1

BASES DE DATOS DISTRIBUIDAS

TEMA 2

PROFESOR: M.C. ALEJANDRO GUTIÉRREZ DÍAZ

2

2. DISEÑO DE BASES DE DATOS DISTRIBUIDAS

2.1 Consideraciones al diseño de bases de datos distribuidas

2.2 Diccionario de datos

2.3 Niveles de transparencia

2.4 Fragmentación y distribución de datos

3

INTRODUCCIÓN

El diseño de un sistema de base de datos distribuido implica la toma de decisiones sobre la ubicación de los programas que accederán a la base de datos y sobre los propios datos que constituyen esta última, a lo largo de los diferentes puestos que configuren una red de ordenadores.

La ubicación de los programas, a priori, no debería suponer un excesivo problema dado que se puede tener una copia de ellos en cada máquina de la red (de hecho, en este documento se asumirá que así es).

Sin embargo, cuál es la mejor opción para colocar los datos: en una gran máquina que albergue a todos ellos, encargada de responder a todas las peticiones del resto de las estaciones – sistema de base de datos centralizado –, o podríamos pensar en repartir las relaciones, las tablas, por toda la red.

En el supuesto que nos decidiéramos por esta segunda opción, ¿qué criterios se deberían seguir para llevar a cabo tal distribución? ¿Realmente este enfoque ofrecerá un mayor rendimiento que el caso centralizado? ¿Podría optarse por alguna otra alternativa? En los

párrafos sucesivos se tratará de responder a estas cuestiones.
Tradicionalmente se ha clasificado la organización de los sistemas de bases de datos distribuidos sobre tres dimensiones: el nivel de compartición, las características de acceso a los datos y el nivel de conocimiento de esas características de acceso (vea la figura 1).
El nivel de compartición presenta tres alternativas: inexistencia, es decir, cada aplicación y sus datos se ejecutan en un ordenador con ausencia total de comunicación con otros programas u otros datos; se comparten sólo los datos y no los programas, en tal caso existe una réplica de las aplicaciones en cada máquina y los datos viajan por la red; y, se reparten datos y programas, dado un programa ubicado en un determinado sitio, éste puede solicitar un servicio a otro programa localizado en un segundo lugar, el cual podrá acceder a los datos situados en un tercer emplazamiento.
Como se comentó líneas atrás, en este caso se optará por el punto intermedio de compartición.

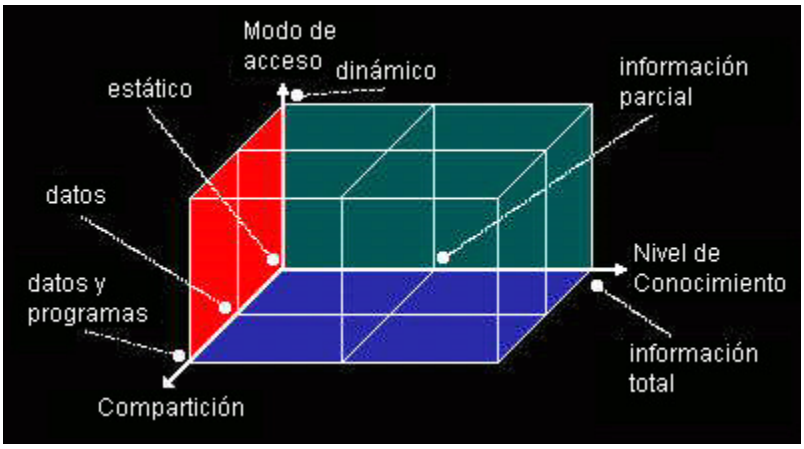

Figura 1. Enfoque de la distribución.
Respecto a las características de acceso a los datos existen dos alternativas principalmente: el modo de acceso a los datos que solicitan los usuarios puede ser estático, es decir, no cambiará a lo largo del tiempo, o bien, dinámico.
SE podrá comprender fácilmente la dificultad de encontrar sistemas distribuidos reales que puedan clasificarse como estáticos.
Sin embargo, lo realmente importante radica, estableciendo el dinamismo como base, cómo de dinámico es, cuántas variaciones sufre a lo largo del tiempo. Esta dimensión establece la relación entre el diseño de bases de datos distribuidas y el procesamiento de consultas.
La tercera clasificación es el nivel de conocimiento de las características de acceso. Una posibilidad es, evidentemente, que los diseñadores carezcan de información alguna sobre cómo los usuarios acceden a la base de datos.

