

Características generales de los SGBD

Conexiones en la red. Paralelismo. Distribución. *Performance*. Escalabilidad. Transparencia. Confiabilidad. Procesamiento de transacciones en línea. Procesamiento para toma de decisión.

La arquitectura de un SGBD está influida en gran medida por el sistema informático subyacente en el que se ejecuta la BD. En la arquitectura se reflejan aspectos que condujeron al desarrollo de diferentes estructuras:

- Conexión en la red
- Arquitectura centralizada
- Paralelismo
- Distribución

No confundir el SGBD con la arquitectura que se elige para implementarlo. Algunos SGBD solo se pueden implantar en una de las arquitecturas y otros en todas ellas.

Conexión en la red

La conexión de varias computadoras en red permite que algunas tareas se ejecuten en un sistema servidor y que otras se ejecuten en sistemas clientes.

Esta división de trabajo condujo al desarrollo de bases de datos cliente-servidor.

El procesamiento en una arquitectura cliente-servidor, dentro de los DBMS, facilita el acceso web a las bases de datos, por lo tanto cualquier LAN o WAN puede considerarse un sistema cliente/servidor desde el momento en que el cliente solicita servicios (datos, impresión, etc.).

Arquitectura centralizada

El SGBD está implementado en una sola plataforma (ordenador) desde donde se gestiona directamente, de modo centralizado, la totalidad de los recursos. Es la arquitectura más clásica de los centros de proceso de datos tradicionales. Se basa en tecnologías sencillas, muy experimentadas y de gran robustez (*hardware*).

Los DBMS corporativos hacen uso de servidores *mainframe* que usan este tipo de arquitectura.

Paralelismo

Un DBMS paralelo usa una colección de recursos (procesadores, discos y memoria) para realizar trabajo en paralelo, junto con los servicios de una red de alta rapidez, sistema operativo y sistema de almacenamiento para coordinar la división del trabajo entre recursos. El grado de recursos compartidos determina las arquitecturas para procesamiento de bases de datos paralelas

El estándar de clasificación de arquitecturas tiene diferentes enfoques:

- SE: todo compartido, memoria y discos se comparten entre una colección de procesadores, usualmente está relacionado como una sola computadora de multiprocesamiento y no como una arquitectura de base de datos paralela. No es necesario el particionamiento porque los procesadores acceden a todos los datos.
- SD: discos compartidos, cada procesador tiene su memoria privada, pero los discos se comparten entre todos los procesadores. No es necesario el particionamiento porque los procesadores acceden a todos los datos.
- SN: nada compartido, los datos se deben particionar entre los procesadores.

Para flexibilidad adicional, las arquitecturas básicas se extienden mediante *clustering*. Un clúster es un acoplamiento firme de dos o más computadoras, de modo que se comparten como una sola computadora.

La necesidad de procesamiento de consultas en paralelo condujo al desarrollo de los sistemas de bases de datos paralelos. En todas las arquitecturas de bases de datos paralelas, el hecho de compartir recursos es transparente a las aplicaciones.

El código de aplicación (SQL y enunciados en lenguaje de programación) no necesita ser modificado para sacar ventaja del procesamiento de bases de datos paralelas.

Los conflictos de diseño que influyen en el rendimiento de arquitecturas de bases de datos paralelas son: equilibrio de carga, coherencia de caché y comunicación de interprocesadores.

El procesamiento en paralelo permite dentro una computadora acelerar las actividades del sistema de BD. Proporciona a las transacciones unas respuestas más rápidas. Da la capacidad de ejecutar más transacciones por segundo.

Las consultas pueden procesarse aprovechando el paralelismo brindado por el sistema informático subyacente.

Distribución

Los DBMS distribuidos soportan solicitudes globales que usan datos almacenados en más de un sitio autónomo. Un sitio es cualquier computadora controlada localmente con una dirección de red única y distribuida geográficamente.

La distribución de datos en distintos sitios de una organización permite que los datos residan en donde se han generado, o en donde sean más necesarios, y continuar siendo accedidos desde otros lugares.

Las solicitudes globales son consultas que combinan datos desde más de un sitio y transacciones que actualizan datos en más de un sitio. Pueden implicar una colección de enunciados que acceden a datos locales en algunos enunciados y a datos remotos en otros.

Los datos locales están controlados por el sitio en el que un usuario normalmente se conecta; los datos remotos implican un sitio diferente en el que un usuario puede incluso no tener cuenta de acceso. Si todas las solicitudes requieren datos de un solo sitio, no se requiere capacidades de procesamiento de bases de datos distribuidas.

Las bases de datos distribuidas son potencialmente útiles para organizaciones que operan en múltiples ubicaciones con control local de recursos computacionales.

Guardar copias de la BD en diferentes sitios permite que las operaciones sobre la BD continúen procesándose aun si algún sitio deja de estar disponible por cualquier motivo (inundación, incendio, terremoto, etc.).

La necesidad de alta disponibilidad de la BD condujo al desarrollo de los sistemas de base de datos distribuidos para poder manejar datos distribuidos geográfica o administrativamente a lo largo de múltiples sistemas de bases de datos.

Para acomodar la distribución de datos son necesarias capas adicionales de descripción de datos.

La arquitectura de esquema para un DBMS distribuido firmemente integrado contiene capas adicionales para fragmentación y asignación.

El esquema de fragmentación contiene la definición de cada fragmento, mientras que el esquema de asignación contiene la ubicación de cada fragmento.

Un fragmento puede definirse como un subconjunto vertical (operación de proyecto), un subconjunto horizontal (operación de restricción) o un fragmento mezclado (combinación de operaciones de proyecto y de restricción).

Un fragmento se asigna a un sitio, aunque en ocasiones a múltiples sitios.

Si el DBMS distribuido soporta copiado, puede asignarse un fragmento a múltiples sitios. En algunos DBMS distribuidos que soportan copiado, una copia de un fragmento se considera la copia primaria y las otras copias son secundarias. Solo se garantiza que la copia primaria sea la actual.

Sistema Gestor de Bases de Datos (SGBD)

Independientemente de la estructura, ¿qué características deben tener las bases de datos?

- *Performance*
- Transparencia
- Escalabilidad
- Disponibilidad / integridad (confiabilidad)

Performance

- Capacidad para ejecutar operaciones en paralelo
- *Multi-threading*
- Tener paralelismo de *input/output*
- Capacidad para ejecutar operaciones utilitarias en paralelo (*backups/restores*) todo en línea
- Paralelismo para el soporte de aplicaciones:
 - Paralelismo de I-O de disco
 - Paralelismo de utilitarios
 - Paralelismo en consultas y procesamiento
- El particionamiento de los datos debe ser independiente de la estructura de *hardware*.
- Paralelismo de I-O de disco:
 - Posibilidad trabajar con tablas particionadas
 - El motor de BD debe permitir la ejecución concurrente de I/O
- Paralelismo de utilitarios:
 - Índices
 - *Backup*
 - *Restore*

- Paralelismo en consultas y procesamiento:
 - Es el más complejo de implementar debido a que una consulta o una modificación de los datos involucra varias operaciones atómicas y resulta complicado hacerlas en paralelo y coordinarlas.
 - El SGBD debe seleccionar operaciones atómicas las cuales pueden ser replicadas y luego ser procesadas en forma concurrente. Luego el resultado se combina como una única operación.
 - Paralelismo horizontal: es la forma de crear múltiples copias de una operación que corre en paralelo.
 - Otra forma de paralelismo: mientras una operación clasifica registros según el valor de un campo clave, en forma concurrente existe una operación que accede a los datos y no espera que estos sean procesados, dejándolos listos para su procesamiento.

Transparencia

Para los DBMS, transparencia significa que los detalles internos de los servicios de las transacciones son invisibles, mejorando así la productividad de los programadores y analistas de bases de datos.

Los DBMS ofrecen dos servicios para garantizar que las transacciones tienen las propiedades ACID:

- Transparencia de recuperación: comprende acciones para manejar las fallas, como errores en la comunicación y colapsos del *software*. El DBMS restaura automáticamente una base de datos a un estado consistente después de una falla.
- Transparencia de concurrencia: comprende acciones para controlar la interferencia entre varios usuarios simultáneos de la base de datos. Los usuarios perciben la base de datos como un sistema para un solo usuario, aunque haya varios usuarios simultáneos.

En el procesamiento de bases de datos distribuidas, la transparencia se relaciona con la independencia de datos. Si la distribución de la base de datos es transparente, los usuarios pueden escribir consultas sin conocimiento de su distribución, esto se conoce como transparencia de fragmentación.

El procesamiento de bases de datos paralelas en las arquitecturas sin nada compartido (SN) involucra transparencia de fragmentación.

Escalabilidad

Propiedad que tiene cualquier sistema para crecer dinámicamente en base a sus necesidades, utilizando los recursos disponibles (*hardware*), sin operaciones complementarias, al incrementar la cantidad de usuarios, procesos, etc.

La BD debe adaptarse sin inconvenientes a *mayor storage*.

Si se agrega más *hardware* a los servidores que soportan la BD, éste debe poder utilizarlo automáticamente y solapar procesos o utilizar más memoria.

La escalabilidad se analiza desde transacciones de lectura.

Dos tipos de escalabilidad:

- Escalabilidad vertical: Se escala sobre el mismo server, con una CPU más rápida o agregando CPU's, agregando una controladora más rápida, agregando más memoria.
- Escalabilidad horizontal: Se utilizan múltiples servers que trabajan en forma transparente distribuyendo la carga de trabajo.

La escalabilidad permite adaptarse rápidamente a los cambios en las organizaciones, producto de reducir a un mínimo los tiempos de planificación y adaptarse a los cambios tecnológicos.

Integridad / disponibilidad

Las BD deben dar confiabilidad/robustez/integridad, por intermedio del *software* y del *hardware*. Dos factores a analizar:

- Administración *on-line*: Utilitarios de operación continua que permitan realizar tareas *on-line*.
- Robustez:
 - Reducir la posibilidad de fallas y realizar recuperos en forma transparente.
 - *Fault tolerant*: se llaman así a los sistemas con alta disponibilidad, implementando redundancia de *hardware* (*mirroring* de disco) y redundancia de datos (*soft mirroring* / replicación).

- Redundancia datos
 - *Soft mirroring (clusters):*
 - Duplicar unidades de datos en discos distintos.
 - Las lecturas se pueden distribuir
 - Las escrituras son transparentes
 - Ante una falla las operaciones se realizan sobre el medio no dañado
 - Para los usuarios es totalmente transparente
 - En el medio dañado una vez reparado se deben sincronizar las copias de los dispositivos
 - Replicación:
 - Similar al *mirroring*, pero se puede localizar en forma remota.
 - Se trata de copiar los datos en diferentes lugares, puede ser en un solo equipo o en varios discos.
 - Esta forma de resguardo no asegura alta disponibilidad, intenta ganar en *performance*.

Confiabilidad y disponibilidad

- Confiabilidad: Un SGBD es confiable si bajo las condiciones de uso determinadas por sus especificaciones cumple con su misión específica en un período determinado. Se utiliza sobre sistemas críticos. Se utiliza un parámetro crítico para evaluar una BD sobre la base de las probabilidades de ocurrencias de fallas. (MTBF *Mean-Time-Between-Failures*).
- Disponibilidad: Porción de tiempo en que la BD se encuentra apta para cumplir su misión, respecto al tiempo que debió haber cumplido su misión y no lo hizo. Para medir la disponibilidad se utiliza el parámetro MTTR (Tiempo Medio de Reparaciones) y el MTBF de la siguiente forma:

$$\text{DISPONIBILIDAD} = \text{MTBF} / (\text{MTBF} + \text{MTTR})$$

Para diseñar un SGBD distribuido confiable es necesario definir los tipos de falla que pueden ocurrir:

- Falla de transacciones: No se cumple con el modelo ACID (atomicidad – concurrencia – aislamiento - durabilidad).
- Fallas de sistema: Son las fallas propias del *hardware* (procesador, memoria, etc.) donde opera el SGBD.

- Fallas del medio de almacenamiento: Es el lugar físico donde se almacenan los datos.
- Fallas de comunicación: Son fallas en el medio de comunicación propiamente dicho que producen perdida de mensajes.

Modos en que trabaja una base de datos

- OLTP: Procesamiento de transacciones en línea
- DSS: Sistemas de soporte de decisión (o sistemas de asistencia de decisiones), también llamados OLAP
- OLCP: Procesamiento complejo en línea

OLTP: Procesamiento de transacciones en línea

El procesamiento de transacciones comprende el lado operativo de las bases de datos. El motor de la base de datos está optimizado para trabajar con un conjunto de transacciones en paralelo y cortas, en otras palabras, se envían al servidor transacciones pequeñas de corta duración. Normalmente realizan la lectura de porciones de información pequeñas.

En este modo de trabajo se generan muchas operaciones de inserción y/o modificación. Los tiempos de ejecución de las transacciones son cortos.

La ejecución de reportes y toma estadísticas degradan la performance de la BD, volviendo lentas todas las operaciones. La ejecución de grandes reportes y copias de respaldo suelen realizarse en momentos de poco uso de la BD.

DSS: Sistemas de soporte de decisión (o sistemas de asistencia de decisiones), también llamados OLAP

El motor de la base de datos está optimizado para realizar lecturas de una gran cantidad de datos, sumarización de datos, es decir soporta procesos largos que van a necesitar leer muchos datos para la generación de reportes, análisis estadísticos, etc. Generalmente realizan lectura de porciones de datos muy grandes.

En este modo de trabajo es casi nula la generación de operaciones de inserción y/o modificación.

Los tiempos de ejecución de las transacciones suelen ser importantes.

Suelen utilizarse para la ejecución de reportes, toma estadísticas.

Para este modo de trabajo suelen utilizarse imágenes relativamente estáticas de la base de datos OLTP para asegurar su integridad mientras se ejecutan los procesos largos.

OLCP: Procesamiento complejo en línea

Es un intermedio o un mix entre el modo OLTP y DSS, intenta alcanzar un balance entre los dos modos reflejando la necesidad de obtener:

- Alto rendimiento de la base (muy buena *performance*)
- Completa disponibilidad de los datos
- Capacidad de realizar copias de respaldo en línea (*backup* en caliente)
- Ejecución de grandes consultas e informes mientras los usuarios actualizan la información diaria de la organización
- Información disponible en todo momento, sin límite de acceso a los usuarios OLTP y DSS.
- Destinada a aquellas organizaciones que necesitan obtener informes críticos en tiempo real.

MÓDULO IV

CARACTERÍSTICAS GENERALES DE LOS SGBD:
CONEXIONES EN LA RED
PARALELISMO
DISTRIBUCIÓN
PERFORMANCE
ESCALABILIDAD
TRANSPARENCIA
CONFIABILIDAD
PROCESAMIENTO DE TRANSACCIONES EN LÍNEA
PROCESAMIENTO PARA TOMA DE DECISIÓN.

Marcela Russo
Laboratorio IV

CARACTERISTICAS GENERALES DE LOS SGBD

- La arquitectura de un SGBD está influenciado en gran medida por el sistema informático subyacente en el que se ejecuta la BD.
- En la arquitectura se reflejan aspectos que condujeron al desarrollo de diferentes estructuras:
 - ***CONEXIÓN EN LA RED***
 - ***ARQUITECTURA CENTRALIZADA***
 - ***PARALELISMO***
 - ***DISTRIBUCION***
- No confundir el SGBD con la arquitectura que se elije para implementarlo. Algunos SGBD solo se puede implantar en una de las arquitecturas y otros en todas ellas.

➤ CONEXION EN LA RED

- La conexión de varias computadoras en red permite que algunas tareas se ejecuten en un sistema servidor y que otras se ejecuten en sistemas clientes.
- Esta división de trabajo condujo al desarrollo de bases de datos cliente-servidor
- El procesamiento en una arquitectura *Cliente-servidor*, dentro de los DBMS, facilita el acceso web a las bases de datos, por lo tanto cualquier LAN o WAN puede considerarse un sistema Cliente/Servidor desde el momento en que el cliente solicita servicios (datos, impresión, etc).

➤ ARQUITECTURA CENTRALIZADA

- El SGBD está implementado en una sola plataforma (ordenador) desde donde se gestiona directamente, de modo centralizado, la totalidad de los recursos.
- Es la arquitectura más clásica de los centros de proceso de datos tradicionales.
- Se basa en tecnologías sencillas, muy experimentadas y de gran robustez (hardware).
- Los DBMS corporativos hacen uso de servidores mainframe que usar este tipo de arquitectura

➤ PARALELISMO

- Un DBMS paralelo usa una colección de recursos (procesadores, discos y memoria) para realizar trabajo en paralelo.
- El trabajo se divide entre recursos para lograr niveles deseados de rendimiento (escalamiento y aceleración) y disponibilidad. Usa los servicios de una red de alta rapidez, sistema operativo y sistema de almacenamiento para coordinar la división del trabajo entre recursos.
- Escalamiento
 - ✓ Involucra la cantidad de trabajo que puede lograrse mediante el aumento de la capacidad de cómputo.
 - ✓ Mide el aumento en tamaño de una labor que puede realizarse mientras se mantiene el tiempo constante.
 - ✓ Lo ideal es que sea lineal, en el que el aumento en capacidad computacional en n veces permite la conclusión de n veces la cantidad de trabajo en el mismo tiempo, no siendo posible en la mayoría de las situaciones debido a los gastos de coordinación.
 - ✓ Se mide como la razón de la cantidad de trabajo concluido con la mayor configuración a la cantidad de trabajo concluido con la configuración original.

➤ PARALELISMO

- Aceleración
 - ✓ Implica la disminución en tiempo para completar una tarea en lugar de la cantidad de trabajo realizado.
 - ✓ Con capacidad de computación añadida, mide la reducción en tiempo mientras se mantiene la constante de tareas.
 - ✓ Las organizaciones necesitan determinar la cantidad de capacidad computacional adicional que garantizará la conclusión del trabajo dentro del tiempo permisible.
 - ✓ Se mide mediante la razón del tiempo de conclusión con la configuración original al tiempo de conclusión con la capacidad adicional.
- Disponibilidad
 - ✓ Es la accesibilidad de un sistema, usualmente medida como el tiempo productivo del sistema.
 - ✓ Para computación de alta disponibilidad o resistente al fallo, un sistema experimenta poco tiempo improductivo y se recupera rápidamente de los fallos.
 - ✓ La computación tolerante al fallo lleva la resistencia al límite en tanto que el procesamiento debe continuarse sin interrupción.

➤ PARALELISMO

- Disponibilidad
 - ✓ El costo del tiempo improductivo determina el grado de disponibilidad deseada, puede incluir pérdidas de salarios, mano de obra perdida y equipo inactivo. Para una organización grande, este costo del tiempo improductivo puede ser de cientos a miles de dólares por hora.
 - ✓ El procesamiento de bases de datos paralelas puede aumentar la disponibilidad porque un DBMS puede ajustarse dinámicamente al nivel de recursos disponibles.
 - ✓ Las fallas de una computadora individual no detendrán el procesamiento en otras computadoras disponibles.

➤ PARALELISMO

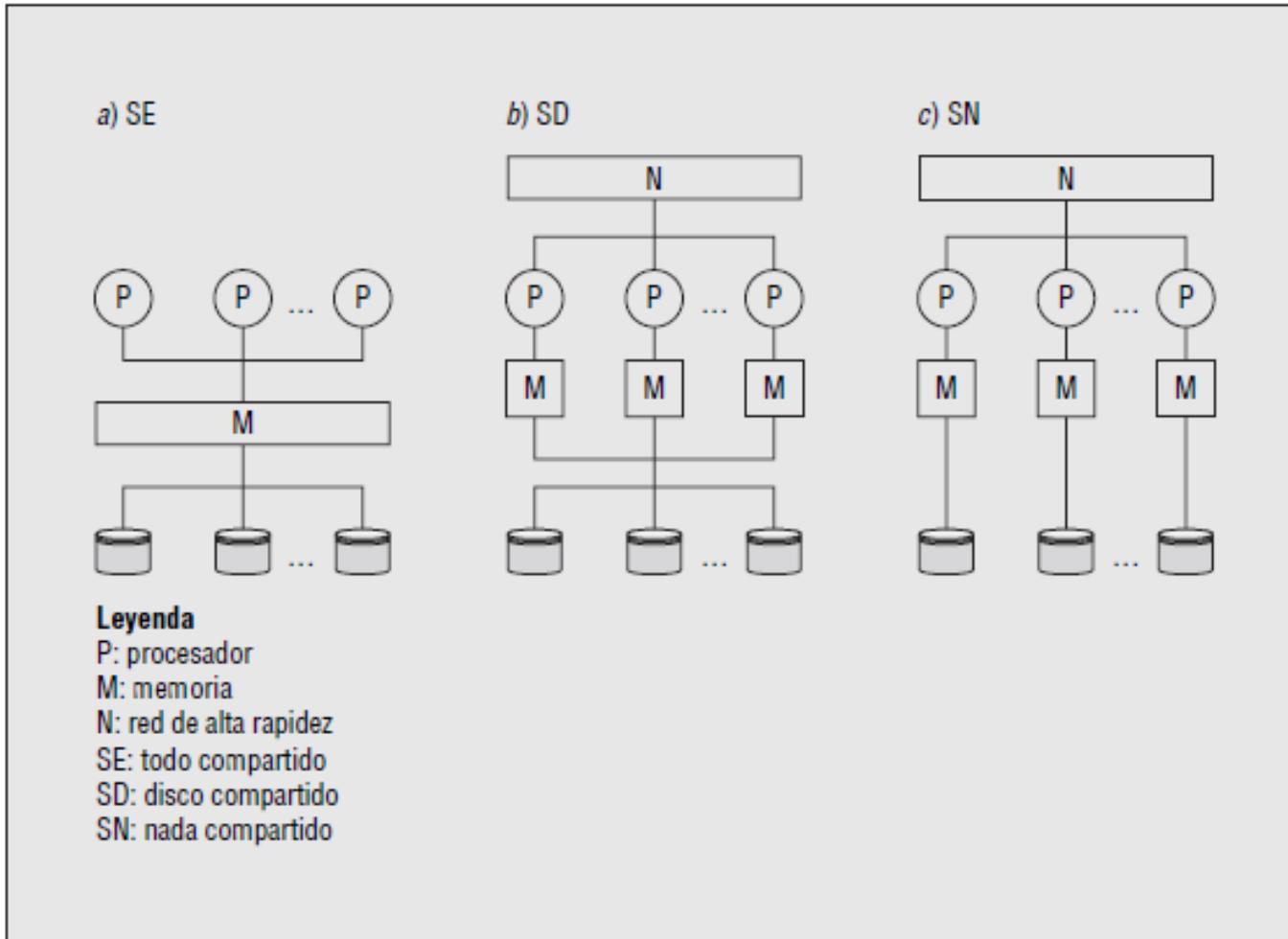
- **Inconveniente del procesamiento de bases de datos paralelas:**
 - ✓ Implica grandes costos en software de DBMS y software de coordinación especializado.
 - ✓ Existen posibles problemas de interoperabilidad, pues se requiere coordinación entre el software DBMS, el sistema operativo y los sistemas de almacenamiento.
 - ✓ Los proveedores de DBMS ofrecen poderosas herramientas para desplegar y administrar la enorme complejidad del procesamiento de bases de datos paralelas.
 - ✓ Si no se predicen mejoras al rendimiento, será un inconveniente significativo. Las mejoras predecibles en rendimiento permiten a las organizaciones planear capacidad adicional y ajustar dinámicamente la capacidad de acuerdo con los volúmenes de procesamiento anticipados y las restricciones en el tiempo de respuesta

➤ PARALELISMO

- El grado de recursos compartidos determina las arquitecturas para procesamiento de bases de datos paralelas
- El estándar de clasificación de arquitecturas tiene diferentes enfoques:
 - ✓ SE: *todo compartido*, memoria y discos se comparten entre una colección de procesadores, usualmente está relacionado como una sola computadora de multiprocesamiento y no como una arquitectura de base de datos paralela. No es necesario el particionamiento, porque los procesadores acceden a todos los datos.
 - ✓ SD: *discos compartidos*, cada procesador tiene su memoria privada, pero los discos se comparten entre todos los procesadores. No es necesario el particionamiento, porque los procesadores acceden a todos los datos.
 - ✓ SN: *nada compartido*, los datos se deben particionar entre los procesadores.

➤ PARALELISMO

- **Grafico de arquitecturas básicas de bases de datos paralelas**



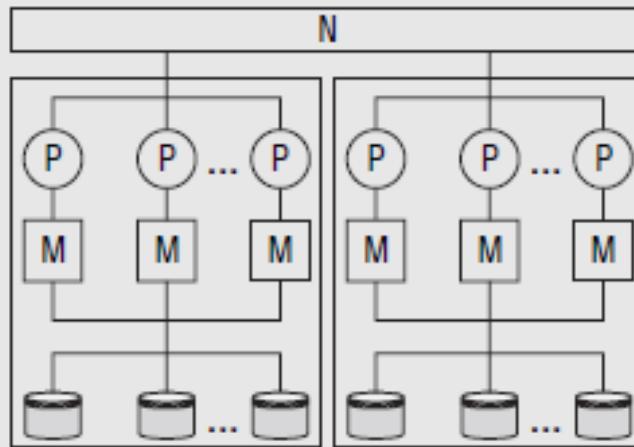
➤ PARALELISMO

- El procesamiento de bases de datos paralelas divide grandes tareas en muchas tareas más pequeñas y las distribuye entre computadoras interconectadas. Por ejemplo, este tipo de procesamiento puede usarse para realizar una operación de unión en grandes tablas
- El uso de las arquitecturas RAID, es una forma simple de procesamiento de bases de datos paralelas.
- Para flexibilidad adicional, las arquitecturas básicas se extienden mediante *clustering*. Un clúster es un acoplamiento firme de dos o más computadoras, de modo que se comparten como una sola computadora
- En las arquitecturas CD (Disco con clúster) los procesadores en cada clúster comparten todos los discos, pero nada se comparte a través de los clústeres.
- En las arquitecturas CN (Nada con clúster) los procesadores en cada clúster no comparten recursos, pero se puede manipular cada clúster para trabajar en paralelo en la realización de una tarea
- Para mayor flexibilidad, el enfoque de clustering, puede configurar dinámicamente el número de clústeres y la membresía a ellos. Además, cada nodo de procesador en un clúster puede ser una computadora de multiprocesamiento o un procesador individual.

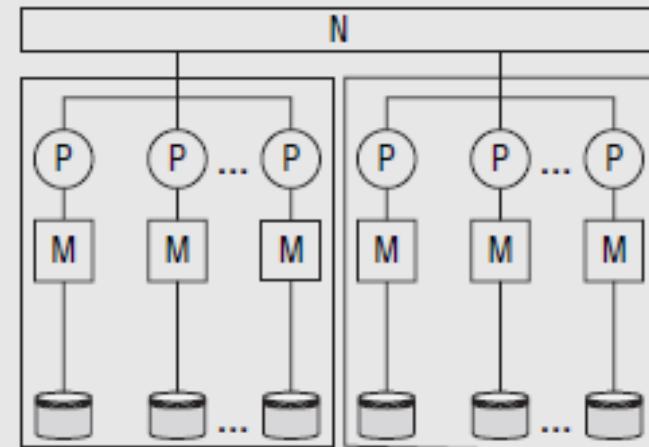
➤ PARALELISMO

- Grafico de arquitecturas con clústering de bases de datos paralelas

a) Disco con clúster (CD)



b) Nada con clúster (CN)



Leyenda

P: procesador

M: memoria

N: red de alta rapidez

➤ PARALELISMO

- La necesidad de procesamiento de consultas en paralelo condujo al desarrollo de los sistemas de bases de datos paralelos.
- En todas las arquitecturas de bases de datos paralelas, el hecho de compartir recursos es transparente a las aplicaciones.
- El código de aplicación (SQL y enunciados en lenguaje de programación) no necesita ser modificado para sacar ventaja del procesamiento de bases de datos paralelas.
- Los conflictos de diseño que influyen en el rendimiento de arquitecturas de bases de datos paralelas son:
 - ✓ **Equilibrio de carga:** involucra la cantidad de trabajo asignado a diferentes procesadores en un clúster. Cada procesador debería tener la misma cantidad de trabajo para utilizar el clúster por completo. Resulta difícil dividir un subconjunto de una base de datos para lograr igual división de trabajo debido a que el sesgo de datos es común entre las columnas de bases de datos
 - ✓ **Coherencia de caché:** implica sincronización entre memorias locales y almacenamiento de disco común. Cuando un procesador direcciona una página de disco, la imagen de ese página permanece en la caché asociada al procesador. Una inconsistencia ocurre si otro procesador cambia la página en su propio búfer. Para evitar inconsistencias, cuando se accede a una página de disco debe hacerse una verificación de otros cachés locales para coordinar los cambios producidos en los cachés de dichos procesadores. Este conflicto se limita a las arquitecturas de disco compartido (SD y CD).

➤ PARALELISMO

- ✓ **Comunicación interprocesadores:** involucra los mensajes generados para sincronizar acciones de procesadores independientes. El paralelismo particionado, como se usa para las arquitecturas nada compartido, puede crear grandes cantidades de comunicación interprocesadores. En particular, las operaciones conjuntas particionadas pueden generar una gran cantidad de comunicación elevada para combinar resultados parciales ejecutados en diferentes procesadores
- **El procesamiento en paralelo:**
 - ✓ Permite dentro una computadora acelerar las actividades del sistema de BD.
 - ✓ Proporciona a las transacciones respuestas más rápidas.
 - ✓ Dá la capacidad de ejecutar más transacciones por segundo.
 - ✓ Las consultas pueden procesarse aprovechando el paralelismo brindado por el sistema informático subyacente.

➤ **DISTRIBUCION**

- **Motivación para datos distribuidos:**

- ✓ Ofrecen algunas ventajas en relación al control de datos, reducción de costos de comunicación y rendimiento mejorado, por lo tanto distribuir una base de datos permite la ubicación de datos de modo que se ajuste a la estructura de una organización
- ✓ Las decisiones acerca de compartir y mantener los datos pueden establecerse localmente para proporcionar un control más cercano al uso de datos.
- ✓ Los datos deben ubicarse de modo que 80 por ciento de las solicitudes sean locales, debido a que incurren en poco o ningún costo de comunicación y retardos, en comparación con las solicitudes remotas.
- ✓ La creciente disponibilidad de datos también puede conducir a rendimiento mejorado, debido a que los datos son más accesibles porque no hay una sola computadora responsable de controlar el acceso y además pueden copiarse de modo que estén disponibles en más de un sitio.

➤ **DISTRIBUCION**

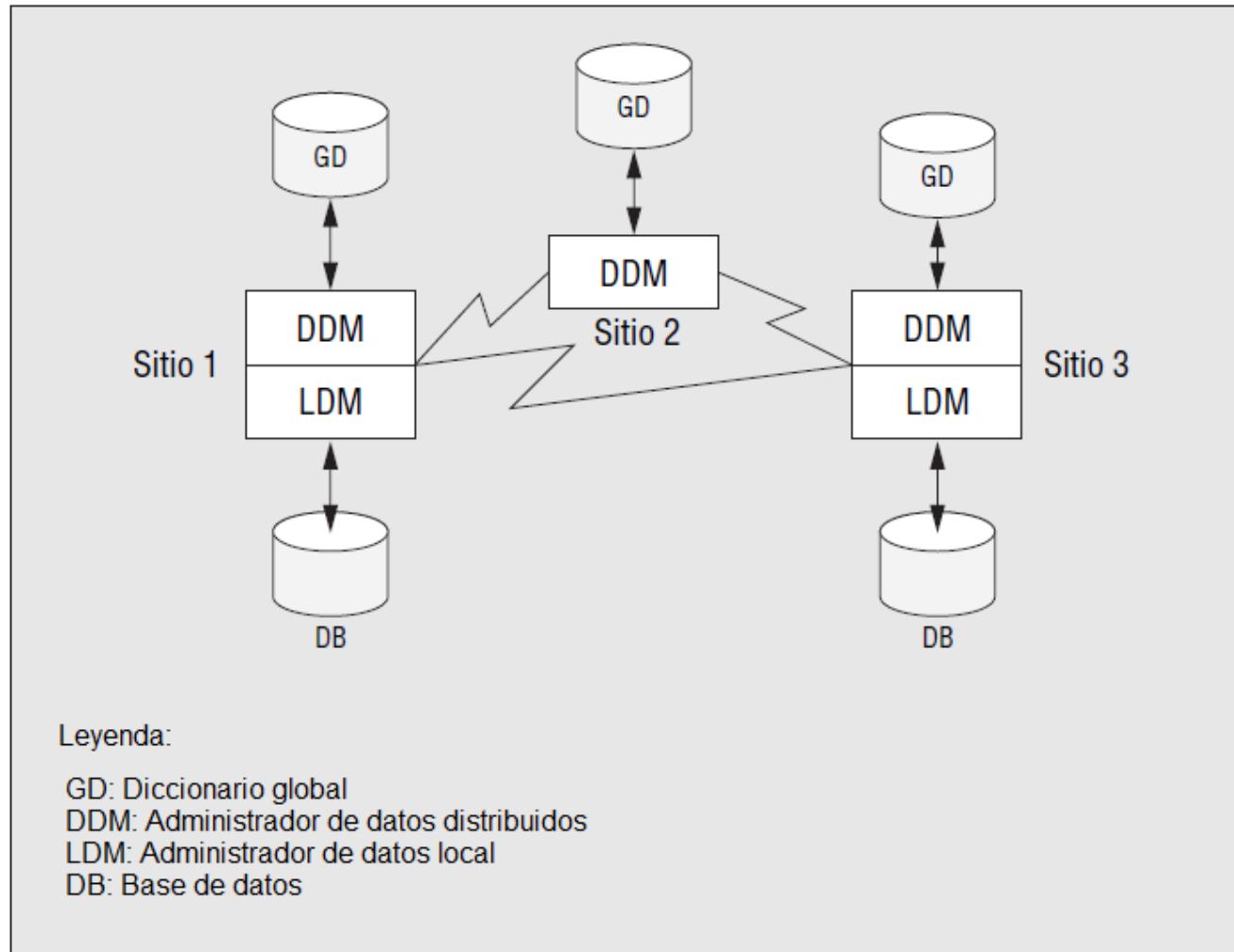
- **Algunos problemas significativos sobre los datos distribuidos están dados por:**
 - ✓ **Un pobre diseño de bases de datos distribuidas puede conducir a enormes costos de comunicación y a pobre rendimiento.**
 - ✓ **El diseño de bases de datos distribuidas es difícil debido a la carencia de herramientas, el número de opciones y las relaciones entre opciones.**
 - ✓ **El procesamiento de transacción distribuida puede agregar considerables gastos, en especial para datos copiados.**
 - ✓ **Los datos distribuidos implican más preocupaciones de seguridad porque son muchos los sitios que pueden manejar datos. Cada sitio debe protegerse adecuadamente de accesos no autorizados.**
- **El procesamiento de bases distribuidas proporcionan autonomía de sitio mientras que las bases de datos paralelas, no. Por lo tanto la autonomía es la diferencia fundamental entre el procesamiento de bases de datos paralelas y distribuidas.**

➤ **DISTRIBUCION**

- Los DBMS distribuidos soportan solicitudes globales que usan datos almacenados en más de un sitio autónomo, un sitio es cualquier computadora controlada localmente con una dirección de red única y con frecuencia se encuentran distribuidos geográficamente, aunque la definición soporta sitios ubicados en proximidad cercana. Por lo tanto, cada servidor con acceso a la base de datos distribuida se conoce como sitio
- Las solicitudes globales son consultas que combinan datos desde más de un sitio y transacciones que actualizan datos en más de un sitio, pueden implicar una colección de enunciados que acceden a datos locales en algunos enunciados y a datos remotos en otros.
- Los datos locales están controlados por el sitio en el que un usuario normalmente se conecta, los datos remotos implican un sitio diferente, en el que un usuario puede acceder sin tener cuenta de acceso. Si todas las solicitudes requieren datos de un solo sitio, no se requiere capacidades de procesamiento de bases de datos distribuidas
- Las bases de datos distribuidas son potencialmente útiles para organizaciones que operan en múltiples ubicaciones con control local de recursos computacionales

➤ DISTRIBUCION

- Posible ordenamiento de los componentes de un DBMS distribuido



➤ **DISTRIBUCION**

- **En el ejemplo:**

- ✓ **Cada servidor con acceso a la base de datos distribuida se conoce como sitio.**
- ✓ **Si un sitio contiene una base de datos, ésta se controla con un administrador de datos local (LDM). Por lo tanto, los administradores de datos locales proporcionan características completas de un DBMS**
- ✓ **El administrador de datos distribuidos (DDM) optimiza la ejecución de consultas a través de sitios, coordina el control de concurrencia y la recuperación a través de sitios, también controla el acceso a datos remotos. Al realizar estas tareas, el DDM utiliza el diccionario global (GD) para localizar partes de la base de datos.**
- ✓ **El GD puede distribuirse en varios sitios de manera similar a como se distribuyen los datos**

➤ DISTRIBUCION

- En la arquitectura de componente, los administradores de base de datos locales (LDM) pueden ser homogéneos o heterogéneos.
 - ✓ **LDM Homogéneos:** Un DBMS distribuido con DBMS local homogéneo está *firmemente integrado*. El administrador de base de datos distribuido puede solicitar componentes internos y acceso al estado interno de los administradores de datos locales. La firme integración permite que el DBMS distribuido soporte eficientemente las consultas y transacciones distribuidas. Sin embargo, el requisito de homogeneidad prohíbe la integración de bases de datos existentes.
 - ✓ **LDM Heterogéneos:** Un DBMS distribuido con administradores de datos locales heterogéneos tiene *integración floja*. El administrador de base de datos distribuido actúa como middleware para coordinar administradores de datos locales. Con frecuencia, SQL proporciona la interfaz entre el administrador de datos distribuido y los administradores de datos locales. La integración floja soporta compartir datos entre sistemas heredados y organizaciones independientes. Sin embargo, el enfoque de integración floja puede no ser capaz de soportar procesamiento de transacciones de forma confiable y eficiente.
- Para acomodar la distribución de datos son necesarias capas adicionales de descripción de datos. Las arquitecturas proporcionan una referencia acerca de los tipos de descripción de datos necesarios y la forma de compartir la descripción de datos.

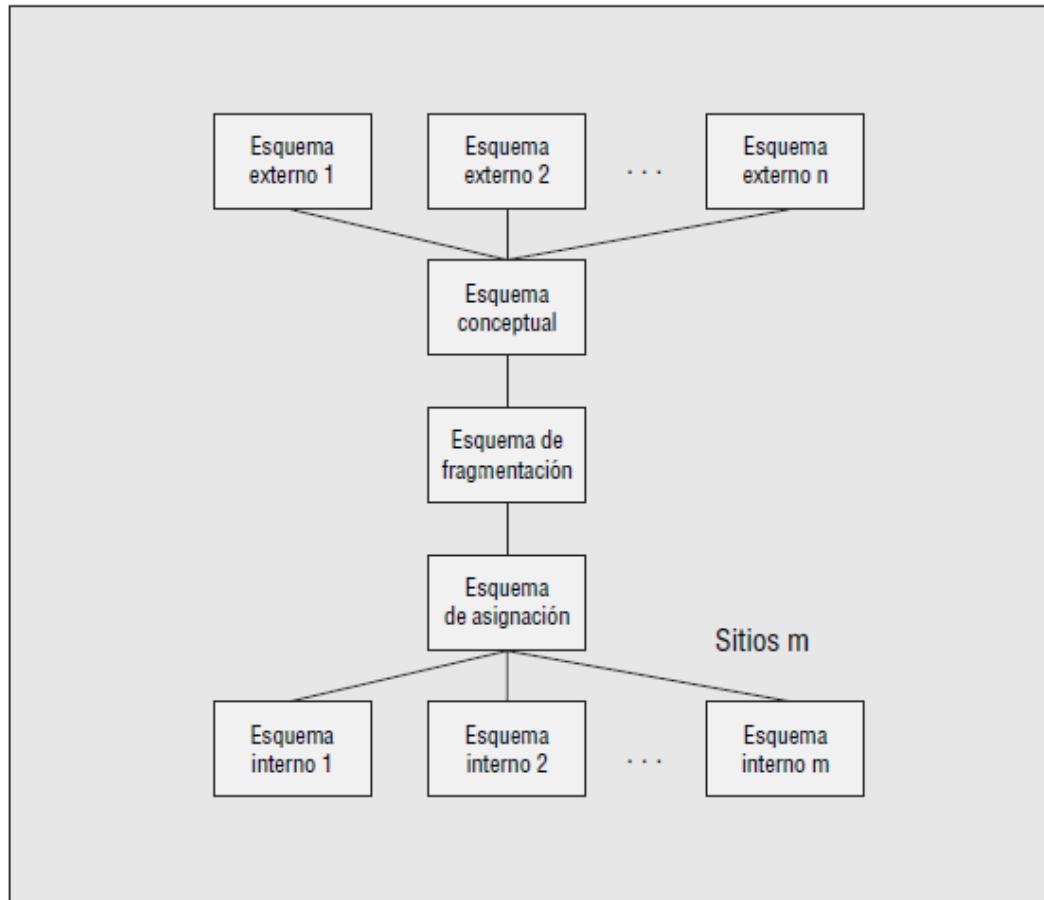
➤ **DISTRIBUCION**

- **DBMS distribuido firmemente integrado:**

- ✓ La arquitectura de esquema contiene capas adicionales para fragmentación y asignación
- ✓ El esquema de fragmentación contiene la definición de cada fragmento, mientras que el esquema de asignación contiene la ubicación de cada fragmento.
- ✓ Un fragmento puede definirse como un subconjunto vertical (operación de proyecto), un subconjunto horizontal (operación de restricción) o un fragmento mezclado (combinación de operaciones de proyecto y de restricción).
- ✓ Un fragmento se asigna a un sitio, aunque en ocasiones a múltiples sitios.
- ✓ Si el DBMS distribuido soporta copiado, puede asignarse un fragmento a múltiples sitios, una copia de un fragmento se considera la copia primaria y las otras copias son secundarias. Sólo se garantiza que la copia primaria sea la actual.

