Base de Datos II

Unidad 4 – Bases de Datos Distribuidas

Objetivos

- Conceptos Básicos.
- Conocer diferentes implementaciones de BDD.
- Fragmentación y Replicación en BDD.
- Transparencia.
- Procesamiento de consultas.
- Distribución del procesamiento.
- Compromiso de dos y tres fases.
- Sistemas Federados

BD Distribuidas

- Un sistema de BDD consiste en una colección de sitios, conectados por medio de algún tipo de red de comunicación, en el cual
  - Cada sitio es un sistema de BD completo por derecho propio
  - Los sitios han acordado trabajar juntos, a fin que un usuario de cualquier sitio pueda acceder a los datos desde cualquier lugar de la red, exactamente como si los datos estuvieran guardados en el propio sitio del usuario.

BD Distribuidas - Ventajas

¿Por qué? Reflexionamos.

- Las empresas ya están distribuidas al menos físicamente.
- Eficiencia de procesamiento.
- Mayor accesibilidad.
- ¿Y desventajas??
  - Complejidad

BDD – Principio fundamental

Ante el usuario, un Sistema de BD Distribuido debe lucir exactamente igual que un sistema que no es distribuido.

BDD – 12 Reglas u objetivos

1. **Autonomía local:** Los sitios deben ser autónomos. Todas las operaciones de ese sitio deben ser administradas por ese sitio. La seguridad, integridad y representación del almacenamiento de los datos locales permanecen bajo el control y jurisdicción del sitio local.

BDD – 12 Reglas u objetivos

2. **No dependencia de un sitio central:** Todos los sitios deben ser tratados como iguales. Esta relacionado con el anterior, pero la no dependencia de un sitio central es necesaria por si misma, aunque no se logra la autonomía local completa.

BDD – 12 Reglas u objetivos

3. **Operación continua:** Deben ofrecer mayor confiabilidad y mayor disponibilidad.
   a) La confiabilidad mejora ya que los sistemas distribuidos pueden continuar operando cuando hay alguna falla en algún sitio individual.
   b) La disponibilidad mejora, en parte por la razón anterior y además por la posibilidad de replicación de datos.

BDD – 12 Reglas u objetivos

4. **Independencia de ubicación:** Los usuarios no tienen que saber donde están almacenados físicamente los datos. Permite que los datos emigren de un sitio a otro sin invalidar programas o actividades.

BDD – 12 Reglas u objetivos

5. **Independencia de fragmentación:** La fragmentación es necesaria por razones de rendimiento: los datos pueden estar almacenados en la ubicación donde son usados más frecuentemente para que la mayoría de las operaciones sean locales y se reduzca el tráfico en la red.

BDD – 12 Reglas u objetivos

6. **Independencia de replicación:** Las réplicas son necesarias por dos razones como mínimo: pueden significar un mejor rendimiento, y pueden significar mejor disponibilidad. La principal desventaja es la actualización de datos replicados, ya que es necesario actualizar todas las copias: propagación de la actualización. Los usuarios deben ser capaces de comportarse como si los datos no estuvieran replicados.

BDD – 12 Reglas u objetivos

7. **Procesamiento de consultas distribuidas:** La optimización en un sistema distribuido es aún mas importante que en uno centralizado. En una consulta habrá muchas formas distintas de satisfacer esa solicitud. Por esto, es crucial la optimización en el procesamiento de BDD.

BDD – 12 Reglas u objetivos

8. **Administración de transacciones distribuidas:** dos aspectos fundamentales son: el control de la recuperación y el control de concurrencia. En los sistemas distribuidos una sola transacción puede involucrar actualizaciones en mas de un sitio. Cada sitio tiene un "agente" que es el proceso realizado en una transacción. El sistema necesita saber cuando dos agentes son parte de la misma transacción, ya que, por ejemplo, no debería permitir que dos agentes que forman parte de la misma transacción caigan en un bloqueo mortal entre ellos.

BDD – 12 Reglas u objetivos

9. **Independencia de hardware:** Existe una necesidad real de poder integrar los datos en maquinas distintas y presentar al usuario una imagen de sistema único. Es necesario tener la posibilidad de ejecutar el mismo DBMS en diferentes plataformas de hardware, y hacer que esas máquinas diferentes participen como socios igualitarios en un sistema distribuido.

