

Bases de datos Distribuidas

no se utilizan mucho porque son caras

José R.R. Viqueira

Centro Singular de Investigación en Tecnoloxías Intelixentes (CITIUS)
Rúa de Jenaro de la Fuente Domínguez,
15782 - Santiago de Compostela.

Despacho: 209

Telf: 881816463

Mail: jrr.viqueira@usc.es

Skype: jrviqueira

URL: <https://citius.gal/team/jose-ramon-rios-viqueira>

Curso 2023/2024

- **Introducción**
 - ▷ Bases de datos homogéneas y heterogéneas
- **Almacenamiento distribuido**
- **Transacciones distribuidas**
- **Protocolos de compromiso**
- **Control de concurrencia**
- **Disponibilidad**
- **Procesamiento distribuido de consultas**
- **Bases de datos distribuidas heterogéneas**

Introducción



■ Bases de datos **Distribuidas** Vs Bases de datos **Paralelas**

tienen que ver con paralelizar las consultas, para ello, ...

▷ Paralelas: Procesadores pueden estar muy acoplados, dentro del mismo SGBDs

▷ Distribuidas: Nodos débilmente acoplados (no comparte componentes hardware)

cada nodo no comparte nada más que la red (habitualmente), todos juntos se comportan como un unico gestor
"se paraleliza el acceso a disco"

■ Distribución de los datos

▷ Necesaria para mejorar (**Volumen, Velocidad**)

– Rendimiento

cuando se mejora una se cae la otra (como norma general)

se mejoran

– Disponibilidad que este funcionando siempre

- consistencia: si tengo varias copias, que sean iguales

▷ Principal causa de las dificultades de su implementación

si tengo muchas copias, si se cae una tengo otra disponible para consultar, pero cuantas mas copias tenga mas posibilidad de inconsistencia

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Bases de datos Homogéneas Vs Bases de datos Heterogéneas

existen dos tipos

▷ Bases de datos Homogéneas

- Sistema Gestor de Bases de datos **idéntico** en cada nodo
- Cada nodo es consciente de la existencia de los demás
- Todos los nodos juntos se comportan como un único sistema, y con un esquema de datos único que se distribuye.
transparencia

En general
asumiremos
el caso de
BDs
homogéneas

▷ Bases de datos Heterogéneas (**Variedad**) sirve para atacar la variedad. No son tan típicas

- Cada nodo puede tener un software de acceso a datos distinto
- Cada nodo puede tener un esquema distinto en sus datos.
- Los nodos pueden no conocer la existencia de los demás
- Los nodo deben de poder operar con total independencia para atender a sus usuarios locales
- Se proporciona algún tipo limitado de cooperación entre los nodos
 - Ejemplo: Consulta integrada
 - Dificultad en el procesamiento de consultas debido a la existencia de esquemas distintos.

Introducción

Almacenam.



Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Dos soluciones para almacenar una relación de forma distribuida

▷ **Replicación**

- _ Varias copias de la misma tabla en varios nodos

▷ **Fragmentación**

- _ Dividir la tabla en pedazos y almacenar cada pedazo en un nodo.

▷ Se puede (suele) combinar la replicación y la fragmentación

- _ Cada fragmento se puede replicar en varios nodos

puede ser vertical u horizontal. Se puede almacenar unas filas en un nodo y otras en otro, o columnas. Ahora existe tambien el sharding, que fragmentación pero solo por filas !!

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Replicación de Datos

se recomienda mínimo 3 copias

- ▷ Cada relación almacenada en dos o más nodos.

replicación me da mayor disponibilidad, aumenta el paralelismo

- ▷ Ventajas y Desventajas

– Disponibilidad ↑ (en lectura)

crea problemas de inconsistencia (si lo pides antes de que se actualicen todas, ya que puede darte la lectura de una copia aun no actualizada o genera problemas de disponibilidad (en escritura) (esto ultimo si le pides inconsistencia, ya que espera a que se actualicen todas las tablas y, si falla una, da error)

- Aumenta. Si un nodo falla, los mismos datos pueden obtenerse de otro.

