

## **INTEGRACIÓN DE HERRAMIENTAS DE AYUDA AL DISEÑO DE BASES DE DATOS DISTRIBUIDAS.**

Autores: MSc. Abel Rodríguez Morffi <sup>1</sup>, Dra. Luisa González González <sup>2</sup>, MSc. Alberto Morell Pérez <sup>3</sup>, Lic. Leonel Cabrera Tamayo, Lic. Michel Artiles Pérez, Lic. Lázaro Sergio Águila Díaz, Lic. Ángel Valdés López. Universidad Central “Marta Abreu” de Las Villas. Cuba.

<sup>1</sup> [arm@cei.uclv.edu.cu](mailto:arm@cei.uclv.edu.cu), <sup>2</sup> [luisagon@cei.uclv.edu.cu](mailto:luisagon@cei.uclv.edu.cu), <sup>3</sup> [amorellp@cei.uclv.edu.cu](mailto:amorellp@cei.uclv.edu.cu).

### **RESUMEN**

La posibilidad de distribuir datos sobre diferentes sitios de una red de computadoras y ejecutar programas que requieran el acceso a datos ubicados en más de un sitio son algunos de los factores que ha incrementado la difusión de las Bases de Datos Distribuidas, convirtiéndose en una alternativa viable para satisfacer las necesidades de información de las empresas. La ausencia de herramientas CASE que ayuden a disminuir la complejidad del diseño han motivado el desarrollo del presente trabajo, el cual integra un conjunto de herramientas de ayuda al diseño de Bases de datos Distribuidas desarrolladas por los autores donde se aplican los avances teóricos en esta área con la programación orientada a objetos, desarrollo basado en componentes y algoritmos genéticos.

### **PALABRAS CLAVES**

Bases de datos distribuidas, herramientas CASE, diseño de bases de datos.

### **I. INTRODUCCIÓN**

La cantidad de innovaciones tecnológicas que ha habido en los últimos años unido al auge de los sistemas de información distribuidos y el desarrollo de los manejadores de Bases de Datos comerciales, ha hecho que los Sistemas de Bases de Datos Distribuidas se conviertan en una alternativa viable para satisfacer las necesidades de información de las empresas. En el diseño de Bases de Datos Distribuidas aparecen las mismas fases que las Bases de Datos Centralizadas y la fragmentación y ubicación de datos en la red. Estos problemas en el diseño son resueltos en las herramientas desarrolladas que se integran en este trabajo.

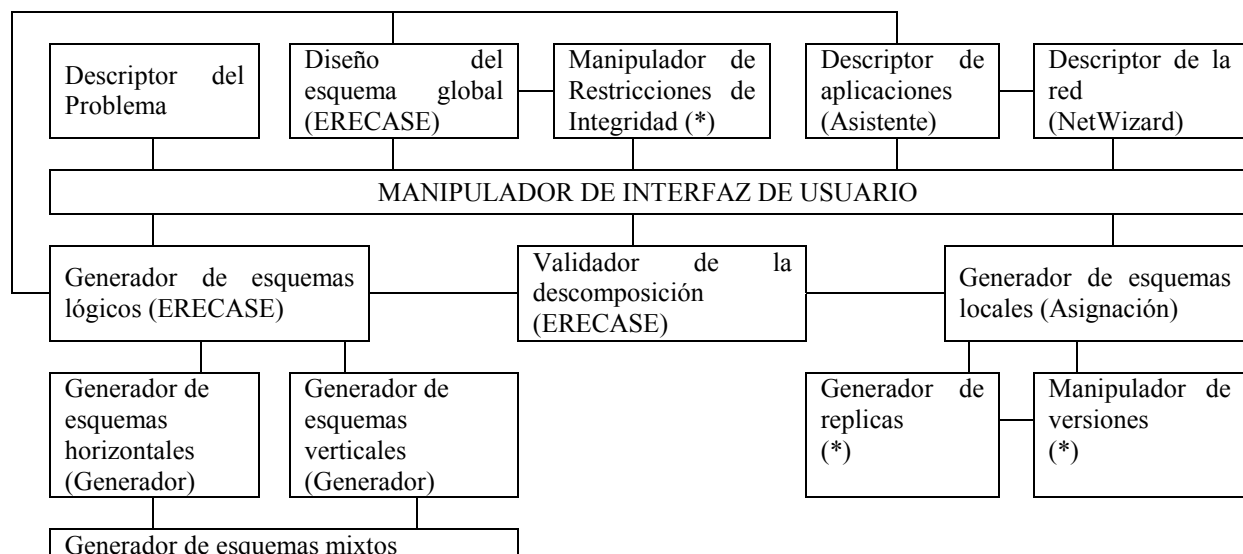
Existe un gran número de investigaciones que abordan este tema (BEL92, CER84, OZS91, OZS94, OZS98, ROD99), pero la mayoría se quedan en un plano teórico, sin aplicarse a un sistema de gestión en particular. Además, prácticamente no existen herramientas que ayuden a disminuir la complejidad del diseño. También hay buenas razones técnicas para distribuir datos, la más obvia es la referente a la sobrecarga de los canales de entrada y salida a los discos en donde se almacena finalmente la información. Otra razón de peso es que las redes de computadoras empezaron a trabajar a velocidades razonables abriendo una puerta a la distribución del trabajo y la información. Hacer una descentralización de la información se justifica desde el punto de vista económico y tecnológico por las siguientes razones: permitir autonomía local y promover la evolución de los sistemas y los cambios en los requerimientos de usuario; proveer una arquitectura de sistemas simple, flexible y tolerante a fallas; y ofrecer buenos rendimientos. Muchas aplicaciones que nacieron distribuidas. Para ellas ha sido necesario el uso de nuevas tecnologías para integrar sistemas de información diferentes, de forma que no se afecte de manera sustancial el estilo de trabajo. Aunque la idea de distribución de datos es bastante atractiva, su realización conlleva la superación de una serie de dificultades tecnológicas entre las que se pueden mencionar: asegurar que el acceso entre diferentes sitios o nodos y el procesamiento de datos se realice de manera eficiente, presumiblemente óptima; transformar datos e integrar diferentes tipos de procesamiento entre nodos de un