BDD – 12 Reglas u objetivos

10. **Independencia de sistema operativo:** Es necesario ejecutar el mismo DBMS en diferentes plataformas de sistema operativo.

BDD – 12 Reglas u objetivos

11. **Independencia de red:** Si el sistema tiene que soportar sitios distintos, con hardware y software distinto, es obviamente necesario tener la posibilidad de soportar también una variedad de redes de comunicación distintas.

BDD – 12 Reglas u objetivos

12. **Independencia del DBMS:** El soporte para la heterogeneidad es definitivamente necesario. El sistema distribuido ideal debe proporcionar independencia de DBMSs.

BDD Homogéneas y Heterogéneas

En las BDD homogéneas todos los sitios tienen idéntico software DBMS, conocen los demás sitios o localizaciones y acuerdan cooperar en el procesamiento de las solicitudes de los usuarios.

En las BDD heterogéneas sitios diferentes pueden utilizar esquemas diferentes y DBMS diferentes. Además pueden no conocer la existencia de los demás. Esto trae aparejado importantes dificultades para el procesamiento de las consultas y también para el procesamiento de transacciones.

Almacenamiento distribuido de datos

Si una relación *r* debe almacenarse en una BD, hay dos enfoques del almacenamiento de esta relación en la BDD:

- Réplica: el sistema conserva réplicas (copias) idénticas de la relación y guarda cada réplica en una localización diferente.
- Fragmentación: el sistema divide la relación en varios fragmentos y guarda cada fragmento en un sitio diferente.

Réplica de datos

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

- Disponibilidad: si alguno de los sitios que contiene la relación r falla, la relación puede hallarse en otro sitio distinto, por lo que, el sistema puede seguir procesando las consultas que impliquen a *r*, a pesar del fallo del sitio.
- Paralelismo incrementado: en caso de que la mayoría de los accesos a r solo resulten en la lectura de la relación, varios sitios pueden procesar en paralelo las lecturas que impliquen a *r*. La réplica de los datos minimiza el movimiento de los datos entre los sitios.
- Sobrecarga incrementada durante la actualización: el sistema debe asegurarse que todas las réplicas de *r*, sean consistentes, en caso contrario pueden producirse cálculos incorrectos. Siempre que se actualiza *r*, hay que propagar la actualización a todos los sitios que contienen réplicas. El resultado es una sobrecarga incremental.

En general la réplica mejora el rendimiento de las operaciones leer y aumenta la disponibilidad de los datos para las transacciones de lectura. La transacciones de actualización suponen una mayor sobrecarga.

Se puede simplificar la gestión de las réplicas de la relación *r*, escogiendo una localización como copia principal de *r*.

Fragmentación de los datos

Si la relación r se fragmenta, se divide en varios fragmentos r_1, r_2, r_3 … r_n. Estos fragmentos contienen suficiente información para permitir la reconstrucción de *r*.

Existen dos esquemas diferentes de fragmentación: *fragmentación horizontal y fragmentación vertical.*

- Fragmentación Horizontal: la relación *r* se divide en varios subconjuntos r_1, r_2, r_3 … r_n . Cada tupla de la relación *r* debe pertenecer como mínimo a uno de los fragmentos. Esta fragmentación suele utilizarse para conservar las tuplas en los sitios que mas se utilizan, minimizando la transferencia de datos.
- Fragmentación Vertical: implica la definición de varios subconjuntos de atributos R_1, R_2, … del esquema *R*. Una forma de asegurar que la relación *r* pueda reconstruirse es incluir los atributos de la clave principal de *R* en cada fragmento R_i.

BDD - Almacenamiento

Se pueden aplicar los dos tipos de fragmentación a un solo esquema, y los fragmentos también pueden replicarse.

En general, podemos decir, que los fragmentos pueden replicarse, las réplicas de los fragmentos pueden fragmentarse, etc.

Transparencia

Los usuarios de BDD no deben conocer la ubicación física de los datos, ni el modo en que se puede tener acceso a ellos en un sitio concreto.

- Transparencia de la fragmentación: no se exige que conozca la forma en que se fragmento la relación.
- Transparencia de la réplica: los usuarios ven cada objeto de datos como único.
- Transparencia de la ubicación: El sistema de BDD debe poder hallar los datos siempre que la transacción del usuario facilite el identificador de los datos.