Es una posibilidad teórica, pero sería muy laborioso abordar el diseño de la base de datos con tal ausencia de información. Lo más práctico sería conocer con detenimiento la forma de acceso de los usuarios o, en el caso de su imposibilidad, conformarnos con una información parcial de ésta.

El problema del diseño de bases de datos distribuidas podría enfocarse a través de esta trama de opciones. En todos los casos, excepto aquel en el que no existe compartición, aparecerán una serie de nuevos problemas que son irrelevantes en el caso centralizado.

A la hora de abordar el diseño de una base de datos distribuida podremos optar principalmente por dos tipos de estrategias: la estrategia ascendente y la estrategia descendente.

Ambos tipos no son excluyentes, y no resultaría extraño a la hora de abordar un trabajo real de diseño de una base de datos que se pudiesen emplear en diferentes etapas del proyecto una u otra estrategia.

La estrategia ascendente podría aplicarse en aquel caso donde haya que proceder a un diseño a partir de un número de pequeñas bases de datos existentes, con el fin de integrarlas en una sola. En este caso se partiría de los esquemas conceptuales locales y se trabajaría para llegar a conseguir el esquema conceptual global.

4

Aunque este caso se pueda presentar con facilidad en la vida real, se prefiere pensar en el caso donde se parte de cero y se avanza en el desarrollo del trabajo siguiendo la estrategia descendente. La estrategia descendente (vea la figura 2) debería resultar familiar a la persona que posea conocimientos sobre el diseño de bases de datos, exceptuando la fase del diseño de la distribución.

Pese a todo, se resumirán brevemente las etapas por las que se transcurre.

