Bases de datos No SQL

SISTEMAS DISTRIBUIDOS

Sistemas distribuidos I

- Conjunto de elementos (o nodos) de **procesamiento autónomos** (capaces de ejecutar programas), **no necesariamente homogéneos**, que **cooperan** en la realización de las tareas que tienen asignadas.
- Están interconectados por una red de comunicaciones.
- Desde el punto de vista del usuario, el sistema se percibe como una sola unidad.
- Se puede distribuir/paralelizar:
  - La ejecución de código (ej. consultas a varios nodos en paralelo)
  - La funcionalidad
  - El almacenamiento de datos
  - El control de la ejecución de tareas (ej. nodo distribuidor)

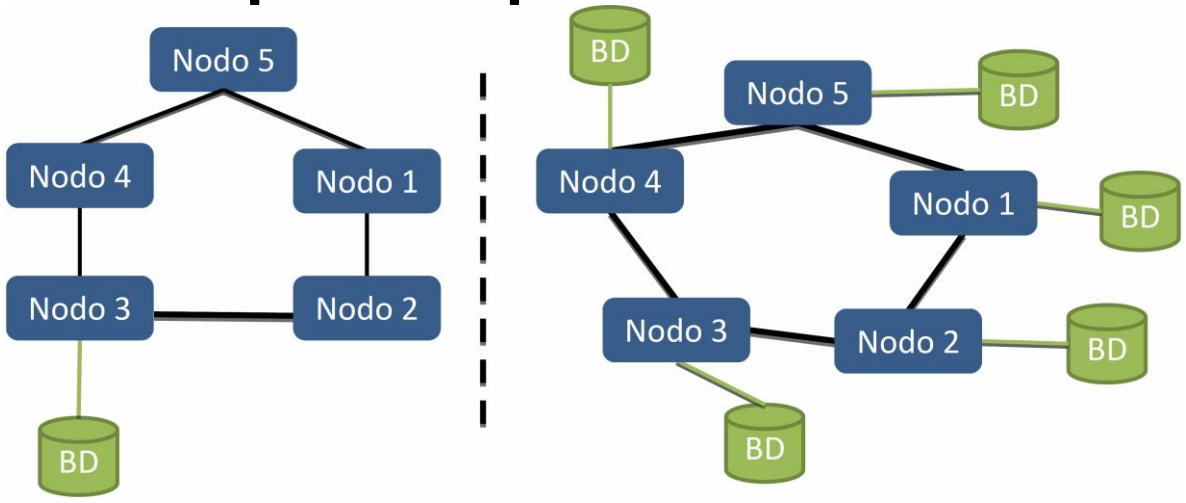
Sistemas distribuidos II

- Se ajustan a la estructura actual de muchas organizaciones que se encuentran geográficamente distribuidas a nivel intranacional o internacional. Ej. Venta online. Ej. LOPD en España (usuarios en servidores ubicados en España).
- Permiten mejorar el rendimiento (paralelización), la disponibilidad y la escalabilidad (capacidad de crecimiento).
- Aumentan los retos a la hora de crear aplicaciones distribuidas como, por ejemplo, la necesidad de coordinación, la depuración de errores o la resolución de los problemas que se derivan de situaciones de fallo.

BASES DE DATOS DISTRIBUIDAS

Bases de datos distribuidas

- Es un conjunto de múltiples bases de datos, cuyos **datos están lógicamente interrelacionados** (en el mismo dominio semántico), que están distribuidas en una red de ordenadores.
- Están gestionadas por un software específico (capa) que hace que la distribución sea **transparente para los usuarios**.



Base de datos centralizada

Base de datos distribuida

Tipos de bases de datos distribuidas

- La evolución de una base de datos centralizada a distribuida se puede producir por:
  - **Distribución impuesta**, por cuestiones de integración de sistemas diseñados de forma independiente (*legacy systems*), operando de forma autónoma (ej. Caixa y Bankia). En ocasiones se les llama **bases de datos federadas** o **multi-bases de datos**. Este enfoque se denomina ***bottom-up***, porque parte de la base de que las bases de datos a "fusionar" ya existen previamente. No se consideran bases de datos distribuidas.
  - **Distribución deseada**, por el diseño de la base de datos, con previsión de grandes volúmenes de datos y de operaciones, así como la ubicación dispersa de los usuarios. Este enfoque se denomina ***top-down***.