Paralelismo ↑

- Aumenta. Varias transacciones de lectura pueden acceder a la misma tabla en paralelo en distintos nodos.

- Aumenta también la probabilidad de tener los datos en local (mismo nodo donde se ejecuta la transacción)

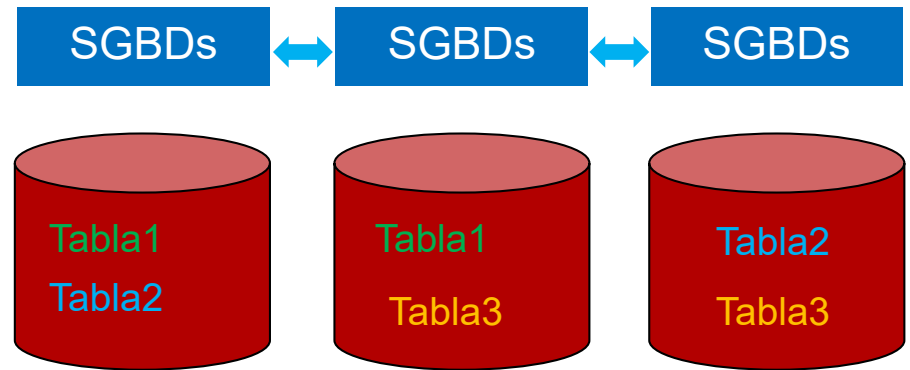
buena para leer pero mala para escribir: hay que actualizar todas las copias

Sobrecarga en las modificaciones ↑

- ▷ Buena para las lecturas, mala para las actualizaciones

es importante ver si mi aplicación se beneficia o no

- ▷ Elegir una réplica como **primaria** simplifica la gestión.