Transparencia

- Los elementos de datos (relaciones, fragmentos, réplicas) deben tener nombres únicos. En las BDD, hay que tener cuidado para asegurarse de que dos sitios no utilicen el mismo nombre para elementos diferentes.
- Una solución es tener un servidor de nombres central. Los inconvenientes son que el servidor de nombres se transforme el cuello de botella, y si queda fuera de servicio, puede que no siga funcionando ningún otro sitio del SBDD.
- Otra solución es que cada sitio anteponga su propio identificador de sitio a cualquier nombre que genere. Pero no logra la transparencia de la ubicación.
- Para superar este ultimo problema el SBD puede crear un conjunto de nombres alternativos o alias para elementos de datos

BDD – Asignación de Fragmentos

Seguido a la decisión de la fragmentación sigue la asignación de los fragmentos y la cantidad de réplicas a generar.

Parámetro para la generación de réplicas:

- Cantidad de lecturas
- Cantidad de escrituras

Comparación de las alternativas de réplica

	Replica total	Replica parcial	Partición
Procesamiento de consultas	Fácil	Difícil	Difícil
Control de concurrencia	Moderado	Difícil	Fácil
Confiabilidad	Muy alta	Alta	Baja
Realidad	Posible aplicación	Realista	Posible aplicación

Transacciones Distribuidas

- El acceso a los diferentes elementos de datos en los sistemas distribuidos suele realizarse mediante transacciones, que deben preservar las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- Hay dos tipos de transacciones que se deben considerar:
  - Transacciones locales: acceden y actualizan en una BD local.
  - Transacciones globales: acceden y actualizan datos en varias BD locales.
- Asegurar las propiedades ACID en transacciones globales es una tarea complicada, dado que participan en la ejecución varios sitios. El fallo de alguno de los sitios, o el de un enlace de comunicación puede dar lugar a cálculos erróneos.

Estructura del SGBDD

- Cada sitio tiene su propio gestor local de transacciones que asegura las propiedades ACID de las transacciones que se ejecutan en ese sitio.
- Cada gestor de transacciones colabora para ejecutar las transacciones globales.
- Cada gestor contiene: el gestor de transacciones, y el coordinador de transacciones que coordina la ejecución de las diferentes transacciones (tanto locales como globales) iniciadas en ese sitio.

Coordinador de transacciones

- Es responsable de:
  - Inicio de la ejecución de la transacción.
  - División de la transacción en varias subtransacciones y distribución de esas subtransacciones a los sitios correspondientes para su ejecución.
  - Coordinación de la terminación de la transacción, que puede hacer que la transacción se comprometa en todos los sitios o que se aborte en todos los sitios.

Modos de fallos en SGBDD

Los sistemas distribuidos pueden sufrir los mismos tipos de fallos que los sistemas centralizados: errores de software, errores de hardware, fallos de discos), pero hay mas fallos propios de entornos distribuidos:

- Fallo de un sitio.
- Pérdida de mensajes.
- Fallo de un enlace de comunicación.
- División de la red.

Protocolos de compromiso

- Si hay que asegurar la atomicidad, todos los sitios en los que se ejecute una transacción T deben coincidir en el resultado final de la ejecución. T debe comprometerse en todos los sitios o abortarse en todos los sitios.
- Para asegurar esta propiedad el coordinador de transacciones de T debe ejecutar un protocolo de compromiso.

Compromiso de dos fases (C2F)

Considere una transacción *T* iniciada en el sitio S_i, en que el coordinador de transacciones es C_i. Cuando T completa su ejecución (cuando todos los sitios donde se ha ejecutado *T* informan a C_i que *T* se ha completado), C_i inicia el protocolo C2F:

- Fase 1: añade al registro <*T preparar*> al registro histórico y obliga a guardar el registro histórico en un lugar de almacenamiento estable. Entonces envía un mensaje *T preparar* a todos los sitios donde se ha ejecutado T. El gestor de transacciones del sitio determina si desea comprometer su parte de *T*. Si la respuesta es negativa, añade un registro <*no T*> al registro histórico y responde enviando a C_i el mensaje de abortar *T*. Si la respuesta es si, añade un registro <*T preparada*> al registro histórico y obliga a que el registro histórico se guarde en un almacenamiento estable. El gestor de transacciones contesta con el mensaje *T preparada*.

