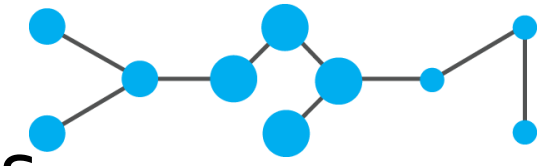
Sistemas de Bases de Datos 2

2024

Ing. Luis Alberto Arias Solórzano

Unidad 5

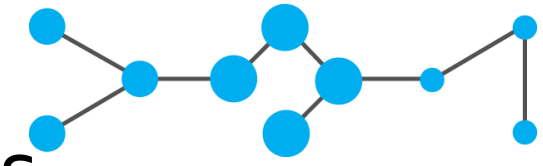


BD Distribuidas
Commit de dos fases

- El concepto de confirmar/deshacer llamado confirmación de **dos** fases. La confirmación de dos fases es importante siempre que una transacción dada pueda interactuar con varios "administradores de recursos" independientes, donde cada uno administra su propio conjunto de recursos recuperables y mantiene su propia bitácora de recuperación.

- Considere una transacción que esté un servidor de base de datos y otra en un servidor separado (dicho sea de paso, tal transacción es perfectamente válida). Si la transacción termina satisfactoriamente se deben confirmar *todas* sus actualizaciones, tanto para los datos del servidor 1 como el servidor 2; por el contrario, si falla, *todas* sus actualizaciones deben ser deshechas. En otras palabras, no debe ser posible confirmar las actualizaciones en 1 y cancelar las actualizaciones de 2, o *viceversa,* ya que entonces la transacción no sería atómica.

- Esto nos lleva a concluir que no tiene sentido que la transacción emita, digamos, un COMMIT para 1 y un ROLLBACK para 2; y a que incluso si emitiera la misma instrucción para ambos, el sistema aún podría fallar entre una y otra, con resultados desafortunados. En vez de ello, la transacción emite un solo COMMIT (o ROLLBACK) a nivel sistema. Ese COMMIT o ROLLBACK "global" es manejado por un componente del sistema llamado coordinador, cuya tarea es garantizar que ambos administradores de recursos confirmen o deshagan al unísono las actualizaciones de las que son responsables, y además proporcionar esa garantía aunque el sistema falle a mitad del proceso. Y es el protocolo de confirmación de dos fases el que permite que el coordinador proporcione esta garantía.

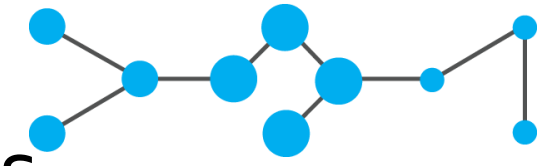


BD Distribuidas
Commit de dos fases

- Funciona de la siguiente forma. Por simplicidad, supongamos que la transacción ha terminado satisfactoriamente su procesamiento de base de datos y por lo tanto, la instrucción al nivel sistema que emite es COMMIT y no ROLLBACK. Al recibir la petición de COMMIT, el coordinador realiza el siguiente proceso de dos fases:

- 1. Primero, da instrucciones a todos los administradores de recursos a fin de que estén listos para manejar la transacción "de una u otra forma". En la práctica esto significa que cada **participante** en el proceso (es decir, cada administrador de recursos involucrado) debe forzar todos los registros de bitácora de los recursos locales usados por la transacción, hacia su propia bitácora física (es decir, hacia el almacenamiento no volátil); esto con el fin de que, sin importar qué pase después, el administrador de recursos tenga ahora un *registro permanente* del trabajo que hizo a nombre de la transacción y por lo tanto sea capaz de confirmar o deshacer las actualizaciones según sea necesario. Suponiendo que la escritura forzada es satisfactoria, el administrador de recursos responde ahora un "OK" al coordinador, y en caso contrario responde "No OK".

