**APUNTES DE BASE DE DATOS II**

**(Lic. EVELIO HERNANDEZ PASCUAL)**

**Ponderación de la Materia**

**1º Examen ----------------------------- 30%**

**2º Examen ----------------------------- 30% N Exámenes en Computadora -------- 40%**

**Tema I:**

**Bases de Datos Distribuidas**

 Conceptos

 Características

 Ventajas y desventajas

**Definición**

 Base de datos distribuida es un conjunto de BD repartidas en una red, cada una situada en un nodo de modo que todas forman una base de datos lógica.

 Un sistema de BD distribuido consiste en un conjunto de sitios conectados entre sí

mediante n tipo de red de comunicaciones en el cual:

 Cada sitio es un sistema de BD en sí mismo.

 Los sitios han convenido en trabajar juntos con el fin de que un usuario desde cualquier punto de la red tenga acceso a cualquier dato tal como si todos los datos estuviesen almacenados en el propio sitio del usuario.

**Características Base de Datos Distribuidas**

 Varios nodos conectados entre sí.

 Cada nodo es capaz de almacenar datos y ejecutar programa.

 No todos los nodos tienen la misma configuración

 Transparencia de red.

 Desde cualquier punto de la red tiene acceso a cualquier dato.

 Realizan dos tipos de transacciones: Local y Global. La transacción es local cuando todos los datos necesarios para su ejecución se encuentran en el computador desde donde se inicia. La Transacción es global cuando se encuentran en diferentes sitios de la red.

**Ventajas Base de Datos Distribuidas**

 Permite la posibilidad de compartir datos.

 Autonomía cada nodo es responsable de todas las operaciones que se realizan en el mismo además es quien controla los recursos que contienen.

 Disponibilidad, la información está disponible para poder acceder a ella desde cualquier punto de la red; incluso si falla un nodo, el resto puede seguir.

**Desventajas Base de Datos Distribuidas**

 Se incrementa el costo de desarrollo de software.

 Mayor probabilidad de error.

 Mayor sobrecarga de procesamiento.

**Premisas para el desarrollo de las BD distribuidas**

 El desarrollo de las redes de computador.

 El desarrollo de HW y del SW.

 Predicción del costo del HW.

 Usos de los protocolos de comunicación:

o TCP: Transfer Control Protocol

o IP: Internet Protocol

**Un sistema está distribuido si cumple:**

 Los diferentes nodo están formados sobre la existencia de los demás

 Aunque algunas tablas estén almacenadas solo en algunos nodos, estos comparten un esquema global común.

 Generalmente los nodos ejecutan el mismo SW de gestión distribuida.

**Factores importantes en la BDD**

Manejo de

Directorio

Procesamiento de consultas

Diseño de distribución

Confiabilidad

Control de concurrencia

Manejo de bloqueos

**Manejo de directorio.-** Si solo existen usuarios globales se debe manejar un solo directorio global. Si además existen usuarios locales el directorio debe combinar tanto información local como global.

**Diseño de distribución (como distribuir los datos).-** Se refiere a cómo va a distribuir la información en los diferentes sitios de la red.

**Confiabilidad**.- Es una propiedad que garantiza que las transacciones se realizan o no se realizan.

**Procesamiento de consulta.-** Lo importante es establecer una estrategia. La mejor manera en el menor trafico posible en red.

**Control de concurrencia.-** Es el que permite que unas transacciones no interfieran con otras.

**Manejo de bloqueo.-** Si varias transacciones necesitan acceder a un mismo registro de la BD, el motor de la BD lo va bloquear de manera tal, que hasta que una transacción no finalice, el trabajo con dicho registro otra no la puede cumplir.

**Sistemas de BD Homogéneo**

Usuario Global Usuario Global

Sistema BDD

SW de manejo de datos

SW de manejo de datos

SW de manejo de datos

BD1 BD2 BD3

 En todos los nodos se emplea el mismo motor de datos.

 En todos los sitios se emplea el mismo modelo de datos.

 No existen usuarios locales.

 El esquema global de la BD está formada por la unión de las de **FALTA** de datos locales y las vistas que los usuarios definen sobre el esquema global.

**Sistemas de BDD heterogéneas**

Usuario Global Usuario Global

Sistema BDD

Usuario Local Usuario Local

SW de manejo de datos

SW de manejo de datos

SW de manejo de datos

BD1 BD2 BD3

 Se puede emplear diferentes motores de BD en los distintos nodos. Se pueden emplear distintos modelos de datos.

 Trabaja con usuarios globales y locales.

 Los usuarios locales acceden a la BD sin verse afectados por la presencia del sistema de manejo de múltiples BD.

**Fragmentación**

Es el proceso mediante el cual un conjunto de datos se divide en F partes conocidas como fragmentos.

**Condiciones para una buena fragmentación**

 Totalidad: Cada dato debe estar localizado en al menos uno de los fragmentos.

 Reconstrucción: A partir de los fragmentos deber ser posible reconstruir la BD en su totalidad.

**Tipos de fragmentación**

 **Horizontal** La fragmentación es horizontal cuando se realiza el operador adicional seleccionar A partir de una tabla original se crea una nueva tabla que estará compuesta por las tuplas de la tabla inicial que satisfacen una determinada condición.



