**U1: ADMINISTRACIÓN DE BD**

**Conceptos básicos**

* SI: es un conjunto de herramientas a través de las que se generan y gestionan datos y diferentes actores interactúan con él a través de distintos perfiles. Ese conjunto de datos, que forman el núcleo del SI, deben ser almacenados en un repositorio confiable (BD) capaz de resguardarlos asegurando la integridad y la seguridad de los mismos.
* BD: SW capaz de almacenar en forma confiable y segura la información, manteniéndola disponible para que pueda ser recuperada cuando lo usuarios del sistema lo soliciten. Es una colección de archivos, datos, información; ordenada, organizada y relacionada con la finalidad de permitir el manejo de la información para su procesamiento. Cada uno de los archivos representan una colección de registros y cada registro está compuesto de una colección de campos. Cada uno de los campos de cada registro permite llevar información de alguna característica o atributo de alguna entidad del mundo real.
* DBMS: es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las BD. Se compone de un Lenguaje de Definición de Datos (DDL: Data Definition Languaje), de un Lenguaje de Manipulación de Datos (DML: Data Manipulation Languaje), y de un Lenguaje de Consulta (SQL: Structured Query Languaje).
* DBA: es el responsable del control general del sistema a nivel técnico y de los permisos que poseen los usuarios de la BD para asegurar el correcto funcionamiento y optimizar el acceso a los datos.

**Implementación del Esquema Conceptual**

La implantación de este esquema es responsabilidad del DBA. Representa la visión organizacional de la BD que se obtiene al integrar los requerimientos de todos los usuarios en una empresa; y es totalmente independiente de las estructuras físicas de almacenamiento y de la representación final de los datos que los usuarios manejan.

Describe la estructura lógica global de la BD mediante un modelo abstracto de datos comprensible por el DBMS. Se definen la descripción de atributos, de entidades, las conexiones y las restricciones de integridad asociadas a la semántica (significado). Describe qué datos son almacenados realmente en la BD y las relaciones que existen entre los mismos. El nivel conceptual de abstracción lo usan los DBA, quienes deben decidir qué información se va a guardar en la BD.

Partes:

* + *Definición de datos*: se describen el tipo de datos y la longitud de campo de todos los elementos direccionables y precisión de la información que será guardada en la BD. Los elementos por definir incluyen artículos elementales (atributos), totales de datos y registros conceptuales (entidades).
  + *Relaciones entre datos*: para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples para obtener información compuesta y procesos complejos.

En este nivel, la BD aparece como una colección de registros lógicos o conceptuales. La transformación de registros conceptuales a registros físicos para el almacenamiento se lleva a cabo por el sistema y es transparente al usuario.

**Implementación del Esquema Interno o Físico**

Es el nivel más bajo de abstracción. El propósito es describir cómo se va a implementar físicamente en memoria el esquema lógico obtenido en la fase anterior. Se debe definir:

* Un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas
* Las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir un acceso óptimo y seguro.
* Un modelo de seguridad del sistema.

Entre el diseño físico y el lógico hay una retroalimentación, ya que alguna de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

Se debe especificar:

* Estrategias de almacenamiento: asignación de espacio de almacenamiento para el conjunto de datos.
* Estrategias de emplazamiento de los datos: para optimizar los recursos a la hora de exportar la BD (tiempo de respuesta, disco, memoria, etc.).
* Caminos de acceso: se incluye la especificación de claves como la de índices y punteros.

**Implementación de los Esquemas Externos**

Es el nivel más alto de abstracción, es lo que el usuario final puede visualizar del sistema terminado, describe sólo una parte de la BD al usuario acreditado para verla. El sistema puede proporcionar muchas visiones para la misma BD.

Vistas: son una especie de tablas virtuales; es decir no existen físicamente, sino que se forman mediante la selección y/o filtrado de los componentes de otras tablas. Una vista puede ser definida en base a una vista previa, por lo que puede crearse dependencia entre las vistas. Por lo tanto, se suprimirá automáticamente la vista dependiente si se suprime la vista original.

El objetivo de las vistas es facilitar al usuario la percepción que éste tiene de la BD así como el trabajo que van a desarrollar sobre ésta.

* *Perspectivas directas*: proporcionarse diversos modelos de información basados en los mismos datos, enfocándolos hacia distintos usuarios con necesidades específicas. Mostrar la información desde distintos ángulos ayuda a crear ambientes de trabajo y operación acordes a los objetivos de una empresa.
* *Transparencias en las modificaciones*: El usuario final no se verá afectado por el diseño o alteraciones que se realicen en el esquema conceptual de la BD.
* *Seguridad*: Las vistas proporcionan de manera natural un medio para ocultar y proteger datos, dado que solo se presenta al usuario una selección de los atributos existentes.

**Diccionario de datos o catálogo**

Es una herramienta que facilita el control y manejo de la información acerca de datos en el diseño, implementación, operación y expansión de fases de una BD**.**

Un DD es una BD misma que contiene mapas guías de una BD con información acerca de los datos. Contiene:

* Información relacionada con los diferentes tipos de registros (tablas)
* Estadísticas (cantidad de registros, índices, alias, etc.)
* Conjunto de estados (controles)
* Privilegios de los usuarios
* Procedimientos almacenados
* Características lógicas de los sitios donde se almacenan los datos del sistema (nombre, descripción, alias, contenido y organización)
* Detalles y descripciones de todos los elementos que forman parte del flujo de datos en todo el sistema (flujos de datos, almacenes de datos y procesos)

Un DD identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

Los DD de los DBMS no son iguales, aunque mantienen los mismos lineamientos o las mismas características.

* Permite mantener la historia.
* Sirve como documentación de cómo fue la BD.
* Facilita el análisis de la performance de la BD (tablas más accedidas, de más crecimiento, más actualizadas o modificadas).
* Sirve para hacer la recuperación de una BD que entra en problemas.
* Sirve para encontrar problemas, errores (ej.: tablas corruptas).

**Funciones del DBA**

* Definir el esquema conceptual de la BD.
* Definir el esquema interno de la BD.
* Conocer a la perfección el lenguaje que utiliza el sistema para comunicarse con la BD cuando un usuario necesita agregar, modificar o extraer información en el sistema, denominado SQL.
* Asegurar la performance de la BD: realizar todas las tareas de mantenimiento de HW y de SW para que la BD responda de manera eficiente a las diferentes actualizaciones, configuraciones, etc.
* Definir las restricciones de seguridad e integridad: políticas de acceso a la información.
* Definir las políticas de backup y recuperación ante fallos de HW y SW.
* Administrar los datos que se van a guardar en la BD.
* Establecer el enlace con los usuarios (desarrolladores).
* Responder a los requerimientos cambiantes: el rol del DBA es continuo ya que se encuentra presente desde el desarrollo hasta la implementación y durante su posterior uso. Debe monitorear constantemente el funcionamiento de las bases, realizando tareas preventivas y correctivas.
* El rol del DBA excede el alcance de un sistema, por lo que deberá tener una visión integral de todos los módulos que integran un único sistema de información.

**DBMS**

Sirve como interface entre la BD física y las peticiones del usuario. Interpreta las peticiones de E/S y las manda al SO para la transferencia de datos entre la MS y la MP. Es el corazón de la BD ya que se encarga del control total de los posibles aspectos que la puedan afectar. Su ventaja es que devuelve tanto los datos como la estructura.

Funciones del DBMS

* Definición de datos y manipulación de datos
* Optimización y ejecución
* Seguridad e integridad de datos
* Recuperación de datos
* Diccionario de datos
* Concurrencia

**Herramientas de un DBMS**

Afinación de la BD (tunning): Ajustes y cambios en la BD para optimizarla y mejorar su performance. No es un proceso inicial, sino continuo. El responsable debería tener toda la libertad necesaria para efectuar los cambios que crea convenientes. Los manejadores actuales de BD ya incorporan medios para la afinación automática (comandos, etc.).

Requisitos de una correcta afinación:

* + - Independencia física de los datos: modificar el esquema interno sin alterar el esquema conceptual ni los programas de aplicación.
    - Medios: para supervisar automáticamente el uso de la BD con el fin de que puedan hacerse los ajustes necesarios.

Medición del desempeño: Es responsabilidad del DBA organizar el sistema de modo que se obtenga el desempeño que sea mejor para la empresa, y realizar los ajustes apropiados cuando cambien los requerimientos. Es necesario reorganizar la BD en forma periódica (descargarla y volverla a cargar) con el fin de garantizar que los niveles de desempeño sigan siendo aceptables. Se realizan pruebas para comparar los datos obtenidos del desempeño con aquellos datos esperados.

Reorganización física y lógica: Consiste en leer el archivo en forma en que se utilizaría al realizar el procesamiento serial y escribir los registros nuevos y viejos en el archivo nuevo, dejando fuera todos los registros marcados como eliminados lógicamente; y se crearán nuevos índices con base a los nuevos valores. La frecuencia de reorganización depende de la actividad de inserción dentro del archivo; y se debe realizar antes de que el archivo esté realmente lleno para evitar problemas en tiempos de mucha actividad.

Las reorganizaciones físicas son necesarias para mejorar el rendimiento, añadir una nueva estructura de acceso, agilizar las operaciones de obtención y actualización, disminuir los tiempos de respuesta, minimizar el espacio de almacenamiento y optimizar el consumo de recursos. Las reorganizaciones lógicas pueden modificar el esquema conceptual, pero no alterar el esquema externo ni los programas de aplicación; puede ser un orden de visualización en las vistas.

**Evolución de los Sistemas de Administración de BD o DBMS**

La evolución de los sistemas de información y el crecimiento no planeado de la información dentro de las organizaciones, ha traído dispersión de los datos en sitios local o geográficamente dispersos. La necesidad de integrar y compartir dicha información implica el nacimiento de una nueva tecnología capaz de conformar de manera consistente la información de las organizaciones.

BD Centralizadas → BD Clientes-servidor → BD Distribuidas.

Centralizadas: Los equipos están conectados a un servidor centralizado, donde está almacenada la BD. La interacción con el usuario y la aplicación residen en el cliente, siendo el servidor del depositario de los datos. En un sistema centralizado, la caída o desconexión del servidor, provoca la caída total del sistema.

Cliente-Servidor: Es la integración distribuida en un sistema de red, con los recursos, medios y aplicaciones, que, definidos modularmente en los servidores, administran, ejecutan y atienden a las solicitudes de los clientes; todos interrelacionados física y lógicamente. Son varias computadoras que están conectadas por medio de una red.