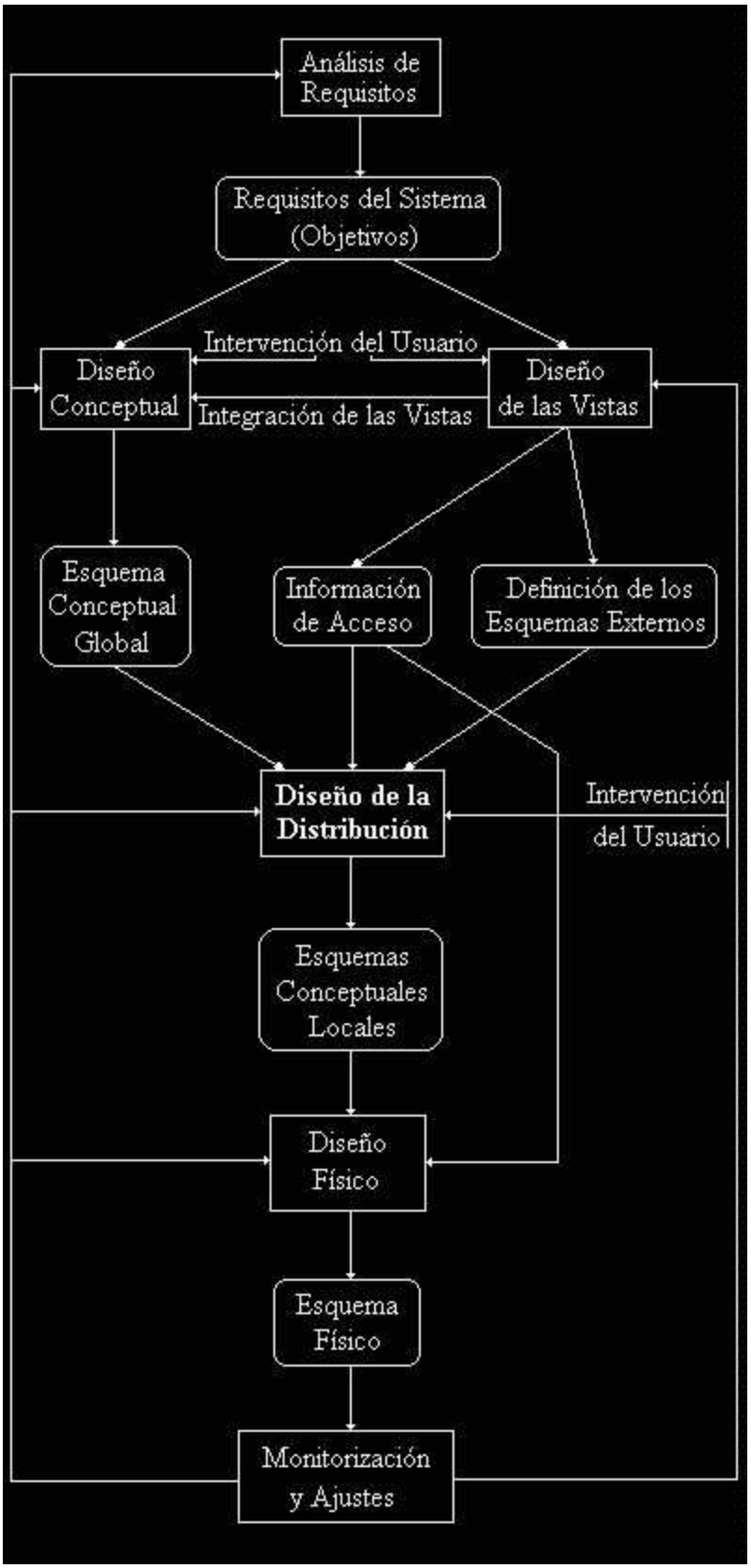

Figura 2. Estrategia descendente.

Todo comienza con un análisis de los requisitos que definirán el entorno del sistema en aras a obtener tanto los datos como las necesidades de procesamiento de todos los posibles usuarios del banco de datos. Igualmente, se deberán fijar los requisitos del sistema, los objetivos que debe cumplir respecto a unos grados de rendimiento, seguridad, disponibilidad y flexibilidad, sin olvidar el importante aspecto económico. Como puede observarse, los resultados de este último paso sirven de entrada para dos actividades que se realizan de forma paralela.
El diseño de las vistas trata de definir las interfaces para el usuario final y, por otro lado, el diseño conceptual se encarga de examinar la empresa para determinar los tipos de entidades y establecer la relación entre ellas. Existe un vínculo entre el diseño de las vistas y el diseño conceptual. El diseño conceptual puede interpretarse como la integración de las vistas del usuario, este aspecto es de vital importancia ya que el modelo conceptual debería soportar no sólo las aplicaciones existentes, sino que debería estar preparado para futuras aplicaciones.
En el diseño conceptual y de las vistas del usuario se especificarán las entidades de datos y se determinarán las aplicaciones que funcionarán sobre la base de datos, así mismo, se recopilarán datos estadísticos o estimaciones sobre la actividad de estas aplicaciones.
Dichas estimaciones deberían girar en torno a la frecuencia de acceso, por parte de una aplicación, a las distintas relaciones de las que hace uso, podría afinarse más anotando los atributos de la relación a la que accede.
Desarrollado el trabajo hasta aquí, se puede abordar la confección del esquema conceptual global. Este esquema y la información relativa al acceso a los datos sirven de entrada al paso distintivo: el diseño de la distribución.
El objetivo de esta etapa consiste en diseñar los esquemas conceptuales locales que se distribuirán a lo largo de todos los puestos del sistema distribuido.
Sería posible tratar cada entidad como una unidad de distribución; en el caso del modelo relacional, cada entidad se corresponde con una relación. Resulta bastante frecuente dividir cada relación en subrelaciones menores denominadas fragmentos que luego se ubican en uno u otro sitio.