➤ DISTRIBUCION

- Ejemplo gráfico de un DBMS distribuido firmemente integrado



➤ **DISTRIBUCION**

- **DBMS distribuido con integración floja:**

- ✓ La arquitectura de esquema soporta más autonomía de sitios de base de datos locales, además de compartición de datos.
- ✓ Cada sitio contiene los tres niveles de esquema tradicionales.
- ✓ Para soportar la compartición de datos, el DBMS distribuido proporciona un esquema de mapeo local para cada sitio.
- ✓ Los esquemas de mapeo local describen los datos exportables en un sitio y proporcionan reglas de conversión para traducir los datos de un formato local a uno global.
- ✓ El esquema conceptual global muestra todos los tipos de datos y relaciones que se pueden usar en solicitudes globales.
- ✓ Los DBMSs distribuidos que no tienen un esquema conceptual global utilizan esquemas globales externos y proporcionan visiones de datos compartidos en un formato común.
- ✓ Los formatos de datos locales pueden ser muy diferentes, los sitios locales pueden usar diferentes DBMSs, cada uno con un conjunto diferente de tipos de datos.

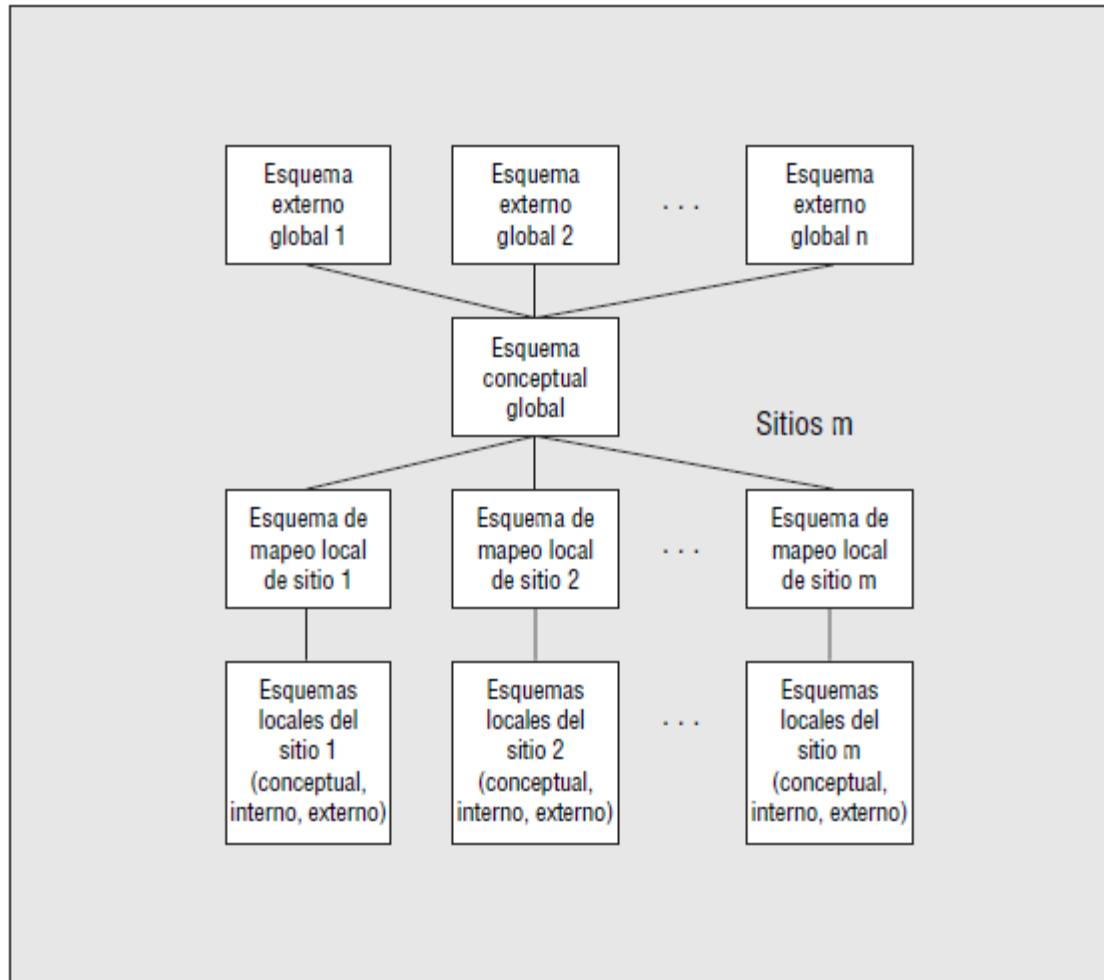
➤ **DISTRIBUCION**

- **DBMS distribuido con integración floja:**

- ✓ Los modelos de datos de los DBMS locales pueden ser diferentes, especialmente si se integran sistemas heredados.
- ✓ Los sistemas heredados pueden usar interfaces de archivo y modelos de datos navegacionales (red y jerarquía) que no soportan SQL, como también si los sitios locales soportan un estándar SQL común, tales como diferentes tipos de datos, escalas, unidades de medida y códigos.
- ✓ Los esquemas de mapeo local resuelven estas diferencias al proporcionar reglas de conversión que transforman los datos de un formato local a un formato global.

➤ DISTRIBUCION

- Ejemplo gráfico de un DBMS distribuido con integración floja



➤ DISTRIBUCION

- Las arquitecturas firmemente integradas y con integración floja representan dos posibilidades extremas, por ello entre estas dos arquitecturas se ha propuesto e implementado muchas variaciones:
 - ✓ Para proporcionar autonomía local adicional con mayor eficiencia para solicitudes globales, un sistema con integración floja puede requerir que todos los sitios locales soporten una interfaz SQL común.
 - ✓ Combinar los enfoques firmemente integrado y con integración floja.
 - ✓ Las redes de bases de datos distribuidas firmemente integradas pueden tener integración floja para compartir datos selectivos en solicitudes globales. Así, el DBMS distribuido con integración floja actúa como puerta de enlace entre las bases de datos distribuidas firmemente integradas

➤ DISTRIBUCION

- La transparencia en el procesamiento de bases de datos distribuidas se relacionan con la independencia de datos, de esta manera los usuarios pueden escribir consultas desconociendo la distribución. Cualquier cambio en la distribución de los datos no impacta en las consultas y tampoco en las transacciones
- La transparencia de fragmentación proporciona el mayor nivel de independencia de datos. Los usuarios formulan consultas y transacciones sin conocimiento de los fragmentos, ubicaciones o formatos locales. Si los fragmentos, ubicaciones o formatos locales cambian, las consultas y transacciones no se afectan
- La transparencia de ubicación proporciona un menor nivel de independencia de datos que la transparencia de fragmentación. Los usuarios necesitan referirse a fragmentos al formular consultas y transacciones. Sin embargo, no es necesario el conocimiento de las ubicaciones y formatos locales.
- La transparencia de mapeo local proporciona un menor nivel de independencia de datos que la transparencia de ubicación. Los usuarios necesitan hacer referencia a fragmentos en sitios al formular consultas y transacciones. Sin embargo, no es necesario el conocimiento de los formatos locales. Si los sitios difieren en formatos, como en las bases de datos distribuidas con integración floja, la transparencia de mapeo local todavía libera al usuario de un trabajo considerable.

➤ DISTRIBUCION

- La transparencia en el procesamiento de bases de datos distribuidas se relacionan con la independencia de datos, de esta manera los usuarios pueden escribir consultas desconociendo la distribución. Cualquier cambio en la distribución de los datos no impacta en las consultas y tampoco en las transacciones.
- Los datos distribuidos pueden agregar complejidad para la formulación de consultas, añaden considerable complejidad al procesamiento de consultas y transacciones.
- El procesamiento de bases de datos distribuidas implica movimiento de datos, procesamiento remoto y coordinación del sitio que están ausentes del procesamiento de bases de datos centralizadas.
- Aunque los detalles del procesamiento de bases de datos distribuidas pueden ocultarse a los programadores y usuarios, a veces no es posible ocultar las implicaciones de rendimiento.

Sistema Gestor de Bases de Datos (SGBD)

INDEPENDIENTEMENTE DE LA ESTRUCTURA

¿QUE CARACTERISTICAS DEBEN TENER LAS BASES DE DATOS?

- PERFORMANCE
- TRANSPARENCIA
- ESCALABILIDAD
- DISPONIBILIDAD / INTEGRIDAD (CONFIABILIDAD)

➤ **PERFORMANCE**

- **Capacidad para ejecutar operaciones en paralelo**
- **Multi-threading**
- **Tener paralelismo de Input/Output**
- **Capacidad para ejecutar operaciones utilitarias en paralelo (Backups/Restores) todo en línea**
- **Paralelismo para el soporte de aplicaciones:**
 - ✓ **Paralelismo de I-O de disco**
 - ✓ **Paralelismo de utilitarios**
 - ✓ **Paralelismo en consultas y procesamiento**
- **El particionamiento de los datos debe ser independiente de la estructura de Hardware.**
- **Paralelismo de I-O de disco:**
 - ✓ **Posibilidad trabajar con tablas particionadas**
 - ✓ **El motor de BD debe permitir la ejecución concurrente de I/O**

➤ **PERFORMANCE**

- **Paralelismo de utilitarios:**

- ✓ Indices
- ✓ Backup
- ✓ Restore

- **Paralelismo en consultas y procesamiento:**

- ✓ Es el más complejo de implementar debido a que una consulta o una modificación de los datos, involucra varias operaciones atómicas y resulta complicado hacerlas en paralelo y coordinarlas.
- ✓ El SGBD debe seleccionar operaciones atómicas las cuales pueden ser replicadas y luego ser procesadas en forma concurrente. Luego el resultado se combina como una única operación.
- ✓ **Paralelismo Horizontal:** es la forma de crear múltiples copias de una operación que corre en paralelo.
- ✓ **Otra forma de paralelismo:** mientras una operación clasifica registros según el valor de un campo clave, en forma concurrente existe una operación que accede a los datos y no espera que estos sean procesados, dejándolos listos para su procesamiento

➤ TRANSPARENCIA:

- Para los DBMS, transparencia significa que los detalles internos de los servicios de las transacciones son invisibles, mejorando así la productividad de los programadores y analistas de bases de datos
- Los DBMS ofrecen dos servicios para garantizar que las transacciones tienen las propiedades ACID:
 - ✓ Transparencia de recuperación: comprende acciones para manejar las fallas, como errores en la comunicación y colapsos del software. El DBMS restaura automáticamente una base de datos a un estado consistente después de una falla
 - ✓ Transparencia de concurrencia: comprende acciones para controlar la interferencia entre varios usuarios simultáneos de la base de datos. Los usuarios perciben la base de datos como un sistema para un solo usuario, aunque haya varios usuarios simultáneos
- En el procesamiento de bases de datos distribuidas, la transparencia se relaciona con la independencia de datos. Si la distribución de la base de datos es transparente, los usuarios pueden escribir consultas sin conocimiento de su distribución, esto se conoce como transparencia de fragmentación.
- El procesamiento de bases de datos paralelas en las arquitecturas nada compartido (SN) involucra transparencia de fragmentación.

➤ **ESCALABILIDAD:**

- Propiedad que tiene cualquier sistema, para crecer dinámicamente, en base a sus necesidades, utilizando los recursos disponibles (Hardware), sin operaciones complementarias, al incrementar la cantidad de usuarios, procesos, etc.
- La BD debe adaptarse sin inconvenientes a mayor storage.
- Si se agrega más hardware a los servidores que soportan la BD éste debe poder utilizarlo automáticamente y solapar procesos o utilizar más memoria.
- La escalabilidad se analiza desde transacciones de lectura
- Dos tipos de escalabilidad:
 - ✓ Escalabilidad Vertical: Se escala sobre el mismo server, con una CPU más rápida o agregando CPU's, agregando una controladora más rápida, agregando más memoria.
 - ✓ Escalabilidad Horizontal: Se utilizan múltiples servers que trabajan en forma transparente distribuyendo la carga de trabajo.

ESCALABILIDAD:

- La escalabilidad permite:
 - ✓ Adaptarse rápidamente a los cambios en las organizaciones, producto de reducir a un mínimo los tiempos de planificación
 - ✓ Adaptarse a los cambios tecnológicos.

INTEGRIDAD / DISPONIBILIDAD

- Las BD deben dar **CONFIABILIDAD/ROBUSTEZ/INTEGRIDAD**, por intermedio del Software y de Hardware
- Dos factores a analizar:
 - ✓ Administración ON-Line: Utilitarios de operación continua que permitan realizar tareas on-line.
 - ✓ Robustez:
 - Reducir la posibilidad de fallas y realizar recuperos en forma transparente.
 - Fault tolerant: se llaman así a los sistemas con alta disponibilidad, implementando redundancia de hardware (mirroring de disco) y redundancia de datos (soft mirroring / replicación).

INTEGRIDAD / DISPONIBILIDAD

- Redundancia datos
 - Soft mirroring (Clusters):
 - ✓ Duplicar unidades de datos en discos distintos.
 - ✓ Las lecturas se pueden distribuir
 - ✓ Las escrituras son transparentes
 - ✓ Ante una falla las operaciones se realizan sobre el medio no dañado.
 - ✓ Para los usuarios es totalmente transparente
 - ✓ En el medio dañado una vez reparado se deben sincronizar las copias de los dispositivos
 - Replicación:
 - ✓ Similar al mirroring, pero se puede localizar en forma remota.
 - ✓ Se trata de copiar los datos en diferentes lugares, puede ser en un solo equipo o en varios discos,
 - ✓ Esta forma de resguardo no asegura alta disponibilidad, intenta ganar en performance

CONFIABILIDAD Y DISPONIBILIDAD

- **CONFIABILIDAD:** Un SGBD es confiable si bajo las condiciones de uso determinadas por sus especificaciones cumple con su misión específica en un período determinado. Se utiliza sobre sistemas críticos. Se utiliza un parámetro crítico para evaluar una BD sobre la base de las probabilidades de ocurrencias de fallas. (**MTBF Mean-Time-Between-Failures**).
- **DISPONIBILIDAD:** Porción de tiempo que la BD se encuentra apta para cumplir su misión, respecto al tiempo que debió haber cumplido su misión y no lo hizo. Para medir la disponibilidad se utiliza el parámetro **MTTR** (Tiempo Medio de Reparaciones) y el **MTBF** de la siguiente forma:

$$\text{DISPONIBILIDAD} = \text{MTBF} / (\text{MTBF} + \text{MTTR})$$

CONFIABILIDAD Y DISPONIBILIDAD

- Para diseñar un SGBD distribuido confiable es necesario definir los tipos de falla que pueden ocurrir:
 - ✓ **FALLA DE TRANSACCIONES:** No se cumple con el modelo **ACID** (**A**tomicidad – **C**oncurrencia – **I**slamiento - **D**urabilidad).
 - ✓ **FALLAS DE SISTEMA:** Son las fallas propias del hardware (procesador, memoria, etc.) donde opera el SGBD.
 - ✓ **FALLAS DEL MEDIO DE ALMACENAMIENTO:** Es el lugar físico donde se almacenan los datos.
 - ✓ **FALLAS DE COMUNICACIÓN:** Son fallas en el medio de comunicación propiamente dicho que producen perdida de mensajes

MODOS EN QUE TRABAJA UNA BASE DE DATOS:

- ***OLTP: Procesamiento de Transacciones en Línea***
- ***DSS: Sistemas de Soporte de Decisión (o Sistemas de Asistencia de Decisiones), también llamados OLAP***
- ***OLCP: Procesamiento Complejo en Línea***

MODOS EN QUE TRABAJA UNA BASE DE DATOS:

➤ ***OLTP: Procesamiento de Transacciones en Línea***

- El procesamiento de transacciones comprende el lado operativo de las bases de datos.
- El motor de la base de datos está optimizado para trabajar con un conjunto de transacciones en paralelo y cortas, en otras palabras se envían al servidor transacciones pequeñas de corta duración.
- Normalmente realizan lectura de porciones de información pequeñas.
- En este modo de trabajo se generan muchas operaciones de inserción y/o modificación
- Los tiempos de ejecución de las transacciones son cortos
- La ejecución de reportes, toma estadísticas degradan la performance de la BD, volviendo lentas todas las operaciones.
- La ejecución de grandes reportes y copias de respaldo suelen realizarse en momentos de poco uso de la BD.

MODOS EN QUE TRABAJA UNA BASE DE DATOS:

➤ **DSS: Sistemas de Soporte de Decisión (o Sistemas de Asistencia de Decisiones), también llamados OLAP**

- El motor de la base de datos está optimizado para realizar lecturas de una gran cantidad de datos, summarización de datos, es decir soporta procesos largos que van a necesitar leer muchos datos para la generación de reportes, análisis estadísticos, etc.
- Generalmente realizan lectura de porciones de datos muy grandes.
- En este modo de trabajo es casi nula la generación de operaciones de inserción y/o modificación.
- Los tiempos de ejecución de las transacciones suelen ser importantes en cuanto al tiempo.
- Suelen utilizarse para la ejecución de reportes, toma estadísticas.
- Para este modo de trabajo suelen utilizarse imágenes relativamente estáticas de la base de datos OLTP para asegurar la integridad de los datos mientras se ejecutan los procesos largos.

MODOS EN QUE TRABAJA UNA BASE DE DATOS:

➤ ***OLCP: Procesamiento Complejo en Línea***

- Es un intermedio “o un mix” entre el modo OLTP y DSS, intentando alcanzar un balance entre los dos modos, reflejando la necesidad de obtener:
 - ✓ Alto rendimiento de la base (muy buena performance)
 - ✓ Completa disponibilidad de los datos
 - ✓ Capacidad de realizar copias de respaldo en línea (backup en caliente)
 - ✓ Ejecución de grandes consultas e informes mientras los usuarios actualizan la información diaria de la organización
 - ✓ Información disponible en todo momento, sin límite de acceso a los usuarios OLTP y DSS.
 - ✓ Destinada a aquellas organizaciones que necesitan obtener informes críticos en tiempo real

MÓDULO V

ARQUITECTURA CENTRALIZADA

ARQUITECTURA CLIENTE SERVIDOR

ARQUITECTURA DE 2 Y 3 CAPAS

TIPOS DE SERVIDORES:

ITERATIVOS

CONCURRENTES

DE TRANSACCIÓN

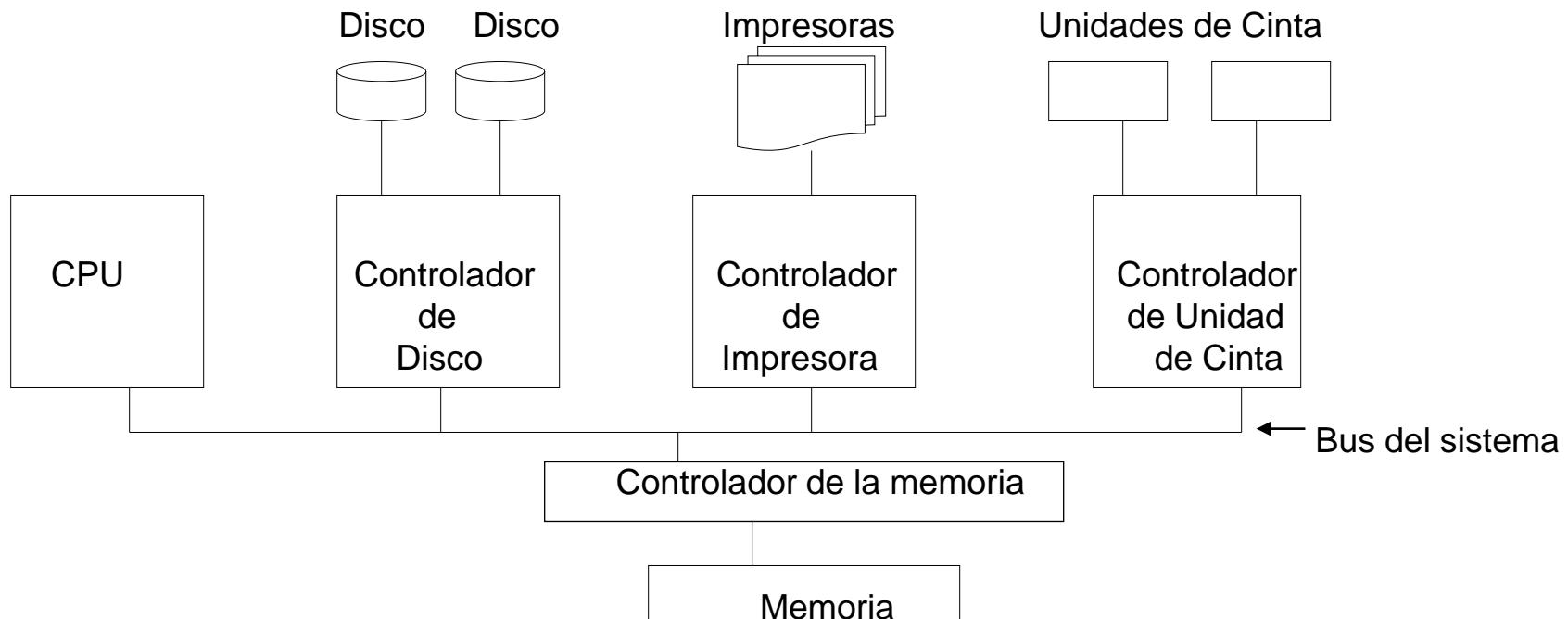
DE DATOS

Marcela Russo
Laboratorio IV

CARACTERISTICAS GENERALES DE LOS SGBD

➤ ARQUITECTURA CENTRALIZADA

- En la ARQUITECTURA CENTRALIZADA el SGBD está implementado en una sola plataforma (ordenador) desde donde se gestiona directamente, de modo centralizado, la totalidad de los recursos
- Los sistemas de BD centralizados se ejecutan en un único sistema informático, no interactúan con otra computadora.



- Una computadora de propósito general consiste de una o más CPU's y controladores para los dispositivos que se encuentren conectados a través de un bus de datos en común que proporciona el acceso a la memoria compartida
- Las CPU's poseen memorias cache locales donde se almacenan copias de ciertas partes de la memoria para acelerar el acceso a los datos.
- Cada controlador se encarga de un tipo de dispositivo específico .
- Las CPU's y Controladores pueden ejecutarse concurrentemente compitiendo así por el acceso a la memoria.
- La Memoria Caché reduce la disputa por el acceso a la memoria principal, ya que la CPU necesita acceder a la memoria compartida menos cantidad de veces.
- Hay dos formas de utilizar las computadoras:
 - ✓ Sistemas monousuarios
 - ✓ Sistemas mutiusuarios

✓ **SISTEMAS MONOUSUARIOS**

- **Computadoras personales y estaciones de trabajo.**
- **Se dispone de una sola CPU, con uno o dos discos físicos.**
- **El sistema operativo permite sólo un usuario.**
- **No poseen control de concurrencia.**
- **Las posibilidades de recupero son escasas y muy primitivas**
- **La mayoría no adminten SQL, proporcionan un lenguaje de consulta muy simple derivado del QBE (Query By Example)**
- **Las bases de datos que se ejecutan en estos sistemas disponen de multitarea, se ejecutan varios procesos a la vez en el mismo procesador, usando tiempo compartido, y de cara al usuario pareciese que los procesos se ejecutan en paralelo**

✓ **SISTEMAS MULTIUSUARIOS**

- Dispone de varias CPU's.
- El sistema operativo es multiusuario.
- Dan servicio a un gran número de usuarios que están conectados al sistema a través de terminales.
- Reciben el nombre de **SISTEMAS SERVIDORES**.
- Utilizan paralelismo disponiendo de unos pocos procesadores (normalmente entre 2 o 4) que comparten la misma memoria principal.
- En estas máquinas las bases de datos ejecutan las consultas en un único procesador, posibilitando la concurrencia de varias consultas.
- Permiten ejecutar un gran número de transacciones por segundo aumentando así la velocidad de respuesta, a pesar que cada transacción individualmente no se ejecuta más rápido.
- Estas máquinas tienen un gran número de procesadores y los sistemas de bases de datos intentan paralelizar las tareas simples.

DISTANCIAMIENTO A LOS SISTEMAS CENTRALIZADOS – INICIOS DE LA ARQUITECTURA CLIENT-SERVIDOR

- Los sistemas se han ido alejando de la arquitectura centralizada debido al aumento de la velocidad y potencia de las computadoras personales y decremento en sus precios.
- Las terminales conectadas a un sistema central han sido suplantadas por computadoras personales.
- La interfaz de usuario gestionada directamente por el sistema central pasó a ser gestionada por computadoras personales.
- Los sistemas centralizados hoy actúan como sistemas servidores que satisfacen las peticiones generadas por los sistemas clientes.
- Esta arquitectura fué la antecesora a la arquitectura *Cliente - Servidor*
- La arquitectura *Cliente - Servidor* se la entiende como la extensión lógica de la programación modular, es decir la división de un programa grande en pequeños programas (llamados *módulos*) facilitando así el desarrollo y el mantenimiento (*divide y vencerás*).

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

- Por lo tanto motivaron el enfoque del procesamiento Cliente-Servidor:
 - ✓ El uso de recursos de cómputo remoto para realizar complejos procesos empresariales que consisten de una diversidad de subtareas.
 - ✓ Los clientes y servidores pueden estar ordenados a través de computadoras en red para dividir el trabajo complejo en unidades más manejables
 - ✓ El ordenamiento más simple es dividir el trabajo entre clientes que procesen en computadoras personales y un servidor que procese en una computadora separada

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

- Generalidades
- ✓ Los clientes acceden a los recursos vía aplicaciones (programas) para entregar la información vía algún protocolo de comunicación (por ejemplo TCP/IP).
- ✓ Así los sistemas Cliente/Servidor no están limitados solamente a aplicaciones de BD.
- ✓ Cualquier aplicación que contiene una interfaz de usuario (front-end, sección frontal o parte cliente) que se ejecute localmente en el cliente y un proceso que se ejecute en el servidor (back-end, sección posterior o sistema subyacente) está en formato de sistema Cliente/Servidor, o simplemente se puede decir que: un cliente es un programa que envía solicitudes a un servidor y un servidor procesa las solicitudes del cliente
- ✓ Conceptualmente las plataformas Cliente/Servidor son parte del concepto de **SISTEMAS ABIERTOS**: todo tipo de computadora, sistemas operativos, protocolos de red y otros, hardware y software, pueden interconectarse y trabajar coordinadamente para lograr los objetivos del usuario
- ✓ Los sistemas cliente-servidor basados en estándares abiertos apoyan la interoperabilidad.
 - Interoperabilidad se refiere a la habilidad de dos o más sistemas para intercambiar y usar software y datos.
 - Los estándares abiertos promueven un mercado de proveedores, lo que conduce a costos más bajos y mayor calidad, siendo INTERNET el área con mayor estandarización

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

- Sistema Cliente/Servidor: uno o más clientes y uno o más servidores, conjuntamente con un sistema operativo y un sistema de comunicación entre procesos, forma un sistema compuesto que permite cómputo distribuido, análisis y presentación de los datos.
- Si existen múltiples servidores de procesamiento de bases de datos, cada uno de ellos deberá procesar una base de datos distinta, para que el sistema sea considerado un sistema *Cliente/Servidor*.
- Los clientes a través de la red, pueden realizar consultas al servidor, y el servidor tiene el control sobre los datos, pudiendo los clientes tener datos privados que residen en sus computadoras.
- El enfoque cliente-servidor apoya el crecimiento escalable de la capacidad de hardware y software. La escalabilidad se refiere a la habilidad para agregar y remover capacidad en unidades pequeñas:
 - ✓ La escalabilidad vertical se refiere a la habilidad para agregar capacidad en el lado del servidor.
 - ✓ La escalabilidad horizontal se refiere a la habilidad para agregar capacidad en el lado del cliente a través de estaciones de trabajo adicionales y movimiento de trabajo entre clientes y servidores.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Características:

- *Principales características de un sistema Cliente/Servidor:*

- El servidor presenta a todos los clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, solo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

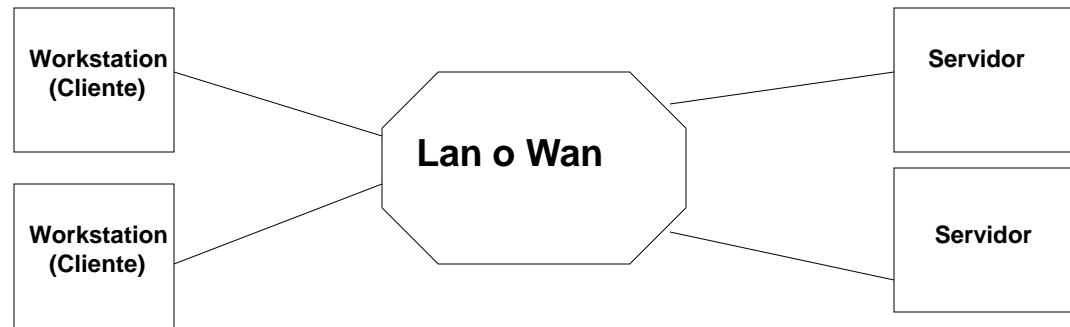
Características:

- Ejemplos de clientes:
 - ✓ Interfaces de usuarios para enviar comandos a un servidor.
 - ✓ APIs (Application Program Interface) para el desarrollo de aplicaciones distribuidas
 - ✓ Herramientas en el cliente para acceder a servidores remotos (ejemplo: servidores SQL)
 - ✓ Aplicaciones que solicitan acceso a servidores para algunos servicios
- Ejemplos de servidores:
 - ✓ Servidores de ventanas como X-windows
 - ✓ Servidores de archivos como NFS (Network File System: Sistemas de Archivos de la Red)
 - ✓ Servidores para el manejo de base de datos (servidores de SQL)
 - ✓ Servidores de diseño y manufactura asistidos por computador

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Partes de un sistema Cliente/Servidor:

- Los principales componentes de un sistema Cliente/Servidor son:
 - ✓ El núcleo (back-end o sección posterior). Es el SGBD propiamente (Servidor)
 - ✓ La Interfaz (front-end). Aplicaciones que funcionan sobre el SGBD (Cliente).



Ambiente Cliente/Servidor Generico

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Diferencia con un sistema centralizado multiusuario:

- El cliente es una terminal “con criterio”.
- El cliente tiene su propio sistema operativo y puede manejar entradas (teclado, razón, etc.) y salidas (pantalla, impresora local, sonido, etc.) sin el servidor.
- El servidor espera pasivamente la petición del cliente .
- La distribución del proceso permite al cliente ofrecer un ambiente más amigable que una terminal “sin criterio”.
- La complejidad del servidor disminuye si la comparamos con un mainframe
-
- El conjunto de un sistema Cliente/Servidor conduce a un ambiente flexible y dinámico.
- La parte cliente maneja la entrada de datos, acepta consulta de los usuarios y muestra los resultados.
- El servidor de la aplicación es quien procesa la consulta que le envía el cliente
- El servidor devuelve los resultados al cliente y este se los muestra al usuario.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

SECCION FRONTAL :

- Son las diversas aplicaciones ejecutadas dentro del SGBD, tanto las escritas por los usuarios como las “integradas” que son las proporcionadas por el proveedor del SGBD o por otros proveedores de aplicaciones.
- No existen diferencias entre las aplicaciones escritas por los usuarios y las integradas para la sección posterior. Todas las aplicaciones utilizan la misma interfaz de usuario con la sección posterior

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

FUNCIONES DEL CLIENTE:

- Administrar la interfaz gráfica de usuario.
- Aceptar datos del usuario.
- Procesar la lógica de la aplicación.
- Generar las solicitudes para la BD.
- Transmitir las solicitudes de la BD al servidor.
- Recibir los resultados del servidor.
- Dar formato a los resultados.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

COMO TRABAJA LA SECCION FRONTAL :

- La secuencia de eventos que ocurren cuando el usuario accede al servidor de la base de datos se puede generalizar en seis (6) pasos básicos:

✓ **Sistema Cliente:**

- 1) El usuario crea una consulta.
- 2) La aplicación cliente da formato a la consulta y la envía al dbms.

✓ **Sistema Servidor:**

- 3) El servidor comprueba los derechos de seguridad del usuario
- 4) El servidor procesa la consulta y devuelve los resultados.

✓ **Sistema Cliente:**

- 5) El cliente recibe la respuesta y le da formato
- 6) El usuario ve y/o manipula los datos

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

COMO TRABAJA LA SECCION FRONTAL :

- El usuario crea la consulta, cualquier acción que el usuario puede hacer sobre la base de datos (pedir datos, insertar, modificar, borrar), creada en el instante, preprogramada o almacenada con anterioridad.
- La aplicación cliente convierte la consulta al SQL utilizado por el servidor de la base de datos y la envía a través de la red al servidor
- El servidor verifica que el usuario tenga los permisos apropiados para la consulta de datos que requiere, de ser así verifica la consulta y envía los datos apropiados de vuelta al cliente.
- La aplicación cliente recibe la respuesta y le da formato para presentarla al usuario
- El usuario ve la respuesta en la pantalla, puede manipular los datos y comenzar de nuevo el proceso

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de aplicaciones clientes

- Aplicaciones escritas por usuarios escritas en un lenguaje de programación convencional o un lenguaje propio (Pl Sql).
- Aplicaciones o herramientas suministradas por los proveedores, para ayudar en el proceso de creación y ejecución de otras aplicaciones que realicen alguna tarea específica. La razón del uso de herramientas es para que los usuarios (finales principalmente) puedan crear aplicaciones sin escribir programas convencionales. Ej una herramienta sería un procesador de lenguaje de consultas para que los usuarios realicen las consultas a la BD.
- Clases de herramientas suministradas por los proveedores:
 - ✓ Procesadores de lenguaje de consultas
 - ✓ Generadores de reportes
 - ✓ Subsistemas de gráfica para negocios
 - ✓ Paquetes estadísticos
 - ✓ Administrar copias
 - ✓ Generadores de aplicaciones (incluyendo procesador de lenguajes de consulta de cuarta generación o 4 GL)
 - ✓ Para desarrollar aplicaciones, incluyendo productos para ingeniería de software asistida por computadora (CASE)

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

SECCION POSTERIOR :

- Es el SGBD en si.
- Permite llevar a cabo todas las funciones básicas de un SGBD:
 - ✓ Definición de datos
 - ✓ Manipulación de datos
 - ✓ Seguridad
 - ✓ Integridad
 - ✓ Transparencia
 - ✓ Establecer todos los aspectos de los niveles externo, conceptual e interno (arquitectura ANSI/SPARC)

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

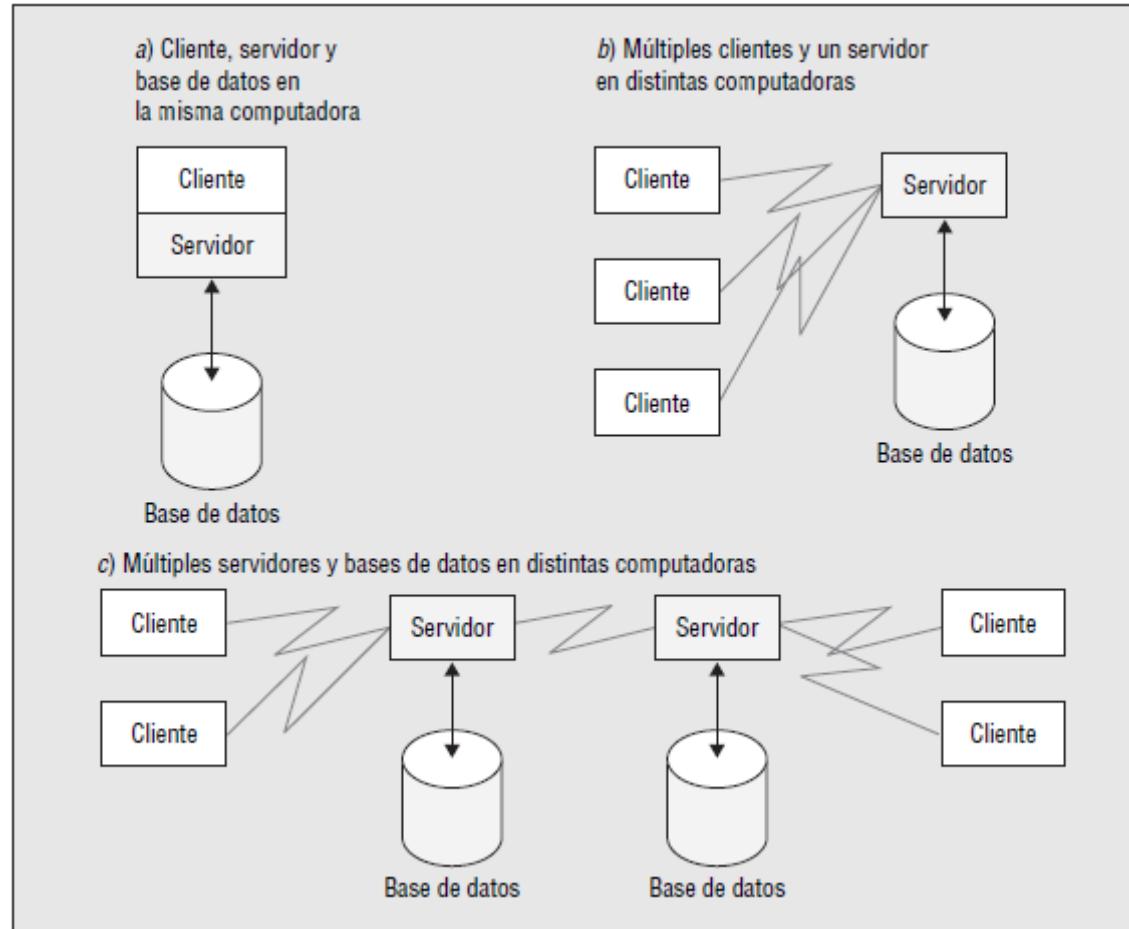
FUNCIONES DEL SERVIDOR:

- Aceptar las solicitudes de la base de datos de los clientes
- Procesar las solicitudes
- Dar formato a los resultados y transmitirlos al cliente
- Llevar a cabo la verificación de integridad
- Mantener los datos generales de la base de datos
- Proporcionar control de acceso concurrente
- Llevar a cabo la recuperación
- Optimizar el procesamiento de consultas y actualizaciones

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

- Así, la arquitectura cliente-servidor soporta muchas formas de distribuir el software y los datos en una red de cómputo, para mejorar el desempeño y la disponibilidad de los datos
- ✓ Esquema simple en el cual el software como los datos en una misma computadora (a)
- ✓ Esquemas que aprovechan las ventajas de una red:
 - El software del servidor y la base de datos se ubican en una computadora remota (b)
 - El software del servidor y la base de datos se ubican en varias computadoras remotas.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR



ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

- El DBMS tiene varias responsabilidades en una arquitectura cliente-servidor; proporciona el software que se puede ejecutar en el cliente y en el servidor.
- Es común que el software del cliente sea responsable de aceptar los comandos del usuario, desplegar los resultados y realizar algún procesamiento a los datos.
- El software del servidor valida las peticiones de los clientes, localiza las bases de datos remotas, las actualiza, en caso de ser necesario, y envía los datos al cliente en un formato que éste pueda entender.
- Las arquitecturas cliente-servidor proporcionan una forma flexible para que los DBMS interactúen con las computadoras de la red. La distribución del trabajo entre clientes y servidores y las alternativas disponibles para ubicar los datos y el software tienen su complejidad

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de servidores

- ***Iterativos***
- ***Concurrentes***
- ***De Transacciones***
- ***De datos***

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de servidores iterativos

- Pasos que realiza
 - ✓ Espera a que llegue la consulta de un cliente
 - ✓ Procesa la consulta
 - ✓ Envía la respuesta al cliente que envió la consulta
 - ✓ Vuelve al estado inicial

Inconveniente:

Durante el tiempo en que el servidor está procesando la consulta, ningún otro cliente es servido

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de servidores concurrentes

- Pasos que realiza
 - ✓ Espera a que llegue la consulta de un cliente
 - ✓ Cuando le llega una nueva consulta, comienza un nuevo proceso para manejar esa consulta (depende del sistema operativo la forma en como se realice este paso). El nuevo servidor maneja la totalidad de la consulta. Cuando se ha procesado completamente este nuevo proceso termina
 - ✓ Se vuelve al primer paso

Ventaja:

- ✓ *El servidor ejecuta un nuevo proceso para manejar cada consulta.*
- ✓ *Cada cliente tiene su "propio" servidor.*
- ✓ *Todo asumiendo que el sistema operativo permite la multiprogramación, clientes múltiples y servicio concurrente*

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de servidores de transacciones o de consultas

- Pueden proporcionar una interfaz para que los clientes envíen peticiones (acciones) al servidor
- El servidor ejecuta las peticiones y devuelve los resultados al cliente
- Los usuarios pueden especificar las peticiones con SQL
- Los usuarios pueden especificar sus peticiones mediante la interfaz de una aplicación utilizando llamadas a procedimientos remotos (RPC: “Remote Procedure Call”)
- Responde a la división funcional entre la parte visible al usuario y el sistema subyacente
- Las computadoras personales gestionan la parte visible al usuario por el procesamiento que requiere el código de la interfaz gráfica.
- Los sistemas servidores tienen almacenado un gran volumen de datos y soportan la funcionalidad del sistema subyacente

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Interacción entre clientes y servidores

- Se han desarrollado diferentes normas para la interacción entre clientes y servidores, por ejemplo ODBC (Open Database Connectivity: Conectividad abierta de base de datos), para permitir que los clientes generen instrucciones SQL para enviarlas al servidor en donde se ejecuten, así cualquier cliente que utilice interfaz ODBC puede conectarse a cualquier servidor que proporcione esa interfaz, anteriormente se necesitaba que la parte visible al usuario y al sistema subyacente fuera provista por el mismo fabricante.
- ODBC es un conjunto de llamadas de bajo nivel que permite a las aplicaciones clientes intercambiar instrucciones con las aplicaciones del servidor y compartir datos, sin necesidad de conocer nada unas respecto de las otras.
- Las aplicaciones emplean módulos llamados *controladores de bases de datos* que unen la aplicación con el SGBD elegido, empleándose SQL como lenguaje de acceso a los datos. El SGBD debe proporcionar los controladores adecuados para que puedan ser utilizados por los lenguajes de programación que soporten ODBC.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de servidores de datos

- Se utilizan en redes de área local donde se alcanza una alta velocidad de conexión entre los clientes y el servidor
- Las máquinas clientes son comparables al servidor en cuanto al poder de procesamiento y se ejecutan tareas de cómputo intensivo.
- En este entorno tiene sentido enviar los datos a las máquinas clientes, allí realizar todo el procesamiento y luego enviar los datos de vuelta al servidor
- Los clientes deben poseer todas las funcionalidades del sistema subyacente
- La arquitectura de estos servidores se ha hecho muy ‘popular’ en los sistemas de base de datos orientadas a objetos, donde se da cabida a tipos de datos binarios para poder guardar código binario que es el que forman los objetos de sonido, imágenes, videos, etc. Los SGBDROO (SGBD Orientada a Objetos) proporcionan toda la potencia y robustez de los SGBD relacionales y permiten gestionar objetos de un modo nativo junto con los campos tradicionales (numéricos y caracteres).
- *El costo en tiempo de comunicación entre el cliente y el servidor es alto comparado al de acceso a una memoria local (milisegundos frente a menos de 100 nanosegundos)*

