via Top-Down and Bottom-Up Approaches: Challenges and Opportunities. En: D.W. Embley; A. Olivé y S. Ram (eds.): Proceedings of the 25th International Conference on Conceptual Modeling - ER 2006, Lecture Notes in Computer Science Vol. 4215, Springer, Tucson, Arizona, USA (2006) 548 – 551

[73] Dowdy, L.W. y Foster, D.V.: Comparative Models of the File Assignment Problem. ACM Comput. Surv., 14(2) 287-313 (1982)

[74] Du, J.; Alhajj, R. y Barker, K.: Genetic algorithms based approach to database vertical partition. Journal of Intelligent Information Systems, 26(2) 167-183 (2006)

[75] Elmasri, R. y Navathe, S.B.: Fundamentals of Database Systems, 4th edition. Addison-Wesley (2004)

[76] Elmeleegy, K.; Chanda, A.; Cox, A.L. y Zwaenepoel, W.: Lazy Asynchronous I/O for Event-Driven Servers. Proceedings of the General Track: 2004 USENIX Annual Technical Conference, USENIX, Boston Marriott Copley Place, Boston, MA, USA (2004) 241-254

[77] Elnikety, S.; Zwaenepoel, W. y Pedone, F.: Database Replication Using Generalized Snapshot Isolation. Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05), IEEE Computer Society (2005) 73-84

[78] Embarcadero: ER/Studio, http://www.embarcadero.com.

[79] Esparza, P.J.; Muñoz-Escoí, F.D.; Irún-Briz, L. y Bernabéu-Aubán, J.M.: RJDBC: A Simple Database Replication Engine. Proceedings of the 6th International Conference on Enterprise Information Systems ICEIS 2004, Porto, Portugal (2004) 587-590

[80] Eugster, P.T.; Felber, P.A.; Guerraoui, R. y Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Computing Surveys, 35(2) 114-131 (2003)

[81] Evergreen-Software-Tools: EasyCase Professional, http://www.esti.com/

[82] Ezeife, C.I.: Selecting and materializing horizontally partitioned warehouse views. Data & Knowledge Engineering, 36(185-210 (2001)

[83] Ezeife, C.I. y Barker, K.: A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System. Distributed and Parallel Databases, 3(3) 247-272 (1995)

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

[85] Ezeife, C.I. y Dey, P.: Incremental Horizontal Fragmentation of Database Class Objects. Proceedings of the 5th International Conference on Enterprise Information Systems ICEIS (1) 2003, Angers, France (2003) 239-245

[86] Fan, R. y Lynch, N.A.: Brief announcement: efficient replication of large data objects. Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003), ACM, Boston, Massachusetts, USA (2003) 335

[87] Fan, R. y Lynch, N.A.: Efficient Replication of Large Data Objects. En: F.E. Fich (ed.): Proceedings of the 17th. International Conference on Distributed Computing DISC 2003, Springer, Sorrento, Italy (2003) 75-91

[88] Fisher, M.L. y Hochbaum, D.S.: Database Location in Computer Networks. Journal of the ACM, 27(4) 718-735 (1980)

[89] Fornaciari, W.; Piuri, V.; Prestileo, A. y Zaccaria, V.: 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, Springer, Montreal, Canada (2001) 153-162

[90] Fraire, H.; Castilla, G.; Hernandez, A.; Gomez, C.; Mora, O.G. y Godoy, V.A.: A model for the distribution design of distributed databases and an approach to solve large instances. Proceedings of the International Workshop on Distributed

Computing - IWDC 2005. Lecture Notes In Computer Science, Vol. 3741, Springer, Kharagpur, India (2005) 506-511

[91] Frølund, S.; Kalogeraki, V.; Pedone, F. y Pruyne, J.: Scalable State Replication with Weak Consistency. 1st International Workshop on Challenges of Large Applications in Distributed Environments (CLADE 2003), IEEE Computer Society, Seattle, WA, USA (2003) 96

[92] Fu, K.; Kaashoek, M.F. y Mazières, D.: Fast and secure distributed read-only file system. ACM Transactions on Computer Systems, 20(1) 1-24 (2002)

[93] Fung, C.-W.; Karlapalem, K. y Li, Q.: An Analytical Approach Towards Evaluating Method-Induced Vertical Partitioning Algorithms, Department of Computer Science. University of Science and Technology, HKUST-CS96-33 (1996)

[94] Fung, C.-W.; Karlapalem, K. y Li, Q.: 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), World Scientific, Melbourne, Australia (1997) 11-20

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

[96] Fung, C.-W.; Karlapalem, K. y Li, Q.: Cost-driven vertical class partitioning for methods in object oriented databases. VLDB Journal, 12(3) 187-210 (2003)

[97] Gançarski, S.; Naacke, H.; Pacitti, E. y Valduriez, P.: 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, Springer, Irvine, California, USA (2002) 410-428

[98] Gançarski, S.; Naacke, H. y Valduriez, P.: 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), Carleton Scientific, Paris, France (2002) 159-170

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

[100] Goel, S.; Sharda, H. y Taniar, D.: 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), Springer, Leipzig, Germany (2003) 93-103

[101] Goel, S.; Sharda, H. y Taniar, D.: Transaction Management in Distributed Scheduling Environment for High Performance Database Applications. Proceedings of the 5th International Workshop on Distributed Computing - IWDC 2003, Springer, Kolkata, India (2003) 120-130

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

[103] Gray, J.: 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) Vol. 3255, Springer, Budapest, Hungary (2004)

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

[105] Hababeh, I.O.; Bowring, N. y Ramachandran, M.: 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 (2003) 268-274

[106] Hababeh, I.O.; Bowring, N. y Ramachandran, M.: 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 (2004) 54 - 59

[107] Hartmann, J.; Palma, R. y Sure, Y.: OMV - Ontology Metadata Vocabulary. Proceedings of the International Symposium on Wearable Computers (ISWC05), IEEE Computer Society, Osaka, Japan (2005) 9

[108] Hoffer, J.A.: An integer programming formulation of computer data base design problems. Inf. Sci., 11(1) 29-48 (1976)

[109] Hoffer, J.A. y Severance, D.G.: The Use of Cluster Analysis in Physical Data Base Design. En: D.S. Kerr (ed.): Proceedings of the International Conference on Very Large Data Bases, ACM, Framingham, MA, USA (1975) 69-86

[110] Holliday, J.; Agrawal, D. y Abbadi, A.E.: Database Replication: If You Must be Lazy, be Consistent. Proceedings of the Eighteenth Symposium on Reliable Distributed Systems, IEEE Computer Society, Lausanne, Switzerland (1999) 304-305

[111] Holliday, J.; Agrawal, D. y Abbadi, A.E.: The Performance of Database Replication with Group Multicast. Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing, Digest of Papers: FTCS-29, IEEE Computer Society, Madison, Wisconsin, USA (1999) 158-165

[112] Holliday, J.; Steinke, R.C.; Agrawal, D. y Abbadi, A.E.: Epidemic Algorithms for Replicated Databases. IEEE Trans. Knowl. Data Eng., 15(5) 1218-1238 (2003)

[113] Hotek, M.: Database Replication. SQL Server Magazine. WindowsITPro. Penton Media Inc., Vol. February 2002 (2002)

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

[115] IEEE: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Vol. Std. 1471-2000 (2000)

[116] Iida, S.; Kuwayama, K.; Kanoh, M.; Kato, S. y Itoh, H.: 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, Springer, Cairns, Australia (2004) 272-283

[117] Irún-Briz, L.; Castro-Company, F.; Decker, H. y Muñoz-Escoí, F.D.: An Analytical Design of a Practical Replication Protocol for Distributed Systems. En: M. Núñez; Z. Maamar; F.L. Pelayo; K. Pousttchi y F. Rubio (eds.): Applying Formal Methods: Testing, Performance and M/ECommerce, FORTE 2004 Workshops EMC, EPEW, ITM, Springer, Toledo, Spain (2004) 248-261

[118] Irún-Briz, L.; Decker, H.; Juan-Marin, R.; Castro-Company, F.; Armendáriz-Inigo, J.E.; Bernabéu-Aubán, J.M. y Muñoz-Escoí, F.D.: MADIS: A slim middleware for database replication. Proceedings of the 11st International Conference on Parallel Processing (EuroPar 2005). Lecture Notes in Computer Science, Vol. 3648, Springer (2005) 349-359

[119] Irún-Briz, L.; Muñoz-Escoí, F.D. y Bernabéu-Aubán, J.M.: An Improved Optimistic and Fault-Tolerant Replication Protocol. En: N. Bianchi-Berthouze (ed.): Proceedings of the Third International Workshop on Databases in Networked Information Systems DNIS 2003, Springer, Aizu, Japan (2003) 188-200

[120] Irún-Briz, L.; Muñoz-Escoí, F.D. y Bernabéu-Aubán, J.M.: The Abortion Rate of Lazy Replication Protocols for Distributed Databases. Proceedings of the 6th International Conference on Enterprise Information Systems ICEIS 2004, Porto, Portugal (2004) 130-135

[121] Irún-Briz, L.; Muñoz-Escoí, F.D.; Decker, H. y Bernabéu-Aubán, J.M.: COPLA: A Platform for Eager and Lazy Replication in Networked Databases. Proceedings of the 5th International Conference on Enterprise Information Systems ICEIS 2003, Angers, France (2003) 273-278

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

[123] Jiménez-Peris, R.; Patiño-Martínez, M.; Kemme, B. y Alonso, G.: Improving the Scalability of Fault-Tolerant Database Clusters: Early results. Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), IEEE Computer Society, Vienna, Austria (2002) 477-484

[124] Johansson, J.M.; March, S.T. y Naumann, J.D.: 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, ACM, Brisbane, Australia (2000) 187-196

[125] Karlapalem, K.; Navathe, S.B. y Morsi, M.M.A.: Issues in distribution design of object-oriented databases. En: M.T. Özsu; U. Dayal y P. Valduriez (eds.): Distributed Object Management, Morgan Kaufmann (1994) 148-164

[126] Karlapalem, K. y Pun, N.M.: 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, Springer, Toulouse, France (1997) 347-356

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

[128] Khalil, N.; Eid, D. y Khair, M.: Availability and reliability issues in distributed databases using optimal horizontal fragmentation. En: T.J.M. Bench-Capon; G. Soda y A.M. Tjoa (eds.): Lecture Notes in Computer Science. Database and Expert Systems Applications., Vol. 1677, Springer (1999) 771-780

[129] Kutter, B.L.; Wilson, D.W. y Bardet, J.P.: Metadata and Data Archives for Geotechnical Model Tests. Submitted paper to the International Conference on Physical Modeling in Geotechnics, St. Johns, Newfoundland, Canada (2002) 13

[130] Larsson, A.: Dia, http://www.lysator.liu.se/~alla/dia.

