

BASE DE DATOS DISTRIBUIDAS

Distributed Database System (DDBS) puede denominarse a la unión de 2 tecnologías: Database System (Sistema de Base de Datos) y Computer Network (Redes).

Debe dejarse bien claro que el propósito más importante de esta tecnología es la **INTEGRACIÓN**, no la **CENTRALIZACIÓN**.

Sistema distribuido: es un número de elementos de procesamiento autónomos (no necesariamente homogéneos) que están interconectados por una red de computadoras, y que cooperan para realizar una tarea asignada.

BASE DE DATOS DISTRIBUIDAS

Es un conjunto de múltiples Bases de Datos, interrelacionadas lógicamente, y que se hallan distribuidas sobre una red de computadoras. En cuanto que el Administrador de BD distribuidas (Distributed Database Management System or Distributed DBMS) es el software que permite la administración de la BD distribuida y hace que dicha distribución sea transparente para los usuarios.

Los más importantes términos de esta definición son “Lógicamente interrelacionados” y “Distribución sobre una red”.

VENTAJAS DE LAS BASES DE DATOS DISTRIBUIDAS

- **Autonomía Local**
- **Mejoramiento de la Performance**
- **Mejoramiento de la Confiabilidad/Disponibilidad:**
- **Economía**
- **Expansibilidad**
- **Acceso compartido**

PROBLEMAS VINCULADOS A LOS SBDD

- **DISEÑO DE LA BDD**: Esta área aborda el problema de la distribución: Replicación vs Fragmentación. Básicamente es cómo distribuir los datos y las aplicaciones.
- **PROCESAMIENTO DISTRIBUIDO DE LAS CONSULTAS (QUERYs)**
- **ADMINISTRACIÓN DISTRIBUIDA DEL DIRECTORIO**
- **(CATÁLOGO)**
- **CONTROL DISTRIBUIDO DEL ACCESO CONCURRENTES**
- **PROPAGACIÓN DE LAS ACTUALIZACIONES**
- **CONTROL DE LA RECUPERACIÓN**

12 REGLAS QUE DEBEN CUMPLIRSE EN UN SISTEMA DISTRIBUIDO

1. **Autonomía local** (los sitios deben ser autónomos), es decir que todas las operaciones en un sitio dado se controlan en ese sitio. Un sitio **X** no debe ser perjudicado por la caída de otro sitio **Y** (En el grado en que esto es posible)
2. **No dependencia de un sitio central:** Muy ligado al anterior objetivo
3. **Operación continua:** No debería apagarse por cuestiones operativas
4. **Independencia respecto a la localización:** Los usuarios no necesitan saber dónde están los datos, simplemente utilizarlos.
5. **Independencia con respecto a la fragmentación.** Hay 2 clases de fragmentación: Vertical y horizontal (resultado de las operaciones de proyección y restricción respectivamente) que se realizan con criterios razonados de volumen y frecuencia de uso. Se supone que los datos se presentan como una unidad ante el usuario
6. **Independencia de la réplica:** Puede ser necesario replicar los datos en sitios remotos donde no existe posibilidad de comunicación en línea. El inconveniente es, por supuesto, el peligro de desincronizar las actualizaciones.

12 REGLAS QUE DEBEN CUMPLIRSE EN UN SISTEMA DISTRIBUIDO

7. **Procesamiento distribuido de consultas:** Ubicación de los datos distribuidos y optimización a través de una estrategia diferente.
8. **Manejo distribuido de transacciones:** Debe conservarse el control de recuperación y el control de concurrencia. Deberá intervenir un “agente” que asegure que una transacción dada sea atómica (todo o nada) en el ambiente distribuido. Todos los agentes que participan en una transacción deben comprometerse al unísono o retroceder al unísono.
9. Independencia con respecto al equipo
10. Independencia con respecto al Sistema Operativo
11. Independencia con respecto a la red
12. Independencia con respecto al DBMS

Transparencia en una DBMS

La “**transparencia**” en una Base de Datos se refiere a la separación ideal entre las características semánticas de “alto nivel” de la aplicación, y las características de “bajo nivel” que tienen que ver con los detalles de implementación. Es decir, idealmente, se espera un alto grado de transparencia lo que permitirá ocuparse netamente de la complejidad de la aplicación.