Las computadoras que procesan los programas de aplicación se llaman clientes y la computadora que soporta el DBMS y el servicio de los datos se llama servidor. Cada computadora puede ser de diferente tipo y marca corriendo diferentes SO.

Existen dos variantes principales de esta arquitectura:

* + Cliente-Servidor de 2 capas:

Primera capa: Lógica de las aplicaciones.

Segunda capa: Administración de los recursos de datos (DBMS)

La intención es mejorar la usabilidad por medio del soporte de interfaces gráficas amigables para el usuario. Mejora la escalabilidad y la flexibilidad. Los servicios GUI se asignan exclusivamente al cliente, los del DBMS al servidor y la administración del procesamiento se distribuye en ambas capas. Es muy común que las reglas del negocio se efectúen como procedimientos y triggers almacenados en el servidor, y el procesamiento de la lógica de la aplicación en el ambiente del cliente.

Se emplean extensivamente donde el procesamiento de información no es crítico en tiempo, donde la administración y operación del sistema no es complejo y donde la carga de transacciones no es alta.

*Costos y limitaciones*: escalabilidad, interoperabilidad (la posibilidad de operar con DBMS distintos puede ser limitada) y administración y configuración de las aplicaciones (incrementa la carga de trabajo).

* División del procesamiento entre el sistema cliente y el servidor de BD.
* Sistema abierto: los clientes y los servidores no están atados por un modelo/marca/SO.
* Independencia de las aplicaciones; los usuarios pueden emplear el SW que les resulte más familiar para acceder al DBMS
* Mantenimiento de los clientes (aplicaciones y HW)
* Costo en aprendizaje y en personal (heterogeneidad de los componentes)
* Configuración de los distintos componentes del sistema y hacer que funcionen coordinadamente es complejo.
  + Cliente-Servidor de 3 capas:

Intenta cubrir las falencias del modelo anterior. Mejora la escalabilidad, la flexibilidad y el mantenimiento de los clientes (ubicando gran parte del procesamiento en la capa del medio, haciendo los clientes más “flacos”).

En la capa del medio se ubican la administración de los procesos y las reglas de negocio (procedimientos y triggers del modelo de dos capas). Administra las funciones de administración de la BD (servicios de datos y archivos) sin usar lenguajes propios del DBMS. Se ocupa de la administración de los procesos, desarrollo y verificación de procesos, servicios y recursos de los procesos. Hace un control de las transacciones y pone en “cola” las mismas para asegurar que las transacciones lleguen a su fin de manera segura.

El servidor de la capa del medio (servidor de aplicaciones) es el que centraliza la lógica del proceso. La centralización hace que la lógica de los procesos y la administración de los cambios sea más fácil.

El “link” entre las dos aplicaciones la hace un “Web Server” que recibe los requerimientos de los usuarios y genera documentos HTML en conjunto con los servicios que le provee el resto de los servidores. Esta es la arquitectura de una aplicación Intranet/Extranet.

Uso: Se emplean en ambientes cliente/servidor distribuidos donde se necesita compartir recursos tales como DBMS heterogéneos y reglas de negocios. Esta arquitectura facilita el desarrollo porque cada capa puede ser construida y ejecutada sobre una plataforma diferente, haciendo la organización e implementación más fácil. Permite, además, que las capas se puedan desarrollar en diferentes lenguajes. También facilita la migración de las aplicaciones, ya que el sistema nuevo podría ir generándose y ejecutándose en paralelo con el nuevo hasta que el sistema se haya migrado completamente.

*Costos y limitaciones*: La construcción es compleja. Las herramientas de programación de la capa del medio no proveen aún de todas las funcionalidades deseables de un ambiente de computación distribuido. Para las aplicaciones, la separación de la lógica del cliente, y la separación entre la lógica de proceso y la de datos, no siempre es clara.

Distribuidas: La BD está dividida en fragmentos almacenados en diferentes sitios de la red. Todas las operaciones realizadas en los sitios se ven reflejadas en la BD que está almacenada en el servidor. El cliente debe conocer la topología de la red, así como la disposición y ubicación de los datos. Se delega parte de la gestión de la BD a los clientes.

Un Sistema de BD Distribuidas es un conjunto de sitios autónomos de procesamiento de datos que se encuentran a una distancia determinada unos de otros conectados por una red y cooperando entre sí.

Un DDBMS debe proveer acceso a los datos como si ellos estuvieran en la computadora local. Los datos físicos están distribuidos en computadoras diferentes, pero el sistema presenta una sola imagen al usuario. Técnicamente es muy difícil proveer una imagen simple de la BD y pocos proveedores presentan una solución completa. No se comparte MP.

* *Mayor paralelismo*: Una consulta puede ser subdividida en varias subconsultas que se ejecuten en paralelo en diferentes sitios.
* *Mayor independencia*: Cada sitio puede administrar sus propios datos de manera local y definir sus estrategias de acceso, seguridad, integridad, etc.
* *Mayor flexibilidad*: La introducción de cambios es más sencilla. (Ej.: nuevos sitios)
* *Mayor disponibilidad*: Si se produce una falla en una localidad las demás localidades puedan seguir trabajando. Además, si los datos están duplicados en otras localidades, es posible encontrar el dato que se necesita en otro lado.
* *Mayor costo/complejidad*: El desarrollo del SW es más difícil, complejo y costoso. Además, hay costo de comunicación y de almacenamiento adicional de las copias.
* *Mayor dificultad en el control*: Al tener trabajo en paralelo, y copias de la BD en diferentes sitios, es más difícil de garantizar que los algoritmos sean correctos.
* *Mayor riesgo de seguridad*: Es más vulnerable a que existan sabotajes, pérdidas y fugas de información.

**U2: SEGURIDAD E INTEGRIDAD**

**Respaldo y Recuperación**

El respaldo y recuperación consiste en contar con un mecanismo que permite la fácil recuperación de los datos en caso de ocurrir fallos en el sistema de BD. Independientemente de la naturaleza de las fallas, estas pueden provocar la pérdida de la MP o MS.

El DBA debe definir y poner en práctica un plan o política de recuperación adecuado que incluya una descarga o vaciado periódico de la BD en un medio de almacenamiento de respaldo y procedimientos para volver a cargar la BD a partir del vaciado más reciente.

En un sistema de BD, recuperación significa restaurar la BD a un estado que se sabe que es correcto, después de una falla que provoca que se considere que el estado actual es incorrecto. Podemos tener la seguridad de que la BD es recuperable si aseguramos que cualquier parte de la información que contiene puede ser reconstruida a partir de otra información que se encuentra almacenada redundantemente en algún lugar del sistema.

Pasos para recuperar la información

* *Detectar el error*: Es posible distinguir una variedad de puntos de entrada en el proceso de recuperación. Se considerarán fallas de sistemas detectadas por falta de acción del sistema o por verificaciones irrecuperables de redundancia y salida incorrecta observada por un usuario.
* *Determinar la fuente del error*: para decidir cuál es la mejor acción correctora. Determinar el tiempo y la causa del error, además de las áreas del archivo de datos que sean sospechosas y cuál fue la transacción que no se concluyó.
* *Ubicar errores secundarios*: Cuando se ha detectado un error que provocó una modificación inadecuada a un archivo, un rastreo a través de las listas de actividad encontrará aquellas transacciones que emplearon el bloque correcto. Entonces es posible volver a introducir automáticamente el bloque correcto de las transacciones afectadas y producir resultados correctos. Si se actualizaron bloques mediante transacciones que leyeron bloques incorrectos antes de existir es necesario restaurar aún más el archivo.
* *Aplicar correcciones*: si la extensión del daño es limitada, puede utilizarse un proceso de volver a aplicar. Las porciones dañadas del archivo se restauran aplicando primero aquellas imágenes anteriores a los bloques en error reemplazando después de las transacciones incompletas.

**Backup**

Es una copia de información de la BD. El objetivo de estas copias es que puedan utilizarse para restaurar la BD después de una eventual pérdida de datos. Fundamentalmente son útiles para recuperarse de un fallo informático o recuperar una pequeña cantidad de archivos que pueden haberse eliminado accidentalmente o corrompido.

Tipos de Backup de BD

*Backup Full (Completo)*: Guarda todos los archivos que sean especificados al tiempo de ejecutarse el respaldo. En el caso que se trate de una BD, realiza el respaldo completo: estructura, datos, procedimientos, triggers, etc.

* Únicamente es posible recuperar toda la información.
* Tiempo de ejecución.

*Backup Diferencial*: Se guardan todos los cambios realizados desde el ultimo Backup Completo.

* Sólo se requiere del último respaldo completo y del último respaldo diferencial.
* Velocidad.
* Incrementa su tamaño y tiempo cuanto más alejado se ejecute un respaldo full.

*Backup Incremental (Logs)*: Se guardan las transacciones.

* Permite que la BD vuelva a un estado consistente.
* Velocidad.

Para una recuperación primero se ejecuta el último full, luego el último diferencial y luego todos los incrementales en orden desde el último diferencial. Una política de backups completa tiene los 3 niveles en su debida periodicidad. El tiempo de pérdida de datos debe quedar claro con la empresa, ya que debe informar la tolerancia de pérdida. El tiempo entre cada incremental depende del tiempo de pérdida tolerado.

**Duplicado de Información en línea**

RAID (Redundant Array of Inexpensive Disks): conjunto de 2 o más discos duros que operan como grupo y logran ofrecer una forma más avanzada de respaldo ya que: es posible mantener copias en línea (redundancia), agiliza las operaciones del sistema y el sistema es capaz de recuperar información sin intervención de un administrador.

Configuraciones de tipo RAID:

* *RAID-0*: cada archivo es dividido y sus fracciones son colocadas en diferentes discos. Se agiliza el proceso de lectura de archivos, pero no proporciona algún tipo de respaldo.
* *RAID-1 (Mirroring)*: otorga cierto nivel de respaldo. Cada archivo se copiará integro a 2 discos (en línea). Agiliza la lectura de archivos.
* *RAID-3*: divide la información de todos los archivos en varios discos, pero ofrece cierto nivel de respaldo. Opera con un disco de paridad que guarda fracciones de los archivos necesarias para recuperar toda su información (es posible reproducir un archivo que se perdió). La desventaja es que el disco de paridad es un punto crítico en el sistema si falla.
* *RAID-5*: distribuye todos los archivos en un grupo de discos y también guarda la información de paridad en todos los discos del sistema. Esta configuración suele ser usada en sistemas que requieren un alto nivel de disponibilidad.

**Implementación Backup**

Horarios: Programar un horario de ejecución de las copias de seguridad aumenta considerablemente su efectividad y nivel de optimización.