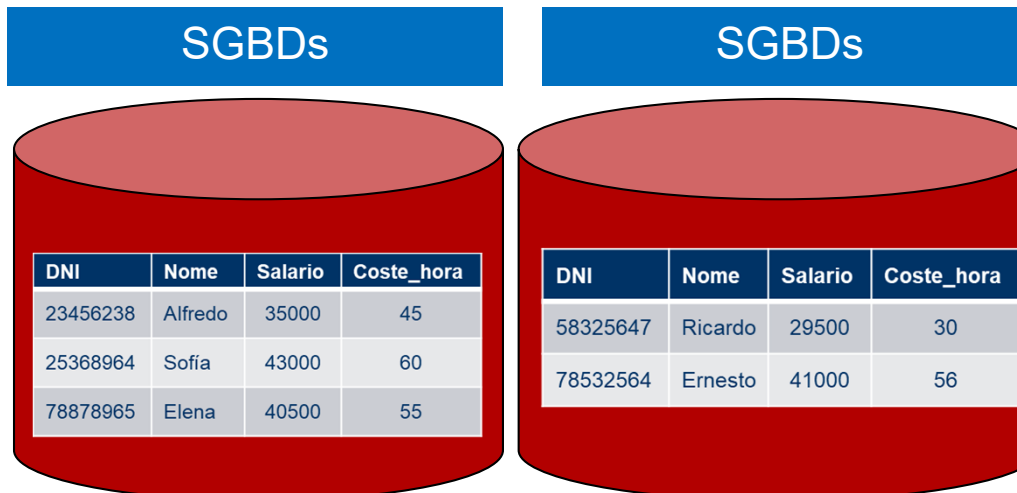


Fragmentación de Datos

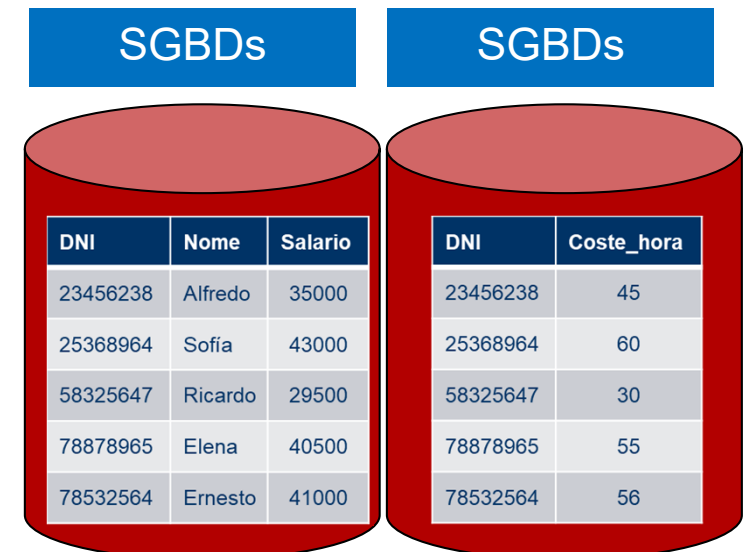
- ▷ **Fragmentación horizontal** sharding
 - Cada fila se envía a una partición.
 - La tabla se reconstruye usando la **Unión**
- ▷ **Fragmentación vertical**
 - Cada columna a una tabla, de manera que se pueda reconstruir usando un Join Natural
- ▷ Se pueden combinar
 - para que el sistema realmente escale, habría que aplicar la fragmentación horizontal, ya que las tablas aumentan por filas

DNI	Nome	Salario	Coste_hora
23456238	Alfredo	35000	45
25368964	Sofía	43000	60
58325647	Ricardo	29500	30
78878965	Elena	40500	55
78532564	Ernesto	41000	56

horizontal



Vertical



Introducción

Almacenam.



Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ **Transparencia** lo importante es tener transparencia

▷ De fragmentación y de replicación

- Los usuarios no necesitan saber como ha sido particionada o replicada una tabla para poder trabajar con ella.

▷ De localización

- Los usuarios no necesitan saber donde están físicamente almacenados los datos. El SGBDs debería de poder localizar los datos a través de sus identificadores.

Introducción

Almacenam.

Transacciones



Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Tipos de transacciones que el sistema debe soportar

▷ Transacciones locales

- _ Acceden y modifican datos solo en la base de datos local (la que recibe la transacción)
- _ ACID se garantiza con las técnicas de bases de datos centralizadas.
aprticionamiento importante → sharding

▷ Transacciones globales

- _ Acceden y modifican datos en varias bases de datos
- _ Garantizar ACID es mucho más complicado
 - Posibles fallos en cada base de datos o en las comunicaciones

aquí se puede caer la red en vez de
las máquinas, lo que causa más
problema que que se caigan las
máquinas

lo que hace el sistema para terminar una transacción: decide si se aborta o no

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Compromiso en arquitecturas centralizadas

- ▷ Se almacena un registro <Comprometida T> para la transacción T en el Registro Histórico
- ▷ El Registro Histórico se almacena en **almacenamiento estable**
 - _ Implementado con varias copias para asegurar la durabilidad



■ Compromiso en arquitecturas distribuidas

- ▷ Todos los SGBDs deben de coordinarse para decidir si la transacción se compromete o se aborta
 - _ No es aceptable un compromiso en algunos nodos y una cancelación en otros.
- ▷ **Coordinador de Transacciones** debe de ejecutar **un protocolo de compromiso**

Introducción

Almacenam.

Transacciones

Compromiso



Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Protocolo de compromiso en dos fases

▷ La transacción **T** se inicia en un Nodo (**N**). **C** es el coordinador de ese nodo **N**.

esto NO implica que finalice con éxito, todavía puede fallar, ya que aun no se ha volcado a disco

▷ Cuando **T** finaliza su ejecución, todos los nodos involucrados en su ejecución informan a **C** que **T** ha finalizado su ejecución. **C** inicia el protocolo.

solo se han ejecutado las instrucciones, pero todavía no se ha comprometido

Puede fallar aún.

▷ **Fase 1**

- **C** añade el registro **<preparar T>** al registro histórico (en almacenamiento estable)
- **C** envía el mensaje (**preparar T**) a todos los nodos involucrados.
- Cada gestor de transacciones de cada nodo decide si quiere comprometer o abortar **T**.
 - Abortar: Almacena **<no T>** en el registro histórico y envía (**abortar T**) al coordinador
 - Comprometer: Almacena **<listo T>** en el registro histórico y envía (**listo T**) al coordinador

Introducción

Almacenam.