[131] Le Pape, C.; Gançarski, S. y Valduriez, P.: 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, Springer, Catania, Sicily, Italy (2003) 14-15

[132] Le Pape, C.; Gançarski, S. y Valduriez, P.: 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, Vol. I, Springer, Agia Napa, Cyprus (2004) 174-193

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

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

[135] Lim, S.J. y Ng, Y.-K.: Vertical Fragmentation and Allocation in Distributed Deductive Database Systems. Inf. Syst., 22(1) 1-24 (1997)

[136] Lim, S.J. y Ng, Y.-K.: A Hybrid Fragmentation Approach for Distributed Deductive Database Systems. Knowl. Inf. Syst., 3(2) 198-224 (2001)

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

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

[139] Lin, X.; Orlowska, M.E. y Zhang, Y.: A Graph Based Cluster Approach for Vertical Partitioning in Database Design. Data Knowl. Eng., 11(2) 151- (1993)

[140] Lin, X.; Orlowska, M.E. y Zhang, Y.: 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, IEEE Computer Society, Sudbury, Ontario, Canada (1993) 539-544

[141] Lin, Y.; Kemme, B.; Patiño-Martínez, M. y Jiménez-Peris, R.: Middleware based Data Replication providing Snapshot Isolation. Proceedings of the ACM

SIGMOD International Conference on Management of Data., ACM Press, Baltimore, Maryland, USA (2005) 419-430

[142] Liu, A.C. y Chang, S.K.: File allocation in a distributed database. International Journal of Computer Information Sciences, 11(5) (1982)

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

[144] Ma, H. y Schewe, K.-D.: Fragmentation of XML documents. Proceedings of the XVIII Simpósio Brasileiro de Bancos de Dados (SBBD 2003), Manaus, Brazil (2003) 200-214

[145] Ma, H. y Schewe, K.-D.: 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, Faculty of CS and IT, Riga Technical University, Riga, Latvia (2004) 31-46

[146] Ma, H. y Schewe, K.-D.: Query cost analysis for horizontal fragmented complex value databases. Proceedings of the 3rd. Chilenean Database Wokshop, Chile (2004)

[147] Ma, H. y Schewe, K.-D.: Query optimisation as part of distribution design for complex value databases. En: Y. Kiyoki; H. Kangassalo; H. Jaakkola y J. Henno (eds.): Proceedings of the 15th European-Japanese Conference on Information Modeling and Knowledge Bases, Tallinn University of Technology, Estonia (2005) 269-276

[148] Ma, H.; Schewe, K.-D. y Wang, Q.: Distribution design for higher-order data models, Massey University, Department of Information Systems, TR-2005-11 (2005)

[149] Ma, H.; Schewe, K.-D. y Wang, Q.: 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) Vol. 49, ACM International Conference Archive Proceeding Series vol. 170. Australian Computer Society Inc., Hobart, Australia (2006)

[150] Malinowski, E. y Chakravarthy, S.: 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, Vol. 1331, Springer (1997) 347-360

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

[152] Mariano, C. y Morales, E.F.: 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, Springer, Freiburg, Germany (2001) 324-335

[153] Maskarinec, M. y Neumann, K.: 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, Vol. 1, CSREA Press, Las Vegas, Nevada, USA (2003) 97-100

[154] Matthew Lavy, A.M.: Windows Management Instrumentation (WMI), 1st Edition. SAMS (2001)

[155] McCormick, W.T.; Schweitzer, P.J. y White, T.W.: A Problem Decomposition and Data Reorganization by Clustering Techniques. Operation Research, 20(5) 993-1009 (1972)

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

[157] Mei, H. y Sheng, O.: A Semantic Based Methodology for Integrated Computer-Aided Distributed Database Design. The 25th Hawaii International Conference on System Sciences, Vol. 3 (1992) 288- 299

[158] Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H. y Teller, E.: Equation of State Calculations by Fasting Computing Machines. Journal of Chemical Physics, 21(1) (1953)

[159] Microsoft: Visio 2003, http://www.microsoft.com/office/visio/default.asp.

[160] Miklós, Z.: Towards an Access Control Mechanism for Wide-Area Publish/Subscribe Systems. Proceedings of the 22nd. International Conference on Distributed Computing Systems, Workshops (ICDCSW '02) IEEE Computer Society, Vienna, Austria (2002) 516-524

[161] Milán-Franco, J.M.; Jiménez-Peris, R.; Patiño-Martínez, M. y Kemme, B.: Adaptive Middleware for Data Replication. En: H.-A. Jacobsen (ed.): Middleware 2004, ACM/IFIP/USENIX International Middleware Conference, Springer, Toronto, Canada (2004) 175-194

[162] Moody, D.L. y Shanks, G.G.: Improving the quality of data models: empirical validation of a quality management framework. Inf. Syst., 28(6) 619-650 (2003)

[163] Morales, E.F. y Sammut, C.: 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), ACM, Banff, Alberta, Canada (2004)

[164] Moser, L.E.; Amir, Y.; Melliar-Smith, P.M. y Agarwal, D.A.: Extended Virtual Synchrony. Proceedings of the 14th International Conference on Distributed Computing Systems, IEEE Computer Society Press, Poznan, Poland (1994) 56-65

[165] Muñoz-Escoí, F.D.; Irún-Briz, L.; Galdámez, P.; Bernabéu-Aubán, J.M.; Decker, J.; Bataller, M. y Bañuls, H.: Flexible Management of Consistency and Availability of Networked Data Replications. En: T. Andreasen; A. Motro; H. Christiansen y H.L. Larsen (eds.): Proceedings of the 5th International Conference on Flexible Query Answering Systems FQAS 2002, Springer, Copenhagen, Denmark (2002) 289-300

[166] Muro, S.; Ibaraki, T.; Miyajima, H. y Hasegawa, T.: Evaluation of the File Redundancy in Distributed Database Systems. IEEE Trans. Software Eng., 11(2) 199-205 (1985)

[167] Muthuraj, J.; Chakravarthy, S.; Varadarajan, R. y Navathe, S.B.: A Formal Approach to the Vertical Partitioning Problem in Distributed Database Design. Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems (PDIS 1993), Issues, Architectures, and Algorithms, IEEE Computer Society, San Diego, CA, USA (1993) 26-34

[168] Navathe, S.B.; Ceri, S.; Wiederhold, G. y Dou, J.: Vertical Partitioning Algorithms for Database Design. ACM Trans. Database Syst., 9(4) 680-710 (1984)

[169] Navathe, S.B.; Karlapalem, K. y Ra, M.: A mixed fragmentation methodology for initial distributed database design, College of Computing, Georgia Institute of Technology, reprint (1995)

[170] Navathe, S.B. y Ra, M.: Vertical Partitioning for Database Design: A Graphical Algorithm. En: J. Clifford; B.G. Lindsay y D. Maier (eds.): Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, ACM Press, Portland, Oregon, USA (1989) 440-450

[171] Nejdl, W.; Siberski, W.; Simon, B. y Tane, J.: 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, Springer, Sardinia, Italy (2002) 236-249

[172] Nelson, H.J.; Monarchi, D.E. y Nelson, K.M.: Ensuring the 'Goodness' of a Conceptual Representation. Proceedings of the 4th European Conference on Software Measurement and ICT Control (FESMA'01), Germany (2001)

[173] Neumann, K. y Henschen, L.J.: Partitioning Algorithms for a Distributed Deductive Databases. Proceedings of the ACM 22rd Annual Computer Science Conference on Scaling up: Meeting the Challenge of Complexity in Real-World Computing Applications, ACM, Phoenix, Arizona, USA (1994) 288-295

[174] Ng, Y.-K. y Seetharaman, A.: A Query-Based Horizontal Fragmentation Approach for Disjunctive Deductive Databases. En: R. Wagner (ed.): Proceedings of the Eighth International Workshop on Database and Expert Systems Applications DEXA '97, IEEE Computer Society Press, Toulouse, France (1997) 604-609

[175] NISO: Understanding Metadata. National Information Standards Organization (NISO) Press, Bethesda, MD, USA (2004) 17

[176] O, J.; Lee, J.W.; Lee, J. y Zhang, B.-T.: 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, Springer, Pisa, Italy (2004) 298-309

[177] Oaks, P.; ter Hofstede, A.H.M.; Edmond, D. y Spork, M.: Extending Conceptual Models for Web Based Applications. En: S. Il-Yeol; S.W. Liddle; L. Tok Wang y S. Peter (eds.): Proceedings of the 22nd International Conference on Conceptual Modeling on Conceptual Modeling - ER 2003, Springer, Chicago, IL, USA (2003) 216-231

[178] Olivé, A.: Specific Relationship Types in Conceptual Modeling: The Cases of Generic and with Common Participants. Proceedings of the 4st International Conference on Enterprise Information Systems ICEIS 2002, Ciudad Real, Spain (2002) 9

[179] Olivé, A.: On the Role of Conceptual Schemas in Information Systems Development. En: A. Llamosí y A. Strohmeier (eds.): Proceedings of the 9th Ada-Europe International Conference on Reliable Software Technologies, Springer, Palma de Mallorca, Spain (2004) 16-34

[180] Oracle: Oracle9i Application Developer's Guide-Advanced Queuing. Oracle, Redwood Shores, CA, USA (2002)

[181] Özel, S.A.; AltingÖvde, I.S.; Ulusoy, Ö.; Özsoyoglu, G. y Özsoyoglu, Z.M.: Metadata-based modeling of information resources on the Web. Journal of the American Society for Information Science and Technology (JASIST), 55(2) 97-110 (2004)

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

[183] Pacitti, E.; Minet, P. y Simon, E.: 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, Morgan Kaufmann, Edinburgh, Scotland, UK (1999) 126-137

[184] Pacitti, E.; Minet, P. y Simon, E.: 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 (1999) 241-260

[185] Pacitti, E.; Minet, P. y Simon, E.: Replica Consistency in Lazy Master Replicated Databases. Distributed and Parallel Databases, 9(3) 237-267 (2001)

[186] Pacitti, E.; Őzsu, M.T. y Coulon, C.: Preventive Multi-master Replication in a Cluster of Autonomous Databases. Proceedings of the 9th International Conference on Parallel Processing (EuroPar 2003) Springer (2003) 318-327

[187] Pacitti, E.; Őzsu, M.T.; Coulon, C. y Özsu, M.T.: Preventive Replication in a Database Cluster. Distributed and Parallel Databases, 17(3) 36 (2005)

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

[189] Papadimitriou, C.H.: NP-Completeness: A Retrospective. En: P. Degano; R. Gorrieri y A. Marchetti-Spaccamela (eds.): Proceedings of the 24th International Colloquium on Automata, Languages and Programming, ICALP'97, Springer, Bologna, Italy (1997) 2-6

[190] Parankusham, K.K. y Madupu, R.R.: Role of metadata in the datawarehousing environment, Luleå University of Technology, Department of Business Administration and Social Sciences. Division of Information Systems Sciences, Sweden, LTU-PB-EX-0624-SE (2006)

[191] Park, S.-J. y Baik, D.-K.: A data allocation considering data availability in distributed database systems. Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS '97), IEEE Computer Society, Seoul, Korea (1997) 708-713

[192] Patiño-Martínez, M.; Jiménez-Peris, R.; Kemme, B. y Alonso, G.: Scalable Replication in Database Clusters. En: M. Herlihy (ed.): Proceedings of the 14th International Conference on Distribute Computing DISC 2000, Springer, Toledo, Spain (2000) 315-329

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

[194] Paul, T.K. y Iba, H.: 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, Vol. 2, Springer, Chicago, IL, USA (2003) 1259-1270

[195] Pedone, F.; Guerraoui, R. y Schiper, A.: 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), Springer, Southampton, UK (1998) 513-520

[196] Pedone, F.; Wiesmann, M.; Schiper, A.; Kemme, B. y Alonso, G.: Understanding Replication in Databases and Distributed Systems. Proceedings of the 20th International Conference on Distributed Computing Systems IEEE Computer Society, Taipei, Taiwan (2000) 464-474

[197] Pentaris, F. y Ioannidis, Y.: Query Optimization in Distributed Networks of Autonomous Database Systems. ACM Transactions on Database Systems, 31(2) 537-583 (2006)

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

[199] Pérez, J. y Pazos, R.A.: Fragmentación, Ubicación y Reubicación Dinámica de Datos en Bases de Datos Distribuidas. Actas de I Jornadas de Investigación y Docencia en Bases de Datos, Universidad Carlos III de Madrid, Getafe, Madrid, Spain (1997) 111-120

[200] Pérez, J.; Pazos, R.A.; Fraire, H.J.; Cruz, L. y Pecero, J.E.: 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 Springer, Pisa, Italy (2003) 154-164

[201] Pérez, J.; Pazos, R.A.; Frausto-Solis, J.; Reyes, G.; Santaolaya, R.; Fraire, H.J. y Cruz, L.: 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, Vol. 3563, Springer (2005) 33-42

[202] Pérez, J.; Pazos, R.A.; Frausto-Solís, J.; Rodríguez, G.; Cruz, L.; Mora, G. y Fraire, H.J.: 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, Springer, Assisi, Italy (2004) 77-86

[203] Pérez, J.; Pazos, R.A.; Mora, G.; Castilla, G.; Martínez, J.A.; Landero, V.; Fraire, H.J. y González, J.J.: 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 Springer, São Luis, Maranhão, Brazil (2004) 376-384

[204] Pérez, J.; Pazos, R.A.; Romero, D.; Santaolaya-Salgado, R.; Rodríguez, G. y Sosa-Sosa, V.J.: 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., Springer, Montreal, Canada (2003) 134-143