De ahí, que el proceso del diseño de la distribución conste de dos actividades fundamentales: la fragmentación y la asignación. El último paso del diseño de la distribución es el diseño físico, el cual proyecta los esquemas conceptuales locales sobre los dispositivos de

almacenamiento físico disponibles en los distintos sitios.
Las entradas para este paso son los esquemas conceptuales locales y la información de acceso a los fragmentos. Por último, se sabe que la actividad de desarrollo y diseño es un tipo de proceso que necesita de una monitorización y un ajuste periódicos, para que si se llegan a producir desviaciones, se pueda retornar a alguna de las fases anteriores

5

2.1 Consideraciones al diseño de bases de datos distribuidas

El problema de diseño de bases de datos distribuidos se refiere, en general, a hacer decisiones acerca de la ubicación de *datos* y *programas* a través de los diferentes sitios de una red de computadoras. Este problema debería estar relacionado al diseño de la misma red de computadoras. Sin embargo, en estas notas únicamente el diseño de la base de datos se toma en cuenta. La decisión de donde colocar a las aplicaciones tiene que ver tanto con el software del SMBDD como con las aplicaciones que se van a ejecutar sobre la base de datos.
El diseño de las bases de datos centralizadas contempla los dos puntos siguientes:
1. Diseño del **"esquema conceptual"** el cual describe la base de datos integrada (esto es, todos los datos que son utilizados por las aplicaciones que tienen acceso a las bases de datos).
2. Diseño **"físico de la base de datos"**, esto es, mapear el esquema conceptual a las áreas de almacenamiento y determinar los métodos de acceso a las bases de datos.
En el caso de las bases de datos distribuidas se tienen que considerar los dos problemas siguientes:

3. Diseño de la **fragmentación**, este se determina por la forma en que las relaciones globales se subdividen en fragmentos horizontales, verticales o mixtos.
4. Diseño de la **asignación de los fragmentos**, esto se determina en la forma en que los fragmentos se mapean a las imágenes físicas, en esta forma, también se determina la solicitud de fragmentos.

6

Objetivos del Diseño de la Distribución de los Datos.
En el diseño de la distribución de los datos, se deben de tomar en cuenta los siguientes objetivos:
Procesamiento local. La distribución de los datos, para maximizar el procesamiento local corresponde al principio simple de colocar los datos tan cerca como sea posible de las aplicaciones que los

utilizan. Se puede realizar el diseño de la distribución de los datos para maximizar el procesamiento local agregando el número de referencias locales y remotas que le corresponden a cada fragmentación candidata y la localización del fragmento, que de esta forma se seleccione la mejor solución de ellas.

Distribución de la carga de trabajo. La distribución de la carga de trabajo sobre los sitios, es una característica importante de los sistemas de cómputo distribuidos. Esta distribución de la carga se realiza para tomar ventaja de las diferentes características (potenciales) o utilizaciones de las computadoras de cada sitio, y maximizar el grado de ejecución de paralelismo de las aplicaciones. Sin embargo, la distribución de la carga de trabajo podría afectar negativamente el procesamiento local deseado.

Costo de almacenamiento y disponibilidad. La distribución de la base de datos refleja el costo y disponibilidad del almacenamiento en diferentes sitios. Para esto, es posible tener sitios especializados en la red para el almacenamiento de datos. Sin embargo el costo de almacenamiento de datos no es tan relevante si éste se compara con el del CPU, I/O y costos de transmisión de las aplicaciones.

7 Existen dos estrategias generales para abordar el problema de diseño de bases de datos distribuidas:

1. El enfoque de arriba hacia abajo (top-down). Este enfoque es más apropiado para aplicaciones nuevas y para sistemas homogéneos. Consiste en partir desde el análisis de requerimientos para definir el diseño conceptual y las vistas de usuario. A partir de ellas se define un esquema conceptual global y los esquemas externos necesarios. Se prosigue con el diseño de la fragmentación de la base de datos, y de aquí se continúa con la localización de los fragmentos en los sitios, creando las imágenes físicas. Esta aproximación se completa ejecutando, en cada sitio, "el diseño físico" de los datos, que se localizan en éste. En la Figura 3.1 se presenta un diagrama con la estructura general del enfoque top-down.

