Diseño y Optimización de Bases de Datos:

Bases de Datos Distribuidas

Departamento de O.E.I. - U.P.M.

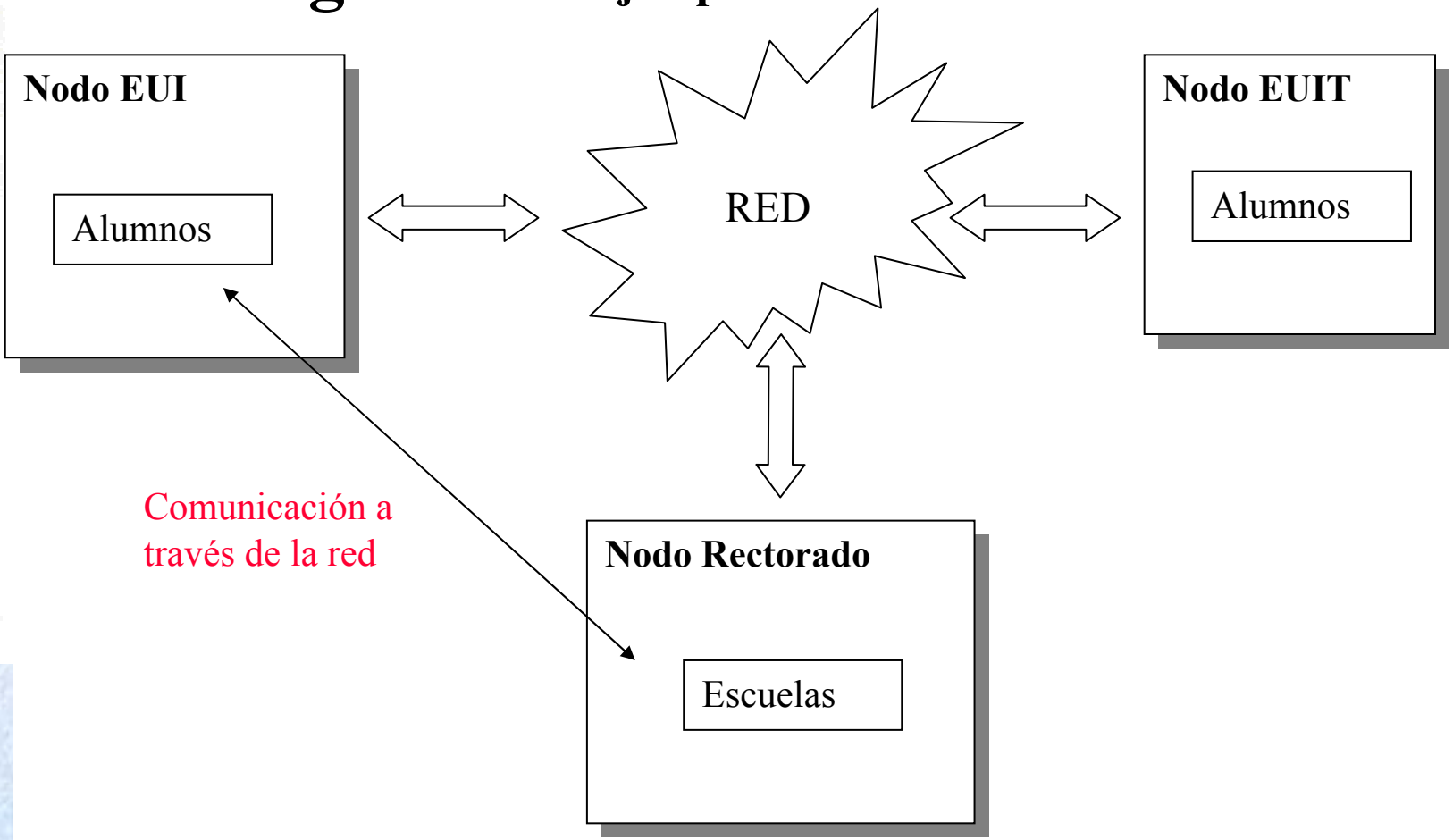
ÍNDICE

1. Introducción

- **Visión general**
  - Los datos se encuentran en diferentes máquinas, generalmente situados en localizaciones geográficas diferentes (homogéneas o no).
  - *Nodo* o *emplazamiento*: Cada uno de los ordenadores que integran el sistema de Bases de Datos distribuido.
  - Tipos de transacciones:
    - Locales: cuando se accede a los datos del único emplazamiento donde se inició la transacción.
    - Globales: Cuando se accede a datos de emplazamientos distintos al nodo donde se inició la transacción.

1. Introducción

- **Visión general.** **Ejemplo de BD distribuida:**



1. Introducción

- **Visión general.** **Ejemplo de BD distribuida:**
- **Nodos de las Escuelas:**

DNI	Escuela	Nombre	Nota ingreso	Beca

- **Nodo del Rectorado:**

Escuela	Situación	Número alumnos

- **Nuevo alumno en la secretaría del centro: transacción local.**
- **Nuevo alumno en el rectorado: transacción global**

1. Introducción

- **Visión general.** **Ejemplo de BD distribuida:**
- **Este sistema será distribuido si cumple que:**
  - Los distintos nodos están informados sobre los demás.
  - Aunque algunas tablas estén almacenadas sólo en algunos nodos, éstos comparten un esquema global común.
  - Cada nodo proporciona un entorno de ejecución de transacciones, tanto local, como global.
  - Generalmente, los nodos ejecutan el mismo software de gestión distribuida. En caso contrario se dice que el sistema es *heterogéneo*.

1. Introducción

- *Ventajas de las Bases de Datos Distribuidas*
  - **Compartimiento de datos**. Los usuarios de un nodo son capaces de acceder a los datos de otro nodo. Por ejemplo, desde el Rectorado, se puede consultar los datos de los alumnos de Informática.
  - **Autonomía**. Cada nodo tiene cierto grado de control sobre sus datos, en un sistema centralizado, hay un administrador del sistema responsable de los datos a nivel global. Cada administrador local puede tener un nivel de autonomía local diferente.
