UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
ESCUELA DE INGENIERIA DE SISTEMAS

"Planificación, Diseño y Aplicación de Bases de Datos Distribuidas utilizando la herramienta CASE GENEXUS."

TESIS PREVIA A LA OBTENCIÓN DEL

TITULO DE INGENIERÍA DE SISTEMAS

DIRECTOR:

ING. FRANCISCO SALGADO ARTEAGA

AUTORES:

EVA MARÍA MERCHAN CORDERO

JORGE GEOVANNY VINTIMILLA PACHECO

CUENCA-ECUADOR
2006

AGRADECIMIENTO

Nuestro profundo agradecimiento a todos nuestros maestros que de manera desinteresada nos han brindado toda su colaboración para sacar adelante este proyecto.

Un reconocimiento especial al Ing. Francisco Salgado, director de nuestra tesis, por su valiosa colaboración y guía en el desarrollo de este trabajo.

Los Autores.

RESUMEN

El objetivo de este proyecto es realizar el estudio de las bases de datos distribuidas. Para ello, se analizan los conceptos básicos, su estructura, su diseño, las ventajas y desventajas de su uso, los enfoques de diseño de distribución, la replicación y la fragmentación de los datos, el procesamiento de consultas con ejemplos gráficos, un análisis breve del control de la concurrencia, y un análisis de los sistemas operativos más adecuados para la implementación de un sistema de bases de datos distribuidas.

Este estudio incluye una aplicación práctica de Bases de Datos Distribuidas, en la que se ha utilizado la herramienta CASE Genexus 8.0. Se presentan los principios de la herramienta, la instalación de los Sistemas Operativos a utilizar, así como de los Gestores de Bases de Datos con los que puede funcionar.

ABSTRACT

The objective of this project is to analyze distributed database systems. For that purpose, this paper studies distributed data base basic concepts, structure, design, and usefulness; distribution design approaches, data replication and fragmentation, querying process with graphic examples, a brief analysis of concurrence control, and an examination of the most appropriate operating systems for the implementation of a distributed database system.

This study includes a practical application of distributed database systems, using CASE Genexus 8.0 software. The paper presents Genexus fundamental concepts, and discusses the platform of operating systems and data base management systems on top of which Genexus may be used.

INDICE GENERAL

CAPÍTULO 3
Configuración de Servidores

CAPÍTULO 4
Configuración de Gestores de Bases de Datos

CAPÍTULO 5
Configuración de la herramienta CASE Genexus

CAPÍTULO 1

CONCEPTOS Y DISEÑO ÓPTIMO DE BASES DE DATOS DISTRIBUIDAS

Introducción:

En este Capítulo presentaremos los conceptos claves para entender la Gestión de Bases de Datos Distribuidas. Estos incluyen aspectos tales como: el estado actual de la tecnología, criterios importantes a considerar en una base de datos distribuida, diseño, procesamiento, recuperación en sistemas distribuidos, control de concurrencia, y por último el sistema operativo.

Inicialmente presentaremos El Estado Actual de la Tecnología de las Bases de Datos Distribuidas en la cual determinamos el manejo transparente de datos distribuidos y replicados, y la confiabilidad que ofrecen los sistemas que manejan esta tecnología.
En los Criterios Importantes a Considerar analizaremos la estructura de una base de datos distribuida, y aspectos como la confiabilidad y el mejor desempeñote las mismas.
A continuación, en el Diseño de Base de Datos Distribuidas presentaremos los enfoques de diseño de distribución, la réplica de datos, lo cual incluye la disponibilidad, mayor paralelismo y el aumento de la sobrecarga de las actualizaciones, y la fragmentación de datos que involucra la fragmentación horizontal, vertical y la fragmentación mixta.
En Transparencia y Autonomía de la Red hablamos del requerimiento de que el sistema reduzca al mínimo la necesidad de que el usuario se de cuenta de cómo y donde está almacenada una fragmentación o una réplica.
Posteriormente analizaremos el Procesamiento de Bases de Datos Distribuidas, sección en la cual examinaremos la operación *semijoin* que consiste en la reducción de tuplas antes de ser transferidas al destino final.
En la sección Recuperación en Sistemas Distribuidos relatamos algunos de los problemas más comunes en este tipos de esquemas, los mismos que deberían ser tomados en cuenta al momento de estructurar la base de datos distribuida, para que el sistema sea menos vulnerable a fallos.
Analizaremos brevemente el control de concurrencia y como se aplica en las transacciones distribuidas.
Finalmente El Sistema Operativo es la sección en la cual recalcamos la importancia de un sistema operativo en el desempeño y un mejor rendimiento del sistema de bases de datos distribuidas.

1.1 Conceptos y Diseño Óptimo de Bases de Datos Distribuidas

1.1.1 Conceptos

Un ***Sistema de Gestión de Base de Datos*** (***DBMS*** DataBase Management System) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. El objetivo primordial de un DBMS es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos.

Según Henry Korth[1] "La gestión de datos implica tanto la definición de estructuras para el almacenamiento de información como la provisión de mecanismos para la gestión de la información. Además, los sistemas de bases de datos deben mantener la seguridad de la información almacenada, pese a caídas del sistema o intentos de accesos no autorizados. Si los datos van ha ser compartidos por varios usuarios, el sistema debe evitar posibles resultados anómalos".

Para los sistemas de bases de datos su primordial objetivo es mantener **la *independencia de los datos***. Se dice que un sistema es independiente de los datos cuando "los requerimientos de la aplicación no determinan la forma de organizar los datos y la técnica de acceder a ellos". Así, Rodolfo Rodríguez[2] describe la independencia de los datos como: "la inmunidad de programas de aplicación a cambios en la organización lógica o física de los datos y viceversa, donde el DBMS debe ser capaz de permitir cambiar la estructura de los archivos de información, cambiar los tipos de datos y longitudes de registros, hacer una administración óptima de la información, permitir crear campos nuevos en base a información existente, importar o exportar datos a formatos estándar sin tener que alterar los programas de la aplicación".

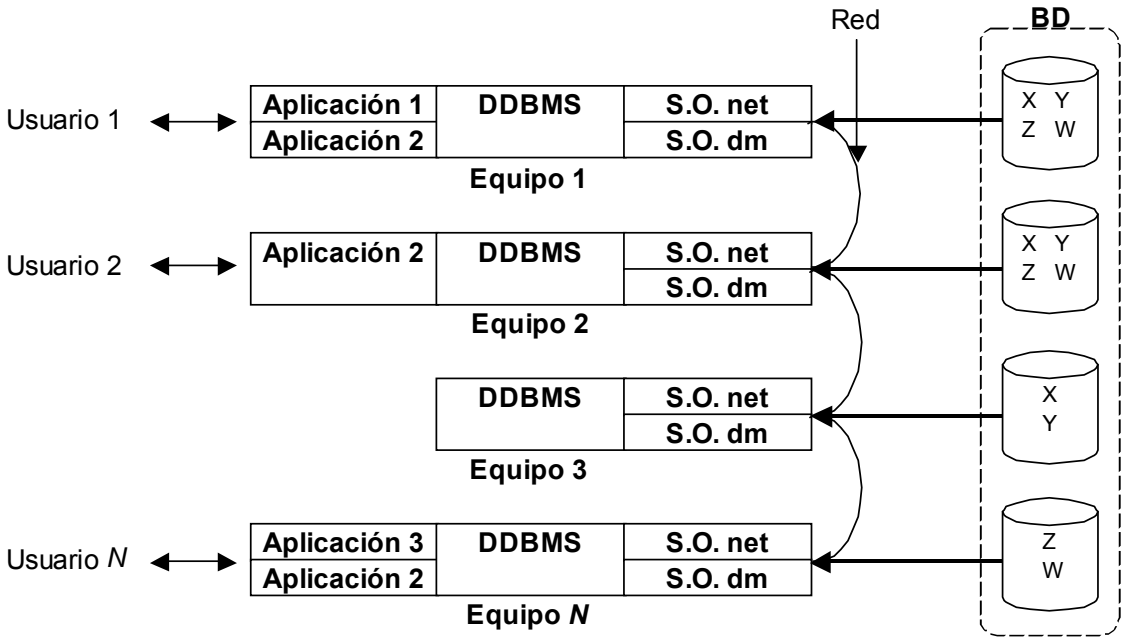
Todos los DBMS deben cumplir con la independencia de los datos, de esta manera la información es global para todos, al ser independientes no importa qué manejador sea el que acceda a la información, estos simplemente son datos y pueden visualizarse y manipularse desde cualquier DBMS. Estas son las ventajas que proporciona la independencia de datos.

Sin duda alguna muchos de los manejadores de bases de datos comerciales son robustos a estas variaciones, incluso los cambios se realizan en muchos de ellos sobre una interfaz gráfica y cada vez esta independencia se hace más transparente e implícita en el DBMS.[3]

Por otro lado hablaremos de los **S*istema de Bases de Datos Distribuidas***, en el cual la base de datos misma está distribuida, lógicamente interrelacionadas por medio de una red de computadoras. El autor David Kroenke[4] así lo explica "en la Figura 1-1, la base de datos (o una porción de la misma) está almacenada en todas las computadoras o en varias de ellas. Tal y como se muestra, las computadoras 1, 2 y *N* procesan tanto aplicaciones como la base

[1] Henry F. KORTH, Fundamentos de Bases de Datos, 2002, Cuarta edición.
[2] Rodolfo RODRÍGUEZ, Manual de Bases de Datos en Internet, fecha de acceso el 14/Mayo/2005 a la página: http://cariari.ucr.ac.cr/~rodolfor/bdatos.html
[3] Ibíd.
[4] David M. KROENKE, Procesamiento de Bases de Datos, 2003.

de datos, y la computadora 3 únicamente procesa la base de datos. La línea punteada alrededor de los archivos indica que la base de datos está compuesta de todos los segmentos de la base de datos en todos los *N* equipos. Tales equipos pudieran estar localizados en un mismo lugar, o en sitios distintos en el mundo, o algo intermedio, así como lo muestra la Figura 1-2".



DBMS = DBMS distribuido
S.O. net = Porción de comunicaciones de red del sistema operativo
S.O. dm = Porción de administración de datos del sistema operativo

Figura 1-1 Arquitectura de Base de Datos Distribuida[1].

Un ***Sistema de Manejo de Base de Datos Distribuida*** (DBMS distribuido) es el software de sistema que permite el manejo de la base de datos distribuidas y hace la distribución transparente a los usuarios.

a) Los datos se almacenan en varios sitios. Se supone que cada sitio consiste lógicamente de un solo procesador. Incluso cuando algunos sitios son máquinas multiprocesadoras, la DBMS distribuida nada tiene que ver con el almacenamiento y manejo de los datos en estas máquinas.

[1] Figura tomada del libro Procesamiento de Bases de Datos, de David M. KROENKE, 2003.

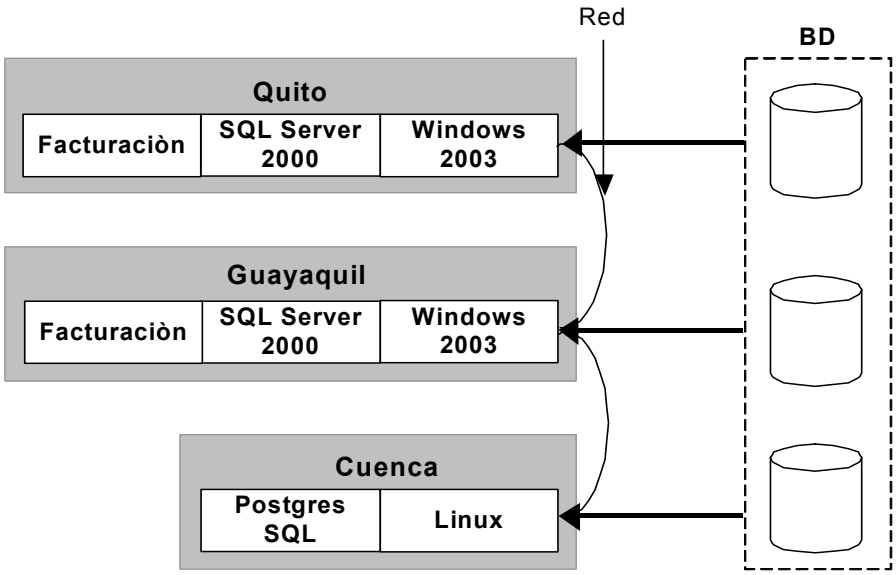


Figura 1-2 Base de Datos Distribuida para la demostración práctica de nuestro proyecto[1].

b) Los procesadores están interconectados por una red de computadoras y no por la configuración de un multiprocesador. El punto importante aquí es la pérdida de interconexión de procesadores que tienen sus propios sistemas operativos y que funcionan independientemente.

c) La base de datos distribuida es pues una base de datos, no una mera colección de archivos que pueden ser almacenados individualmente en cada nodo de la red. Esto señala la distinción entre una base distribuida y un sistema de archivos distribuidos. Para hacer una base de datos distribuida, los datos distribuidos deberían estar lógicamente relacionados de acuerdo con algunos formalismos estructurales, y el acceso a los datos debiera ser a un nivel alto por medio de una interface común.

d) El sistema tiene la funcionalidad completa de un DBMS. No es ni un sistema de archivos distribuidos ni un sistema procesador de transacciones. El procesamiento de transacción no es sólo un tipo de aplicación distribuida, sino que también es una de las funciones entre muchas de las que otorga un DBMS distribuido. Sin embargo, un DBMS distribuido provee otras funciones, como el procesamiento y la organización de los datos. Lo que los sistemas de transacción no necesariamente hacen.

Existen tres factores, que deben ser tomados en cuenta al momento de diseñar una base de datos distribuida, estos son:

Autonomía: se refiere a controlar la distribución e indica el grado en que un DBMS puede operar independientemente. Esto implica una serie de factores que toman en cuenta si los

[1] Adaptación de la figura, tomada del libro Procesamiento de Bases de Datos de David M. KROENKE, 2003.

sistemas componentes que intercambian información pueden ejecutar independientemente transacciones, y si uno está permitido a modificarlos.

Distribución: tiene que ver con los datos. Consideramos dos casos, ya sea que los datos estén distribuidos físicamente en sitios múltiples que se comunican entre sí o en alguna forma por un medio de comunicación o que están almacenados solo en un sitio.

Heterogeneidad: Se da en varias formas en los sistemas distribuidos, van de la heterogeneidad del disco duro y las diferencias de los protocolos de las redes hasta las variaciones de los manejadores locales.
Las formas importantes de heterogeneidad desde la perspectiva de los sistemas de base de datos son las diferencias en los modelos de los datos, interfaces, protocolos del manejo de transacciones.[1]

1.1.2 El estado actual de la tecnología de las bases de datos distribuidas

El manejo transparente de datos distribuidos y replicados: Los sistemas de base de datos centralizados nos han llevado de un paradigma del procesamiento de base de datos, en el que la definición de datos y mantenimiento estaban implicados en cada aplicación, a un paradigma en el que tales funciones son abstraídas de las aplicaciones y colocadas bajo el control de un servidor llamado el DBMS. Esta nueva orientación da como resultado lo que conocemos como independencia de los datos. La tecnología de bases de datos intenta extender el concepto e independencia de datos a ambientes en los que los datos son distribuidos y replicados en un 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 red (y por lo tanto, la distribución), transparencia de la réplica, y transparencia de la fragmentación.

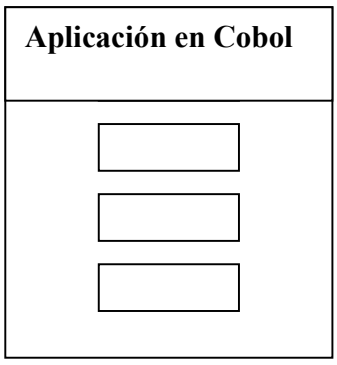


Figura 1-3 Definición y mantenimiento de datos dentro de la misma aplicación.

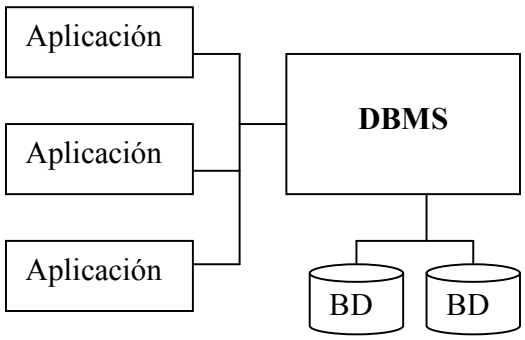


Figura 1-4 Independencia de datos

[1] Rodolfo RODRÍGUEZ, Manual de Bases de Datos en Internet, fecha de acceso el 14/Mayo/2005 a la página: http://cariari.ucr.ac.cr/~rodolfor/bdatos.html,

El acceso transparente a los datos separa un sistema de más alto nivel de uno de bajo nivel en la implementación en otros asuntos. Así, los usuarios de la base de datos verían una sola imagen, lógicamente integrada, aun cuando estuviera físicamente distribuida, permitiéndoles el acceso como si se tratara de una base de datos centralizada.

La mayoría de los DBMS distribuidos comerciales no proveen un nivel suficiente de transparencia; pero el sistema operativo puede ayudar con la reproducción de la transparencia, dejando la tarea de la transparencia en la fragmentación al DBMS distribuido.

Confiabilidad: En la actualidad las bases de datos distribuidas pretenden incrementar la confianza, por medio de componentes replicados[1], y con la garantía de apoyo a las transacciones[2] que ofrece un DBMS, asegurando así, que las transacciones se ejecuten adecuadamente.

1.1.3 Criterios importantes a considerar en una base de datos distribuida

1.1.3.1 Estructura de la Base de Datos Distribuida

Las localidades (microcomputadores, estaciones de trabajo, minicomputadores y sistemas de computadores grandes) en el sistema pueden conectarse físicamente de distintas formas. Las distintas configuraciones se representan por medio de grafos cuyos nodos corresponden a las localidades. Una arista del nodo A al nodo B corresponde a una conexión directa entre las dos localidades. En la Figura 1-3 se ilustra algunas de las configuraciones más comunes.