ambiente distribuido; distribuir datos en los nodos del ambiente distribuido de una manera óptima; controlar el acceso a los datos disponibles en el ambiente distribuido; soportar la recuperación de errores de diferentes módulos del sistema de manera segura y eficiente; y asegurar que los sistemas locales y globales permanezcan como una imagen fiel del mundo real evitando la interferencia destructiva que pueden ocasionar diferentes transacciones en el sistema. Así también, la aplicación de técnicas de distribución de información requiere superar algunas dificultades de índole organizacional y algunas otras relacionadas con los usuarios.

En el diseño de Bases de Datos Distribuidas se debe considerar el problema de cómo distribuir la información entre diferentes sitios (KAR97, MUT93, OZS98, ROD99, SHE96). Existen razones organizacionales, las cuales determinan en gran medida lo anterior. Sin embargo, cuando se busca eficiencia en el acceso a la información, se deben abordar dos problemas relacionados. Primero, cómo fragmentar la información. Segundo, cómo asignar cada fragmento entre los diferentes sitios de la red. En el diseño de la Bases de datos Distribuidas también es importante considerar si la información está replicada, es decir, si existen copias múltiples del mismo dato y, en este caso, cómo mantener la consistencia de la información. Finalmente, una parte importante en el diseño de una Bases de datos Distribuidas se refiere al manejo del directorio global, local o combinado. La información que se necesita para el diseño de distribución puede estar dividida en cuatro categorías: información sobre la BD, información sobre las aplicaciones, información sobre la comunicación de la red, e información sobre el sistema de las computadoras. Las últimas categorías son completamente cuantitativas en cuanto a su naturaleza y son usadas en los modelos de ubicación y no en los algoritmos de fragmentación (MUT93, OZS91, OZS98). El concepto de optimalidad que se usa durante el proceso de distribución y localización de fragmentos hace necesario el manejo y medición de algunos de los parámetros que caracterizan las redes de computadoras, ya que de su control y conocimiento depende en mucho el buen desempeño del sistema que se está diseñando.

## II. DESARROLLO

El diseño de una Bases de datos Distribuidas supone la integración de numerosos procesos (ROD99). Estas herramientas de ayuda al diseño de una Bases de datos Distribuidas han sido concebidas como un sistema interactivo basándose en componentes relativamente independientes que cooperan en la tarea de obtener propuestas de esquemas locales en los diferentes sitios de procesamiento de un sistema de cómputo distribuido. Las componentes y relaciones entre ellas a considerar son mostradas en el diagrama más abajo:



(\*) Concebido mas no suministrado hasta el momento

- Descriptor del problema: Permite describir, mediante un texto a modo de comentario, el problema que se modela.
- Manipulador de interfaz de usuario: Se encarga de la coordinación de todos los sistemas y su interacción con el usuario.
- Procesador Gráfico de Esquema Conceptual Global: Su función es generar, a partir del modelo ER o UML modificados, los esquemas globales independientes de la distribución en un formato interno que sirve como entrada a otros componentes del sistema. Por su utilidad en la modelación conceptual en Base de Datos Centralizadas se generan sentencias SQL estándar correspondientes a los esquemas.
- Manipulador de Restricciones de Integridad: Se encarga de captar las reglas de integridad relacionadas con el esquema global y relevantes en el proceso de diseño.
- Descriptor de las aplicaciones: Se ofrece una notación gráfica para caracterizar las principales aplicaciones que accederán a las Base de Datos Distribuidas.
- Descriptor de la Red: Permite captar los principales parámetros que describen el entorno de red que servirá de soporte al sistema de Bases de Datos Distribuidas.
- Generador de esquemas lógicos: En base a los requerimientos de información de los usuarios recogidos en el Esquema Conceptual Global, las restricciones de integridad definidas, las características de la red y las aplicaciones, se procede a generar los fragmentos que se requieren para garantizar un desempeño eficiente del sistema. Este módulo trata los dos tipos básicos de fragmentación: horizontal y vertical y su combinación en esquemas híbridos.
- Validador de descomposición: Posibilita al usuario interactuar con el sistema para considerar la propuesta que ofrece que ofrece el generador de esquemas lógicos. La propuesta ha sido verificada para comprobar su corrección.
- Generador de esquemas locales: Asigna los fragmentos obtenidos, sin redundancia, a los sitios de procesamiento identificados.
- Generador de réplicas: Permite hacer una asignación redundante de fragmentos a los sitios de procesamiento con el fin de minimizar tráficos de datos en la red y aumentar la capacidad de procesamiento local.
- Manejador de versiones: Brinda al usuario la posibilidad de variar determinados parámetros del modelo de asignación para obtener versiones de un mismo esquema de fragmentación que responda a sus requerimientos operacionales concretos.

## **II.1. COMPONENTES FUNDAMENTALES**

### ***II.1.1. ERECASE: Una herramienta de ayuda a la modelación de esquemas conceptuales globales.***

Las herramientas avanzadas de diseño deben tener una interfaz de usuario avanzada y poderosa que debe ser consistente; reconocer diseños anteriores y usar un lenguaje gráfico; ser flexibles y de alcance global; apoyar el proceso de diseño completo con editores para los diseños del esquema y del flujo de datos, analizadores, sintetizadores y transformadores; ser robustas; estar bien integrada y tener un eficiente apoyo teórico; ser eficiente en la adquisición de las semánticas, no sólo de los gráficos, las decisiones de diseño puedan ser discutidas con el usuario; generar alternativas; tener la capacidad de detectar la falta de información y ayudar a arreglar resultados erróneos. También puede mostrar diferentes versiones del mismo esquema y trabajar de igual forma con una u otra versión; además de ser extensibles en múltiples direcciones. Una herramienta no requiere que el usuario entienda la teoría, las restricciones de la implementación y los problemas de programación al realizar un diseño. Un diseñador inexperto puede crear un diseño correcto usando el sistema. Esta herramienta fue desarrollada con el propósito de incluirla dentro del paquete integrado de ayuda al diseño de bases de Datos Distribuidas que se expone aquí.

#### ***II.1.1.1. Módulos de una herramienta de diseño:***

Editor gráfico: Le permite al usuario especificar gráficamente la estructura de una aplicación, las restricciones que son válidas en la aplicación dada (y los procesos, operaciones y transacciones que sean necesarios). Esta

extensión requiere un soporte para gráficos avanzado y sencillo (ATK96, BAT92). El editor permite al diseñador representar la información del diseño, éste desarrolla la estructura, las semánticas estáticas y dinámicas, y las operaciones de la aplicación. Una vista parcial de la notación seguida puede apreciarse en el apéndice B.

Personalización: La interfaz de usuario es adaptada al nivel, habilidades e intenciones del diseñador.

Persistencia: El sistema de diseño almacena versiones del diseño actual, de anteriores o ejemplos. Estos pueden ser parciales o totalmente incluidos dentro del diseño actual y pueden ser usados para cambiar partes de éste.

Traductor: Esta componente traduce los resultados del proceso de diseño a un modelo lógico o físico y puede generar determinado código que puede ser usado en otras aplicaciones.

La arquitectura interna de ERECASE consta de tres módulos fundamentales; el editor gráfico, el validador y el traductor. Estos módulos interactúan entre sí durante el proceso de diseño de una Base de Datos. Véase la figura 1.