Transacciones

Compromiso



Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Protocolo de compromiso en dos fases

▷ Fase 2

- _ **C** decide si **T** se compromete o se aborta, después de esperar un tiempo por las respuestas de los nodo involucrados.
- _ **C** añade **<comprometer T>** o **<abortar T>** al registro histórico
 - Después de almacenar este registro, **T** está comprometida o abortada.
aun no ha informado a otros nodos
 - Independientemente de lo que ocurra después.
- _ **C** envía mensajes (**comprometer T**) o (**abortar T**) a los nodos involucrados
- _ Cada nodo almacena **<comprometer T>** o **<abortar T>** en el registro histórico

Introducción

Almacenam.

Transacciones

Compromiso 

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Protocolo de compromiso en dos fases

▷ Gestión de los fallos

– Fallos de un nodo participante

- Detección de los fallos por parte del coordinador
 - Antes de que el nodo envíe (**listo T**), se asume que **T** abortará
 - Después de recibir (**listo T**), se asume que **T** se comprometerá.
- Después de reiniciar después del fallo, el nodo que falla examina el registro histórico
 - Si contiene **<comprometer T>**, ejecuta un **<rehacer T>** para asegurarse de que esta actualizado
 - Si contiene **<abortar T>**, ejecuta un **<deshacer T>**
 - Si contiene **<listo T>**, consulta al coordinador, o si el coordinador no responde a otros nodos.
 - En cualquier otro caso, ejecuta **<deshacer T>**

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Protocolo de compromiso en dos fases

▷ Gestión de los fallos

con el protocolo simple se sacrifica consistencia !!

_ Fallos en el coordinador

- Si el coordinador **C** falla en medio de la ejecución del protocolo para **T**, los nodos participantes han de decidir lo que se hace con **T**.
 - En algunos casos no van a poder, y deberán esperar a que el coordinador **C** se recupere
- Si algún nodo tiene en el registro histórico **<comprometer T>**, entonces **T** debe ser comprometida
- Si algún nodo tiene en su registro histórico **<abortar T>**, entonces **T** debe de ser abortada
- Si algún nodo no tiene **<listo T>** en el registro histórico, debería de abortarse también.
- En los demás casos, todos los nodos tienen **<listo T>**, pero nada más. No se puede conocer la decisión del coordinador hasta que este se reinicie. **T** estará bloqueada hasta que **C** se recupere.

Principal problema de este protocolo

para solucionar esto, surge el protocolo en tres fases

Introducción

Almacenam.

Transacciones

Compromiso



Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Protocolo de compromiso en dos fases

▷ Gestión de los fallos

- **Particionamiento de la red** (nodos divididos en dos o mas particiones que non pueden comunicarse entre si)
 - Si todos los nodos participantes y **C** están en la misma partición, el fallo no tiene efecto sobre el protocolo de compromiso de **T**.
 - Si algún nodo involucrado o **C** está aislado de los demás, los nodos de una partición (y el coordinador) creen que los demás han fallado.
 - Los nodos que no tienen al coordinador en su partición ejecutan el protocolo para tratar el fallo del coordinador.
 - Los nodos que tienen al coordinador en su partición, y el propio coordinador, siguen con el protocolo asumiendo fallo de los demás nodos.

Introducción

Almacenam.

Transacciones

Compromiso



Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Protocolo de compromiso en dos fases

▷ Recuperación y control de concurrencia

- _ Proceso de recuperación debe de tratar los **casos dudosos**
 - **<listo T>** en el registro histórico, pero no está **<comprometer T>** o **<abortar T>**
- _ No se puede continuar con el procesamiento de transacciones hasta resolver estos casos dudosos
 - Proceso que puede ser lento o muy lento
 - Puede afectar a otras transacciones concurrentes con **T**
 - Necesidad de soluciones específicas para estos casos, que van a depender del protocolo de control de concurrencia utilizado.