Autentificación: Sobre el curso de operaciones regulares, las cuentas de usuario y/o los agentes del sistema que representan la copia de seguridad necesitan ser autentificados a cierto nivel. Utilizar un mecanismo de autentificación es una buena manera de evitar que el esquema de la copia de seguridad sea usado por actividades sin autorizar.

Cadena de confianza: Los soportes de almacenamiento portátiles son elementos físicos y deben ser gestionados sólo por personas de confianza. Establecer una cadena de confianza individual es crítico para defender la seguridad de los datos.

Validación de copias de seguridad: El proceso por el cual los dueños de los datos pueden obtener información considerando como fueron copiados esos datos. El mismo proceso es también usado para probar conformidad para los cuerpos reguladores fuera de la organización. Terrorismo, complejidad de datos, valores de datos y aumento de la dependencia sobre volúmenes de datos crecientes, todos contribuyen a una ansiedad alrededor y dependencia sobre copias de seguridad satisfactorias.

Reportando: Son útiles para monitorizar los medios usados, el estado de dispositivos, errores, coordinación de saltos y cualquier otra información sobre el proceso de copia de seguridad.

Registrando: Reportes generados por el ordenador, actividades y registros de cambio. Son útiles para entender mejor la copia de seguridad.

Verificación: Muchos programas de copia de seguridad hacen uso de sumas de verificación o hashes. Permiten a la integridad de los datos ser verificados sin hacer referencia al archivo original.

**Seguridad e Integridad**

La seguridad frecuentemente se asocia con la integridad, pero ambos conceptos son bastante diferentes. La seguridad de los datos refiere a la protección de los datos contra su revelación, su alteración o destrucción no autorizada, mientras que la integridad se refiere a la precisión o validez de esos datos.

Seguridad significa proteger los datos ante usuarios no autorizados. Integridad significa protegerlos de usuarios autorizados.

**Seguridad**

Existen muchos aspectos sobre el problema de la seguridad, entre ellos se encuentran:

* *Aspectos legales, sociales y éticos*: La persona que hace la petición ¿tiene derecho legal para conocer la información solicitada?
* *Controles físicos*: El lugar donde se encuentra la computadora ¿está bajo llave o tiene alguna otra protección?
* *Cuestiones de política*: Si se utiliza un esquema de contraseñas ¿Cómo se las mantiene en secreto? ¿Con que frecuencia son cambiadas?
* *Controles de HW*: ¿La unidad de procesamiento posee características de seguridad como claves de protección de almacenamiento o un modo de operación protegido?
* *Soporte del SO*: ¿El SO borra el contenido de la memoria principal y los archivos de disco cuando termina de utilizarlos?

**Seguridad de un DBMS**

Actualmente, los DBMS soportan uno o ambos enfoques:

* Control discrecional: un usuario específico tendrá generalmente diferentes derechos de acceso (privilegios) sobre diferentes objetos. Los esquemas discrecionales son muy flexibles. No se puede poner restricciones a los registros por lo que se usarían vistas.

Se necesita un lenguaje que soporte la definición de las restricciones de seguridad discrecionales. Generalmente se define qué está permitido en lugar de lo que no está, por lo tanto, los lenguajes generalmente soportan la definición de las autorizaciones, en vez de restricciones de seguridad.

La granularidad permitida para el control de acceso discrecional dependerá siempre de cada DBMS.

* Control obligatorio: cada objeto de datos esta etiquetado con un nivel de clasificación determinado y a cada usuario se le da un nivel de acreditación. Un objeto de datos especifico solo puede ser accedido por los usuarios que tengan el nivel de acreditación adecuado. Los esquemas obligatorios tienden a ser jerárquicos y comparativamente rígidos.

Los controles de acceso obligatorio son aplicables a las BD en la que los datos tienen una estructura bastante estática y rígida. Cada dato tiene un nivel de clasificación y cada usuario tiene un nivel de acreditación.

Un usuario puede recuperar un objeto sólo si el nivel de acreditación del usuario es igual o mayor que el nivel de clasificación del objeto. (Propiedad de seguridad simple).

Un usuario puede actualizar un objeto sólo si el nivel de acreditación del usuario es igual al nivel de clasificación del objeto. (Propiedad de estrella)

**Concurrencia**

Los DBMS permiten que muchas transacciones accedan a una misma BD a la vez. Para poder mantener la integridad de los datos, se necesita un mecanismo de control de concurrencia para asegurar que las transacciones concurrentes no interfieran entre sí.

Los 3 problemas comunes de la concurrencia son:

* Actualización perdida:
  + La transacción A recupera alguna tupla t en el tiempo t1.
  + La transacción B recupera la misma tupla t en el tiempo t2.
  + La transacción A actualiza la tupla t en el tiempo t3.
  + La transacción B actualiza la tupla t, en el tiempo t4.
  + La actualización realizada por la transacción A se pierde en el tiempo t4, ya que la transacción B la sobrescribe.
* Dependencia no confirmada:
  + La transacción A ve una actualización no confirmada en el tiempo t2.
  + Esta actualización es posteriormente deshecha en el tiempo t3.
  + La transacción A está operando sobre la suposición falsa de que la tupla t tiene el valor visto en el tiempo t2, siendo que tiene el valor que tenía en el tiempo t1.
  + La transacción A producirá una salida incorrecta.
* Análisis inconsistente:
  + La transacción A está sumando saldos de cuentas.
  + La transacción B está transfiriendo una cantidad X a la cuenta Y.
  + El resultado producido por A, es obviamente incorrecto, y si A continuara, y escribiera ese dato en la BD, dejaría en efecto, la BD en estado inconsistente.
  + Decimos que A vio un estado inconsistente de la BD, y por lo tanto, ha realizado un análisis inconsistente

Todos los problemas pueden ser resueltos por medio de una técnica de control de concurrencia llamada bloqueo.

**Bloqueo**

Cuando una transacción debe asegurarse de que algún objeto en el que está interesado no cambiará de ninguna forma mientras lo está usando, adquiere un bloqueo sobre ese objeto. De esta forma, la transacción tiene la certeza que el objeto permanecerá estable durante el tiempo que lo desee.

Tipos:

* *Bloqueo Exclusivo o de escritura X*: Si una transacción pone un bloqueo exclusivo sobre la tupla , entonces se rechazará una petición de cualquier otra transacción para un bloqueo de cualquier tipo sobre la tupla .
* *Bloqueo compartido o de lectura S*: Si una transacción pone un bloqueo compartido sobre la tupla , entonces:
  + Se rechazará la petición de cualquier transacción para un bloqueo exclusivo sobre .
  + Se otorgará la petición de cualquier transacción para un bloqueo compartido sobre .

Para garantizar que no ocurran los problemas de concurrencia, se tiene un protocolo de bloqueo:

* Una transacción que desea recuperar una tupla debe primero adquirir un bloqueo S sobre esa tupla.
* Una transacción que desea actualizar una tupla, debe primero adquirir un bloqueo X sobre esa tupla. Si ya tiene un bloqueo S sobre la tupla, debe modificar ese bloqueo hacia el nivel X.
* Si una petición de bloqueo es rechazada porque entra en conflicto con un bloqueo ya existente, la transacción pasa a un estado de espera y permanecerá así hasta que la 1º transacción libere el bloqueo.

Con esos tipos de bloque se solucionan los problemas de dependencia no confirmada y análisis inconsistente, pero en el problema de la actualización perdida se genera un bloqueo mortal.

**Bloqueo mortal**

Es una situación donde dos o más transacciones se encuentran en estado simultáneos de espera, cada una esperando que alguna de las demás libere un bloqueo para poder continuar.

La detección de un bloqueo mortal implica la detección de un ciclo en el grafo de espera. La ruptura del bloqueo mortal implica seleccionar una de las transacciones bloqueadas mortalmente como víctima, y entonces deshacerla liberando por lo tanto sus bloqueos y permitiendo que continúen las demás transacciones.

En la práctica no todos los DBMS detectan los bloqueos mortales. Usan un mecanismo de tiempos y asumen que una transacción que no ha realizado ningún trabajo durante cierto periodo preestablecido, está bloqueada mortalmente.

**Bloqueo por Aproximación**

Granularidad del bloqueo

Hasta el momento, supusimos bloqueos a nivel de tuplas, pero se pueden bloquear unidades más grandes o más pequeñas. A esto lo llamamos granularidad del bloqueo.

Definimos 3 nuevos niveles de bloqueo, llamados bloqueos por aproximación, que se aplican sobre las tablas, pero no sobre las tuplas.

* *Bloqueo de aproximación compartida* (IS): bloqueos S sobre tuplas individuales de R.
* *Bloqueo de aproximación exclusiva* (IX): Igual que IS, pero además puede actualizar tuplas individuales, por lo que adquiere bloqueos X sobre esas tuplas.
* *Bloqueo de aproximación compartida exclusiva* (SIX): Combina S e IX, la transacción puede tolerar lectores concurrentes, pero no actualizadores concurrentes en R. Puede actualizar tuplas individuales en R y por lo tanto pondrá bloqueos X sobre esas tuplas.

Antes de que una transacción dada pueda adquirir un bloqueo S sobre una tupla dada deber adquirir un bloqueo IS o una más fuerte sobre la tabla que contiene a esa tupla.

Antes de que una transacción dada pueda adquirir un bloqueo X sobre una tupla dada, primero debe adquirir un bloqueo IX o uno más fuerte sobre la tabla que contiene esa tupla.

Este tipo de bloqueo soluciona el problema de la actualización perdida.

**Seriabilidad**

Es el criterio de corrección aceptado comúnmente para la ejecución de un conjunto dado de transacciones. Se considera que la ejecución de un conjunto de transacciones es correcta cuando es seriable, es decir, cuando produce el mismo resultado que una ejecución serial de las mismas transacciones, ejecutando una a la vez.

Las transacciones individuales son tomadas como correctas: se da por hecho que transforman un estado correcto de la BD en otro estado correcto. También es correcta la ejecución de una transacción a la vez en cualquier orden serial. Una ejecución intercalada es correcta cuando equivale a alguna ejecución serial, es decir, es seriable.

Ejemplo:

T1: Sumar $10 al saldo del cliente 1.

T2: Descontar 2% de gastos administrativos del saldo.

T3: Incrementar intereses de la cuenta en un 5%.

Saldo inicial: $200

¿Cuántos resultados consistentes posibles hay?

Solución: 3! = 6. Son todas las combinaciones posibles del conjunto de transacciones.

Todos los resultados son consistentes, pero puede que no sean iguales.