 **Vertical** Cuando se emplea el operador proyección A partir de una tabla original se crea una tabla que estará compuesta de la tabla original.



 **Mixta:** Si se realiza combinando la selección con proyección se dice que es mixta.

 **Ejemplo:**



**Diseño de distribución**

Se refiere a como se ubican los fragmentos en los F sitios de la red. El diseño de distribución puede ser por:

 **Particionamiento:** Fragmentos → 1 nodo (Una tabla está disponible en un solo lugar).

 **Replicación:** Fragmento → N nodos (una tabla está disponible en N lugares).

**Reglas que deben cumplir las BDD**

 **Autonomía Local:** Todas las operaciones que se realizan en un sitio se controlan en ese propio sitio. Todos los recursos que posee un sitio se administran en el propio sitio.

 **No dependencia de un nodo central:** Si falla l nodo central se cae la red completa.

Aglomera miento de información (cuello de botella). //no se puede entrar

 **Operación continua:** El sistema debe continuar funcionando si se agrega, un nuevo sitio en la red, también si se retira un sitio en la red o si se instala un software en un sitio en la red.

 **Independencia con respecto a la fragmentación de los datos:** Para la realización de FALTA el usuario no tiene porque conocer el tipo de fragmentación que se realizo ni la cantidad de fragmentos en que se dividen.

 **Independencia con respecto a replicas (copia):**

Santa Cruz Oruro Tarija

NUM EMP

DPTO Sueldo NUM EMP

DPTO Sueldo NUM EMP

DPTO Sueldo

E1 DX 4500 E1 DX 4500 E1 DX 4500

E2 DY 4000 E3 DY 5000 E3 DY 5000

E3 DZ 5000 E5 DZ 4000 E5 DZ 4000

E4 DY 6300 E2 DY 4000 E2 DY 4000

E5 DZ 4000 E4 DY 6300 E4 DY 6300

**Ventaja**

 Puede operar sobre información local sin tener que comunicarse con sitios remotos.

 Facilidad en la recuperación de la BD si ocurren desastres (inundaciones, lluvias, fuego, etc.)

**Desventajas**

 Si se actualiza un sitio inmediatamente se tiene que actualizar la réplica porque de lo contrario la BD se puede volver inconsistente.

 Se incrementa el espacio de almacenamiento por el uso de réplica.

**Reglas**

 El usuario no tiene porque conocer i existen réplicas o la ubicación de las réplicas.

 Independencia con respecto a equipamiento.

 Independencia con respecto al S.O.

 Independencia con respecto motor de BD.

 Independencia con respecto a la topología de red.

**Pasos generales para el diseño de una BDD**

**Enfoque top-Down:** Es más apropiado para aplicaciones nuevas y sistemas homogéneas.

Análisis de requerimientos

Objetivos

Diseño Conceptual

Usuarios

Diseño Vistas

Esquema

Conceptual Global

Información de Acceso

Esquema

Externos

Fragmentación y diseño de distribución

Esquemas locales conceptuales

Diseño Físico

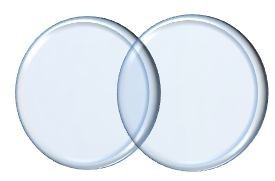
Esquemas internos

Locales

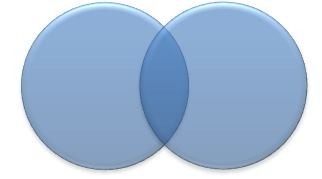
**Enfoque Bottom-Up:** Se emplea cuando se crea la BDD a partir de BD existentes, requiere la selección de un modelo de BD común para describir el esquema global de los datos puede utilizar más motores de BD.

**Operaciones relacionales**

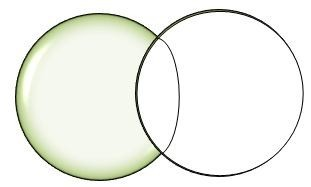
**Unión:** A partir de 2 tablas iníciales se crea una nueva tabla que estaba compuesta por todas las tuplas que contienen las 2 tablas originales no tienen repetidos.



**Intersección:** A partir de 2 tablas originales se crea una nueva tabla que estará compuesta por las tuplas que coinciden en las 2 tablas iníciales.



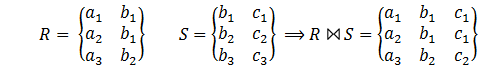
**Diferencia:** A partir de 2 tablas originales se crea una nueva tabla que estará formada por los registros que aparecen en una de las tablas pero no en la otra.



**Producto:** A partir de 2 tablas se genera una nueva para la cual se toma en cuenta todas las posibles combinaciones de tuplas entre las 2 tablas originales.

**Operación relacional:**

**Reunión ( )** A partir de 2 tablas originales se crea una nueva tabla, para lo cual se toma en cuenta todas las posibles combinaciones de tuplas entre las 2 tablas iníciales, pero solo quedan por respuesta aquellas combinaciones que satisfacen una determinada condición.