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia 

Disponibilidad

Pro. Consultas

BDs Hetero.

- Los protocolos de control de concurrencia utilizados necesitan adaptarse para funcionar en bases de datos distribuidas
 - ▷ Deben tener en cuenta la existencia de varias réplicas de cada elemento de datos.
 - ▷ Si el nodo de alguna réplica falla, ya no se pueden procesar modificaciones del elemento de datos.
 - Baja la disponibilidad
 - ▷ **Reto**
 - Continuar con el procesamiento de transacciones incluso si algunos de los nodos dejan de funcionar.

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia 

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Replicación con bajos niveles de consistencia

▷ Replicación Maestro-esclavo

- _ Modificaciones en el nodo primario
 - Se propagan automáticamente hacia los secundarios
- _ Lectura desde cualquier nodo
- _ En las lecturas en nodos secundarios, los datos pueden no estar actualizados, pero sí deberían ser consistente (**transaction-consistent snapshot** de los datos del primario).
 - contesta antes de actualizar
 - No podemos tener una versión de los datos en medio de una transacción
- _ Propagación de las modificaciones de primario: Inmediatamente o periódicamente.
- _ Se adapta muy bien a configuraciones con oficinas centrales y sucursales.
- _ Muy útil cuando tenemos consultas largas que no queremos que afecten al rendimiento de las transacciones.
 - Propagar los cambios por las noches por ejemplo.

Como un Data Warehouse, pero sobre los datos operacionales directamente

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia 



Disponibilidad

Pro. Consultas

BDs Hetero.

■ Replicación con bajos niveles de consistencia

▷ Replicación Multimaestro

- Se permiten modificaciones en cualquier réplica
 - Modificaciones propagadas automáticamente a las demás réplicas.
- Uso de **compromiso en dos fases** para realizar la modificación de las réplicas
- Alternativa: uso de **propagación perezosa (Lazy propagation)**, en lugar de actualizar las réplicas como parte de la propia transacción.
 - Permite el funcionamiento del procesamiento de transacciones incluso si algunos nodos fallan.
 - Aumenta **Disponibilidad** 
 - Baja la **Consistencia** 
 - Dos aproximaciones
 - Propagar modificaciones al primario directamente y al resto de replicas de forma perezosa.
 - Realizar propagación perezosa desde cada réplica a todas las demás.
Causa más problemas de concurrencia que la anterior.

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

- **Alta disponibilidad:** El SGBDs debe funcionar de forma ininterrumpida o casi.
- **Robustez:** Habilidad de continuar funcionando incluso con fallos.
 - ▷ Detectar fallos
 - ▷ Reconfigurar el sistema para seguir funcionando
 - ▷ Recuperar los componentes que fallaron (procesadores, discos, comunicaciones, etc.)
- Distintos tipos de fallos necesitan soluciones distintas.
 - ▷ No es posible distinguir entre fallos de nodos y particionamientos de red.
- Ejemplos de reconfiguraciones
 - ▷ Transacciones activas en nodos que fallan deben de ser abortadas. Cuando el nodo reinicia, debe asegurarse que tiene el último valor de cada réplica de cada elemento de datos.
 - ▷ Si un nodo que falla, el catálogo debe de ser informado para indicarle que sus réplicas ya no están disponibles.
 - ▷ Si falla un componente principal (coordinador, serv. nombres, etc.), se debe de elegir otro nodo para que asuma ese rol.

Introducción

Almacenam.

Transacciones



Compromiso

Concurrencia

Disponibilidad 

Pro. Consultas

BDs Hetero.