**Niveles de Aislamiento**

Es el grado de interferencia que una transacción dada es capaz de tolerar por parte de transacciones concurrentes.

Si queremos garantizar la seriabilidad, no podemos aceptar ningún tipo de interferencia, es decir, el nivel de aislamiento deber ser el máximo imposible.

Los niveles posibles son: READ UNCOMMITED (leer datos que todavía no se grabaron en la BD), READ COMMITED (lectura de datos ya actualizados), REPETEABLE READ o SERIALIZABLE (secuencial).

Generalmente, el nivel predeterminado una BD es SERIALIZABLE (el más alto), pero se puede especificar uno menor. Si alguna transacción se efectúa a un nivel menor de aislamiento, la seriabilidad podría ser violada.

Se definen tres formas específicas en las que se puede violar la seriabilidad:

* Lectura sucia: La transacción T1 realiza una actualización sobre alguna fila, la transacción T2 recupera esa fila y la transacción T1 termina con una transacción deshacer. La transacción T2 ha visto una fila que ya no existe, y que en cierto sentido nunca existió.
* Lectura no repetible: T1 recupera una fila, luego T2 actualiza esa fila y después la transacción T1 recupera nuevamente la misma fila. La transacción T1 ha recuperado la misma fila dos veces, pero ve dos valores diferentes en ella.
* Fantasmas: La transacción T1 recupera el conjunto de las filas que satisfacen alguna condición. La transacción T2 inserta entonces una nueva fila que satisface la misma condición. Si la transacción T1 repite ahora su petición de recuperación, vera una fila que antes no existía, un “fantasma”.

**Niveles de Aislamiento de SQL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nivel de Aislamiento** | **Lectura Sucia** | **Lectura no repetible** | **Fantasmas** |
| READ UNCOMMITED | SI | SI | SI |
| READ COMMITED | NO | SI | SI |
| REPEATABLE READ | NO | NO | SI |
| SERIALIZABLE | NO | NO | NO |

La única forma de evitar los fantasmas es bloquear la ruta de acceso que se utiliza para acceder a los datos en consideración. Este bloqueo puede ser a nivel de un índice.

**Bloqueos pesimistas y optimistas**

Bloqueo Pesimista: Se adquieren bloqueos sobre entradas y luego mantiene losbloqueos hasta que se realiza la confirmación. Esta estrategia de bloqueo proporciona una mayor coherencia a costa del rendimiento. Como se bloquean los datos, no se pueden acceder mientras se están haciendo modificaciones.

Bloqueo Optimista: Se toma una imagen anterior (copia) de cada registro que toca la transacción y compara la imagen con los valores de entrada actuales cuando se confirma la transacción. Si los valores de entrada cambian, la transacción se retrotrae (sigue insistiendo hasta que pueda ingresar la transacción). No se mantiene ningún bloqueo hasta el momento de la confirmación. Esta estrategia de bloqueo proporciona una mejor concurrencia que la estrategia pesimista, con el riesgo de que la transacción se retrotraiga y el costo de memoria de realizar una copia adicional de la entrada. Se pueden leer los datos mientras se está trabajando sobre la copia.

Comparación:

* La gran diferencia es la cantidad de tiempo durante el cual el registro queda bloqueado. En el optimista el tiempo es menor.
* El bloqueo pesimista tiende a saturar el uso del log porque fuerza a la BD a mantener las transacciones abiertas por mucho tiempo.
* En BD donde los accesos de actualización son de menor porcentaje, es preferible el optimista porque el pesimista tiende a arruinar el rendimiento de la BD, a crear situaciones de deadlock (bloqueos mortales) con más frecuencia, a generar filas de espera por la disponibilidad de los datos y a aumentar la competición entre los usuarios.
* Cuanto mayor sea la cantidad de usuarios, mayores serán los trastornos que el tratamiento pesimista causará porque a mayor cantidad de bloqueos, mayor cantidad de problemas.

**Cifrado de Datos**

Cuando hablamos de seguridad, también tenemos que tener en cuenta al usuario que trata de causar daño sobre el sistema (por ejemplo: eliminando físicamente parte de la BD o interviniendo una línea de comunicación). La medida más efectiva contra este tipo de amenazas es el cifrado de datos (o encriptación), que implica guardar y transmitir los datos sensibles en forma cifrada.

Un texto plano es cifrado sometiéndolo a un algoritmo de cifrado. Los detalles del algoritmo de cifrado son públicos, pero la clave de cifrado se mantiene en secreto. El texto cifrado, que debe ser ininteligible para cualquiera que no posea la clave de cifrado, es lo que se guarda en la BD o se transmite por la línea de comunicación.

El esquema de cifrado debería ser tal, que el trabajo involucrado en romperlo sea mayor que cualquier beneficio potencial que pudiera obtenerse al hacerlo.

Estándar de cifrado de datos

*Sustitución*: se usa una clave de cifrado para determinar para cada carácter del texto plano un carácter de texto cifrado que va a sustituir a ese carácter.

*Permutación*: los caracteres del texto plano son simplemente reorganizados en una secuencia diferente.

Los algoritmos que usan una combinación de ambos, brindan un alto grado de seguridad. Uno de estos algoritmos es el Estándar de Cifrado de Datos (DES), desarrollado por IBM.

Cifrado de clave pública

En este esquema, el algoritmo de cifrado y la clave de cifrado están disponibles, por lo que cualquier persona puede convertir texto plano en cifrado. La clave de descifrado correspondiente se mantiene en secreto. La clave de descifrado no puede ser deducida de manera realista a partir de la clave de cifrado.

**U3: OPTIMIZACIÓN**

**Optimización**

La optimización de consultas es el proceso de selección del plan de evaluación de las consultas más eficiente entre muchas estrategias disponibles para el procesamiento de una consulta dada, especialmente si la consulta es compleja.

En nivel del algebra relacional, el DBMS intenta hallar una expresión que sea equivalente a la expresión dada, pero de ejecución más eficiente. Además, debe elegir una estrategia detallada para el procesamiento de la consulta, como puede ser la selección de un algoritmo para ejecutar una operación, la selección de índices concretos que se van a emplear, etc.

La diferencia de costos (en términos de tiempo de evaluación) entre una estrategia buena y una mala suele ser sustancial. Por lo tanto, vale la pena que el sistema pase una importante cantidad de tiempo en la selección de una buena estrategia.

Ejemplo de optimización

*Objetivo*: obtener el nombre de los proveedores que proporcionan la parte P2.

Supongamos que la BD contiene 100 proveedores y 10000 envíos, de los cuales solo 50 son de la parte P2. Supongamos que las tablas proveedor y envío están representadas en disco como dos archivos guardados por separado.

Si el sistema evalúa la expresión “directamente” (sin optimizar), la secuencia sería:

* *Juntar* proveedor con envío: implica la lectura de 10000 envíos, la lectura de cada uno de los 100 proveedores 10000 veces (una vez por cada envío) la construcción de 10000 tuplas juntadas y la escritura de 10000 tuplas de nuevo en disco.
* *Restringir* este resultado sólo a las tuplas para la parte P2: involucra la lectura de las 10000 tuplas juntadas hacia memoria, y produce un resultado de 50 tuplas que quedan en MP.
* *Proyectar* el resultado sobre proveedor: esto produce el resultado final deseado.

El siguiente procedimiento es equivalente (produce el mismo resultado) pero es más eficiente:

* *Restringir* los envíos solamente a las tuplas para la parte P2: esto involucra la lectura de 10000 tuplas, pero produce un resultado de 50 tuplas, que pueden mantenerse en memoria principal.
* *Juntar* el resultado del paso anterior, con proveedor: este paso involucra la lectura de 100 proveedores (solo una vez, y no por cada envío de P2) y produce nuevamente un resultado de 50 tuplas.
* *Proyectar* el resultado del paso anterior sobre proveedor.

El 1º de estos procedimientos implica la E/S de 1.030.000 tuplas. La 2º involucra 10.100. Si tomamos la cantidad de E/S de las tuplas como medida de desempeño, el 2º procedimiento es un poco más de 100 veces mejor que el 1º.

Un cambio muy simple en el algoritmo de ejecución ha producido una reducción drástica en el desempeño. La mejora sería todavía más drástica si los envíos estuvieran además indexados en la parte, ya que la cantidad de tuplas de envíos leídas sería 50 y no 10000, y el nuevo procedimiento sería 7000 veces mejor que el original.

**Información del diccionario o catálogo del optimizador**

, el número de tuplas de la relación .

, el número de bloques que contiene tuplas de la relación .

, el tamaño de cada tupla de la relacion en bytes.

, el factor de bloqueo de la relación , es decir, el número de tuplas de la relación que caben en un bloque.

, el número de valores distintos que aparecen en la relación para el atributo . Si es una clave de la relación , entonces es . Esta estadística también se puede calcular para conjuntos de atributos, en vez de solo para atributos aislados.

Las estadísticas sobre los índices, como las alturas de los arboles B+ y el número de páginas hojas de los índices, también se conservan en el catálogo.

Los optimizadores verdaderos suelen conservar información estadística adicional para mejorar la precisión de sus estimaciones de costos de los planes de evaluación.

Actualización de estadísticas: Si se desean conservar estadísticas precisas, cada vez que se modifica una relación también hay que actualizar las estadísticas. Esto supone una “sobrecarga adicional”. La mayoría de los DBMS no actualizan las estadísticas con cada modificación, sino en los momentos de menor carga. También se pueden forzar. Puede que el plan que elija no sea el mejor. En consecuencia, puede que las estadísticas utilizadas para escoger una estrategia no sean lo suficientemente exactas.

**Estimación del tamaño de la selección**

La estimación del tamaño del resultado de una operación de selección depende del predicado de la selección.

Si hacemos una selección por igualdad, en la que el atributo tome el valor si se supone una distribución uniforme de los valores y que el valor aparece en el atributo de algún registro de , se puede estimar que el resultado de la selección sería: tuplas. Pese al hecho de que la suposición de distribución uniforme no suele ser correcta, resulta una aproximación razonable de la realidad.

Considere una selección donde , si el valor real utilizado en la comparación está disponible en el momento de cálculo de costos, puede hacerse una estimación más precisa. Los valores mínimo y máximo del atributo están almacenados en el catálogo. Suponiendo nuevamente una distribución uniforme, se puede estimar el nº de registros que cumplirán con la condición . Si , son registros y si son registros. En cualquier otro caso:

Para , a restarle la fórmula.