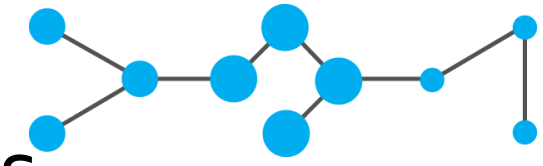
- 2. Cuando el coordinador ha recibido las respuestas de todos los participantes, fuerza una entrada en su propia bitácora física registrando su decisión con respecto a la transacción. Si todas las respuestas fueron "OK", esa decisión es "confirmar", y si alguna respuesta fue "No OK", la decisión es "deshacer". De cualquier forma, el coordinador informa después su decisión a cada participante y luego *cada participante debe confirmar o deshacer la transacción localmente según se le indica.* Observe que cada participante debe hacer lo que dice el coordinador en la fase 2; éste es el protocolo. Observe también que la apariencia del registro de decisión en la bitácora física del coordinador es lo que marca la transición de la fase 1 a la 2.



BD Distribuidas
Commit de dos fases

- Ahora, si el sistema falla durante el proceso general, el procedimiento de reinicio buscará el registro de decisión en la bitácora del coordinador. Si lo encuentra, el proceso de confirmación de dos fases puede continuar donde se quedó. Si no lo encuentra, da por hecho que la decisión fue "deshacer" y de nuevo el proceso puede terminar en forma adecuada.
- *Nota:* Vale la pena señalar que si el coordinador y los participantes están ejecutándose en máquinas diferentes, como puede suceder en un sistema distribuido, una falla en la parte del coordinador puede mantener a algún participante esperando mucho tiempo a que llegue la decisión del coordinador y mientras *está* esperando, cualquier actualización hecha por la transacción a través de ese participante, deberá mantenerse oculta ante otras transacciones (es decir, probablemente esas actualizaciones tendrán que permanecer *bloqueadas*).

- Insistimos en que el administrador de comunicaciones de datos (o el administrador CD) también puede ser visto como un administrador de recursos en el sentido que aquí mencionamos. Es decir, los mensajes también pueden ser vistos como un recurso recuperable (similar a las bases de datos) y el administrador CD debe poder participar en el proceso de consultas.



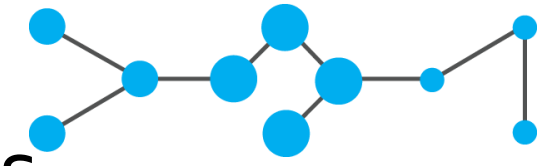
BD Distribuidas Replicacion

Réplica de datos

Si la relación *r* se replica, se guarda una copia de dicha relación en dos o más sitios. En el caso más extremo se tiene una **réplica completa**, en la que se guarda una copia en cada sitio del sistema.

Hay varias ventajas y desventajas en las réplicas.

- • **Disponibilidad.** Si alguno de los sitios que contiene la relación *r* falla, la relación puede hallarse en otro sitio distinto. Por tanto, el sistema puede seguir procesando las consultas que impliquen a *r*, pese al fallo del sitio.
- • **Paralelismo incrementado.** En caso de que la mayoría de los accesos a la relación *r* sólo resulten en la lectura de la relación, varios sitios pueden procesar en paralelo las lecturas que impliquen a *r*. Cuantas más réplicas de *r* haya, mayor será la posibilidad de que los datos necesarios se hallen en el sitio en que se ejecuta la transacción. Por tanto, la réplica de los datos minimiza el movimiento de los datos entre los sitios.
- • **Sobrecarga incrementada durante la actualización.** El sistema debe asegurar que todas las réplicas de la relación *r* sean consistentes; en caso contrario pueden producirse cómputos erróneos. Por eso, siempre que se actualiza *r*, hay que propagar la actualización a todos los sitios que contienen réplicas. El resultado es una sobrecarga incrementada. Por ejemplo, en un sistema bancario, en el que se replica en varios sitios la información de las cuentas, es necesario asegurarse de que el saldo de cada cuenta concuerde en todos los sitios. En general, la réplica mejora el rendimiento de las operaciones leer y aumenta la disponibilidad de los datos.