- Comparativa entre Sistema Remoto de Copia de Seguridad y Sistema Distribuido
 - ▷ Copia de Seguridad: Solo se copian datos y registro histórico
 - Menor coste 
 - ▷ Sistema distribuido: Sistemas de control de concurrencia y recuperación deben de funcionar en todos los nodos del sistema.
 - Mayor disponibilidad 
- Selección del coordinador
 - ▷ Opción 1: Mantener un nodo backup listo para asumir el rol de coordinador.
 - Alta disponibilidad
 - Sobrecarga producida por la ejecución doble en coordinador y nodo backup
 - ▷ Opción 2: Ejecutar un algoritmo de elección para que otro nodo asuma el rol de coordinador

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

- Compromiso entre Consistencia y Disponibilidad • • •
 - ▷ Típicamente, ^{la mitad mas uno} **mayoría** de los nodos con réplicas deben participar para que una modificación se realice
 - Si la red se particiona en más de dos partes, cada partición podría no tener la mayoría suficiente de nodos para seguir.
 - ▷ **Teorema CAP:** Un sistema solo puede tener dos de las siguientes
 - **Consistencia:** Mismo resultado que una ejecución secuencial en un solo nodo
 - **Disponibilidad:** Un nodo accesible, debe de responder a operaciones de lectura y escritura
 - **Tolerancia al particionamiento:** El sistema debe de seguir funcionando si hay particionamiento de red.
 - ▷ En un sistema distribuido, el ^{en general asumimos que asumimos la tercera !!!! y buscamos compromiso entre las otras dos, porque perderíamos mucha disponibilidad si hay que parar el sistema cada vez que falla un nodo} particionamiento de red es inevitable, y la tolerancia al mismo es necesaria (seguir funcionando si hay particionamiento)
 - Debemos sacrificar o consistencia o disponibilidad
 - ▷ Si permitimos modificaciones incluso si alguna réplica no es accesible, entonces tendremos una base de datos inconsistente (sube disponibilidad y baja la consistencia)

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas



BDs Hetero.

leer el mejor árbol de ejecución y ejecutarlo

- Aspectos a tener en cuenta para estimar el coste de una consulta
 - ▷ Acceso a disco (como en sistemas centralizados)
 - ▷ Coste de comunicaciones
 - ▷ Ganancia de rendimiento por ejecución paralela.
- Debido a las distintas combinaciones posibles de fragmentación (horizontal y/o vertical) y replicación, la optimización de consultas es mucho más complicada.
- En concreto, la optimización de las operaciones de Join es compleja y las diferencias entre unas estrategias u otras van a tener mucho más impacto en el rendimiento.
 - ▷ Mejorando mucho el rendimiento por la ejecución paralela
 - ▷ Empeorando mucho el rendimiento por la necesidad de mover datos entre nodos a través de la red.

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

■ Sistema multibase

TODO lo anterior es asumiendo que la base de datos es homogénea (cada nodo tiene el mismo software instalado y es consciente de la existencia de los otros)

► Capa de software encima de los SGBDs existentes

► Sistemas locales pueden tener

- Distintos modelos y lenguajes de definición y consulta de datos
 - Ejemplo: Relacional con SQL, XML con XQUERY, etc.
- Distintos mecanismos de control de concurrencia y recuperación.

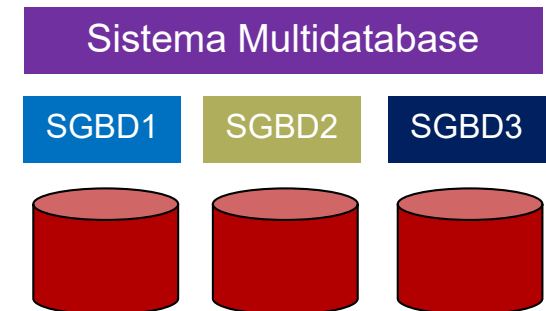
hay que dar la ilusión al usuario

► Ilusión de integración de datos a nivel lógico, sin necesitar integración física.

■ Problemas para integrar todo de forma física (en el mismo SGBDs distribuido)

- Dificultades técnicas: Necesidad de una **gran inversión**. Nuevo SGBDs y migrar todas las aplicaciones.
- Dificultades de organización: Dificultad de integrar físicamente todos los datos de varias organizaciones. En un sistema multibase, los sistemas locales mantienen un **alto grado de autonomía** y sus propias aplicaciones.

esto permite consultar fuentes en tiempo real, un datawarehouse no



Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.