[205] Pérez, J.; Pazos, R.A.; Santaolaya-Salgado, R.; Frausto-Solís, J.; Rodríguez, G.; Cruz, L. y Bravo, M.: 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, Springer, Akademgorodok, Novosibirsk, Rusia (2003) 539-545

[206] Pérez, J.; Pazos, R.A.; Solís, J.F.; Rodríguez, G.; Cruz, L. y Fraire, H.J.: 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, Vol. III, Springer, Assisi, Italy (2004) 415-424

[207] Pérez, J.; Pazos, R.A.; Solís, J.F.; Romero, D. y Cruz, L.: 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.): Proceedings of the 1st Mexican International Conference on Artificial Intelligence MICAI 2000, Springer, Acapulco, Mexico (2000) 75-81

[208] Pérez, J.; Pazos, R.A.; Velez, L. y Rodríguez, G.: 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.): Proceedings of the 2nd Mexican International Conference on Artificial Intelligence MICAI 2002, Springer, Merida, Yucatan, Mexico (2002) 118-127

[209] Pettinger, J.E. y Everson, R.M.: 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.): Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, Morgan Kaufmann, New York, USA (2002) 692

[210] Pierson, J.-M.; Seitz, L.; Duque, H. y Montagnat, J.: MetaData for Efficient, Secure and Extensible Access to Data in a Medical Grid. Proceedings of the 15th International Workshop on Database and Expert Systems Applications DEXA 2004, IEEE Computer Society, Zaragoza, Spain (2004) 562-566

[211] Plattner, C. y Alonso, G.: Ganymed: Scalable Replication for Transactional Web Applications. En: H.-A. Jacobsen (ed.): Proceedings of the 5th. International Middleware Conference ACM/IFIP/USENIX 2004, Springer, Toronto, Canada (2004) 155-174

[212] Poels, G.: Conceptual Modeling of Accounting Information Systems: A Comparative Study of REA and ER Diagrams. En: M.A. Jeusfeld y O. Pastor (eds.): Proceedings of the ER 2003 ECOMO, IWCMQ, AOIS, and XSDM Workshops, Springer, Chicago, IL, USA (2003) 152-164

[213] Poels, G.; Nelson, J.; Genero, M. y Piattini, M.: Quality in Conceptual Modeling - New Research Directions. En: S. Spaccapietra; S.T. March y Y. Kambayashi (eds.): Proceedings of the 21st International Conference on Conceptual Modeling ER 2002, Springer, Tampere, Finland (2002) 243-250

[214] Pramanik, S.; Kao, D.T. y Vineyard, D.: Fragmentation of Recursive Relations in Distributed Databases. En: A. Pirotte; C. Delobel y G. Gottlob (eds.): Proceedings of the 3rd International Conference on Extending Database Technology EDBT'92, Springer, Vienna, Austria (1992) 389-404

[215] Raghuram, A.; Morgan, T.W. y Julien-Laferriere, P.: 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 (1989) 461-472

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

[217] Rammer, I.: Advance .Net Remoting. C# Edition. Apress™ (2002)

[218] Richter, J.: Applied Microsoft .NET Framework Programming. Microsoft Press (2002)

[219] Rodríguez, A.; González, L.M. y Águila, L.S.: Asignación de fragmentos en Bases de Datos Distribuidas mediante la aplicación de Algoritmos Genéticos. Boletín de la Sociedad Cubana de Matemática y Computación, 3(1) (2005)

[220] Rodríguez, A.; González, L.M.; Cabrera, L. y Morell, A.: 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, Cámara Chilena del Libro A.G., Universidad de Atacama, Copiapó, Chile (2002) 49-58

[221] Rodríguez, A.; Rosa, D.; Mainegra, M. y González, L.M.: An Intelligent Agent Using a Q-Learning Method to Allocate Replicated Data in a Distributed Database. Proceedings of the 6th Mexican International Conference on Artificial Intelligence MICAI2007. IEEE Poster Session. IEEE Computer Society, Aguascalientes, México (2007) 10 p.

[222] Rodríguez, A.; Rosa, D.; Mainegra, M. y González, L.M.: A Reinforcement Learning Solution for Allocating Replicated Fragments in a Distributed Database. Revista Computación y Sistemas, aceptado con revisión (2007)

[223] Röhm, U.; Böhm, K.; Schek, H.-J. y Schuldt, H.: FAS - A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components. Proceedings of the Thirtieth International Conference on VLDB 2002. Morgan Kaufmann Publishers, Hong Kong, China (2002) 754-765

[224] Roosta, S.H.: A new model for distributed database systems. International Journal of Computer Mathematics, 82(12) 1447-1454 (2005)

[225] Saccà, D. y Wiederhold, G.: Database Partitioning in a Cluster of Processors. ACM Trans. Database Syst., 10(1) 29-56 (1985)

[226] Salter, A.: Semantic Modelling and a Semantic Normal Form, Staffordshire University. Scool of Computing, SOCTR/01/01 (2001)

[227] Savonnet, M. y Terrasse, M.-N.: 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, Springer, Munich, Germany (2001) 339-348

[228] Savonnet, M.; Terrasse, M.-N. y Yétongnon, K.: 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), IEEE Computer Society, Hsinchu, Taiwan (1999) 283-290

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

[230] Schewe, K.-D.: On the unification of query algebras and their extension to rational tree structures. En: M.E. Orlowska y J. Roddick (eds.): Proceedings of the Australasian Database Conference (ADC2001), Australian Computer Society, Inc. (2001)

[231] Schewe, K.-D.: Fragmentation of object oriented and semi-structured data. En: H.-M. Haav y A. Kalja (eds.): Databases and Information Systems II, Kluwer Academic Publishers (2002) 1-14

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

[233] Shepherd, J.; Harangsri, B.; Chen, H.L. y Ngu, A.H.H.: 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), World Scientific, Singapore (1995) 380-387

[234] Shin, D.-G. y Irani, K.B.: Fragmenting Relations Horizontally Using a Knowledge-Based Approach. IEEE Trans. Software Eng., 17(9) 872-883 (1991)

[235] Silberschatz, A.; Korth, H.F. y Sudarshan, S.: Database Systems Concepts, 5th edition. Mc-Graw-Hill (2006)

[236] Silva, A.R.; Sousa, P.A. y Marques, J.: Development of Distributed Applications with Separation of Concerns. Proceedings of the 2nd Asia-Pacific Software Engineering Conference (APSEC'95), IEEE CS Press, Brisbane, Queensland, Australia (1995)

[237] SmartDraw: SmartDraw Suite Edition, http://smartdraw.com.

[238] Snoeck, M. y Dedene, G.: Existence Dependency: The Key to Semantic Integrity between Structural and Behavioral Aspects of Object Types. IEEE Trans. Software Eng., 24(4) 233-251 (1998)

[239] Snoeck, M.; Michiels, C. y Dedene, G.: Consistency by Construction: The Case of MERODE. En: M.A. Jeusfeld y O. Pastor (eds.): Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003), Springer, Chicago, IL, USA (2003) 105-117

[240] Son, J.H. y Kim, M.-H.: An adaptable vertical partitioning method in distributed systems. Journal of Systems and Software, Elsevier, 73(3) 551-561 (2004)

[241] Stevens, P.; Whittle, J. y Booch, G. (eds.): Proceedings of the 6th International Conference on the Unified Modeling Language UML 2003, Vol. 2863. Springer, San Francisco, CA, USA (2003)

[242] Sun, W.; Shu, J. y Zheng, W.: 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, Vol. II, Springer, Dalian, China (2004) 719-724

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

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

[245] Teorey, T.J.: Distributed Database Design: A Practical Approach and Example. SIGMOD Record, 18(4) 23-39 (1989)

[246] Teorey, T.J.; Yang, D. y Fry, J.P.: A logical design methodology for relational databases using the Extended Entity-Relationship Model. ACM Comput. Surv., 18(2) 23-39 (1986)

[247] theKompany: Data Architect, http://www.thekompany.com.

[248] Tsai, H.-P.; Huang, J.-L.; Chao, C.-M.; Chen, M.-S. y Liao, C.: SIFA: A Scalable File System with Intelligent File Allocation. Proceedings of the 26th International Computer Software and Applications Conference (COMPSAC 2002), IEEE Computer Society, Oxford, England (2002) 793-798

[249] Ulusoy, Ö.: Data Replication and Availability. En: K.-y. Lam y T.-W. Kuo (eds.): Real-Time Database Systems: Architecture and Techniques., Kluwer Academic Publishers (2001) 217-225

[250] Ulusoy, Ö.: Distributed Concurrency Control. En: K.-y. Lam y T.-W. Kuo (eds.): Real-Time Database Systems: Architecture and Techniques, Kluwer Academic Publishers (2001) 205-215

[251] Valduriez, P.; Pacitti, E. y Coulon, C.: Large-scale Experimentation with Preventive Replication in a Database Cluster DDIR 2005, INRIA, France (2005) 6

[252] van Bommel, P.: Database design by computer-aided schema transformations. Software Engineering Journal, 10(4) 125-132 (1995)

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

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

[255] Vitter, D. y Templeman, J.: Visual Studio .NET: The .NET Framework Black Book. Coriolis Group (2002)

[256] Watkins, C.: Learning from Delayed Rewards. Ph.D. Thesis. University of Cambridge, King's College, Cambridge, UK (1999)

[257] Wiesmann, M.; Schiper, A.; Pedone, F.; Kemme, B. y Alonso, G.: Database Replication Techniques: A Three Parameter Classification. Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00), IEEE Computer Society, Nürnberg, Germany (2000) 206-215

[258] Wolfson, O. y Jajodia, S.: 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, IEEE Computer Society Press, Monterey, CA, USA (1992) 62-65

[259] Wolfson, O. y Jajodia, S.: Distributed Algorithms for Dynamic Replication of Data. Proceedings of the 11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM Press, San Diego, USA (1992) 149-163

[260] Wolfson, O. y Jajodia, S.: An Algorithm for Dynamic Data Allocation in Distributed Systems. Inf. Process. Lett., 53(2) 113-119 (1995)

[261] Wolfson, O.; Jajodia, S. y Huang, Y.: An Adaptive Data Replication Algorithm. ACM Trans. Database Syst., 22(2) 255-314 (1997)

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

[263] XTG-Systems: XTG Data Modeller, http://www.xtgsystems.com

[264] Yoshida, M.; Mizumachi, K.; Wakino, A.; Oyake, I. y Matsushita, Y.: Time and Cost Evaluation Schemes of Multiple Copies of Data in Distributed Database Systems. IEEE Trans. Software Eng., 11(9) 954-959 (1985)

[265] Zhou, X.; Zhang, Y. y Orlowska, M.E.: A new Fragmentation Scheme for Recursive Query Processing. Data Knowl. Eng., 13(2) 177-192 (1994)

PRODUCCIÓN CIENTÍFICA DEL AUTOR SOBRE EL TEMA DE TESIS

[1] Rodríguez, A.; González, L.M.: Herramientas de ayuda al diseño de bases de datos distribuidas. Informática ’98, 6to. Congreso de la Sociedad Cubana de Matemática y Computación, Holguin, Cuba (1998)

[2] Rodríguez, A.; Sierra, B.; Morell, A.; Arango, J.L. y González, L.M.: Arquitectura de un sistema de ayuda para el diseño de bases de datos distribuidas. COMPUMAT ’98, 6to. Congreso Internacional de Nuevas Tecnologías y Aplicaciones Informáticas, La Habana, Cuba (1998)

[3] Rodríguez, A.; Zamora, I. y González, L.M.: Diseño de distribución de datos para el sistema de control de la actividad de ICT en la UCLV. CIMAC ’98, I Conferencia Internacional de Matemática Aplicada y Computación, Pinar del Río, Cuba (1998)

[4] Rodríguez, A.; Morell, A. y González, L.M.: Ayuda al proceso de fragmentacón vertical en bases de datos distribuidas. CIMAC ’98, I Conferencia Internacional de Matemática Aplicada y Computación, Pinar del Río, Cuba (1998)

[5] Rodríguez, A. y González, L.M.: Un enfoque integrado para el diseño de bases de datos distribuidas. Universidad Central de Las Villas, Reconocimiento: Logro Científico Universitario (1999)

[6] Zamora, I.; González, L.M. y Rodríguez, A.: Distribución de datos en sistemas de bases de datos distribuidas. GIGA, Año 1999 (1):28-35, ISSN: 1028-270x (1999)

[7] Morell, A.; González, L.M. y Rodríguez, A.: Algoritmos para la fragmentación vertical en bases de datos distribuidas. COMPUMAT 2000. 7mo. Congreso de la Sociedad Cubana de Matemática y Computación, Manzanillo, Cuba (2000)

[8] Morell, A.; González, L.M. y Rodríguez, A.: Un enfoque a la fragmentación vertical en bases de datos distribuidas. Congreso Internacional Informática 2000, Nuevas Tecnologías Informáticas., La Habana, Cuba, ISBN: 959-7160-01-3, (2000)

[9] Rodríguez, A. y González, L.M: Hacia una herramienta CASE integrada para el diseño de bases de datos distribuidas. Primer Encuentro Internacional de Computación Aplicada, Ocotlán, Jal., México (2000)

[10] Rodríguez, A.; Valdés, A. y González, L.M: Asistente para la caracterización de una red de computadoras en el proceso de diseño de bases de datos distribuidas.