Tipos de transparencia

- ◆ **Independencia de los datos:** Se puede hablar de la independencia lógica (si la aplicación trabaja sobre una Vista limitada de los datos, ella no se verá afectada si se agregan o eliminan datos que no pertenecen a la vista. E Independencia física se refiere a que la aplicación no tiene por qué conocer características como tipo de archivo, tipo de acceso, fragmentación de archivos, etc.
- ◆ **Transparencia de la red:** el usuario no tiene que notar la diferencia entre trabajar con una BD centralizada o una BD distribuida. Para lograrlo, tienen que existir también: Transparencia en la localización y transparencia en la denominación: La inclusión del nombre de la localización como parte del nombre de los objetos, obliga al usuario a asignar un “alias” a los objetos. Por otro lado, se generan problemas al cambiar un objeto de uno a otro sitio

Tipos de transparencia

- ◆ **Transparencia de la replicación:** Por razones de seguridad, siempre es recomendable poseer más de una copia de los datos distribuidos en distintas localidades. La administración de las réplicas no debe ser un problema que deba ser trasladado al usuario
- ◆ **Transparencia de la fragmentación:** Ante los inconvenientes de las réplicas totales, se aplica la fragmentación. El problema es tratar con los datos fragmentados: El sistema debe considerar las estrategias de consulta que permitan que un “query global” sea fragmentado.

ESTRATEGIAS DE DISTRIBUCION DE DATOS

- DISTRIBUCION DE ARCHIVOS NO FRAGMENTADOS
- REPLICACIÓN
- FRAGMENTACIÓN
 - Horizontal
 - Vertical

PARA QUE SE FRAGMENTA ?

El punto más importante en cuanto a la fragmentación es considerar las “Unidades de distribución”. La relación completa no es una unidad apropiada:

- Primero: porque normalmente las vistas en aplicaciones del usuario tratan con una parte de los datos.
- Segundo: Si la distribución consiste; o en dejar toda una relación en un sólo sitio, o bien replicar en varios sitios, existirá un alto volumen de acceso remoto innecesario, o bien un desperdicio de almacenamiento.

PARA QUE SE FRAGMENTA?

Por último, si se fragmentan los datos , será posible que un número de transacciones pueda ejecutarse concurrentemente. Eso también puede significar la ejecución paralela de un simple query, dividiéndolo en un grupo de subqueries (sobre cada fragmento).

Existen también desventajas de la **FRAGMENTACIÓN:**

- Degradación de la performance, cuando se hace necesario acceder a más de un fragmento y luego ejecutar el JOIN entre ellos.
- El control semántico y de integridad se hace más difícil, dado que un simple chequeo de dependencias podría significar el recorrido sobre todos los sitios.

REGLAS DE UNA BUENA FRAGMENTACIÓN

- ♦ **Complejitud:** Si una relación R es descompuesta en relaciones R_1, R_2, R_n , entonces cada ítem de datos que se encuentra en R , deberá poder encontrarse también en una o más R_{is} . Esta propiedad es idéntica a la de “descomposición sin pérdida” en el proceso de normalización. En el caso de fragmentación horizontal el término ítem se refiere a una tupla, en tanto que en la fragmentación vertical se refiere a un atributo.
- ♦ **Reconstrucción:** Si una relación R se descompone en fragmentos R_1, R_2, R_n , entonces debe poder definirse un operador relacional ∇ tal que
$$R = \nabla R_i, \quad \forall R_i \in F_R$$