Ventajas y retos de los sistemas gestores de bases de datos distribuidas

- Semejantes a los de los sistemas distribuidos.
- Hacen **transparente al usuario** la existencia de una **red de nodos** y que los **datos estén dispersos**, así como la **replicación**. Esto facilita el desarrollo de apps.
- Se **aumenta** el **rendimiento**, la **fiabilidad** y la **disponibilidad de los datos**, gracias a la reducción del acceso remoto a los datos y a la ejecución de operaciones distribuidas.
- **Facilita la extensión** (crecimiento) de la BD gracias a su descomposición en diferentes componentes.
- A pesar de las ventajas, hay que tener el en cuenta **dificultades**, como el diseño de la BD, el **mantenimiento de la integridad de los datos** (por ejemplo, la coherencia de datos replicados), la **coordinación para la ejecución de las operaciones**, la **resolución de situaciones de fallo**…

DISEÑO DE BASES DE DATOS DISTRIBUIDAS

Diseño de bases de datos distribuidas

- Toma decisiones acerca de cómo fragmentar la base de datos, qué fragmentos se deben replicar y dónde se almacenarán los fragmentos (y sus réplicas, si las hubiera).
- Para esto es necesario analizar las necesidades de los usuarios/aplicaciones:
  - A qué parte de la base de datos necesitan acceder.
  - Qué operaciones (funcionalidades) realizan sobre la base de datos.
  - Cuál es su punto de acceso al sistema
- Los objetivos a conseguir son:
  - Disminuir los costes de transmisión de datos a través de la red.
  - Repartir la carga de trabajo uniformemente entre los nodos.
  - Aumentar la eficiencia en el procesamiento de las solicitudes y garantizar la disponibilidad de los datos.
- Idealmente, la existencia de fragmentos y sus réplicas, y su lugar de almacenamiento, deben ser transparentes al usuario. Esto se llama **transparencia de distribución**.

ESTRATEGIAS DE DISTRIBUCIÓN

Estrategias de distribución

- **Fragmentación**
  - Los datos/esquema se dividen en subconjuntos. Hay tres tipos:
    - **Horizontal**, toma como unidad de distribución conjuntos de datos relacionados.
    - **Vertical**, usa subconjuntos del esquema (y los datos asociados) como unidad de distribución, por ejemplo, grupos de atributos.
    - **Híbrida**, mezcla de las dos anteriores.
- **Replicación**
  - Consiste en repetir (almacenar varias veces) fragmentos del esquema, y de los datos que estos contienen.

Fragmentación

- Las aplicaciones (sus usuarios) no suelen necesitar acceder a toda la base de datos, sino que están interesados en un subconjunto de la misma.
- El objetivo de la fragmentación es encontrar la unidad de acceso a los datos más apropiada para cada dominio de aplicación. Dicha unidad de acceso o fragmento se convierte en la unidad de distribución entre los diferentes nodos que forman la base de datos distribuida.
- Existen tres tipos de fragmentación:
  - **Horizontal**: toma conjuntos de datos relacionados como unidad de distribución, de tal manera que objetos individuales o subconjuntos de objetos que comparten semántica (es decir, objetos que pertenecen a una misma clase) pasan a ser la unidad de distribución entre los diferentes nodos
  - **Vertical**: consiste en usar subconjuntos del esquema, en general, grupos de atributos de los objetos (y los datos asociados a ellos) como unidad de distribución.
  - **Híbrida**: combina la fragmentación horizontal y vertical.

Replicación

- Otra estrategia de distribución es la replicación.
- Consiste en almacenar todos o una parte de los fragmentos diseñados (y por lo tanto, los datos que éstos contienen) en más de un nodo.
- **Las estrategias de fragmentación y replicación son ortogonales entre sí**, es decir, se podrá:
  - **Usar únicamente fragmentación** (la base de datos se divide en fragmentos que se distribuyen entre los diferentes nodos, pero ningún fragmento estará replicado).
  - No fragmentar la base de datos pero sí **replicarla** (si se replica de forma completa estaríamos hablando de **técnicas de *mirroring* de bases de datos**),
  - Utilizar una **combinación de ambas** estrategias. Esta última acostumbra a ser la elección habitual en el caso de una base de datos distribuida.