Primer Encuentro Internacional de Computación Aplicada, Ocotlán, Jal., México (2000)

[11] Rodríguez, A.; González, L.M.; Cabrera, L. y Morell, A.: 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, Cámara Chilena del Libro A.G., Universidad de Atacama, Copiapó, Chile, ISBN: 956-291-506-9 (2002)

[12] Rodríguez, A.; González, L.M.; Morell, A.; Cabrera, L.; Artiles, M.; Águila, L.S. y Valdés, Á.: Integración de herramientas de ayuda al diseño de bases de datos distribuidas. Actas I Workshop de Bases de Datos, Jornadas Chilenas de Computación JCC 2002, Cámara Chilena del Libro A.G., Universidad de Atacama, Copiapó, Chile, ISBN: 956-291-506-9 (2002)

[13] García, C.E.; Rodríguez, A.; González, L.M. y Álvarez, W.A.: ERECASE, una herramienta con validación de diagramas entidad relación. 6to. Simposium Iberoamericano de Computación e Informática SICI 2005, Instituto Tecnológico de Nuevo León, Instituto Tecnológico de Nuevo León, Monterrey, NL, México, ISBN: 968-5823-21-9, (2005)

[14] Rodríguez, A.; González, L.M. y Águila, L.S.: Asignación de fragmentos en bases de datos distribuidas mediante la aplicación de algoritmos genéticos. COMPUMAT 2005, IX Congreso Nacional de Matemática y Computación. Boletín de la Sociedad Cubana de Matemática y Computación, ISSN: 17286042, 3(1) (2005)

[15] García, C.E.; Rodríguez, A. y González, L.M.: Un estudio de la validación estructural de diagramas Entidad-Relación Extendido. II Conferencia Científica de la Universidad de las Ciencias Informáticas UCIENCIA 2006, II Taller de Tecnologías Avanzadas de Bases de Datos, Universidad de las Ciencias Informáticas, La Habana, Cuba (2006)

[16] Rodríguez, A.; Cárdenas, A.; Castro, M.T. y González, L.M.: Caracterización de la red de comunicación y sitios en el diseño de bases de datos distribuidas. Convención de Ingeniería Eléctrica CIE’2007 Universidad Central "Marta Abreu" de Las Villas (2007)

[17] Rodríguez, A.; Cabrera, N.E. y González, L.M.: Fragmenter: un asistente para la fragmentación en bases de datos distribuidas. III Conferencia Científica de la

Universidad de las Ciencias Informáticas UCIENCIA 2007, III Taller sobre Nuevas Tecnologías de Bases de Datos y Programación Avanzada, ISBN: 978-959-286-005-6 (2007)

[18] Rodríguez, A.; Rosa, D.; Mainegra, M. y González, L.M.: Un agente inteligente de aprendizaje reforzado aplicado al problema de asignación en bases de datos distribuidas. COMPUMAT 2007, X Congreso Nacional de Matemática y Computación, Holguin, Cuba (2007)

[19] Rodríguez, A.; Cárdenas, A.; Castro, M.T. y González, L.M.: Caracterización de la red de comunicación y sitios en el diseño de bases de datos distribuidas. Revista Ingeniería Electrónica, Automática y Comunicaciones, ISSN: 0256-5944, (2007) [En proceso]

[20] Rodríguez, A.; González, L.M.; García, C.E.; Álvarez, W.A.; Cárdenas, A.; Cabrera, N.E. y Rosa, D.: SIADBDD: Sistema integrado de ayudas al diseño de bases de datos distribuidas. XVI Forum de Ciencia y Técnica, I Etapa, Premiación: Relevante Especial (2007)

[21] Rodríguez, A.; Rosa, D.; Mainegra, M. y González, L.M.: An Intelligent Agent Using a Q-Learning Method to Allocate Replicated Data in a Distributed Database. Proceedings of the 6th Mexican International Conference on Artificial Intelligence-MICAI2007. IEEE Poster Session. IEEE Computer Society, Aguascalientes, México (2007) 10 p.

[22] Rodríguez, A.; Rosa, D.; Mainegra, M. y González, L.M.: A Reinforcement Learning Solution for Allocating Replicated Fragments in a Distributed Database. Revista Computación y Sistemas, México [Aceptado para publicarse en vol. 11, no. 2] (2007)

[23] Rodríguez, A.; Rosa, D.; Pérez, W. y González, L.M.: Manejo distribuido de datos para facilitar el control de calidad en la producción de azúcar crudo de caña. Revista Centro Azúcar, Cuba [Aceptado para publicarse en año 34, no. 3, Julio-Septiembre de 2007] (2007)

Monografías

[1] Rodríguez, A.; González, L.M.; García, C.E.; Castro, M.T.: Bases de Datos en ambiente distribuido. Editorial Feijóo. ISBN: 978-959-250-365-6. Santa Clara, Cuba (2007)

[2] Rodríguez, A.; González, L.M.; García, C.E.; Castro, M.T.: Manejo de transacciones centralizadas y distribuidas. Editorial Feijóo. ISBN: 978-959-250-364-9. Santa Clara, Cuba (2007)

Patentes y registros de propiedad intelectual

[1] ERECASE: Una herramienta para la modelación de esquemas conceptuales globales. Autores: Abel Rodríguez, Luisa M. González, Leonel Cabrera, Alberto Morell. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 010923-10923 (2002).

[2] NetWizard: Asistente para la caracterización de sitios de procesamiento. Autores: Abel Rodríguez, Luisa M. González, Ángel Valdés, Alberto Morell. Número de Registro del Centro Nacional de Derecho de Autor: 010919-10919 (2002).

[3] AppWizard: Un asistente para la caracterización de aplicaciones en el proceso de diseño de Bases de Datos Distribuidas. Autores: Abel Rodríguez, Luisa M. González, Leonel Cabrera, Michel Artiles. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 1430-2005 (2005).

[4] SIADBDD: Sistema Integrado de Ayudas al Diseño de Bases de Datos Distribuidas. Autores: Abel Rodríguez, Luisa M. González, Carlos E. García, William A. Álvarez, Norma E. Cabrera, Darien Rosa, Marisela Mainegra, Alain Cárdenas. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 2143-2007 (2007).

[5] ERECASE 2.0: Una herramienta para la modelación de esquemas conceptuales globales con validaciones. Autores: Abel Rodríguez, Luisa M. González, Carlos E. García, William A. Álvarez. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 2139-2007 (2007).

[6] NetWizard 2.0: Asistente para la caracterización de sitios de procesamiento en el Diseño de Bases de Datos Distribuidas. Autores: Abel Rodríguez, Luisa M. González, Carlos E. García, Alain Cárdenas. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 2134-2007 (2007).

[7] AppWizard 2.0: Asistente para la caracterización de aplicaciones en el diseño de de bases de datos distribuidas. Autores: Abel Rodríguez, Luisa M. González, Carlos E. García, Norma E. Cabrera, Darien Rosa. Entidad: Universidad Central

de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 2140-2007 (2007).

[8] Fragmenter: Asistente para la fragmentación de bases de datos distribuidas. Autores: Abel Rodríguez, Luisa M. González, Carlos E. García, Norma E. Cabrera, Darien Rosa. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 2138-2007 (2007).

[9] Allocator: Asistente para el diseño de asignación de bases de datos distribuidas. Autores: Abel Rodríguez, Luisa M. González, Carlos E. García, Darien Rosa. Entidad: Universidad Central de Las Villas. Número de Registro del Centro Nacional de Derecho de Autor: 2135-2007 (2007)

ANEXOS

Anexo 1. Esquemas del catálogo

ACCESSTIME(FromStation, ToStation, AverageTime)

APP_ATTRIBUTES(IdApp, SchemataName, IdAtt, OpRead, OpWrite)

APP_SCHEMATA(IdApp, SchemataName, Selectivity)

APP_SITE(IdApp, SiteName, ActivationFreq)

APPLICATIONS(IdApp, AppName, OpRead, OpWrite, MaxAnswerTime, ExecuteTime)

DATATYPES(Idtype, Type)

FRAGMENTATTRIBUTES(IdFragmAtt, IdLogFragm, IdGlobalAtt, SchemaName, IdDataType, FragAttName)

GLOBALATTRIBUTES(SchemaName, IdAtt, AttName, IdType, Pkey, Fkey, AttSize)

LINKS(SchemaMemberName, IdAtt, SchemaOwnerName)

LOGICALFRAGMENTS(IdLogFragm, IdOperation, FragmentName, Cardinality)

OPERATIONS(IdOperation, SchemaName, IdPreviousOp, IdOpType, Expression)

OPERATIONTYPE(IdOpType, OperationType)

OPERATORS(IdOperator, Operator)

PHYSICALFRAGMENT(IdPhysicalFragment, SiteName, PhysicalFragmName, IdLogFragm)

PREDICATES(IdApp, SchemaName, IdAtt, IdOperator, AttValue)

SCHEMATA(SchemataName, Cardinality)

S**ITES**(SiteName, IP, ProccessorType, ProccessorClock, TotalDiskSpace, FreeDiskSpace, ReadAccessTime, WriteAccessTime, MaxAnswerTime)

Anexo 2. Vistas de ERECASE

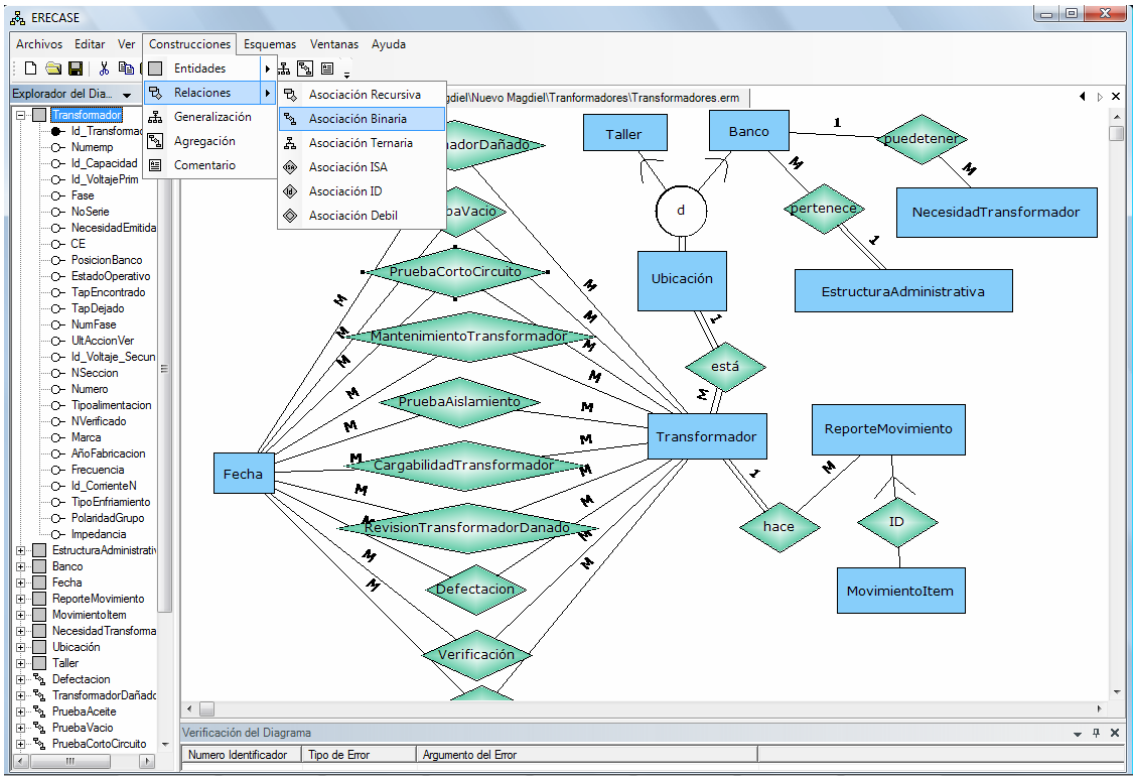

Figura A2.1. Vista de ERECASE durante la caracterización del ECG para el control de transformadores.