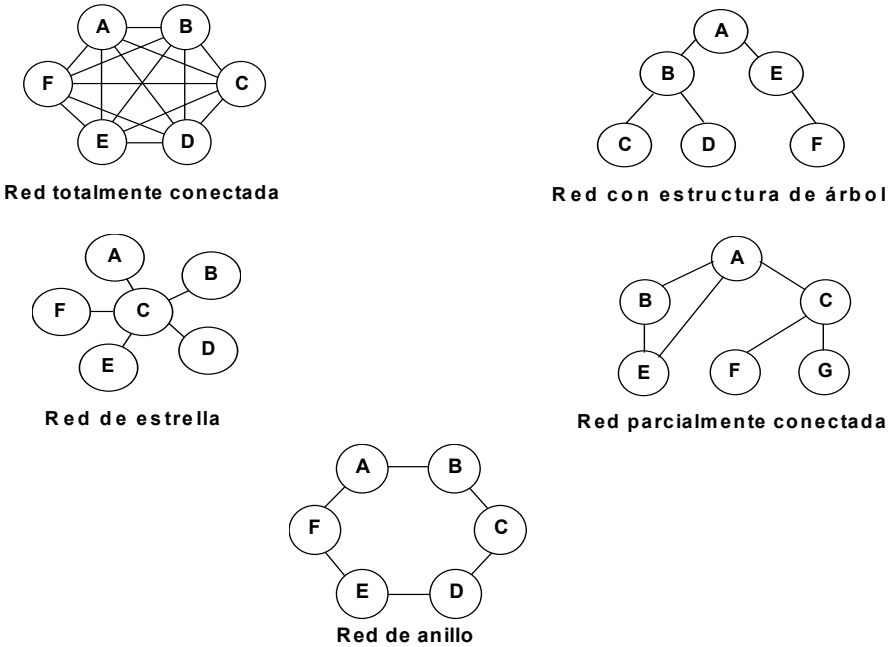


Figura 1-5 Configuración de red[1].

[1] Mayor información en la sección 1.1.4.3, Tema: Réplica de los datos.
[2] Mayor información en la sección 1.1.3, Tema: Confiabilidad.

Las diferencias principales entre estas configuraciones son:

a) Coste de instalación: El coste de conectar físicamente las localidades del sistema.
b) Coste de comunicación: El coste en tiempo y dinero que implica enviar un mensaje desde la localidad A a la B.
c) Fiabilidad: La frecuencia con que falla una línea de comunicación o una localidad.
d) Disponibilidad: La posibilidad de acceder a información a pesar de fallos en algunas localidades o líneas de comunicación.

Como se verá, estas diferencias juegan un papel importante en la elección del mecanismo adecuado para manejar la distribución de los datos.

Puesto que las localidades de red de larga distancia están distribuidas físicamente en un área geográfica extensa, es probable que las líneas de comunicación sean relativamente lentas y menos fiables en comparación con las redes de área local. Por otra parte, como todas las localidades de redes de área local están próximas entre sí, las líneas de comunicación son de mayor velocidad y menor tasa de errores que sus equivalentes en redes de larga distancia[1].

1.1.3.2 Confiabilidad

Las bases de datos distribuidas pretenden mejorar la confiabilidad, ya que tienen componentes replicados y por lo tanto eliminan puntos únicos de fallo. El fallo de un solo sitio, o el fallo en la unión de la comunicación que hace de uno o más sitios inalcanzables, no es suficiente para bajar el sistema completo. En una base de datos distribuida esto significa que algunos de los datos son inalcanzables, pero con el cuidado adecuado, los usuarios pueden acceder a otras partes de la base. Este cuidado especial viene a formar apoyo para las transacciones distribuidas. Una transacción consiste de una secuencia de operaciones ejecutada como una acción atómica[2] que transforma una base de datos de un estado consistente a otro, aun cuando un número no determinado de tales transacciones es ejecutado concurrentemente (a veces llamada transparencia concurrente), y aun cuando las fallas suceden (atomicidad de fallo). Las transacciones distribuidas actúan en sitios diversos, donde acceden a la base de datos local.

El apoyo en la transacción requiere la implementación de:

- Control de concurrencia distribuido: es la actividad de coordinar accesos a las bases de datos, es decir, no permitir que dos o más transacciones se bloqueen entre ellas. El control de concurrencia se preocupa de que muchos usuarios hagan transacciones y que además no se transfieran para que hayan errores.
- Protocolos de confiabilidad distribuidos: Consiste en protocolos de compromiso y procedimientos de recuperación. Los primeros se encargan de garantizar la atomicidad en las transacciones y, los últimos especifican cómo la consistencia de la base de datos global va a ser restaurada después de una falla.

[1] Ibíd.

[2] Atómica: ejecutan completamente todas las instrucciones de la transacción, o no se ejecuta ninguna. Además, en el caso de ejecución concurrente, el efecto de ejecutar una transacción debe ser el mismo que si se ejecutara sola en el sistema.

- Protocolos de ejecución: Estos obligan la atomicidad de transacciones distribuidas asegurando que una transacción determinada tenga el mismo efecto (ejecutar o abortar) en cada sitio donde éste exista.
- Protocolos de control: Es necesaria la implementación de protocolos de control para llevar a cabo el copiado o reproducción de los datos. El procedimiento de control más común es el de la equivalencia de una copia, la que se lleva a cabo por medio del protocolo ROWA (léelo, escríbelo todo). En ROWA, una operación de lectura lógica de un dato reproducido se convierte en una operación de lectura física en cualquiera de sus copias, pero una operación de escritura lógica es convertida en una operación física en todas las copias.

Entonces, un DBMS provee garantías de apoyo a las transacciones distribuidas, donde aplicaciones del usuario pueden acceder a una sola imagen lógica de la base de datos y confiar en el DBMS distribuido para asegurarse que sus peticiones serán ejecutadas correctamente sin importar lo que ocurra en el sistema.

1.1.3.3 Mejor desempeño

El caso del desempeño superior para las bases de datos distribuidas está basado en dos puntos:
Primero, un DBMS distribuido fragmenta la base de datos conceptual (diseño lógico de la base de datos), permitiendo que los datos sean almacenados en el lugar más próximo a su punto de uso. Esta característica, llamada localización de los datos, tiene dos ventajas potenciales:

1. Ya que cada sitio maneja solo una porción de la base de datos, el manejo de los mismos para el CPU y el servicio I/O no es tan difícil como en las bases de datos centralizadas, que manejan la totalidad de la base de datos.

2. La localización reduce la demora al acceso remoto, lo que generalmente ocurre en redes de área grande (por ejemplo, la tardanza en la propagación del mensaje de ida y vuelta mínimo en sistemas de satélite, es de un segundo aproximadamente).

La mayoría de los sistemas de base estructurada están diseñados para ganar el máximo beneficio en la localización de los datos. Pero esto solamente se obtiene por medio de la fragmentación apropiada y la distribución de la base de datos.

Para Rodolfo Rodríguez, la motivación principal para el desarrollo de bases de datos distribuidas es la de reducir la comunicación colocando a los datos tan cerca como sea posible de las aplicaciones que los usan. De este modo en una base de datos bien diseñada "el 90% de los datos deberían ser encontrados en el sitio local, y solo 10% en un sitio remoto". [1]

Sin embargo, esto raramente sucede, más a menudo el diseñador se enfrenta con los intercambios porque varias aplicaciones necesitan acceder a los mismos datos desde

[1] Rodolfo RODRÍGUEZ, Manual de Bases de Datos en Internet, fecha de acceso el 14/Mayo/2005 a la página: http://cariari.ucr.ac.cr/~rodolfor/bdatos.html,

diferentes lugares. En este caso, el diseño más efectivo es el que asegura la localidad al número más grande de aplicaciones, en términos de frecuencia de la ejecución de las peticiones en cada sitio.

Segundo, el paralelismo inherente de los sistemas distribuidos puede ser explotado por el inter-cuestionamiento (interquering) y el paralelismo de intra-cuestinamiento (intraquery). El paralelismo de inter-cuestionamiento es el resultado de la ejecución de múltiples consultas al mismo tiempo. El paralelismo de intra-cuestionamiento se logra cuando se rompe una consulta en varias peticiones (sub-peticiones), cada una realizada o ejecutada en sitios diferentes, entrando a una parte diferente de la base de datos distribuida.

Si el usuario entra a la base de datos solo para consultas (acceso de lectura), entonces el paralelismo de inter-cuestionamiento e intra-cuestionamiento implicaría que se reprodujera la información tanto como fuera posible. Sin embargo, debido a que la mayoría de los accesos no son sólo de lectura, la mezcla de lectura y de actualización, estas operaciones requieren la implementación de protocolos elaborados de ejecución y de control de concurrencia.

1.1.4 Diseño de bases de datos distribuidas

Al escoger el diseño de una base de datos, el diseñador debe hacer un balance entre las ventajas y las desventajas de la distribución de los datos. Se verá además que existen dos enfoques del diseño de distribución. Existen varios factores que deben ser tomados en cuenta al momento de almacenar una relación en la base de datos distribuida, estos son: la réplica y la fragmentación de los datos.

1.1.4.1 Consideraciones para la distribución de la base de datos

Existen varias razones para construir sistemas distribuidos de bases de datos que incluyen compartir la información, fiabilidad y disponibilidad, y agilizar el procesamiento de las consultas. Sin embargo, estas ventajas vienen acompañadas de varias desventajas, como son mayores costes de desarrollo de software, mayor posibilidad de errores y el aumento en el coste extra del procesamiento.

1.1.4.1.1 Ventajas de la distribución de datos

- La principal ventaja de los sistemas distribuidos de bases de datos es la capacidad de compartir y acceder a la información de una forma fiable y eficaz.

- **Utilización compartida de los datos y distribución del control**
  Si varias localidades diferentes están conectadas entre sí, entonces un usuario de cada localidad puede acceder a datos disponibles en otra localidad.
  La ventaja principal de compartir los datos por medio de la distribución es que cada localidad pueda controlar hasta cierto punto los datos almacenados localmente. En un sistema distribuido existe un administrador global de la base de datos que se encarga de todo el sistema.

- **Fiabilidad y disponibilidad**
  Si se produce un fallo en una localidad en un sistema distribuido, es posible que las demás localidades puedan seguir trabajando. En particular, si los datos se repiten en

varias localidades, una transacción que requiere un dato específico puede encontrarlo en más de una localidad. Así, el fallo de una localidad no implica necesariamente la desactivación del sistema.
El sistema debe detectar cuando falla una localidad y tomar las medidas necesarias para recuperarse del fallo. El sistema no debe seguir utilizando la localidad que falló. Por último, cuando se recupere o repare esta localidad, debe contarse con mecanismos para reingresarla al sistema con el mínimo de complicaciones.
Aunque la recuperación de fallos es más compleja en sistemas distribuidos que en los centralizados, la capacidad que tiene el sistema para seguir trabajando, a pesar del fallo de una localidad, da como resultado una mayor disponibilidad. La disponibilidad es fundamental para los sistemas de base de datos que se utilizan en aplicaciones de tiempo real.

- **Agilización del procesamiento de consultas**
  Si una consulta comprende datos de varias localidades, puede ser posible dividir la consulta en varias sub-consultas que se ejecutan en paralelo en distintas localidades. Sin embargo, en un sistema distribuido no se comparte la memoria principal, así que no todas las estrategias de intersección para procesadores paralelos se pueden aplicar directamente a los sistemas distribuidos. En los casos en que hay repetición de los datos, el sistema puede pasar la consulta a las localidades más ligeras de carga.

1.1.4.1.2 Desventajas de la distribución de datos

La desventaja principal de sistemas distribuidos de bases de datos es la mayor complejidad que se requiere para garantizar una coordinación adecuada entre localidades.
El aumento de la complejidad se refleja en:

a) Coste de desarrollo de software: Es más difícil estructurar un sistema de base de datos distribuido y, por tanto, su coste es mayor.
b) Mayor posibilidad de errores: Puesto que las localidades del sistema distribuido operan en paralelo, es más difícil garantizar que los algoritmos sean correctos. Existe la posibilidad de errores extremadamente sutiles.
c) Mayor tiempo extra de procesamiento: El intercambio de mensajes y los cálculos adicionales que se requieren para coordinar las localidades son una forma de tiempo extra que no existe en los sistemas centralizados.

1.1.4.2 Los enfoques TOP-DOWN y BOTTOM-UP de diseño de distribución

El enfoque top-down es típico de las bases de datos distribuidas desarrolladas desde una planeación previa, mientras que el enfoque bottom-up es típico del desarrollo de los sistemas de base de datos múltiples como una agregación a los sistemas de bases ya existentes.

El primer enfoque, como lo demuestra la Figura 1-6, supone que el diseñador entiende los requerimientos de una aplicación de la base de datos del usuario, y la transforma en especificaciones formales. Durante este proceso, el diseñador lleva a cabo fases de diseño conceptuales, lógicas y físicas, las que progresivamente mejoran las especificaciones de alto nivel e independientes del sistema a la base de datos dentro de las especificaciones de bajo nivel y dependientes del sistema. Durante el diseño conceptual, el diseñador ignora

cualquier detalle que tiene que ver con la implementación física (en particular, la distribución de los datos). El resultado es un esquema de base de datos global que incorpora, en un nivel abstracto, todos de los elementos de los datos de la base y los patrones de su uso.

En cambio el modelo Bottom-Up asume que una especificación de las bases de datos ya existe, ya sea por que hay bases de datos que tiene que ser interconectadas a un sistema de bases múltiples, o porque la especificación conceptual de las bases ha sido hecha para cada sitio independientemente. En cualquiera de los dos casos, las especificaciones de los sitios tienen que estar integradas para generar una especificación global.

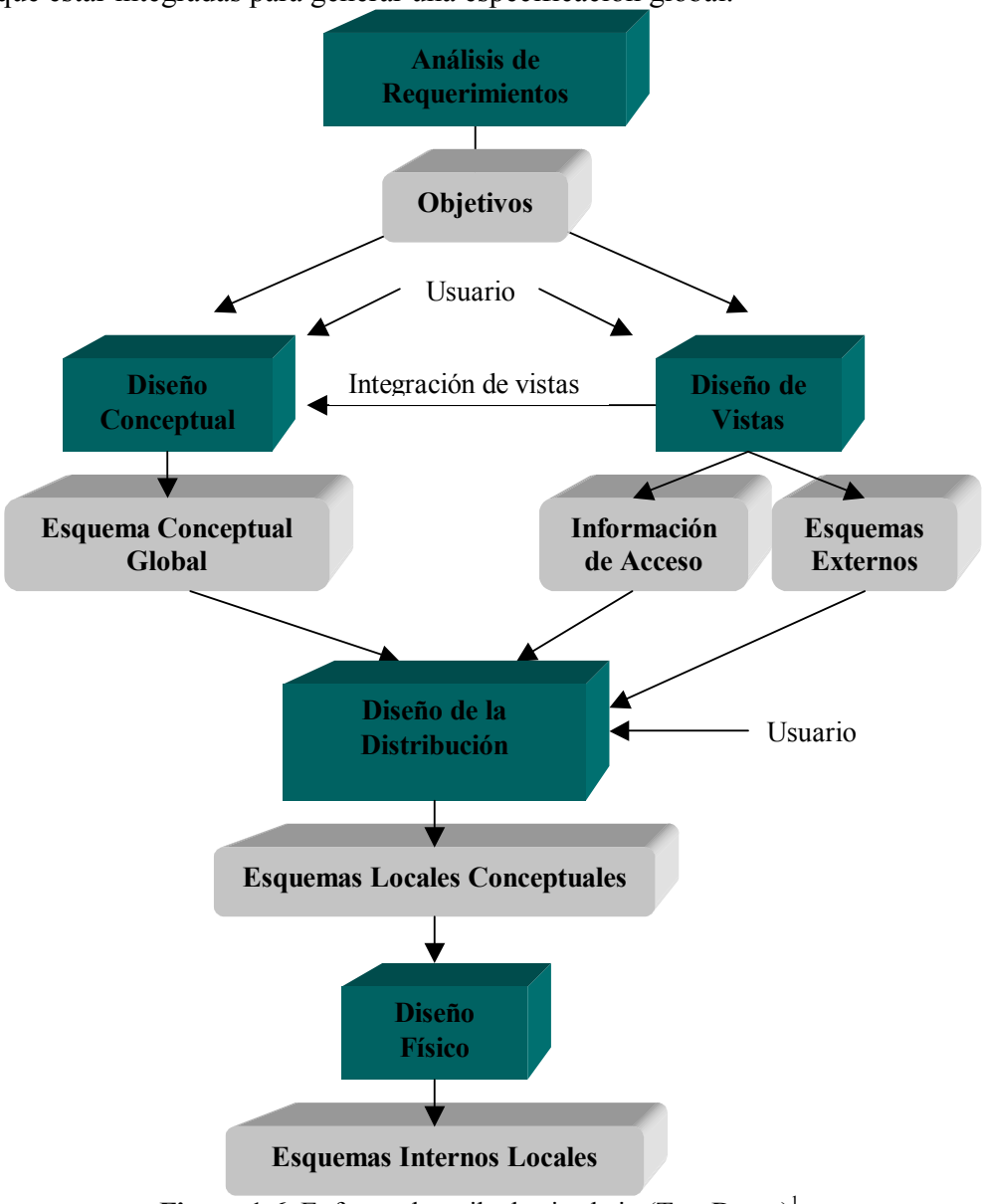


Figura 1-6 Enfoque de arriba hacia abajo (Top-Down)[1].

[1] Tomado del documento Diseño de Base de Datos Distribuidas, 2003. Autores: Universidad Carlos III.

Mientras que ambos enfoques parecen representar dos extremos opuestos, en muchos casos prácticos el diseñador procede usando ambos modelos.