Ejemplo:

SELECT \* FROM Alumnos WHERE edad <= 20

Catalogo: Edad entre 17 y 30

Alumnos tiene 2000 registros

Cantidad de valores distintos: 14

Cuántos registros devuelve?

Cuántos registros iguales a 20 hay?

Consideraciones:

Cuando la selección involucra una condición simple, lo que se hace es verificar si existe un camino de acceso para el atributo involucrado en la selección y emplear el acceso si existe. Si no, se emplea la fuerza bruta.

Cuando se tienen condiciones de selección del tipo conjuntivas (AND), y si hay varios caminos de acceso, el optimizador debe elegir el camino de acceso más directo, es decir que recupera la menor cantidad de registros al menor costo. En este caso se debe tener en cuenta:

El diccionario de datos también guarda estimaciones de la selectividad.

Cuando se tienen condiciones de selección del tipo disyuntiva (OR) son más difíciles de procesar y optimizar. En este caso si cualquiera de las condiciones no tiene un camino de acceso, lo único que se puede aplicar es la fuerza bruta.

En algunos casos, cuando la consulta forma parte de un procedimiento almacenado, puede que el valor no esté disponible cuando se optimice la consulta. En estos casos se supondrá que aproximadamente la mitad de los registros cumplen la condición de comparación. Esta estimación puede resultar muy imprecisa, pero es lo mejor que se puede hacer sin información.

Estimación del tamaño de las reuniones

El producto cartesiano contiene tuplas. Cada tupla de ocupa bytes (tamaño + tamaño), de donde se puede calcular el tamaño del producto cartesiano.

* Si la intersección entre R y S es vacía, es decir las relaciones no tienen atributos en común, entonces se puede usar la técnica de estimación anterior para los productos cartesianos.
* Si la intersección entre R y S es una clave de R, entonces se sabe que cada tupla de se combinará como máximo con una tupla de . Por lo tanto, el nº de tuplas no es mayor que el número de tuplas de S.
* El costo más difícil es que la intersección entre R y S no sea una clave de R ni de S.
* Se realizan estimaciones para todas las consultas: proyección, agregación, operaciones de conjunto, etc.

**Procesamiento de consultas del optimizador**

1. Convertir la consulta a su forma interna: Implica transformar la consulta original en su representación interna que sea más adecuada para manejarla en la máquina. El procesamiento de vistas también se realiza en esta etapa.

Por lo general, la forma interna seleccionada es algún tipo de árbol de sintaxis abstracto o árbol de consulta. Podemos pensar que esta representación interna representa alguno de los formalismos del algebra relacional.

1. Convertir a la forma canónica: En esta etapa, el optimizador hace optimizaciones que son garantizadas como buenas, sin tener en cuenta las rutas de acceso físicas (índices) ni los valores actuales de los datos. En este paso, el optimizador convierte la representación interna en alguna forma canónica equivalente, con el objeto de encontrar una representación que sea en cierta forma, más eficiente que la anterior.
2. Seleccionar procedimientos candidatos de bajo nivel: El optimizador debe elegir cómo ejecutar dicha consulta transformada. Aquí entran en juego consideraciones tales como la existencia de índices u otras rutas de acceso físicas, la distribución de los valores, el agrupamiento físico de los datos almacenados, etc. La estrategia básica consiste en considerar la expresión como operaciones de “bajo nivel” con cierta independencia entre sí. Luego para cada operación de bajo nivel, el implementador tendrá una serie de procedimientos de implementación predefinidos. Cada procedimiento tendrá una fórmula de costo asociada, que indique el costo de ejecutar ese procedimiento.
3. Generar planes de consulta y seleccionar el de menor costo: Cada plan se construye por medio de combinación de una serie de procedimientos predefinidos candidatos. La elección del plan de menor costo, requiere de un método para asignar un costo a cualquier plan dado. Este costo es básicamente la suma de los costos de los procedimientos individuales que conforman ese plan. La estimación precisa de los costos puede ser un problema difícil.

Divide y vencerás

Una estrategia para optimizar es la descomposición de la consulta, y es una estrategia muy usada en ambientes distribuidos y de procesamiento en paralelo. Se divide una consulta que involucra muchas variables en una secuencia de consultas pequeñas que generalmente involucren cada una, a una o dos de estas variables, usando separación y sustitución de tuplas para lograr esta descomposición.

**Análisis de consultas y transacciones en la BD**

1° Análisis de consultas

1. Los archivos a los que se acceden con las consultas.
2. *Los archivos con los que se realizan las condiciones de selección de las consultas.*
3. *Los atributos con los que se realizan las condiciones de join o las condiciones sobre tablas y objetos múltiples.*
4. Los atributos que se recuperan con las consultas.

Los atributos que se listen en 2 y 3 son candidatos para la definición de estructuras de acceso (índices).

2° Análisis de transacciones de actualización

1. Los archivos a ser actualizados.
2. Tipos de operación sobre cada archivo (insert, update o delete).
3. *Los atributos con los que se realizan las condiciones de selección para un delete o update.*
4. Los atributos cuyos valores cambian por una operación de actualización.

Los atributos que se listen en 3 son candidatos para la definición de estructuras de acceso, en cambio los que se listan en 4 deben evitarse que formen parte de una estructura de acceso (sobrecarga el sistema que se utilicen como índices columnas que cambian constantemente, ya que se debe tener una tabla asociada de índices actualizada todo el tiempo).

Otros análisis:

1. *Análisis de la frecuencia esperada de consultas y transacciones en la BD*: por lo general se cumple la regla del 80-20.
2. *Análisis de las restricciones de tiempo de consultas y transacciones en la BD*: por ejemplo, transacciones que se deben completar en menos de 5 segundos en el 95% de los casos, y que nunca debe pasar de los 20 segundos.
3. *Análisis de la frecuencia de operaciones de actualización*: un archivo que se actualiza frecuentemente debe tener un mínimo de caminos de acceso.
4. *Análisis de las restricciones de unicidad en los atributos*: es para especificar los caminos de acceso de todos los atributos que son claves primarias o candidatas (condición de unicidad).

**Índices**

* *Cuándo indexar*: El atributo debe ser clave o debe ser parte de una consulta en una operación JOIN y WHERE.
* *Qué atributos indexar*: Se puede construir sobre uno o múltiples atributos, en el caso de atributos múltiples, el índice se debe generar con el mismo orden de atributos que cuando se realizan las consultas.
* *Cuándo generar un índice-agrupación*: Cuando el archivo es ordenado físicamente con el atributo que se indexa y no es clave. Si una tabla requiere varios índices, la decisión es si es necesario un índice de agrupación. Si se busca por rango, la generación de este tipo de índices puede ser beneficiosa.
* *Cuándo generar un índice hash en lugar de un árbol B*: Los arboles B funcionan bien para cuando tenemos búsquedas particulares y por rango. Los índices hash funcionan bien para condiciones de igualdad y cuando se buscan registros que tienen que igualar condiciones de JOIN.
* *Cuándo generar hash dinámico*: Para archivos volátiles, el empleo de hashing dinámico puede ser apropiado. Esto no se ofrece en muchos DBMS comerciales.
* Son desventaja en aquellas tablas en las que se utiliza frecuentemente operaciones de escritura porque se actualizan cada vez que se modifica una columna y en tablas muy pequeñas porque no es necesario ganar tiempo en consultas.
* No son aconsejables cuando pretendemos que la tabla sobre la que se aplica devuelva una gran cantidad de datos en cada consulta.
* Debemos tener en cuenta que ocupan espacio y en algunas ocasiones incluso más espacio que los propios datos.

Tipos de índices:

* **Árbol B:** un árbol B de orden p es un árbol de búsqueda n-ario donde los nodos tienen p hijos como máximo, y en el cual se añade la condición de balanceo de que todas las hojas estén al mismo nivel. Es decir, los nodos internos deben tener un número variable de nodos hijo dentro de un rango predefinido. Para que siga manteniéndose el número de nodos dentro del rango predefinido, los nodos internos se juntan o se parten. Pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados. Los límites (superior e inferior) en el número de nodos hijo son definidos para cada implementación en particular.

Un árbol B elimina el almacenamiento redundante de los valores de la clave búsqueda. Cuando se inserta o se elimina un dato de la estructura, la cantidad de nodos hijo varía dentro de un nodo.

* **Árbol B+:** Un índice de árbol B+ toma la forma de un árbol equilibrado donde los caminos de la raíz a cada hoja del árbol son de la misma longitud. Representa una colección de datos ordenados de manera que se permite una inserción y borrado eficientes de elementos. Es un índice multinivel dinámico con un límite máximo y mínimo en el número de claves por nodo. Un árbol B+ es una variación de un árbol B. Toda la información se guarda en las hojas. Los nodos internos sólo contienen claves y punteros. Todas las hojas se encuentran en el mismo nivel, que corresponde al más bajo. Los nodos hoja se encuentran unidos entre sí a través de una lista enlazada para permitir principalmente recuperación en rango mediante búsqueda secuencial.
* **Árbol B\*:** su principal característica es que, si en los árboles B los nodos deben estar ocupados como mínimo hasta la mitad, en los B\* tienen que estar ocupados más de dos tercios del máximo. Para mantener esto, los nodos, en lugar de generar inmediatamente un nodo cuando se llenan, comparten sus claves con el nodo adyacente. Cuando ambos están llenos, entonces los dos nodos se transforman en tres. También requiere que la clave más a la izquierda no sea usada nunca.
* **Índices Hash:** Es una estructura de datos que asocia claves con valores. Se le denomina función hash o de dispersión h(K) a la función que transforma una clave K en una dirección, la cual se usa como base para la búsqueda y almacenamiento de registros. Funciona transformando la clave con una función hash en un hash, un número que identifica la posición donde la tabla hash localiza el valor deseado.

Esta función produce ‘colisiones’ que deben ser administradas.

**Buenas prácticas de optimización**

Diseño de la BD

* Normalizar las tablas para asegurar que no hay duplicidad de datos y aprovechar al máximo el almacenamiento. Si hay que desnormalizar alguna tabla, se debe pensar en la ocupación y en el rendimiento antes de proceder.
* Los primeros campos de cada tabla deben ser aquellos campos requeridos (no nulos) y dentro de los requeridos primero se definen los de longitud fija y después los de longitud variable.
* Ajusta al máximo el tamaño de los campos para no desperdiciar espacio.
* Es muy habitual dejar un campo de texto para observaciones en las tablas. Si a este campo se va a acceder con poca frecuencia o si se ha definido con gran tamaño, es mejor crear una nueva tabla que contenga la clave primaria de la primera y el campo para observaciones.