  - **Disponibilidad**. Si en un sistema distribuido falla un nodo, los nodos restantes pueden seguir funcionando. Si se duplican los datos en varios nodos, la transacción que necesite un determinado dato puede encontrarlo en cualquiera de los diferentes nodos.

1. Introducción

- *Inconvenientes de la Bases de Datos Distribuidas*
  - **Coste de desarrollo del software**. La complejidad añadida que es necesaria para mantener la coordinación entre nodos hace que el desarrollo de software sea más costoso.
  - **Mayor probabilidad de errores**. Como los nodos que constituyen el sistema funcionan en paralelo, es más difícil asegurar el funcionamiento correcto de los algoritmos, así como de los procedimientos de recuperación de fallos del sistema.
  - **Mayor sobrecarga de procesamiento**. El intercambio de mensajes y ejecución de algoritmos para el mantenimiento de la coordinación entre nodos supone una sobrecarga que no se da en los sistemas centralizados.

2. Almacenamiento distribuido de datos

- **Réplica:**
  - El sistema conserva varias copias o réplicas idénticas de una tabla. Cada réplica se almacena en un nodo diferente.
  - Ventajas:
    - **Disponibilidad**: el sistema sigue funcionando aún en caso de caída de uno de los nodos.
    - **Aumento del paralelismo**: Varios nodos pueden realizar consultas en paralelo sobre la misma tabla. Cuantas más réplicas existan de la tabla, mayor será la posibilidad de que el dato buscado se encuentre en el nodo desde el que se realiza la consulta, minimizando con ello el tráfico de datos entre nodos.

2. Almacenamiento distribuido de datos

- **Réplica:**
  - Inconveniente:
    - **Aumento de la sobrecarga en las actualizaciones**: El sistema debe asegurar que todas las réplicas de la tabla sean consistentes. Cuando se realiza una actualización sobre una de las réplicas, los cambios deben propagarse a todas las réplicas de dicha tabla a lo largo del sistema distribuido.