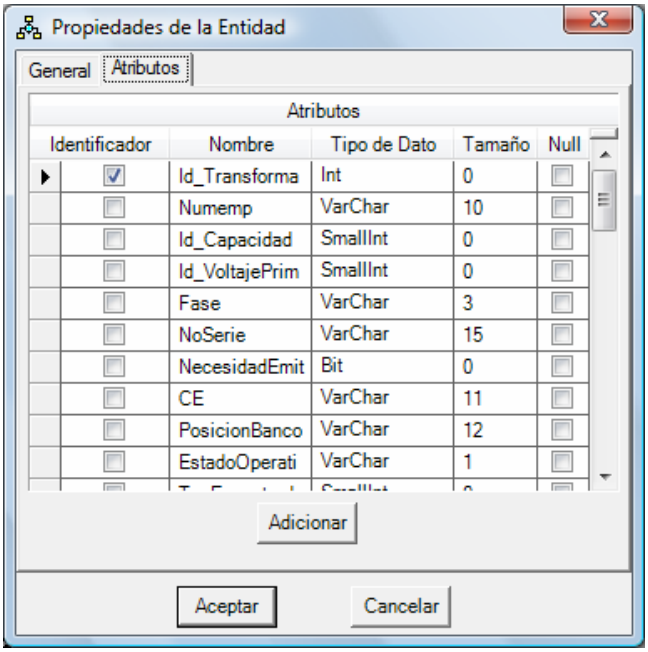

Figura A2.2. Edición de propiedades de las entidades en ERECASE.

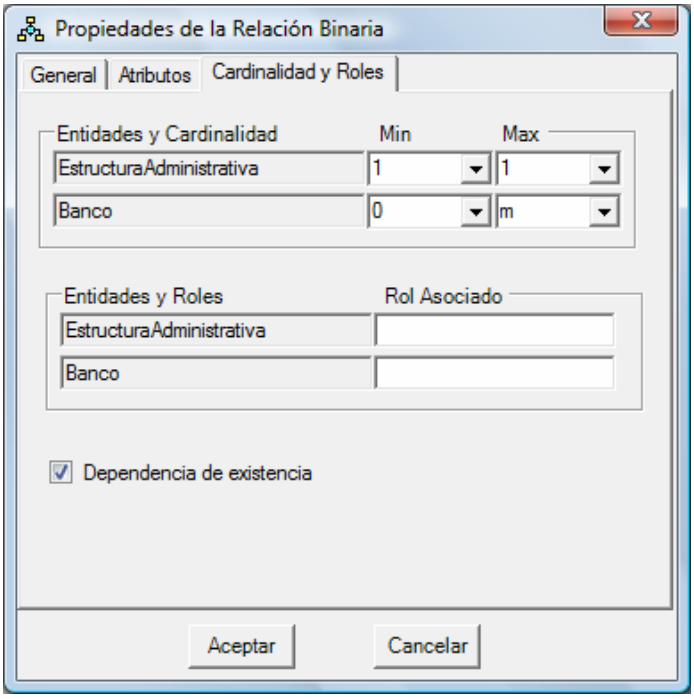

Figura A2.3. Edición de propiedades de las interrelaciones en ERECASE.

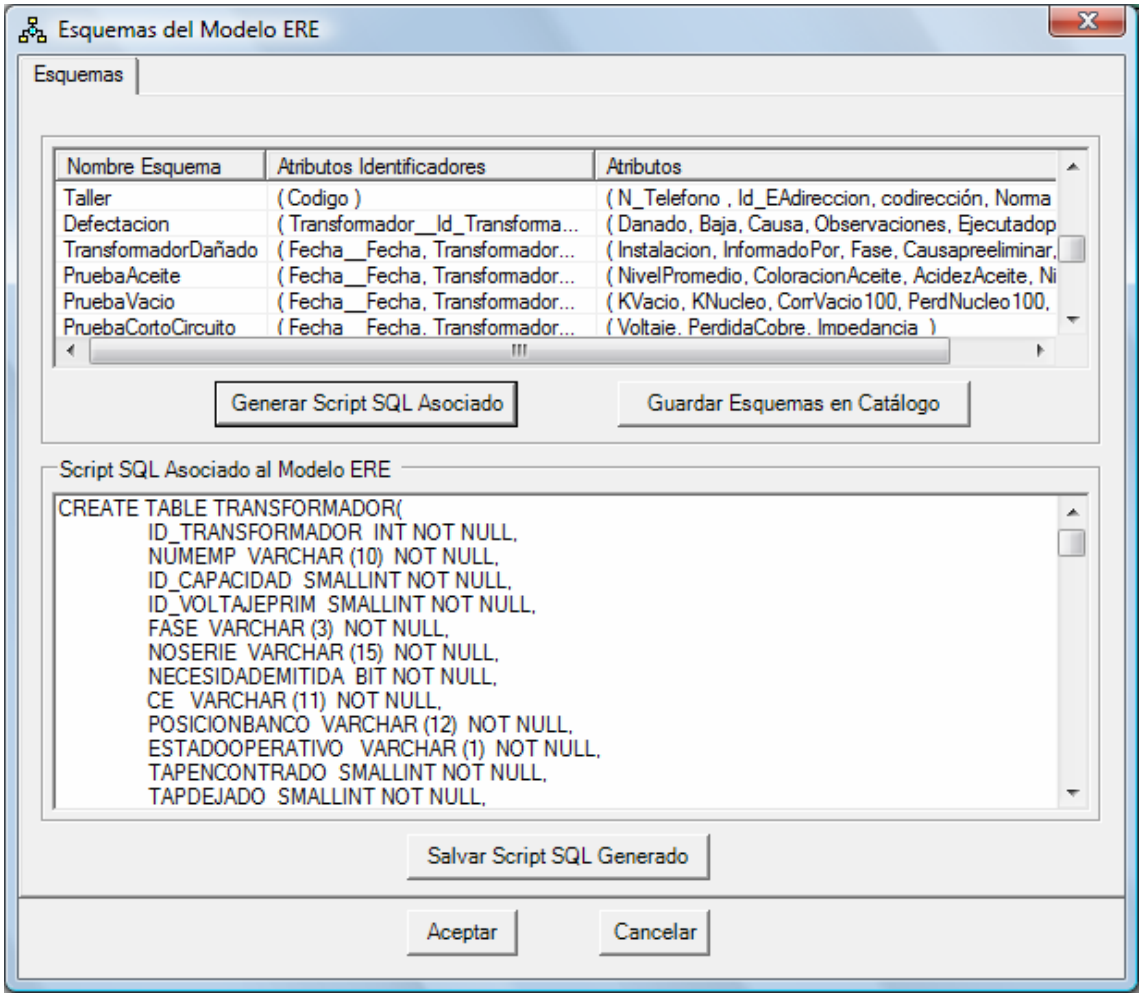

Figura A2.4. Generación de esquemas en ERECASE.

Anexo 3. Esquemas relacionales para el control de transformadores

TRANSFORMADOR (Id_EAdministrativa, Id_Transformador, Código, Numemp, Id_Capacidad, Id_VoltajePrim, Fase, NoSerie, NecesidadEmitida, CE, PosiciónBanco, EstadoOperativo, TapEncontrado, TapDejado, NumFase, UltAcciónVer, Id_Voltaje_Secun, NSección, Número, Tipoalimentación, NVerificado, Marca, AñoFabricación, Frecuencia, Id_CorrienteN, TipoEnfriamiento, PolaridadGrupo, Impedancia)

DEFECTACIÓN(Id_Transformador, Id_EAEquipo, Fecha, Dañado, Baja, Causa, Observaciones, Ejecutadopor, OParme, OPenrollado, OPconexiones, OPemsamblado, OPaceite, OPhermeticidad, OPpruebas, Taller)

TRANSFORMADORDAÑADO(Id_Transformador, Id_EAEquipo, Fecha, Instalación, InformadoPor, Fecha, Fase, Causapreeliminar, Tipodaño, CausaDefinitiva, Observaciones)

PRUEBAACEITE(Id_Transformador, Id_EAEquipo, Fecha, NivelPromedio, ColoraciónAceite, AcidezAceite, NivelAceite)

PRUEBAVACÍO(Id_Transformador, Id_EAEquipo, Fecha, KVacío, KNúcleo, CorrVacío100, CorrVacío110, PerdNúcleo100, PerdNúcleo110)

PRUEBACORTOCIRCUITO(Id_Transformador, Id_EAEquipo, Fecha, Voltaje, PérdidaCobre, Impedancia)

PRUEBAAISLAMIENTO(Id_Transformador, Id_EAEquipo, Fecha, Temperatura, BajaAltaTierra1, AltaBajaTierra1, BajaAltaTierra2, AltaBajaTierra2, VoltMegger)

VERIFICACIÓN(Id_Transformador, Id_EAEquipo, FechaEjecución, TipoTrabajo, CartaTécnica, Hermeticidad, Polaridad, PAltoVoltaje, Aceptado, EjecutadaPor, Observaciones, OrdenTrabajo, AltoVoltajeFrecuenciaIndustrial, PruebaAltoVoltaje, Folio, Name , año, Id_FabricanteActual, UltAcciónVer, Id_FabricanteAnterior, Impedancia, Peso, PesoAislante, Código, Temperatura)

MANTENIMIENTOTRANSFORMADOR(Id_Transformador, Id_EAEquipo, FechaEjecución, Fase, EstadoPinturaTanque, EstadoPinturaRótulos, Hermeticidad)

REVISIÓNTRANSFORMADORDAÑADO(Id_Transformador, Id_EAEquipo, FechaEjecución, Código, TipoDaño, BushingPPartido, BushingPFlojo, BushingPFCont, BushingSPartido, BushinfSFlojo, BushingSFCont, Salidero, TanquePerforado, BajantesSMalos, SobrecargaEvidente, CausaDaño, VientosFuertes, Lluvias,

TormentasEléctricas, AccidenteTránsito, Derrumbe, contaminación, InstalaciónReciente, Observaciones, EjecutadaPor, TipoDaño, OT, Folio, año, CorrienteNominal, CorrienteOperación, EstadoDesconectivo, DesconectivoCliente, EstadoCliente, Dilatado, Enrredado, Árboles, AcometidaCC, Tendedera, CCInterno, CantidadConsumidores, DistUltimoConsumidor)

CARGABILIDADTRANSFORMADOR(Id_Transformador, Id_EAEquipo, FechaEjecución, KVAMedido, CodTomaCarga)

MOVIMIENTOITEM (Id_Movimiento, Id_Transformador, Id_EAEquipo, CausaMov, Desde, Hacia, FaseConectada)

REPORTEMOVIMIENTO(Id_Movimiento , FechaEjecución, Descripción, NoMovMedioBásico, H19, EjecutadaPor, Observaciones, OrdenTrabajo, Folio, año)

NECESIDADTRANSFORMADOR(Id_Necesidades, Código, Id_Capacidad, NumFases, Id_VPrimario, Id_VSecundario, CausaNecesidad, Prioridad, Fecha, Estado, Confirmada, AcciónAutomática, TipoAfectación, TipoServicio)

BANCO(Código, CódigoAntiguo, Id_EAdirección, codirección , Seccionalizador, Circuito, Conexión, Tipoalimentación, NSección, Tipoterreno, Id_VoltajeSalida, TipoSalida, Id_VoltajePrimario, Id_EAdministrativa, EstadoOperativo, Id_TipoCarga, NumClientes)