Comparación entre estrategias de distribución

FRAGMENTACIÓN	REPLICACIÓN
+ Permite acercar los datos allí donde se necesitan más: **localidad de los datos** (*data locality*).	+ Permite acercar los datos allá donde se necesitan más: **localidad de los datos** (*data locality*).
+ Incrementa el nivel de **concurrencia** (facilita el paralelismo).	+ Incrementa la **disponibilidad de los datos** en situaciones de fallo (nodo caído o innaccesible).
- El **rendimiento empeora** cuando las operaciones necesitan recuperar fragmentos almacenados en diferentes nodos. **Lo ideal** sería que **cada usuario sólo necesitase acceder a un nodo**, y que **el acceso de los usuarios se repartiese de forma uniforme entre todos los nodos** (equilibrio de carga de trabajo).	+ Permite **mejorar** la eficiencia de las operaciones de consulta (**lectura**).
- El control semántico de los datos se puede complicar: restricciones de integridad.	- **Empeora** la inserción de datos y su modificación (**escritura**) es menos eficiente.
	- **Compromete la consistencia**: todas las réplicas de un mismo dato deben ser idénticas.

Fragmentación horizontal I

- Muy adecuada para organizaciones geográficamente dispersas que esencialmente acceden a los datos que guardan localmente (maximiza la *data locality*).
- Para decidir qué fragmentos crear, se analizan los filtros o condiciones aplicados en las operaciones (consultas) que formulan los usuarios/aplicaciones.
- Toma como unidad de distribución objetos individuales o subconjuntos de objetos. Estos subconjuntos de objetos son instancias de una misma clase.
- Los subconjuntos se suelen establecen en función del valor de un atributo o grupo de atributos del objeto.

Fragmentación horizontal II

La relación Cliente se divide en tres fragmentos en función del valor del atributo paísCliente.

Cliente

idCliente	nombreCliente	paísCliente	totalFacturado	numProyectos	idComercial
c1	FX	Francia	30000	3	com1
c2	FW	Francia	100000	5	com2
c3	FT	Francia	50000	1	com1

idCliente	nombreCliente	paísCliente	totalFacturado	numProyectos	idComercial
c4	UX	USA	6000	1	com10
c5	UW	USA	12000	2	com23

idCliente	nombreCliente	paísCliente	totalFacturado	numProyectos	idComercial
c6	EX	España	100000	4	com1
c7	EW	España	40000	2	com2
c8	ET	España	25000	1	com1
c9	EZ	España	12000	8	com1