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Servidores de datos: Envío de Páginas

- La unidad de comunicación de datos puede ser de una página o de una tupla.
- Si la unidad de comunicación de datos es una tupla la sobrecarga por la transferencia de mensajes es alta comparada con el número de datos transmitido.
- En lugar de enviar una sola tupla es conveniente el envío de otras tuplas que es probable se utilicen en el futuro próximo.
- Preextracción: es la acción de buscar y enviar tuplas antes de que sea estrictamente necesario.
- Al residir varias tuplas en una página, el envío de páginas puede considerarse una forma de preextracción, así cuando un proceso desee acceder a una única tupla de la página, se enviarán todas las tuplas de esta página

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Servidores de datos: Bloqueos

- El bloqueo de las tuplas de datos que el servidor envía a los clientes la realiza habitualmente el propio servidor.
- Un inconveniente del envío de páginas es que los clientes pueden recibir implícitamente el bloqueo de todas las tuplas que residen en la página (los clientes adquieren bloques sobre todas las tuplas preextraídas, aunque no estén accediendo a alguna de ellas)
- El bloqueo de todas las tuplas que residen en la página puede detener innecesariamente el procesamiento de otros clientes que necesitan bloquear esos elementos.
- Se han desarrollado algunas técnicas para la liberación de bloqueos, en las que el servidor pide a los clientes que le devuelvan el control sobre los bloqueos de las tuplas extraídas. Si el cliente no necesita la tupla extraída, puede devolver los bloqueos sobre esa tupla al servidor para que éstos puedan ser asignados a otros clientes

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Servidores de datos: Cache de datos

- Los datos que se envían al cliente a favor de una transacción se puede alojar en un caché del cliente, incluso una vez completada la transacción, si hay espacio de almacenamiento libre suficiente
- Las transacciones sucesivas en el mismo cliente pueden hacer uso de los datos en caché
- *Coherencia de la caché:* la transacción que encuentra los datos en la caché debe asegurarse que estén actualizados dado que esos datos pudieron haber sido modificados por otros clientes, entonces se debe establecer una comunicación con el servidor para comprobar la validez de los datos y adquirir un bloqueo sobre ellos.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Servidores de datos: Cache de bloqueos

- Los bloqueos, también, se pueden almacenar en la memoria caché del cliente si la utilización de los datos está prácticamente dividida entre los clientes.
- Ejemplo:
 - ✓ En la memoria caché se encuentra el elemento de dato buscado y el bloqueo requerido para acceder al mismo, así el cliente puede acceder al elemento de dato sin comunicar nada al servidor.
 - ✓ El servidor debe seguir el “rastro” de los bloqueos en caché
 - ✓ Si un cliente solicita un bloqueo al servidor, este debe comunicar a todos los clientes sobre el bloqueo sobre el elemento de datos que se encuentra en caché de otros clientes.
 - ✓ La tarea se vuelve muy complicada al tener en cuenta las posibles fallas en las máquinas
- Esta técnica se diferencia de la liberación de bloqueos en la caché de bloqueo que se realiza a través de las transacciones, de otra forma las dos técnicas serían similares

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

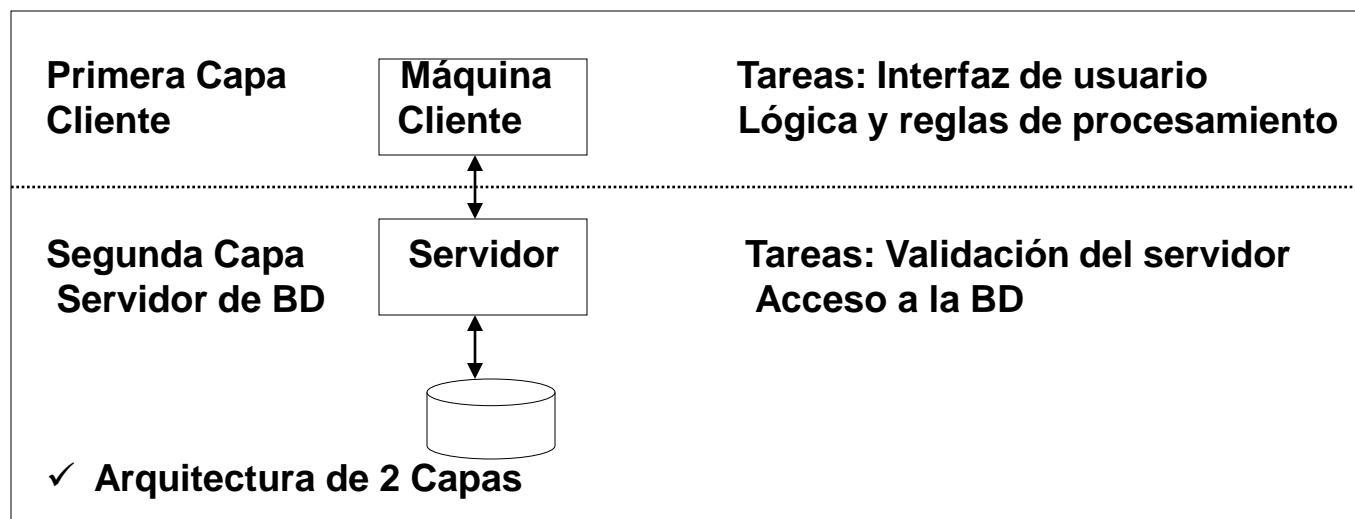
Tipos de Arquitecturas:

- ***Arquitectura de 2 capas***
- ***Arquitectura de 3 capas***

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de Arquitecturas: Arquitectura de 2 capas

- Esta es la arquitectura tradicional.
- Tiene 3 componentes:
 - ✓ Interfaz de usuario
 - ✓ Gestión del procesamiento
 - ✓ Gestión de la base de datos



ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de Arquitecturas: Hay 2 tipos de arquitectura Cliente/Servidor de 2 capas

- **Thick Client:** la mayor parte de la lógica de la aplicación (gestión del procesamiento) reside junto a la lógica de la presentación (interfaz de usuario) en el cliente, con la porción de acceso a datos en el servidor
- **Thin Clients:** solo la lógica de la presentación reside en el cliente, con el acceso a los datos y la mayoría de la lógica de la aplicación en el servidor

Es posible que un servidor funcione como cliente de otro servidor: Diseño de dos capas encadenados

Limitaciones:

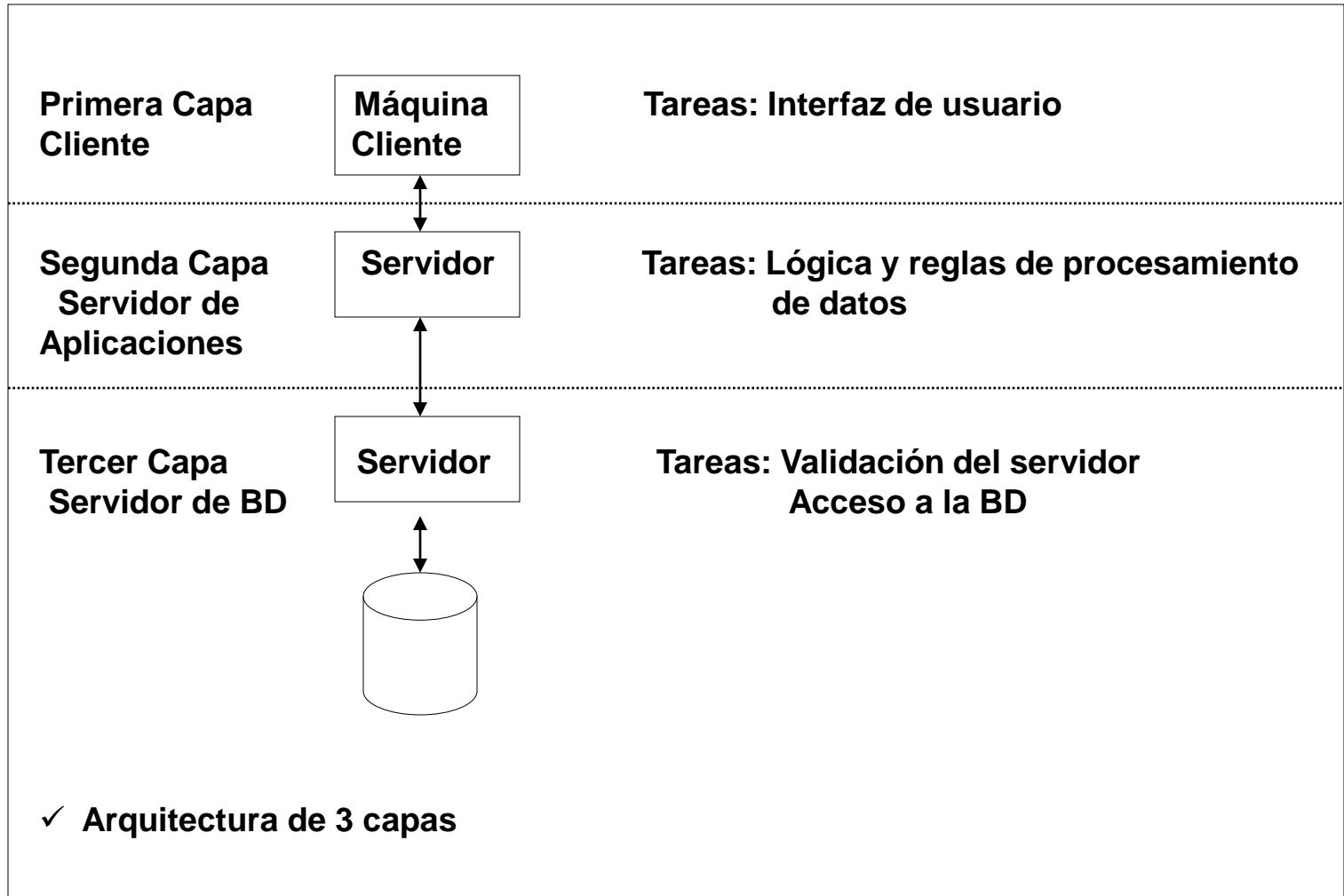
- El máximo número de usuarios es limitado, sino se excede la capacidad de procesamiento
- No hay independencia entre la interfaz de usuario y los tratamientos, lo que va en contra de la evolución de las aplicaciones
- Es difícil relocate las capas de tratamiento consumidoras de cálculo
- Bajo esta arquitectura se complica reutilizar los programas desarrollados

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de Arquitecturas: Arquitectura de 3 capas

- Surge para superar las limitaciones de la arquitectura de 2 capas
- La tercer capa Servidor intermedio está entre la interfaz de usuario (Cliente) y el gestor de la base de datos (Servidor).
- El Servidor intermedio proporciona gestión del procesamiento y en ella se ejecutan las reglas y lógica del procesamiento, permitiendo que un número “ilimitado” de usuarios interactúen en esta arquitectura
- Se utiliza para incrementar el rendimiento, la flexibilidad, mantenibilidad, reusabilidad y escalabilidad del sistema Cliente/Servidor, “escondiendo” la complejidad del procesamiento distribuido al usuario.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR



ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de Arquitecturas: Arquitectura de 3 capas

Limitaciones:

- Las herramientas de programación que soportan el diseño de arquitectura de 3 capas no proporcionan todos los servicios deseados necesarios para soportar un ambiente distribuido.
- Un problema potencial es la separación de la interfaz gráfica de usuario, la lógica de gestión de procesamiento y la lógica de datos.
- La ubicación de una función particular en una capa u otra debería basarse en criterios como los siguientes:
 - ✓ Facilidad de desarrollo y comprobación
 - ✓ Facilidad de administración
 - ✓ Escalabilidad de los servidores
 - ✓ Funcionamiento (incluyendo procesamiento y carga de la red)

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

Tipos de Arquitecturas: Middleware

- Es el software que proporciona un conjunto de servicios que permite el acceso transparente a los recursos en una red
- Es un módulo intermedio que actúa como conductor entre dos módulos de software, los cuales para compartir datos no necesitan saber como comunicarse entre ellos, sino como comunicarse con el módulo de middleware
- Es el encargado de acceso a los datos acepta las consultas y datos recuperados directamente de la aplicación y los transmite por la red
- Es el responsable de enviar de vuelta a la aplicación, los datos de interés y de la generación de los códigos de error.

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

VENTAJAS E INCONVENIENTES:

Ventajas

- *Interoperabilidad*: los componentes claves trabajan juntos: cliente, servidor y red
- *Flexibilidad*: la nueva tecnología puede incorporarse al sistema
- *Escalabilidad*: cualquiera de los elementos del sistema puede reemplazarse cuando es necesario, sin impactar sobre otros elementos. Si la base de datos crece, las computadoras clientes no tienen que equiparse con memoria o discos adicionales. Los cambios afectan solo a la computadora en la que se ejecuta la BD
- *Usabilidad*: facilita el uso a los usuarios
- *Integridad de los datos*: entidades, dominios e integridad referencial son mantenidas en el servidor de la base de datos
- *Accesibilidad*: los datos pueden ser accedidos desde múltiples clientes
- *Rendimiento*: se puede optimizar el rendimiento por hardware y procesos
- *Seguridad*: la seguridad de los datos está centralizada en el servidor

ARQUITECTURA DE BASE DE DATOS CLIENTE/SERVIDOR

VENTAJAS E INCONVENIENTES:

Inconvenientes

- Alta complejidad tecnológica al tener que integrar una gran variedad de productos
- El mantenimiento de los sistemas es complejo por la interacción de diferentes partes de hardware y de software, distribuida por distintos proveedores, dificultando el diagnóstico de fallas
- Implica el rediseños de todos los elementos involucrados en los sistemas de información (modelos de datos, procesos, interfaces, comunicaciones, almacenamiento de datos, etc).
- No es sencillo dividir las aplicaciones entre la parte cliente y la parte servidor.
- Es más complicado asegurar un elevado grado de seguridad en una red de clientes y servidores que en un sistemas con una única computadora centralizada (cuanto más distribuida es la red, mayor es su vulnerabilidad).
- Problemas de congestión en la red pueden reducir el rendimiento del sistema por debajo de lo que se obtendría con una arquitectura centralizada.
- La interfaz gráfica de usuario puede a veces enlentecer el funcionamiento de la aplicación.
- La existencia de costos ocultos pueden encarecer su implantación (nuevas tecnologías, cambios organizativos, etc)

Resumen sobre base de datos

Base de datos relacional

- Los DBMS relacionales dominan el mercado de los DBMS de negocio.
- Los sistemas de bases de datos relacionales fueron desarrollados inicialmente por su familiaridad y simplicidad debido a que las tablas se usan para comunicar ideas en muchos terrenos del conocimiento, la terminología de tablas, filas y columnas no es desconocida del todo para la mayoría de los usuarios.
- A pesar de la familiaridad y de la simplicidad de las bases de datos relacionales, existe también una fuerte base matemática.
- Las matemáticas de las bases de datos relacionales incluyen la conceptualización de las tablas como conjuntos.
- La combinación de la familiaridad y la simplicidad con los fundamentos matemáticos es tan poderosa que los DBMS relacionales dominan comercialmente.

Bases de datos no SQL

- Se puede decir que la aparición del término NoSQL - “*Not Only SQL*”- aparece con la llegada de la web 2.0 ya que hasta ese momento solo subían contenido a la red aquellas empresas que tenían un portal, pero con la llegada de aplicaciones como Facebook, Twitter o YouTube, cualquier usuario podía subir contenido, provocando así un crecimiento exponencial de los datos.
- Algunas de sus características pertenecen al modelo no relacional.
- SQL no es el lenguaje de consulta/modificación de datos principal, aunque sí lo soportan, de ahí el nombre No Solo SQL.
- Almacena datos no estructurados o semiestructurados.
- Surge para el manejo y almacenamiento de grandes volúmenes de información.

Bases de datos embebidas

- La demanda de tecnología de bases de datos de objetos es impulsada por la necesidad de almacenar grandes cantidades de datos complejos y la integración de datos complejos con datos simples.
- El término “BD embebida” presta a confusión puesto que abarca múltiples tecnologías e implementaciones, y en la realidad, solo un pequeño número de SGBDs son utilizados en sistemas de tiempo real embebidos o dispositivos de electrónica de consumo.
- Una BD embebida provee una biblioteca de acceso en tiempo de ejecución para su manipulación en forma nativa desde la aplicación que la usa.
- Por sus características, los inicios de las BBDD embebidas son difusos.
- Existen varios ejemplos de SGBD embebidos comerciales desde, al menos, principios de la década del 80: Empress en 1981, Btrieve en 1982, etc.
- En la actualidad existen múltiples soluciones: Apache Derby, HSQLDB, H2, Berkeley DB, eXtremeDB, SQLite, etc.

Bases de datos orientadas a objetos

- La demanda de tecnología de bases de datos de objetos es impulsada por la necesidad de almacenar grandes cantidades de datos complejos y la integración de datos complejos con datos simples.
- Los primeros productos se usaron en aplicaciones donde no eran importantes las consultas, optimización de consulta y procesamiento de transacción *ad hoc*, enfatizaban el soporte para datos complejos en grandes sistemas de *software*.
- La mayoría de los DBMS orientados a objetos comenzaron como lenguajes de programación extendidos con soporte para objetos persistentes (es decir: objetos que existen después de que un programa termina).
- Gradualmente, los DBMS orientados a objetos proporcionaron consulta, optimización de consulta y eficiente soporte de transacción *ad hoc*.
- Aun así, hay dudas acerca de la habilidad de los DBMS orientados a objetos para proporcionar alto rendimiento para aplicaciones empresariales tradicionales y para competir efectivamente en el mercado.

Bases de datos en la nube

- Una base de datos en la nube es una base de datos que corre en una infraestructura propia o en la infraestructura de un proveedor de servicios en la nube.
- Existen diferentes tipos de nube (pública, privada, etc.).
- Existen diferentes tipos de servicios de administración (por personal de TI, por un proveedor).

MODULO 7

ARCHIVOS DE LOG, DEFINICIÓN Y USOS MÉTODOS DE RECUPERACIÓN

Marcela Russo
Laboratorio IV

Resguardo y recuperación

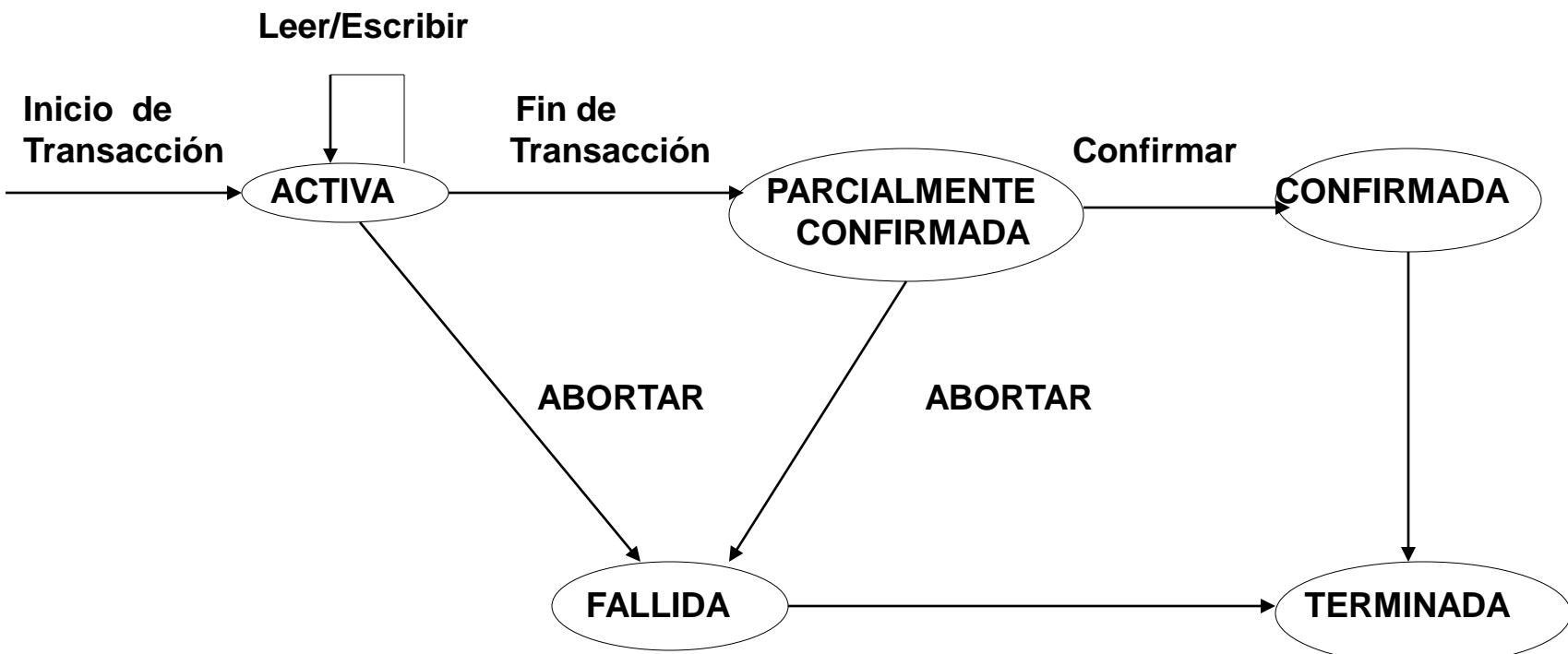
- Un DBMS debe proporcionar los medios para la recuperación ante fallas de hardware o de software.
- El subsistema de respaldo y recuperación del DBMS es el responsable de la recuperación.
- Si en medio de una transacción ocurre una falla, el subsistema de recuperación debe asegurar que la base de datos sea restaurada al estado en el que se encontraba, antes del inicio de la transacción.
- También es necesario realizar copias de seguridad de los archivos físicos para los casos en los que ocurra una falla catastrófica.
- Decimos que una base de datos está en un estado consistente si obedece a todas las restricciones de integridad definidas sobre ella.

PROTECCION DE DATOS ...Transacciones

- Como ya hemos visto, existen las **UNIDADES LÓGICAS DE EJECUCIÓN** llamadas **TRANSACCIONES**, con las siguientes características:
 - ✓ Las transacciones terminan con éxito y son grabadas en la base de datos.
 - ✓ Las transacciones fallan y la base de datos debe ser restaurada para regresar al estado en que se encontraba antes del inicio de la transacción.
 - ✓ Una transacción incluye una o más operaciones de acceso a la base de datos.
 - ✓ Transacciones de lectura-escritura (read-write) son las que involucran operaciones de inserción, eliminación, modificación (actualización) o recuperación.
 - ✓ Transacciones de solo lectura (read only) son las que involucran solo operaciones de lectura.
- Las transacciones pueden declararse en los programas de aplicación utilizando begin transaction y end transaction. Un programa de aplicación puede contener más de una transacción.
- Las transacciones pueden ejecutarse desde un lenguaje de alto nivel como el lenguaje SQL.
- El DBMS es el responsable de asegurar que todas las operaciones en la transacción se completen con éxito, es decir que alcancen el commit, y su efecto se registre de forma permanente en la base de datos, o que la transacción no tenga ningún efecto sobre la base de datos (transacción abortada)

PROTECCION DE DATOS ...Transacciones

- Para fines de recuperación, el sistema necesita realizar un seguimiento de cada transacción, para saber cuando comienza, cuando finaliza y como lo hizo (commit o abort) . Por lo tanto, el subsistema de recuperación del DBMS debe realizar un seguimiento de las siguientes operaciones:
 - ✓ **BEGIN_TRANSACTION:** Indica el comienzo de la ejecución de la transacción.
 - ✓ **READ o WRITE:** especifican operaciones de lectura o escritura en la base de datos.
 - ✓ **END_TRANSACTION:** indica que las operaciones de transacciones lectura y escritura han finalizado y marca el final de la ejecución de la transacción. Sin embargo, en este punto puede ser necesario comprobar si los cambios introducidos por la transacción se puede aplicar permanentemente a la base de datos (commit) o si la transacción tiene que ser abortada (abort) por alguna falla.
 - ✓ **COMMIT_TRANSACTION:** indica un final exitoso de la transacción para que cualquier cambio ejecutado por la transacción quede confirmado de forma segura en la base de datos y no se deshará.
 - ✓ **ROLLBACK (ó ABORT):** indica que la transacción ha finalizado sin éxito, para deshacer los cambios o efectos que la misma haya realizado sobre la base de datos.

Diagrama de transición de Estados en la ejecución de una transacción

- Una transacción pasa a un estado activo inmediatamente después de que comience la ejecución, donde puede ejecutar sus operaciones de LECTURA y ESCRITURA.
- Cuando finaliza la transacción, pasa al estado PARCIALMENTE CONFIRMADO. En este punto:
 - Algunos tipos de protocolos de control de concurrencia pueden hacer comprobaciones adicionales para ver si la transacción puede alcanzar, o no, el commit.
 - Algunos protocolos de recuperación necesitan asegurarse de que una falla del sistema no resulte en la imposibilidad de registrar los cambios de la transacción de forma permanente (normalmente registrando los cambios en el sistema de log).
 - Si estas comprobaciones tienen éxito, la transacción ha alcanzado su punto de compromiso y entra en el estado CONFIRMADO. Esto implica que la transacción ha concluido su ejecución con éxito y todos sus cambios deben ser registrados permanentemente en la base de datos, incluso si ocurre una falla en el sistema.
 - Sin embargo, una transacción puede pasar al estado FALLIDA si una de las comprobaciones falla o si la transacción es abortada durante su estado ACTIVA. Es posible que la transacción tenga que revertirse para deshacer el efecto de sus operaciones WRITE en la base de datos.
- El estado TERMINADA corresponde a la transacción saliendo del sistema.
- La información de la transacción que se mantiene en las tablas del sistema mientras la transacción se ha estado ejecutando y se elimina cuando termina la transacción.
- Las transacciones fallidas o abortadas se pueden reiniciar más tarde, ya sea automáticamente o después de que el usuario las haya vuelto a enviar, como transacciones nuevas.

Fallas que pueden ocurrir

- En los sistema pueden ocurrir fallas que ponen en riesgo la integridad y la existencia misma de la BD, provocando la perdida de los datos.
- Las fallas pueden ocurrir en las transacciones, en el sistema o en los medios, como por ejemplo:
 - ✓ Fallas de la computadora, provocando la caída del sistema (Ej. Falla en la memoria)
 - ✓ Fallas por error de la transacción o del sistema (Ej. División por 0)
 - ✓ Fallas por excepciones no programadas (Ej. datos inexistentes para realizar la transacción).
 - ✓ Fallas en el control de concurrencia (Ej. Para evitar abrazos mortales)
 - ✓ Fallas de disco (Ej. bloqueo del cabezal de lectura/escritura del disco)
 - ✓ Fallas por catástrofes o problemas físicos (Ej. fallas de energía o aire acondicionado, incendio, robo, sabotaje, sobreescritura discos o cintas por error, etc.)
- Los problemas aparecen en las operaciones de actualización de la base de datos, la LECTURA NO ES causal de problemas.
- El sistema debe mantener suficiente información para una rápida recuperación ante la ocurrencia de una falla.

Propiedades deseables de una transacción y los subsistemas del DBMS:

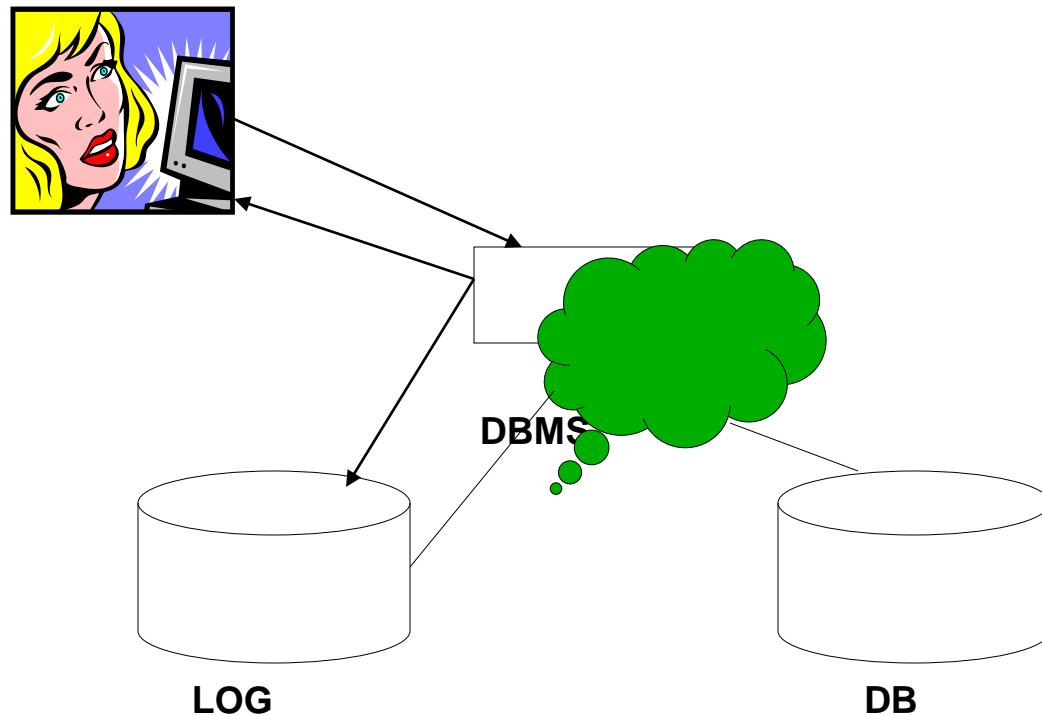
- Las transacciones deben poseer varias propiedades, a menudo denominadas propiedades ACID; deben hacerse cumplir mediante el control de concurrencia y los métodos de recuperación del SGBD
- La base de datos que cumple con la prueba ACID, asegura que los cambios realizados por las transacciones finalizadas y confirmadas con éxito, perduran aunque el sistema falle con posterioridad.

Prueba ACID	Funcionalidad	Subsistema de
A tomicidad	Se ejecuta todo o nada	Respaldo-Recuperación
C onsistencia	La base de datos pasa de un estado consistente a otro estado consistente	Integridad, pueden intervenir los programadores
I slamiento	La transacción no muestra los cambios que produce hasta que finaliza	Control de Concurrencia
D urabilidad	Una vez que la transacción finaliza con éxito y es confirmada, los cambios perduran aunque el sistema falle después	Respaldo-Recuperación

- El subsistema de control de concurrencia y el subsistema de respaldo-recuperación, están integrados dentro del funcionamiento base de datos en tiempo de ejecución.

Sistema de LOGS

- *El Sistema de LOGs de la base de datos es el registro de todas las operaciones que se realizan en cada transacción*



Sistema de LOG

- Para poder recuperarse de las fallas que afectan las transacciones, el DBMS mantiene un log para realizar un seguimiento de todas las operaciones de transacción que afectan los datos de la base de datos, así como otra información de la transacción que pueda ser necesaria para permitir recuperación de fallas.
- El log es un archivo secuencial, se mantiene en el disco y se lo respalda periódicamente en otro medio de almacenamiento, como por ejemplo en cinta, para protegerlo y evitar que se pierda.
- Los registros se van agregando al final del último registro grabado. No se ve afectado por ningún tipo de falla a excepción de fallas de disco o alguna catástrofe.
- Generalmente, uno o más buffers de la memoria principal, llamados log buffers, mantienen la última parte del archivo de log, para que las entradas de log se agreguen primero al log buffer de la memoria principal. Cuando el log buffer se completa, o cuando ocurren determinadas condiciones, el log buffer que está en la memoria se agrega al final del archivo de log que está en el disco.

LOGS RECORDS

- Se denomina *logs records* a los tipos de entradas que se escriben en el archivo de log y la acción correspondiente para cada registro de log.
- En estas entradas, T identifica un transacción única generada automáticamente por el sistema para cada transacción y que se utiliza para identificarla:
 1. *Start_transaction, T*: Indica que la transacción T inició la ejecución.
 2. *Write_item, T, X, Old_value, New_value*: Indica que la transacción T cambió el valor del ítem X de la base de datos de *old_value* a *new_value*
 3. *Read_item, T, X*: Indica que la transacción T ha leído el valor del ítem X de la base de datos
 4. *Commit, T*: Indica que la transacción T se ha completado satisfactoriamente, y confirma que lo ejecutado puede ser registrado permanentemente en la base de datos, es decir que se puede ejecutar el commit.
 5. *[Abort, T]*: Indica que la transacción T fue abortada

LOGS RECORDS

- Los protocolos de recuperación que evitan los rollback en cascada , incluidos en casi todos los protocolos prácticos, no requieren que las operaciones READ sean escritas en el sistema de logs.
- Si el log también se utiliza para otros fines, como por ejemplo la auditoría, la cual se utiliza para hacer un seguimiento de todas las operaciones de la base de datos, las operaciones de READ pueden incluirse en el log.
- Algunos protocolos de recuperación requieren entradas de ESCRITURA más simples, que solo incluyen new_value u old_value en lugar de incluir ambos.
- Debido a que el log contiene una entrada por cada operación de ESCRITURA que cambia el valor de un ítem en la base de datos, es posible deshacer (UNDO) el efecto de estas operaciones de ESCRITURA de una transacción T rastreando hacia atrás a través del log y restableciendo todos los datos cambiados por una operación WRITE de la transacción a sus valor anterior (old_value).
- También puede ser necesario rehacer (REDO) una operación si la transacción tiene sus actualizaciones registradas en el registro, pero se produce una falla antes de que el sistema pueda asegurar que todos los nuevos_valores (new_value) se han escrito desde los búferes de memoria principal a los archivos físicos que soportan a la base de datos

Punto de COMMIT de una transacción

- Una transacción T alcanza su punto de COMMIT cuando todas sus operaciones se han ejecutado con éxito y el efecto de todas las operaciones ejecutadas en la transacción se han registrado en el log.
- Se dice que la transacción fue “comiteada” y el efecto debe registrarse permanentemente en los archivos físicos de la base de datos.
- La transacción escribe un punto de COMMIT [commit, T] en el LOG.
- Si ocurre una falla del sistema, es posible buscar en el LOG todas las transacciones que han escrito un [start_transaction, T] dentro del mismo, pero que no han registrado aún el [commit, T], pueden tener que revertirse para deshacer (UNDO) su efecto en la base de datos durante el proceso de recuperación.
- Las transacciones que han registrado su [commit, T] en el LOG, también registraron todas sus operaciones de ESCRITURA en el LOG, por lo que su efecto en la base de datos se puede rehacer (REDO) desde los registros de LOG.
- El archivo de LOG debe mantenerse en el disco.

COMMIT de una transacción

- Actualizar un archivo de disco implica copiar el bloque apropiado del archivo del disco a un búfer en memoria principal, actualizando el búfer en la memoria principal y copiando el búfer a disco.
- Es común mantener uno o más bloques del archivo de log en los bufferes de la memoria principal, llamado log buffer, hasta que se llenen con entradas en el log y luego volver a escribirlos en el disco una sola vez, en lugar de escribirlos en el disco cada vez que se agrega una entrada en el log.
- Esto ahorra la sobrecarga de múltiples escrituras del mismo buffer de log en disco.
- Ante un crash del sistema, sólo las entradas de log que hayan sido escritas nuevamente en el disco se consideran en el proceso de recuperación si el contenido de la memoria principal se pierde. Por lo tanto, antes que una transacción alcance su punto de commit, deberá escribirse en el disco, cualquier parte del log que aún no haya sido escrito en el mismo.
- El proceso mencionado se denomina force-writing (escritura forzada)

PARA que los LOG sean UTILES DEBEN RESIDIR EN UN ALMACENAMIENTO ESTABLE

- La escritura se realiza de dos formas:
 - SINCRONA: por cada registro en el LOG, la página de registro se guarda en un almacenamiento estable
 - ASINCRONA: Las páginas de registro se mueven en forma periódica o cuando el LOG se completa
- Cada vez que una transacción va a realizar una operación en la BD, se agrega un registro en el LOG.
- Típicamente esta actualización se realiza en un buffer de memoria, por lo tanto cuando una transacción se confirma:
 - Primero se baja a disco todo el LOG de la transacción que aún no haya sido guardado
 - Luego se actualiza la BD con los efectos de la transacción
- NUNCA SE DEBEN PERDER A LA VEZ LA BD Y EL LOG, por lo tanto:
 - Deben estar en discos distintos
 - Deben realizarse respaldos independientes de uno y otro

DBMS- Políticas específicas de reemplazo de buffers

- El DBMS cache mantiene las páginas del disco que contienen información que esté siendo procesada, en los buffers de memoria principal.
- Si todos los buffers en el DBMS Cache se encuentran ocupados y nuevas páginas del disco deben cargarse en la memoria principal desde el disco, se aplicará una política de reemplazo de páginas para seleccionar los buffers que serán reemplazados.
- Las políticas o métodos de reemplazo son:
 - Separación de Dominios
 - Hot Set (Configuración en caliente)
 - DBMIN

Protocolo de recuperación y algoritmos de recuperación

- El log del sistema generalmente mantiene información sobre los cambios que se aplicaron a los ítems de datos por parte de varias transacciones y la recuperación de errores de transacción generalmente significa que la base de datos se restaura a el estado consistente más reciente antes del momento de la falla.
- Un esquema de recuperación y categorización de algoritmos de recuperación típica se puede resumir brevemente de la siguiente manera:
 - Si una gran parte de la base de datos se encuentra dañada debido a una falla catastrófica, como un crash del disco, el método de recuperación restaura una copia de la base de datos de la que se realizó una copia de seguridad en otro almacenamiento (ejemplo: en cinta) y reconstruye una versión al estado más actual volviendo a aplicar, o rehaciendo las operaciones de transacciones que alcanzaron el commit, desde el log resguardado, hasta el momento de la falla.
 - Si la base de datos en el disco no está físicamente dañada y no ocurrió una falla catastrófica, la estrategia de recuperación es identificar cualquier cambio que pueda causar una inconsistencia en la base de datos. Por ejemplo, una transacción que ha actualizado algunos ítems de la base de datos en el disco pero no ha sido confirmado necesita que sus cambios se reviertan deshaciendo sus operaciones de escritura. También puede ser necesario rehacer algunas operaciones, para restaurar la base de datos a un estado consistente; por ejemplo, si la transacción fue confirmada, pero algunas de sus operaciones de escritura aún no se han sido escrito en el disco.
 - Para fallas no catastróficas, el protocolo de recuperación no necesita una copia de la base de datos de la que se realizó una copia de seguridad en otro almacenamiento de archivo completa de la base de datos. Más bien, las entradas mantenidas en el log del sistema en línea en el disco, se analizan para determinar la acciones apropiada para la recuperación.

LOGS: Algoritmos de recuperación

- ***Si hubo un desastre (se pierde el disco, etc.) entonces:***
 - Se debe recuperar el último respaldo conocido de la BD (último resguardo válido)
 - Se debe recuperar el LOG hasta donde se pueda
 - Se deben rehacer (redo) todas las operaciones indicadas en el LOG desde el momento en se hizo el respaldo de la DB

- ***Si la falla fue menor (corte de energía, falla de la red, etc.)***
 - Puede ser que haya que deshacer (undo) cambios ya realizados.
 - Puede ser que haya que rehacer cambios que no se hayan confirmado

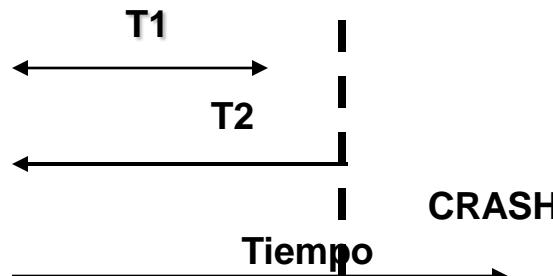
Algoritmos de recuperación:

- Conceptualmente, podemos distinguir dos políticas principales para la recuperación de situaciones no catastróficas ante fallas en las transacciones
 - Actualización Diferida: Cada transacción trabaja en un área local de disco o memoria y recién se baja al disco después que la transacción alcanza el COMMIT. Si hay un Abort o una Falla, no es necesario deshacer ninguna operación (No UNDO/REDO)
 - Actualización Inmediata: La base de datos se actualiza antes que la transacción alcance el commit. Si hay un Abort o una Falla, se deben deshacer las operaciones de la transacción. Siempre se graba primero el log para garantizar la recuperación

LOGS: BeFore Image (old_value) y AFter Image (new_value)

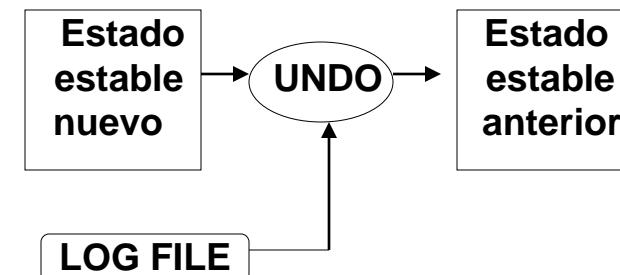
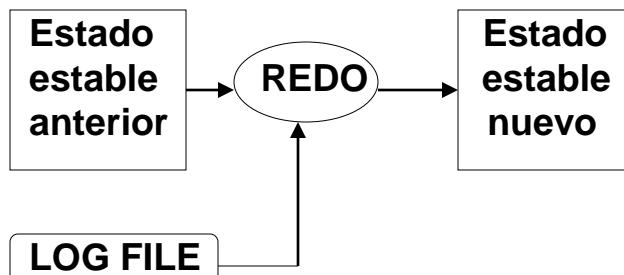
- Para que el mecanismo de recuperación sea viable, es necesario considerar al menos dos valores para cada ítem:
 - ✓ Before Image (BFIM): El valor del ítem antes de ser actualizado por una transacción.
 - ✓ After Image (AFIM): El valor del ítem después de ser actualizado por una transacción.
- De esta forma, los registros del Log pueden estar clasificados en:
 - De Undo: Contienen la operación y el BFIM
 - De Redo: Contienen la operación y el AFIM
 - Combinados: Contienen la operación y los dos. Se usan en estrategias UNDO/REDO.

LOGS



- T1 terminó con éxito (COMMIT).
- T2 no pudo concluir.
- Es probable que los cambios de T1 no hayan sido escritos a la BD estable y por lo tanto se deben realizar todas las operaciones de T1 nuevamente (REDO)

- Por otra parte es posible que otras operaciones de T2 hayan pasado al almacenamiento estable, por lo que es necesario deshacer las mismas (UNDO).