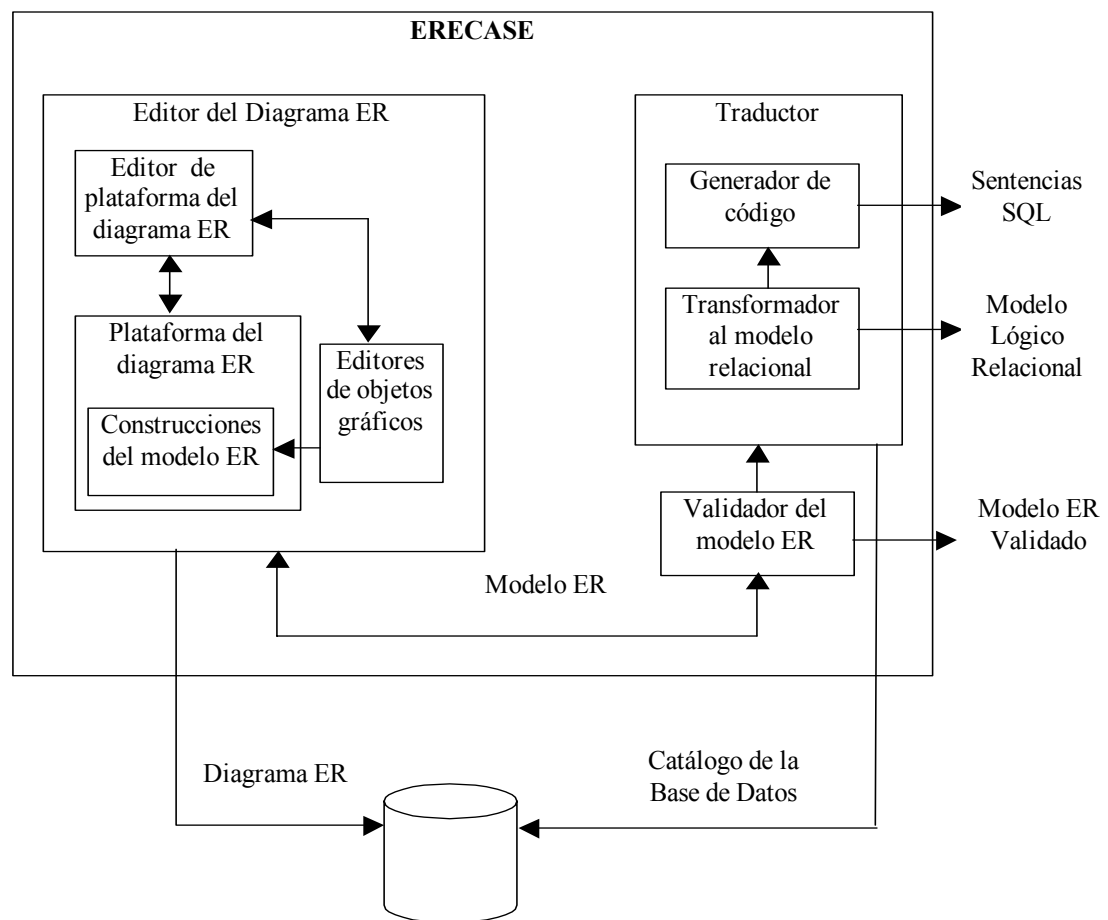


Figura 1. Arquitectura interna de ERECASE

El módulo editor gráfico tiene tres componentes fundamentales:

1. Plataforma del diagrama ER. Su función es contener las construcciones del diagrama ER y brindar el espacio necesario para su creación.
2. Editor de la plataforma. Es el encargado de manipular las funciones generales del diseño como la creación de nuevas construcciones, el copiado, pegado y eliminación; dibuja la rejilla de la plataforma para la ubicación de las construcciones; también es el encargado de notificar los eventos a los editores de las construcciones y actualizar el explorador del modelo.

3. Editores de los objetos gráficos. Son los que realizan las funciones gráficas específicas para cada construcción, entiéndase mover, redimensionar, invocar el editor de propiedades correspondiente, construir el menú contextual, etc.

El validador del modelo ER. Es el encargado de aplicar las reglas de validación a un esquema conceptual determinado determinando si éste es correcto o no.

Las partes fundamentales del módulo traductor son:

1. Transformador del modelo ER al relacional. Genera a partir de un esquema conceptual previamente validado y correcto el modelo lógico correspondiente.
2. Generador de código SQL. Construye las sentencias de creación de tablas a partir de un modelo relacional previamente generado.

#### *II.1.1.2. Construcciones que ofrece ERECASE para la creación del diagrama ER:*

ERECASE es una herramienta gráfica que apoya la creación de diagramas ER Extendido (ERE), la forma más ampliamente aceptada de definir el modelo conceptual de un sistema de Bases de Datos, y la transformación automática del modelo conceptual al modelo lógico equivalente (modelo relacional). A través del uso de interfaces gráficas, la herramienta le permite al usuario editar las propiedades de los conjuntos de entidades, interrelaciones y otras construcciones del modelo ERE.