FRAGMENTACIÓN VERTICAL

- Es la división del conjunto de atributos de un objeto en subconjuntos, que pueden solaparse.
- Los fragmentos se obtienen proyectando la relación original sobre cada conjunto de atributos.
- Para que la fragmentación sea sin pérdida, cada fragmento deberá incluir la PK.
- Se define entonces al fragmento $R_i = \pi_{\langle \text{atributos} \rangle} R$
- Se reconstruye $R = R_i \bowtie R_j \bowtie \dots R_n$

FRAGMENTACIÓN HORIZONTAL

- Es la división de las tuplas de una relación en fragmentos.
- Normalmente los fragmentos son disjuntos.
- Para que la fragmentación sea sin pérdida, cada fragmento deberá incluir la PK.
- Se define entonces al fragmento como
$$R_i = \sigma_{\langle \text{condición} \rangle} R$$
- Se reconstruye $R = R_i \cup R_j \cup \dots R_n$

Arquitecturas para la DDBMS

La arquitectura de un sistema distribuido debe contar:

- ◆ Para cada sitio: con un esquema interno local (Local Internal Schema): para definir la organización física de cada sitio.
- ◆ Para cada sitio: con un esquema Conceptual Local (Local Conceptual Schema) :para definir el esquema conceptual de cada sitio
- ◆ Un solo esquema Conceptual Global (Global Conceptual Schema) que describe la visión de los datos de toda la empresa, así como la administración de los fenómenos de fragmentación y replicación
- ◆ Un Esquema Externo (External Schema), que también es global, que corresponde a las aplicaciones y accesos de los usuarios.

Arquitecturas para la DDBMS

Puede hablarse también de un Diccionario Global (Global Dictionary/ Directory), y de Directorios o diccionarios locales (Local Dictionary/Directory).

Directorios globales

- Los directorios globales contienen **meta-datos**. Son una extensión de los diccionarios locales y contienen información local de los fragmentos y cómo ellos están conformados.
- El diccionario puede localizarse en **un sitio** (con lo que se recarga dicho sitio, produce cuellos de botella y aumenta el tráfico en dicho sitio), o **distribuirse** (con lo que aumenta la complejidad).
- Así mismo el diccionario puede también replicarse, haciéndolo de más fácil acceso y dando mayor confiabilidad al conjunto, pero también añadiendo riesgos de desactualización y complejidad.
- En suma los factores a tener en cuenta son: tiempo de respuesta, tamaño del diccionario, capacidad del equipo, requerimientos de confiabilidad, frecuencia de cambios.

DISEÑO DE UNA DBD

El diseño de un Sistema Distribuido involucra decisiones tales como la ubicación de programas y datos en los sitios de la red, así como el diseño de la red en sí. En lo que respecta a un DDBMS, el diseño de aplicaciones involucra 2 aspectos:

- ◆ La distribución del software DDBMS
- ◆ La distribución de los programas de aplicación que corren sobre él.

Estrategias alternativas de diseño

Las 2 alternativas principales de diseño son la aproximación top-down y la bottom-up

Proceso Top-down

La estrategia consiste en analizar los requerimientos que definen el “ambiente del sistema”, tomando en cuenta las necesidades de datos y procedimientos. Por otro lado es muy importante determinar a qué objetivo se pretende llegar con respecto a los grandes propósitos de una DDB (performance, confiabilidad, disponibilidad, etc.)

El diseño debe empezar por el Esquema Conceptual Global (como si se tratara de una BD centralizada) y posteriormente se llegará al proceso de distribución de entidades en los sitios. En muchos casos, las relaciones deberán ser divididas en fragmentos. Por lo tanto el punto más importante es determinar tanto la fragmentación como la distribución.



Proceso Bottom up

Es aplicable cuando ya existen varias bases de Datos y se las quiere integrar .

El punto de inicio es la de definir los esquemas conceptuales individuales de cada sitio.

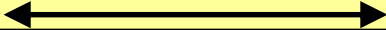
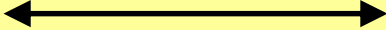
ALTERNATIVAS DE UBICACIÓN DE LOS DATOS:

Una vez que se decide la ubicación de un ítem de datos, éste puede ser replicado, o se puede mantener una sola copia del mismo.