**Condición:** “Tener una ocurrencia igual de atributo”**.**

**Ejemplo.-**

**EMPLEADO**

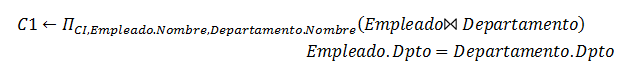
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CI** | **NOMBR** | **DOMICILIO** | **DP** | **SUELDO** |
| 10 | Pepe | Av. Irala #25 | D1 | 3875 |
| 20 | Luis | Av. Cañoto#8 | D1 | 4215 |
| 30 | Carla | C/ Seoane #19 | D2 | 6114 |
| 40 | Aurora | C/ Cuellar #19 | - | 2800 |
| 50 | Miriam | C/ Aroma #2 | D2 | 6114 |
| 60 | Raúl | C/ Rafael Peña | - | 1900 |

**DEPARTAMENTO**

|  |  |  |  |
| --- | --- | --- | --- |
| **DPTO** | **NOMBRE** | **DIRECCION** | **TELEFONO** |
| D1 | Contabilidad | C/ México | 3333333 |
| D2 | Auditoria | C/ México | 3444444 |
| D3 | Almacén | C/ Quijarro | 3555555 |

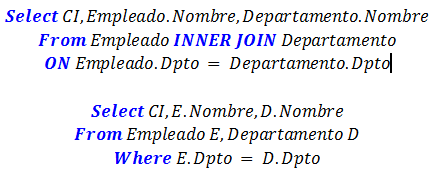
Mostrar CI y NOMBRE de cada EMPLEADO además el NOMBRE del DEPARTAMENTO

que trabaja cada uno.



**m**

|  |  |  |
| --- | --- | --- |
| **CI** | **Empleado.Nombre** | **Departamento.No** |
| 10 | Pepe | Contabilidad |
| 20 | Luis | Contabilidad |
| 30 | Carla | Auditoria |
| 50 | Miriam | Auditoria |



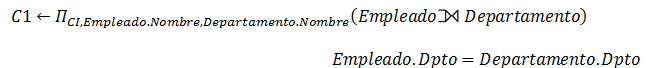
**Reunión externa izquierda**



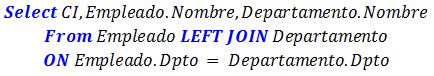
Conserva todas las tuplas de la relación R o relación de la izquierda aun cuando no encuentre una tupla coincidente en la relación S, los atributos correspondientes a S en el resultado se rellenan con valores nulos.

**Ejemplo.-**

Mostrar de cada EMPLEADO CI, NOMBRE y el NOMBRE del DEPARTAMENTO donde trabaja cada uno. Incluir en el resultado a aquellos empleados que no están asignados a ningún DPTO.



|  |  |  |
| --- | --- | --- |
| **CI** | **Empleado.Nombre** | **Departamento.Nombre** |
| 10 | Pepe | Contabilidad |
| 20 | Luis | Contabilidad |
| 30 | Carla | Auditoria |
| 40 | Aurora | - |
| 50 | Miriam | Auditoria |
| 60 | Raúl | - |



**Reunión externa derecha**

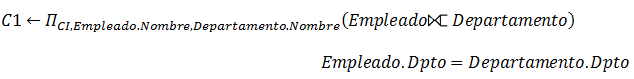


Conserva todas las tuplas de la relación S o relación de la derecha aun cuando no encuentre una tupla coincidente en la relación R. Los atributos correspondientes a R en el resultado se rellenan con valores nulos.

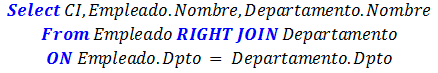
**Ejemplo.-**

Mostrar de cada EMPLEADO CI, NOMBRE y el NOMBRE del DEPARTAMENTO donde trabaja cada uno. Incluir en el resultado a aquellos departamentos que no tienen asignados a ningún EMPLEADO.

|  |  |  |
| --- | --- | --- |
| **CI** | **Empleado.Nombre** | **Departamento.No** |
| 10 | Pepe | Contabilidad |
| 20 | Luis | Contabilidad |
| 30 | Carla | Auditoria |
| 50 | Miriam | Auditoria |
| Null | Null | Almacén |

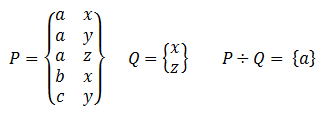


**m**



**División**

A partir de dos relaciones una binaria y otra unaria se construyen una nueva tabla, con todos los valores de un atributo de la relación binaria que coinciden con todos los valores de la relación unaria.



**Ejemplo.-**

|  |  |
| --- | --- |
| **CI** | **NroProy** |
| 123 | 1 |
| 123 | 2 |
| 666 | 3 |
| 453 | 1 |
| 453 | 2 |
| 333 | 2 |
| 333 | 3 |
| 333 | 10 |
| 333 | 20 |
| 999 | 30 |
| 999 | 10 |
| 987 | 10 |
| 987 | 30 |
| 987 | 20 |
| 987 | 25 |
| 888 | 20 |
| 777 | 1 |
| 777 | 2 |

**Empleado\_Proyecto Proyecto\_Lopez**

**Nro\_Proy**

1

2

Mostrar el CI de los empleados que trabajan en todos los proyectos en lo que trabaja el empleado López suponga que el CI de López es el 123.



|  |
| --- |
| **CI** |
| 123 |
| 453 |
| 777 |

**TEMA # 2**

**TRANSACCIONES**

Es una unidad lógica de trabajo que produce varias actualizaciones sobre la BD.