Gestión y elección de índices

Los índices son campos elegidos arbitrariamente por el constructor de la BD que permiten la búsqueda a partir de dicho campo a una velocidad notablemente superior. Esto se ve contrarrestada por el hecho de ocupar mucha más memoria (el doble más o menos) y de requerir para su inserción y actualización un tiempo de proceso superior.

No podemos indexar todos los campos de una tabla extensa ya que doblamos el tamaño de la BD. Tampoco sirve de mucho el indexar todos los campos en una tabla pequeña ya que las selecciones pueden efectuarse rápidamente de todos modos.

Un caso en el que los índices pueden resultar muy útiles es cuando realizamos peticiones simultáneas sobre varias tablas. Son más útiles en las igualdades. En cambio, en las desigualdades sólo si los registros están ordenados.

Los índices pueden resultar contraproducentes si los introducimos sobre campos triviales a partir de los cuales no se realiza ningún tipo de petición ya que, además del problema de memoria, estamos ralentizando otras tareas de la BD como son la edición, inserción y borrado.

Dónde escribir las sentencias

* Siempre es preferible utilizar consultas almacenadas dentro del motor de BD y no dentro de la aplicación, ya que se ejecuta mucho más rápido y directamente sobre el motor.
* Muchas de las aplicaciones permiten unir y realizar los joins y consultas directamente en la herramienta produciendo una baja de performance realmente considerable. Lo correcto y más eficiente seria generar la consulta en un procedimiento almacenado con todos los joins y guardar el resultado en una tabla temporal, de donde posteriormente el reporte solo deberá mostrar los datos sin necesidad de trabajarlos primero. Dependiendo los joins que intervengan y los registros involucrados se puede mejorar drásticamente la performance.

Formulación de las consultas

* No utilizar sin necesidad SELECT \* porque el motor debe leer primero la estructura de la tabla antes de ejecutar la sentencia.
* Seleccionar solo aquellos campos que se necesiten.
* Utilizar JOIN para unir las tablas en lugar del WHERE, esto permite que a medida que se declaran las tablas se unan mientras que si utilizamos el WHERE el motor genera primero el producto cartesiano de todos los registros de las tablas para luego filtrar las correctas, un trabajo definitivamente lento.

Refinamiento de las consultas

* Una consulta debe ser refinada si realiza demasiados accesos a disco y si el plan de ejecución muestra que no se emplean índices relevantes.
* Muchos optimizadores no emplean los índices en presencia de cálculos aritméticos, comparación numérica de atributos de diferente tipo y precisión, comparaciones con null o con subcadenas.
* Algunos DISTINCTS pueden ser redundantes y evitarse.
* Los índices no se emplean para subconsultas que tienen el operador IN.
* Se debe evitar el uso de tablas resultado temporarias colapsando múltiples consultas en una simple.
* El procesamiento de JOINS se ve afectado por el orden de las tablas en el FROM.
* Si existen varias opciones de JOIN para una consulta se debe elegir la que usa un índice en lugar de comparaciones de cadenas.
* No se debe abusar del uso de vistas porque a veces es preferible hacer una consulta sobre la tabla base que sobre una vista.
* En una consulta con condiciones múltiples conectadas por operadores OR el optimizador puede no usar el índice.

Para hacer más rápida una consulta se debe intentar las siguientes transformaciones:

* Las condiciones NOT se deben transformar en condiciones positivas.
* Bloques SELECT anidados que usan IN, =ALL, =ANY e =SOME, se deben reemplazar por JOINS.
* Las condiciones WHERE se pueden reescribir para utilizar índices sobre columnas múltiples.

Ejemplo de optimización de consultas:

*Consulta*: para todos los proyectos localizados en Santa Fe, liste el número del proyecto, el departamento de la empresa que lo ejecuta, el apellido del administrador del departamento, su dirección y fecha de nacimiento.

SEELCT P.numero, D.Dnum, E.Apellido, E.direccion, E.Fnac

FROM Proyecto P, Departamento D, Empleado E

WHERE P.Dnum = D.Dnum AND D.DNI = E.DNI AND P.ubicacion = “Santa Fe”

*Para optimizar*: agregar índice en ubicación, número de departamento y dni de empleado.

Al tener índices, el optimizador se da cuenta de que tiene que empezar por el WHERE y no por el FROM para ahorrar operaciones.

El DBMS va a hacer un árbol de la consulta: Primero filtra por ubicación, luego hace el producto cartesiano con la tabla de departamentos y empleados, luego la proyección. En los árboles, las relaciones de entrada (tablas) que participan en la consulta son nodos hoja, y las operaciones del álgebra relacional que se ejecutan se representan como nodos internos. La ejecución del árbol de consulta consiste en efectuar la operación interna de nodo y reemplazar el operando por la relación que resulta de ejecutar toda la operación.

El optimizador, para elegir el mejor plan de ejecución, evalúa cada costo: compara la cantidad de registros que tiene que recorrer.

Buenas prácticas Particulares: MySQL

Especificar el alias de la tabla delante de cada campo definido en el select, esto le ahorra tiempo al motor de tener que buscar a que tabla pertenece el campo especificado.

El orden de ubicación las tablas en el from deberían ir en lo preferible de menor a mayor según el número de registros, de esta manera reducimos la cantidad de revisiones de registros que realiza el motor al unir las tablas a medida que se agregan.

Buenas prácticas Particulares: SQL Server

Si desea guardar objetos grandes en la BD como imágenes, documentos, etc., no hacerlo en la misma tabla de la entidad.

Índice clustered: ordena físicamente los registros por ese índice. Útil para desigualdades y agrupamientos. Cada tabla soporta un solo índice clustered. Se debe prestar atención a su creación ya que puede tener impacto negativo en la performance, pero en general se recomienda que todas las tablas lo tengan.

**Vistas materializadas**

Cuando se define una vista, la BD sólo almacena la consulta que define la vista. Al definir una vista materializada también se almacena el contenido calculado. Constituyen datos redundantes, pero resulta mucho más económico en muchos casos leer el contenido de la vista materializada que calcular el contenido de una vista ejecutando la consulta que la define.

Las vistas materializadas resultan importantes para la mejora del rendimiento de algunas aplicaciones. Un problema es su mantenimiento. Pueden mantenerse mediante código escrito (triggers o procedimientos almacenados).

La decisión de qué vistas materializar debe tomarse teniendo en cuenta la carga de trabajo del sistema.

**Implementación de los operadores relacionales para resolver producto cartesiano**

Métodos directos para implementar algunos operadores relacionales, especialmente el JOIN. Estos métodos pertenecen al 3º paso en el procesamiento de consultas (procedimientos candidatos de implementación de bajo nivel).

Fuerza bruta

Es lo que podría llamarse el caso “llano” donde se inspeccionan todas las combinaciones posibles de tuplas (cada tupla de R es examinada en conjunción con cada tupla de S). Es el más costoso de todos.

Costos asociados (mirando solamente el costo de E/S, aunque en realidad sean importantes otros costos): el enfoque requiere claramente un total de operaciones de lectura de tuplas, pero la cantidad de escrituras será distinta según la cardinalidad de la relación:

* En el caso de que un JOIN de muchos a uno (en particular de clave externa con una clave candidata coincidente), es claro que la cardinalidad del resultado es igual a la cardinalidad (m o n) de R o de S, dependiendo del cual representa el lado de la clave externa del JOIN.
* En el caso de muchos a muchos, si suponemos una distribución uniforme de los valores, la cantidad total de tuplas del JOIN sería , donde dCRoS es la cantidad de valores distintos del atributo de junta C en la relación R o S (el menor de los dos).

Este es el procedimiento del peor caso, ya que asume que la relación S no está indexada ni dispersada sobre el atributo de junta C. Para ganar velocidad, se puede aplicar éste método con índice o tabla Hash.

Búsqueda con índice

Consideremos el caso en el que existe un índice X, sobre el atributo S.C de la relación interna.

La ventaja de esta técnica sobre la fuerza bruta es que para una tupla dada de la relación externa R, se puede ir “directamente” a las tuplas coincidentes de la relación interna S. La cantidad de lecturas de tuplas de las relaciones R y S es simplemente la cardinalidad del resultado juntado.

Búsqueda con dispersión (tabla Hash)

Es similar a la búsqueda con índice, con excepción de que la ruta de acceso rápida a la relación interna S, sobre el atributo de junta S.C, se hace con dispersión en lugar de un índice. Se accede a los datos a través de una estructura hash.

La tabla Hash tiene el resultado de una función que apunta directamente a la tabla de datos.

Mezcla

La técnica de mezcla supone que las dos relaciones R y S están físicamente guardadas en secuencia por los valores del atributo de junta C. Si este es el caso, las dos relaciones pueden ser revisadas en secuencia física, las dos revisiones pueden ser sincronizadas y la unión completa puede ser realizada en una sola pasada sobre los datos.

El agrupamiento físico de los datos relacionados lógicamente es uno de los factores críticos de desempeño. En ausencia de tal agrupamiento, a menudo es buena idea ordenar una o ambas relaciones en tiempo de ejecución.

Por más que las tablas sean largas, es una lectura rápida.

Dispersión

Al igual que la técnica de mezcla, la técnica de dispersión requiere de una sola pasada sobre cada una de las dos relaciones.

La primera pasada construye una tabla de dispersión para la relación S sobre los atributos de junta S.C; las entradas de esa tabla contienen el valor del atributo de junta y un apuntador hacia la tupla correspondiente en el disco.

La segunda pasada revisa la relación R y aplica la misma función de dispersión al atributo de junta R.C.

La gran ventaja de esta técnica sobra la de mezcla es que las relaciones R y S no necesitan estar guardadas en ningún orden específico y no es necesario ordenamiento.

Es un método lento porque armar la tabla Hash es muy costoso. Se usa generalmente para no usar fuerza bruta sin índice porque es más lento.

**U4: BD DISTRIBUIDAS**

**BD Distribuidas**

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

Principio: Ante el usuario, un SBDD debe lucir exactamente igual que un sistema que no es distribuido.

* Las empresas ya están distribuidas al menos físicamente.
* Eficiencia de procesamiento.
* Mayor accesibilidad. El fallo en un nodo no se nota.
* Complejidad y costo en cuanto a SW y administración.
* Seguridad.