RAZONES PARA REPLICAR: Eficiencia y confiabilidad en operaciones de sólo consulta: Si existen réplicas, existen oportunidades de acceder a los datos aún cuando exista un fallo

DESVENTAJAS DE LA REPLICACIÓN: El sistema debe asegurarse que todas las operaciones de actualización actualizan todas las copias apropiadamente.

ALTERNATIVAS PARA LA REPLICACIÓN

	Replicación Total	Replicación Parcial	Particionamiento
Proceso de consultas	Fácil	Misma Dificultad 	
Administración del Diccionario	Fácil o no Existe	Misma Dificultad 	
Control de Concurrencia	Moderada	Difícil	Fácil
Confiabilidad	Muy alta	Alta	Baja

SEGURIDAD DE LOS DATOS

El control de las autorizaciones asegura que los usuarios que están autorizados con un cierto nivel de acceso accedan solamente a aquellos objetos indicados por dicho nivel.

El control de autorización se complica en un ambiente distribuido:

- Autenticación de usuarios remotos
- Administración de reglas de autorización distribuidas
- Manejo de vistas
- Manejo de grupos

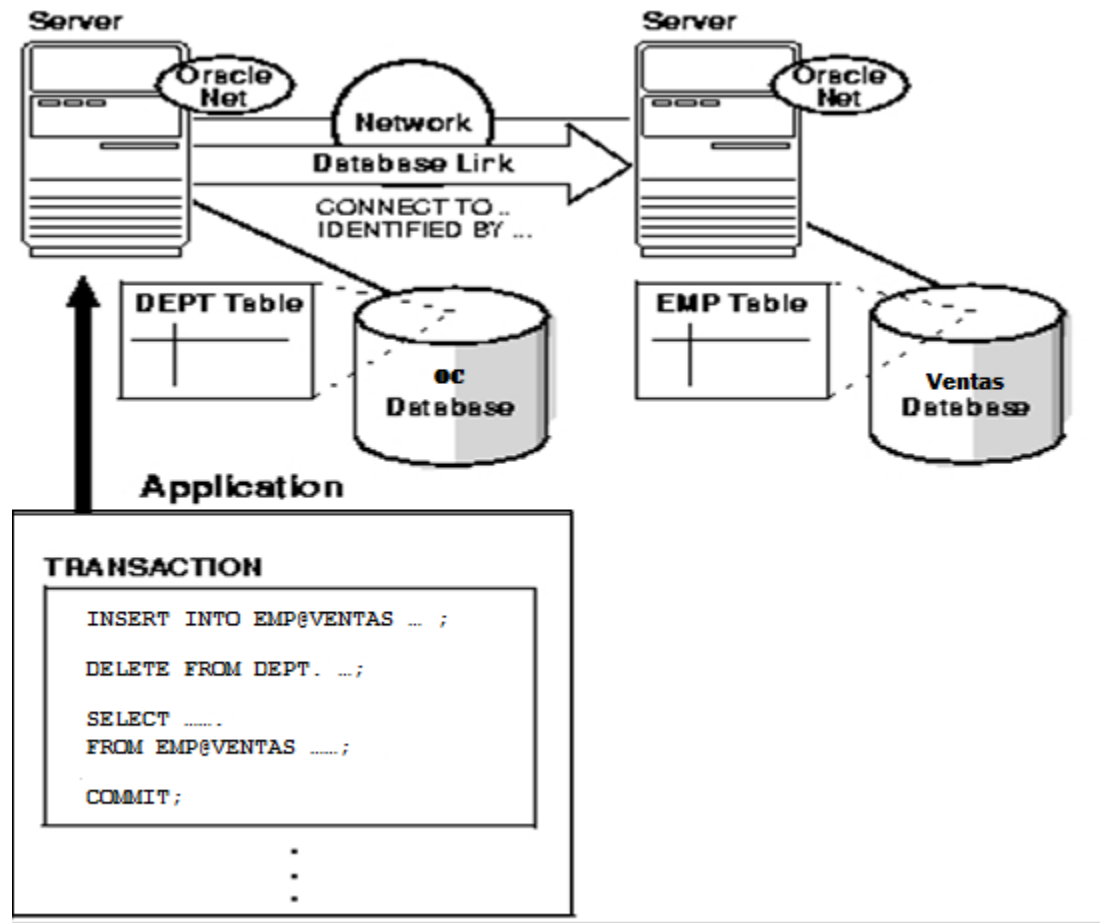
ALTERNATIVAS

- Almacenar la información de los usuarios en el catálogo de todos los sitios
- Cada sitio identifica a sus propios usuarios, lo que le da apertura a los otros sitios



Implementación en Oracle

Esquema de BD distribuida en ORACLE



Esquema de BD distribuida en ORACLE

- **Un nodo de un sistema de base de datos distribuido puede actuar como un cliente, como un servidor, o como ambos, dependiendo de la situación**
- **La figura muestra dos sitios donde están las bases de datos llamadas OC (oficina central) y Ventas. Por ejemplo, en la aplicación mostrada que se ejecuta en la oficina central, para una sentencia SQL emitida contra datos locales (por ejemplo, DELETE FROM DEPT ...), el computador OC actúa como un servidor, mientras que para una sentencia contra datos remotos (por ejemplo, INSERT INTO EMP@VENTAS), el computador OC actúa como cliente.**