ESTRUCTURAADMINISTRATIVA(Id_EAdministrativa, Nombre, Jefe, Código, Tipo, Subordinada, N_Teléfono, N_Fax, E_Mail, Km_Línea, N_Consumidores, KVAInstalados, Calculado, Centro_de_costo, Id_EAdirección, codirección)

PERTENECE(ID__PERTENECE, Id_EAdministrativa, Código)

TALLER(Código, N_Teléfono, Id_EAdirección, codirección, Norma)

Anexo 4. Vistas de NETWIZARD

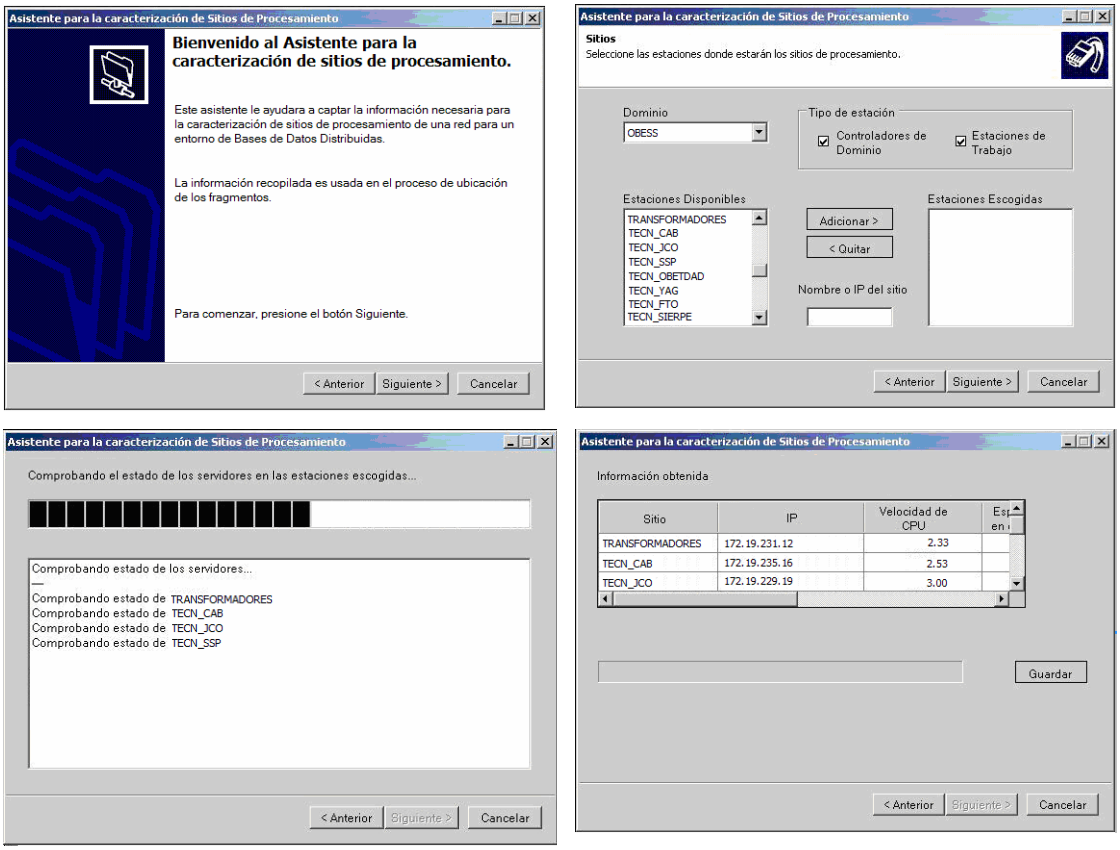

Figura A4.1. Vistas de NETWIZARD caracterizando los sitios de procesamiento.

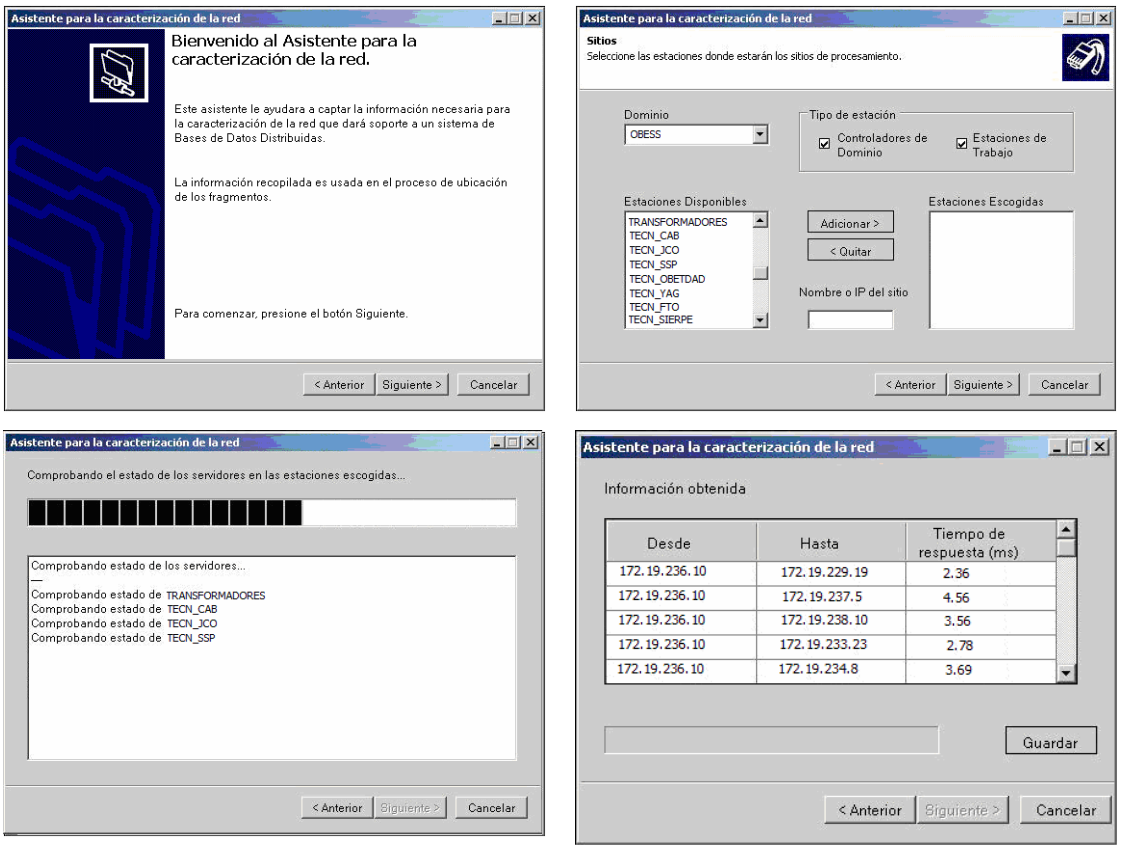

Figura A4.2. Vistas de NETWIZARD caracterizando la red de comunicación.

Anexo 5. Vistas de APPWIZARD

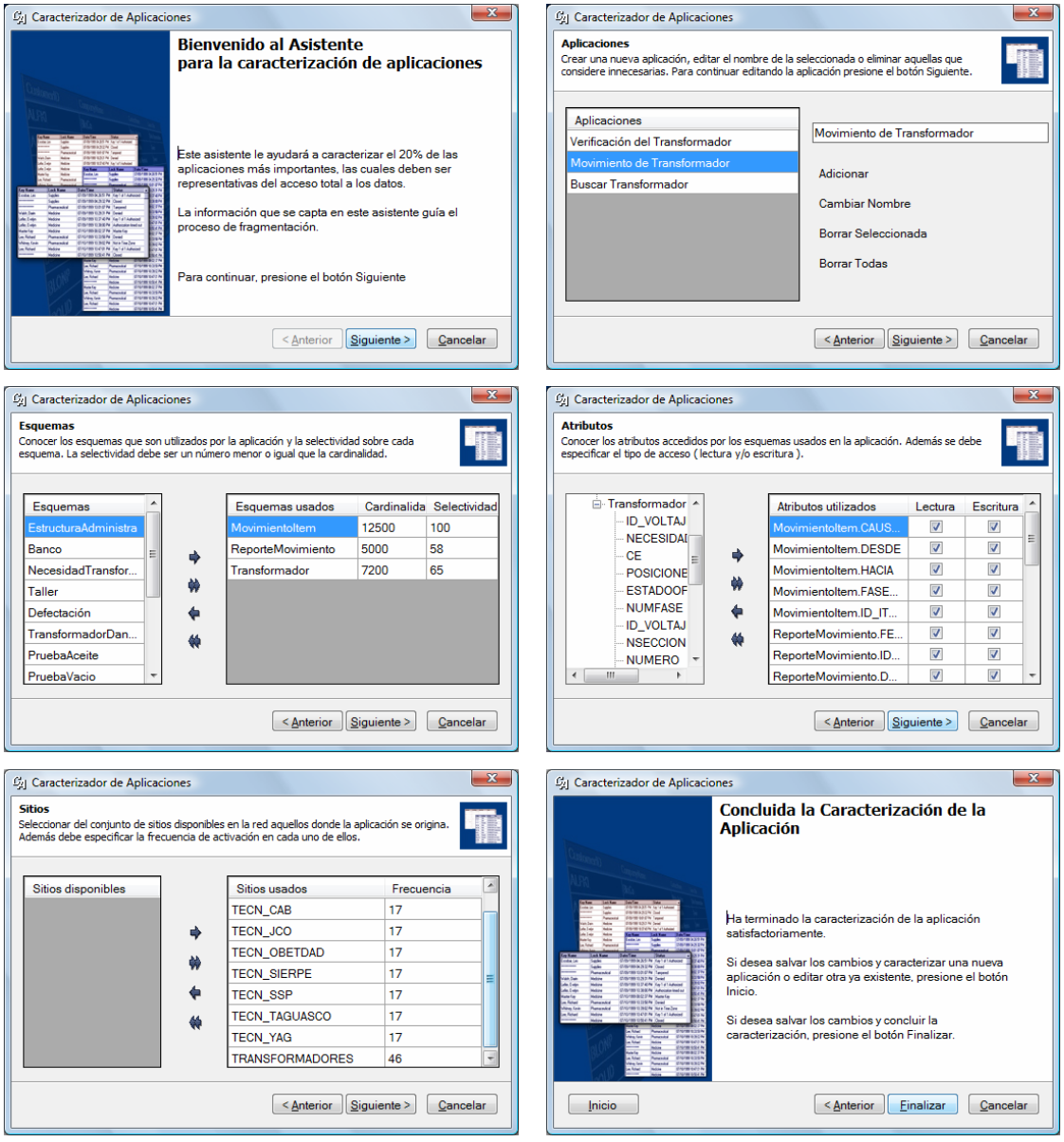

Figura A5.1. Vistas de APPWIZARD caracterizando la aplicación “Movimiento de transformador”.