1.1.4.3 Réplica de los datos

Al hablar de réplica indicamos que el sistema conserva varias copias o réplicas idénticas de la tabla. Cada réplica se almacena en un nodo diferente, lo que da lugar a redundancia de datos. La réplica presenta las siguientes ventajas e inconvenientes:

- **Disponibilidad**: Como se vio anteriormente, la réplica permite que el sistema siga funcionando aún en caso de caída de uno de los nodos.

- **Mayor paralelismo**: En el caso de que la mayoría de los accesos a la tabla *T* sean de lectura, varios nodos pueden realizar consultas en paralelo sobre la misma tabla. Cuantas más réplicas existan de la tabla, mayor será la posibilidad de que el dato buscado se encuentre en el nodo desde el que se realiza la consulta, minimizando con ello el tráfico de datos entre nodos. Ver Figura 1-7.

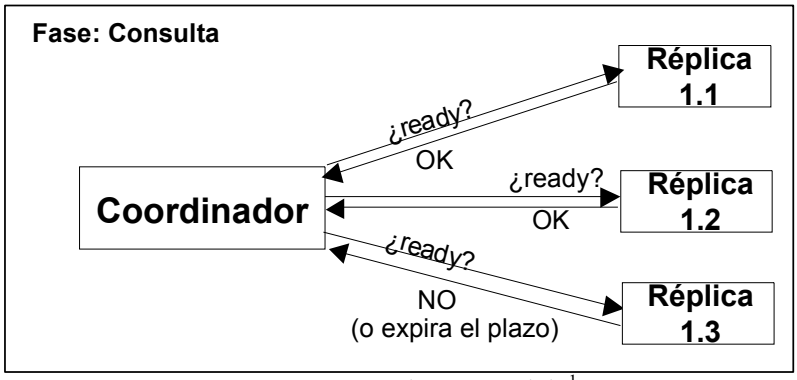


Figura 1-7 Consulta en paralelo[1].

- **Aumento de la sobrecarga en las actualizaciones**: El sistema debe asegurar que todas las réplicas de la tabla sean consistentes, por tanto, cuando se realiza una actualización sobre una de las réplicas, los cambios deben propagarse a todas las réplicas de dicha tabla a lo largo del sistema distribuido. Esto hace que las actualizaciones sean más costosas que en los sistemas centralizados. Ver Figura 1-8.

[1] Adaptación de la figura, tomada del documento Diseño de Base de Datos Distribuidas, UNIVERSIDAD CARLOS III.

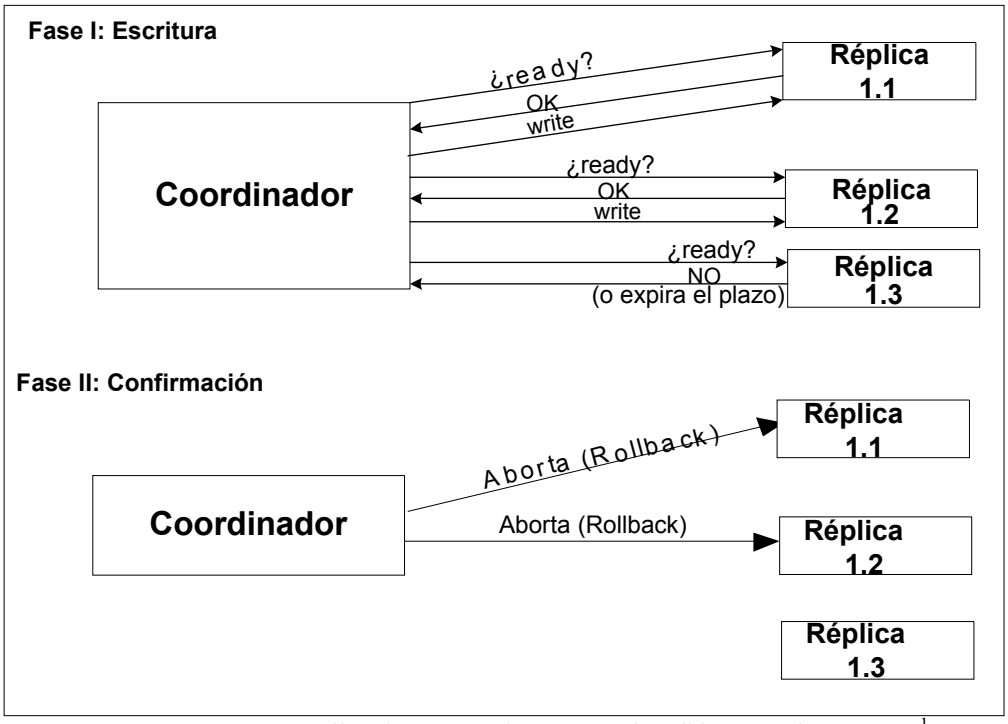


Figura 1-8 Actualizaciones consistentes en los diferentes fragmentos[1].

En general, el sistema de réplica hace que las consultas sean más eficientes, pero complica las actualizaciones debido al problema de la redundancia de datos y el mantenimiento de la consistencia. Normalmente, se elige una de las réplicas como copia principal para simplificar la administración.

1.1.4.4 Fragmentación de los datos

Razones para fragmentar:

- ✓ mejor unidad de distribución
- ✓ menos transmisión por red
- ✓ menos datos duplicados
- ✓ mayor grado de concurrencia
- ✓ ejecución en paralelo de consultas

La tabla *T* se puede separar en varios fragmentos[2] que contendrán suficiente información para reconstruir la tabla original.[3] La fragmentación puede ser horizontal o vertical y se puede aplicar en forma sucesiva a la misma tabla, así como lo muestra la figura 1-9. Cada fragmento se almacena en una localidad diferente.

[1] Adaptación de la figura, tomada del documento Diseño de Base de Datos Distribuidas, UNIVERSIDAD CARLOS III.

[2] Un fragmento puede definirse como una selección en la tabla global *T*.

[3] La reconstrucción de la tabla original puede llevarse a cabo ya sea aplicando la operación de unión o un tipo especial de operación de unión sobre los diversos fragmentos.

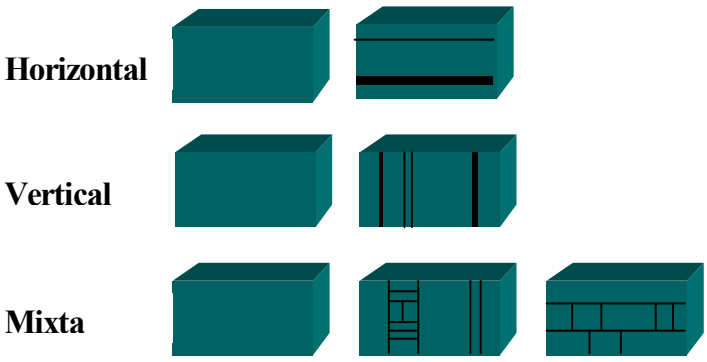


Figura 1-9 Tipos de fragmentación[1].

- **Fragmentación horizontal:**
  La tabla *T* se divide en subconjuntos, T_1, T_2, ...T_n. Cada tupla de *T* debe pertenecer al menos a uno de los fragmentos para poder reconstruir la tabla original a partir de los fragmentos. Los fragmentos se definen a través de una operación de *selección* y su reconstrucción se realizará en base a una operación de unión de los fragmentos componentes.

  En el ejemplo siguiente, se ilustra una posible fragmentación de la tabla Personas de dos fragmentos: uno para el nodo de la C y otro para el nodo de la V.

Código	Nombre	Apellido	CIRuc	Tipo
1	Juan	Perez	102030405	C
2	Luis	Noboa	203040506	C
1	Juan	Vendedor	506070809	V
4	Miguel	Cordero	405060708	C
3	Carlos	Vendedor	001020304	V
2	Pedro	Vendedor	708090100	V
3	Angel	Bravo	60708011	C
5	Pedro	Villa	304050607	C

$\sigma_{Tipo="C"}(T)$

Código	Nombre	Apellido	CIRuc	Tipo
1	Juan	Perez	102030405	C
2	Luis	Noboa	203040506	C
3	Angel	Bravo	60708011	C
4	Miguel	Cordero	405060708	C
5	Pedro	Villa	304050607	C

$\sigma_{Tipo="V"}(T)$

Código	Nombre	Apellido	CIRuc	Tipo
1	Juan	Vendedo	506070809	V
2	Pedro	Vendedor	708090100	V
3	Carlos	Vendedor	001020304	V

[1] Tomado del documento Diseño de Base de Datos Distribuidas, UNIVERSIDAD CARLOS III.

CAPÍTULO 2

CONCEPTOS DE LA HERRAMIENTA CASE GENEXUS

Introducción:

Para este Capítulo se presentarán los conceptos con el objeto de entender que es una Herramienta CASE, las ventajas que su uso nos ofrece y, de manera especial, nos enfocamos en la utilización del CASE Genexus. Se incluyen aspectos como: Genexus versión 8.0, metodología incremental, y elementos de Genexus 8.0.

En un principio abordamos el tema Genexus versión 8.0 en el cual determinamos los diferentes generadores que proporciona y en que plataformas puede ser utilizada esta herramienta.
En Metodología Incremental realizamos una breve análisis de la metodología tradicional versus la metodología incremental y que beneficios nos proporcionan.
Finalmente veremos los Objetos Genexus que la versión 8.0 nos ofrece, y cómo y para qué pueden ser utilizados.

2.1 Conceptos de la Herramienta CASE Genexus

2.1.1 Introducción

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información. Desde que se crearon éstas herramientas (1984) hasta la actualidad, las CASE cuentan con una credibilidad y exactitud que tienen un reconocimiento universal, siendo usadas por cualquier desarrollador y/o programador que busca un resultado óptimo y eficiente, pero sobre todo que busca esa minuciosidad necesaria de los procesos y entre los procesos.

Herramienta CASE (Computer-Aided Software Engineering).- De acuerdo con Kendall & Kendall[1] la ingeniería asistida por ordenador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.
Estas herramientas están desarrollando una cultura de ingeniería nueva para muchas empresas, con el consiguiente beneficio de un incremento de su competencia en el mercado.

Las ventajas que nombraremos sobre estas herramientas versus los lenguajes de programación son las siguientes:

- Mayor productividad
- Menor tiempo de aprendizaje
- Aplicaciones generadas más estables y seguras
- Menor tiempo de implementación
- Reducción en costos de mantenimiento
- Mayor portabilidad en algunas herramientas
- Optimización en el uso de los lenguajes de programación

Debido a la gran demanda que tienen las CASE su exigencia en cuanto a su uso ha ido aumentando, por lo que toda CASE debe entre otras cosas:

a) Proporcionar topologías de aplicación flexibles
b) Proporcionar aplicaciones portátiles
c) Brindar un control de versión
d) Crear código compilado en el servidor
e) Dar un soporte multiusuario
f) Ofrecer seguridad

Tecnología CASE

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información, y se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones
- Simplificar el mantenimiento de los programas
- Mejorar y estandarizar la documentación
- Aumentar la portabilidad de las aplicaciones
- Facilitar la reutilización de componentes de software

[1] KENDALL & KENDALL, Análisis y Diseño de Sistemas,1997, Tercera edición.

- Permitir un desarrollo y un rendimiento visual de las aplicaciones, mediante la utilización de gráficos.

Componentes de una herramienta CASE

De una forma esquemática podemos decir que una herramienta CASE se compone de los siguientes elementos:

- Repositorio (diccionario) donde se almacena los elementos definidos o creados por la herramienta, y cuya gestión se realiza mediante el apoyo de un Sistema de Gestión de Base de Datos (SGBD) o de un sistema de gestión de ficheros.
- Metamodelo (no siempre visible), que constituye el marco para la definición de las técnicas y metodologías soportadas por la herramienta.
- Carga o descarga de datos, son facilidades que permiten cargar el repertorio de la herramienta CASE con datos provenientes de otros sistemas. Este elemento proporciona un medio de comunicación con otras herramientas.
- Comprobación de errores, facilidades que permiten llevar a cabo un análisis de la exactitud, integridad y consistencia de los esquemas generados por la herramienta.
- Interfaz de usuario, que constará de editores de texto y herramientas de diseño gráfico que permiten, mediante la utilización de un sistema de ventanas, iconos y menús, con la ayuda del ratón, definir los diagramas, matrices, etc. que incluyen las distintas metodologías.

La herramienta ***CASE Genexus*** es una herramienta para el desarrollo de aplicaciones de bases de datos, incluyendo a las bases de datos distribuidas. La facilidad que proporciona ésta herramienta a sus desarrolladores hace que los proyectos sean implementados de una manera más rápida y sencilla, de manera que el esfuerzo sea mucho menor, mejor orientado y más confiable.

Según Toledo y Sánchez[1], los ambientes, idiomas y arquitecturas actualmente soportadas por Genexus son:

- Multiplataforma:
    - Servidores con Sistemas Operativos: OS/400, Unix, Linux, Microsoft Windows Server.
    - Bases de Datos: IBM DB2UDB, Oracle, Informix, Microsoft SQL Server, PostgreSql.
    - Lenguajes de programación: Java, C#, Visual Basic, Visual FoxPro, C/SQL, RPG, Cobol, etc.
    - Internet: Java, HTML, C#, Visual Basic (ASP), C/SQL.
    - Servidores Web: Apache, WebSphere, Microsoft IIS.
    - Servidores de Aplicciones: Tomcat y WebSphere, Jrun, Resin.

- Multiarquitectura de aplicaciones:
    - Centralizada, genera en la plataforma iSeries AS/400 y en los lenguajes RPG y Cobol.
    - Cliente/Servidor (dos y tres capas), genera para plataformas cliente en Windows con Visual Basic, Visual FOXPRO, C, Java, y Csharp, adicional a esto para plataformas de servidores en: iSeries AS/400, Windows Server,

[1] Demetrio TOLEDO y Henry SÁNCHEZ, Crear un generador de programas para la herramienta CASE Genexus, 2004.

Linux, Unix, con los siguientes motores de Base de Datos DB2UDB, Oracle, Informix, SQL Server y PostgreSQL.
- Web, genera aplicaciones con Java, Csharp, Visual Basic y C, con los Motores de Base de Datos descritos anteriormente.

➢ Las nuevas plataformas de ejecución
- Java y Microsoft.NET

2.1.2 Genexus versión 8.0[1]

Según la empresa ARTech la versión 8.0 de Genexus hace más rápida y fácil que nunca la construcción e integración de aplicaciones críticas del negocio en múltiples plataformas.

Genexus se supera a sí misma en lo que hace única a la herramienta, al agregar mayor productividad en el desarrollo, y aún más inteligencia en el modelado de aplicaciones en la versión 8.0. Además, Genexus 8.0 simplifica el uso de Web Services para integrar diferentes sistemas, incorpora generadores para nuevas plataformas (Pocket PC), así como importantes funcionalidades para el desarrollo de aplicaciones Web.

Más fácil integrar aplicaciones integrando Web Services
Genexus 8.0 simplifica en forma notable el consumo y la producción de Web Services que actualmente se utilizan para integrar aplicaciones tanto dentro de la empresa, como en distintas empresas.

Desarrollo Web más potente
Genexus 8.0 facilita la migración de aplicaciones GUI (Graphic User Interfase) a Web, así como el desarrollo y mantenimiento de aplicaciones Web, al incorporar importantes funcionalidades que se traducen en mayor productividad en el desarrollo y menor costo de mantenimiento.

Modelado más inteligente
Para facilitar al desarrollador el manejo de aplicaciones a medida que transcurre el tiempo y se requiere incorporar nuevas funcionalidades, Genexus 8.0 incluye importantes mejoras en algo en lo que ya era bueno: disminuir al máximo la necesidad de escribir código para representar la realidad.

Mayor productividad en el desarrollo
Además de hacer aún más productivo el desarrollo, Genexus 8.0 mejora la usabilidad de la herramienta facilitando el trabajo tanto a los usuarios nuevos como a los expertos.

Soporte a más generadores y plataformas
Genexus 8.0 continúa la evolución tecnológica con su propuesta multiplataforma. Esta versión permite generar aplicaciones GUI o Web para las plataformas Microsoft.Net y Java, así como también incorpora un generador para dispositivos Pocket PC.
Se incluye oficialmente como motor de base de datos a PostgreSql, para plataformas Linux y una futura versión está evaluando MySql.

---

[1] Publicación de la empresa ARTech, artículo: ARTech anuncia el lanzamiento de Genexus 8.0, página: http://www.genexus.com/portal/hgxpp001.aspx?2,9,105,O,S,0,PAG;CONC;72;2;D;21723;1;PAG;MNU;E;17;4;MNU, fecha de publicación: Octubre/2003, fecha de acceso: Marzo/2005

2.1.3 Metodología incremental

La ***forma tradicional de desarrollar aplicaciones***[1] parte de la premisa básica: es posible construir un modelo de datos estable de la empresa.