2. Almacenamiento distribuido de datos

- *Fragmentación de datos*
  - **Fragmentación horizontal:**
    - Una tabla T se divide en subconjuntos, T_1, T_2, ...T_n. Los fragmentos se definen a través de una operación de selección y su reconstrucción se realizará con una operación de unión de los fragmentos componentes.
    - Cada fragmento se sitúa en un nodo.
    - Pueden existir fragmentos no disjuntos: combinación de fragmentación y replicación.

2. Almacenamiento distribuido de datos

- *Fragmentación de datos*
  - **Fragmentación horizontal:** Ejemplo.
    - Tabla inicial de alumnos de la UPM (T)

DNI	Escuela	Nombre	Nota ingreso	Beca
87633483	EUI	Concha Queta	5.6	No
99855743	EUI	Josechu Letón	7.2	Si
33887293	EUIT	Oscar Romato	6.1	Si
05399075	EUI	Bill Gates	5.0	No
44343234	EUIT	Pepe Pótamo	8.0	No
44543324	EUI	Maite Clado	7.5	Si
66553234	EUIT	Ernesto Mate	6.6	No

2. Almacenamiento distribuido de datos

- *Fragmentación de datos*
  - **Fragmentación horizontal:** Ejemplo.

Fragmento de la EUI: $\sigma_{Escuela="EUI"}(T)$

DNI	Escuela	Nombre	Nota ingreso	Beca
87633483	EUI	Concha Queta	5.6	No
99855743	EUI	Josechu Letón	7.2	Si
05399075	EUI	Bill Gates	5.0	No
44543324	EUI	Maite Clado	7.5	Si

Fragmento de la EUIT: $\sigma_{Escuela="EUIT"}(T)$

DNI	Escuela	Nombre	Nota ingreso	Beca
33887293	EUIT	Oscar Romato	6.1	Si
44343234	EUIT	Pepe Pótamo	8.0	No
66553234	EUIT	Ernesto Mate	6.6	No

2. Almacenamiento distribuido de datos

- *Fragmentación de datos*
  - **Fragmentación vertical:**
    - Una tabla T se divide en subconjuntos, T_1, T_2, ...T_n. Los fragmentos se definen a través de una operación de proyección.
    - Cada fragmento debe incluir la clave primaria de la tabla. Su reconstrucción se realizará con una operación de join de los fragmentos componentes.
    - Cada fragmento se sitúa en un nodo.
    - Pueden existir fragmentos no disjuntos: combinación de fragmentación y replicación.

2. Almacenamiento distribuido de datos

- *Fragmentación de datos*
  - **Fragmentación vertical:** Ejemplo

Datos Rectorado (R)

Escuela	Situación	Número alumnos
EUI	Campus sur	3000
EUIT	Campus sur	2800
TOPOGRAFIA	Campus sur	800
ETSIT	Ciudad Universitaria	2500
FI	Campus Montegancedo	2100

$\Pi_{\text{Escuela,Situación}}(R)$

Departamento infraestructura

Escuela	Situación
EUI	Campus sur
EUIT	Campus sur
TOPOGRAFIA	Campus sur
ETSIT	Ciudad Universitaria
FI	Campus Montegancedo

$\Pi_{\text{Escuela,Número\_alumnos}}(R)$

Departamento ordenación académica

Escuela	Número alumnos
EUI	3000
EUIT	2800
TOPOGRAFIA	800
ETSIT	2500
FI	2100

2. Almacenamiento distribuido de datos

- *Fragmentación de datos*
  - **Fragmentación Mixta:** Ejemplo

Secretaría

DNI	Escuela	Nombre	Beca
87633483	EUI	Concha Queta	No
99855743	EUI	Josechu Letón	Si
05399075	EUI	Bill Gates	No
44543324	EUI	Maite Clado	Si

$\Pi_{DNI,Escuela,Escuela,Nombre,Beca}(E)$

Datos EUI (E)