**Ejemplo.-** Se retira de una cuenta 1000$ y se hacen 4 depósitos en otras 4 cuentas de 250$

cada uno, la transacción está compuesta por 5 operaciones.

 Una de las principales características de las transacciones es que se realiza todo o nada.

**Transaction Manager**

Es el manejador y gestor de las transacciones en un motor de BD.

Para realizar su tarea el T.M. se apoya en dos órdenes COMMIT y ROLLBACK.

**COMMIT:** Significa que la transacción se realizo efectivamente.

**ROLLBACK:** Emite está orden cuando ha existido algún tipo de problema y ha sido necesario abortar la transacción.

Data

Log

**Diario bitácora**, en ella se almacena el detalle de todas las acciones que realiza la

transacción sobre la BD

**Punto de sincronización**

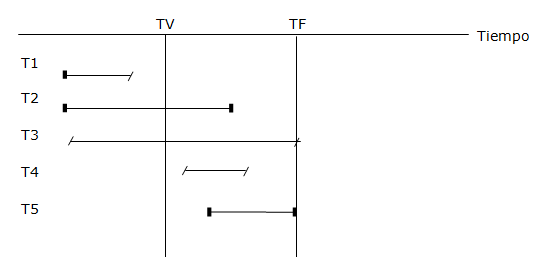
Es el límite entre dos transacciones consecutivas. Las únicas operaciones que establecen puntos de sincronización son COMMIT, ROLLBACK y el inicio de un programa.

Cuando se arriba a un punto de sincronización se graban en el disco duro todas las actualizaciones realizadas por las transacciones que hayan finalizado.

**Punto crítico o de falla**

Es aquel donde se produce un fallo, se pierde el contenido de la memoria RAM se borra, y al borrarse esto da un problema.

¿Cómo sabe el motor de BD que transacciones debe ejecutar nuevamente y que transacciones no será necesario ejecutar?



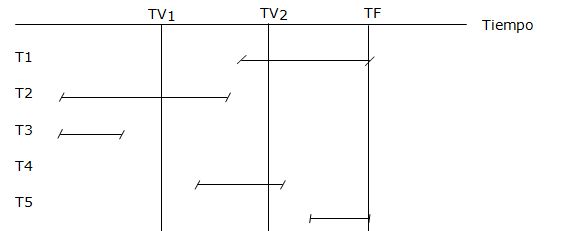
La T1 finalizo durante el tiempo de verificación (TV) No será necesario ejecutarla nuevamente.

La T2 finalizo después del tiempo de verificación, por lo tanto no grabo sus operaciones en el disco duro y SI es necesario ejecutarla nuevamente.

La T3 finalizo después del tiempo de verificación y además durante el tiempo de falla, por lo tanto SI es necesario ejecutarla nuevamente.

La T4 es similar al de T2. La T5 es similar al de T3.

En el grafico se ilustran cinco transacciones que se desarrollan en el tiempo, existen dos puntos de verificación se produce un fallo en el tiempo TF, al restaurarse el sistema indique que transacciones será necesario ejecutar nuevamente y que transacciones no se ejecutaran. Justifique su respuesta para cada uno.



La T1 no término antes del fallo, se debe ejecutar nuevamente. La T2 termino antes del TV2, almacena en el D.D., No.

La T3 termino antes de TV1, almacena en el D.D., No.

La T4 no termino antes TV2 y al llegar al TF no ha terminado todavía, se debe ejecutar nuevamente.

La T5 se inicio durante el TF, se debe ejecutar nuevamente.

R/ Las T1, T4 y T5 se deben ejecutar nuevamente al restaurar sistema.

**Propiedades de las transacciones (ACID)**

 **Atomicidad,** es la propiedad que establece que la transacción se realiza completamente o no se realiza.

 **Coherencia,** es la propiedad que permite que la BD no pase a un estado de inconsistencia, garantiza que los datos sean coherentes.

 **Issolation (Aislamiento),** separa las transacciones concurrente de las actualizaciones producidas por otras transacciones que no se han completado se consigue a través de candado o bloqueo, impide efectos secundarios que podrían distorsionar la BD.

 **Durabilidad,** después de confirmar una transacción sus efectos se mantienen aun cuando se produzca un fallo en el sistema.

**Elementos del motor de BD que intervienen en una transacción**

Procesador Transaction

Manager

Log

Manager

Buffer

Manager

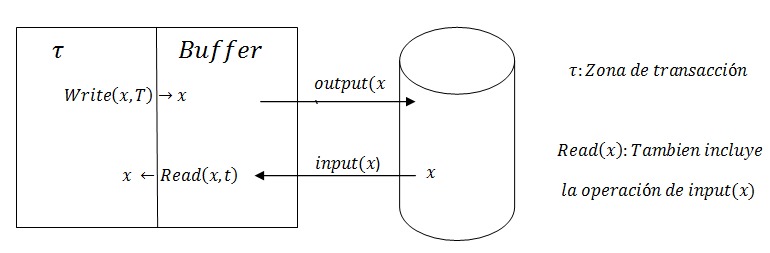
Recovery

Manager

Registro que almacena la historia de los cambios que se van produciendo en la BD para garantizar su consistencia y seguridad

Data

Log



**Input(x, T):** Copia el elemento x de la BD que se encuentra en el D.D. y lo coloca en el buffer de memoria.

**Read(x, T):** Copia el valor del elemento x que se encuentra en el buffer de memoria a la transacción. Si el elemento x, no encuentra en el buffer de memoria lo va a buscar en el D.D.