2. **El diseño de abajo hacia arriba (bottom-up)**. Se utiliza particularmente a partir de bases de datos existentes, generando con esto bases de datos distribuidas. En forma resumida, el diseño bottom-up de una base de datos distribuida requiere de la selección de un modelo de bases de datos común para describir el esquema global de la base de datos. Esto se debe es posible que se utilicen diferentes SMBD. Después se hace la traducción de

cada esquema local en el modelo de datos común y finalmente se hace la integración del esquema local en un esquema global común.

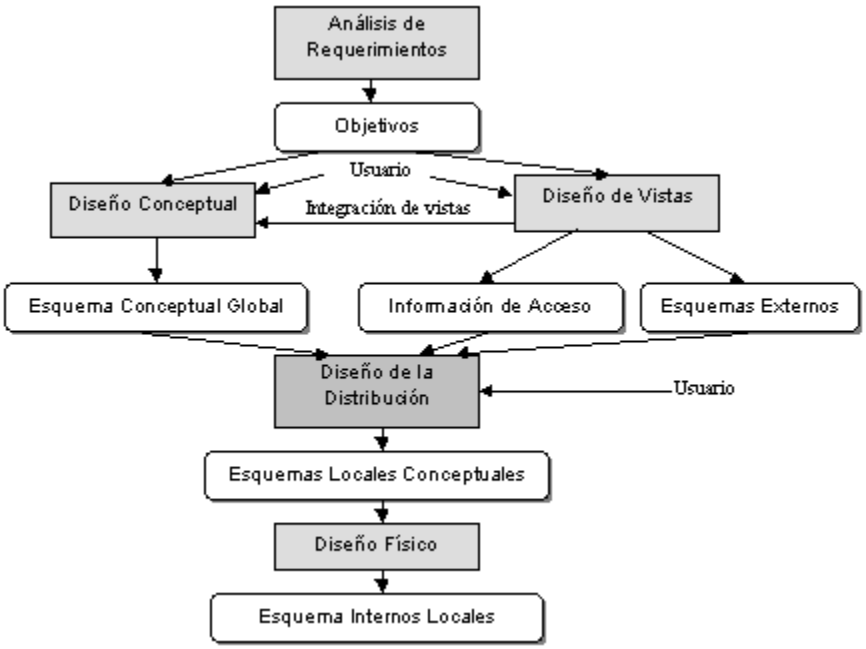

Figura 3.1. El enfoque top-down para el diseño de bases de datos distribuidas.

8

La clase de ***sistemas heterogéneos*** es aquella caracterizada por manejar diferentes SMBD en los nodos locales. Una subclase importante dentro de esta clase es la de los **sistemas de manejo multi-bases de datos**. Un sistema multi-bases de datos (Smulti-BD) tiene múltiples SMBDs, que pueden ser de tipos diferentes, y múltiples bases de datos existentes. La integración de todos ellos se realiza mediante subsistemas de software. La arquitectura general de tales sistemas se presenta en la Figura 2.12. En contraste con los sistemas homogéneos, existen usuarios locales y globales. Los usuarios locales accesan sus bases de datos locales sin verse afectados por la presencia del Smulti-BD.
En algunas ocasiones es importante caracterizar a los sistemas de bases de datos distribuidas por la forma en que se organizan sus componentes. En la Figura 2.13 se presenta la arquitectura basada en componentes de un SMBD distribuido.
Consiste de dos partes fundamentales, el procesador de usuario y el procesador de datos. El procesador de usuario se encarga de procesar las solicitudes del usuario, por tanto, utiliza el esquema externo del usuario y el esquema conceptual global. Así también, utiliza un

diccionario de datos global.
El procesador de usuario consiste de cuatro partes: un manejador de la interfaz con el usuario, un controlador semántico de datos, un optimizador global de consultas y un supervisor de la ejecución global. El procesador de datos existe en cada nodo de la base de datos distribuida. Utiliza un esquema local conceptual y un esquema local interno.
El procesador de datos consiste de tres partes: un procesador de consultas locales, un manejador de recuperación de fallas locales y un procesador de soporte para tiempo de ejecución.