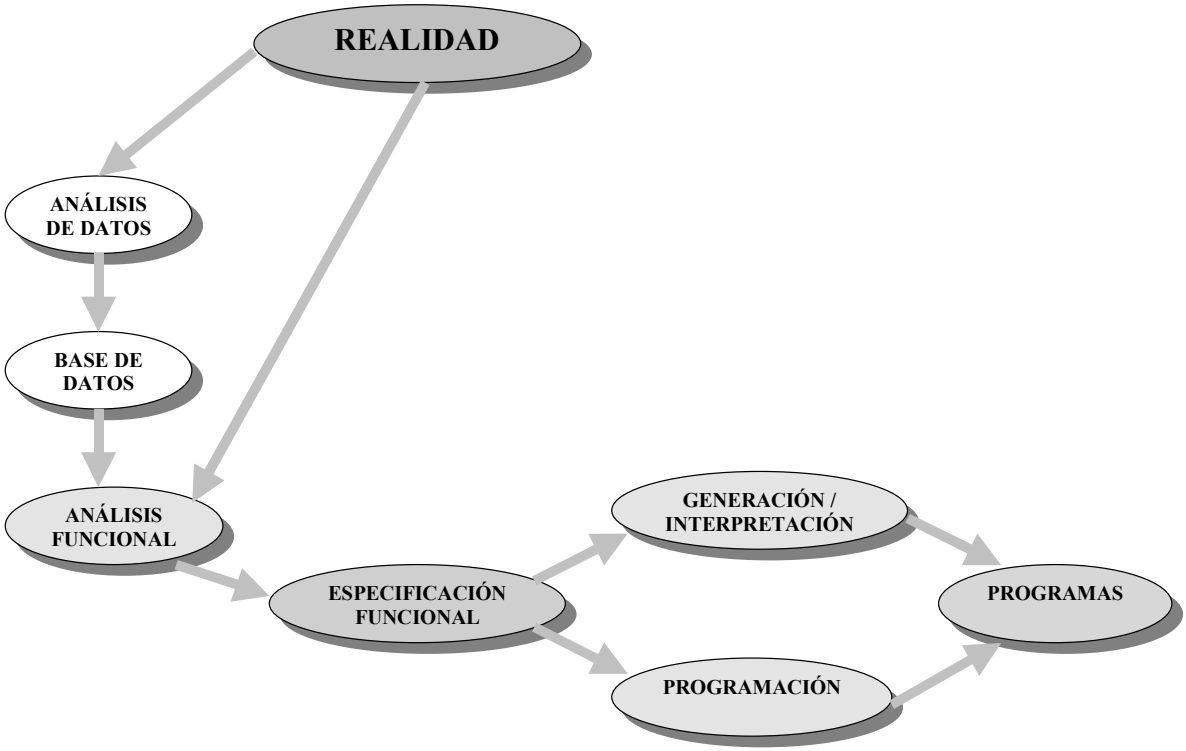


Figura 2-1 Esquema de la metodología tradicional[2].

Basándose en esta premisa, la primera tarea a encarar es el análisis de datos, donde se estudia la realidad en forma abstracta y se obtiene como producto el modelo de datos de la empresa. La segunda tarea es diseñar la base de datos. Es muy sencillo diseñar la base de datos partiendo del modelo de datos ya conocido.
Una vez que se ha estudiado la realidad desde el punto de vista de los datos, se hace lo propio desde el punto de vista de las funciones (análisis funcional), pasando a la implementación de las mismas.
Sin embargo, aún en el caso ideal, donde se conocen exactamente las necesidades organizacionales y, entonces, es posible definir la base de datos óptima, el modelo no podrá permanecer estático porque deberá acompañar la evolución de la empresa, por tanto decimos que la premisa en la que se basa la metodología tradicional, es falsa.

Una manera alternativa de resolver el problema pasa por la sustitución de la premisa básica enunciada: asumir que no es posible construir un modelo de datos estable de la empresa y, en cambio, utilizar una ***metodología incremental***[3]. Genexus implementa esta filosofía.
Cuando una aplicación se desarrolla con Genexus la primera etapa consiste en hacer el diseño de la misma registrando las visiones de usuarios (a partir de las cuales el sistema captura y sistematiza el conocimiento).

[1] Demetrio TOLEDO y Henry SÁNCHEZ, Tesis: Crear un generador de programas para la herramienta CASE Genexus, 2004.
[2] Figura tomada de la tesis: Crear un generador de programas para la herramienta CASE Genexus, 2004
[3] En un esquema incremental no se encaran grandes problemas, sino que se van resolviendo los pequeños problemas a medida que se presentan.

Posteriormente se pasa a la etapa de prototipación en donde Genexus genera la base de datos (estructura y datos) además de los programas para el ambiente de prototipo. Una vez generado el prototipo debe ser puesto a prueba por el analista y los usuarios.
Si durante la prueba del prototipo se detectan mejoras o errores se retorna a la fase de diseño, se realizan las modificaciones correspondientes y se vuelve al prototipo. Llegaremos a este ciclo de diseño/prototipo.
Una vez que el prototipo está aprobado, se pasa a la etapa de base de datos y programas para el ambiente de producción.

En resumen, una aplicación comienza con un diseño, luego se genera un prototipo, luego se implementa y en cualquiera de los pasos anteriores se puede regresar al diseño para realizar modificaciones.

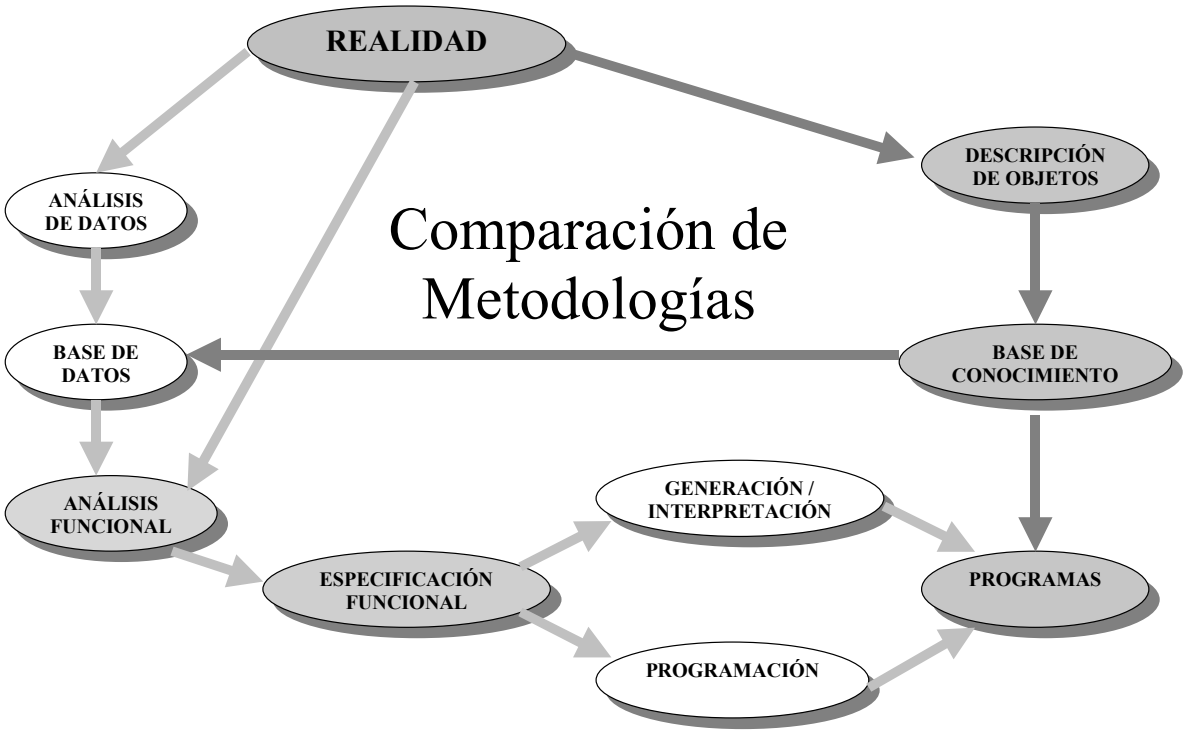


Figura 2-2 Esquema de la metodología tradicional versus la metodología Genexus[1].

2.1.4 Objetos Genexus

A continuación se describen los objetos Genexus[2] más importantes (no siendo los únicos):

- **Transacciones[3]**
  Permiten definir objetos de la realidad -reales o imaginarios- que el usuario manipula (Ejemplo: analistas de compras, artículos, proveedores, pedidos, etc.). Son los primeros objetos en definirse, ya que a través de las transacciones, Genexus infiere el diseño de la base de datos.

[1] Figura tomada de la tesis: Crear un generador de programas para la herramienta CASE Genexus, 2004
[2] Una Implementación del Desarrollo Incremental: Genexus, página: http://www.siscon.com.mx/genex5.asp, fecha de acceso: Mayo/2005
[3] No debe confundirse el lector con el término "transacción de base de datos" que se refiere al conjunto de operaciones sobre la base de datos que se ejecutan todas o ninguna. En Genexus ese concepto recibe el nombre de Unidad Transaccional Lógica (UTL).

CAPÍTULO 3

CONFIGURACIÓN DE SERVIDORES

Introducción:

En el desarrollo de este Capítulo se mostrará la forma de instalar y configurar los servidores necesarios para que funcione una base de datos distribuida.

Se ha seleccionado los sistemas operativos de servidor más conocidos en el mercado como son: Windows Server 2003 y Linux Red Hat 9.0, a la vez que Windows XP como estación de trabajo.
Primero realizaremos la instalación y configuración de los servidores en los cuales parametrizaremos los usuarios y seguridades requeridas para el acceso a las bases de datos.
Luego configuraremos la comunicación entre los diferentes servidores de manera que las bases de datos que usa cada uno de ellos tengan un acceso común.

3.1 Instalación y configuración de Servidores

3.1.1 Windows XP

Diseñado exclusivamente para la informática doméstica. Windows XP pone las experiencias más interesantes de la era digital a nuestro alcance. Desde fotografías, música y vídeo digital, hasta la creación de una red doméstica.

Windows XP ha sido creado sobre la sólida base de Windows 2000, esto agregado a las nuevas características de seguridad de Internet, se combinan con capacidades para compartir el equipo ofreciendo así el sistema operativo Windows más fiable hasta el momento.

3.1.1.1 Instalación y configuración

Figura 3-1 Inicialización de la instalación.

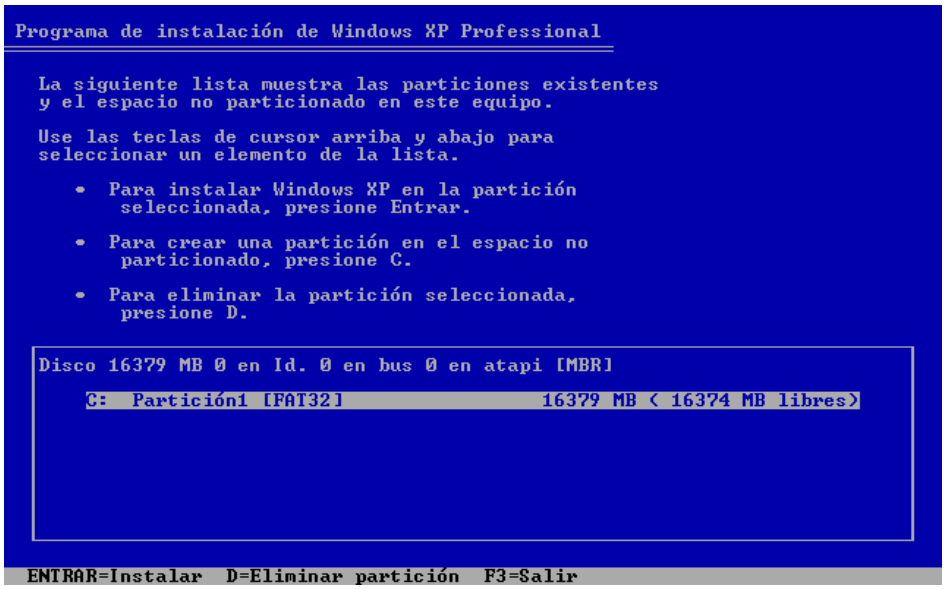


Figura 3-2 Selección de la partición en la que se instalará el sistema operativo.

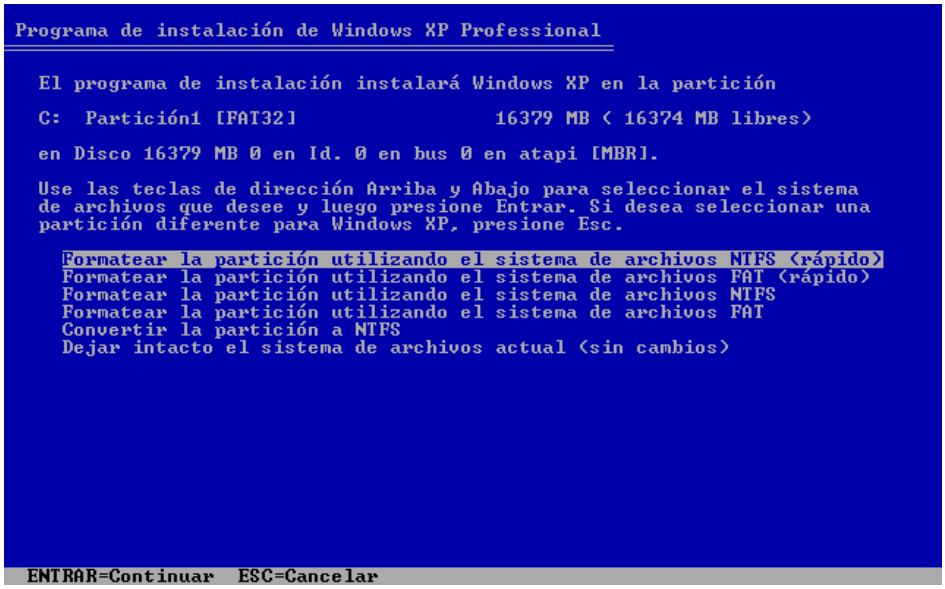


Figura 3-3 Sistema de archivos.

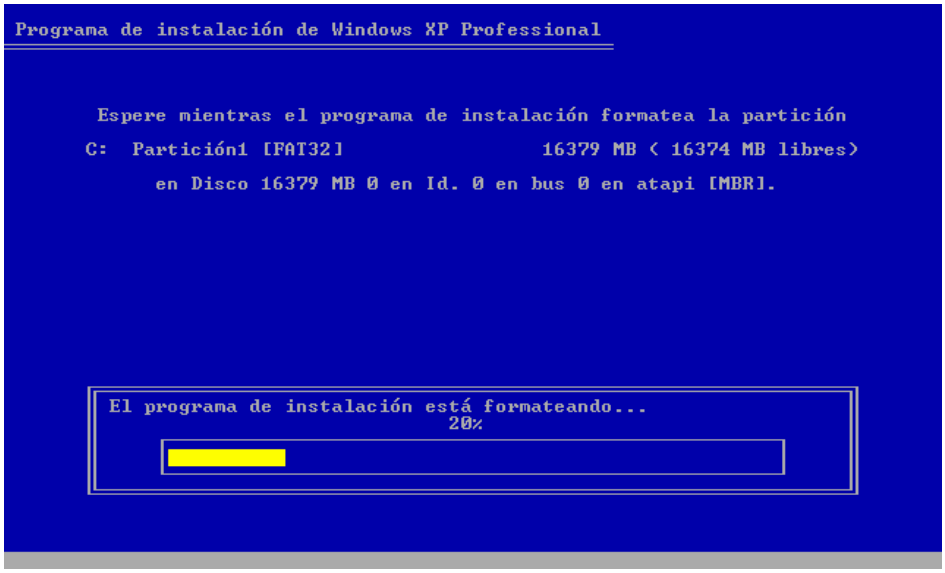


Figura 3-4 Formato en proceso de la partición escogida.

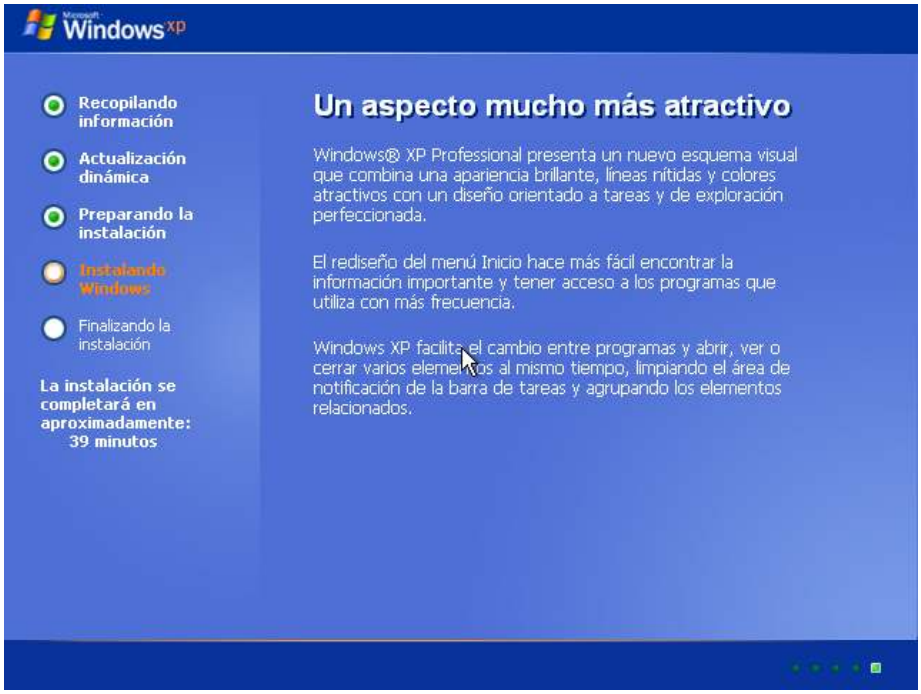


Figura 3-5 Instalación en transcurso.