■ Vista unificada de los datos

- ▷ El sistema multibase debe utilizar un **modelo de datos común**.
 - _ Por ejemplo: Relacional con SQL
 - Ejecución de consultas SQL sobre fuentes no relacionales
 - Ejemplo: **PostgreSQL Foreign Data Wrappers**
- ▷ Necesidad de proporcionar un esquema conceptual común
 - _ Problemas con la heterogeneidad semántica
 - Columnas con nombres iguales y significados distintos, etc.
 - Tipos de datos no soportados por algunos sistemas
 - Transformación de tipos puede ser complicada
 - Distintas unidades de medidas en los datos de algunos atributos
 - Diferencias en la semántica de los propios datos y no solo en los metadatos
 - Ejemplo: En un nodo el país puede llamarse "Greece" y en otro "Ellada".

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.

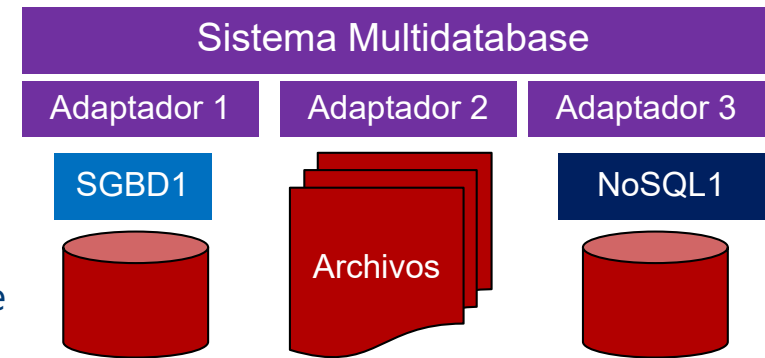
■ Procesamiento de consultas

▷ Problemas

- Consulta global ha de traducirse a consultas sobre los esquemas locales. Los resultados de cada subsistema han de traducirse al modelo común y unirse.
 - Tarea que se simplifica si se usan **Adaptadores (Wrappers)**
 - Vista global de datos locales. Traducen consultas y resultados.
- **Capacidades de consulta** de los distintos subsistemas pueden ser distintas.
 - Ejemplo: Un subsistema puede no permitir hacer joins, con lo que deben hacerse en la capa del sistema multibase
- Necesidad de **procesar los resultados** de cada sitio para unirlos con los demás (eliminación de duplicados, etc.)
- **Optimización global** de las consultas es muy compleja
 - Dificultad de conocer el coste de la ejecución de planes concreto en subsistemas concretos.
 - Solución: Optimización local + heurísticas a nivel global.

▷ Sistemas **Mediador**

- Solo proporcionan capacidades de consulta globales, y no transacciones.



Notaciones alternativas

- Sistemas multibase
- Sistemas mediador
- Bases de datos Virtuales

Introducción

Almacenam.

Transacciones

Compromiso

Concurrencia

Disponibilidad

Pro. Consultas

BDs Hetero.



■ Gestión de transacciones

- ▷ Tipos de transacciones
 - **Locales:** Se ejecutan en cada sistema de bases de datos
 - **Globales:** Se ejecutan bajo el control del sistema multibase
- ▷ Al mantener la autonomía de cada subsistema, el sistema multibase no puede saber que transacciones se están ejecutando en cada subsistema
 - Sincronización de estas ejecuciones no es posible (control de concurrencia)
- ▷ Simplifica imponer restricciones como que las transacciones globales solo puedan ser de lectura

Bases de datos Distribuidas

Capítulo 19: Bases de datos distribuidas. A. Silberschatz,
H.F. Korth, S. Sudarshan, Database System Concepts, 6th
Edition, McGraw-Hill, 2014

José R.R. Viqueira

Centro Singular de Investigación en Tecnoloxías Intelixentes (CITIUS)
Rúa de Jenaro de la Fuente Domínguez,
15782 - Santiago de Compostela.

Despacho: 209

Telf: 881816463

Mail: jrr.viqueira@usc.es

Skype: jrviqueira

URL: <https://citius.gal/team/jose-ramon-rios-viqueira>

Curso 2023/2024