Esquema de BD distribuida en ORACLE

- Todas las bases de datos Oracle en un sistema de base de datos distribuidos (SBDD) utilizan el software de red Oracle Net para comunicación entre bases de datos. Oracle Net permite a las bases de datos comunicarse a través de redes para soportar transacciones distribuidas y remotas. Empaqueta sentencias SQL en uno de los muchos protocolos de comunicación para facilitar al cliente la comunicación con el servidor y después empaqueta y devuelve los resultados de forma similar al cliente. Cada base de datos tiene un nombre global único proporcionado por una ordenación jerárquica de nombres de dominio de red que se prefija antes de dar nombre la base de datos para hacerlo único.

Creación de enlace de BD

- Un ENLACE de BD es un objeto en un esquema de BD que habilita el acceso a otra BD Oracle.
- Cuando se define un enlace, el usuario queda habilitado para referirse a un objeto en un esquema remoto agregando el sufijo `@nombre_enlace` al objeto (tabla, vista o procedimiento)

Creación de enlace de BD

- Para crear un enlace de BD se debe tener el privilegio siguiente
`CREATE DATABASE LINK`
- También debe tenerse privilegio de `CREATE SESSION` en el esquema remoto

SINTAXIS

```
CREATE [PUBLIC] DATABASE LINK  
<nombre_enlace>  
CONNECT TO <usuario> IDENTIFIED BY  
    <contraseña>  
USING <cadena de conexión>
```

Mecanismos de réplica

- Versiones anteriores de ORACLE utilizaban mecanismos de réplica denominadas “instantáneas” (snapshots)
- Actualmente las réplicas pueden ser generadas con Vistas Materializadas. El SNAPSHOT se incorpora al ORACLE 10g solamente por cuestiones de compatibilidad.