Configuración regional y de idioma

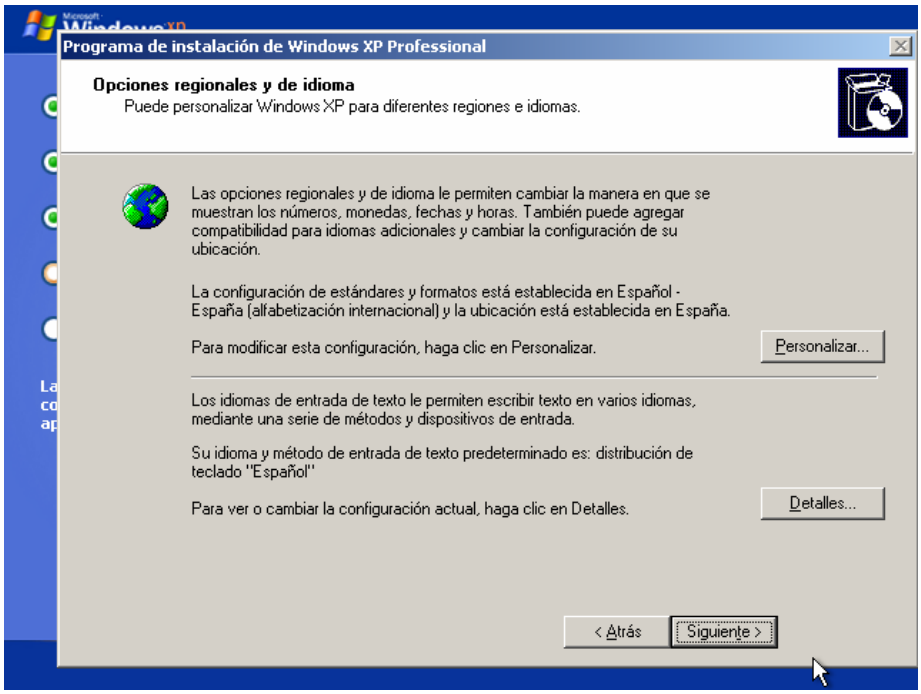


Figura 3-6 Configuración regional y de idioma.

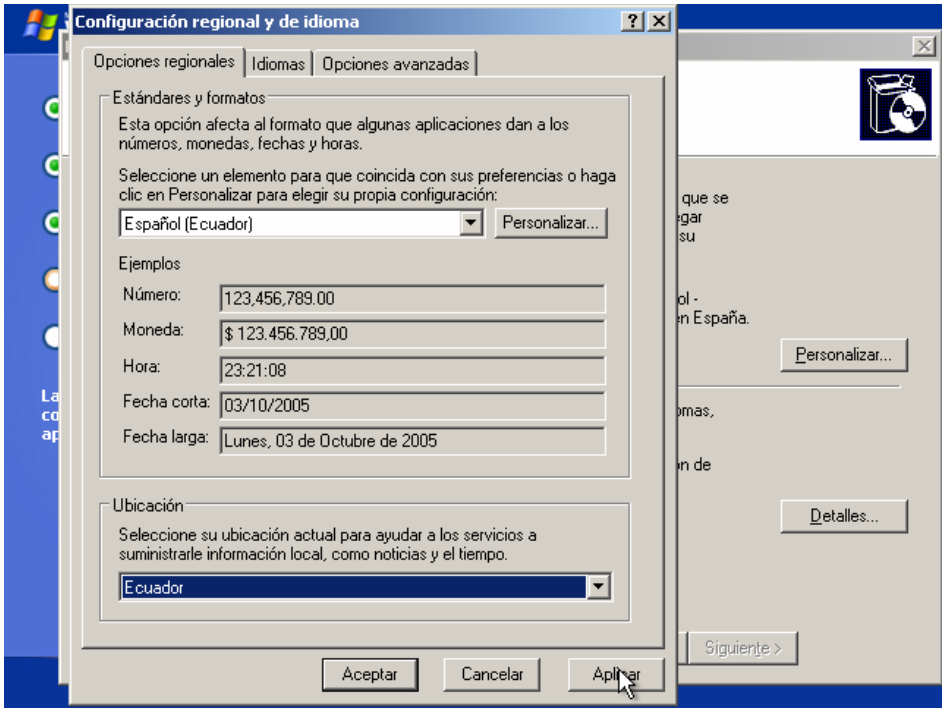


Figura 3-7 Opciones regionales.

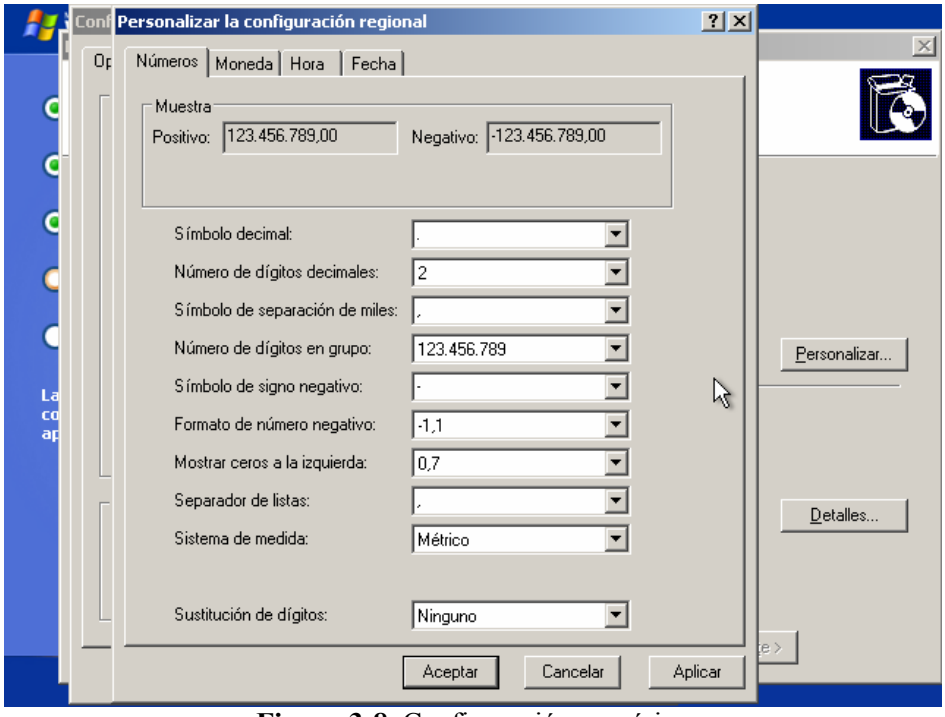


Figura 3-8 Configuración numérica.

Configuración de la fecha y hora

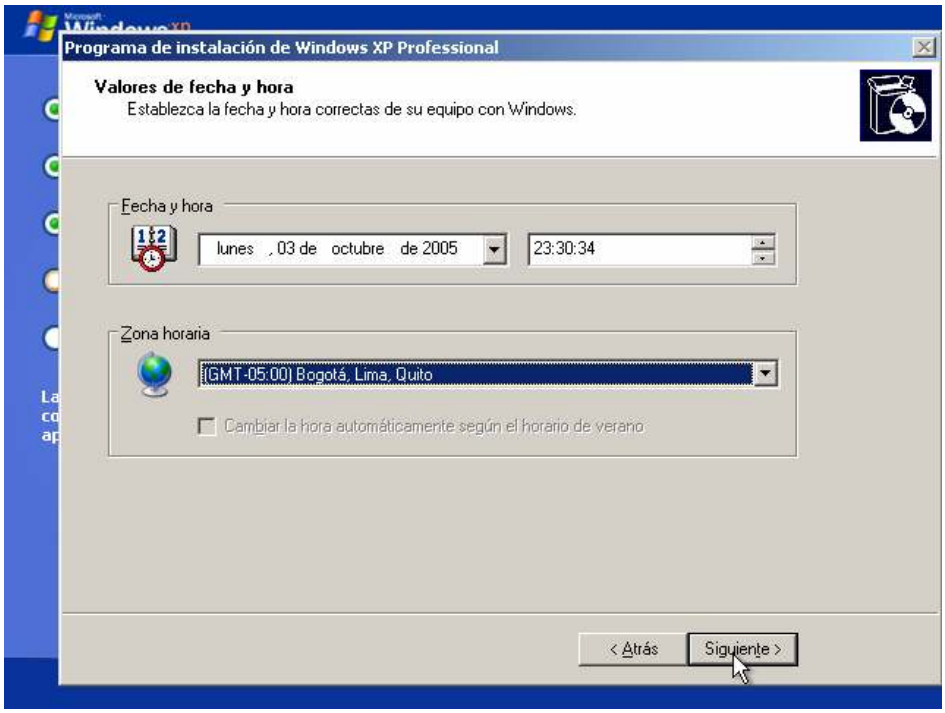


Figura 3-9 Configuración de fecha y hora del sistema.

Licencia del software

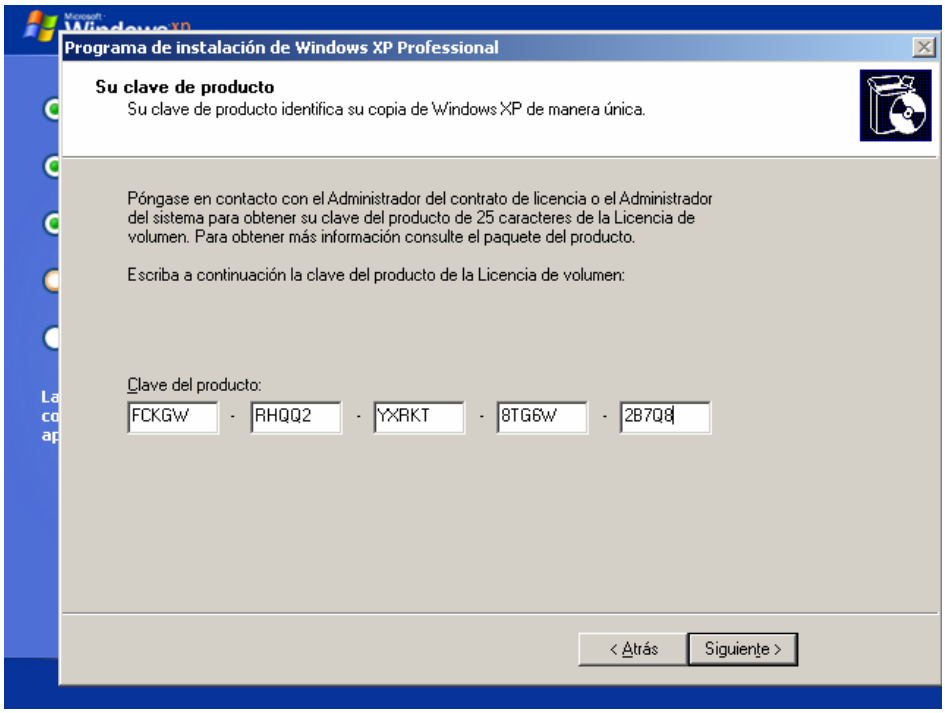


Figura 3-10 Licencia de instalación para Windows XP.

Nombre del equipo y contraseña del administrador

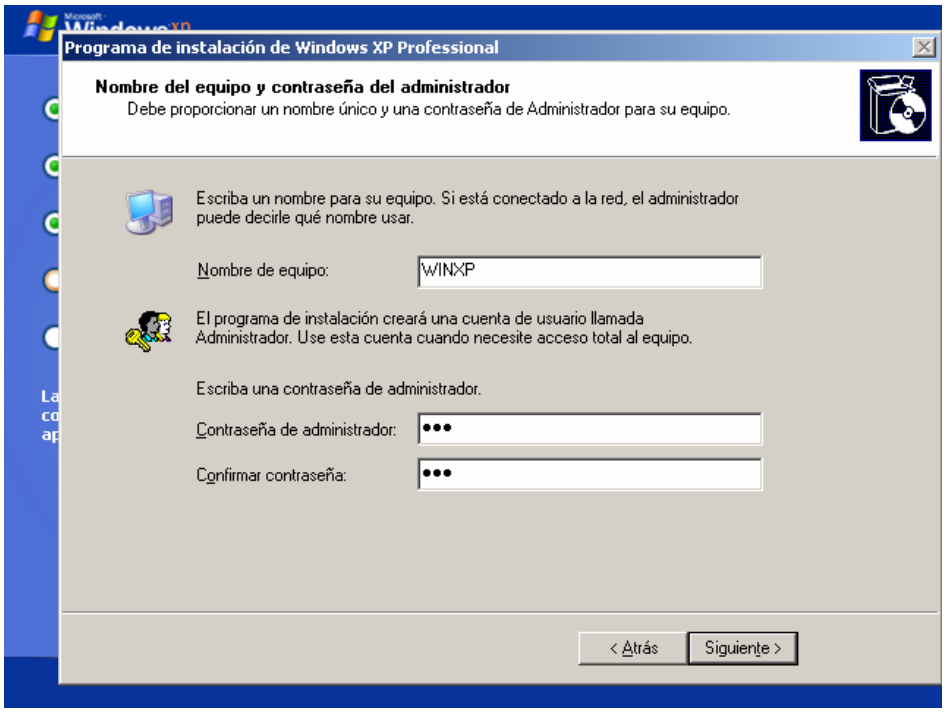


Figura 3-11 Datos del Administrador del equipo.

Configuración de la red

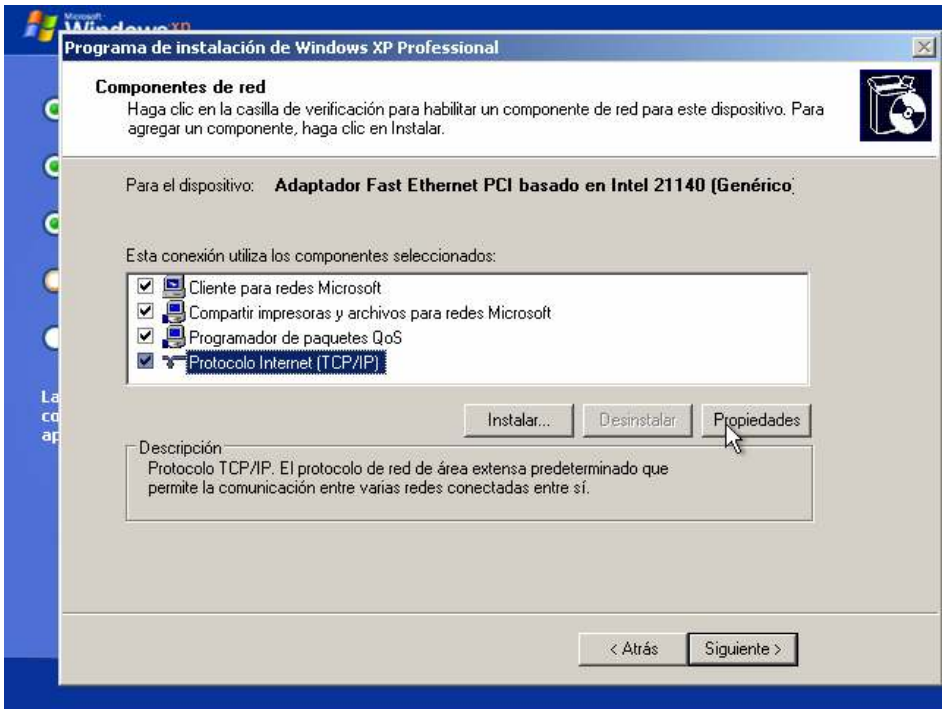


Figura 3-12 Componentes de red.

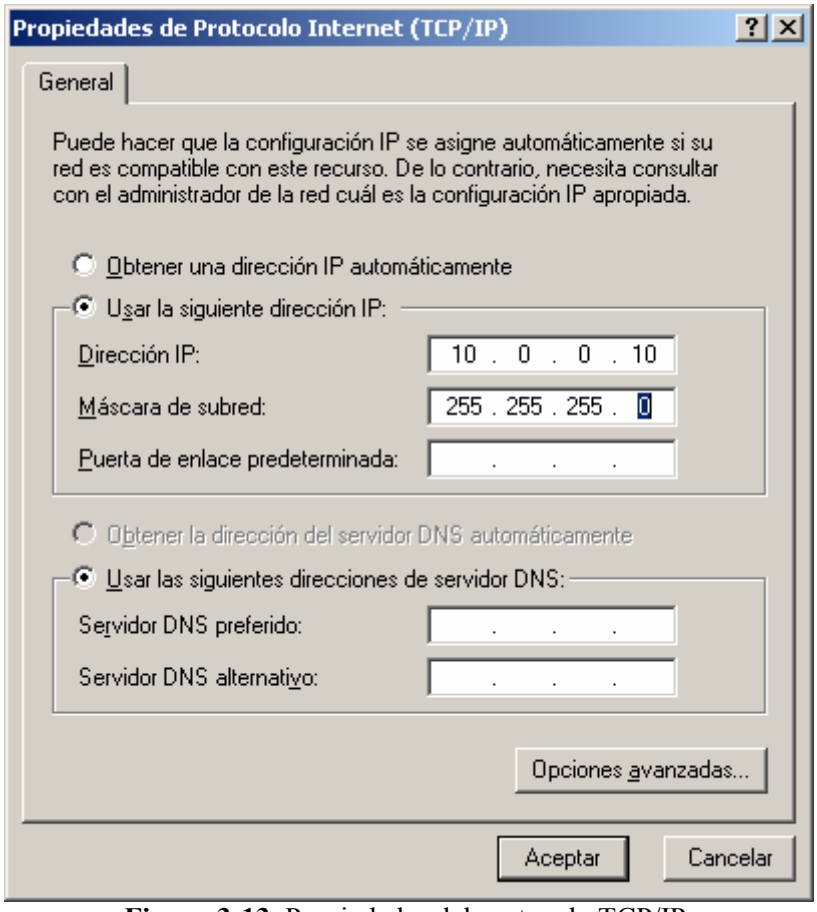


Figura 3-13 Propiedades del protocolo TCP/IP.