Compromiso de dos fases (C2F)

- Fase 2: Cuando C_i recibe las respuestas al mensaje de preparar *T* de todos los sitios, o cuando ha transcurrido un intervalo de tiempo especificado, C_i puede determinar si la transacción *T* puede comprometerse o abortarse. En función del resultado, se añade al registro histórico un registro <*T comprometida*> o <*T abortada*> y se obliga a que el registro se guarde en almacenamiento estable. A partir de este momento, el coordinador envía un mensaje *comprometer T* o *abortar T* a todos los sitios participantes. Cuando un sitio recibe este mensaje, lo guarda en su registro histórico.

C2F – Trato de Fallos

- Fallo de un sitio participante: Si el coordinador C_i detecta que un sitio ha fallado emprende las acciones siguientes. Si falla antes de responder con el mensaje *T* preparada, el coordinador asume que ha respondido con el mensaje abortar *T*. Si el sitio falla después que el coordinador haya recibido del sitio el mensaje *T* preparada, el coordinador ejecuta el resto del protocolo ignorando el fallo del sitio.
- Cuando un sitio S_k se recupera de un fallo debe examinar su registro histórico para determinar el destino de las transacciones que se hallaban en ejecución cuando se produjo el fallo. Si *T* es una de esas transacciones, se toman en consideración cada uno de los casos posibles. *(rehacer T, deshacer T)*

C2F – Trato de Fallos

- Fallo del coordinador: si el coordinador falla durante la ejecución del protocolo de compromiso para la transacción *T*, los sitios deben decidir el destino de *T*. En ciertos casos los sitios participantes no pueden decidir si comprometer o abortar *T*, y deben esperar a la recuperación del coordinador que ha fallado. Esta situación puede hacer que los elementos de datos no estén disponibles , no solo en el sitio que fallo, sino en sitios activos, por bloqueos, ya que *T* queda bloqueada mientras espera la recuperación del sitio coordinador.

C2F – Trato de Fallos

- División de la red: cuando una red queda dividida hay dos posibilidades:
  - El coordinador y los sitios participantes siguen en una partición, por lo que el fallo no tiene ningún efecto sobre el protocolo de compromiso.
  - El coordinador y los participantes quedan en varias particiones. Esto se maneja como si los sitios que están en una partición distinta al coordinador hubieran fallado.

Compromiso de 3 Fases

Es una extensión del C2F que evita el problema del bloqueo con determinadas suposiciones.

- Se supone que no se producen fragmentaciones de la red y que no fallan mas de *k* sitios, donde *k* es un numero predeterminado.

El protocolo evita el bloqueo introduciendo una tercer fase adicional en que se implican varios sitios en la decisión de comprometer.

Antes de anotar la decisión de comprometer en su almacenamiento persistente, el coordinador se asegura antes que al menos *k* sitios sepan que pretende comprometer la transacción.

Compromiso de 3 fases

Si el coordinador falla, los sitios seleccionan un nuevo coordinador, este nuevo coordinador verifica el estado del protocolo a partir de los demás sitios, si el coordinador había decidido comprometer, al menos uno de los *k* sitios restantes a los que informo estará funcionando y asegurará que se respete la decisión.

El nuevo coordinador vuelve a iniciar la 3 fase del protocolo, ya sea comprometiendo o abortando.

Este protocolo se usa muy poco, debido a la gran sobrecarga. Tiene la propiedad de no bloquearse, a menos que fallen *k* sitios.

Sistemas Federados

Problemática: Una consulta cuya respuesta requiere acceder a diversas BD. Estas BD tienen diferentes arquitecturas, y diferentes DBMS.

Se trata de: poder formular una sola consulta y recibir una única respuesta de modo de que en la preparación de la respuesta intervienen datos de varias BD.

Ejemplos:

- Dos empresas, cada una con sus bases de datos, que se fusionan o pasan a formar parte de un mismo holding.
- Ministerios que quieren compartir sus datos.
- Provincias o territorios autónomos que desean acceder mutuamente a ciertos datos
- Países de un mercado común que necesitan intercambiar datos.