Para la creación del modelo conceptual se ofrece en esta herramienta un conjunto variado de construcciones del modelo ERE. Dichas construcciones son:

- Conjunto de entidades fuertes y débiles.
- Asociaciones recursivas, binarias, ternarias y múltiples.
- Interrelaciones ISA, ID, TypeOf.
- Generalizaciones.
- Agregaciones.

Para representar el comportamiento de las construcciones se diseñaron dos jerarquías de clases paralelas, en una de estas jerarquías se representa el comportamiento interno de las construcciones o las características propias del modelo ERE; en la otra se representa el comportamiento visual de las construcciones o las características del diagrama ERE. Para el uso de estas dos jerarquías por otros módulos sin tener que conocer las características propias de cada una, se crearon un conjunto de interfaces que resumen el comportamiento de las construcciones del modelo ERE, cada clase implementa las interfaces respectivas y así para otros módulos una clase, ya sea visual o no, tiene el mismo comportamiento. Una clase visual delega la implementación de los métodos de las interfaces del modelo ERE que soporta, en la clase correspondiente que representa su comportamiento interno ERECASE tiene la capacidad de determinar si un esquema conceptual determinado es correcto o no, para esto aplica varios tipos de chequeos al esquema, estos son:

- Chequeo de unicidad de nombres: no permitir que existan dos construcciones o atributos dentro de las construcciones con el mismo nombre.
- Chequeo de llave primaria: 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.
- Chequeo de identificación de conjuntos de entidades débiles: todo conjunto de entidades débiles debe tener al menos una interrelación de identificación.
- Chequeo de conjunto de entidades huérfana: detectar cuándo un conjunto de entidades no forma parte de ninguna interrelación.
- Chequeo de la validez estructural de las asociaciones recursivas.
- Chequeo de ciclos de pertenencia de conjuntos de entidades débiles.

Se tomaron como punto de referencia (CUA99, DUL99, ELM97, SHO99)

#### *II.1.1.3. Transformador del modelo ERE al relacional.*

ERECASE sigue las reglas generales para la transformación del modelo ERE a modelo relacional. El transformador del modelo lógico es el encargado de aplicar estas reglas (ELM97, CUA99, DUL99) a un modelo ERE determinado y devolver el modelo lógico resultante. El modelo lógico constituye una lista de esquemas, que a su vez contienen una lista de campos. Para la implementación del modelo lógico se creó un conjunto de clases e interfaces que definen sus funciones principales. Estas reglas para la transformación del modelo ERE a modelo relacional son implementadas en una clase que, a partir de un modelo ERE especificado mediante un método de preparación, genera el modelo lógico equivalente. Este método de esta clase se debe invocar antes de la ejecución de la transformación para especificar el modelo ERE fuente y el modelo lógico destino. Adicionalmente hay una función que indica si el modelo lógico fue generado satisfactoriamente o si hubo algún error durante el proceso.

El generador de código SQL es el encargado de, a partir de un modelo lógico determinado, generar las sentencias de creación de tablas. Una clase implementa esta función a través de un método de generación, el código resultante de la generación se muestra usando otra clase construida para tal efecto. Se integran todos los módulos en una clase que también brinda la interfaz necesaria para la comunicación de la herramienta con el diseñador. Esta contiene todos los controles y opciones que permiten al diseñador interactuar con la herramienta, también es la encargada de cada vez que el diseñador abra un nuevo modelo, crear una plataforma para éste y el editor correspondiente.

#### *II.1.2. Asistente para la caracterización de aplicaciones.*

En el presente trabajo se realizó un estudio para determinar la información que sobre las aplicaciones es necesaria para el proceso de fragmentación, construyéndose un asistente para la caracterización de las mismas, el cual permite la captación de dicha información. Los datos de entrada y la salida que brindan los módulos para la caracterización están condicionados al diseño de dicha herramienta. Aquí se ha definido un catálogo al que todos los módulos que la componen tienen acceso tanto para obtener sus valores de entrada así como para guardar sus resultados. Para que el diseño de la distribución sea óptimo se necesita conocer información acerca de la base de datos, de las aplicaciones, sobre la red de computadoras y sobre las computadoras. El asistente para la caracterización de las aplicaciones permite obtener la información sobre las aplicaciones que es necesaria para realizar el proceso de fragmentación. Como no es posible investigar y caracterizar todas las aplicaciones que actuarán sobre la base de datos, se debe caracterizar al menos las más importantes. Se puede utilizar la regla “80/20”, es decir se debe caracterizar el 20 por ciento de las aplicaciones más activas las cuales son representativas del acceso total a los datos. Una vista reducida de uno de los pasos en la ejecución puede apreciarse en el apéndice A.