9

2.3 Niveles de transparencia

El propósito de establecer una arquitectura de un sistema de bases de datos distribuidas (SBDD) es ofrecer un nivel de transparencia adecuado para el manejo de la información. La ***transparencia*** se puede entender como la separación de la semántica de alto nivel de un sistema de las aspectos de bajo nivel relacionados a la implementación del mismo. Un nivel de transparencia adecuado permite ocultar los detalles de implementación a las capas de alto nivel de un sistema y a otros usuarios.
En sistemas de bases de datos distribuidos el propósito fundamental de la transparencia es proporcionar *independencia de datos* en el ambiente distribuido. Se pueden encontrar diferentes aspectos relacionados con la transparencia. Por ejemplo, puede existir transparencia en el manejo de la red de comunicación, transparencia en el manejo de copias repetidas o transparencia en la distribución o fragmentación de la información.
La ***independencia de datos*** es la inmunidad de las aplicaciones de usuario a los cambios en la definición y/u organización de los datos y viceversa. La independencia de datos se puede dar en dos aspectos: lógica y física.
1. Independencia lógica de datos. Se refiere a la inmunidad de las aplicaciones de usuario a los cambios en la estructura lógica de la base de datos. Esto permite que un cambio en la definición de un esquema no debe afectar a las aplicaciones de usuario. Por ejemplo, el agregar un nuevo atributo a una relación, la creación de una nueva relación, el reordenamiento lógico de algunos atributos.
2. ***Independencia física de datos***. Se refiere al ocultamiento de los detalles sobre las estructuras de almacenamiento a las aplicaciones de usuario. Esto es, la descripción física de datos puede cambiar sin afectar a las aplicaciones de usuario. Por ejemplo, los datos pueden ser movidos de un disco a otro, o la organización de los datos puede cambiar.

10

La ***transparencia al nivel de red*** se refiere a que los datos en un SBDD se accedan sobre una red de computadoras, sin embargo, las aplicaciones no deben notar su existencia. La transparencia al nivel de red conlleva a dos cosas:

1. Transparencia sobre la localización de datos. Esto es, el comando que se usa es independiente de la ubicación de los datos en la red y del lugar en donde la operación se lleve a cabo. Por ejemplo, en Unix existen dos comandos para hacer una copia de archivo. Cp se utiliza para copias locales y rcp se utiliza para copias remotas. En este caso no existe transparencia sobre la localización.

2. ***Transparencia sobre el esquema de nombramiento***. Lo anterior se logra proporcionando un nombre único a cada objeto en el sistema distribuido. Así, no se debe mezclar la información de la localización con en el nombre de un objeto.

La ***transparencia sobre replicación*** de datos se refiere a que si existen réplicas de objetos de la base de datos, su existencia debe ser controlada por el sistema no por el usuario. Se debe tener en cuenta que al cuando el usuario se encarga de manejar las réplicas en un sistema, el trabajo de éste es mínimo por lo que se puede obtener una eficiencia mayor. Sin embargo, el usuario puede olvidarse de mantener la consistencia de las réplicas teniendo así datos diferentes.

La ***transparencia a nivel de fragmentación de datos*** permite que cuando los objetos de la bases de datos están fragmentados, el sistema tiene que manejar la conversión de consultas de usuario definidas sobre relaciones globales a consultas definidas sobre fragmentos. Así también, será necesario mezclar las respuestas a consultas fragmentadas para obtener una sola respuesta a una consulta global. El acceso a una base de datos distribuida debe hacerse en forma transparente.

11

Como un ejemplo se utilizará a lo largo de estas notas una base de datos que modela una compañía de ingeniería.