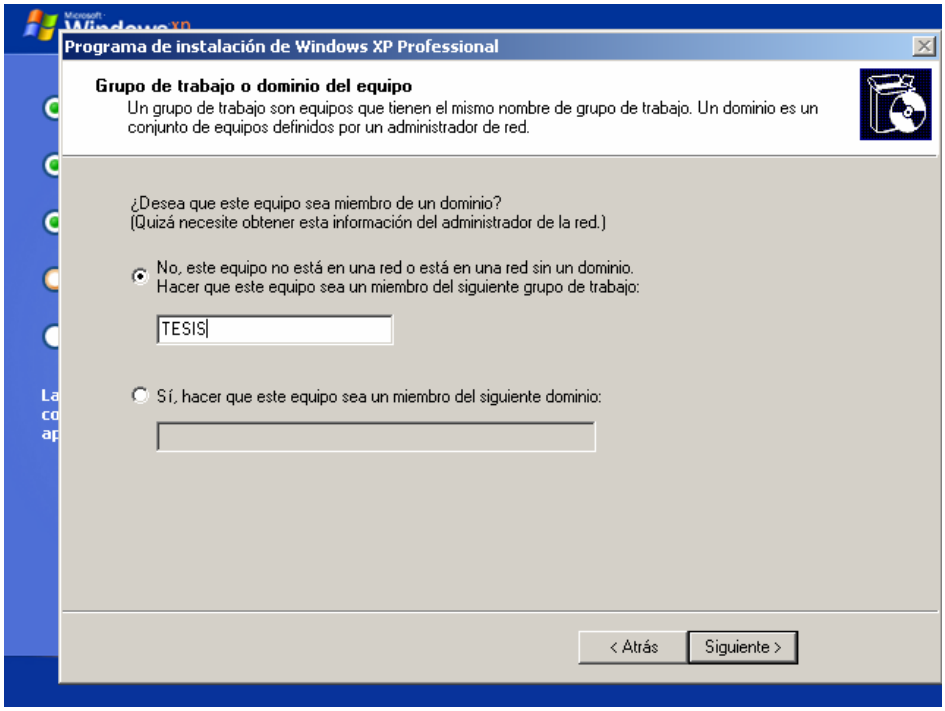


Figura 3-14 Denominación del grupo de trabajo.

Figura 3-15 Finalización de la instalación.

Usuarios del equipo

Figura 3-16 Denominación de los usuarios del equipo

Inicio de sesión

Figura 3-17 Inicio de sesión con Windows XP.

3.1.2 Windows Server 2003

Windows Server 2003 es un sistema operativo de propósitos múltiples, capaz de manejar una gran gama de funciones de servidor, en base a sus necesidades, tanto de manera centralizada como distribuida. Algunas de estas funciones del servidor son:

- Servidor de archivos e impresión.
- Servidor Web y aplicaciones Web.
- Servidor de correo.
- Terminal Server.
- Servidor de acceso remoto/red privada virtual (VPN).
- Servicio de directorio, Sistema de dominio (DNS), y servidor DHCP.
- Servidor de transmisión de multimedia en tiempo real (Streaming).
- Servidor de infraestructura para aplicaciones de negocios en línea (tales como planificación de recursos de una empresa y software de administración de relaciones con el cliente).

Entre los principales fundamentos[1] de Windows Server 2003 están los siguientes:

- Como servidor de ficheros es de un 100% a un 139% más rápido que Windows 2000 Server y un 200% más que Windows NT Server 4.0.
- Las características mejoradas del Directorio Activo permiten realizar tareas más fácilmente, entre las que destacan la habilidad de renombrar dominios, la posibilidad de redefinir el esquema y una replicación más eficiente.
- Ofrece la mejor conectividad, facilitando al máximo la configuración de enlaces entre delegaciones, acceso inalámbrico seguro y acceso remoto a aplicaciones a

1 Héctor GERSON y Oliva ULLOA, Microsoft Windows Server 2003, fecha de acceso el 20/Septiembre/2005 a la página: http://www.monografias.com

través de los Terminal Services, así como en su integración mejorada con dispositivos y aplicaciones.

3.1.2.1 Instalación y configuración

Figura 3-18 Inicialización de la instalación.

Figura 3-19 Selección de la partición en la que se instalará el sistema operativo.

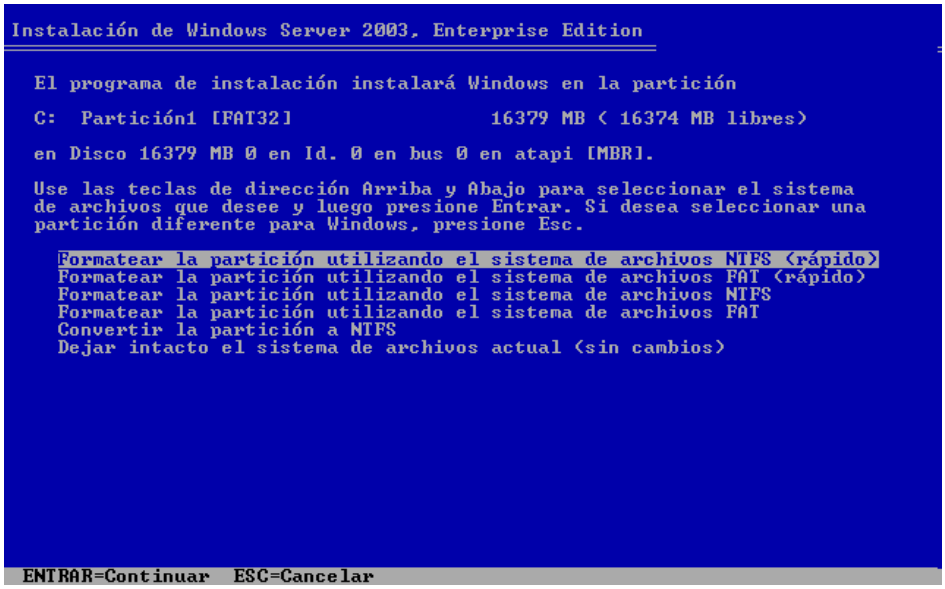


Figura 3-20 Sistema de archivos.

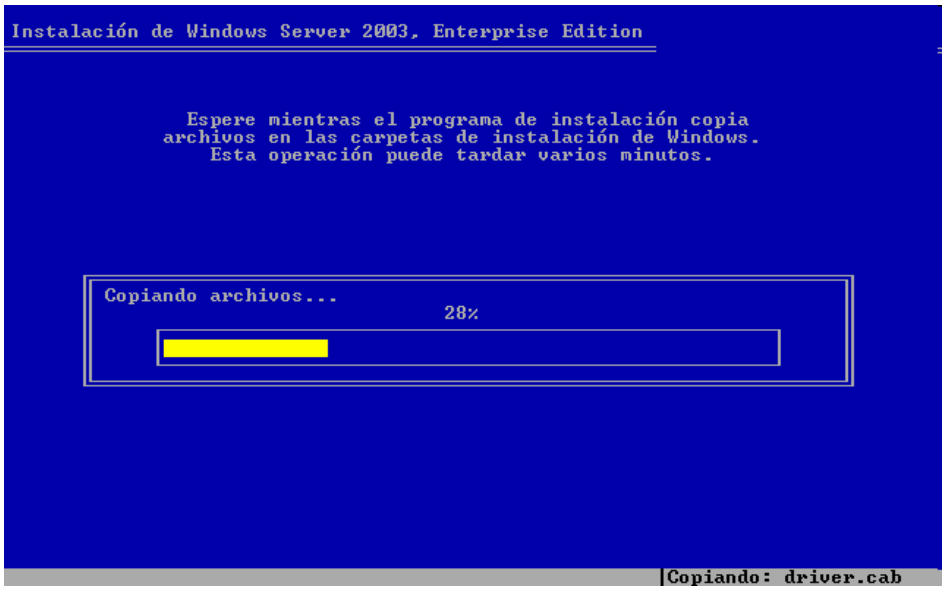


Figura 3-21 Formato en proceso de la partición escogida.

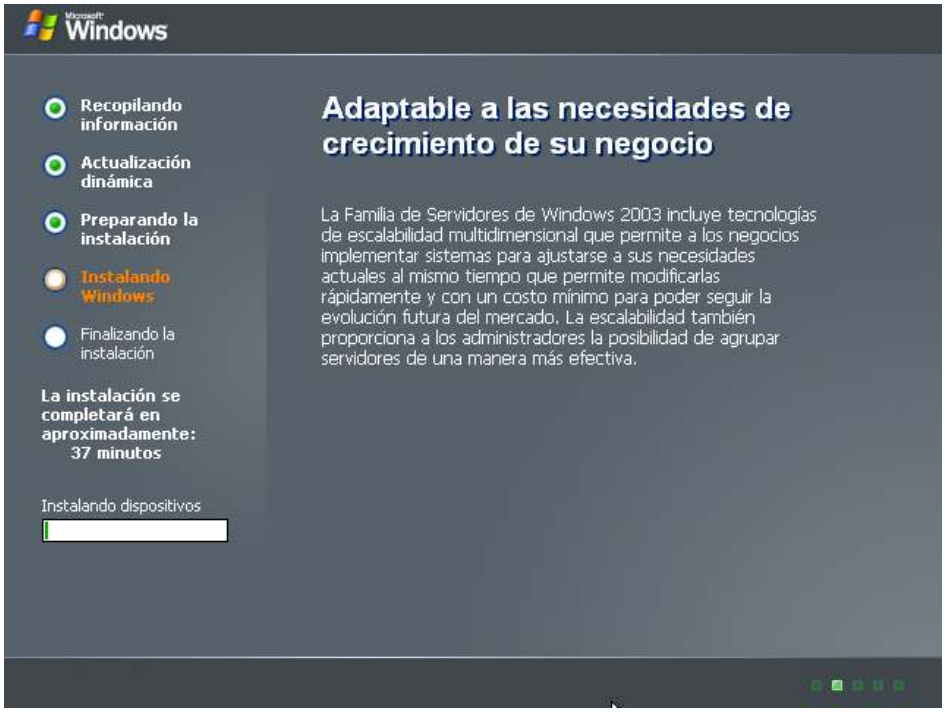


Figura 3-22 Instalación en transcurso.

Configuración regional y de idioma

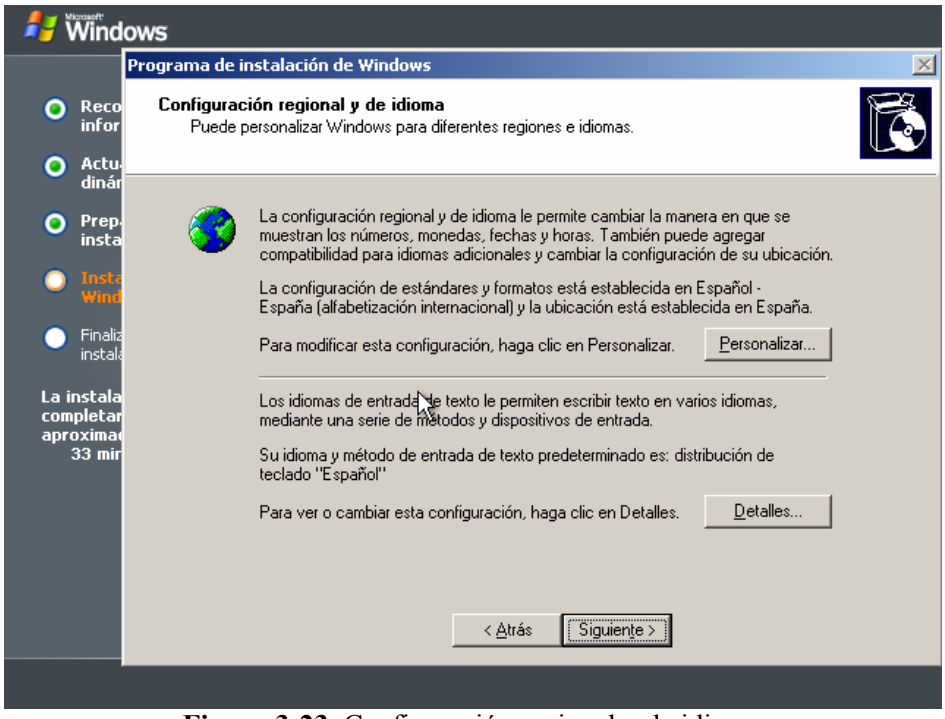


Figura 3-23 Configuración regional y de idioma.

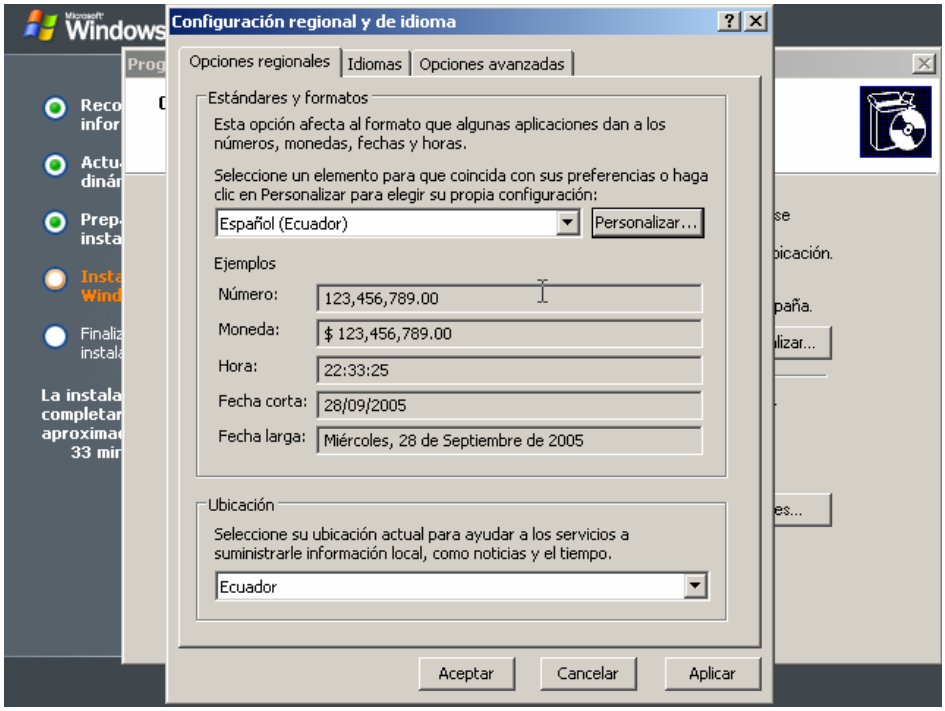


Figura 3-24 Opciones regionales.

Licencia del software

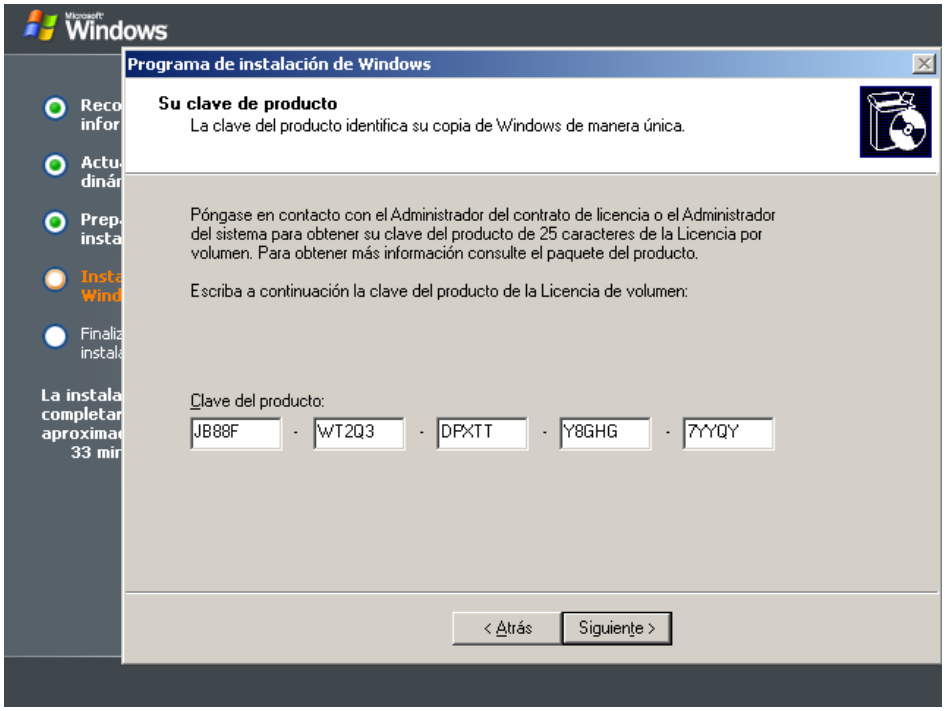


Figura 3-25 Licencia de instalación para Windows Server 2003.

Nombre del equipo y contraseña del administrador

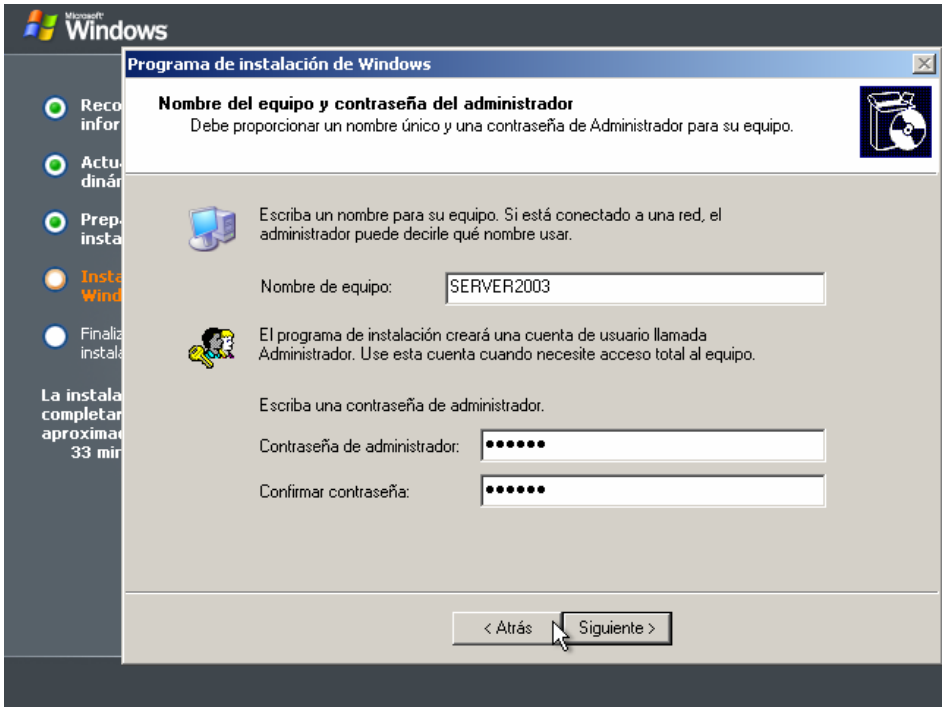


Figura 3-26 Clave del administrador del equipo.