RECUPERACION

- **CHECKPOINTS:**

- Es el punto en el cual las operaciones en la BD, sincronizan las páginas en el disco con las páginas en la memoria compartida y conjunto de buffer.
- Cuando un CHECKPOINT se completa, se completan las operaciones físicas y la base de datos está físicamente consistente.
- Registro del Log que indica que todos los buffers modificados de la base fueron actualizados al disco.

ESTABLECER PUNTOS DE REVISION IMPLICA:

- Grabar físicamente el contenido de los buffers de datos a la base de datos física.
- Grabar físicamente un registro de punto de revisión en el archivo de log.
- Los puntos marcados como CHECKPOINT, permiten la recuperación de la base de datos en línea, es decir, después de la caída del sistema se obtiene la dirección del registro más reciente y se recorre el archivo de log desde el punto marcado como checkpoint.
- Ninguna transacción que aparece en el Log antes que el checkpoint necesita rehacerse (Redo).

RECUPERACION

- **CHECKPOINTS:** El checkpoint consiste de los siguientes pasos:

- ✓ Suspender la ejecución de todas las transacciones.
- ✓ Grabar todos los buffers modificados en el disco
- ✓ Registrar el checkpoint en el log y grabar el Log en disco.
- ✓ Permitir la continuación de las transacciones

Write-Ahead Logging

- ✓ Es una estrategia en la cual el mecanismo de recuperación garantiza que el BFIM está en el Log y el Log está en el disco **ANTES** que el AFIM actualice el BFIM en la base en el disco.
- ✓ Un protocolo WAL(***Write-Ahead Logging***) podría ser el siguiente:
 - ✓ Un AFIM de un ítem no puede actualizar el BFIM de ese ítem hasta que todos los registros del Log de tipo Undo para esa transacción haya sido escritos en el disco.
 - ✓ Nunca se puede completar el commit de una transacción hasta que todos los registros de Log (Undo o Redo) hayan sido escritos en el disco.
- ✓ Para implementar esto, se deben llevar listas de transacciones activas, confirmadas y abortadas.

RECUPERACION

- **ROLLBACK:**

- ✓ Se debe realizar cuando una transacción aborta o no termina
- ✓ Es la recuperación de todos los BFIM's de los ítems que modificó esa transacción y de todas las transacciones que leyeron de la que abortó. (Abortos en Cascada)
- ✓ Los Abortos (o Rollbacks) en Cascada pueden consumir mucho tiempo por lo que deben evitarse garantizando historias estrictas.

RECUPERACION

- **ALGORITMOS DE RECUPERACION**
- ✓ Cualquier algoritmo de recuperación debería implementar las siguientes operaciones o procedimientos:
 - Recuperación: Indica cual es la estrategia general de recuperación
 - Undo: Indica cómo se debe hacer el undo de una operación.
 - Redo: Indica como se debe hacer el redo de una operación.
- ✓ Es fundamental que la operación de Redo de los mismos resultados si ejecuta varias veces.
- ✓ En general, se asume que se están generando historias estrictas y serializables.

RECUPERACION

- RECUPERACION BASADA EN ACTUALIZACION DIFERIDA
- ✓ Idea Básica: Demorar la escritura de la base en el disco hasta que la transacción alcance el commit.
- ✓ Para esto, durante la ejecución la actualización se realiza en el Log y en los buffers o en un área local a la transacción.
- ✓ El protocolo tiene dos reglas básicas:
 - Los cambios realizados por una transacción T nunca son grabados en el disco hasta que la T alcanza el commit.
 - Una transacción T nunca puede alcanzar el commit hasta que grabó todas sus operaciones de actualización en el Log y el log fue grabado en el disco

RECUPERACION

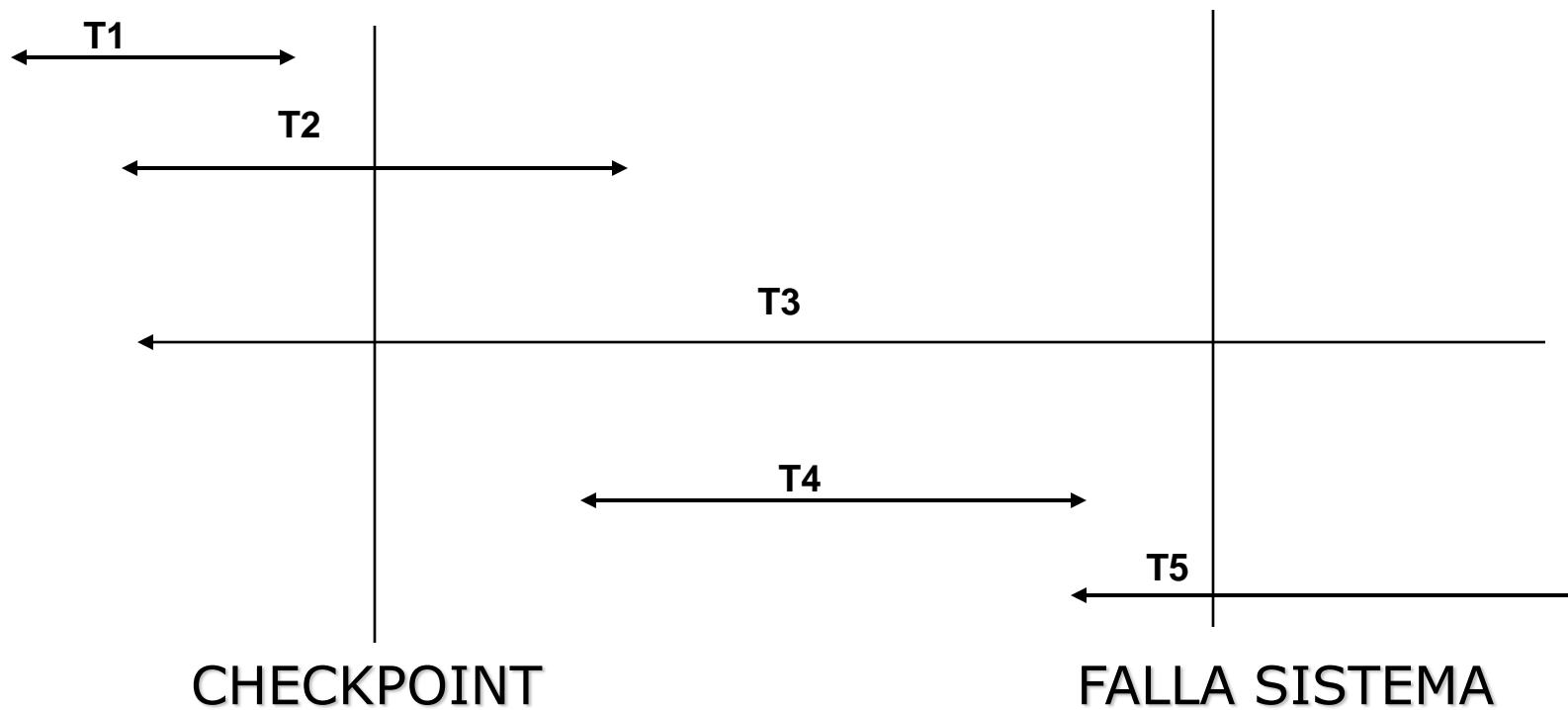
- RECUPERACION BASADA EN ACTUALIZACION INMEDIATA
- ✓ Idea Básica: grabar en el disco sin esperar al commit.
- ✓ Siempre se trabaja con la estrategia WAL (grabar el log antes que los datos).
- ✓ Dos familias de algoritmos::
 - Undo/No-Redo: hay que garantizar que todas las modificaciones efectivamente fueron grabadas en la base.
 - Undo/Redo: No hay que garantizar nada en particular.
- ✓ Undo/Redo es más complejo.
- ✓ Se asume que se generan historias estrictas y serializables.

RECUPERACION

La transacción T1 no se ve afectada por la falla del sistema, ni por el proceso de recuperación, por haberse completado antes del último punto de recuperación.

Las transacciones t2 y t4, a pesar de haber terminado no han sido grabadas en la base de datos, ya que esto ocurriría en un checkpoint, corresponde REDO

Las transacciones t3 y t5 deberán deshacerse (UNDO) ya que no han concluído.



RECUPERACION 2FC

Se utiliza para el control sobre transacciones distribuidas, el protocolo posee dos faces:

- 1. El coordinador hace que todos los participantes estén listos para escribir los resultados en la BD**
- 2. Todos escriben el resultado en la BD**

La transacción se completa a través de un compromiso global:

- 1. El coordinador aborta una transacción si y solamente si al menos un participante indica que se aborta.**
- 2. El coordinador hace un commit de la transacción si y solo si todos los participantes indican que se haga un commit.**

Definiciones

Definición de dato y quién lo administra

Definición de información

Definición de base de datos y quién la administra

Componentes de una base de datos

**Definición de un sistema de gestión de base de datos
(SGBD)**

Definición de dato y quién lo administra

¿Qué es un dato?

- Los datos son símbolos que describen un objeto, condición o situación.
- Son el conjunto básico de hechos referentes a una persona, cosa o transacción.
- Incluyen cosas como: tamaño, cantidad, descripción, volumen, nombre o lugar.
- Su contenido no es relevante para el comportamiento en un momento dado.
- Son elementos pasivos que al ser agrupados y organizados pueden ser utilizados por los programas, permitiendo así extraer información de utilidad.
- Los datos pasan a ser uno de los activos más importantes en las organizaciones.

Administrador de datos (AD)

- Tiene la responsabilidad central sobre los datos.
- Es imperativo que exista una persona que entienda los datos junto con las necesidades de la organización con respecto a esos datos.
- Es labor del administrador decidir qué datos deben ser almacenados en la base de datos y establecer políticas para mantener y manejar esos datos una vez almacenados.

Definición de información

La información la componen datos que se han colocado en un contexto significativo y útil, y se han comunicado a un receptor, quien la utiliza para tomar decisiones.

Implica:

- Datos transformados para comunicar un significado o conocimiento.
- Comunicación y recepción de conocimiento.

La información está compuesta de datos, imágenes, texto, documentos y voz, organizados en un contexto significativo.

Definición de base de datos y quién la administra

¿Qué es una base de datos (BD)?

- Colección de datos interrelacionados, resguardados juntos, sin redundancias perjudiciales o innecesarias.
- Su finalidad es servir a una o más aplicaciones.
- Los datos están resguardados de modo que resultan independientes de los programas que lo usan.
- Los datos pueden ser accedidos por varios usuarios en forma simultánea.
- Los datos están relacionados desde el punto de vista lógico, junto con la descripción de estos (metadata).
- Sirve para satisfacer las necesidades de información de una organización.

Administrador de base de datos (DBA)

- Técnico responsable de la implementación de las decisiones tomadas por el AD.
- Debe ser un profesional de IT.
- Crea la BD real e implementa los controles técnicos que hacen cumplir las políticas definidas por el AD.
- Responsable de que la BD opere con el rendimiento adecuado.
- Proporciona una gran variedad de servicios técnicos para asegurar la disponibilidad de la BD.

Componentes de una base de datos (BD)

- **Datos:** ya definido.
- **Hardware:** parte física del sistema informático que permite el almacenamiento de la información.
- **Software:** más conocido como administrador de una base de datos, es la parte lógica que permite realizar las diferentes operaciones con los datos almacenados.
- **Usuarios:**
 - Programadores de aplicaciones.
 - Administrador de BD.
 - Usuarios finales.

Definición de un sistema de gestión de base de datos (SGBD)

Sistema Gestor de Bases de Datos (SGBD)

- Es una colección de datos relacionados entre sí (BD), estructurados y organizados, y un conjunto coordinado de programas, procedimientos, lenguajes, etc., que acceden y gestionan esos datos.
- Se los conoce también como Data Base Management System (DBMS).
- Suministra a los usuarios los medios necesarios para describir, recuperar y manipular los datos almacenados en una BD.
- Mantiene la integridad, la confidencialidad y la seguridad.
- Permite al DBA especificar:
 - Los datos que integran la BD, su estructura y relaciones que existen entre ellos.
 - Definir las reglas de integridad semántica.
 - Controlar los accesos a la BD.
 - Definir las características físicas de la BD.
 - Definir las vistas lógicas de los usuarios.

Niveles de abstracción de una BD

Es la manera en la que los diseñadores de base de datos esconden a los usuarios la complejidad de cómo se almacenan y mantienen los datos; realizan esta operación a través varios niveles de abstracción.

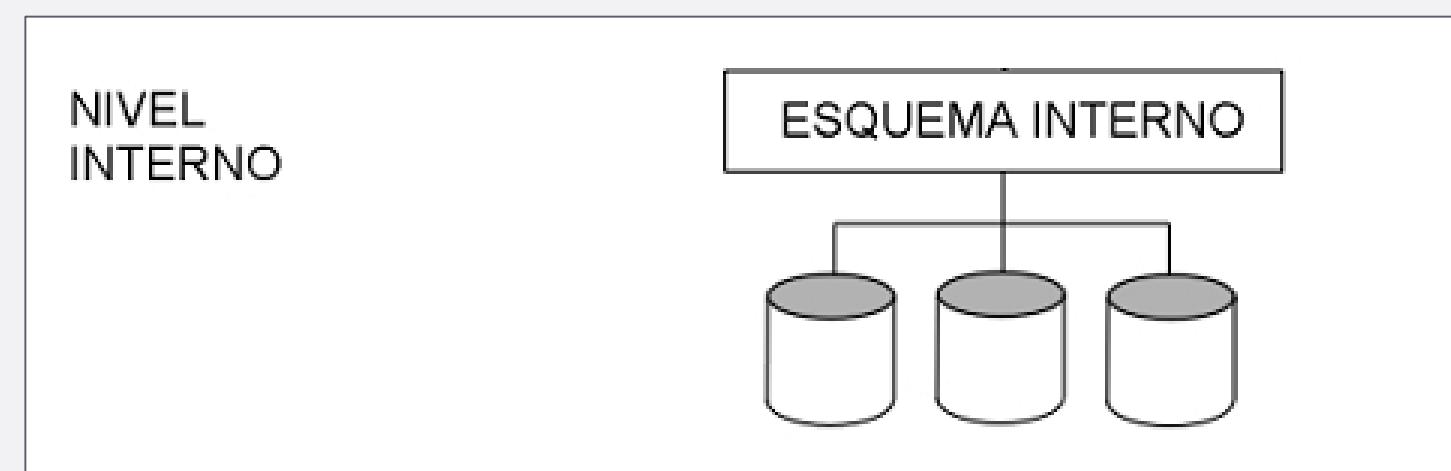
En 1975, el comité ANSI-SPARC (American National Standard Institute – Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la BD física.

En esta arquitectura el esquema de una BD se define en tres niveles de abstracción distintos:

- **Físico o interno.**
- **Conceptual o lógico.**
- **De vista o externo.**

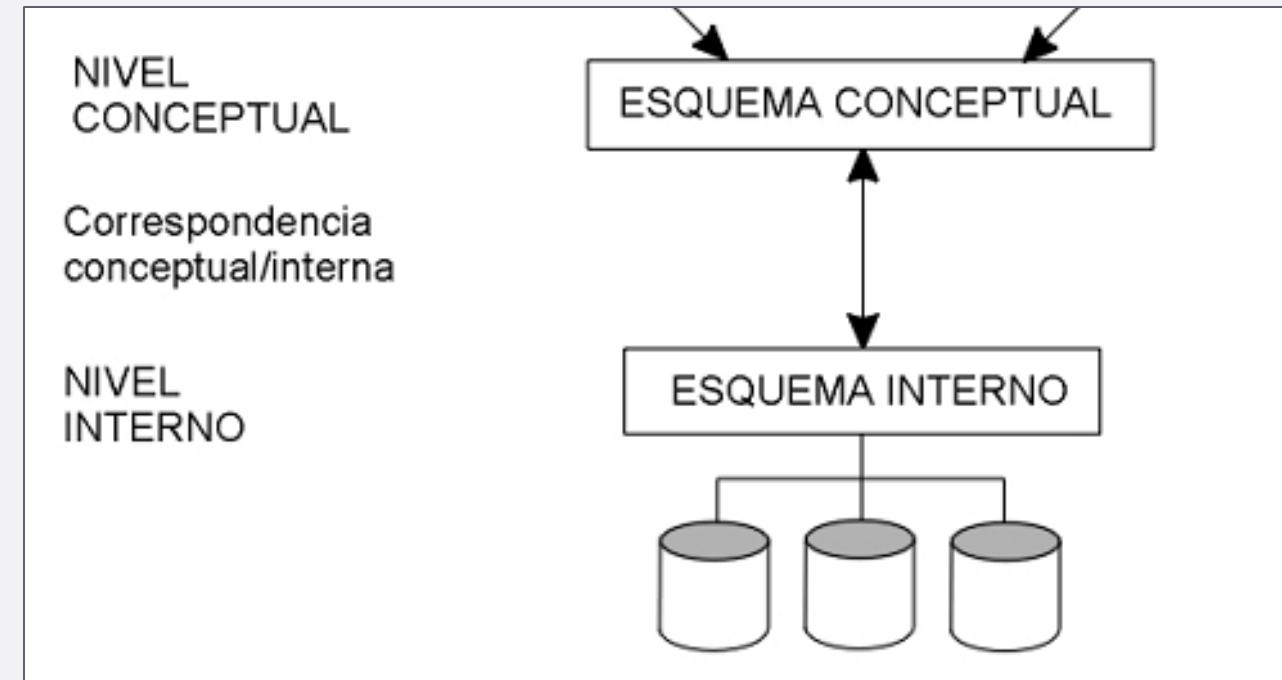
Niveles de abstracción: físico o interno

- Es el nivel más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. En otras palabras, es el nivel más bajo de abstracción.
- Describe la estructura física de la BD mediante un esquema interno.
- Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etc.



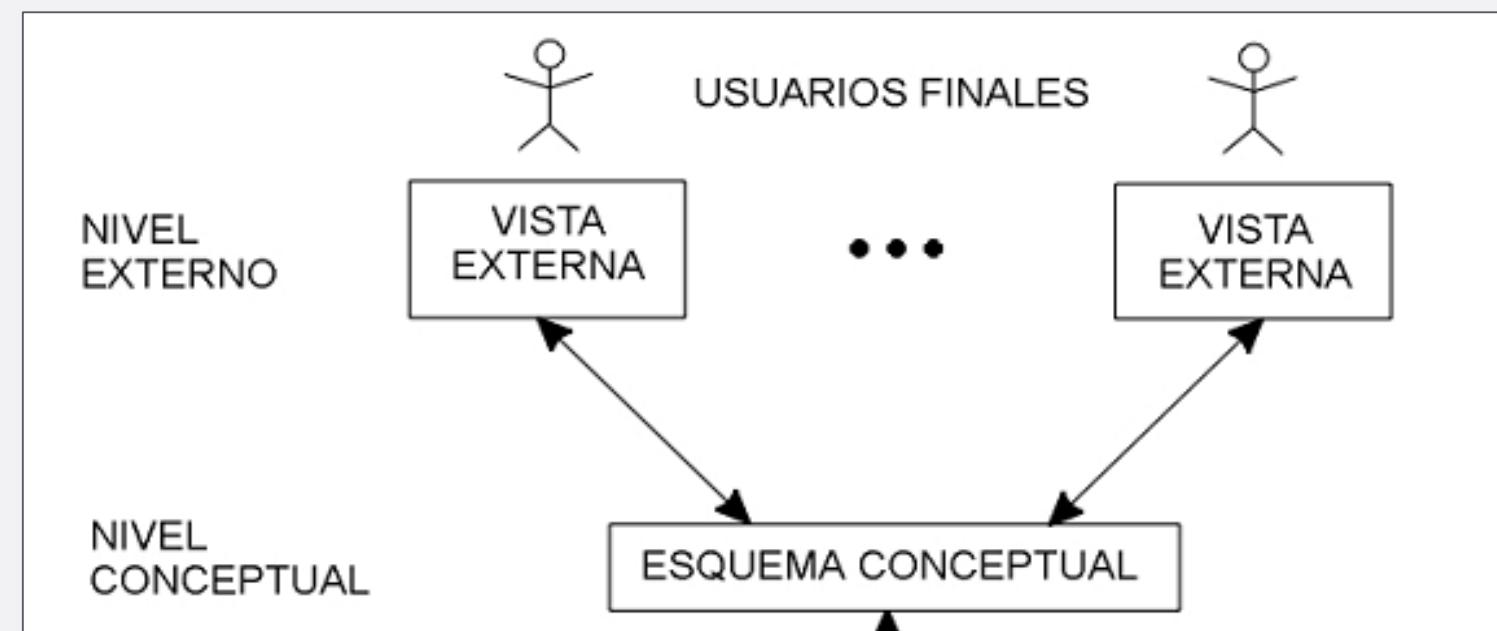
Niveles de abstracción: conceptual o lógico

- Describe la estructura de los datos de toda la BD para un grupo de usuarios mediante un esquema conceptual.
- Oculta los detalles de las estructuras físicas de almacenamiento y se concentra en describir las entidades, atributos, relaciones, operaciones de los usuarios y restricciones.
- Representa la información contenida en la BD.



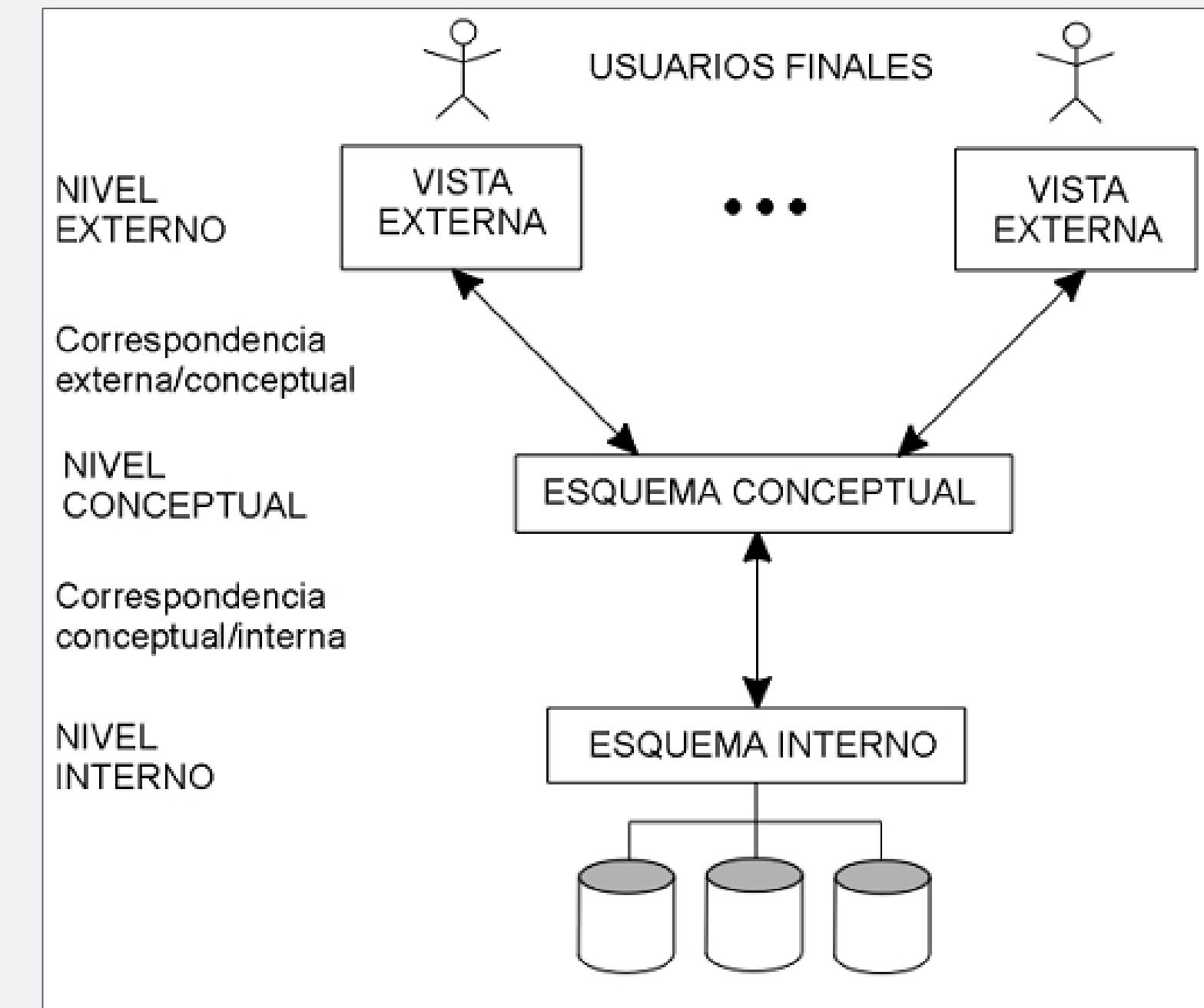
Niveles de abstracción: de vista o externo

- Es el más cercano a los usuarios.
- Es donde se describen varios esquemas externos o vistas de usuarios.
- Cada esquema describe la parte de la BD que interesa a un grupo de usuarios, ocultando el resto de la BD que no le interesa.
- En este nivel se representa la visión individual de un usuario o de un grupo de usuarios.
- La información se manipula sin saber cómo está almacenada internamente (nivel interno) ni su organización (nivel conceptual).



Niveles de abstracción: arquitectura ANSI

- La arquitectura describe los datos a tres niveles de abstracción.
- Los únicos datos que existen están a nivel físico almacenados en discos u otros dispositivos.



SGBD basados en arquitectura ANSI-SPARC

- Permiten que cada grupo de usuarios haga referencia a su propio esquema externo.
- Debe de transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos de esquema conceptual.
- Debe de transformar una petición conceptual a una petición expresada en términos de esquema interno, la que finalmente se procesará sobre la BD almacenada.
- Para una BD específica solo hay un esquema interno y uno conceptual, pero puede haber varios esquemas externos definidos para uno o para varios usuarios.

Proceso de correspondencia o transformación

Es el proceso de transformar peticiones y resultados de un nivel a otro nivel.

El SGBD es capaz de interpretar una solicitud de datos y realiza los siguientes pasos:

- El usuario solicita unos datos y crea una consulta.
- El SGBD verifica y acepta el esquema externo para ese usuario.
- Transforma la solicitud al esquema conceptual.
- Verifica y acepta el esquema conceptual.
- Transforma la solicitud al esquema físico o interno.
- Selecciona la o las tablas implicadas en la consulta y ejecuta la consulta.
- Transforma del esquema interno al conceptual y del conceptual al externo.
- Finalmente, el usuario ve los datos solicitados.

Independencia de datos

Con la arquitectura a tres niveles se introduce el concepto de independencia de datos, se definen dos tipos de independencia:

1. **Independencia lógica:** capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se podrá modificar el esquema conceptual para ampliar la BD o para reducirla, por ejemplo, si se elimina una entidad; los esquemas externos que no se refieran a ella no se verán afectados.
2. **Independencia física:** capacidad de modificar el esquema interno sin tener que alterar ni el esquema conceptual ni los externos. Por ejemplo, se pueden reorganizar los archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización, o se pueden añadir nuevos archivos de datos porque los que había se han llenado. La independencia física es más fácil de conseguir que la lógica, pues se refiere a la separación entre las aplicaciones y las estructuras físicas de almacenamiento.

SGBD: componentes y lenguajes

Componentes: paquetes de software que proporcionan una serie de servicios para almacenar y explotar los datos de forma eficiente.

Componentes principales → Lenguajes de los SGBD: permiten al DBA especificar los datos que componen la BD, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios.

Los lenguajes del SGBD se clasifican en:

- Lenguaje de definición de datos (LDD o DDL).
- Lenguaje de manipulación de datos (LMD o DML).

SGBD: componentes y lenguajes

Lenguaje de definición de datos (LDD o DDL): se utiliza para especificar el esquema de la BD, las vistas de los usuarios y las estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la BD.

Lenguaje de manipulación de datos (LMD o DML): para leer y actualizar los datos de la BD. Es el utilizado por los usuarios para consultar, insertar, modificar y eliminar.

Los hay:

Procedurales

En los que el usuario será normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan. Las sentencias de un LMD procedural están embebidas en un lenguaje de alto nivel llamado ANFITRIÓN. Las BD jerárquicas y en red utilizan estos LMD procedurales.

No procedurales

Son los lenguajes declarativos. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde una terminal, también pueden ir embebidas en un lenguaje de programación de alto nivel. Estos lenguajes permiten especificar los datos a obtener en una consulta o los datos a modificar mediante sentencias sencillas. Las BD relacionales utilizan lenguajes no procedurales como SQL (Structured Quero Language) o QBE (Query By Example).

SGBD: diccionario de datos

Diccionario de datos

- Lugar donde se deposita información acerca de todos los datos que forman la BD.
- Es una guía en la que se describe la BD y los objetos que la forman.
- Contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización.
- Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información.

SGBD: diccionario de datos

Un diccionario de datos debe cumplir las siguientes características:

- Soportar las descripciones de los modelos conceptual, lógico, interno y externo de la BD.
- Estar integrado dentro del SGBD.
- Apoyar la transferencia eficiente de información al SGDB.
- La conexión entre los modelos interno y externo debe ser realizada en tiempo de ejecución.
- Comenzar con la reorganización de versiones de producción de la BD.
- Reflejar los cambios en la descripción de la BD. Cualquier cambio a la descripción de programas ha de ser reflejado automáticamente en la librería de descripción de programas con la ayuda del diccionario de datos.
- Estar almacenado en un medio de almacenamiento con acceso directo para la fácil recuperación de información.

SGBD: seguridad e integridad de datos

Para garantizar la seguridad e integridad de los datos un SGBD debe:

- Garantizar la protección de los datos contra accesos no autorizados.
- Implantar restricciones de integridad en la BD para protegerla contra daños accidentales o no.
- Los valores de los datos que se almacenan deben satisfacer ciertos tipos de restricciones de consistencia y reglas de integridad, que especificará DBA.
- El SGBD puede determinar si se produce una violación a las restricciones.
- Proporcionar herramientas y mecanismos para la planificación y realización de copias de seguridad y restauración.
- Ser capaz de recuperar la BD llevándola a un estado consistente en caso de ocurrir algún suceso que la dañe.
- Soportar múltiples usuarios asegurando el acceso concurrente y ofrecer mecanismos para conservar la consistencia de los datos en el caso de que varios usuarios actualicen la BD de forma concurrente.
- Tener independencia física y lógica.
- Poder extenderse con o sin el agregado de hardware.

SGBD: seguridad e integridad de datos: ACID

¿Qué es una transacción?

Es la **módulo** de trabajo que agrupa una o más operaciones en la BD. Para garantizar la seguridad e integridad de los datos un SGBD debe satisfacer la “prueba ACID” para las transacciones.

- “**A**” representa **atomicidad**: una transacción se ejecuta toda o nada.
- “**C**” representa **consistencia**: todas las BD tienen restricciones de consistencia o expectativas acerca de las interrelaciones entre los elementos de datos. Se espera que las transacciones conserven esas restricciones haciendo que la BD pase de un estado consistente a otro consistente
- “**I**” representa **aislamiento** (*isolated*): cada transacción debe “parecer” que se ejecuta como si ninguna otra transacción se estuviera ejecutando en el mismo momento. Mientras una transacción T1 se está ejecutando ninguna otra transacción T2...Tn ve los cambios de T1 hasta que T1 confirme (COMMIT) los cambios.
- “**D**” representa **durabilidad**: es la condición de que todos los cambios hechos sobre la BD nunca deben perderse después que hayan sido confirmados (COMMIT), no tienen vuelta atrás y perduran aunque el sistema falle.



© Universidad de Palermo
Prohibida la reproducción total o parcial de imágenes y textos.

SGBD: Introducción a las unidades físicas y unidades lógicas

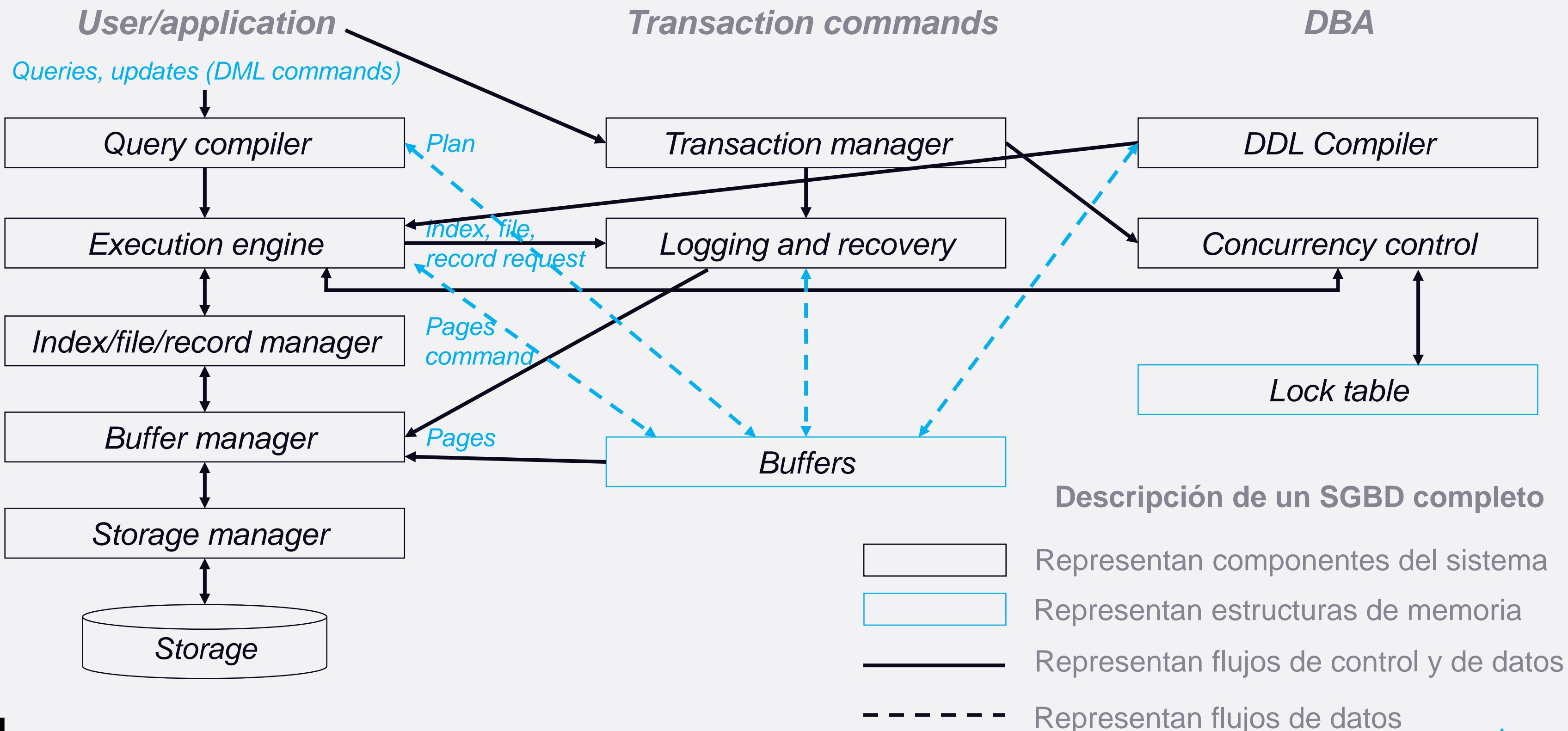
Unidades físicas:

- Almacenan los datos y la parte que describe la estructura de los datos (datos y metadata) (DB file)
- Almacenan la información necesaria para volver la base de datos a un estado consistente ante la ocurrencia de fallas que interrumpan el funcionamiento de la base de datos (LOG file).

Unidades lógicas:

- Porciones de memoria que procesan consultas y operaciones de definición y/o de manipulación para asegurar satisfacer la “**prueba de ACID**” para las transacciones, estas son:
 - Administrador de almacenamiento: Compuesto por el administrador de memoria, el administrador de lockeos (*deadlocks*), el administrador de I/O y el administrador de *fetch* o procesos.
 - Administrador de consultas: Básicamente compuesto por el administrador de parseo, por el administrador de optimización y el motor de ejecución, en algunos casos un compilador puede ser quien ejecute.
 - Administrador transacciones: Compuesto por un gestor de transacciones (*transaction manager*).

Unidades lógicas



SGBD: Análisis del gráfico

Existen dos fuentes de comandos:

- Los de los programas de aplicación y de los usuarios convencionales que consultan o modifican datos.
- Los de los DBAs, responsables de la estructura o esquema de la base de datos.

Análisis de los componentes del sistema

Procesamiento de los comandos del lenguaje de definición de datos (LDD):

- El procesador de comandos DDL (*DDL Compiler*) recibe y analiza los comandos de definición de datos y alteración de esquemas.
- Envía los comandos al motor de ejecución (EE - *Execution Engine*)
- El motor de ejecución pasa los datos a través del **administrador de registros, archivos e índices** para alterar la metadata (información del esquema de la BD)

Procesamiento de consultas:

Se inicia una acción por medio del lenguaje DML o de una consulta (*query*)

- Las sentencias DML se ejecutan mediante dos subsistemas:
 - Responder la consulta (*query*)
 - Procesamiento de transacciones (DML)

SGBD: Análisis del gráfico (continuación)

Procesamiento de consultas:

Responder la consulta:

- El compilador de consultas (*query compile*) analiza la consulta y la optimiza armando el plan de consulta para responderla. Pasa el plan al motor de ejecución (EE -- *Execution Engine*).
- El motor de ejecución emite una secuencia de solicitudes de pequeñas porciones de datos (registros o tuplas de relaciones) a un administrador de recursos que conoce los archivos de datos (*data files*) que contienen las relaciones, el tamaño y formato de los registros en esos archivos, y los archivos de índices, que ayudan a encontrar elementos de los archivos de datos rápidamente.
- La solicitud de datos se pasa al gestor de buffers (*buffer manager*).
 - El *buffer manager* lleva las porciones apropiadas de los datos desde el almacenamiento secundario a los *buffers* de memoria principal (generalmente la página o bloque de disco es la unidad de transferencia entre *buffer* y disco).
 - El *buffer manager* se comunica con el gestor de almacenamiento (*storage manager*) para obtener los datos del disco.
 - El *storage manager* puede involucrar comandos del sistema operativo, pero lo más común es que el SGBD emita comandos directamente al controlador de disco (*disk controller*).

SGBD: Análisis del gráfico (continuación)

El procesador de consultas (*query processor*)

- Es la parte del SGBD que más afecta el rendimiento que ve el usuario.
- Está representado por dos componentes:
 - a. **Compilador de consultas (*query compiler*):**
 - Encargado de traducir la consulta en una forma interna llamada plan de consulta (*query plan*).
 - *Query plan*: secuencia de operaciones a ser ejecutada sobre los datos, generalmente implementada como operaciones de álgebra relacional.
 - Utiliza los metadatos y estadísticas acerca de los datos para decidir qué secuencia de operaciones es la mejor candidata a ser la más rápida.
 - El compilador consta de tres partes:
 1. *Query parser*: Analizador que construye una estructura de árbol del texto de la consulta.
 2. *Query preprocessor*: Preprocesador de la consulta que lleva a cabo la comprobación semántica (por ej.: que existen las relaciones mencionadas en la consulta) y ejecuta algunas transformaciones en el árbol para convertirlo en un árbol de operadores algebraicos que representen el plan de consulta inicial.
 3. *Query optimizer*: Optimizador de consultas que transforma el plan de consulta inicial en la mejor secuencia disponible de operaciones sobre los datos actuales.

SGBD: Análisis del gráfico (continuación)

El procesador de consultas (*query processor*)

b. Motor de ejecución (*execution engine*):

- Encargado de ejecutar cada uno de los pasos seleccionados en el plan de consulta.
- Interactúa con la mayoría de los otros componentes del SGBD directamente o a través de los *buffers*.
- Lleva los datos desde la BD a los *buffers* con el objetivo de manipularlos.
- Interactúa con el planificador (*scheduler*) para evitar el acceso a los datos bloqueados y con el gestor de *logs* para asegurarse de que se registren todos los cambios en la BD.

SGBD: Análisis del gráfico (continuación)

Procesamiento de transacciones

- Los comandos DML se agrupan en transacciones (unidades que deben ser ejecutadas atómicamente y aisladas unas de otras).
- Una acción DML puede ser una transacción en sí misma.
- La ejecución de la transacción debe ser durable/perdurable, el efecto producido por una transacción que se ejecutó en forma completa debe ser preservado siempre, aún ante fallas del sistema.
- El procesador de transacciones se divide en dos partes:
 - Gestor del control de concurrencia (concurrency control): asegura la atomicidad y aislamiento de la transacción.
 - Gestor de recuperación y log (logging and recovery): responsable de la durabilidad de las transacciones.

SGBD: Análisis del gráfico (continuación)

Gestión de *buffers* y almacenamiento

- Su función es controlar la ubicación de los datos en disco y su movimiento entre el disco y la memoria principal.
- Los SGBD generalmente controlan el almacenamiento en disco directamente, bajo determinadas circunstancias y por razones de eficiencia.
- El gestor de almacenamiento lleva registro de la ubicación de los archivos en el disco y obtiene el bloque o bloques de un archivo según la solicitud del gestor de *buffers*.
- El gestor de *buffers* es el responsable del particionamiento de la memoria principal disponible en *buffers*.
- Los *buffers* disponibles son regiones de memoria donde se pueden transferir los bloques.
- Todos los componentes del SGBD que necesiten información del disco van a interactuar con los *buffers* y el gestor de *buffers*, que podrá ser directamente o a través del motor de ejecución.

SGBD: Análisis del gráfico (continuación)

Gestión de *buffers* y almacenamiento

- La clase de información que se puede necesitar incluye:
 - Datos: el contenido de la BD.
 - Metadatos: el esquema de la base de datos que describe la estructura y las restricciones de la BD.
 - Registros de log: información acerca de los cambios recientes a la BD.
 - Estadísticas: información reunida y almacenada por el SGBD acerca de las propiedades como tamaños y valores de las distintas relaciones y otros componentes.
 - Índices: estructuras de datos que respaldan el acceso eficiente a los datos.

SGBD: Análisis del gráfico (continuación)

Procesamiento de transacciones

- Un SGBD ofrece garantía de durabilidad sobre las transacciones.
- El gestor de transacciones (*transaction manager*) acepta comandos de transacción de una aplicación.
- La aplicación informa al gestor de transacciones cuando comienzan y cuando terminan las transacciones, y las expectativas sobre las mismas (alguna transacción podría solicitar la no atomicidad).
- Procesador de transacciones, tareas que realiza:
 - *Logging*
 - Control de concurrencia
 - Resolución de *deadlocks*

SGBD: Análisis del gráfico (continuación)

Procesamiento de transacciones

- Procesador de transacciones, tareas que realiza:
 - *Logging:*
 - Cada cambio en la BD se registra separadamente en disco, asegurando la durabilidad (**ACID**).
 - El gestor de *log* aplica alguna política para asegurar que ante una falla o caída de la BD, un gestor de recuperación va a examinar el registro de cambios y restaurar la BD a un estado consistente.
 - El gestor de *log* primero escribe el *log* en *buffers* y “negocia” con el gestor de *buffers* para asegurarse la escritura en disco a un tiempo apropiado en los archivos donde los datos puedan sobrevivir a una caída.