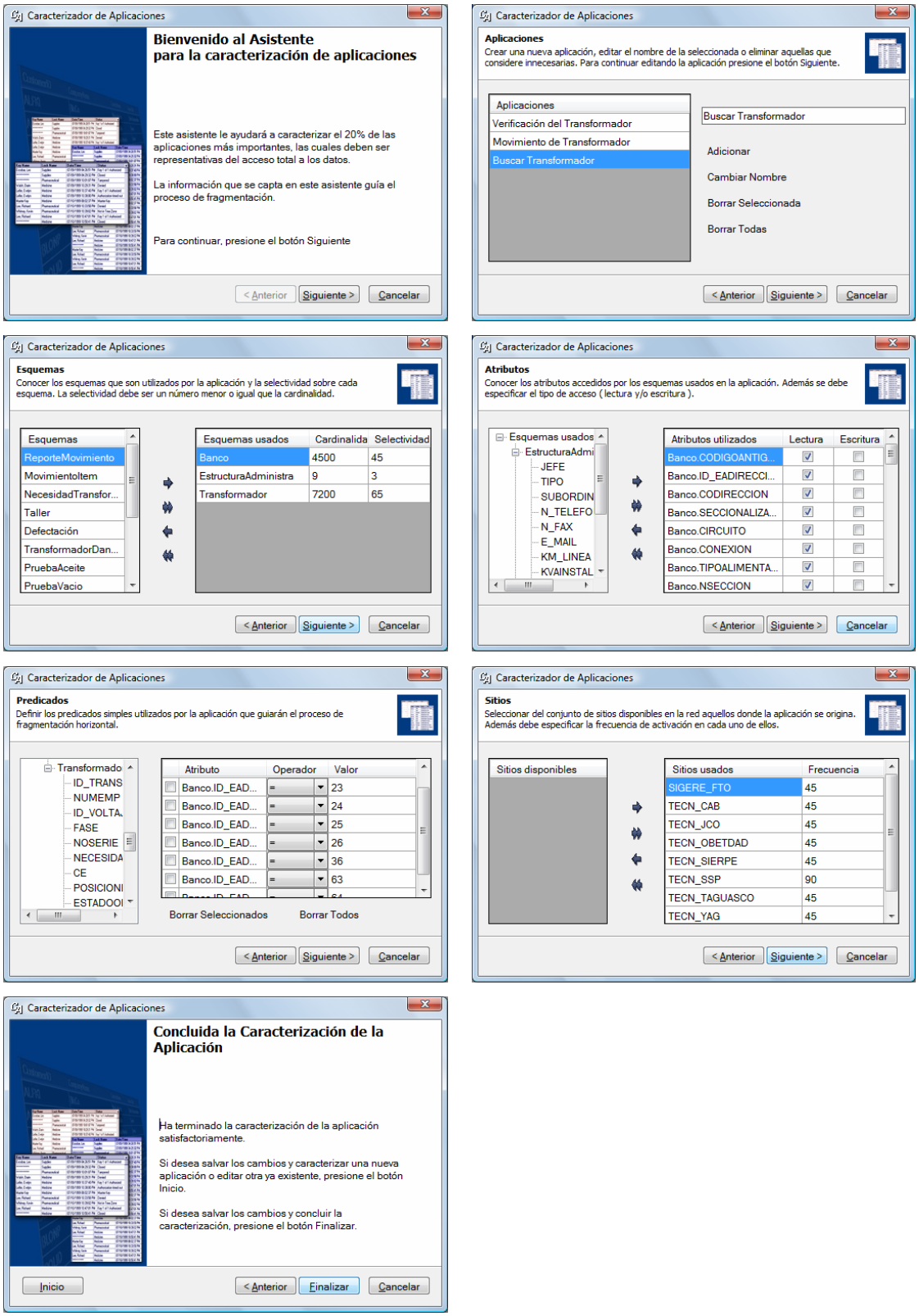

Figura A5.2. Vistas de APPWIZARD caracterizando la aplicación “Buscar transformador”.

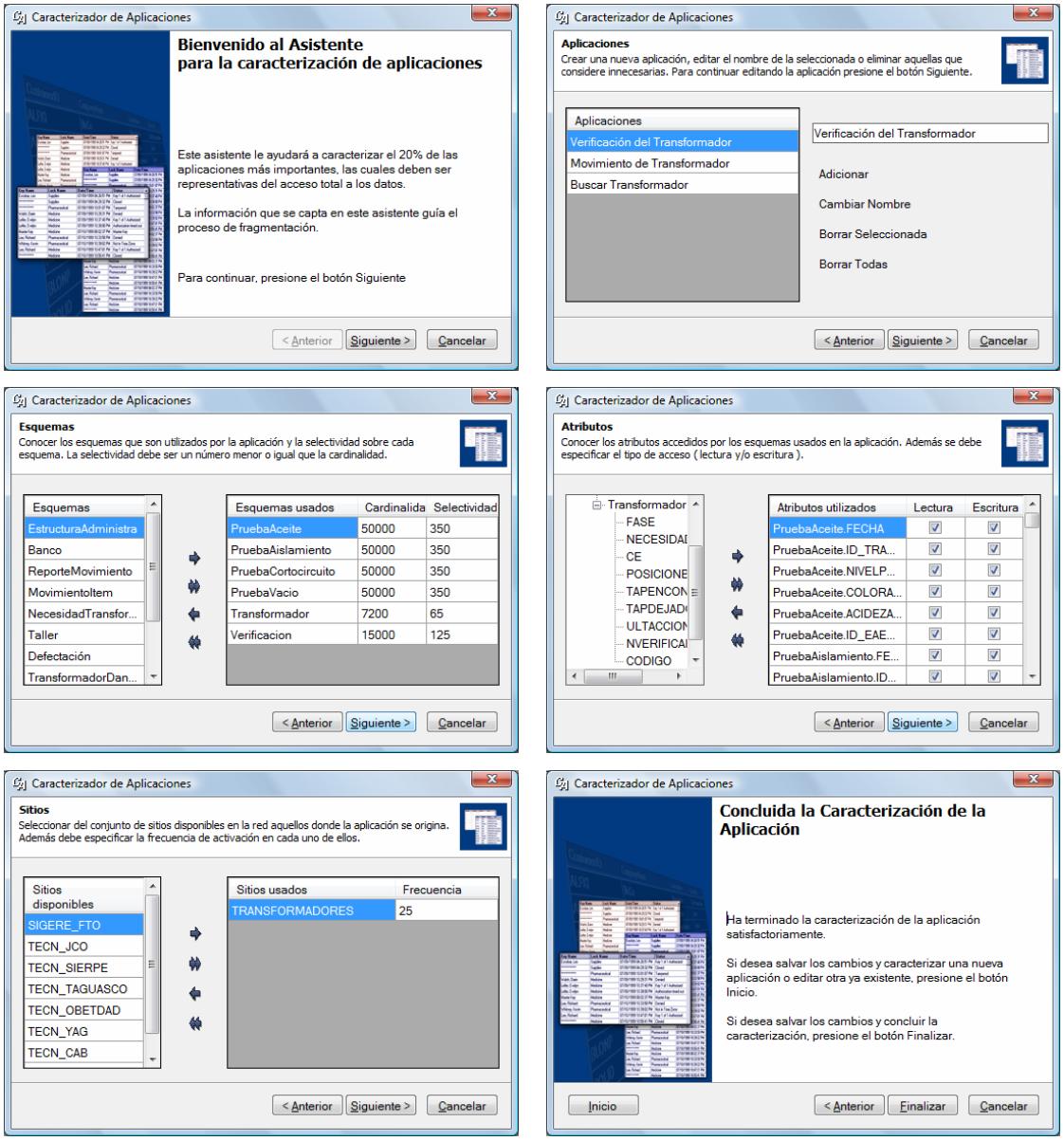

Figura A5.3. Vistas de APPWIZARD caracterizando la aplicación “Verificación del transformador”.

Anexo 6. Vistas de FRAGMENTER

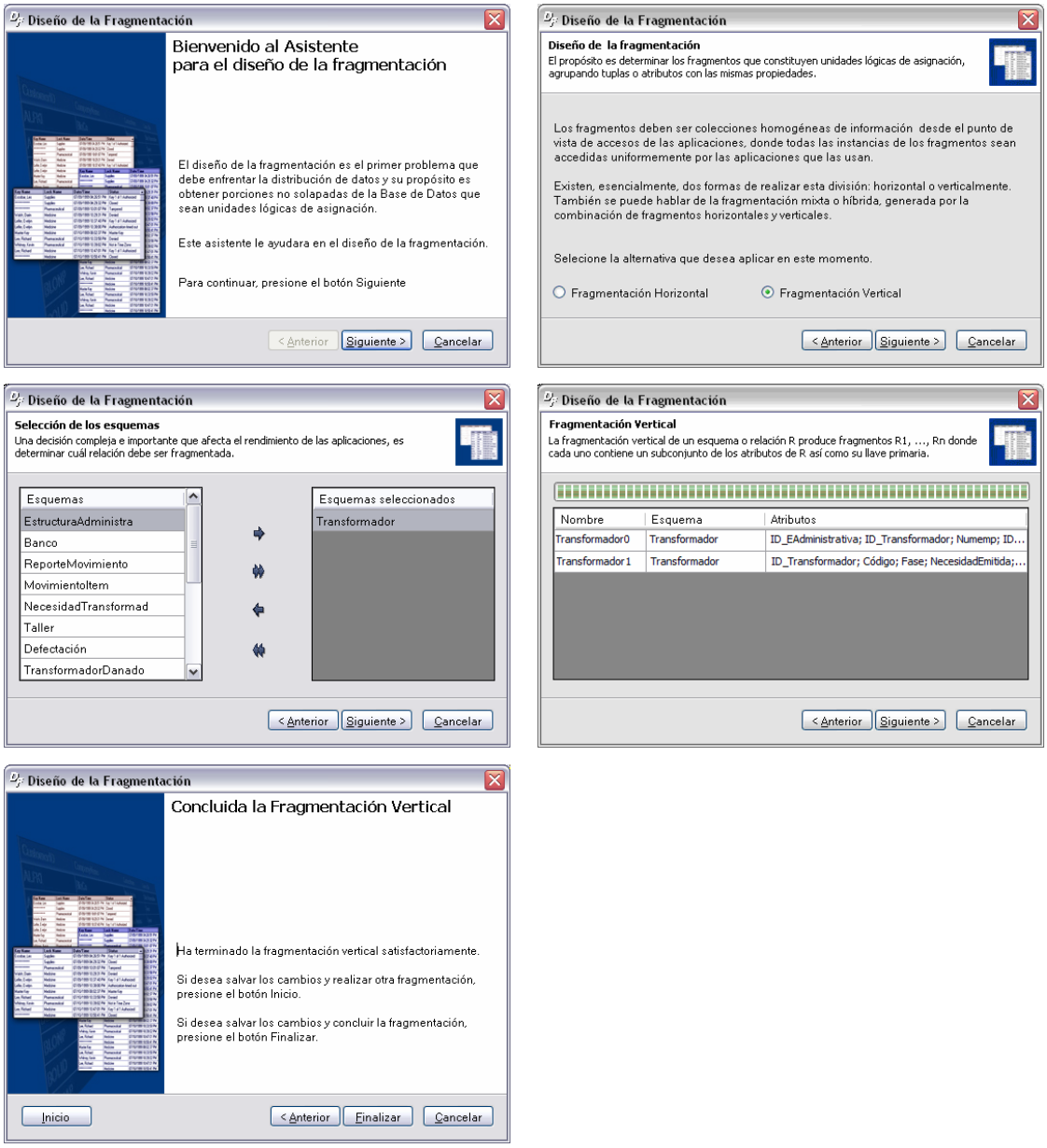

Figura A6.1. Vistas de FRAGMENTER realizando la fragmentación vertical.