Este módulo permite captar las siguientes informaciones sobre las aplicaciones:

1. Nombre de la aplicación.
2. Esquemas que son utilizados por cada aplicación.
3. Información para la fragmentación vertical
  - De cada aplicación, qué atributos usa de cada esquema.
  - Los sitios donde se activa cada Aplicación y con qué frecuencia.
  - La selectividad de una aplicación en cada esquema, es decir cantidad de tuplas del esquema que accede la aplicación.
  - El tipo de acceso de cada aplicación, si es de lectura y/o escritura.
4. Información para la fragmentación horizontal
  - Los predicados simples.

El asistente para la caracterización de las aplicaciones utiliza el catálogo definido en la herramienta, como fuente para obtener los datos y donde guardar los resultados de la caracterización.

#### *Tablas de entrada*

- Schemata (Esquemas globales )
- GlobalAttributes(Atributos de los esquemas globales
- Sites (Sitios disponibles en la red)

#### *Tablas de salida*

- Applications (Conjunto de aplicaciones caracterizadas)
- App\_Schemata (Esquemas utilizados por las aplicaciones)
- App\_Attributes (Atributos utilizados por las aplicaciones)
- App\_Site (Sitios origen de las aplicaciones)
- Predicates (Predicados utilizados por las aplicaciones)

### **II.1.3. Generador de esquemas lógicos ( fragmentos).**

El diseño de la fragmentación es el primer problema que debe enfrentar la distribución de datos y su propósito es obtener porciones no solapadas de la BD que sean unidades lógicas de asignación. Estas unidades de asignación no pueden ser los atributos ni las tuplas por separado pues el sistema de asignación sería inmanejable, en su lugar se buscan agrupaciones o fragmentos con iguales propiedades desde el punto de vista de la asignación. La idea básica es que si todos los elementos del mismo fragmento tienen las mismas propiedades desde el punto de vista de su asignación, cualquier método usado para la asignación los ubicará juntos. El generador usa los algoritmos propuestos por (OZS91, MUT93, NAV84) y usa las siguientes estructuras:

#### *Tablas de entrada:*

- Schemata, Predicates y GlobalAttributes (véanse tablas manejadas por el caracterizador de aplicaciones)
- Links (Enlace entre los esquemas)

#### *Tablas de salida:*

- Operations (Operaciones realizadas sobre los esquemas)
- LogicalFragments (Fragmentos lógicos)

### **II.1.4. NetWizard: Descriptor de la red.**

La herramienta *NetWizard* implementa un modelo distribuido para la sociedad de agentes involucradas en la tarea de describir la red, o sea, capturar todos los parámetros necesarios para dar solución a los problemas de fragmentación y ubicación. Las tecnologías de COM y DCOM (MIC97, MIC98a, MIC98b) fueron utilizadas para la creación de objetos remotos a través de la red con el objetivo de lograr la comunicación entre los servidores (en este caso los Agentes) y el cliente, y para almacenar los datos que describen la red de computadoras. En cada sitio se localiza un servidor (agente). El modelo implementado es totalmente distribuido pues se permite tanto que los agentes como el conocimiento que se gestiona se encuentren físicamente distribuidos en una red. Un agente queda definido como un servidor que describe, a través de un conjunto de parámetros, el estado de su “ambiente”, es el responsable de la gestión de esos parámetros, así como de su procesamiento. Está diseñado para ser usado por el diseñador de Bases de datos Distribuidas. Debido a la complejidad del sistema, se hace una división lógica en tres dominios:

1. Servidor: Conformar la parte servidora del sistema. El que entrega los valores de los parámetros.
2. Cliente: Representa la parte cliente del sistema. El que solicita los parámetros.
3. Servicio: Posee las mismas funcionalidades del cliente, salvo que se planifica su funcionamiento al nivel de servicio de Windows NT

La interfaz de usuario está elaborada en forma de Asistente con varias páginas, en las cuales se predeterminan los parámetros de ejecución, se planifica la ejecución de la aplicación como servicio por días de la semana, o por fecha, se configuran parámetros de tipo estadístico y de formato de los resultados. La cantidad de interacciones por ejecución implica la cantidad de muestra con la que se va a trabajar. Las demás opciones están referidas a los gráficos que se obtienen luego de ejecutar el sistema. Una vista reducida de uno de los pasos en la ejecución puede apreciarse en el apéndice A.