SGBD: Análisis del gráfico (continuación)

Procesamiento de transacciones

- Procesador de transacciones, tareas que realiza:
 - *Control de concurrencia:*
 - Las transacciones deben parecer que se ejecutan aisladamente a pesar de que hay muchas transacciones ejecutándose simultáneamente (ACID).
 - El planificador o *scheduler* (gestor de control de concurrencia) debe asegurar que las acciones individuales de múltiples transacciones se ejecuten en un orden tal que el efecto neto sea el mismo que si las transacciones se hubiesen ejecutado en forma completa una por vez.
 - El planificador utiliza bloqueos (*locks*) en ciertas piezas de datos para prever que dos transacciones accedan a la misma pieza de datos de forma que interactúen perjudicialmente.
 - Los bloqueos son mantenidos en la tabla de bloqueos (*lock table*) de memoria principal.
 - El planificador afecta la ejecución de consultas y otras operaciones de la BD prohibiéndole al motor de ejecución el acceso a las partes bloqueadas de la BD.

SGBD: Análisis del gráfico (continuación)

Procesamiento de transacciones

- Procesador de transacciones, tareas que realiza:
 - *Resolución de deadlocks:*
 - Las transacciones compiten por recursos a través de los bloqueos que concede el planificador.
 - Puede ocurrir que ninguna transacción pueda proseguir debido a que cada transacción necesita algo de lo que se bloqueó por otra transacción.
 - El procesador de transacciones interviene y cancela una o más transacciones para que otras puedan seguir su ejecución.

Índices

- El índice de una BD es una estructura de datos que mejora la velocidad de las operaciones para permitir un rápido acceso a los registros de una tabla en una BD.
- Los índices se suelen utilizar en aquellos campos sobre los cuales se hacen búsquedas en forma frecuente.
- Los índices pueden ser creados usando una o más columnas, proporcionando la base tanto para búsquedas rápidas al azar como de un ordenado acceso a registros eficiente.
- El espacio en disco requerido para almacenar el índice es típicamente menor que el espacio de almacenamiento de la tabla ya que generalmente contienen solo los campos que componen el índice y excluyen el resto de los campos de la tabla.
- En una BD relacional un índice es una copia de parte de la tabla.
- Algunas BD amplían la potencia del indexado al permitir que los índices sean creados de funciones o expresiones, nativas de la BD o definidas por el usuario, por ejemplo un índice sobre la función UPPER (APELIDO) el cual almacenaría en el índice solamente las versiones mayúsculas del campo apellido.
- Un índice puede ser definido como único o no único. Un índice único actúa como una restricción en la tabla previniendo filas idénticas en el índice.

Índices

Tipos básicos de índices:

- Índices ordenados: basados en una disposición ordenada de los valores.
- Índices asociativos (*hash index*): basados en una distribución uniforme de los valores a través de una serie de cajones (*buckets*). El valor asignado a cada cajón (*bucket*) está determinado por una función de asociación (*hash function*)

Índices

Técnicas de indexación y asociación: No hay una técnica mejor que otra, cada técnica será la más apropiada para una aplicación específica de BD. Las técnicas serán valoradas según los siguientes criterios:

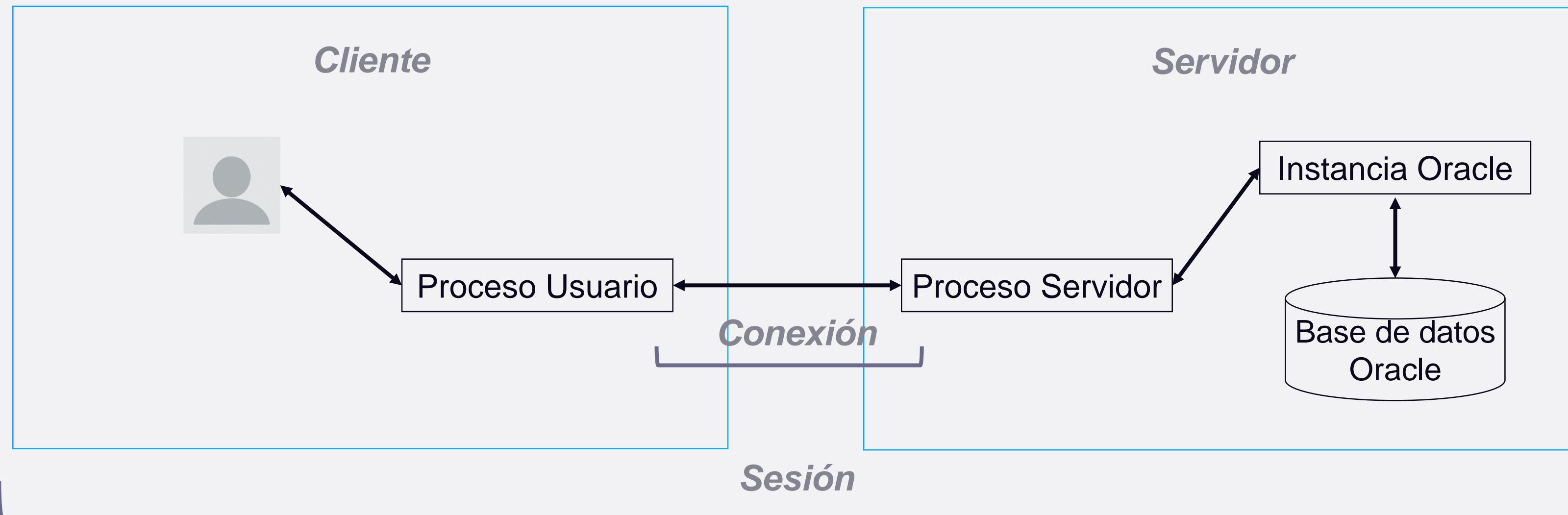
- Tipos de acceso: Tipos de acceso soportados eficazmente, pueden incluir la búsqueda de registros con un valor concreto en un atributo o para un rango específico de valores.
- Tiempo de acceso: Tiempo aplicado en la búsqueda de un determinado elemento de dato, o conjunto de elementos.
- Tiempo de inserción: Tiempo empleado en insertar el nuevo elemento de datos. Este valor incluye el tiempo empleado en la búsqueda del lugar apropiado para donde insertar el nuevo elemento de datos y el tiempo empleado en actualizar la estructura del índice.
- Tiempo de borrado: Tiempo empleado en borrar un elemento de dato. Este valor incluye el tiempo empleado en buscar el elemento a borrar y el tiempo empleado en actualizar la estructura del índice.
- Espacio adicional requerido: El espacio adicional ocupado por la estructura del índice. Muchas veces es razonable sacrificar el espacio para alcanzar un mejor rendimiento en la búsqueda de los datos.



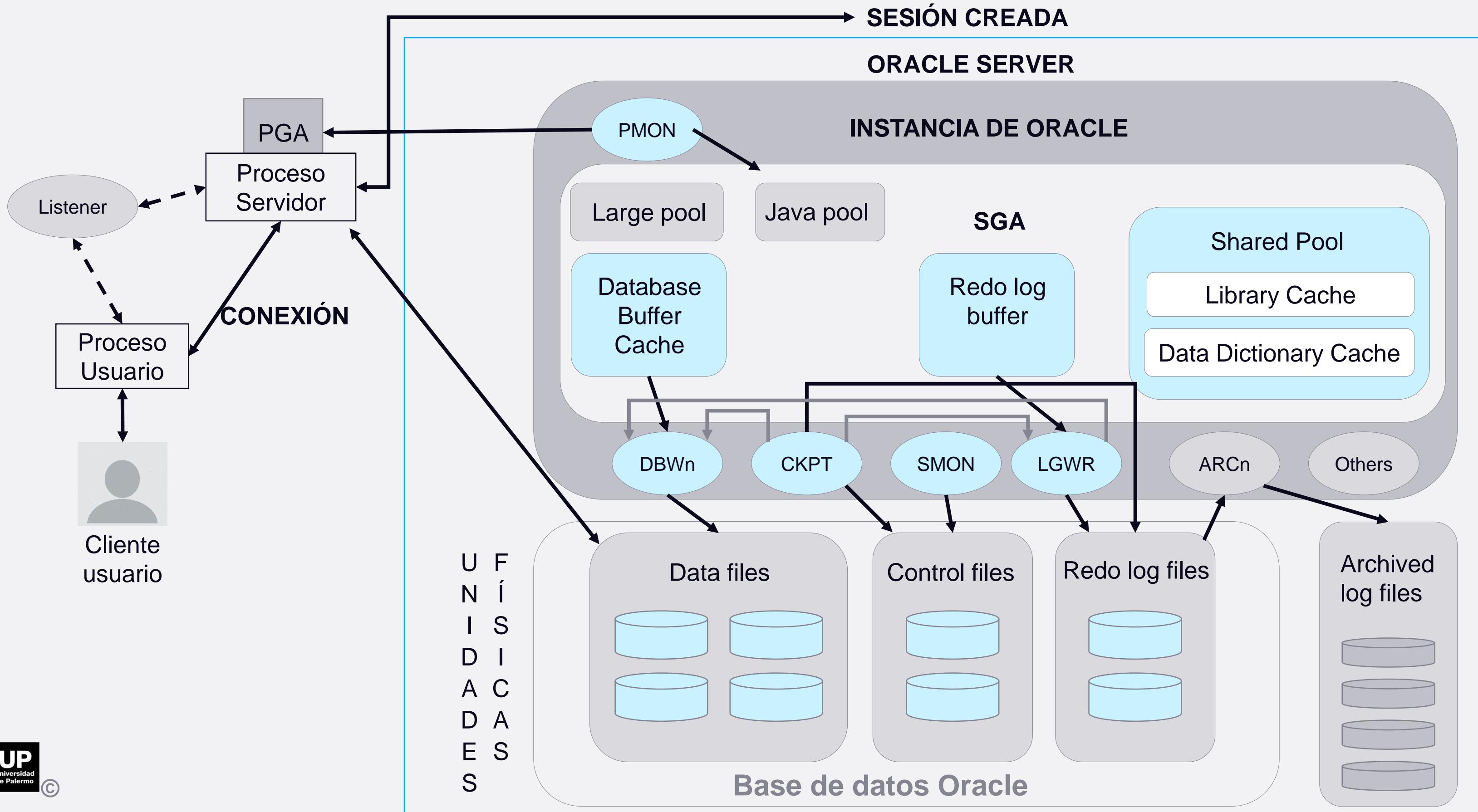
© Universidad de Palermo
Prohibida la reproducción total o parcial de imágenes y textos.

Arquitectura general base de datos Oracle

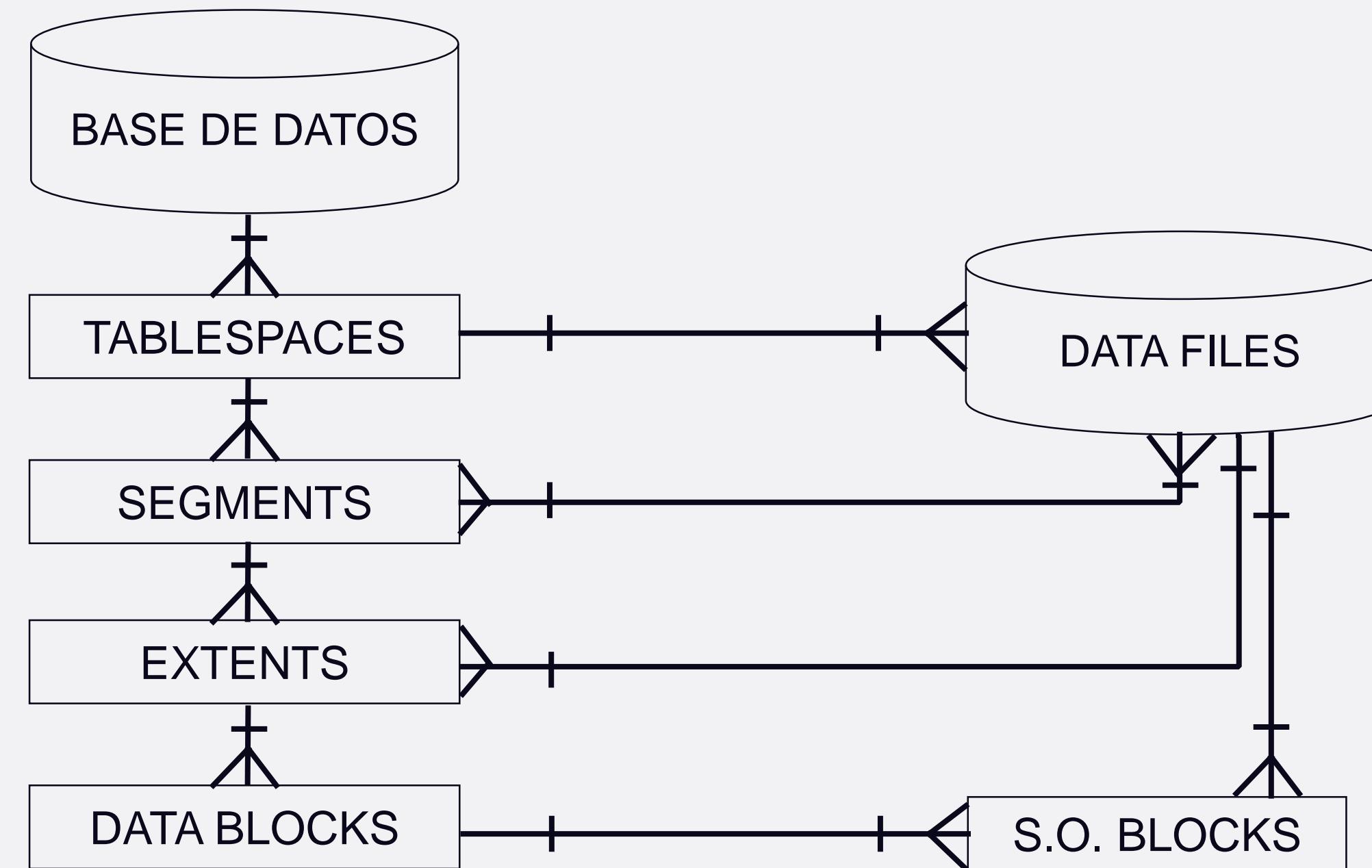
Oracle: Componentes que intervienen en la comunicación de una arquitectura cliente/servidor



Arquitectura de Oracle: componentes principales

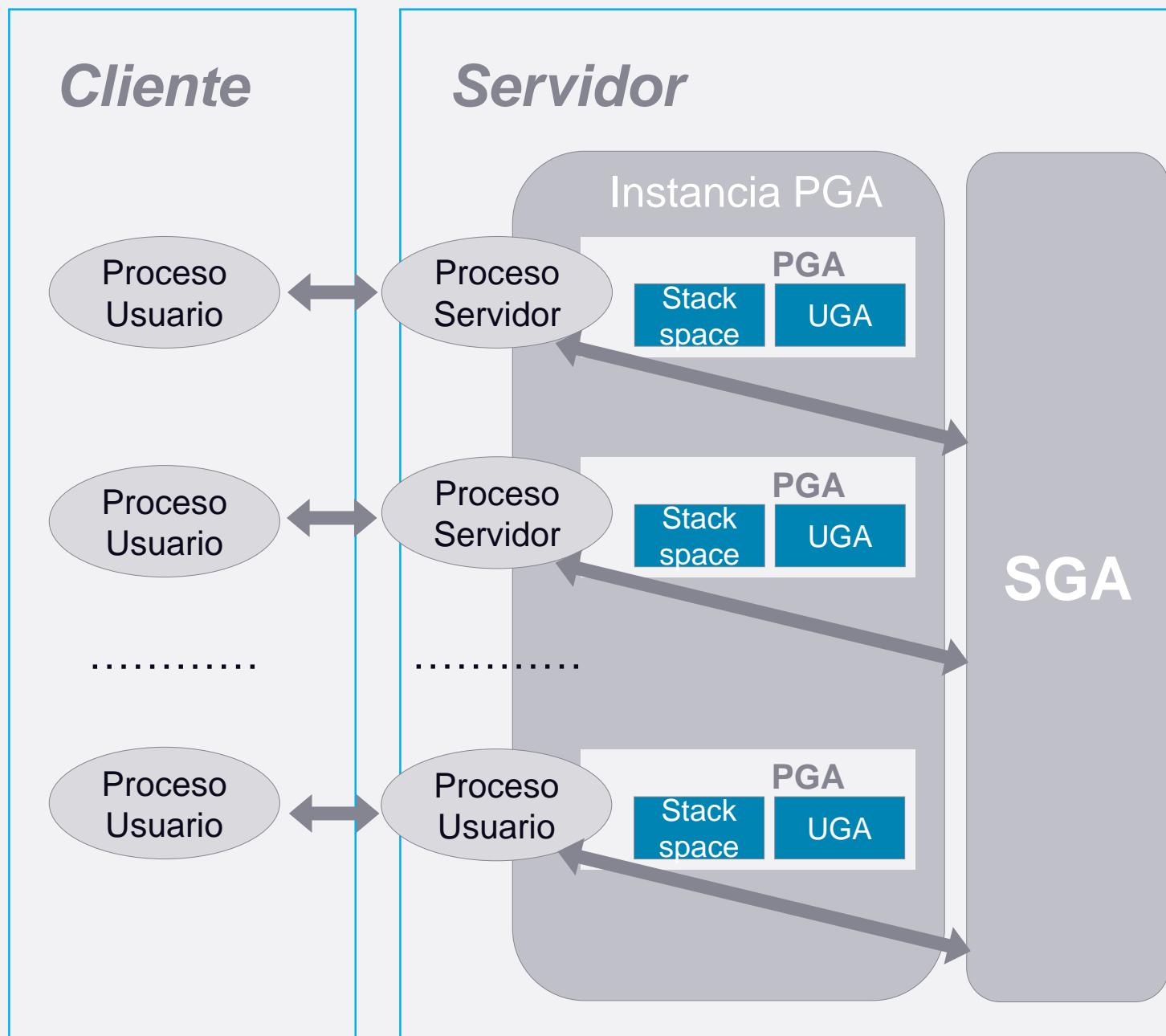


Estructura de almacenamiento: unidades lógicas y su relación con las unidades físicas

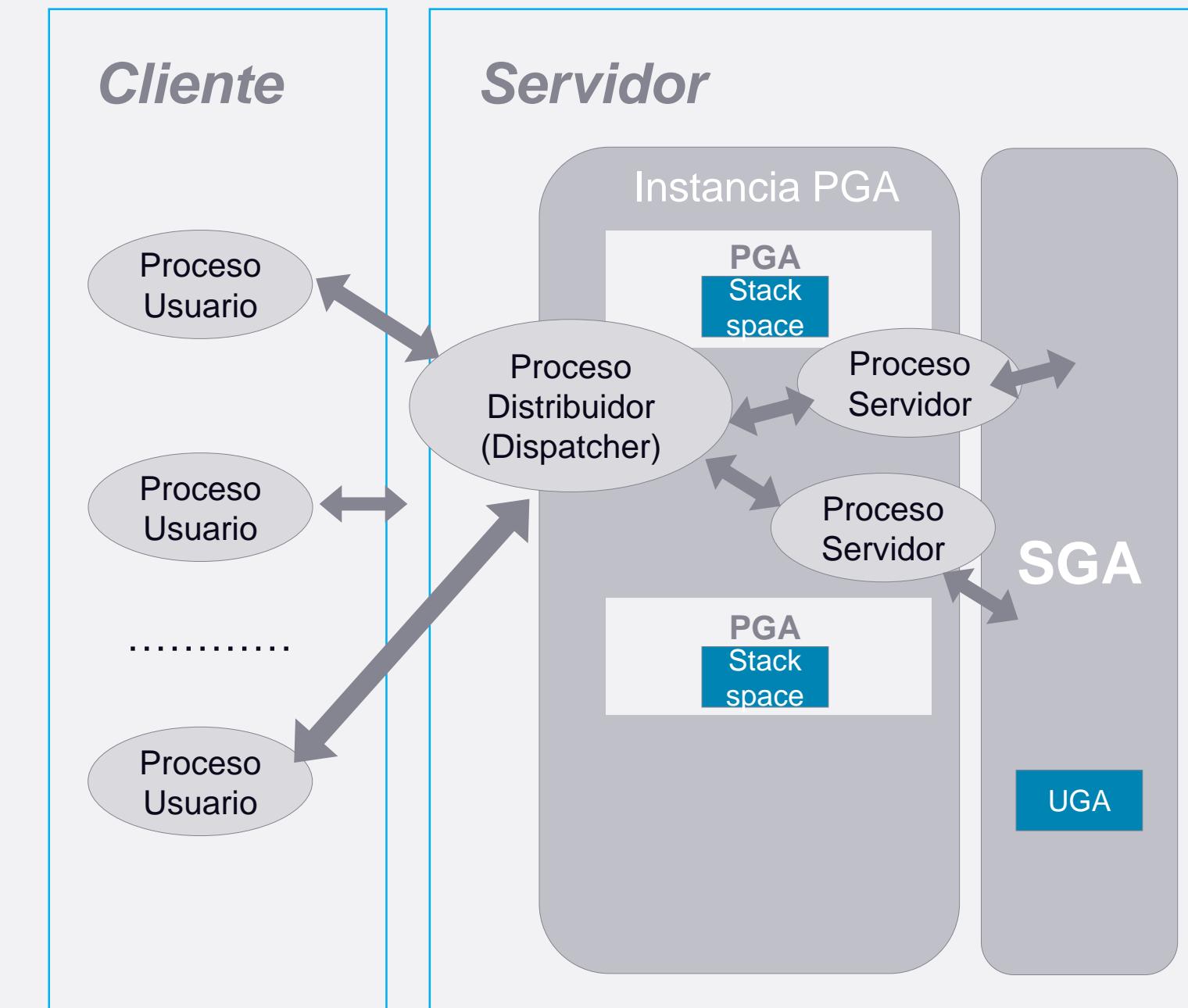


Modos de funcionamiento

Funcionamiento dedicado



Funcionamiento compartido





© Universidad de Palermo
Prohibida la reproducción total o parcial de imágenes y textos.

Concurrencia

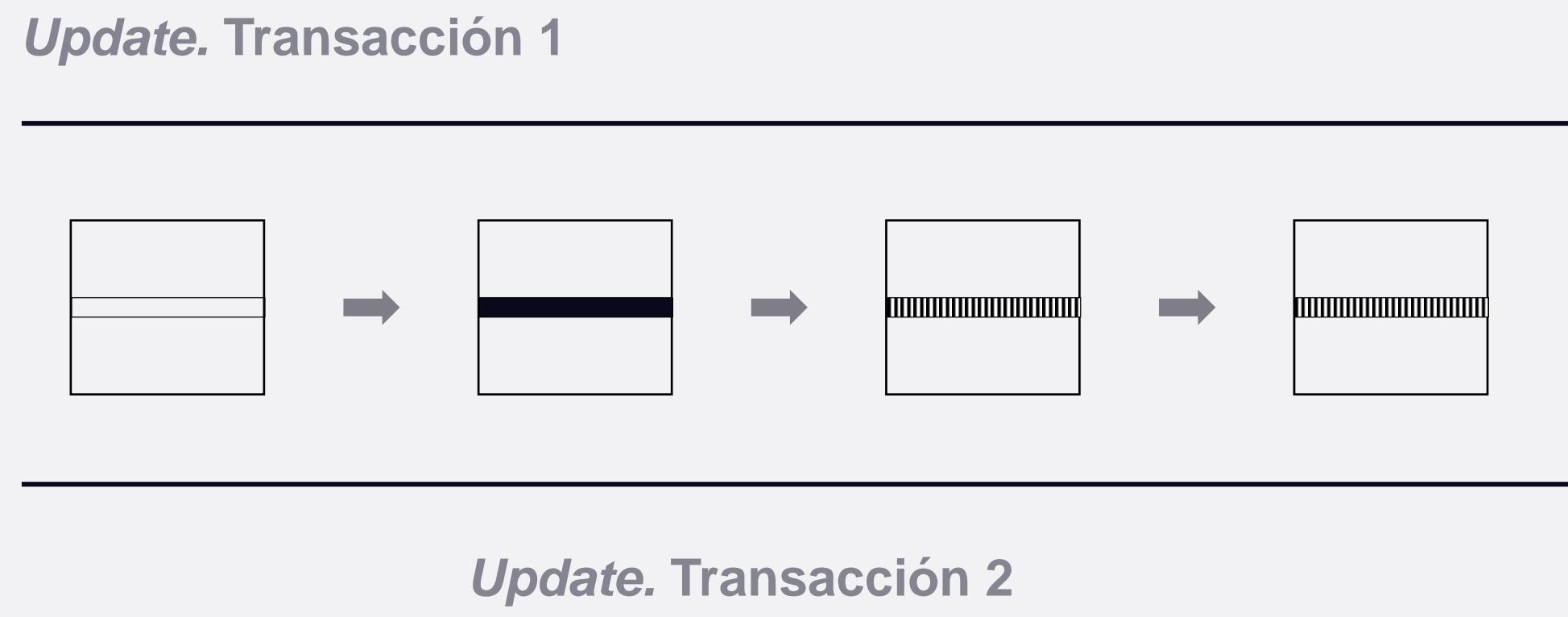
Concurrencia

- ➔ Son procesos que garantizan la consistencia de la BD con el objetivo de mantener un alto número de transacciones activas, buscando un balance con la sobrecarga de controles necesarios para su implementación.
- ➔ Los problemas ocasionados por falta del control de concurrencia suceden cuando múltiples transacciones enviadas por varios usuarios interfieren entre sí de una manera que producen resultados incorrectos.
- ➔ Si no se hace un adecuado control de concurrencia, se pueden perder actualizaciones de datos provocando que los efectos de algunas transacciones no se reflejen en la base de datos. Por tal motivo, pueden ocurrir recuperaciones de datos inconsistentes.
- ➔ Algunas anomalías que pueden suceder se las conoce como:
 - Lost Update
 - Dirty Read (o The Temporary Update)
 - Unrepeatable Read
 - Phantom

Concurrencia (cont.)

→ *Lost Update*

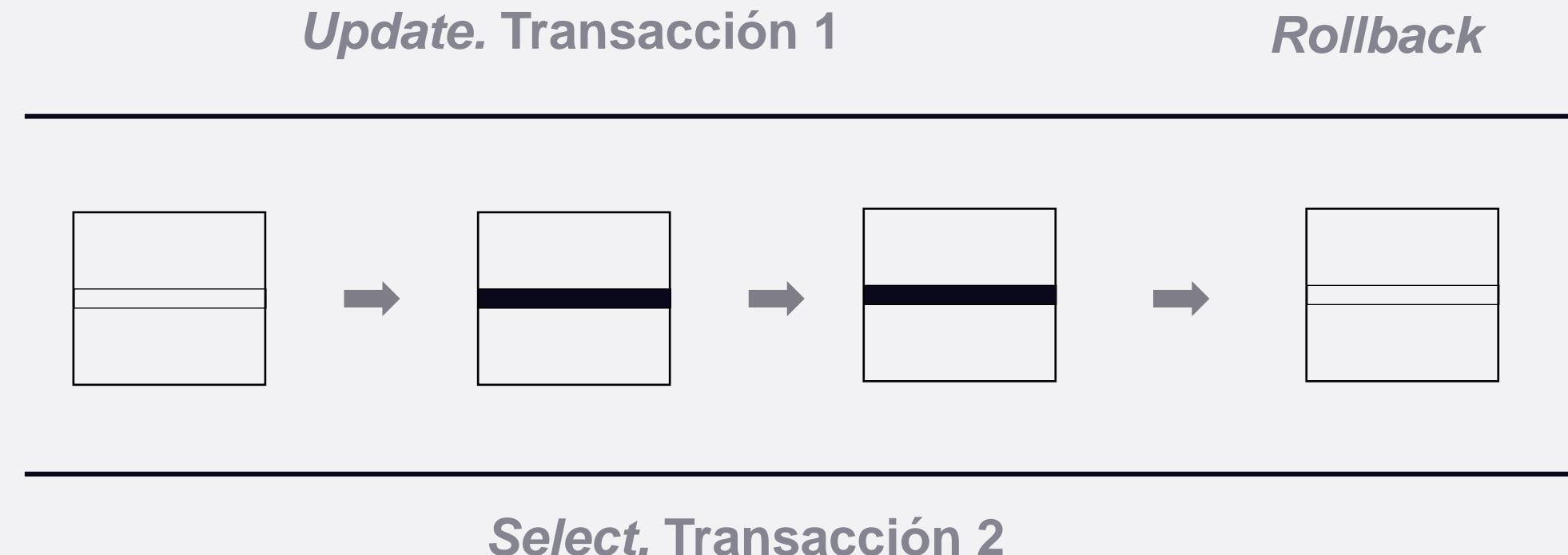
- Ocurre cuando dos transacciones intentan modificar el mismo dato y una de las actualizaciones se pierde.
- La T1 y la T2 leen la misma fila y calculan un nuevo valor.
- Si la T1 actualiza la fila con el nuevo valor y la T2 actualiza la misma fila, se pierden los cambios efectuados por la primera.



Concurrencia (cont.)

→ *Dirty Read*

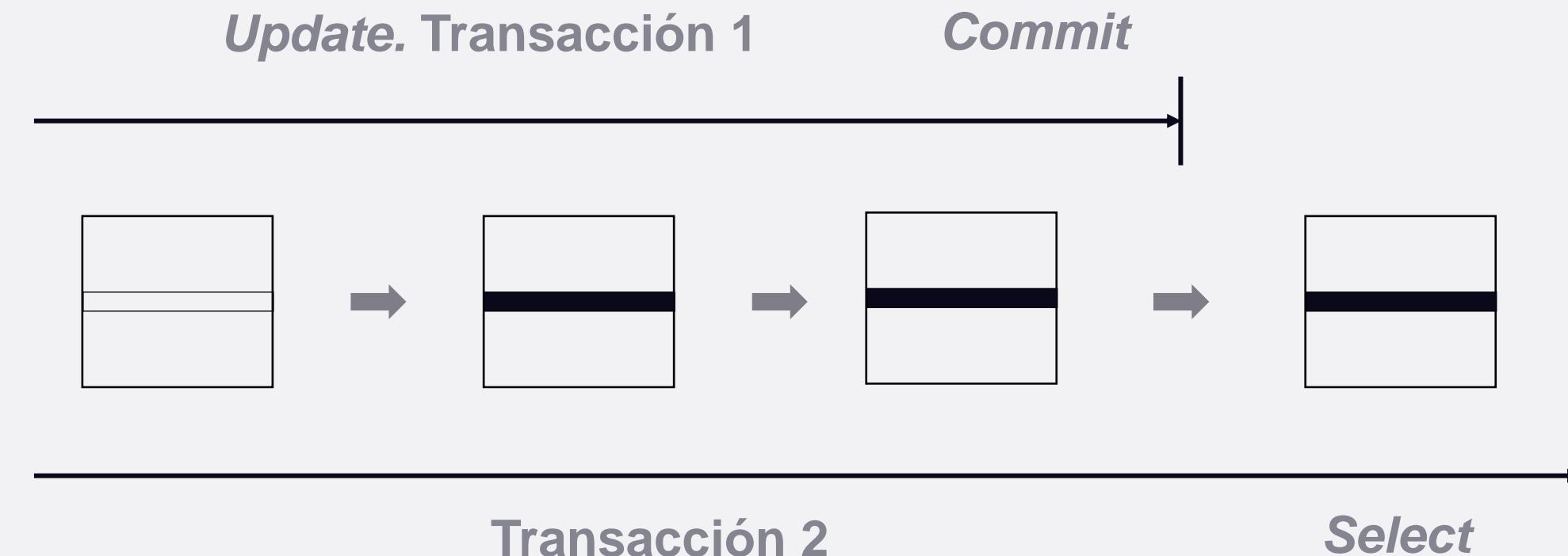
- Ocurre cuando una transacción lee datos no confirmados (*no commit*).
- La T1 cambia una fila.
- La T2 lee la fila antes que la T1 sea confirmada.
- Si la T1 hace un *rollback*, la T2 leyó datos que nunca existieron.



Concurrencia (cont.)

→ *Unrepeatable Read o Nonrepeatable Read*

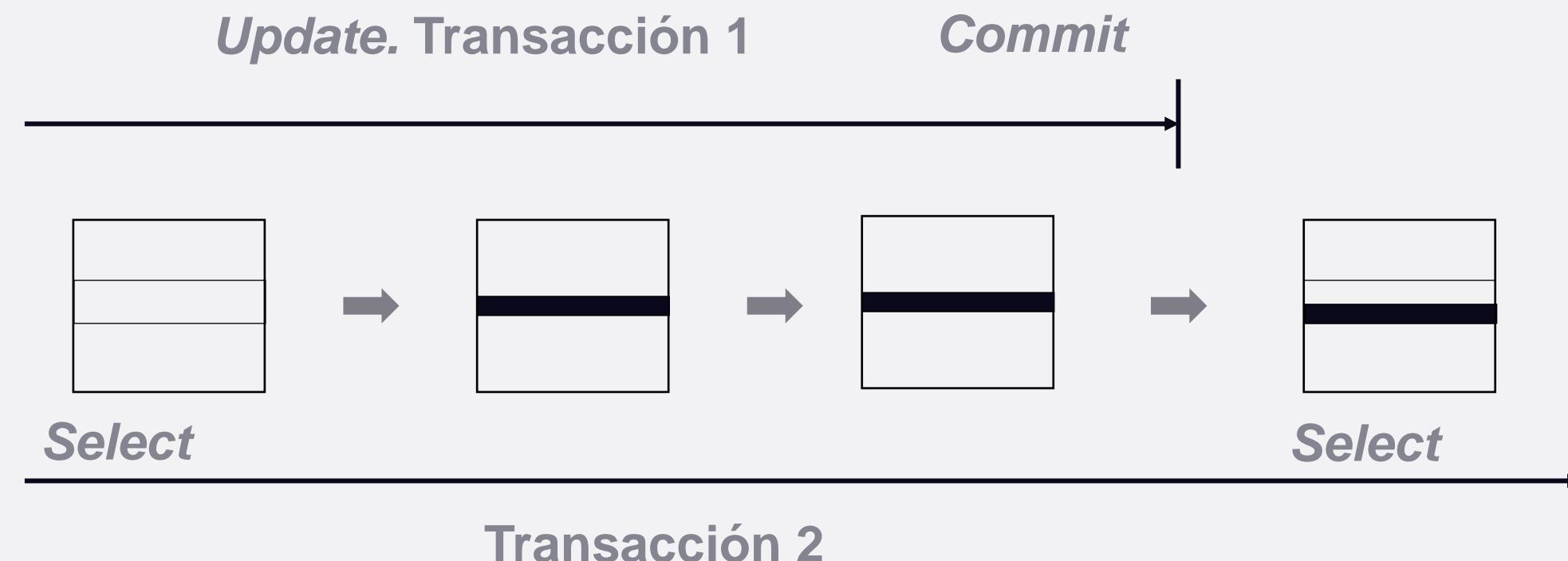
- Ocurre cuando una transacción lee la misma fila dos veces con valores distintos en ambas lecturas.
- La T1 lee una fila.
- La T2 cambia o borra la fila y confirma los cambios.
- Cuando T1 relee la fila se encuentra con datos distintos o directamente no los encuentra.



Concurrencia (cont.)

➔ *Phantom*

- Ocurre cuando una fila que cumple un cierto criterio de búsqueda no es hallada en una primera lectura, pero sí en operaciones siguientes.
- La T1 lee un conjunto de filas.
- La T2 inserta una nueva fila que cumple con el criterio.
- Al re-ejecutarse la consulta se obtiene un conjunto diferente de filas.



Concurrencia (cont.)

- ➔ **Serealización de transacciones:** Define el orden de ejecución de todas las operaciones en su dominio.
- ➔ **Conflicto de operaciones y serialización**
 - La función primaria de un controlador de concurrencia es generar un orden serializado para la ejecución de todas las transacciones.
 - El problema radica en desarrollar algoritmos que garanticen que únicamente se genere un orden serializado
 - Dos operaciones que acceden al mismo dato de la BD se dice que están en CONFLICTO si al menos una de ellas es de escritura.
 - Las operaciones de lectura no tienen conflicto consigo mismas
- ➔ Existen dos tipos de conflictos:
 - *Read-write (o write-read)*
 - *Write-write*

Concurrencia (cont.)

→ Mecanismos de control de concurrencia

- Algoritmos basados en LOCK con accesos mutuamente exclusivos a datos compartidos.
- Algoritmos basados en protocolos, que ordenan la ejecución de las transacciones de acuerdo a un conjunto de reglas.

→ Estos métodos se aplican según dos puntos de vista:

- Pesimista: Sincronizan la ejecución concurrente de las transacciones en la etapa inicial del ciclo de ejecución.
- Optimista: Retrasan la sincronización de las transacciones hasta su finalización

Mecanismos de control de concurrencia

Técnicas pesimistas:

- Algoritmos basados en candados

Existen candados de lectura (rl), de escritura (wl) y existe un administrador de candados (LM) que recibe del administrador de transacciones el elemento de dato y la operación a realizar.

El LM verifica la compatibilidad de los candados y si no hay conflicto fija un nuevo candado, de lo contrario retrasa la transacción.

Una vez terminada la transacción se eliminan todos los candados.

Las técnicas de bloqueo o candado pueden producir DEADLOCK, donde dos o más transacciones están esperando cada una de ellas que la otra libere algún objeto antes de seguir.

- Algoritmos basados en *slice* de tiempo

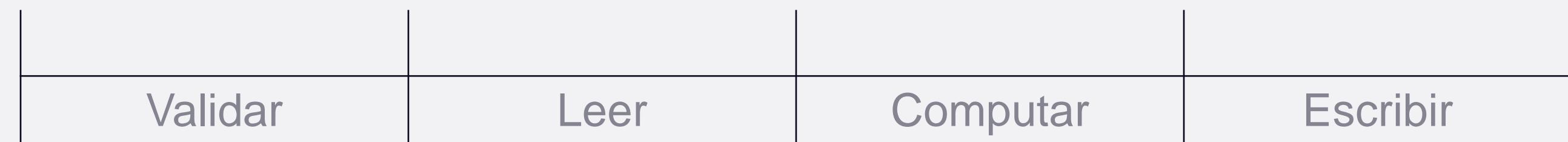
Seleccionan un orden de serialización a priori y ejecutan las transacciones de acuerdo a ellas.

Para establecer el orden, el administrador de transacciones le asigna a cada transacción T_i un *slice* de tiempo único $TS(T_i)$.

Un *slice* de tiempo es un identificador simple que sirve para identificar cada transacción de manera única.

Mecanismos de control de concurrencia

Transacción pesimista:



Concurrencia (cont.)

→ Técnicas optimistas

- Las transacciones acceden libremente a los elementos y antes de finalizar se determina si ha habido interferencias.
- Las operaciones de lectura, computo y escritura de cada transacción se procesan libremente sin actualizar la BD de corriente.
- Cada transacción inicialmente hace sus cambios en copias locales de los datos.
- La fase de validación consiste en verificar si esas actualizaciones conservan la consistencia de la BD, si la respuesta es positiva, los cambios se hacen globales (escritos en la BD corriente). De otra manera, la transacción es abortada y tiene que reiniciarse.

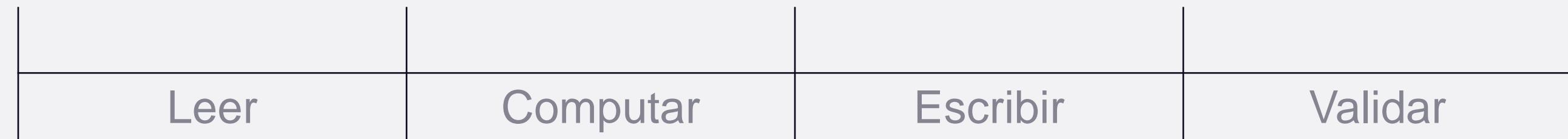
Concurrencia (cont.)

→ Técnicas optimistas

- Este tipo de técnicas considera que las transacciones tienen 3 fases:
 - **Lectura:** Las transacciones realizan operaciones sobre copias privadas de los objetos (accesibles solo para la transacción).
 - **Validación:** Se comprueba si el conjunto de objetos modificados por una transacción se solapa con el conjunto de objetos modificados por alguna otra transacción que haya hecho la validación durante la fase de lectura de dicha transacción.
 - **Grabación:** en el caso de no detectar interferencias se graban las modificaciones, convirtiendo las versiones privadas de los objetos en versiones actuales
- En todo sistema de concurrencia se tiene que tener cuidado con los *deadlocks*.

Concurrencia (cont.)

Transacción optimista:



Integridad

→ Integridad

- Tiene como función proteger la BD contra operaciones que introduzcan inconsistencia en los datos, en el sentido de corrección, validez o precisión de los datos.
- El subsistema de **integridad** de un SGBD **debe**, por tanto, detectar y corregir, en la medida de lo posible, las operaciones incorrectas.
- En la práctica casi toda la verificación de integridad se realiza mediante código de procedimientos escritos por los usuarios.

Integridad (cont.)

→ Operaciones semánticamente inconsistentes:

- Transgreden las restricciones que ha definido el administrador al diseñar la BD:
- Restricciones estáticas:
 - Restricciones sobre los dominios, por ejemplo: el dominio edad debe estar comprendido entre 18 y 65 años.
 - Restricciones sobre atributos, por ejemplo: la edad de los empleados ingenieros debe ser mayor a 21 años.
- Restricciones dinámicas:
 - El sueldo de un empleado no puede disminuir.

Integridad (cont.)

→ Otra forma de clasificar las restricciones

- **Simples:** Si se aplican a una ocurrencia de un atributo con independencia de los demás, por ejemplo: el sueldo de un empleado tiene que ser mayor que 6000.
- **Compuestas:** Si implican más de una ocurrencia, como es el caso de las restricciones de comparación, por ejemplo: el sueldo de un empleado debe ser menor que el de su jefe.
- **De globalidad,** por ejemplo el sueldo medio de los empleados de un determinado departamento debe ser menor de 25000.