BD Distribuidas fragmentacion

Fragmentación de los datos

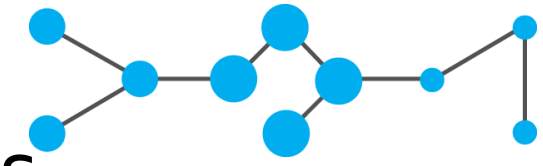
Si la relación *r* se fragmenta, *r* se divide en varios *fragmentos* $r1, r2, \ldots, rn$. Estos fragmentos contienen suficiente información como para permitir la reconstrucción de la relación original *r*. Hay dos esquemas diferentes de fragmentación de las relaciones: fragmentación *horizontal* y fragmentación *vertical*.

- La fragmentación horizontal divide la relación asignando cada tupla de *r* en uno o más fragmentos.
- La fragmentación vertical divide la relación descomponiendo el esquema *R* de la relación *r*. Estos enfoques se ilustrarán fragmentando la relación *cuenta*, con el esquema *esquema-cuenta* = (*número-cuenta*, *nombre-sucursal*, *saldo*)

En la **fragmentación horizontal** la relación *r* se divide en varios subconjuntos, $r1, r2, \ldots, rn$. Cada tupla de la relación *r* debe pertenecer como mínimo a uno de los fragmentos, de modo que se pueda reconstruir la relación original, si fuera necesario.

- A modo de ejemplo, la relación *cuenta* puede dividirse en varios fragmentos, cada uno de los cuales consiste en tuplas de cuentas que pertenecen a una sucursal concreta. Si el sistema bancario sólo tiene dos sucursales (Guadarrama y Cercedilla) habrá dos fragmentos diferentes: $cuenta1 = \sigma\ nombre\text{-}sucursal$ = «Guadarrama» (*cuenta*) $cuenta2 = \sigma\ nombre\text{-}sucursal$ = «Cercedilla» (*cuenta*) La fragmentación horizontal suele utilizarse para conservar las tuplas en los sitios en que más se utilizan, para minimizar la transferencia de datos. En general, los fragmentos horizontales pueden definirse como una *selección* de la relación global *r*.
- Es decir, se utiliza un predicado *Pi* para construir fragmentos *ri*:

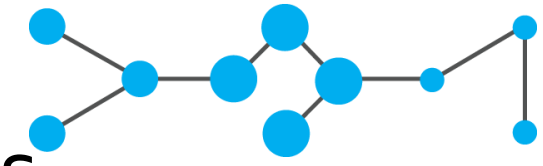
  $ri = \sigma Pi\ (r)$
- Se reconstruye la relación *r* tomando la unión de todos los fragmentos; es decir, $r = r1 \cup r2 \cup \cdots \cup rn$
- En el ejemplo los fragmentos son disjuntos. Al cambiar los predicados de selección empleados para crear los fragmentos se puede hacer que una tupla concreta de *r* aparezca en más de uno de los fragmentos *ri*. En su forma más sencilla la fragmentación vertical es igual que la descomposición.



BD Distribuidas fragmentacion

La **fragmentación vertical** de $r(R)$ implica la definición de varios subconjuntos de atributos $R1, R2, \ldots, Rn$ del esquema R de modo que $R = R1 \cup R2 \cup \cdots \cup Rn$