A fin de hacer un análisis de la afectación que el sistema pudiera ocasionar el rendimiento de la red; se le aplicaron pruebas de medición de tráfico. En dichas pruebas se usó CommView versión 2.6 de TamoSoft Inc, tomando una muestra por espacio de 24 horas del desempeño de la red del Centro de Estudios de Informática, Universidad Central “Marta Abreu” de Las Villas, Cuba, sin la aplicación del software, y una segunda parte por el mismo espacio de tiempo con el software ejecutando a manera de servicio con una frecuencia de 15 minutos. Luego de un análisis del comportamiento de las medias y las varianzas en ambos casos se llegó a la conclusión de que el asistente sobrecarga el tráfico de la red en menos de 0.1 %, concluyendo que este valor no es significativo.

### ***II.1.5. Generador de esquemas locales: asignación de fragmentos.***

En un ambiente distribuido, los fragmentos de datos son ubicados en sitios diferentes, de manera que en el procesamiento de las consultas se combinan datos de diferentes sitios para obtener la respuesta final. Para hacer esto de una forma económica se requiere de una buena función de costo que justifique la fragmentación de los datos, contemple las frecuencias de uso de todas las consultas y actualizaciones, y los sitios a los que los resultados deben ser enviados. En el presente trabajo se hace un estudio de la asignación de datos en la red según propuestas en (OZS91), (SHE96) y se enfrenta el problema usando algoritmos genéticos generacionales (GOL89).

El catálogo incluye las tablas generadas y usadas por la fragmentación y los que aparecen más abajo:

- AccessTime (*Tiempos de acceso entre los sitios*)
- PhysicalFragment (*Fragmentos físicos*)
- FragmentAttributes (*Atributos del Fragmento*)
- DataType (*Tipos de datos de SQL Server*)

Una vista reducida de uno de los pasos en la ejecución puede apreciarse en el apéndice A.

## **III. CONCLUSIONES**

El presente trabajo consiste en el diseño e implementación de varias herramientas que se integran para dar solución al problema relacionado con el proceso de diseño de Bases de Datos Distribuidas que permiten ayudar a los diseñadores a distribuir sus datos con el empleo de métodos científicos. La herramienta descrita es capaz de captar toda la información requerida en el diseño de bases de datos distribuidas y efectuar la distribución, además, como dicho proceso es iterativo y tentativo, también le permite a los diseñadores conservar versiones de los estados alcanzados durante tal proceso. Esta herramienta aún está en fase de pruebas y correcciones, así sólo está disponible para la comunidad académica de la Universidad Central de Las Villas, Cuba.

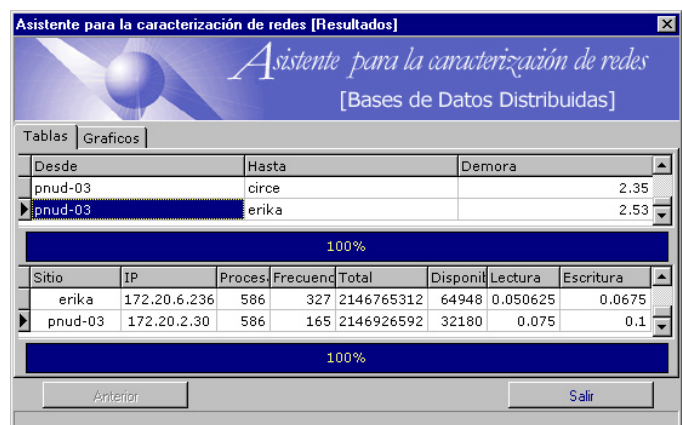
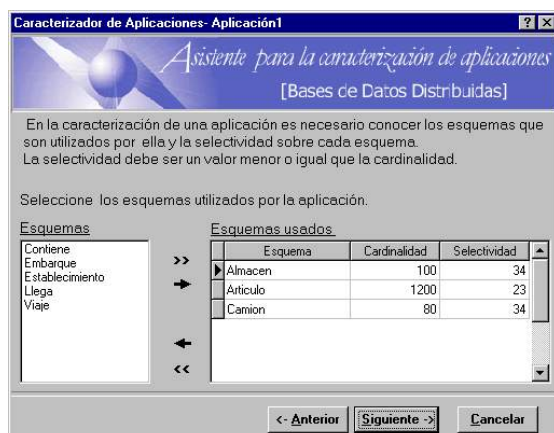
## **IV. BIBLIOGRAFÍA Y REFERENCIAS BIBLIOGRÁFICAS**