Las reglas de **integridad** deben almacenarse en el diccionario de datos, como parte integrante de los datos (control centralizado de la semántica), de modo que no han de incluirse en los Programas.



© Universidad de Palermo
Prohibida la reproducción total o parcial de imágenes y textos.

Definición, metodologías y mejores prácticas para la optimización

Optimización

- ➔ La optimización de una base de datos tiene como objetivo reducir el tiempo de respuesta del sistema, maximizando la velocidad y eficiencia con la que recuperan los datos, mejorando la experiencia del usuario final.
- ➔ Para lograr una buena optimización es necesario contar con un diseño físico de base de datos eficiente.
- ➔ Para ser más eficiente en el diseño físico de bases de datos es necesario llevar a cabo el proceso del diseño físico de bases de datos incluyendo entradas, salidas y objetivos, junto con dos conceptos críticos del entorno: estructuras de archivos y optimización de consultas.

Diseño físico de las bases de datos: Generalidades

- Las decisiones en la fase del diseño físico de bases de datos involucran el nivel de almacenamiento de una base de datos (esquema interno).
- El objetivo del diseño físico de bases de datos es minimizar los tiempos de respuesta para acceder y modificar una base de datos

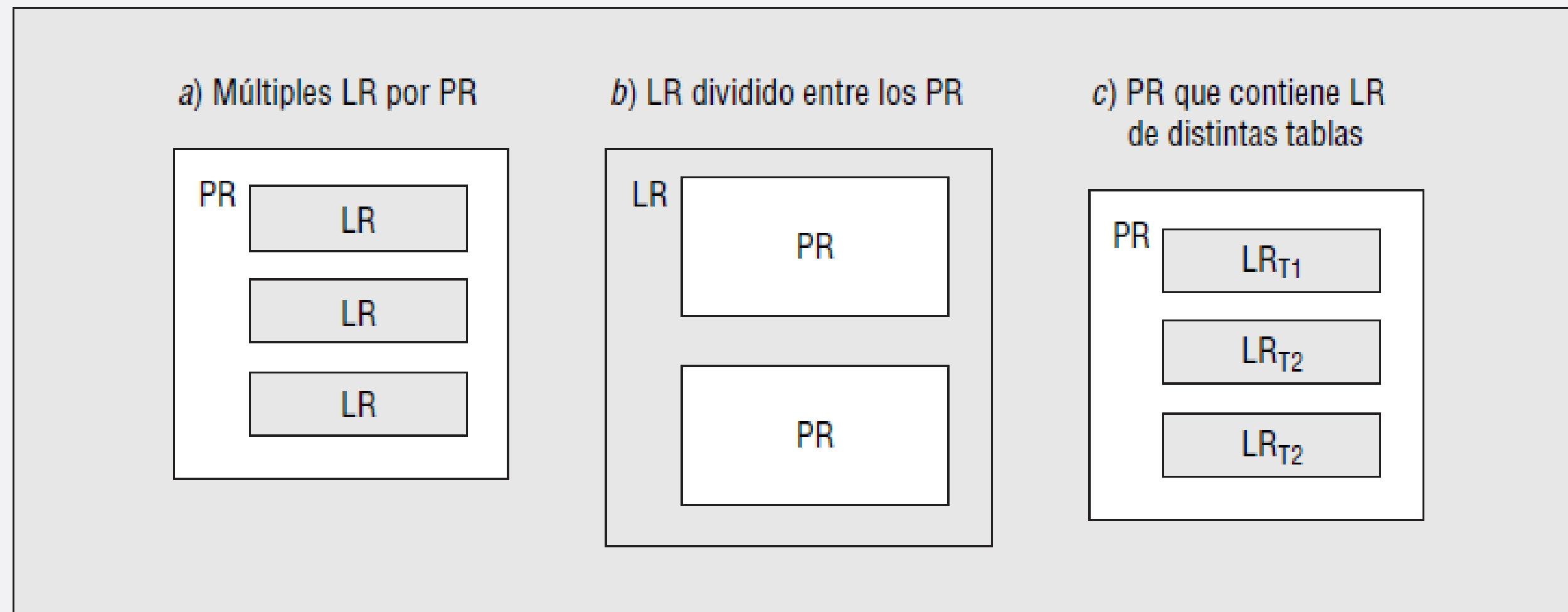
Diseño físico de las bases de datos: Generalidades (cont.)

→ El nivel de almacenamiento:

- Es el más cercano al *hardware* y al sistema operativo.
- La base de datos está formada de registros físicos (también conocidos como bloques o páginas) organizados en archivos.
- Un registro físico es un conjunto de bytes que se transfieren entre el almacenamiento volátil de la memoria principal y el almacenamiento fijo de un disco.
- A la memoria principal se le considera como almacenamiento volátil porque los contenidos de la memoria principal se pueden perder si ocurre alguna falla.
- Un archivo es un conjunto de registros físicos organizados para conseguir un acceso eficiente.
- Un registro físico puede contener varios registros lógicos.
- El tamaño de un registro físico es una potencia del número dos, por ejemplo 1 024 (2 potencia 10).
- Un registro lógico muy grande se puede dividir entre varios registros físicos.
- Los registros lógicos de más de una tabla se pueden almacenar en el mismo registro físico.
- El DBMS y el sistema operativo trabajan de manera conjunta para satisfacer las solicitudes de registros lógicos hechas por las aplicaciones.

Diseño físico de las bases de datos: Generalidades (cont.)

- Relaciones entre los registros lógicos (RL) (filas de una tabla) y registros físicos (PR) almacenados en un archivo.



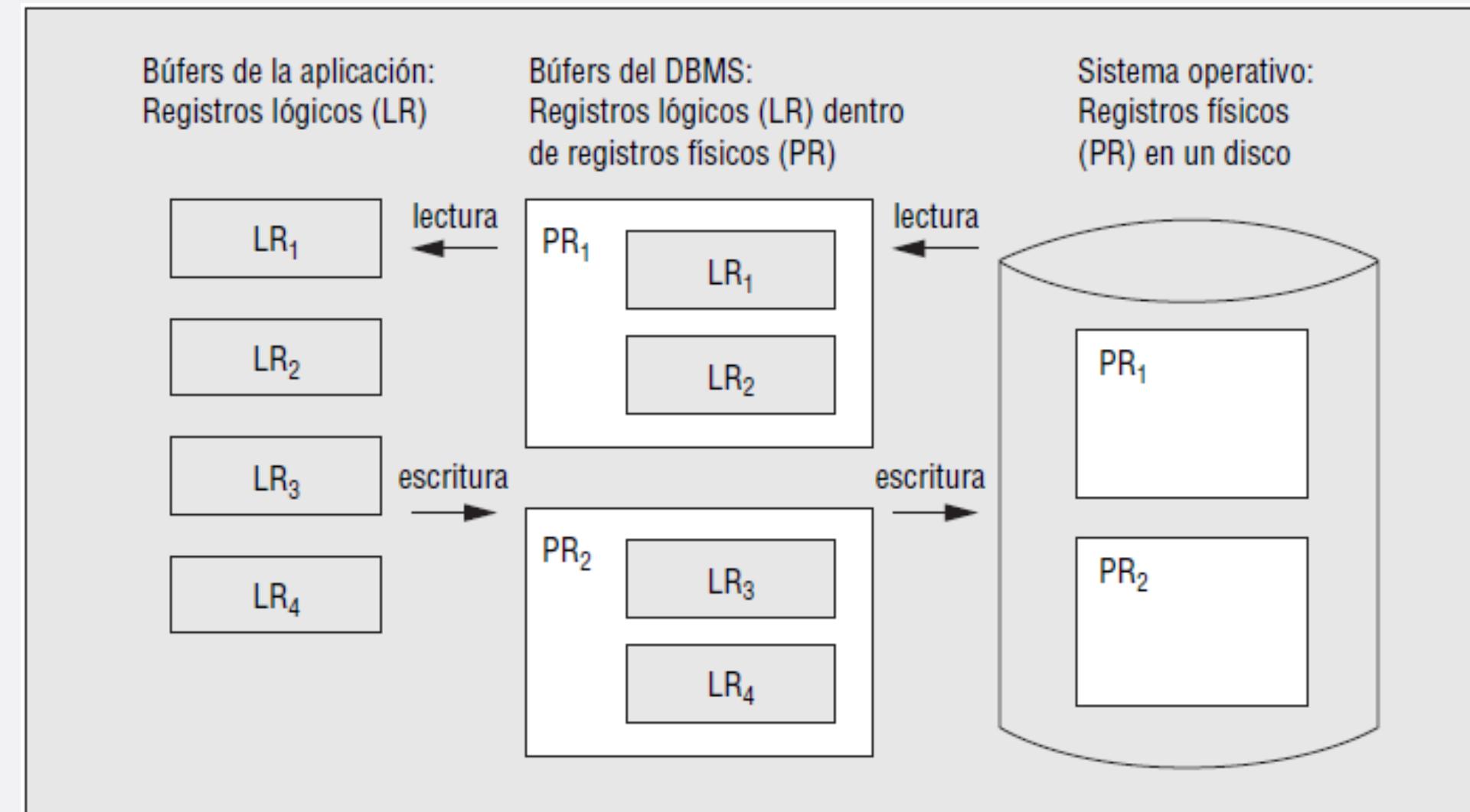
Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.1, pp. 250), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- El DBMS y la aplicación tienen áreas de memoria separadas (*buffers*).
- Cuando una aplicación hace una solicitud para un registro lógico, el DBMS ubica al registro físico que lo contiene.
- Para las operaciones de lectura, el sistema operativo transfiere el registro físico del disco al área de memoria del DBMS.
- El DBMS transfiere el registro lógico al *buffer* de la aplicación.
- Para las operaciones de escritura, el proceso de transferencia se invierte.
- Un requerimiento de registro lógico no siempre resulta en una transferencia de registro físico debido al proceso denominado *buffering*, así el DBMS se anticipa a las solicitudes de las aplicaciones para que los registros físicos correspondientes residan ya en sus *buffers*.
- Una dificultad significativa acerca de la predicción del desempeño de la base de datos es conocer cuándo una solicitud de registro lógico conduce a una transferencia de registro físico.
- Si varias aplicaciones están accediendo a los mismos registros lógicos, los registros físicos correspondientes pueden residir en los *buffers* del DBMS, resultando en una posible dificultad durante el diseño físico de la base de datos.

Diseño físico de las bases de datos: Generalidades (cont.)

- Proceso de transferencia de registros físicos (PR) y registros lógicos (LR) entre un disco, *buffers* del DBMS y *buffers* de la aplicación.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.2, pp. 251), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

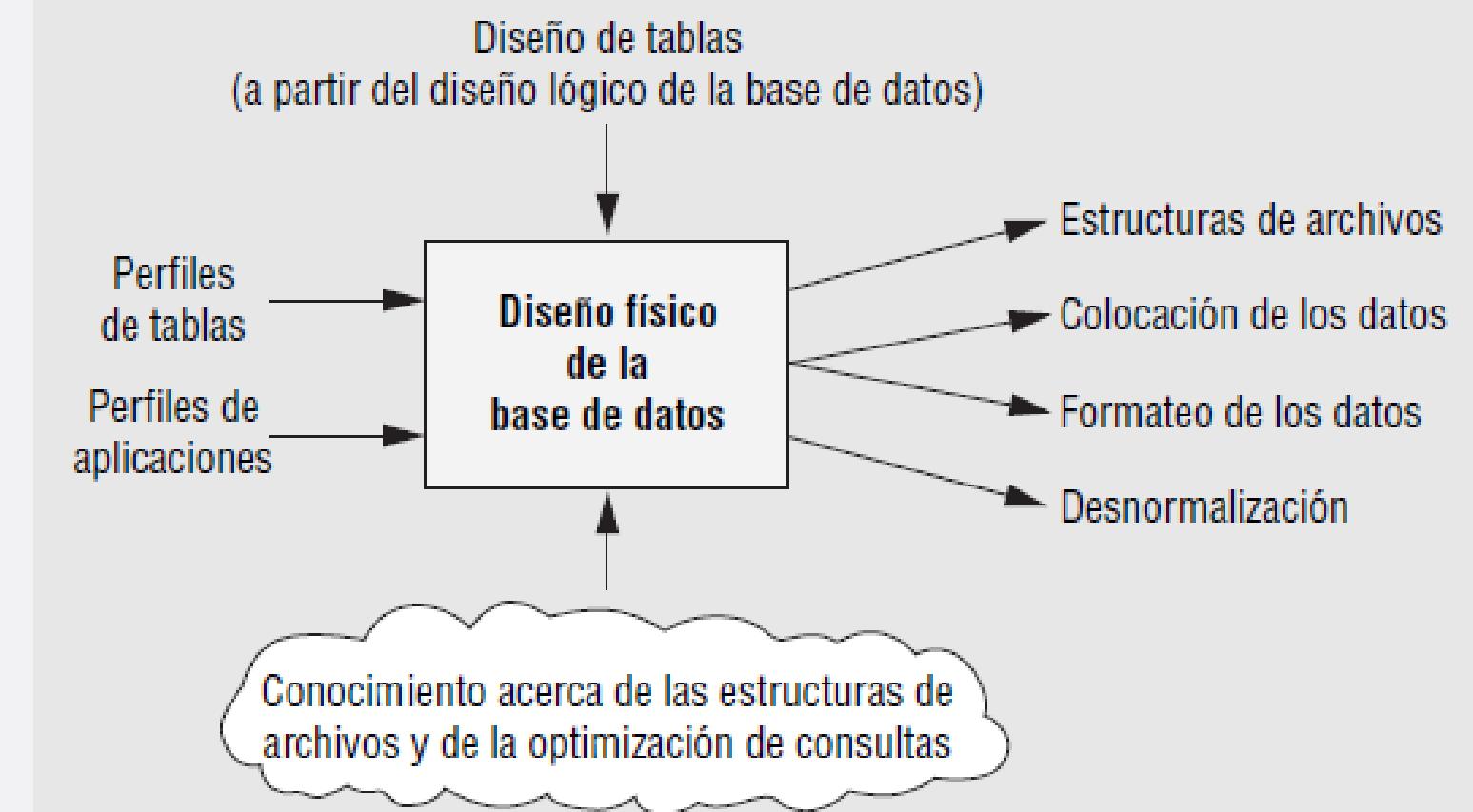
- ➔ El tiempo de respuesta es difícil de estimar de modo directo y como medida sustituta se aplica la minimización de los recursos de cómputo.
- ➔ Los recursos que consume el procesamiento de la base de datos son la transferencia de registros físicos, las operaciones de la unidad central de procesamiento (CPU), la memoria principal y el espacio en disco.
- ➔ La memoria principal y espacio en disco son considerados como restricciones, en lugar de recursos a minimizar.
- ➔ La minimización de la memoria principal y del espacio en disco puede ocasionar tiempos largos de respuesta. El número de accesos a los registros físicos limita el desempeño de la mayoría de las aplicaciones de bases de datos.
- ➔ El acceso a un registro físico (medido en milisegundos) puede ser muchas veces más lento que el acceso a la memoria principal (medido en nanosegundos).
- ➔ La reducción del número de accesos a registros físicos mejorará el tiempo de respuesta.
- ➔ El uso de CPU es otro factor que influyen en los tiempos de respuestas aplicaciones de bases de datos.

Diseño físico de las bases de datos: Generalidades (cont.)

- ➔ Se utiliza un peso cercano a 0 para reflejar que muchas operaciones del CPU se pueden realizar en el tiempo en el que se realiza una transferencia de registro físico.
- ➔ En el diseño físico de bases de datos es muy importante balancear las necesidades de recuperación y actualización de las aplicaciones, siendo usualmente fijas las cantidades de memoria principal y de espacio en disco.
- ➔ La memoria principal y el espacio en disco son restricciones del proceso de diseño físico de bases de datos, por lo que deben considerarse los efectos ocasionados cuando se modifican las cantidades proporcionadas de memoria principal y espacio en disco, un incremento de las cantidades puede mejorar el desempeño.
- ➔ La cantidad de mejora del desempeño puede depender de muchos factores tales como el DBMS, el diseño de las tablas y las aplicaciones que usa la base de datos.
- ➔ El *software* de optimización compleja generalmente es parte del compilador de SQL.

Diseño físico de las bases de datos: Generalidades (cont.)

- El diseño físico de bases de datos está formado por varias entradas y diferentes salidas.
- El punto de partida es el diseño de tablas a partir de la fase del diseño lógico.
- Los perfiles de las tablas y aplicaciones se usan específicamente para el diseño físico de bases de datos.
- Las salidas más importantes son las decisiones de las estructuras de archivos y la colocación de los datos, las decisiones acerca de otras salidas se hacen de forma separada, a pesar de que estén relacionadas



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.3, pp. 252), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

- ➔ El diseño físico de bases de datos es una secuencia de procesos de toma de decisiones.
- ➔ El conocimiento de las estructuras de archivos y la optimización de consultas están dentro del entorno del diseño físico de bases de datos.
- ➔ El diseño físico de las bases de datos se puede dificultar debido al número de decisiones, relaciones entre las decisiones, entradas detalladas, complejidad del entorno e incertidumbre para predecir los accesos a los registros físicos.
- ➔ El diseño físico de las bases de datos requiere de entradas detalladas tanto para los perfiles de las tablas como los perfiles de las aplicaciones.
- ➔ El perfil de una tabla resume una tabla como un todo, las columnas dentro de la tabla y la relación entre las tablas. La mayoría de los DBMS proporcionan programas estadísticos para construir perfiles de tabla de forma automática, se suele establecer la periodicidad de ejecución de esos programas. Para bases de datos muy grandes, los perfiles de las tablas se pueden estimar con ejemplos de la base de datos, para reducir el tiempo insumido por esta tarea.
- ➔ Para los resúmenes de columnas y relaciones, la distribución expresa el número de filas y filas relacionadas para los valores de las columnas. Se suele asumir que los valores de las columnas están distribuidos de modo uniforme, es decir que cada valor tiene un número igual de filas y solo se necesitan los valores mínimo y máximo.

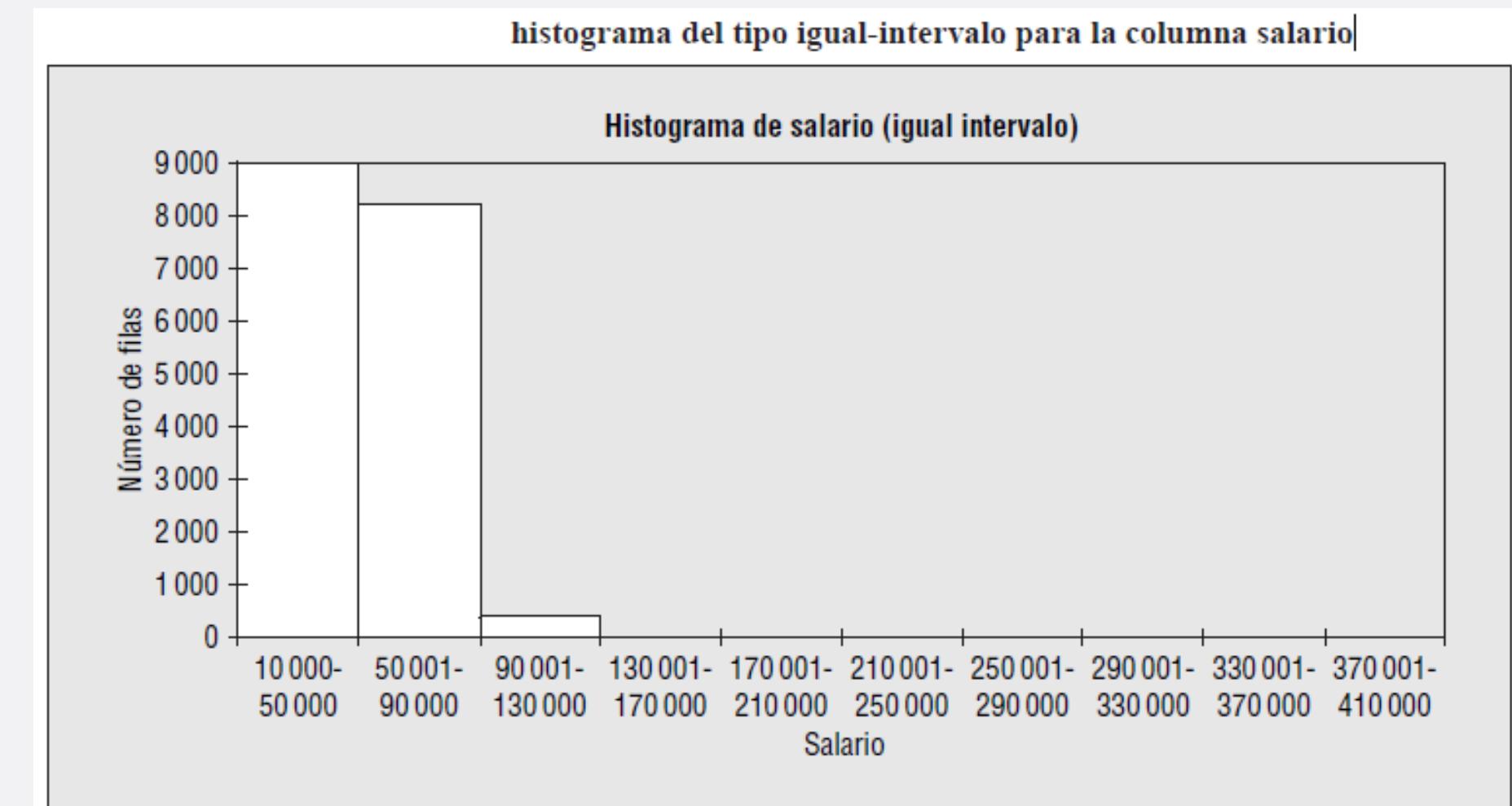
Diseño físico de las bases de datos: Generalidades (cont.)

→ Histogramas:

- Se utilizan para especificar con más detalle la distribución.
- Es un gráfico de dos dimensiones en la que el eje X representa los rangos de las columnas y el eje Y representa el número de filas.

→ Tipos de histogramas:

1. Igual intervalo para los valores de una columna

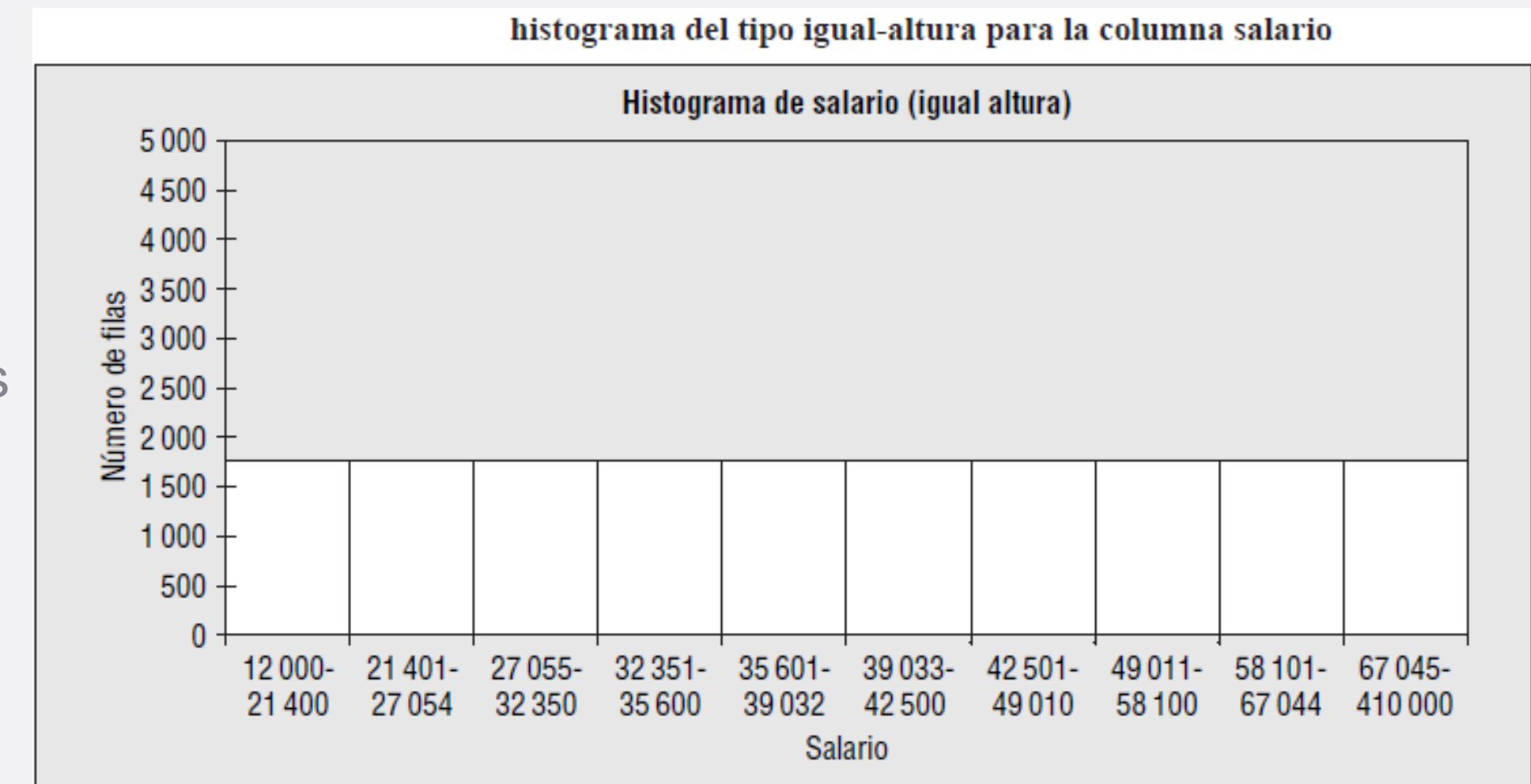


Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.4, pp. 254), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

→ Histogramas:

- Los histogramas tradicionales del mismo intervalo no funcionan bien con datos asimétricos, ya que es necesario un gran número de rangos para controlar los errores estimados.
- Debido a que los datos asimétricos pueden conducir a estimaciones pobres con el uso de los histogramas tradicionales (intervalo igual), la mayoría de los DBMS utilizan histogramas del tipo igual-altura.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.5, pp. 255), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

→ 2. Igual altura para los valores de una columna

Diseño físico de las bases de datos: Generalidades (cont.)

→ Histogramas:

- En los histogramas de igual-altura para los valores de una columna, la anchura de los rangos varía, pero la altura es aproximadamente la misma.
- La mayoría de los DBMS usan histogramas del tipo igual-altura, dado que los errores de estimación máximos y esperados se pueden controlar aumentando el número de rangos.
- Generalmente el número de registros físicos se usa para calcular los accesos a registros físicos para extraer las filas de una tabla.
- La distribución de los valores de las columnas es necesaria para estimar la fracción de filas que satisfacen una condición en una consulta.
- Es de gran utilidad guardar datos estadísticos de columnas que se encuentren relacionadas para evitar caer en errores, cuando se estime la fracción de filas que satisfacen las condiciones que conectan las columnas a través de los operadores lógicos (AND, OR).

Diseño físico de las bases de datos: Generalidades (cont.)

- ➔ Los perfiles de las aplicaciones resumen las consultas, formularios y reportes que acceden a una base de datos.
- ➔ Para los formularios se debe especificar la frecuencia del uso del formulario para cada operación (inserción, actualización, eliminación y recuperación).
- ➔ Para las consultas y reportes, la distribución de los valores de los parámetros codifica el número de veces que se ejecuta la consulta/reporte con varios valores para los parámetros.
- ➔ La frecuencia de los datos se especifica como un promedio de la unidad de tiempo, por ejemplo un día. Se pueden resumir las frecuencias con más detalle, por ejemplo, considerando las frecuencias pico y la varianza de las frecuencias.
- ➔ Clasificar las aplicaciones según su importancia, por ejemplo considerando los tiempos de respuesta límite, puede ser de utilidad para que los diseños físicos de las bases de datos tengan cierta parcialidad hacia las aplicaciones críticas.

Diseño físico de las bases de datos: Generalidades (cont.)

- ➡ Otro aspecto fundamental dentro del diseño físico de las bases de datos es la selección de la estructura de los archivos.
- ➡ Las estructuras de archivos disponibles son:
 - Archivos secuenciales: tienen una organización simple en la que los registros se almacenan en el orden de inserción o mediante el valor de una clave, son más fáciles de mantener y proporcionan un buen desempeño al procesar un gran número de registros. Se desempeñan bien en búsquedas secuenciales pero mal en búsquedas con claves.
 - Archivos hash: una estructura de archivos especializada que soporta la búsqueda por medio de una clave, transforman el valor de una clave en una dirección para proporcionar un acceso rápido. Se desempeñan bien en búsquedas con claves pero mal en búsquedas secuenciales.
 - Archivos Btree: estructura de archivos soportada por la mayoría de los DBMS que proporciona un buen desempeño tanto en búsquedas con claves como secuenciales. Un archivo Btree es un árbol multiforme, balanceado.
 - Archivos B+Tree: la variante más popular de un Btree. En un B+Tree, todas las claves se encuentran almacenadas de forma redundante en los nodos hoja. El B+Tree proporciona un mejor desempeño en las búsquedas secuenciales y de rangos.

Diseño físico de las bases de datos: Generalidades (cont.)

- ➔ Características de un archivo Btree: es un tipo especial de árbol, con una estructura en la cual cada nodo tiene solo padre, a excepción del nodo raíz o nodo superior.
- ➔ Características de la estructura Btree:
 - Balanceado: todos los nodos hoja (nodos que no tienen hijos) residen en el mismo nivel del árbol.
 - Tupido (bushy): el número de ramas de un nodo es grande. Multiforme, con más de dos, es un sinónimo de arbusto. El ancho (número de flechas a partir de un nodo) y la altura (número de nodos entre los nodos raíz y hoja) están inversamente relacionados: mientras aumente el ancho, disminuye la altura. El Btree ideal es amplio (de tipo arbusto) pero pequeño (pocos niveles).
 - Orientado a bloques (*Block-Oriented*): cada nodo de un Btree es un bloque o un registro físico. Para buscar en un Btree, se comienza en la raíz y se sigue una ruta hasta el nodo hoja que contenga los datos que le interesan. La altura de un Btree es importante porque determina el número de accesos a registros físicos durante la búsqueda.
 - Dinámico: la forma de un Btree cambia mientras se insertan y borran registros lógicos. Nunca es necesario hacer una reorganización periódica para un Btree.
 - Ubicuo: el Btree es una estructura de archivos ampliamente implementada y usada.

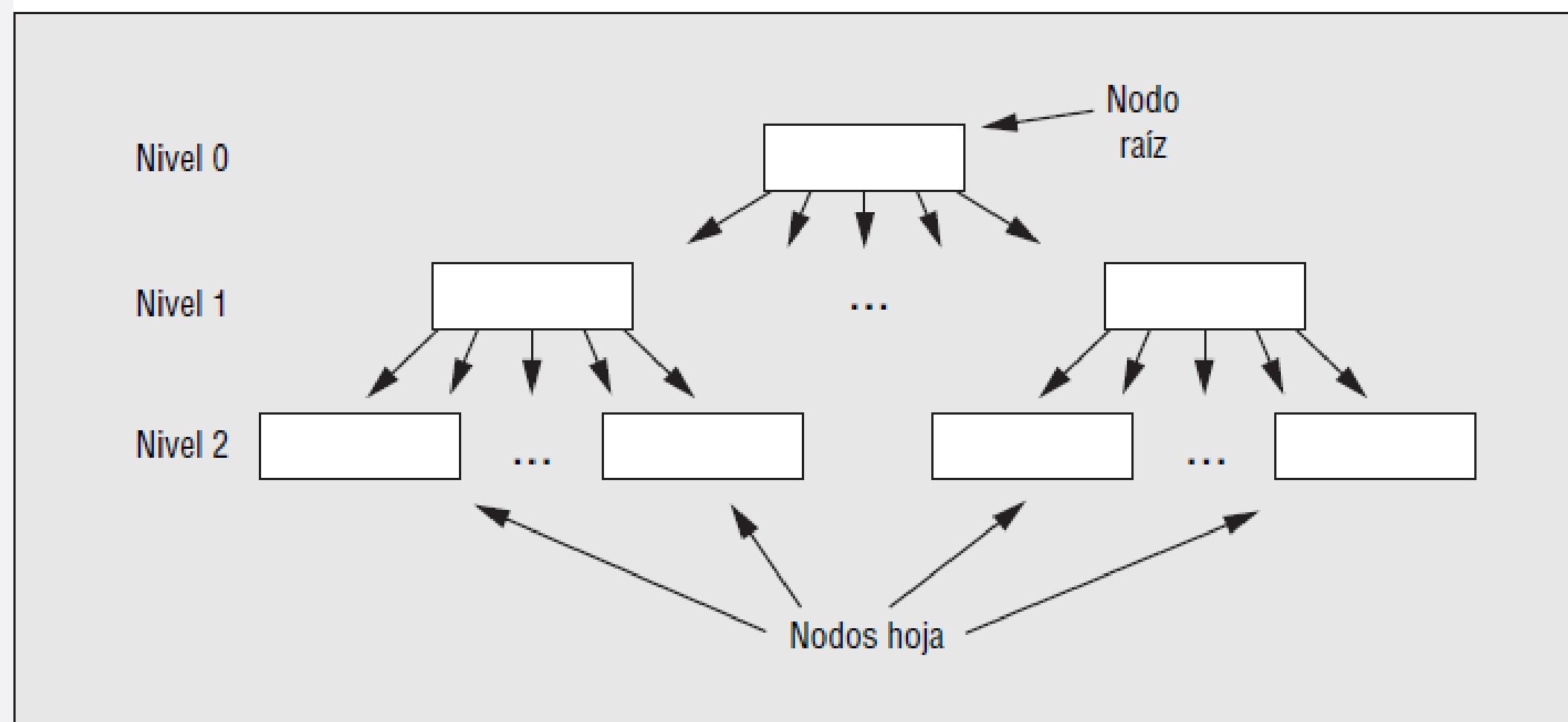
Diseño físico de las bases de datos: Generalidades (cont.)

→ Contenido de un nodo:

- Cada nodo está formado por pares con un valor clave y un puntero (dirección física del registro), ordenados por el valor de la clave.
- El puntero identifica el registro físico que contiene el registro lógico con el valor de la clave. Los otros datos pueden almacenarse en registros físicos separados, o en los nodos hoja.
- Cada nodo, excepto el raíz, debe estar lleno por lo menos hasta la mitad.
- El tamaño del registro físico es de 1.024 bytes, el tamaño de la clave es de 4 bytes y el tamaño del puntero de 4 bytes.
- La máxima capacidad de un nodo es de 128 pares <clave, puntero>
- El tamaño de la clave determina el número de ramas.
- Generalmente no son buenos cuando se usan tamaños de claves grandes, ya que se tienen menos ramas por cada nodo resultando Btrees más altos y menos eficientes.

Diseño físico de las bases de datos: Generalidades (cont.)

Estructura de un Btree de nivel 3

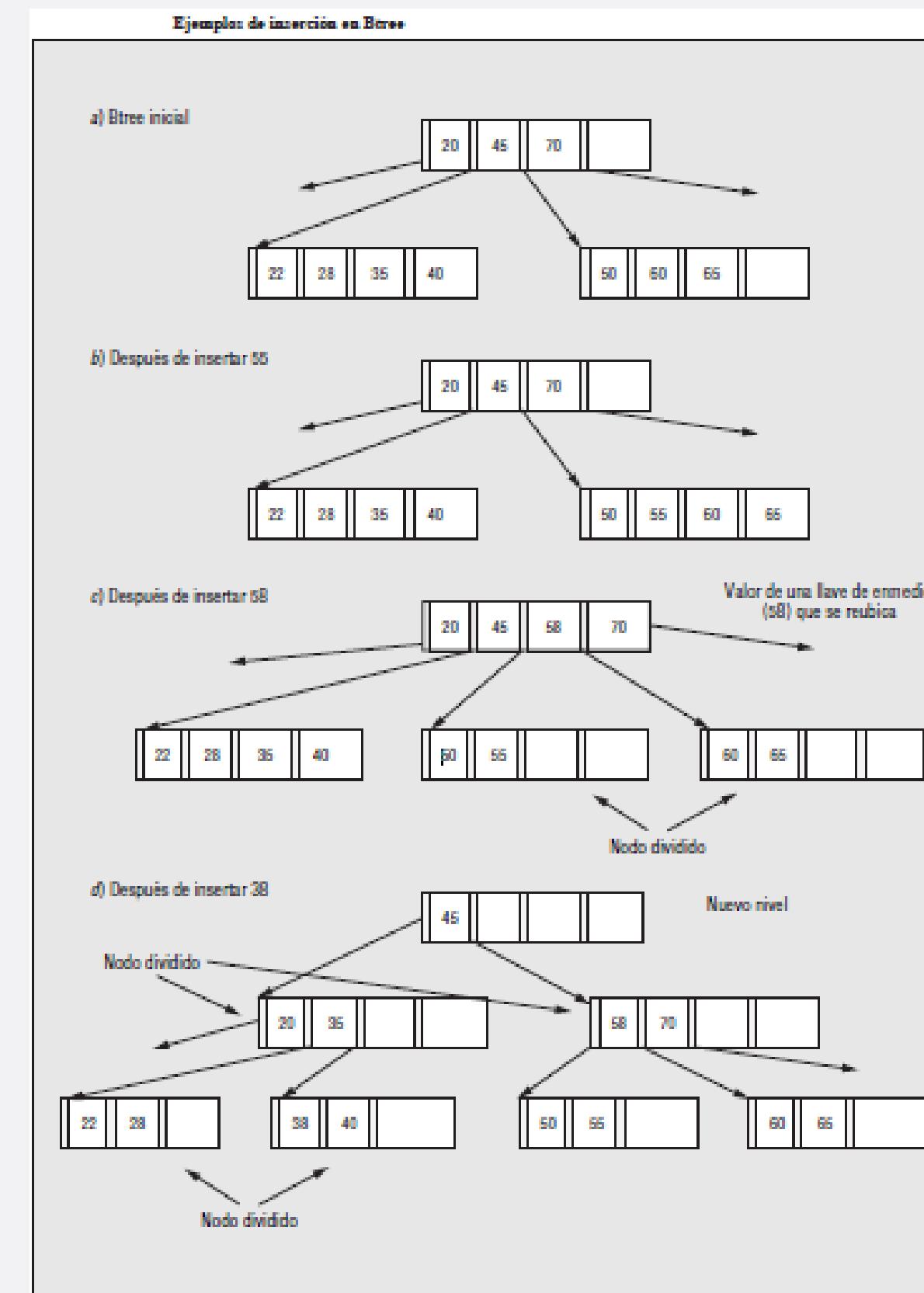


Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.10, pp. 260), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

→ División e integración de nodos:

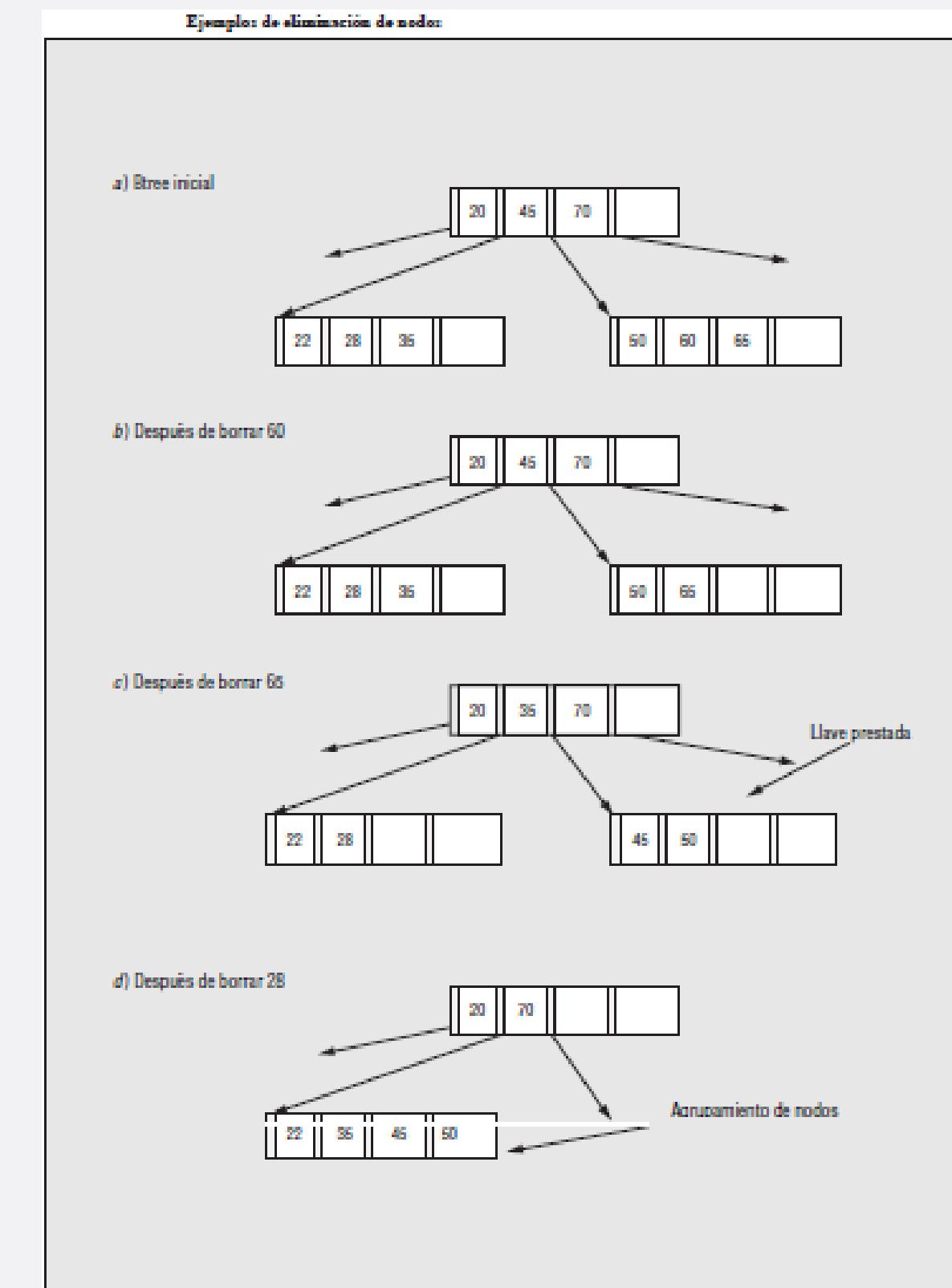
- Las inserciones suceden al colocar una nueva clave en un nodo que no esté lleno o bien dividiendo nodos.
- Puede suceder que al agregar un valor nuevo, el nodo se divida en dos nodos y se coloque un valor de la clave en el nodo raíz, el nodo se dividirá en dos niveles.
- Si ambos nodos están llenos, cuando ocurre una división en la raíz, el árbol crece otro nivel.