DNI	Escuela	Nombre	Nota ingreso	Beca
87633483	EUI	Concha Queta	5.6	No
99855743	EUI	Josechu Letón	7.2	Si
05399075	EUI	Bill Gates	5.0	No
44543324	EUI	Maite Clado	7.5	Si

Resultante de fragmentación horizontal previa

$\Pi_{DNI,Escuela,Nombre,Nota\ ingreso}(E)$

Jefatura estudios

DNI	Escuela	Nombre	Nota ingreso
87633483	EUI	Concha Queta	5.6
99855743	EUI	Josechu Letón	7.2
05399075	EUI	Bill Gates	5.0
44543324	EUI	Maite Clado	7.5

2. Almacenamiento distribuido de datos

- *Réplica y fragmentación de datos*
  - **Las técnicas de réplica y fragmentación se pueden aplicar sucesivamente a la misma relación de partida. Un fragmento se puede replicar y a su vez esa réplica ser fragmentada, para luego replicar alguno de esos fragmentos.**

3. Transparencia de la red

- **Se apoya en los siguientes aspectos:**
  - La denominación de los elementos de datos.
  - La réplica de los elementos de datos.
  - La fragmentación de los elementos de datos.
  - La ubicación de los fragmentos y las réplicas.

3. Transparencia de la red

- *Denominación de los elementos de datos*
  - Elementos de datos: **relaciones**, **fragmentos** y **réplicas**.
  - Nombre único para cada elemento.
  - Denominación centralizada de elementos
    - Servidor de nombres central en el que se registren los elementos.
    - Servidor de nombres que direcciona los elementos.
      Problemas:
      - Cuello de botella para acceder a los nombres de los elementos.
      - Si falla, es posible que ningún nodo siga funcionando.

3. Transparencia de la red

- *Denominación de los elementos de datos*
  - Denominación distribuida de fragmentos:
    - Cada elemento tiene como prefijo el nodo en el que se encuentra.
    - Se pierde transparencia de la red.
    - Sistema de alias para recuperar la transparencia. Las tablas de conversión de alias deben estar en todos los nodos.
    - Cada fragmento y réplica debe ser identificado de forma única mediante sufijos. **Ejemplo:**
    - EUI.ALUMNOS.f3.r2 hace referencia a la réplica 2 del fragmento 3 del elemento ALUMNOS del nodo EUI.

Existirá una tabla en el catálogo que permita al sistema determinar en qué fragmentos o réplicas están los datos que el usuario solicita y mantendrá las réplicas actualizadas cuando se produzca una sentencia que modifique los datos.

4. Procesamiento distribuido de consultas

- **Se estudia el coste de las comunicaciones.**
- **Objetivo: la** reducción de la cantidad de datos transferidos**.**
- **Optimización mediante operación de *semijoin*.**

4. Procesamiento distribuido de consultas

- **Ejemplo de consulta distribuida**

NODO1: EMPLEADO

Nombre	Apellido	COD	Dir	Sexo	Sueldo	fecha Nac.	Dpto.

10.000 tuplas.

Cada tupla tiene 100 bytes de longitud.

El campo COD tiene 9 bytes de longitud.

El campo Dpto tiene 4 bytes de longitud.

El campo Nombre tiene 15 bytes de longitud.

El campo Apellido tiene 15 bytes de longitud.

Tamaño de la relación: $100 * 10.000 = 10^6$ bytes

4. Procesamiento distribuido de consultas

- **Ejemplo de consulta distribuida**

NODO2: DEPARTAMENTO

NombreDpto	NDpto	Responsable	Edificio

100 tuplas.

Cada tupla tiene 35 bytes de longitud.

El campo NombreDpto tiene 10 bytes de longitud.

El campo NDpto tiene 4 bytes de longitud.

El campo Responsable tiene 9 bytes de longitud.