Soluciones

- Integración manual
- Consultar separadamente cada BD, e integrar manualmente las respuestas
- Integración de datos
- Crear una nueva BD que integre todos los datos de las preexistentes
- Integración del acceso
- Construir un Sistema Federado en el que las bases de datos interoperen
- Crear un Data Warehouse

Se debe seleccionar la mejor solución

Consultas separadas e integración manual

Las personas que lo realice deben:

- Saber qué bases de datos están accesibles
- Saber qué datos hay en cada base de datos
- Saber descomponer la consulta en las consultas parciales a cada BD
- Conocer el modelo de cada base de datos
- Conocer el lenguaje de cada base de datos
- Saber integrar los resultados parciales para producir el resultado deseado

Viable sólo muy excepcionalmente

Integración de datos

- Diseñar la nueva base de datos (distribuida?)
- Convertir los programas
- Transferir los datos de las bases de datos preexistentes a la nueva
- Preparar y enseñar los nuevos modos de trabajar de los usuarios

La gestión autónoma de cada base de datos se pierde, en aras de una gestión única

Acceso integrado

- Superponer un sistema nuevo sobre los DBMSs de las bases de datos preexistentes
- El nuevo sistema acepta la consulta y devuelve la respuesta, generando internamente las consultas parciales e integrando sus respuestas
- La existencia de múltiples bases de datos puede ser *transparente al usuario*
- Los programas y usuarios preexistentes no se ven afectados

Se preserva la autonomía de cada base de datos

Autonomía y Flexibilidad

Una de las principales diferencias entre las soluciones radica en el grado en que se mantiene la autonomía de las bases de datos preexistentes:

- Se pierde totalmente en la integración de datos en una BDD
- Se preserva totalmente en la integración manual
- Se obtienen grados intermedios en bases de datos federadas

Se pueden obtener las mismas conclusiones respecto a la flexibilidad de añadir más Bases de Datos

Cualquiera de las soluciones debe superar dificultades técnicas importantes

En muchos casos es preferible el acceso integrado, por razones de:

- viabilidad,
- preservación de autonomía,
- flexibilidad y evolución

No se integran los datos, sino el acceso a los datos

Sistemas de BD Federados

Dos niveles:

- el de las bases de datos preexistentes, que denominaremos bases de datos componentes: NIVEL COMPONENTE
- el del conjunto de bases de datos que interoperan, que llamaremos NIVEL FEDERADO

Los sistemas de bases de datos componentes se FEDERAN para dar lugar a un Sistema de Bases de Datos Federado (FDBS)

Sistemas de BD Federados

FDBS consiste en una colección de Sistemas de Bases de Datos componentes:

- Cooperantes y autónomos
- Interoperando según:
  - las necesidades de los usuarios de la federación
  - el deseo de los administradores de las bases de datos en participar y compartir sus datos
- Manipulados y controlados de manera coordinada por un software llamado Sistema de Gestión de Bases de Datos Federadas (FDBMS)

Características de los SBDF

- Autonomía y heterogeneidad
- El sistema es distribuido
- No hay un esquema conceptual global único, común a toda la federación
- Las federaciones pueden formarse y desaparecer
- Sistemas de bases de datos pueden entrar y salir de una federación
- Un sistema componente puede ser distribuido o a su vez otro FDBS
- Un sistema de base de datos puede ser componente de varias federaciones al mismo tiempo

Características de los SBDF

Autonomía:

- Los SBD estaban normalmente controlados de forma separada e independiente
- ¿Compartir datos? ... únicamente si se retiene el control
- La autonomía de cada componente puede ser de:
  - diseño
  - comunicaciones
  - ejecución
  - asociación

Autonomía

Diseño:

- Principal fuente de heterogeneidad
- Capacidad para elegir su propio diseño en todos sentidos:
  - universo de discurso: qué
  - conceptualización: qué conceptos
  - representación (modelo, lenguaje) y nombres: *cómo*
  - comportamiento (operaciones, restricciones)
  - implementación: hard, soft, SGBD, esquema interno, etc.

Comunicación:

- Capacidad de un SGBD para decidir:
  - si acepta mensajes de otros sistemas componentes
  - cómo y cuándo responder a las solicitudes de otros sistemas

Autonomía

Ejecución:

Capacidad de cada BD componente para:

- decidir el orden en que son ejecutadas las operaciones
- ejecutar operaciones sin interferencia de operaciones externas
- abortar cualquier operación que no cumpla con las restricciones propias
- que las operaciones a nivel componente no se afecten lógicamente con la participación en un FDBS
- no tener que informar a un sistema externo el orden de ejecución

Los SGBD componentes tratan a las operaciones externas de la misma forma que a las operaciones propias: no siempre pueden distinguirlas

Autonomía

Asociación:

- Capacidad de un SGBD para decidir
  - con quién compartir
  - qué compartir
  - cómo compartir
  - si asociarse / desasociarse de una federación