Configuración de la fecha y hora

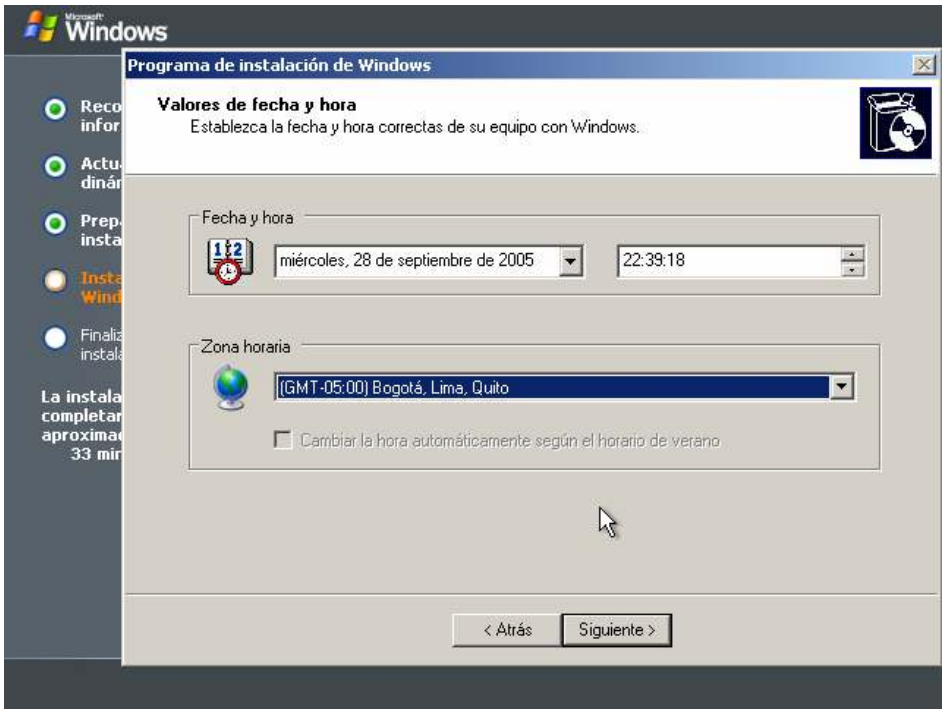


Figura 3-27 Configuración de fecha y hora del sistema.

CAPÍTULO 4

CONFIGURACIÓN DE GESTORES DE BASES DE DATOS

Introducción:

En el desarrollo de este Capítulo se mostrará la forma de instalar y configurar los Gestores de Bases de Datos SQL Server 2000 y PostgreSQL, adecuados para el funcionamiento de una base de datos distribuida.

Al comenzar este Capítulo daremos una breve descripción sobre el Gestor de Base de Datos SQL Server 2000, luego abordaremos el tema instalación y configuración, así también describiremos la replicación con este Gestor de Base de Datos.

Posteriormente veremos algunas ventajas que ofrece el Gestor de Base de Datos PostgreSQL, así como su forma de instalación y configuración.

4.1 Configuración de Gestores de Bases de Datos

4.1.1 Instalación y configuración de SQL Server 2000

SQL Server es un motor de base de datos cliente/servidor, lo cual implica que SQL Server ha sido diseñado para almacenar datos en un sitio central llamado servidor (pueden ser varios) y distribuirlos a otros sistemas llamados clientes. Éstos realizan requerimientos (consultas) al servidor, el cual los procesa y, luego, entrega los resultados (conjunto de registros) a los clientes que los solicitaron.

La ventaja de esta arquitectura es que sus requerimientos de hardware no son demasiado exigentes, aunque sí es conveniente poseer un equipamiento robusto del lado del servidor.

SQL Server, entre otras, tiene las siguientes características:

- **SQL Server:** este servicio es el motor o núcleo de las bases de datos y de todos los componentes del paquete SQL Server, siendo el único capaz de modificar datos. Además, administra los recursos entre los diferentes usuarios y es el encargado de interpretar las declaraciones SQL. También protege los datos y define las acciones que pueden realizar los clientes por medio de permisos. Algunas de ellas son:
  - Respetar las reglas de negocios de la organización por medio de disparadores (*triggers*) y procedimientos almacenados (*stored procedures*).
  - Evitar que dos usuarios intenten acceder al mismo dato simultáneamente.
  - Vigilar que los datos que se encuentren almacenados en distintas ubicaciones conserven cierta coherencia.
- SQL Server puede administrar cerca de un millón de terabytes, por lo que es muy poco probable que una empresa alcance ese límite.
- SQL Server cuenta con un mejor almacenamiento de datos, la posibilidad de generar múltiples disparadores (*triggers*) por cada tabla, que se utilizan para definir y validar las reglas de negocios, y, además, permite el bloqueo de registros por tabla.
- Además, una de las características más importantes de SQL Server es la posibilidad de crear réplicas. Esto implica que, ante la modificación de una de las copias, las demás también la reflejarán.

4.1.2 Instalación y configuración de SQL Server 2000

A continuación presentamos imágenes de la instalación de este software:

Figura 4-1 Servicios que ofrece el instalador.

Nombre del equipo

Como se puede apreciar en la figura 4-2 en la instalación se nos pide el nombre del equipo y la forma de instalación, si es local o remota, en caso de ser remota deberemos informar desde que equipo realizaremos la instalación.

Figura 4-2 Nombre del equipo y forma de la instalación.

Opciones de instalación

Existen tres opciones de instalación, como se puede observar en la figura 4-3:

- Crear una nueva instancia del SQL Server o instalar las herramientas clientes.
- Actualizar, quitar o agregar componentes a una instalación existente de SQL Server. Esta opción sólo estará activa si ya tenemos una instalación previa de este software y queremos modificarla.
- Opciones avanzadas, donde podremos crear instalaciones desatendidas o reconstruir el registro del SQL Server si estuviera dañado.

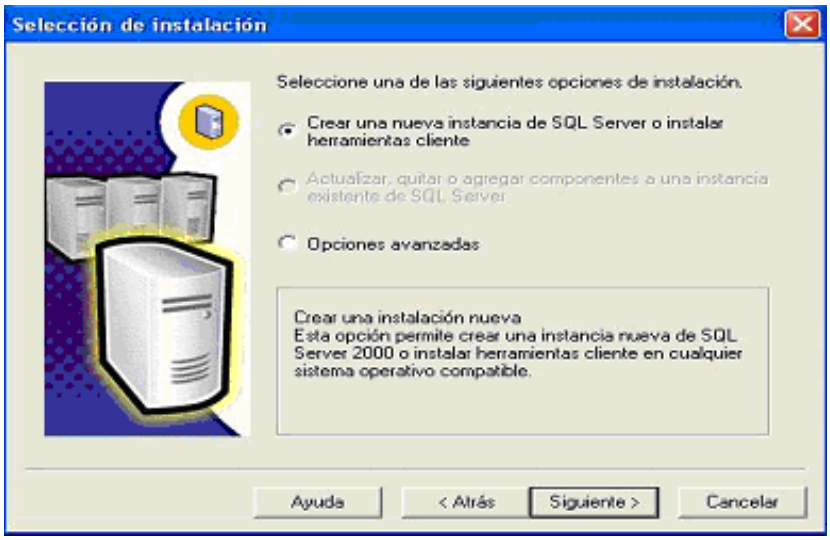


Figura 4-3 Tipos de instalación.

Herramientas de instalación

En la siguiente imagen se muestran los tres tipos de herramientas existentes:

1. Sólo herramientas cliente, nos permite tener acceso a un servidor remoto.
2. Herramientas de cliente y servidor, como gestor de bases de datos.
3. Sólo conectividad, que instala únicamente el MDAC[1].

Figura 4-4 Herramientas de instalación.

[1] Microsoft Data Access Components, es una arquitectura de Microsoft para proveer acceso a la información e involucra Active X Data Objects (ADO), OLE DB, y Open Database Connectivity (ODBC).

Tipos de instalación
Personalmente recomendamos una instalación personalizada y elegir la ruta de los datos a una unidad con suficiente espacio en disco.

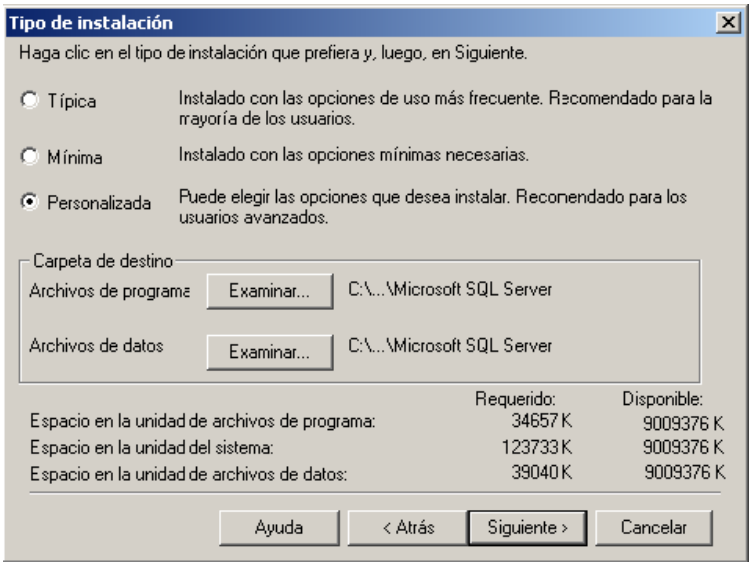


Figura 4-5 Tipos de instalación.

Selección de componentes
Selección de componentes a instalar, entre ellos ejemplos y ayudas (muy importante la ayuda, es el mejor manual de SQL Server).

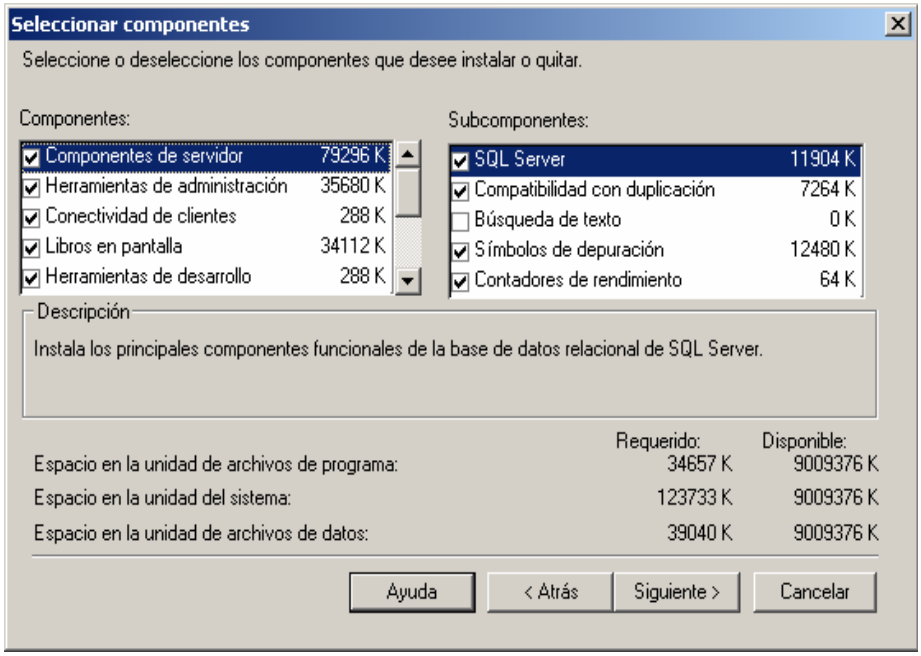


Figura 4-6 Componentes en una instalación personalizada.

CAPÍTULO 5

CONFIGURACIÓN DE LA HERRAMIENTA CASE GENEXUS

Introducción:

En el exposición de este Capítulo se mostrará como instalar y configurar la herramienta CASE Genexus 8.0, requerida para nuestra investigación.

En la parte inicial de este capítulo presentaremos los requisitos de hardware y software necesarios para la correcta instalación y funcionamiento de la Herramienta.
Luego presentaremos un listado de los gestores de bases de datos soportados por Genexus 8.0.
Finalmente expondremos todos los pasos de la instalación y configuración del Case Genexus versión 8.0.

5.1 Requisitos de Hardware y Software

5.1.1 Requerimientos de Hardware[1]

- Procesador: 500 MHz Intel Pentium
- Memoria: un mínimo de 64 MB de RAM (se recomienda 256 MB)
- Disco Duro: un mínimo de 50 MB de espacio libre en disco para instalar el modelador más un promedio de 10 MB para cada generador.
- Para crear aplicaciones Genexus se necesita espacio adicional o un disco compartido para crear las Bases de Conocimiento de las aplicaciones generadas.
- Video: 800 x 600 píxeles de resolución o superior, con 256 colores.

5.1.2 Requerimientos de Software[2]

- Microsoft Windows 98 o superior. Si se usa Windows NT debe instalarse el Service Pack 6.0 o superior.
- Paquete Redistribuible de Microsoft .NET Framework 1.1.
- Microsoft Internet Explorer 6.0 SP1 o superior.
- Microsoft SQL Server 2000 Desktop Engine o cualquiera de los DBMSs soportados por Genexus.

5.2.1 Gestores de Bases de Datos soportados

Para crear la base de datos de una aplicación y ejecutar dichas aplicaciones, requerirá uno de los siguientes DBMSs:

- DB2 UDB para iSeries
- DB2 Universal Database
- Informix
- Microsoft SQL Server
- Microsoft SQL Server Desktop Engine (MSDE)
- MySQL
- Oracle
- PostgreSQL

5.3.1 Instalación y Configuración de Genexus 8.0

5.3.1.1 Prerrequisitos de instalación

Antes de proceder la instalación de la herramienta Case Genexus 8.0 es necesario cumplir con ciertas condiciones previas como es la instalación de los siguientes componentes:

1. Internet Explorer 6 Service Pack 1, tal como se puede apreciar en la figura 5-1 se realiza la instalación de éste componente. Luego de haber culminado este proceso se deberá reiniciar el computador.

[1] Artech, Primeros pasos con Genexus 8.0, 2005
[2] Ibíd.

Figura 5-1 Prerrequisito, instalar Internet Explorer 6 Service Pack 1.

2. El siguiente paso es la instalación de Microsoft .NET Framework 1.1.

Figura 5-2 Instalación de Microsoft .NET Framework 1.1

3. El requisito final es la instalación de Microsoft Visual J# Sharp Redistributable Package 1.1, como se ve en la siguiente figura.

Figura 5-3 Instalación de Microsoft Visual J# .NET.

CAPÍTULO 6

DESARROLLO DE LA APLICACIÓN MODELO DE FACTURACIÓN USANDO LA HERRAMIENTA CASE GENEXUS

Introducción:

En este Capítulo presentaremos algunas imágenes del desarrollo de la aplicación modelo de facturación, utilizando la herramienta CASE Genexus.

6.1 Desarrollo de la aplicación

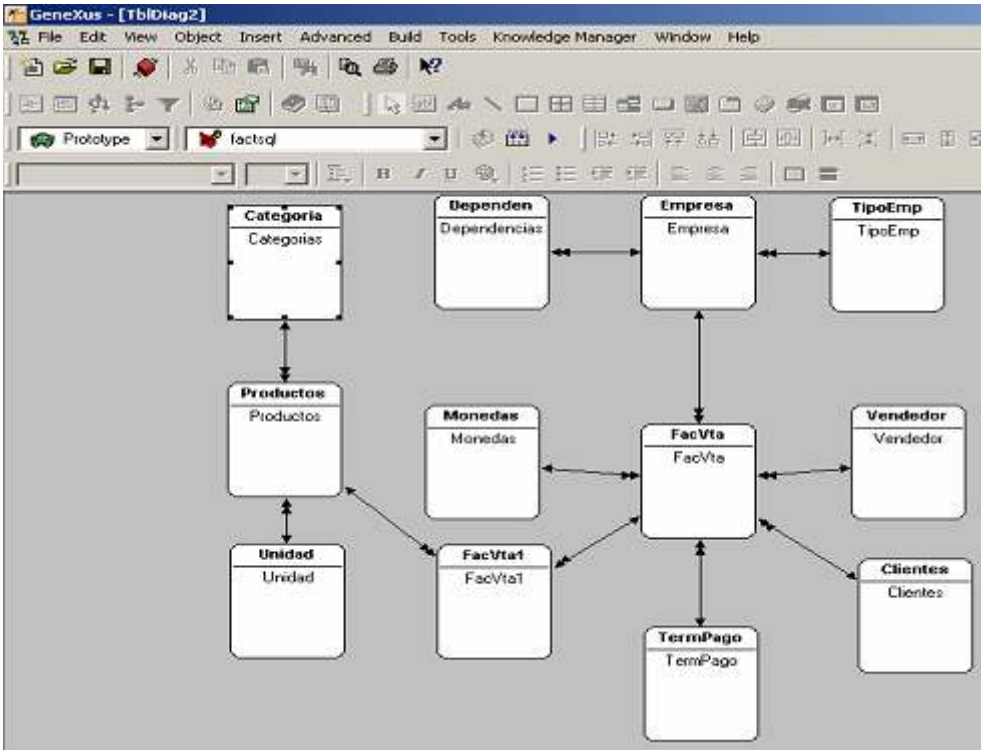


Figura 6-1 Modelo Entidad-Relación de nuestra aplicación.