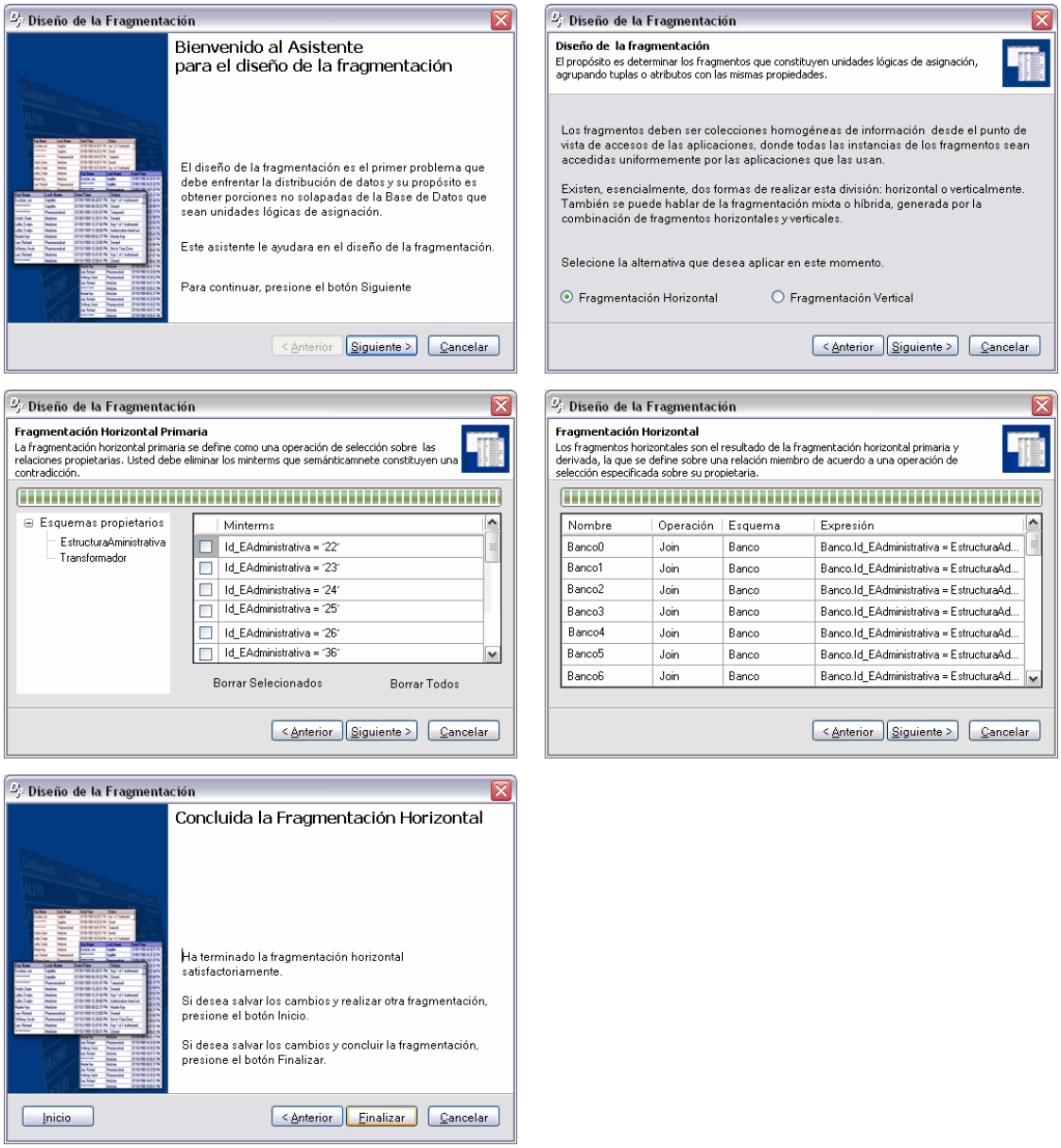

Figura A6.2. Vistas de FRAGMENTER realizando la fragmentación horizontal.

Anexo 7. Vistas de ALLOCATOR

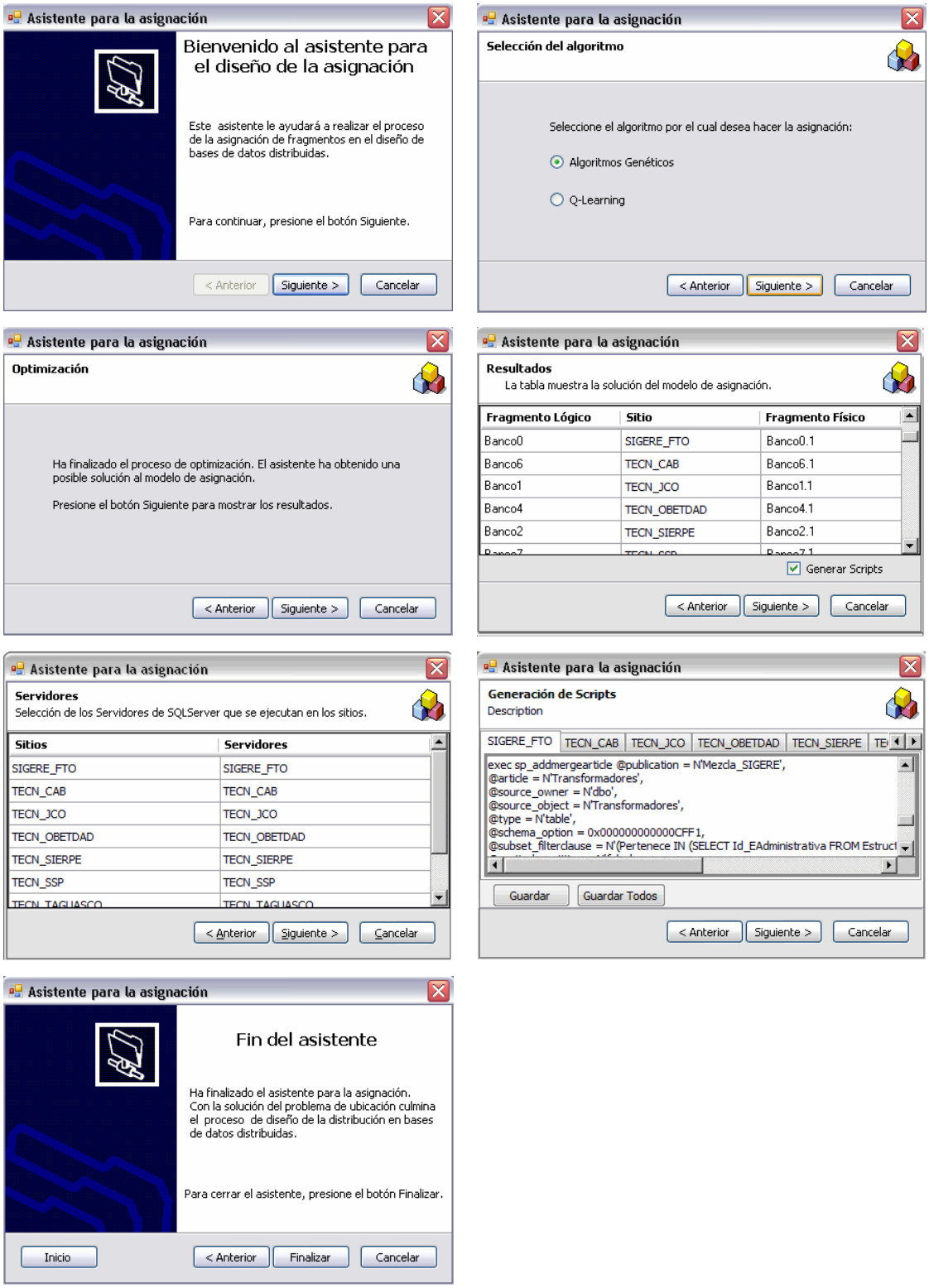

Figura A7.1. Vistas de ALLOCATOR realizando la asignación.

Anexo 8. Experimentación de M-AG y M-QL

Tabla A8.1. Resultados cuantitativos de la experimentación con M-AG y M-QL.

		M-AG					***M-QL***				
Caso	*Óptimo*	*AVE*	*VO*	*Max*	*T*	*ER*	*AVE*	*VO*	*Max*	*T*	*ER*
1	61 104,85	61 104,85	20	61 104,85	3,42	0,0000	61 106,27	19	61 133,38	0,25	0,0023
2	13 549,53	13 599,85	16	13 961,25	1,65	0,3714	13 559,16	17	13 613,77	0,24	0,0711
3	4 479,58†	4 479,58	20	4 479,58	2,50	0,0000	4 496,48	19	4 817,70	0,63	0,3773
4	52 659,82†	52 951,11	11	53 809,06	1,88	0,5532	52 659,82	20	52 659,82	0,67	0,0000
5	102 502,55†	103 040,35	1	104 292,49	3,60	0,5247	102 571,99	5	102 913,56	0,66	0,0677
6	54 498,22†	54 726,56	13	55 288,28	2,86	0,4190	54 651,85	15	55 285,40	0,67	0,2819
7	71 426,56†	71 918,28	6	72 957,96	4,86	0,6884	71 962,17	3	72 621,84	1,75	0,7499
8	26 087,00†	26 371,27	2	26 987,04	5,85	1,0897	26 672,93	2	27 983,13	2,12	2,2461
9	40 975,05†	41 251,02	1	41 613,12	7,49	0,6735	41 922,69	4	43 213,53	1,67	2,3127
10	51 282,82†	52 540,99	1	53 964,49	6,38	2,4534	52 080,81	2	52 861,03	2,40	1,5561
11	252 440,20†	257 096,76	2	26 1547,33	19,49	1,8446	259 728,43	2	266 100,01	8,37	2,8871
12	198 615,14†	204 943,25	2	212 613,52	52,15	3,1861	210 416,34	1	223 640,47	33,55	5,9417
13	369 647,17†	386 241,44	1	406 948,68	105,65	4,4892	400 167,30	1	426 956,78	60,91	8,2566
14	339 398,45†	349 973,94	1	361 746,39	125,80	3,1160	379 218,21	1	403 171,71	105,53	11,7325
15	549 275,72†	563 707,64	1	580 360,95	210,79	2,6274	629 684,21	1	688 023,40	170,20	14,6390

Óptimo: Costo de encontrar el óptimo, dado en milisegundos.

AVE: Costo promedio en encontrar las soluciones, dado en milisegundos.

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

Max: Costo de la peor solución.

T: Tiempo promedio en encontrar la solución, dado en segundos.

$$\text{ER} = \frac{\left|\text{AVE} - \text{Óptimo}\right|}{\text{Óptimo}} \cdot 100 : \text{Error relativo.}$$

† No se conoce el óptimo con exactitud.

Anexo 9. Resultados experimentales de los métodos M-AG y M-QL

Las figuras A9.1 y A9.2 muestran las gráficas del comportamiento de cada algoritmo tomando en cuenta el error relativo de las soluciones respecto al óptimo, en la primera, y el tiempo de ejecución de cada algoritmo, en la segunda, para los diferentes casos de prueba.

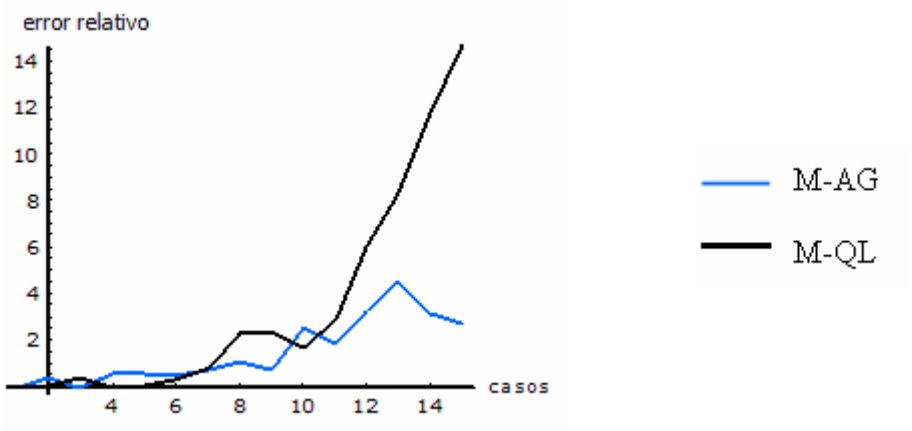

Figura A9.1. Comportamiento de los algoritmos teniendo en cuenta el error relativo.

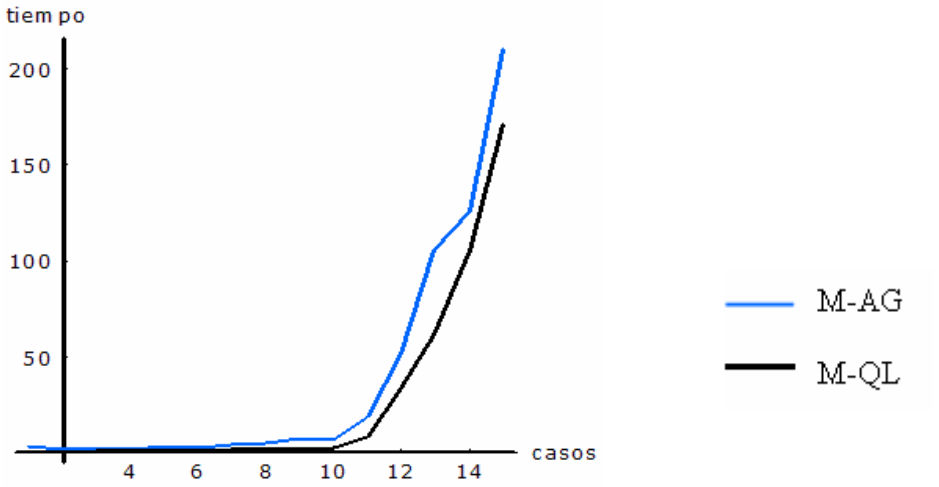

Figura A9.2. Comportamiento de los algoritmos teniendo en cuenta el tiempo de ejecución.

Anexo 10. Interfaz de la herramienta integrada