Diseño físico de las bases de datos: Generalidades (cont.)

→ División e integración de nodos:

- Las eliminaciones se manejan quitando la clave eliminada de un nodo y reparando la estructura en caso de ser necesario.
- Si el nodo aún se encuentra parcialmente lleno, ninguna acción adicional ocurrirá.
- Si el nodo se encuentra a menos de la mitad, se modifica la estructura.
- Si un nodo vecino contiene más de la mitad de su capacidad, se puede pedir prestada una clave.
- Si no se puede obtener una clave, se deben juntar los nodos.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.11, pp. 262), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

➔ Costos de las operaciones:

- La altura de un Btree es más pequeña para una tabla grande cuando el factor de ramificación es grande.
- En las operaciones Btree la altura domina el número de accesos a registros físicos.
- El costo en términos de accesos a registros físicos para encontrar una clave es menor que o igual a la altura.
- Si los datos de la fila no están almacenados en el árbol, se requiere de otro acceso a un registro físico para obtener los datos de la fila después de encontrar la clave.
- El costo de insertar alguna clave incluye el costo para localizar la clave más cercana, más el costo de modificar los nodos.
- En el mejor de los casos, el costo adicional es un acceso a un registro físico para modificar el registro del índice y un acceso a un registro físico para escribir los datos de la fila.
- El peor de los casos ocurre cuando se agrega un nivel nuevo al árbol, teniendo en cuenta que en este último escenario aún domina la altura del árbol.

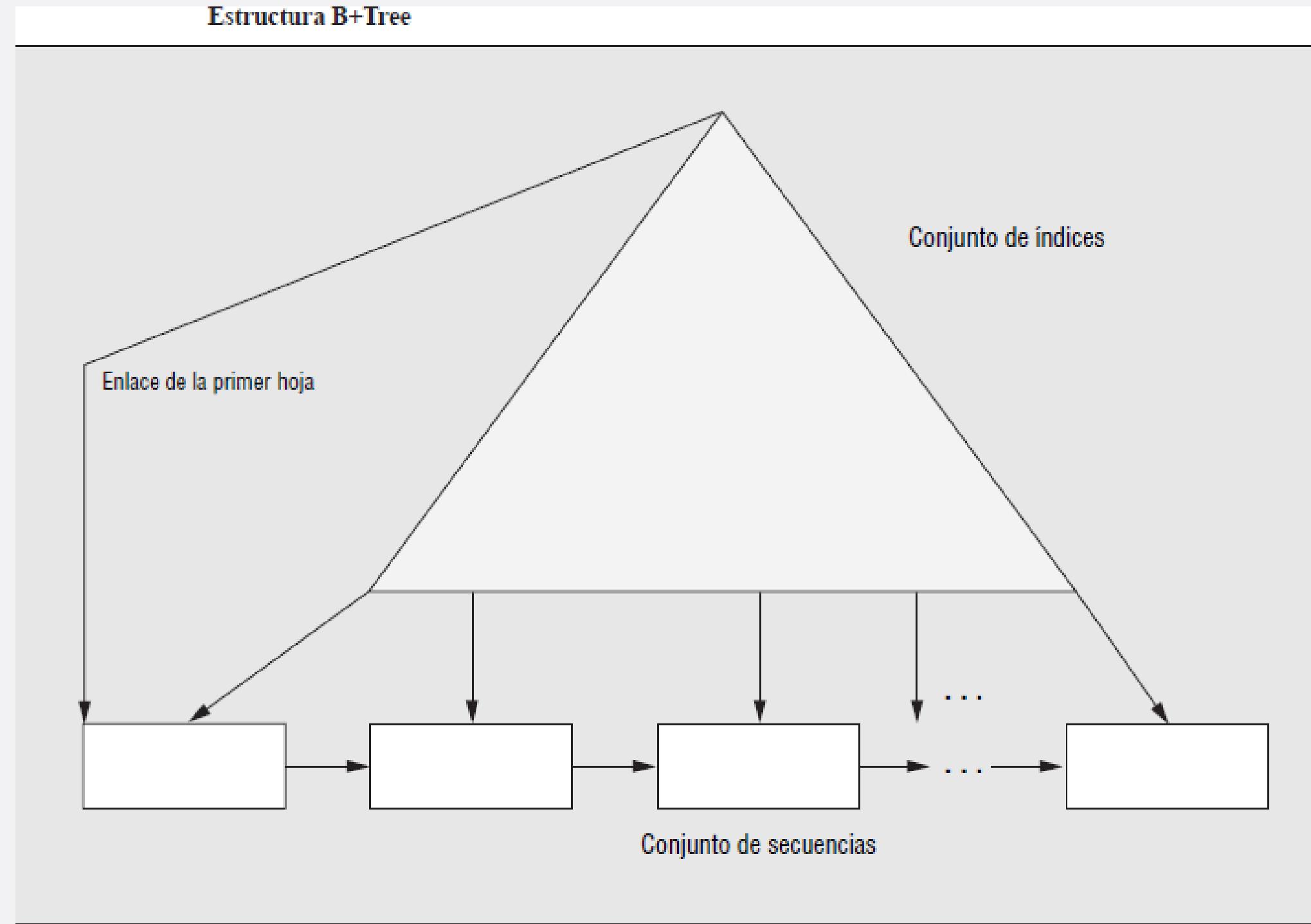
Diseño físico de las bases de datos: Generalidades (cont.)

➔ Costos de las operaciones (cont.):

- Las búsquedas secuenciales pueden ser un problema con los Btrees.
- Para realizar una búsqueda de un rango, el procedimiento de búsqueda debe viajar hacia arriba y hacia abajo del árbol.
- Este procedimiento presenta problemas con la conservación de registros físicos en memoria.
- Los sistemas operativos pueden reemplazar los registros físicos si no fueron accesados recientemente.
- Debido a que suele pasar cierto tiempo antes de que se acceda nuevamente a un nodo padre, el sistema operativo puede reemplazarlo con otro registro físico en caso de que la memoria principal se llene.
- Por ende, tal vez sea necesario otro acceso a un registro físico cuando se acceda nuevamente al nodo padre.
- Para asegurar que no se reemplacen los registros físicos, usualmente se implementa la variación del B+Tree.

Diseño físico de las bases de datos: Generalidades (cont.)

→ B+Tree:



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.14, pp. 264), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Diseño físico de las bases de datos: Generalidades (cont.)

→ B+Tree:

- El triángulo (conjunto de índices) representa un índice normal de un Btree.
- La parte inferior (conjunto de secuencias) contiene los nodos hoja.
- Todas las claves se ubican en los nodos hoja incluso cuando una clave aparece en el conjunto de índices.
- Los nodos hoja están conectados para que las búsquedas secuenciales no necesiten moverse hacia arriba del árbol.
- Una vez que se encuentra la clave inicial, el proceso de búsqueda accede únicamente a los nodos del conjunto de la secuencia.

Diseño físico de las bases de datos: Generalidades (cont.)

→ Coincidencia de índices:

- Se refiere a la determinación de si se puede usar un índice en una consulta.
- Se puede usar un Btree para almacenar todos los datos en los nodos (estructura de archivos primaria) o solo los punteros a los registros de datos (estructura de archivos secundaria o índice).
- Un Btree es especialmente versátil como índice, ya que se puede usar para almacenar diversas consultas.
- En una cláusula WHERE cuando una condición hace referencia a una columna indexada, el DBMS debe determinar si es posible usar el índice.
- Para índices de columna única, un índice coincide con una condición si la columna aparece sola, sin funciones u operadores, y el operador de comparación coincide con uno de los siguientes elementos:
 - =, >, <, >=, <= (pero no <>)
 - BETWEEN
 - IS NULL
 - IN <lista de valores constantes>
 - LIKE ‘Patrón’ en el cual patrón no contiene ningún metacarácter (%,_)como la primera parte del patrón.

Diseño físico de las bases de datos: Generalidades (cont.)

→ Coincidencia de índices (cont.):

- Para los índices compuestos que involucren más de una columna, las reglas de coincidencia son más complejas y restrictivas.
- Los índices compuestos están ordenados de la columna más significativa (primera columna del índice) a la columna menos significativa (última columna del índice).
- Un índice compuesto coincide con las condiciones de acuerdo con las siguientes reglas:
 - La primera columna del índice debe tener una condición de coincidencia.
 - Las columnas coinciden de izquierda (más significativo) a derecha (menos significativo).
 - La coincidencia se detiene cuando la siguiente columna del índice no coincide.
 - Por lo menos una condición BETWEEN coincide.
 - Ninguna otra condición coincide después de la condición BETWEEN.
 - Por lo menos una condición IN coincide con una columna índice.
 - Las coincidencias se detienen después de la siguiente condición.
 - La segunda condición no puede ser IN o BETWEEN.

Diseño físico de las bases de datos: Generalidades (cont.)

→ Coincidencia de índices (cont.):

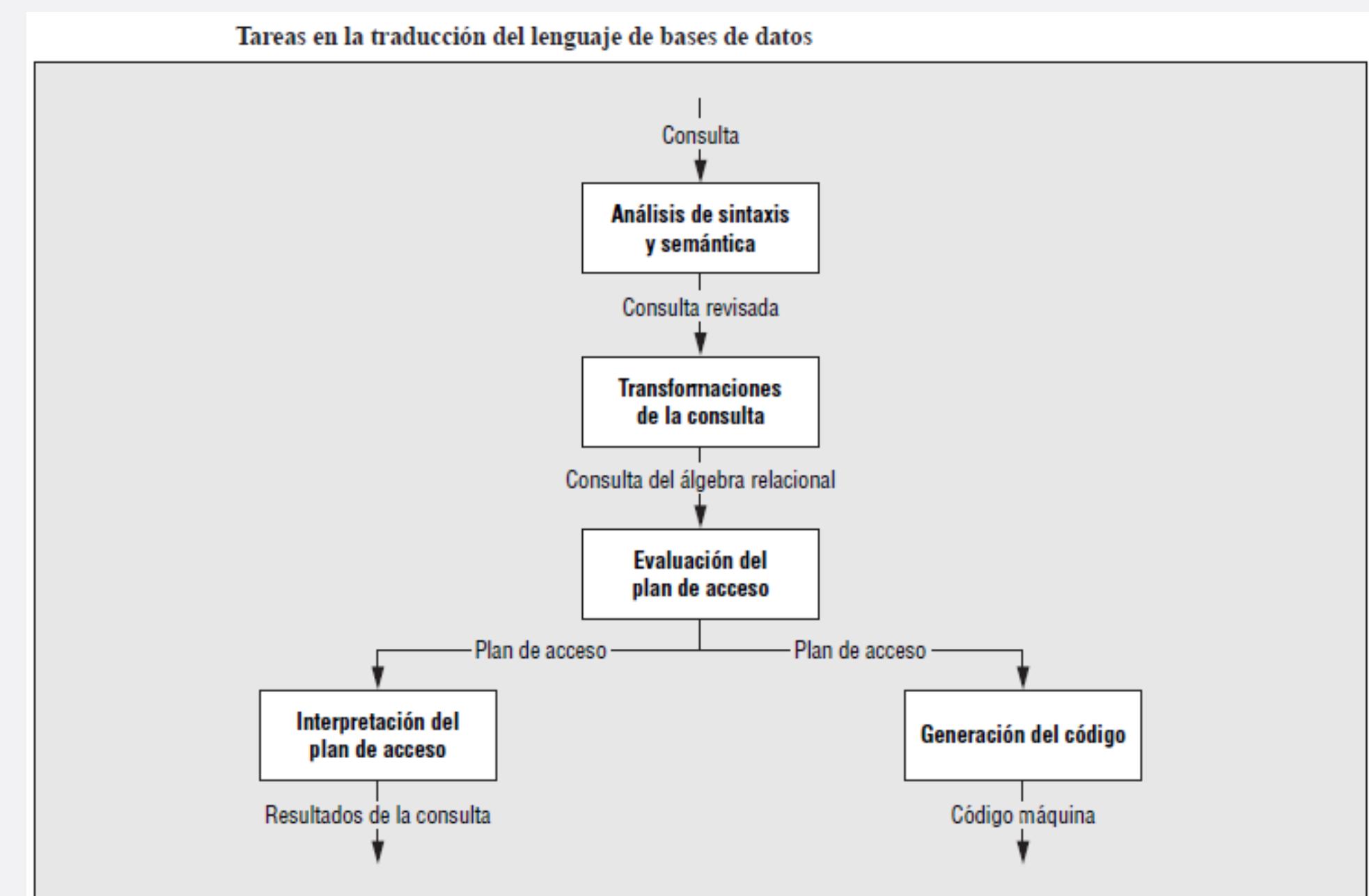
- Cuando se utiliza un índice compuesto, las condiciones pueden estar en cualquier orden.
- Los índices compuestos se deben usar con precaución debido a las reglas de restricción de coincidencias.
- Generalmente, es mejor crear índices sobre las columnas individuales, ya que la mayoría de los DBMS pueden combinar los resultados de varios índices cuando se responde a una consulta.

→ Índices bitmap:

- Estructura secundaria de archivos consistente en un valor de columna y un bitmap.
- Un bitmap contiene una posición de bit para cada fila de la tabla referenciada.
- Un índice de columna bitmap hace referencia a las filas que contienen el valor de la columna.
- Un índice bitmap de enlace hace referencia a las filas de una tabla hija que se une con filas de la tabla madre contenidas en la columna.
- Los índices bitmap funcionan correctamente para columnas estables con algunos valores típicos de tablas en un almacén de datos.

Optimización de consultas

- El componente de optimización de consultas asume la responsabilidad de escoger la mejor implementación de las consultas sobre la base de datos física.
- En algunas ocasiones se pueden mejorar esas decisiones de optimización comprendiendo los principios del proceso de optimización.
- Tareas de traducción: El componente de optimización de consultas traduce la consulta SQL en cuatro fases.



Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.17, pp. 269), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Optimización de consultas (cont.)

→ Fase 1: Sintaxis y análisis de semántica

- Analiza una consulta para encontrar errores de sintaxis y semántica simple.
- Los errores de sintaxis involucran el mal uso de palabras clave, por ejemplo la palabra reservada FROM.
- Los errores de semántica incluyen el mal uso de columnas y tablas.
- Para encontrar errores de semántica, el DBMS usa las definiciones de tablas, columnas y relaciones, tal como se almacenan en el diccionario de datos.
- El compilador del lenguaje de datos puede detectar únicamente errores simples de semántica que involucren tipos de datos incompatibles.
- Por ejemplo, una condición WHERE que compare columnas con tipos de datos incompatibles, generando un error semántico.

Optimización de consultas (cont.)

→ Fase 2: Transformación de consultas

- Transforma una consulta en un formato simplificado y estandarizado.
- Como con la optimización de los compiladores de lenguajes de programación, los traductores del lenguaje de base de datos pueden eliminar las partes redundantes de una expresión lógica. Por ejemplo, la expresión lógica (`OffYear = 2006 AND OffTerm = 'WINTER'`) OR (`OffYear = 2006 AND OffTerm = 'SPRING'`) se puede simplificar en `OffYear = 2006 AND (OffTerm = 'WINTER' OR OffTerm = 'SPRING')`.
- El formato estandarizado generalmente se basa en el álgebra relacional.
- Las operaciones del álgebra relacional se reacomodan de tal forma que la consulta se pueda ejecutar de manera más rápida.
- Las operaciones de restricción se combinan para que se puedan probar de manera conjunta.
- Las operaciones de proyección y restricción se mueven para que estén antes de las operaciones de enlace (join) para eliminar las columnas y renglones innecesarios antes de las operaciones de enlace costosas.
- Las operaciones de productos cruz se transforman en operaciones de enlace si una condición existe en la cláusula WHERE.

Optimización de consultas (cont.)

→ Fase 3: Evaluación del plan de accesos

- Genera un plan de acceso para implementar la consulta reacomodada del álgebra relacional.
- Un plan de accesos indica la implementación de una consulta como operaciones en archivos.
- En un plan de accesos, los nodos hoja son tablas individuales de la consulta y las flechas apuntan hacia arriba para indicar el flujo de datos.
- Los nodos que están sobre los nodos hoja señalan las decisiones sobre el acceso a las tablas individuales.
- Los índices Btree se usan para acceder a las tablas individuales.
- Las estructuras de archivos Btree proporcionan la clasificación requerida por algoritmos de MERGE JOIN.
- El componente de optimización de consultas evalúa un gran número de planes de acceso.
- Los planes de acceso varían debido al orden de los JOINS, estructuras de archivo y algoritmos de JOIN.

Optimización de consultas (cont.)

→ Fase 3: Evaluación del plan de accesos (cont.)

- Para las estructuras de archivos, algunos componentes de optimización pueden considerar operaciones de conjuntos (intersección para las condiciones conectadas por un AND y las condiciones de enlace conectadas por un OR) para combinar los resultados de múltiples índices sobre la misma tabla.
- El componente de optimización de consultas puede evaluar muchos más planes de acceso que los que un programador de bases de datos experimentado pudiese considerar.
- La evaluación de los planes de acceso puede involucrar una significativa cantidad de tiempo cuando la consulta contiene más de cuatro tablas.
- La mayoría de los componentes de optimización utilizan un pequeño conjunto de algoritmos de *join*.
- Para cada operación de *join* en una consulta, el componente de optimización considera cada algoritmo de *join* soportado.
- Para los *join* anidados y los algoritmos híbridos, el componente de optimización también debe seleccionar las tablas externa e interna.
- Todos los algoritmos, a excepción del enlace de estrella (*star join*), involucran dos tablas al mismo tiempo.

Optimización de consultas (cont.)

→ Fase 3: Evaluación del plan de accesos (cont.)

- El enlace estrella puede combinar cualquier número de tablas que coincidan con el patrón de estrella (una tabla hija rodeada por tablas madre con relaciones 1-M).
- El algoritmo de ciclos anidados se puede usar con cualquier operación de enlace, no sólo con una operación EQUI-JOIN.
- El componente de optimización de consultas usa fórmulas de costos para evaluar los planes de acceso.
- Cada operación en un plan de acceso tiene una fórmula de costo correspondiente que estima los accesos de registros físicos y operaciones del CPU.
- Las fórmulas de costos usan los perfiles de las tablas para estimar el número de filas de un resultado. Por ejemplo, el número de filas que se generan de una condición WHERE se puede estimar usando los datos de distribución, tales como un histograma.
- El componente de optimización de consultas selecciona el plan de acceso con el menor costo.

Diseño físico de las bases de datos: Generalidades

→ Resumen de algoritmos de JOIN comunes:

Algoritmo	Requerimientos	Cuándo usarlo
Ciclos anidados	Seleccione la tabla externa e interna; se puede usar para todos los <i>joins</i> .	Apropiado cuando existen pocas filas en la tabla externa o cuando todas las páginas de la tabla interna caben en la memoria. Un índice en la llave foránea de la columna de <i>join</i> permite un uso eficiente del algoritmo de ciclos anidados cuando existen condiciones de restricción en la tabla madre
Mezcla ordenada (<i>sort merge</i>)	Ambas tablas deben estar ordenadas (o usar un índice) en las columnas de <i>join</i> ; solo se usa para los <i>equi-joins</i> .	Apropiado si el costo del ordenamiento es pequeño o si existe un índice de <i>joins agrupados</i> .
Enlace híbrido (<i>hash join</i>)	Archivo hash interno construido para ambas tablas; solo se usa para los <i>equi-joins</i> .	Un enlace <i>hash</i> es mejor que un <i>sort merge</i> cuando las tablas no están ordenadas o no existen índices.
Enlace de estrella (<i>star join</i>)	Enlace de múltiples tablas en el cual existe una tabla hija relacionada con múltiples tablas madre en relaciones 1-M; se requiere un índice de <i>join bitmap</i> en cada tabla madre; solo se usa para <i>equi-joins</i> .	Es el mejor algoritmo de <i>join</i> para las tablas que coinciden con el patrón de estrella con índices de <i>join</i> de bitmap en especial cuando existen condiciones altamente selectivas en las tablas madre; ampliamente usado para optimizar las consultas de una <i>data warehouse</i> .

Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.8, pp. 271), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Optimización de consultas (cont.)

→ Fase 4: Ejecución del plan de acceso

- La última fase ejecuta el plan de acceso seleccionado.
- El componente de optimización de consultas genera el código máquina o interpreta el plan de acceso.
- La ejecución del código máquina genera una respuesta más rápida que la interpretación del plan de acceso.
- La mayoría de los DBMS interpretan los planes de acceso debido a la amplia variedad de *hardware* soportado.
- La diferencia en el desempeño entre la interpretación y la ejecución del código máquina generalmente no es significativa para la mayoría de los usuarios.

Resumen de entradas, salidas y entorno del diseño físico de bases de datos

Elemento	Descripción
Entradas	
Perfil de tablas	Estadísticas para cada tabla, como número de filas y de columnas de valores únicos.
Perfil de aplicación	Estadísticas para cada formulario, reporte y consulta, tales como accesos/actualizaciones y la frecuencia de los accesos/actualizaciones.
Salidas	
Estructuras de archivos	Métodos de organización de registros físicos para cada tabla.
Colocación de datos	Criterios para acomodar los registros físicos de forma cercana.
Formateo de datos	Uso de la comprensión y de los datos derivados.
Desnormalización	Combinación de tablas separadas en una sola tabla
Conocimiento del entorno	
Estructuras de archivos	Características como las operaciones respaldadas y fórmulas de costo.
Optimización de consultas	Decisiones de accesos hechas por el componente de optimización para cada una de las consultas.

Nota: Adaptado de *Administración de base de datos diseño y desarrollo de aplicaciones* (fig. 8.1, pp. 252), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Ejemplo práctico. Tablas a considerar

Ejemplo de tabla de inscripción *Enrollment*

OfferNo	StdSSN	EnrGrade
1234	123-45-6789	3.3
1234	234-56-7890	3.5
4321	123-45-6789	3.5
4321	124-56-7890	3.2

Ejemplo de tabla de cursos ofrecidos *Offering*

OfferNo	CourseNo	OffTerm	OffYear	OffLocation	OffTime	FacSSN	OffDays
1111	IS320	SUMMER	2006	BLM302	10:30 AM	-	MW
1234	IS320	FALL	2005	BLM302	10:30 AM	098-76-5432	MW
2222	IS460	SUMMER	2005	BLM412	1:30 PM	-	TTH
3333	IS320	SPRING	2006	BLM214	8:30 AM	098-76-5432	MW
4321	IS320	FALL	2005	BLM214	3:30 PM	098-76-5432	TTH
4444	IS320	SPRING	2006	BLM302	3:30 PM	543-21-0987	TTH
5678	IS480	SPRING	2006	BLM302	10:30 AM	987-65-4321	MW
5679	IS480	SPRING	2006	BLM412	3:30 PM	876-54-3210	TTH
9876	IS460	SPRING	2006	BLM307	1:30 PM	654-32-1098	TTH

Nota: Tablas adaptadas de *Administración de base de datos diseño y desarrollo de aplicaciones* (tab. 3.3, pp. 48 y tab. 3.4, pp. 48), por Michael V. Mannino, 2007, McGraw-Hill/Interamericana.

Ejemplo práctico. Tablas a considerar (cont.)

Ejemplo de tabla de inscripción *Faculty*

FacSSN	FacFirstName	FacLastName	FacCity	FacState	FacDept	FacRank	FacSalary	FacSupervisor	FacHireDate	FacZipCode
098-76-5432	LEONARD	VINCE	SEATTLE	WA	MS	ASST	\$35,000	654-32-1098	01-Apr-95	98111-9921
543-21-0987	VICTORIA	EMMANUEL	BOTHELL	WA	MS	PROF	\$120,000	-	01-Apr-96	98111-2242
654-32-1098	LEONARD	FIBON	SEATTLE	WA	MS	ASSC	\$70,000	543-21-0987	01-Apr-95	98121-0094
765-43-2109	NICKI	MACON	BELLEVUE	WA	FIN	PROF	\$65,000	-	01-Apr-97	98015-9945
876-54-3210	CRISTOPHER	COLAN	SEATTLE	WA	MS	ASST	\$40,000	654-32-1098	01-Apr-99	98114-1332
987-65-4321	JULIA	MILLS	SEATTLE	WA	FIN	ASSC	\$75,000	765-43-2109	01-Apr-00	98114-9954

Ejemplo práctico. SQL a considerar

Enlace de tres tablas

```
SELECT FacName, CourseNo, Enrollment.OfferNo, EnrGrade  
      FROM Enrollment, Offering, Faculty  
     WHERE CourseNo LIKE 'IS%' AND OffYear = 2005  
           AND OffTerm = 'FALL'  
           AND Enrollment.OfferNo = Offering.OfferNo  
           AND Faculty.FacSSN = Offering.FacSSN
```

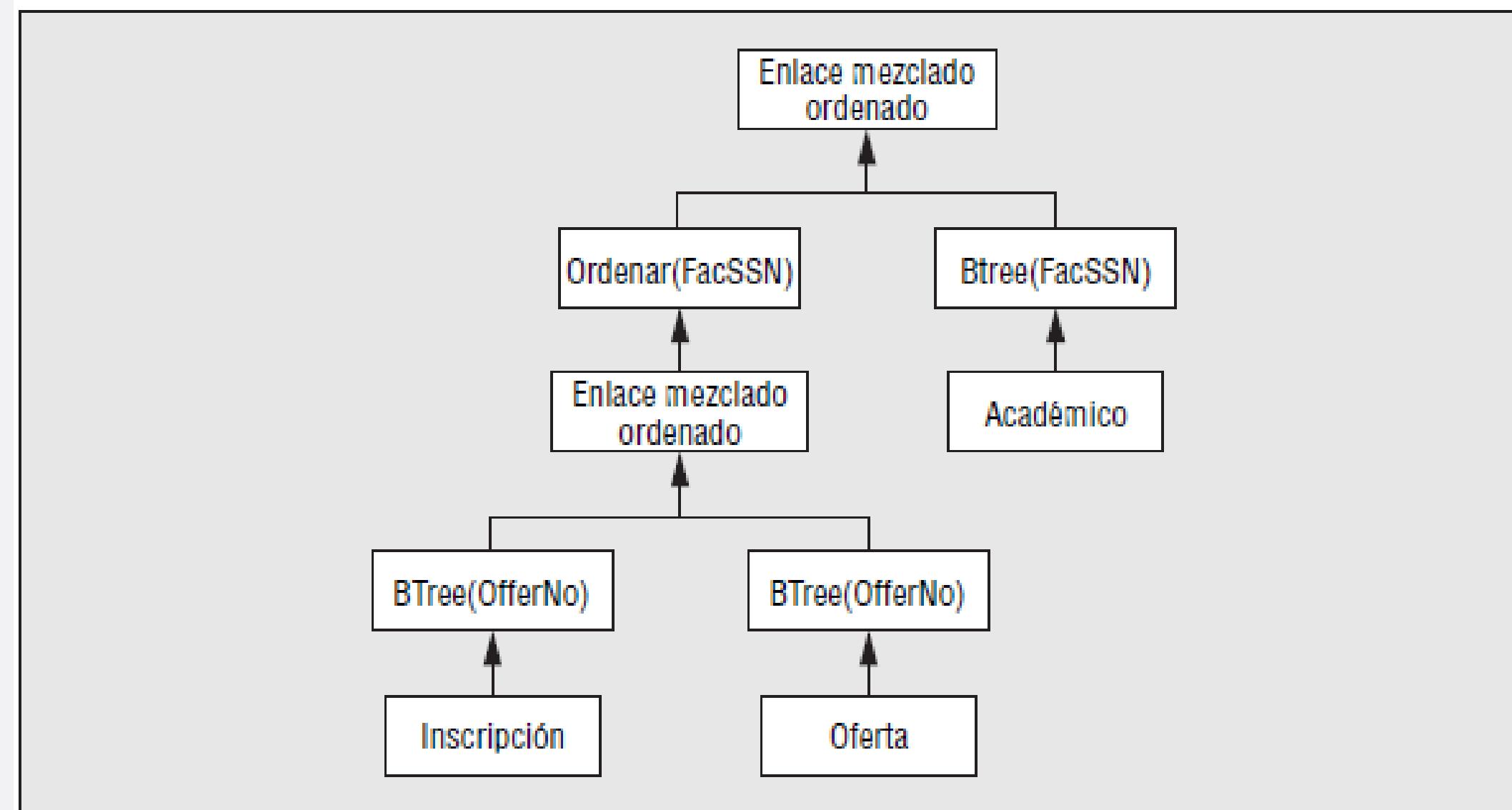
Ejemplo práctico

→ Plan de acceso 1:

- Los nodos hoja son tablas individuales de la consulta y las flechas apuntan hacia arriba para indicar el flujo de datos.
- Los nodos que están sobre los nodos hoja señalan las decisiones sobre el acceso a las tablas individuales.
- Los índices Btree se usan para acceder a las tablas individuales.
- El primer enlace combina las tablas Enrollment y Offering.
- Las estructuras de archivos Btree proporcionan la clasificación requerida por el algoritmo de merge join.
- El segundo enlace (*join*) combina el resultado del primer *join* con la tabla Faculty.
- El resultado intermedio debe estar ordenado en FacSSN antes de que se pueda usar el algoritmo de *merge join*.

Ejemplo práctico

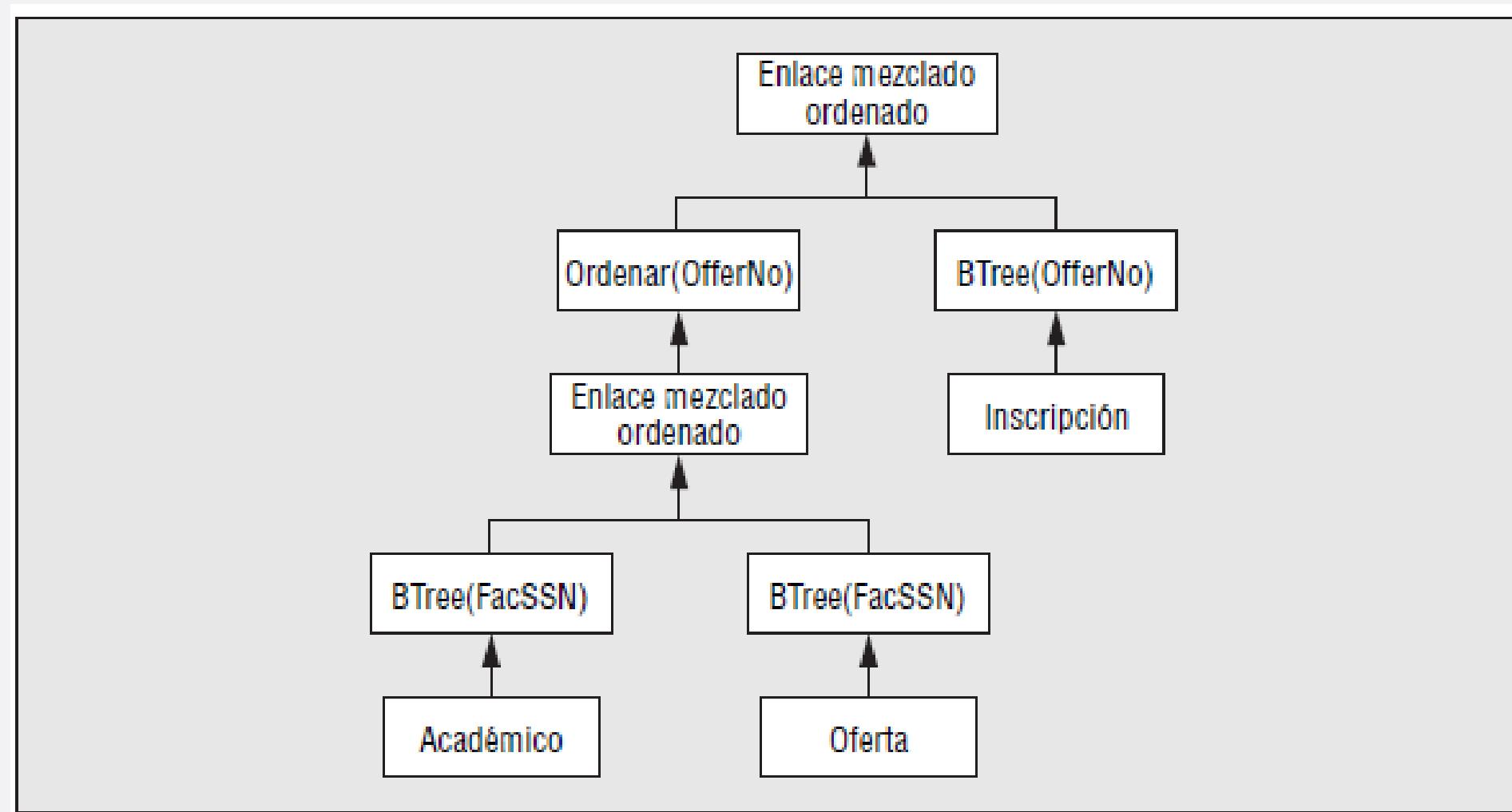
→ Gráfico plan de acceso 1:



Ejemplo práctico

→ Plan de acceso 2:

- Variación del plan de acceso 1 en el que el orden del *join* está modificado. Los planes de acceso varían debido al orden de los *joins*, estructuras de archivo y algoritmos de *join*.1





© Universidad de Palermo
Prohibida la reproducción total o parcial de imágenes y textos.

MÓDULO X

MONITOREO Y AFINAMIENTO DE BASE DE DATOS

MONITOREO SOBRE EL COMPORTAMIENTO DE LOS SGBD Y LOS SISTEMAS OPERATIVOS.

GENERALIDADES DE AFINAMIENTO AL SGBD Y LOS SISTEMAS OPERATIVOS

Marcela Russo
Laboratorio IV

MONITOREO Y AFINAMIENTO

- Si bien el componente de optimización de consultas lleva a cabo su función de forma automática, el administrador de bases de datos (DBA) tiene la responsabilidad de monitorear el comportamiento de la base de datos y aplicar las acciones necesarios para optimizar su rendimiento.
- El DBA debe revisar los planes de acceso de consultas y actualizaciones que se desempeñen mal.
- Los DBMS corporativos proporcionan despliegues gráficos de los planes de acceso para facilitar la revisión. Para mejorar las malas decisiones de los planes de accesos, algunos DBMS corporativos permiten incluir pistas que influyan en la selección de los planes de acceso.
- Algunos motores de base de datos proporcionan sugerencias para seleccionar la meta de optimización, las estructuras de archivos para acceder a las tablas individuales, el algoritmo y el orden del JOIN. Las sugerencias deben usarse con precaución ya que pueden invalidar la decisión del optimizador.
- Las propuestas acerca de los algoritmos de JOIN y el orden de los JOIN son especialmente problemáticos debido a lo sutil de estas decisiones. Anular las decisiones del optimizador sólo debe hacerse como último recurso después de determinar la causa del mal desempeño.
- El DBA puede reparar problemas con deficiencias en el perfil de tablas y en el estilo de codificación de consultas para mejorar el desempeño en lugar de anular la decisión del optimizador.

MONITOREO Y AFINAMIENTO

- Deficiencias del perfil de tablas:
 - ✓ El componente de optimización de consultas necesita estadísticas detalladas y actualizadas para evaluar los planes de acceso.
 - ✓ La mayoría de los DBMS proporciona control sobre el nivel de detalle de las estadísticas y su actualización.
 - ✓ Algunos DBMS permiten el muestreo dinámico de la base de datos durante el tiempo de optimización, pero normalmente este nivel de actualización de datos no es necesario.
 - ✓ Si las estadísticas no se recopilan para una columna, la mayoría de los DBMS utilizan el valor uniforme que se asume para estimar el número de filas.
 - ✓ El uso del valor uniforme que se asume generalmente lleva a un acceso secuencial de archivos en lugar de un acceso con un Btree, esto en el caso de que la columna tenga suficientes inclinaciones en sus valores.

MONITOREO Y AFINAMIENTO

- Deficiencias del perfil de tablas:
 - ✓ Por ejemplo, en una consulta para listar a los empleados con salarios mayores a \$100 000:
 - Si el rango del salario va de \$10 000 a \$2 000 000, cerca del 95 % de la tabla de empleados deberá satisfacer esta condición utilizando el valor uniforme que se asume.
 - Es probable que para la mayoría de las empresas, pocos empleados tengan un salario mayor a \$100 000.
 - Por lo tanto, utilizando la estimación del valor uniforme que se asume, el optimizador escogerá un archivo secuencial en vez de un Btree para acceder a la tabla de empleados.
 - Como resultado, la estimación no mejorará mucho utilizando un histograma de igual intervalo debido a la inclinación extrema de los valores de los salarios.
 - Un histograma de la misma altura proporcionará mucho mejores estimaciones.
 - Para mejorar el cálculo utilizando un histograma de la misma altura, debe aumentar el número de rangos.
 - Por ejemplo, con 10 rangos el error máximo es de alrededor de 10 % y el error esperado es de cerca de 5%.

MONITOREO Y AFINAMIENTO

- Deficiencias del perfil de tablas:
 - Para disminuir las estimaciones de los errores máximo y esperado en 50%, se debe duplicar el número de rangos.
 - Un DBA debe aumentar el número de rangos si los errores estimados para el número de filas ocasionan selecciones inadecuadas para acceder a las tablas individuales.
 - Una sugerencia puede ser útil para condiciones que incluyan valores de parámetros.
 - Si el DBA sabe que los típicos valores de parámetros generan un conjunto de pocas filas, se puede utilizar una sugerencia para obligar al componente de optimización a que use un índice.

MONITOREO Y AFINAMIENTO

- Un histograma de la misma altura proporcionará mucho mejores estimaciones.
- Para mejorar el cálculo utilizando un histograma de la misma altura, debe aumentar el número de rangos.
- Por ejemplo, con 10 rangos el error máximo es de alrededor de 10 % y el error esperado es de cerca de 5%. Para disminuir las estimaciones de los errores máximo y esperado en 50%, se debe duplicar el número de rangos.
- Un DBA debe aumentar el número de rangos si los errores estimados para el número de filas ocasionan selecciones inadecuadas para acceder a las tablas individuales.
- Una sugerencia puede ser útil para condiciones que incluyan valores de parámetros.
- Si el DBA sabe que los típicos valores de parámetros generan un conjunto de pocas filas, se puede utilizar una sugerencia para obligar al componente de optimización a que use un índice.
- un componente de optimización, en ocasiones necesita estadísticas detalladas de las combinaciones de columnas.
- Si una combinación de columnas aparece en la cláusula WHERE de una consulta, las estadísticas de la combinación de columnas son importantes cuando las columnas no son independientes.

MONITOREO Y AFINAMIENTO

- Por ejemplo, los salarios de los empleados y sus puestos están generalmente relacionados.
- Una cláusula WHERE con las dos columnas probablemente tenga pocas filas que satisfagan ambas condiciones. Un componente de optimización que desconozca la relación entre estas columnas probablemente sobreestimará el número de filas del resultado.
- La mayoría de los componentes de optimización asumen que las combinaciones de columnas son estadísticamente independientes para simplificar la estimación del número de filas.
- Son pocos los DBMS que conservan estadísticas de las combinaciones de columnas.
- Si un DBMS no conserva las estadísticas de las combinaciones de columnas, el diseñador de bases de datos pudiera utilizar las sugerencias para anular la selección del DBMS cuando una condición de JOIN en la cláusula WHERE genere pocas filas.
- El uso de una sugerencia puede obligar al componente de optimización a combinar índices cuando acceda a una tabla en vez de usar un escaneo secuencial de la tabla.

MONITOREO Y AFINAMIENTO

- Prácticas en la codificación de consultas:

- ✓ Las consultas mal escritas pueden ocasionar una ejecución lenta de las mismas.
- ✓ El DBA debe revisar las consultas que se desempeñen mal buscando las prácticas de codificación que ocasionen su lento desempeño.

- ✓ Consideraciones a tener en cuenta en la codificación de consultas:

- No utilizar funciones en columnas que tengan índices, ya que anulan la utilización del índice, incluso las conversiones de tipo implícito eliminan la posibilidad de usar un índice, estas ocurren cuando no coinciden el tipo de dato de una columna y el valor constante asociado.
- La velocidad de ejecución de una consulta se determina principalmente por el número de operaciones de JOIN, por lo que la eliminación de operaciones de JOIN innecesarias puede disminuir significativamente el tiempo de ejecución.
- Para consultas que involucren relaciones 1-M en las que exista una condición en la columna de JOIN, la condición debe estar en la tabla madre en lugar de la tabla hija, esto reduce el esfuerzo al enlazar las tablas.
- Para consultas que involucren la cláusula HAVING, eliminar las condiciones que no involucren funciones agregadas. Las condiciones que involucran comparaciones sencillas de las columnas de la cláusula GROUP BY pertenecen a la cláusula WHERE, no a la cláusula HAVING, esto eliminará las filas más pronto y la ejecución será más rápida

MONITOREO Y AFINAMIENTO

- Prácticas en la codificación de consultas:

- ✓ Consideraciones a tener en cuenta en la codificación de consultas:

- Evitar las consultas anidadas de tipo II (consulta anidada en la que la consulta interna hace referencia a alguna tabla usada en la consulta anidada externa), en especial cuando la consulta anidada lleve a cabo la agrupación con cálculos agregados. Muchos DBMS son lentos porque los componentes de optimización de consultas generalmente no consideran formas eficientes para implementar las consultas anidadas del tipo II. Se puede mejorar la velocidad de ejecución de una consulta reemplazando la consulta anidada del tipo II con una consulta separada.
 - Las consultas con vistas complejas pueden conducir a un lento desempeño debido a que se puede ejecutar una consulta adicional.
 - El proceso de optimización puede consumir tiempo, en especial para las consultas que contienen más de cuatro tablas. Para reducir el tiempo de optimización, la mayoría de los DBMS guardan planes de acceso para evitar las fases que consumen tiempo del proceso de traducción.
 - La ligadura de consultas es el proceso de asociar una consulta con un plan de acceso.
 - La mayoría de los DBMS hacen las ligaduras de forma automática si una consulta cambia o si lo hace la base de datos (estructuras de archivos, perfiles de tablas, tipos de datos, etc.).