- [ATK96] Atkins, C. Prescription or Description: Some observations on the conceptual modeling process. Proceedings of International Conference on Software Engineering: Education and Practice, 1996.
- [BAT92] Batini, C., Ceri, S. y Navathe, S.B. Conceptual Database Design: an Entity-Relationship approach. The Benjamin/Cummings Publishing Company, Inc. 1992.
- [BEL92] Bell, D. A. and Grimson, J. B. Distributed Database Systems. Addison-Wesley, 1992.
- [CER84] Ceri, S. and Pelagatti, G. Distributed Databases - Principles and Systems. McGraw Hill, 1984.
- [CUA99] Cuadra, D., Nieto, C. y Martínez, P.; de Miguel, A. Control de restricciones de cardinalidad en una metodología de desarrollo de Bases de Datos Relacionales. Novática Jul/Ago 1999 no 140 pp 28-33.
- [DUL99] Dullea, J. y Song, Il-Ycol. A Taxonomy of Recursive Relationships and Their Structural Validity in ER Modeling. The proceeding of 16th International Conference on the Entity-Relationship Approach. pp 384-398.



- [ELM97] Elmasri, R. y Navathe, S. B. Sistemas de Bases de Datos. Conceptos fundamentales. Segunda Edición. Addison-Wesley Iberoamericana, Wilmington, Delaware, E.U.A, 1997.
- [GOL89] Goldberg, D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, 1989, 412 p.
- [KAR97] Karlapalem, K.; Ng Moon Pun, Query-Driven Data Allocation Algorithms for Distributed Database Systems, DEXA 1997. <http://www.cs.ust.hk/>
- [MIC95] Microsoft Corporation and Digital Equipment Corporation, "The Component Object Model Specification", Octubre 1995
- [MIC98a] Microsoft Corporation "The Registry" Microsoft Developer Library
- [MIC98b] Microsoft Corporation, "Inside Ole Second Edition" Microsoft Developer Library (Abril 1998).
- [MUT93] Muthuraj, J.; Chakravarthy, S.; Varadarajan, R.; Navathe, S.B., A Formal Approach to the Vertical Partitioning Problem in Distributed Database Design. PDIS, 1993. <http://www.cis.evl.edu/>
- [NAV84] Navathe, S.B.; Ceri, S.; Wiederhold, G.; Dou, J., Vertical Partitioning of Algorithms for Database Design, ACM, diciembre 1984.
- [OZS91] Öszu, M.T.; Valduriez, P., "Distributed Database Systems: where are we now?", IEEE Computer, August 1991, 24(8).
- [OZS94] Öszu, M.T.; Valduriez, P., "Distributed Database Systems: unsolved problems and new issues," In Readings in Distributed Computing Systems, T. L. Casavant and M. Singhal eds. IEEE Press, 1994.
- [OZS98] Öszu, M.T.; Valduriez, P., Principles of Distributed Database Systems 2nd Ed. Prentice Hall, 1998.
- [RAM91] Ram, S. (ed.). IEEE Computer. Special issue on heterogeneous distributed database systems, 1991, 24(12).
- [ROD99] Rodríguez Santos, J., "Diseño de Bases de datos Distribuidas". Disponible en: <http://www.angelfire.com/ar/hary/bdistribuidas.html>
- [SHE96] Shepherd, J.A.; Harangsri, B.; Chen, H.L.; Ngu, A.H.H., A Two-Phase Approach to Data Allocation in Distributed Databases, Fourth International Conference on Database Systems for Advanced Applications, World Scientific Press, Singapur, Abril 1995, 1996.
- [SHO99] Shoval, P. y Balaban, M. Resolving de "Weak Status" of a Weak Entity Types in Entity Relationship Schemas. The proceeding of 16th International Conference on the Entity-Relationship Approach. pp 369-383.
- [SIL98] Silberschatz, A., Korth. H. F. y Sudarshan, S. Database System Concepts. McGraw-Hill. Madrid, 1998.
- [WEI92] M. Weiss and F. Stetter, "A Hierarchical Blackboard Architecture for Distributed AI Systems". En Proceedings of Internacional Conference on Software Engineering and Knowledge Engineering, julio 1990.

## APÉNDICE A. Vista reducida de los asistentes implementados



Datos complementarios para el modelo

*Asistente para la asignación*  
[Bases de Datos Distribuidas]

Información adicional sobre las aplicaciones

Aplicación	Síto	Fragmento	Accesos de Lectura	Accesos de escritura	Selectividad
App1	Master-01	Enfermedad	0	0	0
App1	Master-01	Estudio	0	0	0
App1	Master-01	Medicina	0	0	0
App1	Master-01	Medico	0	0	0
App1	Master-01	Paciente	0	0	0
App1	Master-01	Tratamiento	0	0	0
App1	Master-01	Sala12	0	0	0
App1	Master-01	Sala13	0	0	0
App1	Master-01	Sala14	0	0	0
App1	Master-01	Sala15	0	0	0

< Anterior    Siguiente >    Cancelar

## APÉNDICE B. Vista parcial de ERECASE.