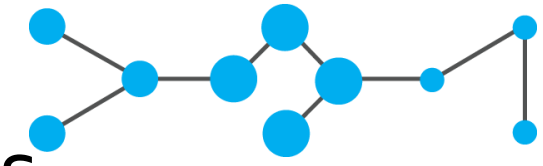
- Cada fragmento ri de r se define mediante $ri = \Pi Ri(r)$
- La fragmentación debe hacerse de modo que se pueda reconstruir la relación r a partir de los fragmentos tomando la reunión natural $r = r1\ r2\ r3 \cdots rn$
- Una manera de asegurar que la relación r pueda reconstruirse es incluir los atributos de la clave principal de R en cada uno de los fragmentos Ri. De manera más general, se puede utilizar cualquier superclave. Suele resultar conveniente añadir un atributo especial, denominado *id-tupla*, al esquema R. El valor id-tupla de una tupla es un valor único que distingue cada tupla de todas las demás. El atributo id-tupla, por tanto, sirve como clave candidata para el esquema aumentado y se incluye en cada uno de los fragmentos Ri. La dirección física o lógica de la tupla puede utilizarse como id-tupla, dado que cada tupla tiene una dirección única.
- Para ilustrar la fragmentación vertical considérese una base de datos universitaria con una relación *info-empleado* que almacena, para cada empleado, *id-empleado*, *nombre*, *puesto* y *salario*. Por motivos de preservación de la intimidad puede que esta relación se fragmente en una relación *empleado-infoprivada* que contenga *idempleado* y *salario*, y en otra relación *empleado-infopública* que contenga los atributos *id-empleado*, *nombre* y *puesto*. Puede que las dos relaciones se almacenen en sitios diferentes, nuevamente, por motivos de seguridad.



BD Distribuidas
Transparencia

No se debe exigir a los usuarios de los sistemas distribuidos de bases de datos que conozcan la ubicación física de los datos ni el modo en que se puede tener acceso a ellos en un sitio local concreto. Esta característica, denominada **transparencia de los datos,** puede adoptar varias formas:

- **Transparencia de la fragmentación.** No se exige a los usuarios que conozcan el modo en que se ha fragmentado la relación.
- **Transparencia de la réplica.** Los usuarios ven cada objeto de datos como lógicamente único. Puede que el sistema distribuido replique los objetos para incrementar el rendimiento del sistema o la disponibilidad de los datos. Los usuarios no deben preocuparse por los objetos que se hayan replicado ni por la ubicación de esas réplicas.
- **Transparencia de la ubicación.** No se exige a los usuarios que conozcan la ubicación física de los datos. El sistema distribuido de bases de datos debe poder hallar los datos siempre que la transacción del usuario facilite el identificador de los datos.



BD Distribuidas
Modos de fallo

Modos de fallo del sistema

Los sistemas distribuidos pueden sufrir los mismos tipos de fallos que los sistemas centralizados (por ejemplo, errores de software, errores de hardware y fallos de discos). No obstante, hay más tipos de fallos con los que hay que tratar en los entornos distribuidos. Los tipos básicos de fallos son:

- Fallo de un sitio
- Pérdida de mensajes
- Fallo de un enlace de comunicaciones
- División de la red

- La pérdida o deterioro de los mensajes siempre constituye una posibilidad en los sistemas distribuidos. El sistema utiliza protocolos de control de las transmisiones, como TCP/IP, para tratar esos errores. Se puede encontrar información sobre esos protocolos en los libros de texto estándar sobre redes (véanse las notas bibliográficas). No obstante, si dos sitios *A* y *B* no se hallan conectados de manera directa, los mensajes de uno a otro deben *encaminarse* mediante una serie de enlaces de comunicaciones. Si falla un enlace de comunicaciones los mensajes que se deberían haber transmitido por el enlace deben reencaminarse. En algunos casos resulta posible hallar otra ruta por la red de modo que los mensajes puedan alcanzar su destino. En otros casos el fallo puede hacer que no halla conexión entre algunos pares de sitios. Un sistema está **dividido** si se ha dividido en dos (o más) subsistemas, denominados **particiones**, que carecen de conexión entre ellas. Téngase en cuenta que, con esta definición, cada subsistema puede consistir en un solo nodo.

Gracias