**Write(x, T):** Copia el valor del elemento x que se encuentra en la zona de transacciones y lo escribe en el buffer de memoria.

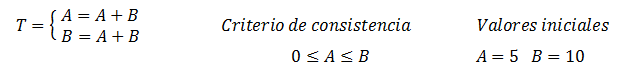
**Output(x):** Copia el valor del elemento x que se encentra en el buffer de memoria y lo graba en el D.D.

**Ejemplo.-**

Consideremos un BD con dos elementos A y B ambos deben ser siempre iguales para mantener la consistencia de la BD.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Acción** | **T** | **Mem A** | **Mem B** | **Disk A** | **Disk B** |
| Read(A, T) | 8 | 8 | - | 8 | 8 |
| T=T \* 2 | 16 | 8 | - | 8 | 8 |
| Write(A, T) | 16 | 16 | - | 8 | 8 |
| Read(B, T) | 8 | 16 | 8 | 8 | 8 |
| T = T \* 2 | 16 | 16 | 8 | 8 | 8 |
| Write(B, T) | 16 | 16 | 16 | 8 | 8 |
| OutPut(A) | 16 | 16 | 16 | 16 | 8 |
| OutPut(B) | 16 | 16 | 16 | 16 | 16 |



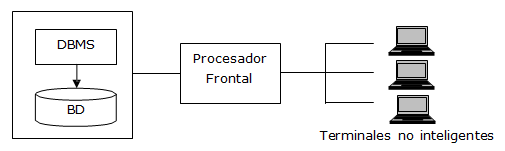
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Acción** | **T** | **Mem A** | **Mem B** | **Disk A** | **Disk B** |
| Read(B, T) | 10 | - | 10 | 5 | 10 |
| Read(A, T) | 5 | 5 | 10 | 5 | 10 |
| T = T + B | 15 | 5 | 10 | 5 | 10 |
| Write(A, T) | 15 | 15 | 10 | 5 | 10 |
| T = T + B | 25 | 15 | 10 | 5 | 10 |
| Write(B, T) | 25 | 15 | 25 | 5 | 10 |
| OutPut(A) | 25 | 15 | 25 | 15 | 10 |
| OutPut(B) | 25 | 15 | 25 | 15 | 25 |

**Niveles de distribución de datos y procesos**

|  |  |  |
| --- | --- | --- |
|  | **Datos en un solo sitio** | **Datos en varios sitios** |
| Procesos en un solo sitio | DBMS anfitrión (Mainframe) | No aplica |
| Proceso en varios sitios | DBMS Cliente / Servidor | DBMS Cliente / Servidor totalmente Distribuido |

**Procesamiento en un solo sitio y datos en un solo sitio**

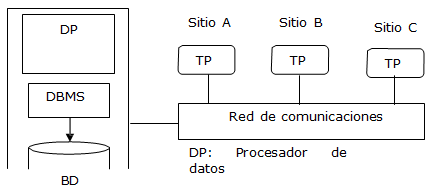
Todo el procesamiento se realiza en un solo CPU o computadora anfitriona (Mainframe, minicomputadora ó PC) todos los datos se guardan en el disco local de la computadora anfitriona donde además está localizado el motor de BD a esta computadora se puede acceder por terminales o inteligentes conectadas a ellas. //típica computadora centralizada.



Sistema centralizado → No distribuido

**Procesamiento en varios sitios y datos en un solo sitio**

Se realizan procesos múltiples en distintas computadoras que comparten un solo deposito de datos esto requiere un servidor de archivos de red que ejecuta operaciones convencionales a las que acceden mediante LAN.



**Procesamiento en varios sitios y datos en varios sitios**

Estamos en presencia de un sistema de administración de BD totalmente distribuido con soporte para múltiples procesadores de datos y transacciones en distintos sitios.

 **Transparencia de fragmentación**

Es el mayor nivel de transparencia, el usuario o programador no necesita conocer que la BD está fragmentada, no es necesario especificar los nombres ni la ubicación de los fragmentos para acceder a los datos.

 **Transparencia de ubicación**

Ocurre cuando el usuario o programador debe especificar los nombres de los fragmentos de la BD pero no su ubicación.

 **Transparencia de ubicación local**

Existe cuando el usuario o programador debe especificar tanto nombres de los fragmentos como las ubicaciones.

Si la sentencia SQL requiere:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre Fragmento** | **Ubicación** | **El motor de BD**  **soporta** | **Nivel de transparencia** |
| SI | SI | Transparencia de ubicación local | Bajo |
| SI | NO | Transparencia de ubicación | Medio |
| NO | NO | Transparencia de fragmentación | Alto |

Suponga que tenemos la tabla: EMPLEADO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | Dirección | DPTO | Sueldo | Fecha\_Nac | CI |

Los datos están distribuidos en 3 lugares: Cochabamba, Santa Cruz y La Paz. La tabla está dividida por ubicación, ósea, los datos de los empleados de Cochabamba, Santa Cruz y La Paz están almacenados en el fragmento de Cochabamba, Santa Cruz y La Paz respectivamente.