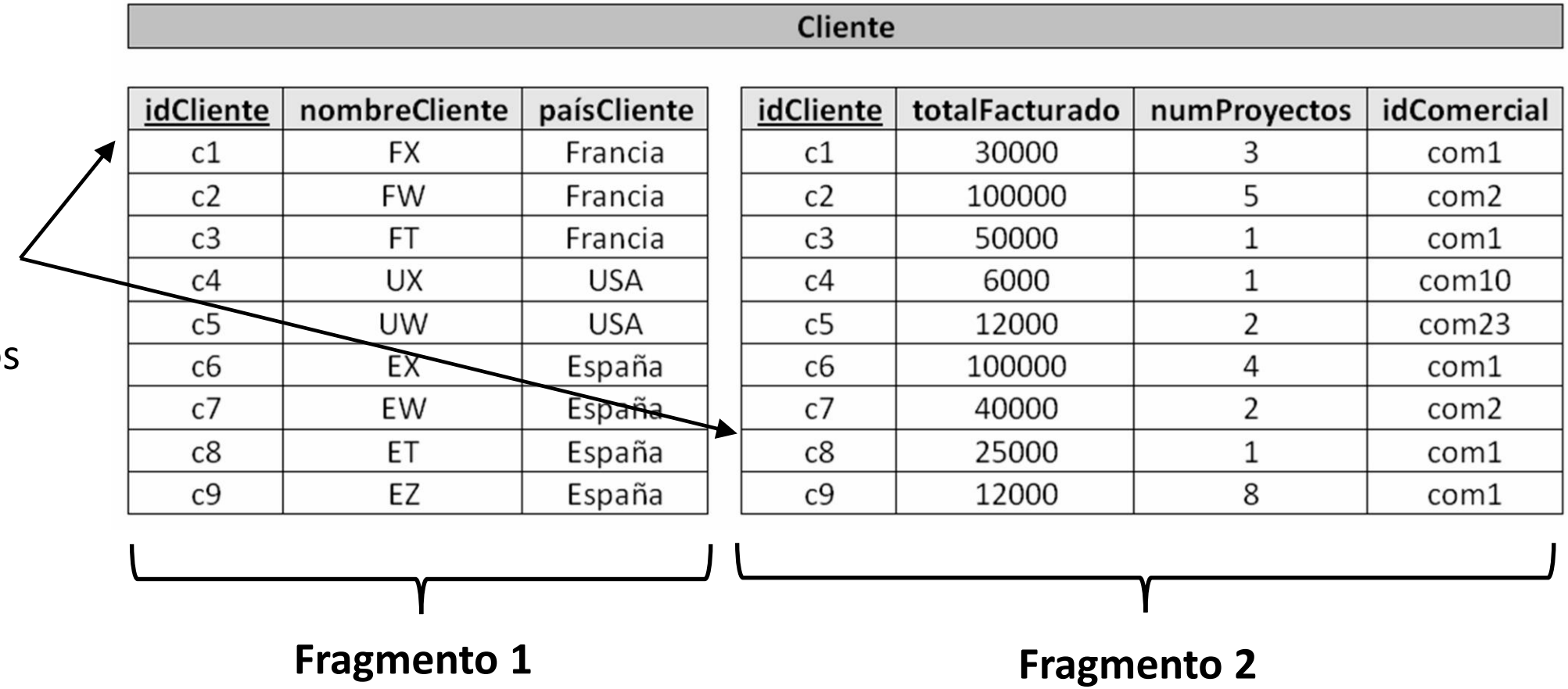
Fragmentación vertical I

- Consiste en usar subconjuntos del esquema, normalmente **grupos de atributos de los objetos** (y los datos asociados a ellos), **como unidad de distribución**.
- Es menos usada que la horizontal:
  - Para el **diseño de fragmentos** es necesario analizar la **afinidad entre atributos** (qué atributos se recuperan de forma conjunta en las operaciones de consulta). Esta tarea suele ser complicada.
  - Puede complicar las operaciones de inserción y modificación, y comprometer la consistencia de los datos (atributos de un objeto en diferentes fragmentos almacenados en diferentes nodos).
- Mejorar el índice de datos útiles recuperados en operaciones de consulta, ya que sólo se leen los atributos relevantes.
- Especialmente útil en sistemas decisionales (sistemas expertos).
- Un ejemplo son los almacenes de columnas (*columna stores*).

Fragmentación vertical II

La relación Cliente se divide en dos fragmentos con grupos de atributos diferentes.

Cada fragmento comparte el idCliente.



Fragmentación híbrida

- La fragmentación híbrida combina las estrategias de fragmentación vertical y horizontal.

La relación Cliente se divide en dos fragmentos verticales

Cliente

idCliente	nombreCliente	paísCliente
c1	FX	Francia
c2	FW	Francia
c3	FT	Francia

idCliente	nombreCliente	paísCliente
c4	UX	USA
c5	UW	USA

idCliente	nombreCliente	paísCliente
c6	EX	España
c7	EW	España
c8	ET	España
c9	EZ	España

idCliente	totalFacturado	numProyectos	idComercial
c1	30000	3	com1
c2	100000	5	com2
c3	50000	1	com1
c4	6000	1	com10
c5	12000	2	com23
c6	100000	4	com1
c7	40000	2	com2
c8	25000	1	com1
c9	12000	8	com1

En este caso, tras haber fragmentado verticalmente en dos fragmentos la relación de clientes, se aplicado fragmentación horizontal (en función del país de procedencia del cliente) sobre el primero de dichos fragmentos.