POR LO QUE ES IMPORTANTE REVISAR EL **Ejemplo 2.1**. EN LOS APUNTES QUE DESCRIBE UNA BDD

Las entidades a ser modeladas son ingenieros y proyectos. Para cada ingeniero, se desea conocer su número de empleado (ENO), su nombre (ENOMBRE), el puesto ocupado en compañía

(TITULO), el salario (SAL), la identificación de los nombres de proyectos en los cuales está trabajando (JNO), la responsabilidad que tiene dentro del proyecto (RESP) y la duración de su responsabilidad en meses (DUR). Similarmente, para cada proyecto se desea conocer el número de proyecto (JNO), el nombre del proyecto (JNOMBRE), el presupuesto asignado al proyecto (PRESUPUESTO) y el lugar en donde se desarrolla el proyecto (LUGAR).
be tener en cuenta que en cada sitio de la corporación puede haber esquemas diferentes o repetidos. Por ejemplo, en la Figura 2.2 se presentan esquemas diferentes para el manejo de proyectos, empleados y puestos en cada sitio de la bases de datos del Ejemplo 2.1.

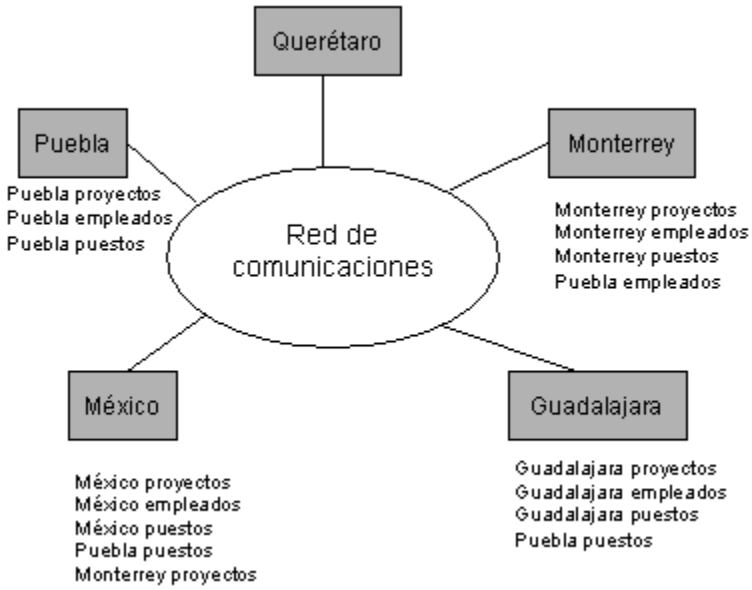

Figura 2.2. Diferentes sitios de una corporación.

12

En resumen, la transparencia tiene como punto central la independencia de datos. Los diferentes niveles de transparencia se pueden organizar en capas como se muestra en la Figura 2.3. En el primer nivel se soporta la transparencia de red. En el segundo nivel se permite la transparencia de replicación de datos. En el tercer nivel se permite la transparencia de la fragmentación. Finalmente, en el último nivel se permite la transparencia de acceso (por medio de lenguaje de manipulación de datos).
La responsabilidad sobre el manejo de transparencia debe estar compartida tanto por el sistema operativo, el sistema de manejo de bases de datos y el lenguaje de acceso a la base de datos distribuida. Entre estos tres módulos se deben resolver los aspectos sobre el procesamiento distribuido de consultas y sobre el manejo de nombres de

objetos distribuidos.

Figura 2.3. Organización en capas de los niveles de transparencia.

BASE DE DATOS DISTRIBUIDAS MIS 515

15

13

BREAK====== HAGAMOS BREAK PARA VER EN SIGUIENTE SESION FRAGMENTACION.

14

2.4 Fragmentación y distribución de datos

El problema de fragmentación se refiere al particionamiento de la información para distribuir cada parte a los diferentes sitios de la red, como se observa en la Figura 3.2. Inmediatamente aparece la siguiente pregunta: cuál es la unidad razonable de distribución? Se puede considerar que una relación completa es lo adecuado ya que las vistas de usuario son subconjuntos de las relaciones. Sin embargo, el uso

completo de relaciones no favorece las cuestiones de eficiencia sobre todo aquellas relacionadas con el procesamiento de consultas.
El diseño de una base de datos distribuida, cualquiera sea el enfoque que se siga, debe responder satisfactoriamente las siguientes preguntas:

- Por qué hacer una fragmentación de datos?
- Cómo realizar la fragmentación?
- Qué tanto se debe fragmentar?
- Cómo probar la validez de una fragmentación?
- Cómo realizar el asignamiento de fragmentos?