Mostrar los datos de los empleados que nacieron antes del 1º de enero de 1960.

**DBMS Distribuido**

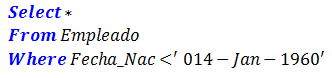
E1 E2 E3

SCZ CBBA LPZ

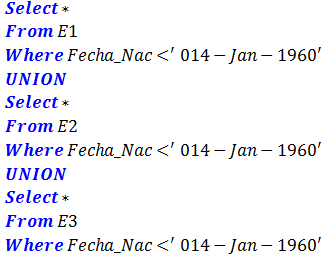
Fragmentación

Ubicación

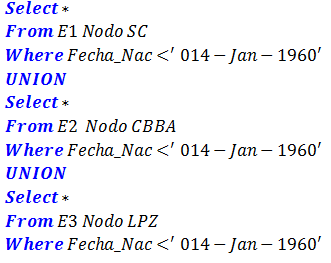
**Caso 1:** La BD soporta transparencia de fragmentación.



**Caso 2:** La BD soporta transparencia de ubicación.



**Caso 3:** La BD soporta transparencia de ubicación local.



**Transparencia de transacción**

Es una propiedad de los motores de BDD que garantiza la integridad y consistencia de la BD la transacción será completa solo si todos los sitios implicados en ella completan las partes que les corresponden.

**Solicitud remota**

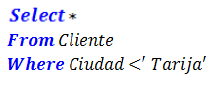
Permite acceder a datos que serán procesados en solo procesador de BD remoto.

Sitio A

TP DP RED

TP: Procesador de Transacciones

DP: Procesador de Datos



**Transacción remota**

Está compuesta por varias solicitudes y se accederá a un solo sitio.

**Cliente**

Sitio A

Sitio B

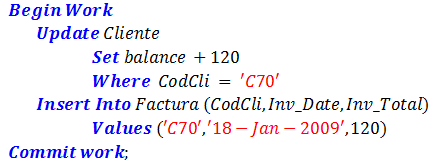
TP DP RED

**Factura**

TP: Procesador de Transacciones

DP: Procesador de Datos

**Cliente**



 La transacción actualiza las tablas Cliente y Factura, ambas tablas están en el sitio B.

 La transacción solo puede hacer referencia a un procesador de datos remoto.

**Transacción distribuida**

Permite que una transacción haga referencia a varios sitios de procesamiento de datos diferentes (locales y remotos) aunque cada solicitud puede hacer referencia solo a un sitio de procesamiento de datos remoto, la transacción como un todo puede hacer referencia a varios sitios.

Sitio A TP

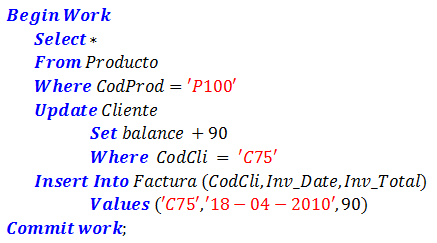
RED

Sitio B DP

**Producto**

TP: Procesador de Transacciones

DP: Procesador de Datos



Sitio C DP

**Factura**

**Cliente**

 La transacción hace referencia a dos sitios B y C la primera solicitud (Select) se procesa en el sitio B, las siguientes solicitudes (Update, Insert) se procesan en el sitio C.

 “Cada solicitud solo accede a un sitio remoto a la vez”.

**Solicitud distribuida**

Permite hacer referencia a datos que están ubicados en varios sitios remotos, cada solicitud puede acceder a datos que se encuentren en varios sitios, la capacidad de ejecutar una solicitud distribuida proporciona la posibilidad de que el procesamiento de datos sea totalmente distribuido, porque se puede:

 Segmentar una tabla (Dividirla en fragmentos).

 Hacer referencia a uno o más fragmentos solamente con una solicitud

(transparencia de fragmentación).

Sitio A TP

RED

Sitio B DP

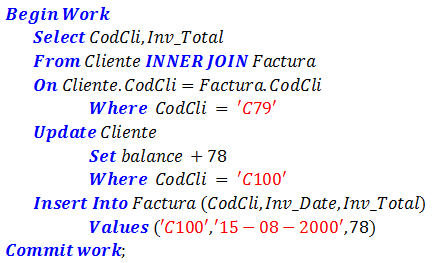
**Producto**

TP: Procesador de Transacciones

DP: Procesador de Datos

Sitio C DP

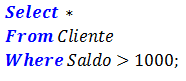
**Factura**



 La característica de la solicitud distribuida también permite, que una sola solicitud haga referencia a una tabla físicamente dividida en fragmentos, suponga que la tabla cliente está dividida en 2 fragmentos C1 y C2 localizados en los sitios B y C respectivamente.

**Ejercicios.-** Realizar las siguientes consultas:

Mostrar los datos de todos los clientes que tienen saldo superior a 1000.



N**ota:** La transparencia de fragmentación solo es provista por un DBMS que soporte “ sol i ci t u d es d i stri b ui d as”.

**Sistemas distribuidos:**

 INGRES / STAR

 ORACLE

 DB2

 SQL (algún tipo de soporte)

Ver el archivo (.SQL) Ejercicio1.