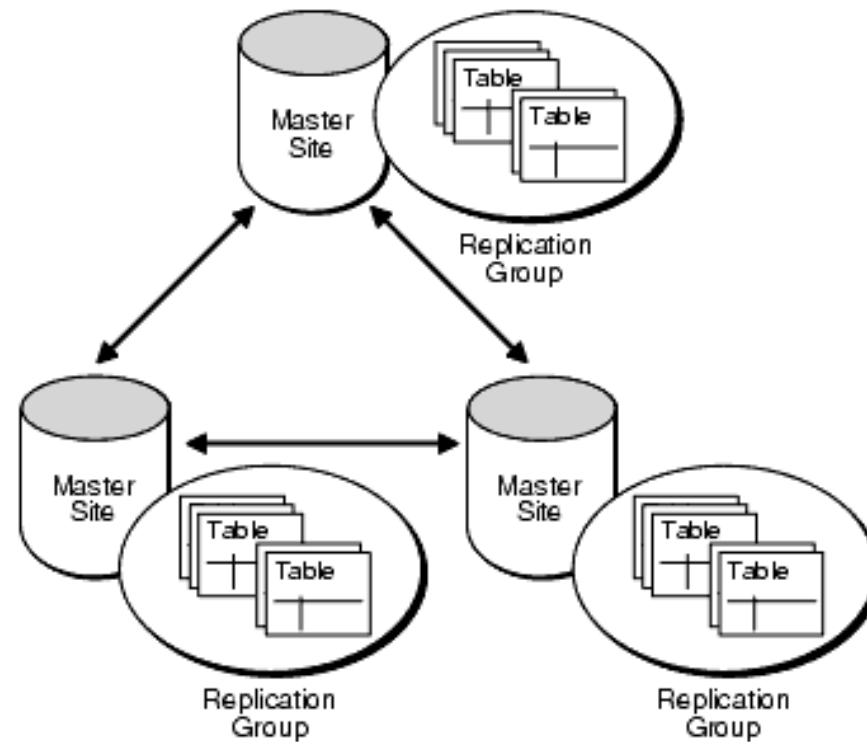
Resumen de los tipos de replicación en ORACLE

- Replicación Multimaster
- Replicación con Vistas Materializadas
- Configuración Híbrida

Replicación Multimaster

- Este tipo de replicación permite que múltiples sitios, administren grupos de objetos de base de datos. Cada sitio administra grupos de objetos replicados actuando como un “master site”, que se comunica con los demás.
- Las aplicaciones pueden actualizar cualquier tabla replicada en cualquier sitio en una configuración de tipo multimaster. El servidor de Oracle trabaja para que todas las réplicas converjan asegurando la consistencia global y la integridad de la aplicación.
- La replicación puede ser asincrónica o sincrónica

Replicación Multimaster



Replicación con vistas materializadas

- Las vistas materializadas proveen los siguientes beneficios:
 - Permiten el acceso local, lo cual mejora el tiempo de respuesta y la disponibilidad
 - Permite acceso local, lo cual mejora el tiempo de respuesta y la disponibilidad
 - En lugar de acceder con las queries al sitio remoto, pueden realizar consultas
 - Incrementan la seguridad de los datos permitiendo replicar solamente el subconjunto de datos requeridos
 - La vista materializada puede ser de sólo lectura o permitir la actualización

USO DE SINÓNIMOS

- El uso de sinónimos provee:
 - Independencia de la localización
 - Transparencia
- Permiten que las aplicaciones funcionen sin ninguna modificación independientemente del usuario y del esquema o base de datos al que pertenece el objeto.

USO DE SINÓNIMOS

- Los sinónimos públicos (Public synonym) son accesibles a todos los usuarios. Sin embargo, el usuario tiene que tener privilegios sobre el objeto subyacente.
- Oracle utiliza el sinónimo cuando el objeto referenciado no está precedido del esquema y tampoco va seguido del database link.

USO DE SINÓNIMOS

Ejemplo

```
CREATE PUBLIC SYNONYM EMP  
FOR EMP@VENTAS;
```

EJERCICIO

- Dos instancias de BD:
 - ORCL
 - DIST
- ORCL tiene el esquema de empleados de ASUNCION
- En DIST, ubicada en CDE, se replicó al inicio el mismo esquema de empleados, pero existen movimientos a nivel local

EJERCICIO

```
CREATE PUBLIC DATABASE LINK db_dist01  
CONNECT TO dist01 IDENTIFIED BY dist01  
USING 'dist';
```