- Cómo considerar los requerimientos de la información?

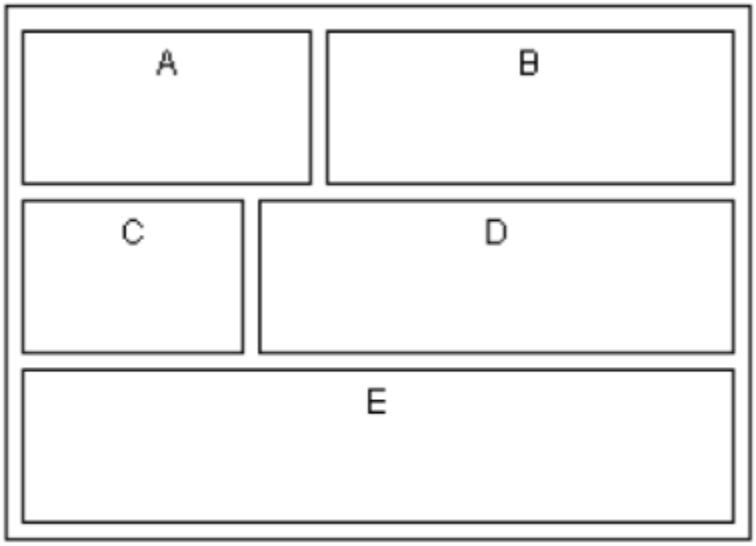

Figura 3.2. El problema de fragmentación de relaciones.

La otra posibilidad es usar fragmentos de relaciones (sub-relaciones) lo cual favorece la ejecución concurrente de varias transacciones que accedan porciones diferentes de una relación. Sin embargo, el uso de sub-relaciones también presenta inconvenientes. Por ejemplo, las vistas de usuario que no se pueden definir sobre un solo fragmento necesitarán un procesamiento adicional a fin de localizar todos los fragmentos de una vista.
Aunado a esto, el control semántico de datos es mucho más complejo ya que, por ejemplo, el manejo de llaves únicas requiere considerar todos los fragmentos en los que se distribuyen todos los registros de la relación. En resumen, el objetivo de la fragmentación es encontrar unnivel de particionamiento adecuado en el rango que va desde tuplas o atributos hasta relaciones completas (ver Figura 3.3).

Ejemplo 3.1. Considere la relación J del ejemplo visto en el capítulo 2.
J:

JNO	JNOMBRE	PRESUPUESTO	LUGAR
J1	Instrumentación	150000	Monterrey
J2	Desarrollo de bases de datos	135000	México
J3	CAD/CAM	250000	Puebla
J4	Mantenimiento	310000	México
J5	CAD/CAM	500000	Guadalajara

JNO JNOMBRE PRESUPUESTO LUGAR
J1 Instrumentación 150000 Monterrey

JNO	JNOMBRE	PRESUPUESTO	LUGAR
J1	Instrumentación	150000	Monterrey
J2	Desarrollo de bases de datos	135000	México

J2 Desarrollo de
bases de datos

JNO	JNOMBRE	PRESUPUESTO	LUGAR
J3	CAD/CAM	250000	Puebla
J4	Mantenimiento	310000	México
J5	CAD/CAM	500000	Guadalajara

135000 México
J3 CAD/CAM 250000 Puebla
J4 Mantenimiento 310000 México

J5 CAD/CAM 500000 Guadalajara

La relación J se puede fragmentar horizontalmente produciendo los siguientes fragmentos.

J1: proyectos con presupuesto menor que $200,000

JNO JNOMBRE PRESUPUESTO LUGAR

J1 Instrumentación 150000 Monterrey

J2 Desarrollo de
bases de datos
135000 México

J2: proyectos con presupuesto mayor que o igual a
$200,000

JNO JNOMBRE PRESUPUESTO LUGAR

J3 CAD/CAM 250000 Puebla

J4 Mantenimiento 310000 México

J5 CAD/CAM 500000 Guadalajara