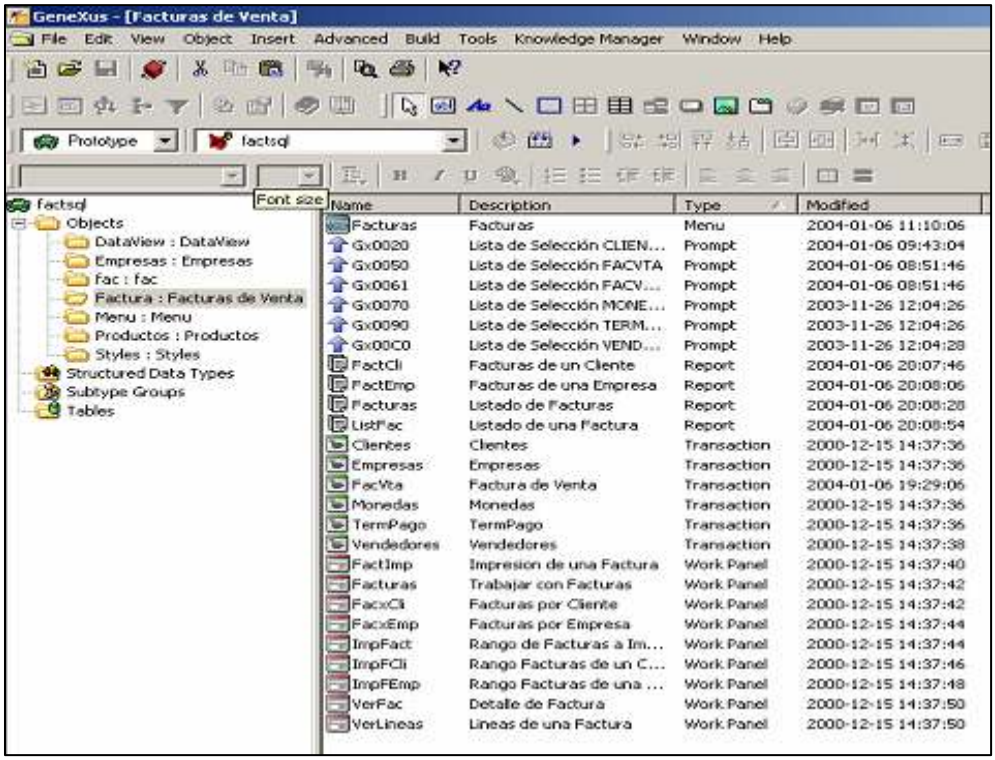


Figura 6-2 Prototipo de la aplicación.

CAPITULO 7

PARAMETRIZAR LAS BASES DE DATOS DISTRIBUIDAS DE LA APLICACIÓN DESARROLLADA EN GENEXUS

Introducción:

En este Capítulo se mostrará brevemente la forma de parametrizar las Bases de Datos en la aplicación desarrollada con Genexus 8.0

7.1 Parametrización de las Bases de Datos Distribuidas

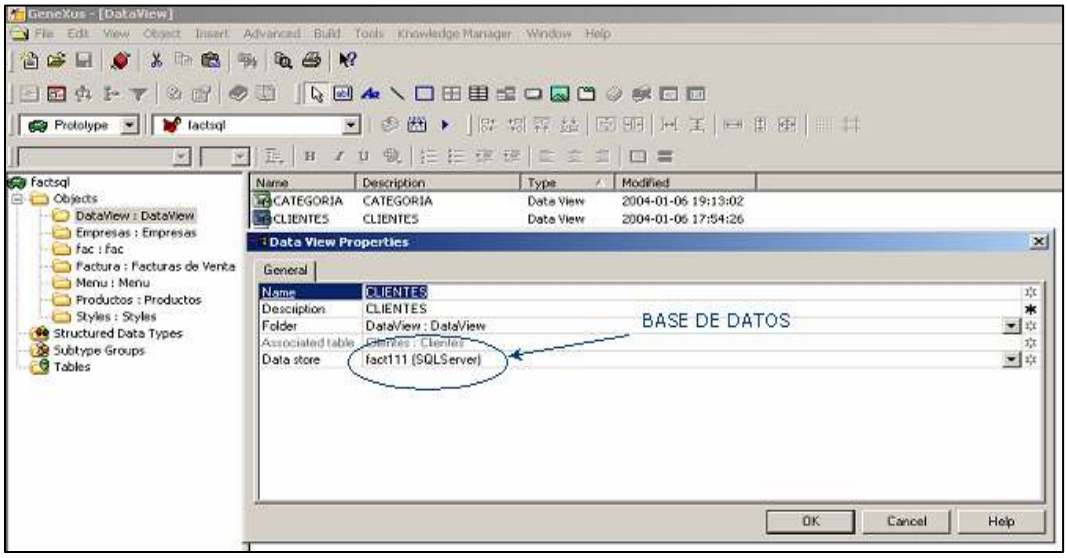


Figura 7-1 Propiedades de un DataView.

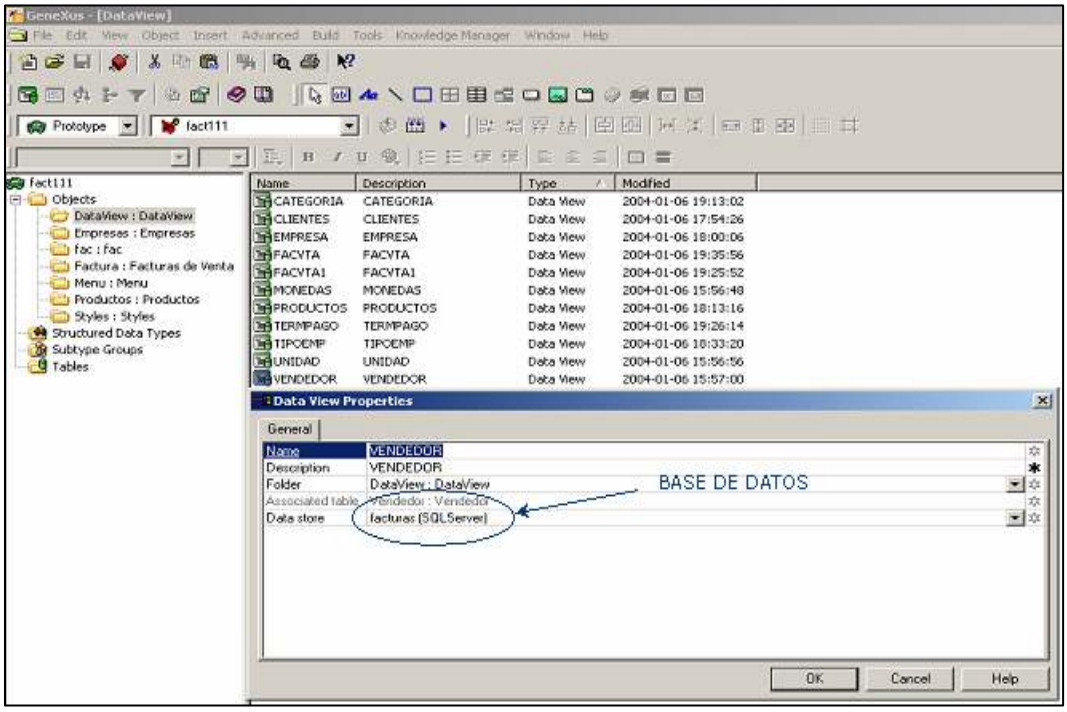


Figura 7-2 Propiedades de un DataView.

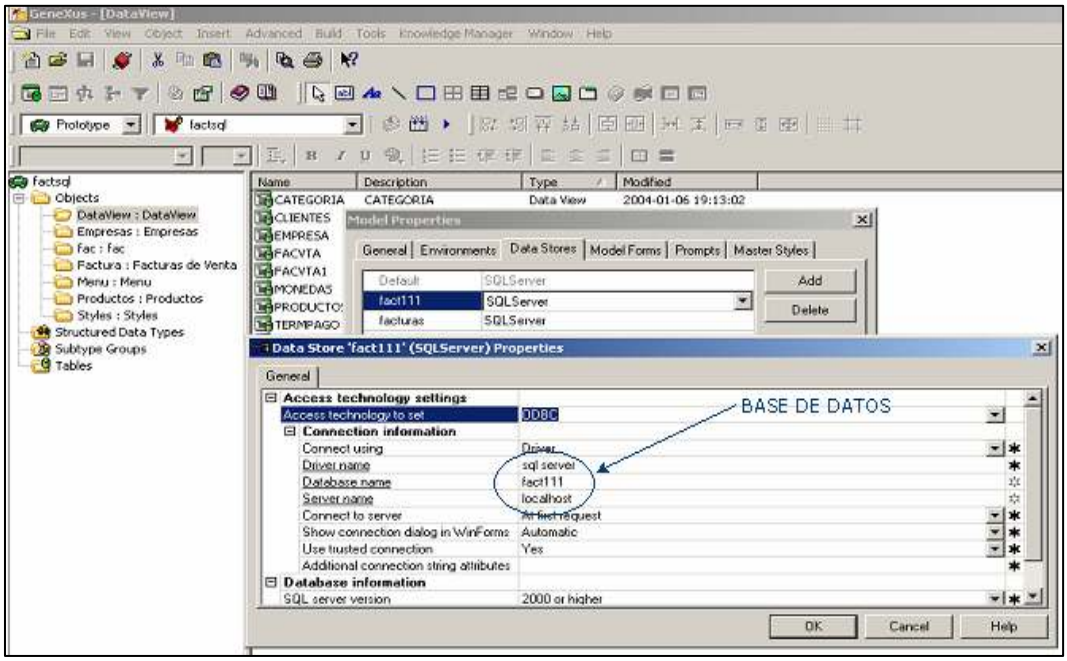


Figura 7-3 Propiedades de Data Store en el modelo.

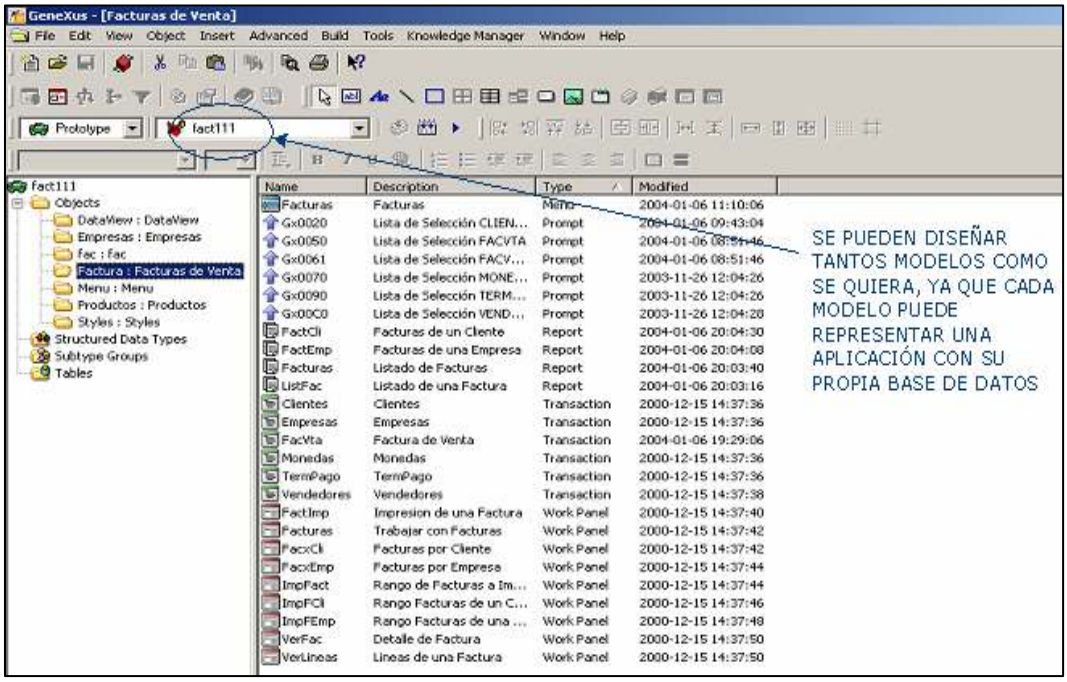


Figura 7-4 Diseño de modelos en Genexus.

CAPÍTULO 8

IMPLEMENTACIÓN DE LA APLICACIÓN

Introducción:

El presente Capítulo mostrará la implementación de la aplicación modelo de facturación con Bases de Datos Distribuidas utilizando la herramienta CASE Genexus 8.0.

8.1 Implementación

Para la implementación de la aplicación modelo de facturación, hemos utilizado:

- Dos Gestores de Bases de Datos SQL Server 2000
- Un Gestor de Base de Datos PostgreSQL
- Una red local de datos tipo C con topología tipo estrella
- Un Switch de 5 puertos
- Dos servidores con Windows Server 2003
- Un servidor con Red Hat Linux 9.0
- Y opcionalmente un equipo con Windows XP

Como se muestra en el siguiente gráfico SQL Server 2000 se ha configurado en los servidores Windows, y PostgreSQL en el servidor Linux.
La ejecución de la aplicación se puede hacer desde cualquiera de los equipos con sistema operativo Windows, ya que la aplicación generada en Genexus es para este ambiente.

La aplicación realizará el ingreso de datos para facturas de clientes, las mismas que utilizarán en forma transparente las tablas ubicadas en diferentes servidores. La consulta de los diferentes datos se ejecuta en forma transparente, de manera que el usuario no requiere saber la ubicación física de las Bases de Datos.

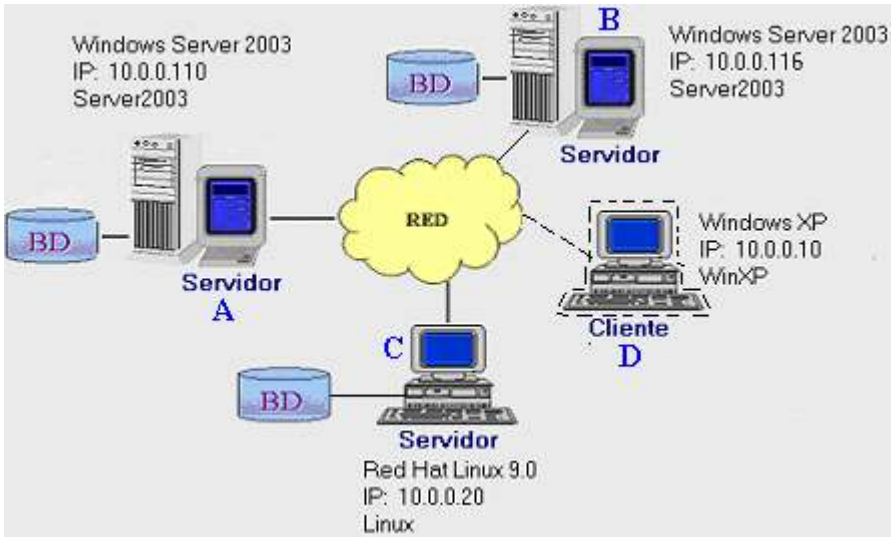


Figura 8-1 Configuración de la red para el proyecto.

Al ejecutar la aplicación obtendremos las siguientes pantallas

CONCLUSIONES GENERALES

Si partimos de que un ***Sistema de Bases de Datos Distribuidas*** es aquel cuya base de datos misma está distribuida, lógicamente interrelacionadas por medio de una red de computadoras, entonces un ***Sistema de Gestión de Base de Datos Distribuido*** (DBMS distribuido) provee garantías de apoyo de las transacciones distribuidas.

Compartir la información, brindar fiabilidad y disponibilidad, y agilizar el procesamiento de las consultas son varias razones para construir sistemas distribuidos de bases de datos, aunque existen varias desventajas como son mayores costes de desarrollo de software, mayor posibilidad de errores y el aumento en el coste extra del procesamiento.

La replicación de datos permite conservar varias copias idénticas de tablas que pueden o no estar en diferentes servidores, lo cual permite obtener disponibilidad de los datos en cualquier nodo, así como también de un mayor paralelismo de manera que se puedan hacer consultas simultáneas sobre una misma tabla.

Fragmentar los datos permite mejorar la unidad de distribución, ofrece menos transmisión por red, menos datos duplicados, mayor grado de concurrencia y ejecución en paralelo de consultas. Hay tres tipos de fragmentación: horizontal, vertical o mixta.

Por otro lado, los Objetos GeneXus, que facilitan obtener información de tablas externas (Data View), permiten diseñar una base de datos distribuida, ya que al manejar fuentes de datos diversas, por medio de ODBCs, definen la ubicación física de las tablas de manera transparente al usuario final.

Para la demostración práctica de ésta investigación se ha desarrollado una aplicación con la herramienta Case Genexus 8.0 en menor tiempo que lo que nos tomaría desarrollar en forma manual o tradicional, utilizando los sistemas operativos Windows XP, Windows Server 2003 y Red Hat Linux 9.0 con equipos en redes LAN.

Como Gestores de Bases de Datos empleamos SQL Server 2000 y PostgreSQL, observando un buen rendimiento como motores de bases de datos en aplicaciones **cliente/servidor,** y que además permiten la replicación de datos y son indicados para el manejo de bases de datos distribuidas.

El rendimiento de una aplicación podría verse afectado al cambiarse cualquiera de los parámetros de red, sistema operativo, gestores de Bases de Datos y la cantidad de datos procesados. Pero el concepto de las Bases de Datos Distribuidas utilizando Genexus como alternativa para el desarrollo de aplicaciones no cambia.

RECOMENDACIONES

Al concluir este proyecto podemos recomendar la utilización de la herramienta Case Genexus 8.0 para el desarrollo de aplicaciones Cliente/Servidor en ambientes Windows, pues permite aprovechar su metodología incremental, con lo cual se reduce significativamente el tiempo de desarrollo, implementación y mantenimiento.

Si los datos son requeridos por diferentes nodos, en diferentes localizaciones y con un medio de comunicación óptimo, es recomendable la implementación de Bases de Datos Distribuidas ya que éstas permiten compartir la información, brindan fiabilidad y disponibilidad, y agilizan el procesamiento de las consultas.

En cuanto a los Gestores de Bases de Datos, y de acuerdo a la experiencia obtenida en esta investigación, recomendaríamos el uso de SQL Server 2000 en sistemas operativos Windows y/o PostgreSQL en Linux, por su facilidad y poderío.