Tamaño de la relación: 35 * 100 = 3500 bytes

4. Procesamiento distribuido de consultas

- **Ejemplos de consulta distribuida:**
  - "*Por cada empleado, obtener el nombre del empleado y el nombre del departamento al que pertenece*"
  - Q1(1): $\Pi_{Nombre,Apellido,NombreDPto}$(EMPLEADO*DEPARTAMENTO)
  - La consulta se lanza desde el nodo 3 (***nodo respuesta***) que no tiene datos implicados en la consulta.
  - El resultado de ésta consulta constará de 10.000 tuplas. Cada tupla resultante será de una longitud de 40 bytes. El tamaño del resultado será por tanto de 400.000 bytes.
  - Existen tres alternativas para resolver la consulta.

(1) Query (Q): Identificador de Consulta

4. Procesamiento distribuido de consultas

- **Ejemplos de consulta distribuida**
  - Primera alternativa:

    Transferir, tanto la relación EMPLEADO, como la relación DEPARTAMENTO al nodo respuesta (nodo 3) y realizar allí mismo la operación de join. En éste caso se transfieren:

    1.000.000 + 3.500 = 1.003.500 bytes.

4. Procesamiento distribuido de consultas

- **Ejemplos de consulta distribuida**
  - Segunda alternativa:

    Transferir la relación EMPLEADO al nodo 2, ejecutar el join en este nodo y enviar el resultado al nodo 3. Esto implicaría transferir:

    1.000.000 + 400.000 (resultado) = 1.400.000 bytes

4. Procesamiento distribuido de consultas

- **Ejemplos de consulta distribuida**
  - Tercera alternativa:

    Transferir la relación DEPARTAMENTO al nodo 1, ejecutar el join en este nodo y enviar el resultado al nodo 3. En este caso, los bytes transferidos serán:

    3.500 + 400.000 (resultado) = 403.500 bytes.

4. Procesamiento distribuido de consultas

- **Ejemplos de consulta distribuida**
  - "*Para cada departamento, obtener el nombre del departamento y el de su director*"
  - Q2: $\Pi_{NombreDPto,Nombre,Apellido}$(DEPARTAMENTO * EMPLEADO)
  - La consulta se lanza desde el nodo 3. El resultado de ésta consulta constará de 100 tuplas (4.000 bytes).
  - Opción 1: transferimos las relaciones DEPARTAMENTO y EMPLEADO al nodo 3. Se transfieren:

    3.500 + 1.000.000 = 1.003.500 bytes.
  - Opción 2: transferimos la relación EMPLEADO al nodo 2 y enviamos el resultado del join al nodo 3. Se transfieren:

    1.000.000 + 4.000 = 1.004.000 bytes.
  - Opción 3: transferimos la relación DEPARTAMENTO al nodo 1 y enviamos el resultado del join al nodo 3. Se transfieren en este caso:

    3.500 + 4.000 = 7.500 bytes.

4. Procesamiento distribuido de consultas

- **Ejemplos de consulta distribuida**
  - NUEVO SUPUESTO: las consultas anteriores se lanzan desde el **nodo 2**.
    - Opción 1: transferir la relación EMPLEADO al nodo 2, realizar el join y presentar el resultado al usuario del nodo 2. De ésta manera se transferirán el mismo número de bytes para la consulta Q1 y la Q2: 1.000.000 bytes.
    - Opción 2: transferir la relación DEPARTAMENTO al nodo 1, realizar el join y enviar el resultado al nodo 2. En este caso se transfieren:
      - Para la consulta Q1: 3.500 de DEPARTAMENTO y 400.00 de resultado = **403.500 bytes.**
      - Para la consulta Q2: 3.500 de DEPARTAMENTO y 4.000 de resultado = **7.500 bytes.**

La segunda opción es más optima: origen del *semijoin*.

4. Procesamiento distribuido de consultas

- **Proceso distribuido de consultas utilizando semijoin**
  - Reducción de el número de tuplas antes de ser transferidas a otro nodo.
  - Se envía la columna con la que se va a realizar el join de una relación R al nodo donde se encuentra la otra relación, allí se realiza el join con la otra relación S
  - Se envían las columnas implicadas en el resultado al nodo inicial y se vuelve a realizar el join con R.
  - Sólo se transfieren las columnas de R que intervienen en la realización del join en una dirección y el subconjunto de columnas de S resultantes en la otra.