**12 Reglas u objetivos**

1. Autonomía local: Los sitios deben ser autónomos. Todas las operaciones de ese sitio deben ser administradas por ese sitio. La seguridad, integridad y representación del almacenamiento de los datos locales permanecen bajo el control y jurisdicción del sitio local.
2. No dependencia de un sitio central: Todos los sitios deben ser tratados como iguales. Esta relacionado con el anterior, pero la no dependencia de un sitio central es necesaria por sí misma, aunque no se logra la autonomía local completa.
3. Operación continua: Deben ofrecer mayor confiabilidad y mayor disponibilidad.
   1. La confiabilidad mejora ya que los sistemas distribuidos pueden continuar operando cuando hay alguna falla en algún sitio individual.
   2. La disponibilidad mejora, en parte por la razón anterior y además por la posibilidad de replicación de datos.
4. Independencia de ubicación: Los usuarios no tienen que saber dónde están almacenados físicamente los datos. Permite que los datos emigren de un sitio a otro sin invalidar programas o actividades.
5. Independencia de fragmentación: La fragmentación es necesaria por razones de rendimiento. Los datos pueden estar almacenados en la ubicación donde son usados más frecuentemente para que la mayoría de las operaciones sean locales y se reduzca el tráfico en la red. La fragmentación debe hacerse de la forma más óptima.
6. Independencia de replicación: Las réplicas son necesarias por dos razones como mínimo: pueden significar un mejor rendimiento, y pueden significar mejor disponibilidad. La réplica se debe mantener al mínimo. La principal desventaja es la actualización de datos replicados. Los usuarios deben ser capaces de comportarse como si los datos no estuvieran replicados.
7. Procesamiento de consultas distribuidas: La optimización en un sistema distribuido es aún más importante que en uno centralizado. En una consulta habrá muchas formas distintas de satisfacer esa solicitud. Por esto, es crucial la optimización en el procesamiento de BDD.
8. Administración de transacciones distribuidas: Dos aspectos fundamentales son: el control de la recuperación y el control de concurrencia. Una sola transacción puede involucrar actualizaciones en más de un sitio. Cada sitio tiene un “agente” que es el proceso realizado en una transacción. El sistema necesita saber cuándo dos agentes son parte de la misma transacción, ya que, por ejemplo, no debería permitir que dos agentes que forman parte de la misma transacción caigan en un bloqueo mortal entre ellos.
9. Independencia de HW: Existe una necesidad real de poder integrar los datos en máquinas distintas y presentar al usuario una imagen de sistema único. Es necesario tener la posibilidad de ejecutar el mismo DBMS en diferentes plataformas de HW, y hacer que esas máquinas diferentes participen como socios igualitarios en un sistema distribuido.
10. Independencia de SO: Es necesario ejecutar el mismo DBMS en diferentes plataformas de SO.
11. Independencia de red: Si el sistema tiene que soportar sitios distintos, con HW y SW distinto, es obviamente necesario tener la posibilidad de soportar también una variedad de redes de comunicación distintas.
12. Independencia del DBMS: El soporte para la heterogeneidad es definitivamente necesario. El sistema distribuido ideal debe proporcionar independencia de DBMSs.

**BDD Homogéneas y Heterogéneas**

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

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

**Almacenamiento distribuido de datos**

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

* *Réplica*: el sistema conserva réplicas o copias idénticas de la relación y guarda cada una en un sitio diferente.
* *Fragmentación*: el sistema divide la relación en varios fragmentos y guarda cada fragmento en un sitio diferente.

Réplica de datos

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

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

En general la réplica mejora el rendimiento de las operaciones de lectura y aumenta la disponibilidad de los datos para las transacciones de lectura. Las transacciones de actualización suponen una mayor sobrecarga. Se puede simplificar la gestión de las réplicas de la relación , escogiendo una localización como copia principal de .

Fragmentación de los datos

Si la relación se fragmenta, se divide en varios fragmentos . Estos fragmentos contienen suficiente información para permitir la reconstrucción de r.

Existen dos esquemas diferentes de fragmentación:

* *Fragmentación Horizontal*: la relación se divide en varios subconjuntos . Cada tupla de la relación debe pertenecer como mínimo a uno de los fragmentos. Esta fragmentación suele utilizarse para conservar las tuplas en los sitios que más se utilizan, minimizando la transferencia de datos. Se tienen todas las columnas de algunos registros particulares.
* *Fragmentación Vertical*: implica la definición de varios subconjuntos de atributos del esquema . Una forma de asegurar que la relación pueda reconstruirse es incluir los atributos de la clave principal de R en cada fragmento . Se tienen todos los registros, pero sólo algunas columnas.

Almacenamiento

Se pueden aplicar los dos tipos de fragmentación a un solo esquema, y los fragmentos también pueden replicarse. En general, podemos decir, que los fragmentos pueden replicarse, las réplicas de los fragmentos pueden fragmentarse, etc.

**Transparencia**

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

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

Los elementos de datos (relaciones, fragmentos, réplicas) deben tener nombres únicos. En las BDD, hay que tener cuidado para asegurarse de que dos sitios no utilicen el mismo nombre para elementos diferentes.

Una solución es tener un servidor de nombres central. Los inconvenientes son que el servidor de nombres se transforme el cuello de botella, y si queda fuera de servicio, puede que no siga funcionando ningún otro sitio del SBDD.

Otra solución es que cada sitio anteponga su propio identificador de sitio a cualquier nombre que genere, pero no logra la transparencia de la ubicación. Para superar este problema, el SBDD puede crear un conjunto de nombres alternativos o alias para elementos de datos.

**Asignación de Fragmentos**

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

Parámetro para la generación de réplicas:

* Cantidad de lecturas
* Cantidad de escrituras

Comparación de las alternativas de réplica

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Réplica Total** | **Réplica Parcial** | **Partición** |
| Procesamiento de consultas | Fácil | Difícil | Difícil |
| Control de concurrencia | Moderado | Difícil | Fácil |
| Confiabilidad | Muy alta | Alta | Baja |
| Realidad | Posible aplicación | Realista | Posible aplicación |

**Transacciones Distribuidas**

El acceso a los diferentes elementos de datos en los sistemas distribuidos suele realizarse mediante transacciones, que deben preservar las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).

Hay dos tipos de transacciones que se deben considerar:

* Transacciones locales: acceden y actualizan en una BD local.
* Transacciones globales: acceden y actualizan datos en varias BD locales.

Asegurar las propiedades ACID en transacciones globales es una tarea complicada, dado que participan en la ejecución varios sitios. El fallo de alguno de los sitios, o el de un enlace de comunicación puede dar lugar a cálculos erróneos.

**Estructura del SGBDD**

Cada sitio tiene su propio gestor local de transacciones que asegura las propiedades ACID de las transacciones que se ejecutan en ese sitio. Cada gestor de transacciones colabora para ejecutar las transacciones globales.

Cada gestor contiene: el gestor de transacciones, y el coordinador de transacciones que coordina la ejecución de las diferentes transacciones (tanto locales como globales) iniciadas en ese sitio.

El coordinador de transacciones es responsable de:

* Inicio de la ejecución de la transacción.
* División de la transacción en varias subtransacciones y distribución de esas subtransacciones a los sitios correspondientes para su ejecución.
* Coordinación de la terminación de la transacción (que la transacción se comprometa o aborte en todos los sitios).

**Modos de fallos en SGBDD**

Los sistemas distribuidos pueden sufrir los mismos tipos de fallos que los sistemas centralizados: errores de software, errores de hardware, fallos de discos, pero hay más fallos propios de entornos distribuidos: fallo de un sitio, pérdida de mensajes entre nodos, fallo de un enlace de comunicación, división de la red (grupos de nodos conectados por separado).

**Protocolos de compromiso**

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

Compromiso de dos fases (C2F): Al comprometer un dato, se aplica bloqueo.

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

* Fase 1: añade al registro al registro histórico y obliga a guardar el registro histórico en un lugar de almacenamiento estable (Log). Entonces envía un mensaje a todos los sitios donde se ha ejecutado . El gestor de transacciones del sitio determina si desea comprometer su parte de . Si la respuesta es negativa, añade un registro al registro histórico y responde enviando a el mensaje de abortar . Si la respuesta es sí, añade un registro al registro histórico y obliga a que el registro histórico se guarde en un almacenamiento estable. El gestor de transacciones contesta con el mensaje .
* Fase 2: Cuando recibe las respuestas al mensaje de preparar de todos los sitios, o cuando ha transcurrido un intervalo de tiempo especificado, puede determinar si la transacción puede comprometerse o abortarse. En función del resultado, se añade al registro histórico un registro o y se obliga a que el registro se guarde en almacenamiento estable. A partir de este momento, el coordinador envía un mensaje o a todos los sitios participantes. Cuando un sitio recibe este mensaje, lo guarda en su registro histórico.

Trato de Fallos

* Fallo de un sitio participante: Si el coordinador detecta que un sitio ha fallado emprende las acciones siguientes. Si falla antes de responder con el mensaje , el coordinador asume que ha respondido con el mensaje . Si el sitio falla después que el coordinador haya recibido del sitio el mensaje , el coordinador ejecuta el resto del protocolo ignorando el fallo del sitio.
* Cuando un sitio se recupera de un fallo debe examinar su registro histórico para determinar el destino de las transacciones que se hallaban en ejecución cuando se produjo el fallo. Si T es una de esas transacciones, se toman en consideración cada uno de los casos posibles. (, )
* Fallo del coordinador: si el coordinador falla durante la ejecución del protocolo de compromiso para la transacción , los sitios deben decidir el destino de . En ciertos casos los sitios participantes no pueden decidir si comprometer o abortar , y deben esperar a la recuperación del coordinador que ha fallado. Esta situación puede hacer que los elementos de datos no estén disponibles, no solo en el sitio que fallo, sino en sitios activos, por bloqueos, ya que queda bloqueada mientras espera la recuperación del sitio coordinador.
* División de la red: cuando una red queda dividida hay dos posibilidades:
  + El coordinador y los sitios participantes siguen en una partición, por lo que el fallo no tiene ningún efecto sobre el protocolo de compromiso.
  + El coordinador y los participantes quedan en varias particiones. Esto se maneja como si los sitios que están en una partición distinta al coordinador hubieran fallado.

Compromiso de 3 Fases

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

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

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

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

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

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

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

**Sistemas Federados**

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

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

Ejemplos:

* Dos empresas, cada una con sus BD, que se fusionan o pasan a formar parte de un mismo holding (unión que puede ser temporaria o no, pero las BD siguen teniendo autonomía).
* Ministerios que quieren compartir sus datos.
* Provincias o territorios autónomos que desean acceder mutuamente a ciertos datos.
* Países de un mercado común que necesitan intercambiar datos.

Soluciones: Se debe seleccionar la mejor

* Integración manual: Consultar separadamente cada BD, e integrar manualmente las respuestas.