**Técnicas de restauración**

**I. Undo logging (Deshacer).-**

Log

Data

<Start T>

<Commit T> //La transacción se realizo sin ningún problema

<Abort T> //Hay que abortar la transacción por algún problema

<T, x, V> //T: Identificador de transacción

//X:Elemento de la BD que se ha modificado

//V: Valor que tenía x antes de que lo modificaran

**Undo Logging Rules**

**1.** Si la transacción T modifica un elemento de la BD entonces el registro <T, x, v> tiene que ser escrito en el disco antes que un nuevo valor de x se grabe en el propio disco.

**2.** Si se produce Commit Log Record tiene que ser grabado en el disco después de los cambios efectuados por la transacción en la BD.

**Orden de escritura en el disco**

**1.** Los registros log que indican cambios en la BD.

**2.** Cambios en la BD.

**3.** Commit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Buffer** | | **Hard Disk** | |  |
| **Paso** | **Action** | **T** | **MA** | **MB** | **D-A** | **D-B** | **Log** |
| 1 | - | - | - | - | - | - | <Start T> |
| 2 | Read(A,T) | 8 | 8 | - | 8 | 8 | Fallo 3 |
| 3 | T = T \* 2 | 16 | 8 | - | 8 | 8 | <T, A, 8> |
| 4 | Write(A,T) | 16 | 16 | - | 8 | 8 | \*Fallo 2 |
| 5 | Read(B,T) | 16 | 16 | 8 | 8 | 8 |  |
| 6 | T = T \* 2 | 16 | 16 | 8 | 8 | 8 | <T, B, 8> |
| 7 | Write(B,T) | 16 | 16 | 16 | 8 | 8 | \*Fallo 1\* |
| 8 | OutPut(A) | 16 | 16 | 16 | 16 | 8 |  |
| 9 | OutPut(B) | 16 | 16 | 16 | 16 | 16 | <Commit> |
| - | - | - | - | - | - | - | Fallo 4 |

**¿Qué ocurre cuando se produce un fallo?**

Cuando se produce un fallo, el Recover Manager recorre la bitácora en sentido contrario, deshace los cambios efectuados con las transacciones que no han finalizado y graba el registro <Abort T>. Las transacciones que han concluido las ignora.

Porque la transacción finalizo.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fallo 1** | **Fallo 2** | **Fallo 3** | **Fallo 4** |
| B = 8 | A = 8 | <Abort> | Nada |
| A = 8 | <Abort T> |  |  |
| <Abort T> |  |  |  |

**Punto de chequeo o verificación (CKPT)**

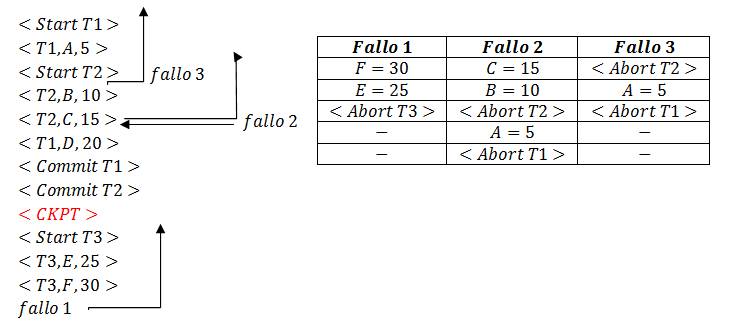
En un punto de verificación:

**1.** Aceptar nuevas transacciones.

**2.** Esperar hasta que todas las transacciones finalice o aborte y se haya escrito un registro Commit o Abort en la bitácora.

**3.** Escribir un registro <CKPT>.

**4.** Concluir aceptando transacciones



**Punto de verificación flexible**

**1.** Escribir un registro



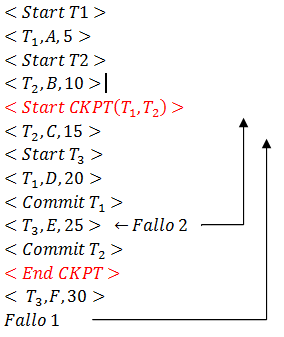
Transacciones que están activas al momento de inicializarse la verificación.

**2.** Esperar hasta que todas las transacciones activas finalice o aborte, pero no prohibir que se inicialicen otras transacciones

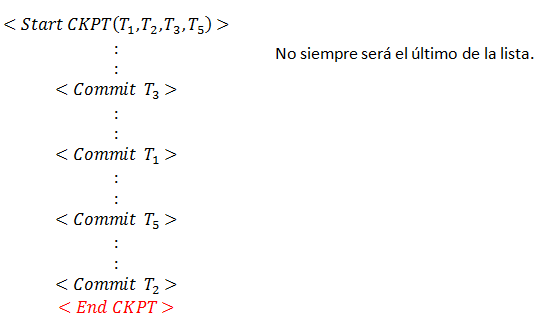


**3.** Escribir un registro . //Fin de verificación.

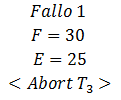
**¿Dónde e ubica el registro** *<End CKPT>* **?**



En la técnica Undo Logging el registro CKPT se coloca después del Commit correspondiente a la última transacción que finaliza.