4. Procesamiento distribuido de consultas

- **Proceso distribuido de consultas utilizando semijoin**
  - **Semijoin de las consultas Q1 y Q2. Paso 1.**

Consulta Q1: proyección en DEPARTAMENTO sobre atributos que van a intervenir en la operación de join y transferencia al nodo 1.

F1: Π_{NDpto}(DEPARTAMENTO).

Tamaño resultante: 4 bytes del atributo NDpto por 100 tuplas de DEPARTAMENTO = 400 bytes transferidos.

Consulta Q2: F2: $\Pi_{Responsable}$(DEPARTAMENTO).

Tamaño resultante: 9 bytes del atributo Responsable por 100 tuplas de DEPARTAMENTO = 900 bytes transferidos.

4. Procesamiento distribuido de consultas

- **Proceso distribuido de consultas utilizando semijoin**
  - **Semijoin de las consultas Q1 y Q2:**

Paso 2.

Consulta Q1: realización del join de los tuplas transferidas en el paso anterior. Transferencia del resultado del join de nuevo al nodo 1. Se transfieren sólo los atributos necesarios para realizar el join final:

$$\mathbf{R1: \Pi_{Dpto,Nombre,Apellido}(F * EMPLEADO)}$$

Tamaño: (4 + 15 + 15) * 10.000 = 340.000 bytes transferidos.

Consulta Q2: $\mathbf{R2: \Pi_{Responsable,Nombre,Apellido}(F' * EMPLEADO)}$

Tamaño: (9+ 15 + 15) * 100 = 3900 bytes transferidos.

Total transferido: 340.400 bytes para Q1 y 4.800 bytes para Q2.

Recuperación

- **Fallo de los nodos.**
  - **Cuando un nodo falla, el sistema deberá continuar trabajando con los nodos que aún funcionan. Si el nodo a recuperar es una base de datos local, se deberán separar los datos entre los nodos restantes antes de volver a unir de nuevo el sistema.**
- **Copias múltiples de fragmentos de datos.**
  - **El subsistema encargado del control de concurrencia es el responsable de mantener la consistencia en todas las copias que se realicen y el subsistema que realiza la recuperación es el responsable de hacer copias consistentes de los datos de los nodos que han fallado y que después se recuperarán.**

Recuperación

- **Transacción distribuida correcta.**
  - **Se pueden producir fallos durante la ejecución de una transacción correcta si se plantea el caso de que al acceder a alguno de los nodos que intervienen en la transacción, dicho nodo falla.**
- **Fallo de las conexiones de comunicaciones.**
  - **El sistema debe ser capaz de tratar los posibles fallos que se produzcan en las comunicaciones entre nodos. El caso mas extremo es el que se produce cuando se divide la red. Esto puede producir la separación de dos o más particiones donde las particiones de cada nodo pueden comunicarse entre si pero no con particiones de otros nodos.**

5. Transacciones distribuidas

- **Protocolo de compromiso en dos fases.** ***(Two phase commit protocol)***

1. *El coordinador envía una solicitud de voto (vote request) a los nodos participantes en la ejecución de la transacción.*
2. *Cuando los participantes reciben la solicitud de voto, responden enviando al coordinador un mensaje con su voto (Sí o No). Si un participante vota No, la transacción se aborta (abort).*
3. *El coordinador recoge los mensajes con los votos de todos los participantes. Si todos han votado Sí, entonces el coordinador también vota si y envía un mensaje commit a todos los participantes. En otro caso, el coordinador decide abandonar y envía un mensaje abort a todos los participantes que han votado afirmativamente.*
4. *Cada participante que ha votado sí, espera del coordinador un mensaje commit o abort para terminar la transacción de forma normal o abortarla.*