MONITOREO Y AFINAMIENTO

Resumen de las prácticas de codificación

Práctica de codificación	Recomendación	Elemento de desempeño
Funciones en las columnas en condiciones	Evite las funciones en columnas	Elimina la posibilidad del uso de un índice
Conversiones de tipo implícito	Use constantes con tipos de datos que coincidan con las columnas correspondientes	Elimina la posibilidad del uso de un índice
Operaciones de enlace adicionales	Elimine operaciones de enlace innecesarias buscando las tablas que no involucren condiciones o columnas	El tiempo de ejecución se determina principalmente por el número de operaciones de enlace
Condiciones en columnas de enlace	Las condiciones sobre las columnas de enlace deben usar la tabla madre y no la tabla hija	La reducción del número de filas de la tabla madre disminuirá el tiempo de ejecución de las operaciones de enlace
Condiciones de las filas en la cláusula HAVING	Mueva las condiciones de las filas de la cláusula HAVING a la cláusula WHERE	Las condiciones sobre las filas en la cláusula WHERE permiten la reducción en el tamaño del resultado intermedio
Consultas anidadas del tipo II con agrupación	Convierta las consultas anidadas del tipo II en consultas separadas	Los componentes de optimización de consultas generalmente no consideran maneras eficientes para implementar las consultas anidadas del tipo II
Consultas usando vistas complejas	Reescriba las consultas que usan vistas complejas para eliminar las referencias a vistas	Se puede ejecutar una consulta adicional
Reutilización de consultas	Asegúrese de que las consultas de un procedimiento almacenado se usen sólo una vez	El uso repetitivo involucra una sobrecarga considerable para las consultas complejas

MONITOREO Y AFINAMIENTO

- Selección de Índices:
 - ✓ Índice: Estructura secundaria de archivos que proporciona una ruta alternativa hacia los datos. En un índice agrupado, el orden de los registros de datos es cercano al orden del índice. En un índice desagrupado, el orden de los registros de los datos no está relacionado al orden del índice.
 - ✓ La selección de índices es la decisión más importante y muy difícil, para el diseñador de la base de datos física. La selección incluye dos tipos de índices: agrupados y desagrupados.
 - Índices agrupados: El orden de las filas es cercano al orden del índice. Cercano significa que los registros físicos que contienen las filas no serán consultados más de una vez si el índice se consulta de forma secuencial. La forma más sencilla de hacer un índice agrupado es ordenando los datos de las filas mediante la columna índice.
 - Índices desagrupados: No tiene la propiedad de cercanía. El orden de las filas no está relacionado con el orden del índice. El mismo registro físico puede ser consultado de forma repetitiva cuando se usa un conjunto de secuencias. Los punteros los nodos del conjunto de secuencias hacia las filas se cruzan muchas veces, indicando que el orden del índice es distinto al orden de las filas.
 - ✓ Generalmente, el problema de selección de índices está restringido a índices Btree y a archivos separados para cada tabla. Los tipos de índices hash y la ubicación de datos de varias tablas en el mismo archivo también pueden influir si no se agregan mejoras al desempeño.

MONITOREO Y AFINAMIENTO

- Selección de Índices:
 - ✓ La mejor selección de índices balancea una recuperación rápida con actualizaciones más lentas.
 - ✓ Un índice desagrupado puede mejorar las recuperaciones al proporcionar un acceso rápido a los registros seleccionados.
 - ✓ Generalmente, menos del 5% de las filas deben satisfacer una condición para que el índice desagrupado sea útil.
 - ✓ Para que los optimizadores soporten accesos a índices múltiples para la misma tabla, los índices desagrupados pueden ser útiles incluso cuando un índice sencillo no proporcione por sí mismo suficiente selección de filas.
 - ✓ La habilidad de usar múltiples índices sobre la misma tabla aumenta la utilidad de los índices desagrupados.
 - ✓ Un índice desagrupado también puede ser útil en un JOIN cuando una de las tablas del enlace tenga un número pequeño de filas en el resultado.

MONITOREO Y AFINAMIENTO

- Selección de Índices:
 - ✓ Un índice agrupado puede mejorar las recuperaciones en más situaciones que un índice desagrupado.
 - ✓ Un índice agrupado es útil en las mismas situaciones que un índice desagrupado a excepción de que el número de filas resultantes puede ser mayor.
 - ✓ Un índice agrupado puede ser útil:
 - Si tal vez 20% de las filas satisfacen la condición asociada de la consulta.
 - En los enlaces ya que evita la necesidad de ordenar, se pueden enlazar mezclando las filas de cada tabla.
 - La mezcla de filas generalmente es una forma más rápida para enlazar tablas cuando éstas no necesiten estar ordenadas.
 - ✓ El costo de conservar índices como resultado de las sentencias INSERT, UPDATE y DELETE balancea las mejoras de recuperación. Las sentencias INSERT y DELETE afectan a todos los índices de una tabla.
 - ✓ La preferencia es que la tabla no tenga muchos índices cuando tiene operaciones frecuentes de inserción y eliminación.

MONITOREO Y AFINAMIENTO

- Selección de Índices:
 - ✓ Las sentencias UPDATE afectan sólo a las columnas enlistadas en la cláusula SET. Si las sentencias UPDATE hechas sobre una columna son frecuentes, se pierde el beneficio del índice.
 - ✓ Las alternativas de los índices agrupados son más sensibles al mantenimiento que las de los índices desagrupados.
 - ✓ Los índices agrupados son más costosos de mantener que los índices desagrupados porque el archivo de datos debe cambiar de forma similar a como lo hace un archivo secuencial ordenado.
 - ✓ Para los índices desagrupados, el archivo de datos se puede mantener como se hace con un archivo secuencial desordenado.

MONITOREO Y AFINAMIENTO

- Dificultades en la selección de Índices: La selección de índices por varias razones, por tal motivo, los DBMS corporativos y algunos otros fabricantes proporcionan herramientas asistidas por computadora para ayudar en la selección.
- ✓ Es difícil especificar los pesos de las aplicaciones. Las decisiones que combinan la frecuencia e importancia pueden hacer que el resultado sea subjetivo.
- ✓ En ocasiones se necesita la distribución de los valores de los parámetros. Muchas sentencias SQL usadas en los reportes y formularios usan valores en los parámetros. Si los valores de los parámetros varían desde ser muy selectivos hasta no serlo, la selección de índices es difícil.
- ✓ Se debe conocer el comportamiento del componente de optimización de consultas. Incluso si un índice parece ser útil para alguna consulta, el componente de optimización de consultas debe usarlo. Puede haber razones sutiles para que el componente de optimización de consultas no use un índice, en especial un índice desagrupado.
- ✓ El número de alternativas es grande. Incluso si los índices que están en las combinaciones de las columnas son ignorados, el número teórico de alternativas es exponencial con el número de columnas (2^{NC} en donde NC es el número de columnas). Aunque muchas de estas alternativas se pueden eliminar fácilmente, el número de alternativas prácticas todavía es muy grande.
- ✓ Las alternativas de los índices se pueden interrelacionar. Las interrelaciones pueden ser sutiles, en especial cuando la selección de índices puede mejorar el desempeño de los JOIN.

MONITOREO Y AFINAMIENTO

- Reglas de selección de índices: para reducir la mala selección de índices se pueden aplicar las siguientes reglas:
 - ✓ Regla 1: Una clave primaria es un buen candidato para un índice agrupado.
 - ✓ Regla 2: Para respaldar los JOIN, considerar los índices sobre las claves foráneas. Un índice desagrupado sobre una clave foránea es una buena idea cuando existen consultas importantes con condiciones altamente selectivas hechas sobre una tabla relacionada. Un índice agrupado es una buena opción cuando la mayoría de los enlaces usan una tabla madre con un índice agrupado sobre su clave primaria, y las consultas no tienen condiciones altamente selectivas sobre la tabla madre.
 - ✓ Regla 3: Una columna con muchos valores puede ser una buena opción para un índice desagrupado cuando se usan en condiciones de igualdad. El término muchos valores significa que la columna es casi única.
 - ✓ Regla 4: Una columna que se usa en un rango de condiciones altamente selectivas es buena candidata para convertirse en un índice desagrupado.
 - ✓ Regla 5: Una combinación de columnas usada en forma conjunta con las condiciones de una consulta puede ser candidata para convertirse en índices desagrupados cuando las condiciones del enlace regresen pocas filas, el optimizador del DBMS soporte el acceso a varios índices y las columnas sean estables. Los índices individuales deben crearse para cada columna.

MONITOREO Y AFINAMIENTO

- Reglas de selección de índices: para reducir la mala selección de índices se pueden aplicar las siguientes reglas:
 - ✓ Regla 6: Una columna que se actualiza frecuentemente no es un buen candidato para un índice.
 - ✓ Regla 7: Las tablas volátiles (con muchas inserciones y eliminaciones) no deben tener muchos índices.
 - ✓ Regla 8: Las columnas estables con pocos valores son candidatas a convertirse en índices de tipo bitmap cuando las columnas se encuentren dentro de las condiciones WHERE.
 - ✓ Regla 9: Evitar los índices con combinaciones de columnas. La mayoría de los componentes de optimización pueden usar varios índices sobre la misma tabla. Un índice que se haga sobre una combinación de columnas no es tan flexible como varios índices que se hagan para las columnas individuales de la tabla.

MONITOREO Y AFINAMIENTO

- Otras opciones que colaboran con la mejora del desempeño de las bases de datos:
 - ✓ Desnormalización.
 - ✓ Formateo de registros.
 - ✓ Procesamiento en paralelo

MONITOREO Y AFINAMIENTO

- Desnormalización

- ✓ Los diseños normalizados:

- Tienen un mejor desempeño para las actualizaciones.
- Requieren menos código para obligar a que se cumplan las restricciones de integridad.
- Soportan más índices para mejorar el desempeño de las consultas.

- ✓ La desnormalización:

- Ignorar una dependencia funcional si no conduce a anomalías significativas de las modificaciones.
- Después de combinar las tablas, la nueva tabla puede violar alguna de las formas normales como el BCNF (forma normal de Boyce-Codd, por sus siglas en inglés).
- Si bien, algunas de las técnicas de desnormalización no conducen a violaciones de alguna forma normal, hacen que el diseño sea más fácil de consultar y más difícil de actualizar.
- La desnormalización siempre debe hacerse con mucho cuidado ya que un diseño normalizado tiene importantes ventajas.

MONITOREO Y AFINAMIENTO

- Desnormalización

- ✓ Grupos de repetición

- Un grupo de repetición es un conjunto de valores asociados.
 - Las reglas de normalización obligan a que los grupos de repetición se almacenen en una tabla hija separada de su tabla madre asociada.
 - La desnormalización puede ser una alternativa factible si siempre se accede a un grupo repetido mediante su tabla madre asociada.
 - Aunque el diseño desnormalizado no viola la forma normal BCNF, es menos flexible para las actualizaciones que el diseño normalizado.
 - El diseño desnormalizado no requiere un JOIN para combinar los datos.

MONITOREO Y AFINAMIENTO

- Desnormalización

- ✓ Jerarquías de generalización

- La regla de conversión de jerarquías de generalización permite obtener muchas tablas.
 - Si las consultas necesitan combinar regularmente estas tablas separadas, es factible almacenarlas como una sola tabla.
 - Las tablas tienen relaciones 1-1, ya que representan una jerarquía de generalización.
 - Aunque el diseño desnormalizado no viola la BCNF, la tabla combinada puede desperdiciar mucho espacio debido a los valores nulos; sin embargo, el diseño desnormalizado
 - evita el uso del operador de JOIN externos (OUTER JOIN) para combinar las tablas.

MONITOREO Y AFINAMIENTO

- Desnormalización

- ✓ Códigos y significados

- Las reglas de normalización requieren que las claves foráneas se almacenen en forma aislada para representar las relaciones 1-M.
 - Si una clave foránea representa un código, generalmente el usuario necesita un nombre asociado o descripción, además del valor de la clave foránea.
 - El almacenamiento de la columna del nombre o de la descripción junto con el código viola la BCNF, pero elimina algunas operaciones de JOIN.
 - La desnormalización puede ser una opción razonable si la columna del nombre o de la descripción no cambia de forma constante.

MONITOREO Y AFINAMIENTO

- **Formateo de registros**
 - ✓ Las decisiones sobre el formateo de registros incluyen la compresión y datos derivados.
 - ✓ La compresión es un elemento importante con mayor énfasis en el almacenamiento de tipos de datos complejos, tales como audio, video e imágenes.
 - ✓ La compresión tiene sus ventajas y desventajas con respecto al esfuerzo del procesamiento de entradas-salidas.
 - ✓ La compresión reduce el número de registros físicos transferidos, pero puede requerir de un esfuerzo de procesamiento considerable para comprimir y descomprimir los datos.
 - ✓ Las decisiones sobre datos derivados involucran ventajas y desventajas entre las operaciones de consultas y actualizaciones.
 - ✓ Para efectos de consultas, el almacenamiento de datos derivados reduce la necesidad de recuperar datos requeridos para calcular los datos derivados.
 - ✓ Las actualizaciones para los datos utilizados en el cálculo requieren de actualizaciones adicionales de los datos derivados.
 - ✓ Puede ser razonable almacenar datos derivados para reducir las operaciones de JOIN.

MONITOREO Y AFINAMIENTO

- Procesamiento paralelo
 - ✓ El desempeño se puede mejorar de forma significativa al efectuar operaciones de recuperación y modificación a través del procesamiento paralelo.
 - ✓ Las recuperaciones que involucren muchos registros se pueden mejorar al leer los registros físicos en paralelo.
 - ✓ El desempeño suele mejorarse de forma significativa para las aplicaciones que trabajan por lotes con muchas operaciones de escritura y de lectura/escritura de enormes registros físicos, tales como las imágenes.
 - ✓ En los arreglos redundantes de discos independientes (RAID), el controlador RAID permite que un arreglo de discos se muestre al DBMS como un disco único muy grande. Para obtener un alto desempeño, el controlador RAID puede controlar hasta 90 discos.
 - ✓ Debido a este controlador, el almacenamiento RAID no requiere cambios que deban tomarse en cuenta para el procesamiento en paralelo al hacer la evaluación de un plan de acceso.
 - ✓ La distribución es un concepto importante del almacenamiento RAID.
 - ✓ La distribución incluye la colocación de los registros físicos en distintos discos.
 - ✓ Una distribución es un conjunto de registros físicos que pueden leerse o escribirse en paralelo,

MONITOREO Y AFINAMIENTO

- Procesamiento paralelo
 - ✓ Para utilizar el almacenamiento RAID han surgido varias arquitecturas. Las arquitecturas RAID-X soportan el procesamiento en paralelo con diferentes elementos de desempeño y confiabilidad.
 - ✓ La confiabilidad es un elemento importante ya que el tiempo entre las fallas (una medida de la confiabilidad de los discos) disminuye conforme aumenta el número de discos.
 - ✓ Para combatir las preocupaciones de la confiabilidad, las arquitecturas RAID incorporan redundancia con el uso de discos en espejo, códigos de corrección de errores y discos de repuesto.
 - ✓ Para la mayoría de estos propósitos dominan las arquitecturas RAID-1 y RAID-5.

MONITOREO Y AFINAMIENTO

- Procesamiento paralelo

- ✓ RAID-1:

- Incluye un espejo completo o arreglo redundante de discos para mejorar la confiabilidad.
 - Cada registro físico se escribe en los dos arreglos de discos en paralelo.
 - Las operaciones de lectura de consultas separadas pueden acceder al arreglo de discos en paralelo para mejorar el desempeño entre las consultas.
 - incluye la mayor sobrecarga de almacenamiento comparada con otras arquitecturas RAID.

MONITOREO Y AFINAMIENTO

- Procesamiento paralelo

- ✓ RAID-5:

- Utiliza tanto las páginas de datos como de corrección de errores (conocidas como páginas de paridad) para mejorar la confiabilidad.
- Las operaciones de lectura se pueden llevar a cabo en paralelo en las distribuciones.
- Las operaciones de escritura involucran una página de datos y una página de corrección de errores en otro disco.
- Para reducir la contención de los discos, las páginas de corrección de errores se localizan de manera aleatoria entre los discos.
- Usa el espacio de almacenamiento de forma más eficiente que RAID-1, pero puede llevar a tiempos de escritura más lentos debido a las páginas de corrección de errores.
- Por lo general, se prefiere RAID-1 para las partes altamente volátiles de una base de datos.

MONITOREO Y AFINAMIENTO

- Procesamiento paralelo

- ✓ RAID-5:

- Utiliza tanto las páginas de datos como de corrección de errores (conocidas como páginas de paridad) para mejorar la confiabilidad.
- Las operaciones de lectura se pueden llevar a cabo en paralelo en las distribuciones.
- Las operaciones de escritura involucran una página de datos y una página de corrección de errores en otro disco.
- Para reducir la contención de los discos, las páginas de corrección de errores se localizan de manera aleatoria entre los discos.
- Usa el espacio de almacenamiento de forma más eficiente que RAID-1, pero puede llevar a tiempos de escritura más lentos debido a las páginas de corrección de errores.
- Por lo general, se prefiere RAID-1 para las partes altamente volátiles de una base de datos.

MONITOREO Y AFINAMIENTO

- Procesamiento paralelo
 - ✓ Para aumentar la capacidad más allá de RAID y eliminar la dependencia de los dispositivos de almacenamiento propios del servidor, se han desarrollado las redes de áreas de almacenamiento (Storage Area Networks o SANs).
 - ✓ Una SAN es una red especializada de alta velocidad que conecta los dispositivos de almacenamiento y los servidores.
 - ✓ El objetivo de la tecnología SAN es integrar de forma sencilla distintos tipos de subsistemas de almacenamiento en un solo sistema y eliminar el potencial cuello de botella de un único servidor que controle los dispositivos de almacenamiento.
 - ✓ Muchas organizaciones grandes usan las SAN para integrar sistemas de almacenamiento de bases de datos operacionales, data warehouses, almacenamiento histórico de documentos y sistemas de archivos tradicionales.

MONITOREO Y AFINAMIENTO

- Otras consideraciones para mejorar el desempeño de las bases de datos
 - ✓ Para mejorar el desempeño bajo el tipo de procesamiento transaccional se puede agregar capacidad de cómputo (más procesadores y más rápidos, memoria y disco duro) y evaluar las ventajas y desventajas en el diseño transaccional.
 - ✓ Para mejorar el desempeño de los data warehouses se puede agregar capacidad de cómputo y diseñar tablas nuevas con datos derivados.
 - ✓ Para mejorar el desempeño bajo el tipo de procesamiento de bases de datos distribuidas se puede colocar el procesamiento y los datos en varias computadoras. Los datos se pueden colocar dividiendo una tabla de forma vertical (subconjunto de columnas) y horizontal (subconjunto de filas) para ubicar los datos cerca de donde se usen. La mayoría de los DBMS proporcionan guías y herramientas para monitorear y controlar la fragmentación.

MÓDULO XI

PROCESAMIENTO DE CONSULTA DISTRIBUIDA PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

Marcela Russo
Laboratorio IV

PROCESAMIENTO DE CONSULTA DISTRIBUIDA IMPLICA

- Optimización local (intra-sitio) y optimización global (inter-sitio - decisiones de movimiento de datos y de selección de sitio). Por tal motivo existen muchos más planes de acceso posibles para una consulta distribuida que para su correspondiente consulta centralizada.
- Las complejidades también existen en las bases de datos paralelas con arquitecturas nada compartidas y la diferencia principal es la comunicación en redes mucho más rápida y más confiable que las que son usadas en el procesamiento de bases de datos paralelas.
- Persigue múltiples objetivos de optimización
- En un entorno centralizado, minimizar el uso de recursos (entrada-salida y procesamiento) es consistente con la reducción del tiempo de respuesta.
- En un entorno distribuido, minimizar los recursos puede entrar en conflicto con la reducción del tiempo de respuesta, debido a las oportunidades de procesamiento paralelo.
- El procesamiento paralelo puede reducir el tiempo de respuesta para aumentar la cantidad global de recursos consumidos (entrada-salida, procesamiento y comunicación).
- Además, la ponderación de los costos de comunicación contra los costos locales (entrada-salida y procesamiento) depende de las características de la red.

PROCESAMIENTO DE CONSULTA DISTRIBUIDA IMPLICA

- Para redes de área amplia, los costos de comunicación pueden dominar sobre los costos locales.
- Para redes de área local, los costos de comunicación están más equitativamente ponderados con los costos locales.
- La elección de un mal plan de acceso puede conducir a un rendimiento extremadamente pobre.
- Los planes de acceso distribuidos muchas veces se ajustan para las condiciones del sitio.
- Si un sitio no está disponible o está sobrecargado, un plan de acceso distribuido deberá elegir dinámicamente otro sitio. Por tal motivo, parte del proceso de optimización puede requerir que se ejecute dinámicamente (durante el tiempo de ejecución) en vez de estáticamente (durante el tiempo de compilación).
- Para determinar el mejor plan de acceso se muchas veces es necesario realizar análisis adicionales de los costos de procesamiento locales.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- El procesamiento de transacción distribuida se ajusta a los principios de procesamiento de cualquier transacción.
- Las transacciones obedecen las propiedades ACID.
- Los DBMS distribuidos proporcionan concurrencia y transparencia de recuperación.
- Los sitios que operan independientemente deben estar coordinados.
- El entorno distribuido hace más difícil la implementación de los principios de optimización, debido a que en la red de comunicación se presentan nuevos tipos de falla y es necesario aplicar nuevos protocolos

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **CONTROL DE CONCURRENCIA DISTRIBUIDA:**

- ✓ Los sitios locales deben coordinarse a través de mensajes sobre una red de comunicación, derivando en más gastos generales que el control de concurrencia centralizada.
- ✓ El esquema más simple implica coordinación centralizada.
- ✓ Al comienzo de una transacción, se elige el sitio de coordinación y la transacción se divide en subtransacciones realizadas en otros sitios.
- ✓ Cada sitio que alberga una transacción emite solicitudes de candado y liberación al sitio coordinador mediante el uso de reglas normales de candado en dos fases.
- ✓ La coordinación centralizada implica la menor cantidad de mensajes y la detección más simple de bloqueo.
- ✓ La confiabilidad en un coordinador centralizado puede hacer que el procesamiento de transacción sea menos confiable.
- ✓ Para aliviar la confianza en un sitio centralizado, el administrador de candados puede distribuirse entre sitios.
- ✓ El precio por mayor confiabilidad es más gastos generales por mensaje y detección de bloqueo más compleja.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- CONTROL DE CONCURRENCIA DISTRIBUIDA:

- ✓ El número de mensajes puede duplicarse en el esquema de coordinación distribuida en comparación con el esquema de coordinación centralizada.
- ✓ Los datos copiados implican un problema tanto en la coordinación centralizada como en la distribuida.
- ✓ Actualizar los datos copiados implica gastos generales adicionales debido a que se debe obtener un candado de escritura en todas las copias antes de que cualquier copia se actualice.
- ✓ Obtener candados de escritura en múltiples copias puede causar retardos e incluso regresiones si una copia no está disponible.
- ✓ Para reducir la saturación con múltiples copias de candado puede usarse el protocolo de copia primaria.
- ✓ En el protocolo de copia primaria, una copia de cada fragmento copiado se designa como la copia primaria, mientras que las otras copias son secundarias.
- ✓ Los candados de escritura son necesarios sólo para la copia primaria.

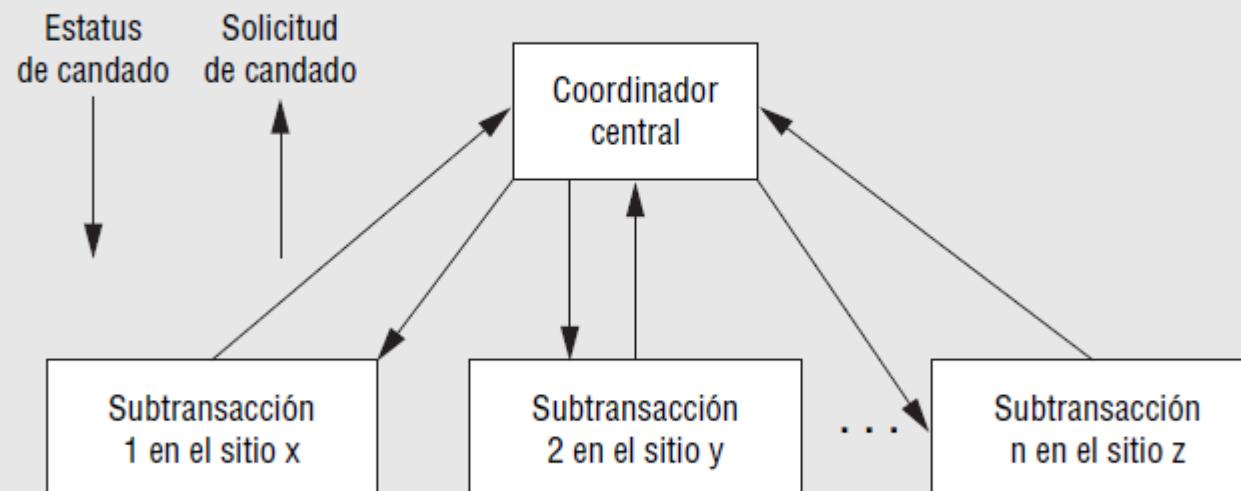
PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **CONTROL DE CONCURRENCIA DISTRIBUIDA:**

- ✓ Después de que una transacción actualiza la copia primaria, las actualizaciones se propagan a las copias secundarias.
- ✓ Sin embargo, las copias secundarias pueden no actualizarse sino hasta después de comprometer la transacción.
- ✓ El protocolo de copia primaria proporciona rendimiento mejorado pero al costo de copias secundarias no concurrentes.
- ✓ Debido a que la reducción de gastos generales con frecuencia es más importante que las copias secundarias actuales, muchos DBMS distribuidos usan el protocolo de copia primaria.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

Control de concurrencia centralizada



Mannino tabla 17.17 página 633

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **PROTOCOLO DE COPIA PRIMARIA:**

- ✓ Es un protocolo para el control de concurrencia de transacciones distribuidas.
- ✓ Cada fragmento copiado está designado como la copia primaria o como la copia secundaria.
- ✓ Durante el procesamiento de transacción distribuida sólo la copia primaria tiene garantía de ser la actual al final de una transacción.
- ✓ Las actualizaciones pueden propagarse a copias secundarias después del fin de la transacción.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **PROCESAMIENTO DE COMPROMISO DISTRIBUIDO:**

- ✓ Los DBMS distribuidos deben enfrentar fallas de comunicación en vínculos y sitios.
- ✓ La detección de fallas implica coordinación entre sitios.
- ✓ Si un vínculo o sitio falla, debe abortarse cualquier transacción que involucre al sitio y ese sitio debe evitarse para futuras transacciones hasta que la falla se resuelva.
- ✓ Las fallas pueden ser más complejas que sólo un sitio sencillo o un vínculo de comunicación.
- ✓ Muchos sitios y vínculos pueden fallar simultáneamente dejando una red particionada.
- ✓ En una red particionada las diferentes particiones (colecciones de sitios) no pueden comunicarse, aunque se comuniquen los sitios en la misma partición.
- ✓ El administrador de transacción debe asegurar que diferentes partes de una red particionada actúan al unísono.
- ✓ No es posible que los sitios en una partición decidan comprometer una transacción, sino que los sitios en otra partición deben decidir no comprometer una transacción.
- ✓ Todos los sitios deben comprometer o abortar.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **PROTOCOLO DE COMPROMISO EN DOS FASES:**

- ✓ Es el protocolo más utilizado para procesamiento de compromiso distribuido.
- ✓ Para cada transacción, se elige un sitio como el coordinador y la transacción se divide en subtransacciones realizadas en otros sitios participantes.
- ✓ El coordinador y los sitios participantes interactúan en una fase de votación y una fase de decisión.
- ✓ Al final de ambas fases, cada sitio participante actúa al unísono para comprometer o abortar su subtransacción.
- ✓ Las fases de votación y decisión requieren acciones tanto del coordinador como de los sitios participantes.
- ✓ En la fase de decisión, el coordinador envía un mensaje a cada participante preguntando si está listo para comprometer.
- ✓ Antes de responder, cada participante obliga todas las actualizaciones a disco cuando termina el trabajo de la transacción local.
- ✓ Si no ocurre una falla, el participante escribe un registro de READY-COMMIT y envía un voto READY al coordinador.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **PROTOCOLO DE COMPROMISO EN DOS FASES:**

- ✓ En este punto, el participante tiene un estatus incierto porque el coordinador puede solicitar más tarde al participante abortar.
- ✓ La fase de decisión comienza cuando el coordinador recibe votos de cada participante u ocurre un vencimiento.
- ✓ Si ocurre un vencimiento o al menos un participante envía un voto de ABORT, el coordinador aborta toda la transacción mediante el envío de mensajes ABORT a cada participante, cada participante realiza una regresión a sus cambios.
- ✓ Si todos los participantes votan READY, el coordinador escribe el registro de compromiso global y pide a cada participante comprometer sus subtransacciones.
- ✓ Cada participante escribe un registro de compromiso, libera los cierres y envía un reconocimiento al coordinador.
- ✓ Cuando el coordinador recibe el reconocimiento de todos los participantes, el coordinador escribe el registro de fin de transacción global.
- ✓ Si ocurre una falla en la fase de votación o en la de decisión, el coordinador envía un mensaje ABORT a todos los sitios participantes.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **PROTOCOLO DE COMPROMISO EN DOS FASES:**

- ✓ La falla durante la recuperación y vencimientos, pueden complicar el protocolo.
- ✓ Las modificaciones al protocolo básico pueden reducir el número de mensajes necesarios para forzar al protocolo.
- ✓ El protocolo de compromiso en dos fases puede usar un coordinador centralizado o uno distribuido.
- ✓ Las negociaciones son similares a la coordinación centralizada frente a la distribuida para el control de concurrencias.
- ✓ La coordinación centralizada es más simple que la coordinación distribuida, pero puede ser menos confiable.
- ✓ El protocolo de compromiso en dos fases no maneja ningún tipo concebible de falla.
- ✓ El protocolo de compromiso en dos fases puede tener un funcionamiento inadecuado si los registros de log se pierden.

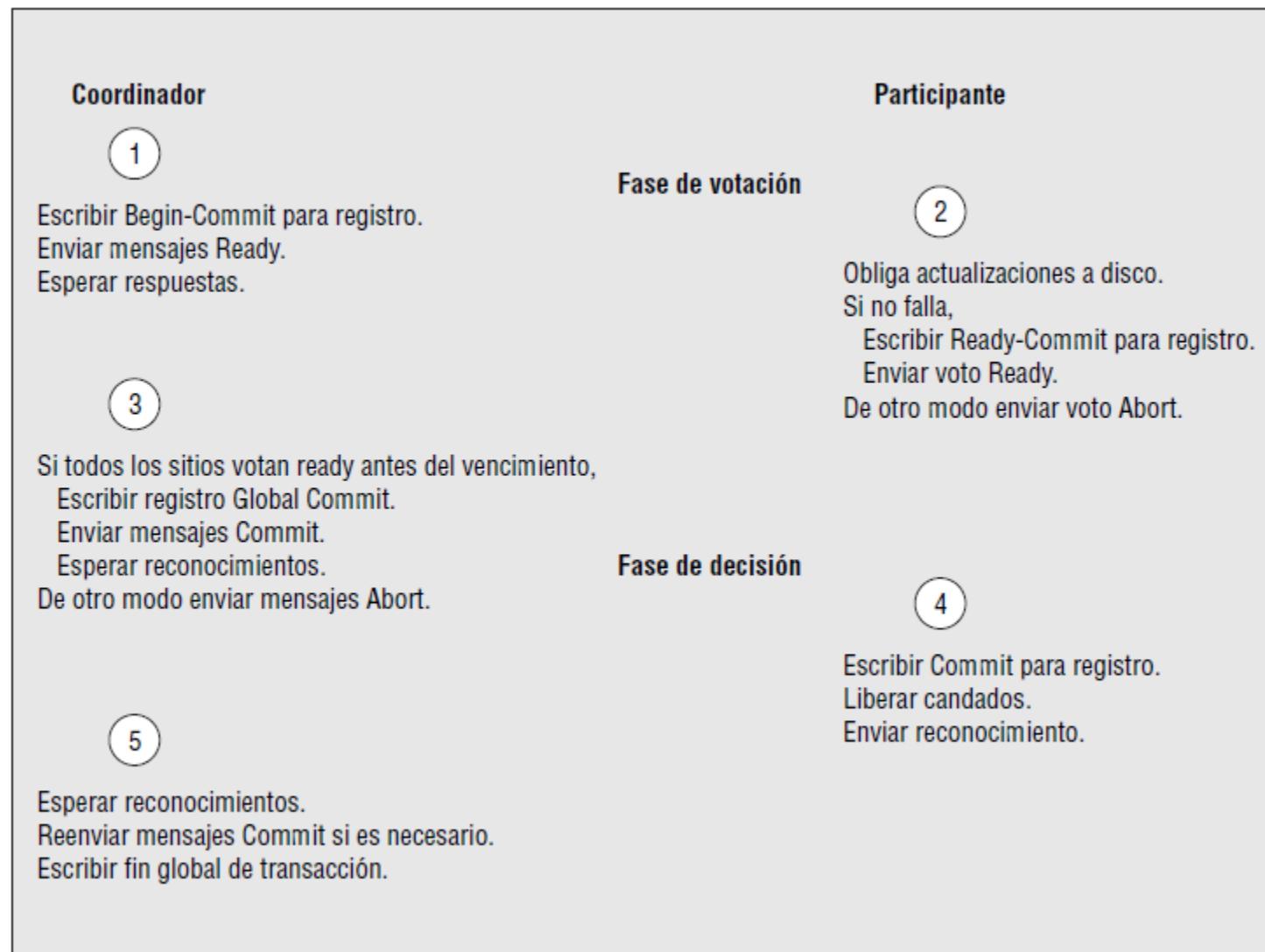
PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

- **PROTOCOLO DE COMPROMISO EN DOS FASES:**

- ✓ No hay protocolo que garantice que todos los sitios actúan al unísono para comprometer o abortar al presentarse fallas arbitrarias.
- ✓ Dado que el protocolo de compromiso en dos fases maneja eficientemente tipos comunes de fallas, se usa ampliamente en procesamiento de transacción distribuida.

PROCESAMIENTO DE TRANSACCIÓN DISTRIBUIDA

Procesamiento de compromiso en dos fases para coordinador y participantes



MÓDULO XII

**CUADRO RESUMEN SOBRE BASE DE DATOS:
OPEN SOURCE
REPLICACIÓN DE DATOS EN BASES DE DATOS DISTRIBUIDAS
MINERIA DE DATOS**

Marcela Russo
Laboratorio IV

➤ OPEN SOURCE

- Cualquier persona dispone del código fuente de la base de datos.
- El código fuente es privado. Suelen ser gratuitas o tener algún costo accesible.
- La comunidad al tener acceso al código fuente puede proporcionar modificaciones útiles y detectar vulnerabilidades o errores para luego arreglarlos.
- No se suele garantizar un soporte técnico estable para este tipo de base de datos.
- El único tipo de soporte que tienen los usuarios es la propia comunidad, por lo que depende completamente de la relevancia que tenga esa base de datos en la actualidad.
- La instalación y actualizaciones son administradas por el usuario.

➤ REPLICACIÓN DE DATOS EN BASES DE DATOS DISTRIBUIDAS

- La replicación: copia y mantiene los objetos de las bases de datos en las múltiples bases de datos que levantan un sistema distribuido.
- La replicación puede mejorar el funcionamiento y proteger la disponibilidad de las aplicaciones.
- La replicación tiene diferentes niveles:
 - ✓ Replicación básica: las réplicas de tablas se gestionan para accesos de sólo lectura. Para modificaciones, se deberá acceder a los datos del sitio primario.
 - ✓ Replicación avanzada (simétrica): amplían las capacidades básicas de sólo- lectura de la replicación, permitiendo que las aplicaciones hagan actualizaciones a las réplicas de las tablas, a través de un sistema replicado de la base de datos. Con la replicación avanzada, los datos pueden proveer lectura y acceso a actualizaciones a los datos de las tablas

➤ MINERIA DE DATOS

- La demanda de tecnología de bases de datos de objetos es impulsada por la necesidad de almacenar grandes cantidades de datos complejos y la integración de datos complejos con datos simples.
- Es la exploración y análisis por medios automáticos o semiautomáticos, de datos para descubrir patrones y reglas.
- Las 3 disciplinas científicas que intervienen:
 - ✓ Estadística
 - ✓ Inteligencia artificial
 - ✓ Aprendizaje automático
- Explora y analiza grandes cantidades de información para detectar patrones y tendencias significativas.
- Contribuye a la toma de decisiones tácticas y estratégicas.

MÓDULO XIII

INTRODUCCIÓN A LA EXPLORACIÓN DE DATOS: CONCEPTOS FUNDAMENTOS CARACTERÍSTICAS

Marcela Russo
Laboratorio IV

CONCEPTOS

- **Analizar los datos es el primer paso para explorar y visualizar datos que permite descubrir conocimientos desde el inicio o bien identificar patrones para profundizarlos.**
- **La exploración de datos facilita la toma de las mejores decisiones sobre dónde profundizar en los datos y tener una comprensión amplia de la organización.**
- **Ayuda a la familiarización con los datos, descubrir patrones y ahondar con preguntas reflexivas que estimularán a un análisis más profundo y valioso.**
- **El tipo de datos que se usa para fines de apoyo a la toma de decisiones es conceptualmente distinto al que se utiliza en las bases de datos de procesamiento de transacciones.**
- **Las bases de datos que pueden almacenar dichos datos son diferentes.**
- **Las arquitecturas de cómputo que pueden procesar los datos son diferentes.**

FUNDAMENTOS

- El procesamiento de apoyo a las decisiones ayuda a la alta gerencia a darle dirección a una organización a mediano y largo plazos.
- El apoyo a las decisiones necesita una visión amplia que integra procesos empresariales.
- Utilizar una base de datos común para procesamiento operacional y para procesamiento de apoyo a las decisiones puede degradar en forma considerable el desempeño y hacer que sea difícil resumir la actividad en todos los procesos empresariales.

CARACTERISTICAS DE DATA WAREHOUSE

- El data warehouse se refiere a un depósito de datos central donde se integran, depuran y estandarizan los datos de bases de datos operacionales y otras fuentes para apoyar la toma de decisiones.
- Las actividades de transformación (depuración, integración y estandarización) son esenciales para lograr beneficios.
- Los beneficios tangibles de un data warehouse pueden incluir mayores ingresos y menores gastos permitidos por el análisis empresarial, que no eran posibles antes de la utilización de los data warehouse.
- El procesamiento de apoyo a las decisiones utiliza data warehouse con datos históricos tanto en el plano individual como en el resumido:
 - ✓ Los datos del plano individual proporcionan flexibilidad para responder a una amplia variedad de necesidades de apoyo a las decisiones.
 - ✓ Los datos resumidos dan respuesta rápida a consultas repetitivas.
- Los proyectos de data warehouse se emprenden generalmente por razones competitivas con el fin de lograr una ventaja estratégica o seguir siendo competitivos.

CARACTERISTICAS DE DATA WAREHOUSE

- Los requerimientos de procesamiento de las aplicaciones de apoyo a las decisiones han llevado a cuatro características distintivas para los data warehouse:
 - ✓ Orientado a los sujetos: Un data warehouse se organiza en torno de los principales sujetos o entidades, como clientes, pedidos y productos. Esta orientación a los sujetos contrasta con la mayor orientación hacia los procesos en el procesamiento de transacciones.
 - ✓ Integrado: Los datos operativos de múltiples bases de datos y fuentes de datos externas se integran en un data warehouse para proporcionar una base de datos individual y unificada para el apoyo a las decisiones. La consolidación de los datos requiere convenciones de nomenclatura consistente, formatos de datos uniformes y escalas de medición comparables en las bases de datos y en las fuentes de datos externas.

CARACTERISTICAS DE DATA WAREHOUSE

- ✓ Variante de tiempo: Los data warehouse usan marcas de tiempo para representar datos históricos. La dimensión del tiempo es crítica para identificar tendencias, pronosticar operaciones futuras y establecer objetivos de operación. Los data warehouse esencialmente consisten en una extensa serie de fotografías instantáneas, cada una de las cuales representa datos operativos capturados en un momento en el tiempo.
- ✓ No volátil: Los datos nuevos en un data warehouse se anexan, en vez de reemplazarse, de modo que se preservan los datos históricos. El acto de anexar datos nuevos se conoce como refrescar el data warehouse. La falta de operaciones de actualización y eliminación garantiza que un data warehouse no contiene anomalías de actualización o eliminación. Los datos de las transacciones se transfieren a un data warehouse sólo cuando se ha completado la mayoría de las actividades de actualización.
- Los data warehouse mejoran la calidad de la toma de decisiones al consolidar y agregar datos de transacciones.
- Es posible aumentar el valor de un data warehouse si se pueden descubrir patrones ocultos en los datos.
- El modelo multidimensional de datos soporta la representación y operación de datos especialmente diseñados para el procesamiento de apoyo a las decisiones en los data warehouse

MINERÍA DE DATOS

- Se refiere al proceso de descubrimiento de patrones implícitos en datos almacenados en un data warehouse y la utilización de esos patrones como una ventaja empresarial.
- Facilita la capacidad de detectar, entender y pronosticar patrones.
- La aplicación más común de las técnicas de minería de datos es el marketing de objetivo.
- Está considerada como un anexo de un data warehouse maduro.
- Necesita datos más detallados que los que proporcionan los data warehouse tradicionales.
- Los volúmenes de datos y su dimensionalidad pueden ser mucho mayores en el caso de las técnicas de minería de datos que en los de las otras herramientas de análisis del data warehouse.
- Las técnicas de minería de datos se enriquecen con datos depurados de transacciones altamente dimensionales.
- La minería de datos requiere una serie de herramientas que se extienden más allá de las herramientas de análisis estadístico tradicional.
- Las herramientas de análisis estadístico tradicional no son adecuadas para datos muy dimensionales con una mezcla de datos numéricos y categóricos.

MINERÍA DE DATOS

- La minería de datos incluye generalmente los siguientes tipos de herramientas:
 - ✓ Herramientas de acceso a datos para extraer y muestrear datos de transacciones de acuerdo con criterios complejos para grandes bases de datos.
 - ✓ Herramientas de visualización de datos que permiten que una persona que toma decisiones adquiera una comprensión más profunda e intuitiva de los datos.
 - ✓ Una rica colección de modelos para agrupar, pronosticar y determinar reglas de asociación de grandes cantidades de datos. Los modelos implican redes neuronales, algoritmos genéticos, inducción en árboles de decisiones, algoritmos para descubrir reglas, redes de probabilidad y otras tecnologías de sistema experto.
 - ✓ Una arquitectura que proporciona optimización, procesamiento cliente-servidor y consultas paralelas para escalar a grandes cantidades de datos.