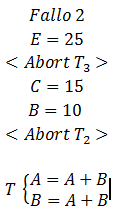
**Caso 1:** Si el Recover Manager encuentra 1º al registros



*<End CKPT>* significa que

todas las transacciones incompletas comenzaron después del registro *<Start CKPT>*

**Caso 2:** Si el Recover Manager encuentra 1º al registro *<Start CKPT>* onces las transacciones incompletas que se venían verificando antes de verificar al *<Start CKPT>* y aquellas transacciones activas no se completaron antes del fallo.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Paso** | **Action** | **T** | **Mem A** | **Mem B** | **Disk A** | **Disk B** | **Log** |
| 1 | - | - | - | - | - | - | <Start> |
| 2 | Read(B, T) | 10 | - | 10 | 5 | 10 | - |
| 3 | Read(A, T) | 5 | 5 | 10 | 5 | 10 | Fallo |
| 4 | T=T+B | 15 | 5 | 10 | 5 | 10 | - |
| 5 | Write(A,T) | 15 | 15 | 10 | 5 | 10 | <T, A, 5> |
| 6 | T=T+B | 25 | 15 | 10 | 5 | 10 | Fallo |
| 7 | Write(B,T) | 25 | 15 | 25 | 5 | 10 | <T, B, 10> |
| 8 | OutPut(A) | 25 | 15 | 25 | 15 | 10 | Fallo |
| 9 | OutPut(B) | 25 | 15 | 25 | 15 | 25 | - |
| 10 | - | - | - | - | - | - | Commit |
| 11 | - | - | - | - | - | - | \*Fallo\* |

**II. Redo Logging.-**

**Diferencia entre Redo Logging y Undo Logging**

Mientras que *Undo Logging* cancela el efecto de las transacciones incompletas e ignora las transacciones concluidas durante la restauración, *Redo Logging* ignora las transacciones incompletas y repite los cambios efectuados por las transacciones que han finalizado.

Mientras que *Undo Logging* requiere que las modificaciones a los elementos de la BD se escriban en el DD antes que Commit, en *Redo Logging* el registro Commit se debe grabar en el DD antes que cualquier modificación.

Mientras que *Undo Logging* graba en el registro actualizador <T, x, V> los valores anterios de la modificaión *Redo Logging* graba los valores actuales.

**Redo Logging Rules**

**1.** Antes de modificar algun elemento x de la BD en DD es necesario que se graben los registros actualizadores que pertenecen a la modificación y Commit.

**Orden de escritura en el disco**

**1.** Registro Log que indican camios en la BD.

**2.** Commit.

**3.** Cambios en la BD.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Buffer** | | **Hard Disk** | |  |
| **Paso** | **Action** | **T** | **MA** | **MB** | **D-A** | **D-B** | **Log** |
| 1 | - | - | - | - | - | - | <Start T> |
| 2 | Read(A,T) | 8 | 8 | - | 8 | 8 | Fallo 1 |
| 3 | T = T \* 2 | 16 | 16 | - | 8 | 8 |  |
| 4 | Write(A,T) | 16 | 16 | - | 8 | 8 | <T, A, 16> |
| 5 | Read(B,T) | 8 | 16 | 8 | 8 | 8 |  |
| 6 | T = T \* 2 | 16 | 16 | 8 | 8 | 8 | Fallo 2 |
| 7 | Write(B,T) | 16 | 16 | 16 | 8 | 8 | <T, B, 16> |
| 8 | - | - | - | - | - | - | <Commit> |
| 9 | OutPut(A) | 16 | 16 | 16 | 16 | 8 |  |
| 10 | OutPut(B) | 16 | 16 | 16 | 16 | 16 | Fallo 3 |

**Como funciona Redo Logging cuando se produce un fallo**

Al producirse un fallo Recover Amnager recorrela la vitacora en sentido contrario rehace los cambios producidos por las transacciones que han finalizado e ignora las transacciones que no han finalizado (graba un registro <abort T> por cada uno).

|  |  |  |
| --- | --- | --- |
| **Fallo 1** | **Fallo 2** | **Fallo 3** |
| <Abort T> | <Abort T> | B=16 |
|  |  | A=16 |

**Punto de chequeo o verificación (CKPT)**

En un punto de verificación:

**1.** Escribir un registro



**2.** Grabar en el DD todos los elementos de la BD que se encontraban en el Buffer de Memoria grabados por las transacciones que habían concluido cuando el registro fue grabado en la bitácora.



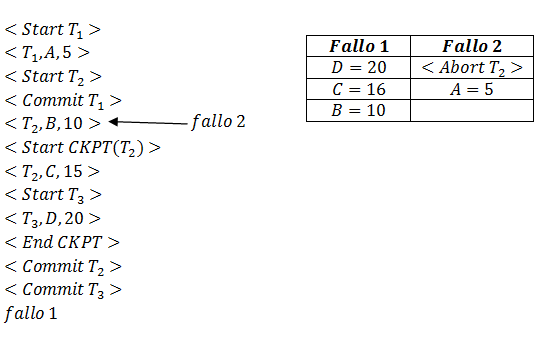
**3.** Escribir el registro



**¿Donde se coloca el registro**



Se coloca antes del Commit correspondiente a la última transacción que finaliza.



1: En este caso comienza después del *<End CKPT>* entonces las transacciones finalizadas antes del *<Start CKPT>*no se toman en cuenta.