Puede considerarse parte de la autonomía de diseño

Semejanzas y diferencias entre un SBDF y SBDD

Semejanzas:

- hay datos en varias instalaciones
- las instalaciones están interconectadas: *sistema distribuido*
- hay dos niveles
- se formula una pregunta y se obtiene una respuesta
- **Diferencias:**
- de diseño (top-down vs. bottom-up)
- de terminología (global/local – federado/componente)
- de autonomía
- de transparencias
- de arquitectura

En FDBS la "distribución" es una consecuencia de la autonomía

Construcción de un SBDF

El proceso de negociación:

Hay dos casos:

- Para formar una federación nueva
- Para incorporar una BD componente a un FDBS existente

El administrador de la BD componente cede acceso a datos

- a cambio de poder acceder a otros datos
- siguiendo órdenes de rango superior

La BD pierde autonomía: debe obedecer *protocolos*

- respecto a modificar su esquema nativo
- respecto a abandonar el FDBS

Si el FDBS es fuertemente acoplado, la negociación es más compleja

La complejidad de los sistemas de seguridad de las BD plantea dificultades

Construcción de un SBDF

Integración de esquemas:

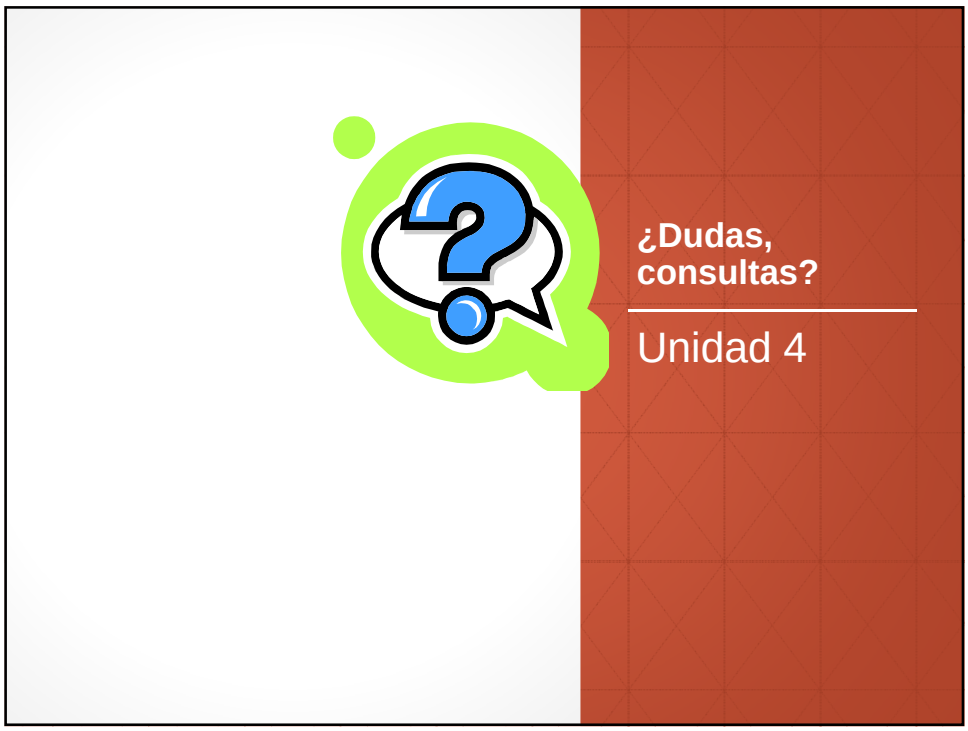
Proceso de integración:

- seleccionar un conjunto de esquemas de exportación a integrar
- integrarlos para producir un esquema federado
- generar las correspondencias entre esquemas (directorio)
- desarrollar el procesador de construcción:
  - operaciones sobre el esq. Federado generan operaciones sobre los esq. Exportación y viceversa.

A partir de aquí se derivan los esquemas externos de acuerdo con las aplicaciones que van a interactuar con los mismos

Objetivos

- ✓Conceptos Básicos.
- ✓Conocer diferentes implementaciones de BDD.
- ✓Fragmentación y Replicación en BDD.
- ✓Transparencia.
- ✓Procesamiento de consultas.
- ✓Distribución del procesamiento.
- ✓Compromiso de dos y tres fases.
- ✓Sistemas Federados


¿Dudas, consultas?
Unidad 4