Las personas que lo realice deben saber qué BD están accesibles y qué datos hay en cada BD, deben saber descomponer la consulta en las consultas parciales a cada BD, deben conocer el modelo y el lenguaje de cada BD y deben saber integrar los resultados parciales para producir el resultado deseado.

*Viable sólo muy excepcionalmente.*

* Integración de datos: Crear una nueva BD que integre todos los datos de las preexistentes.

Se debe diseñar la nueva BD (no necesariamente distribuida), convertir los programas, transferir los datos de las BD preexistentes a la nueva y preparar y enseñar los nuevos modos de trabajar de los usuarios.

*La gestión autónoma de cada BD se pierde en aras de una gestión única.*

* Integración del acceso: Construir un Sistema Federado en el que las BD interoperen.

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

*Se preserva la autonomía de cada BD.*

* Crear un Data Warehouse: es una colección de datos orientada a un determinado ámbito, integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza. Se trata de un expediente completo de una organización, más allá de la información transaccional y operacional, almacenado en una BD diseñada para favorecer el análisis y la divulgación eficiente de datos.

Autonomía y Flexibilidad

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

* Se pierde totalmente en la integración de datos en una BDD.
* Se preserva totalmente en la integración manual.
* Se obtienen grados intermedios en BD federadas (acceso integrado).

Se pueden obtener las mismas conclusiones respecto a la flexibilidad de añadir más BD. Cualquiera de las soluciones debe superar dificultades técnicas importantes.

En muchos casos es preferible el acceso integrado, por razones de viabilidad, preservación de autonomía, flexibilidad y evolución.

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

**Sistemas de BD Federados**

Dos niveles:

* El de las BD preexistentes, que denominaremos BD componentes: nivel componente.
* El del conjunto de BD que interoperan, que llamaremos nivel federado.

Los sistemas de BD componentes se federan para dar lugar a un Sistema de BD Federado (FDBS). Un FDBS consiste en una colección de Sistemas de BD componentes que son cooperantes y autónomos, interoperan según las necesidades de los usuarios de la federación y el deseo de los administradores de las BD en participar y compartir sus datos, y además son manipulados y controlados de manera coordinada por un software llamado Sistema de Gestión de BD Federadas (FDBMS).

**Características de los SBDF**

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

**Autonomía**

Los SBD estaban normalmente controlados de forma separada e independiente. Se comparten datos únicamente si se retiene el control. La autonomía de cada componente puede ser de diseño, comunicaciones, ejecución o asociación.

Diseño: Principal fuente de heterogeneidad. Capacidad para elegir su propio diseño en todos sentidos:

* + Universo de discurso: qué.
  + Conceptualización: qué conceptos.
  + Representación (modelo, lenguaje) y nombres: cómo.
  + Comportamiento (operaciones, restricciones).
  + Implementación: hard, soft, SGBD, esquema interno, etc.

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

Ejecución: Capacidad de cada BD componente para:

* Decidir el orden en que son ejecutadas las operaciones
* Ejecutar operaciones sin interferencia de operaciones externas
* Abortar cualquier operación que no cumpla con las restricciones propias
* Que las operaciones a nivel componente no se afecten lógicamente con la participación en un FDBS
* No tener que informar a un sistema externo el orden de ejecución

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

Asociación: Capacidad de un SGBD para decidir con quién compartir, qué compartir, cómo compartir y si asociarse o desasociarse de una federación.

Puede considerarse parte de la autonomía de diseño.

**Semejanzas y diferencias entre un SBDF y SBDD**

Semejanzas:

* Hay datos en varias instalaciones.
* Las instalaciones están interconectadas: sistema distribuido.
* Hay dos niveles.
* Se formula una pregunta y se obtiene una respuesta consistente.

Diferencias:

* De diseño (top-down vs. bottom-up). Las DBB van de lo más general a los más específico. En las BDF es al revés.
* De terminología (global/local – federado/componente).
* De autonomía. En las BDF el diseño es uno.
* De transparencias. En BDF se tiene que saber qué comparte cada uno de las BD.
* De arquitectura. BBD nodos de igual jerarquía.

En FDBS la “distribución” es una consecuencia de la autonomía.

**Construcción de un SBDF**

El proceso de negociación:

* Hay dos casos: para formar una federación nueva o para incorporar una BD componente a un FDBS existente.
* El administrador de la BD componente cede acceso a datos a cambio de poder acceder a otros datos y siguiendo órdenes de rango superior.
* La BD pierde autonomía: debe obedecer protocolos respecto a modificar su esquema nativo y a abandonar el FDBS.
* Si el FDBS es fuertemente acoplado, la negociación es más compleja.
* La complejidad de los sistemas de seguridad de las BD plantea dificultades.

Proceso de integración de esquemas:

* Seleccionar un conjunto de esquemas de exportación a integrar
* Integrarlos para producir un esquema federado
* Generar las correspondencias entre esquemas (directorio)
* Desarrollar el procesador de construcción: operaciones sobre el esquema Federado generan operaciones sobre los esquemas Exportación y viceversa.

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

**Motivación BD Distribuidas**

Las organizaciones necesitan acceder a los datos que se encuentra en diferentes sitios. Los avances en las tecnologías de comunicación que permiten un flujo constante entre diferentes sitios. La creciente globalización de las empresas y la necesidad de compartir información con clientes y proveedores.

Antes había problemas en las redes de comunicaciones por lo que la BD podía quedar inaccesible. Ahora, con la mejora de las redes de comunicación, se prefiere cliente-servidor con BD centralizada y procesamiento distribuido.

Ahora se utiliza réplica en un mismo data center, porque en distribuido trae problemas.

Ventajas:

* Las empresas ya están distribuidas, al menos físicamente, y ven como algo natural el enfoque de BDD que refleja la estructura organizacional.
* El uso más frecuente de los datos almacenados localmente brinda una mayor eficiencia en las transacciones.
* Crecimiento proporcional, cuando se suman nuevas sucursales o sedes de las organizaciones, las BDD proporcionan una solución con mínimo impacto.
* Se reduce la sobrecarga de las comunicaciones, ya que mayormente, una alta proporción de las transacciones se resuelve de manera local.
* Mejora el rendimiento, teniendo un alto grado de paralelismo, la carga es distribuida entre los diferentes nodos/sitios.
* Mayor confiabilidad y disponibilidad, al tener redundancia de datos, lo que brinda una alta disponibilidad de los sistemas. Esto requiere además mayores controles.

**Comparación entre BDC y BDD**

|  |  |  |
| --- | --- | --- |
|  | **BD CENTRALIZADAS** | **BD DISTRIBUIDAS** |
| **Definición** | Los SBDC se ejecutan en un único DBMS.  Son accedidos desde diferente terminales o sistemas, pero todas las transacciones se resuelven sobre ese único software de administración. | Conjunto de múltiples BD relacionadas lógicamente, las cuales se encuentran distribuidas en diferentes espacios lógicos e interconectados por una red de comunicaciones. Tienen la capacidad de realizar procesamiento autónomo, esto permite realizar operaciones locales o distribuidas.  Los múltiples sitios de BD están ligados por un sistema de comunicaciones de tal forma que un usuario en cualquier sitio puede acceder los datos en cualquier parte de la red exactamente como si estos fueran accedidos de forma local. |
| **Ventajas** | Se evita la redundancia: En sistemas que no usan BDC, cada aplicación tiene sus propios archivos privados o se encuentran en diferentes localidades. Esto origina enorme redundancia en los datos almacenados, así como desperdicio del espacio de almacenamiento.  Se evita la inconsistencia: ya que, si un hecho específico se representa por una sola entrada, la no concordancia de datos no puede ocurrir. Pueden hacerse cumplir las normas establecidas. Con un control central de la BD, el DBA puede garantizar que se cumplan todas las formas aplicables a la representación de los datos.  Pueden aplicarse restricciones de seguridad. El DBA puede:   * Asegurar que el único medio de acceder la BD sea a través de los canales establecidos. * Definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles. | Autonomía local: Un sitio puede controlar los datos que le pertenecen.  Disponibilidad: Un fallo en una parte del sistema solo afectará a un fragmento, no a toda la BD.  Rendimiento: Los datos generalmente se ubican cerca del sitio con mayor demanda, también los sistemas trabajan en paralelo, lo cual permite balancear la carga en los servidores.  Economía: Es más barato crear una red de muchas computadoras pequeñas, que tener una sola computadora muy poderosa.  Modularidad: Se pueden modificar, agregar o quitar sistemas de la BD distribuida sin afectar a los demás sistemas (módulos). |
| **Desventajas** | * Los mainframes no ofrecen mejor proporción precio/rendimiento que los microprocesadores de los sistemas distribuidos. * Cuando un SBDC falla, se pierde toda la disponibilidad de procesamiento y sobre todo de la información confiada al sistema. * Las cargas de trabajo no se pueden difundir entre diferentes computadoras, ya que los trabajos siempre se ejecutarán en la misma máquina. | * Costo de desarrollo del SW: Es complejo mantener la coordinación entre nodos. * Mayor probabilidad de errores: Como los nodos del sistema funcionan en paralelo, es más difícil asegurar el funcionamiento correcto de los algoritmos, así como de los procedimientos de recuperación de fallos del sistema. * Mayor sobrecarga de procesamiento: El intercambio de mensajes y ejecución de algoritmos para el mantenimiento de la coordinación entre nodos supone una sobrecarga que no se da en los SBDC. |

Estructura cliente-servidor: es en parte centralizado y en parte distribuido. La BD está en un solo servidor pero el procesamiento se puede hacer en diferentes servidores ya que la lógica de negocio está en una capa por arriba de la BD (aplicación).

**Tareas de Administración de una BDD**

Actividades típicas de toda BD: seguridad, integridad, optimización, etc.

Actividades específicas de una BDD: descripción del esquema global, descripción de la fragmentación/replicación, descripción de las localizaciones, estadísticas de la BD (para optimizar consultas).

**Pasos a seguir para diseñar una BDD**

1. Diseño del "*esquema conceptual*": describe la BD integrada (todos los datos que son utilizados por las aplicaciones que tienen acceso a las BD).
2. Diseño "*físico*" de la BD: mapear el esquema conceptual a las áreas de almacenamiento y determinar los métodos de acceso.
3. Diseño de la *fragmentación* lógica: determinar la forma en que las relaciones globales se subdividen en fragmentos horizontales, verticales o mixtos.
4. Diseño de la *asignación de los fragmentos*: determinar la forma en que los fragmentos se mapean a las imágenes físicas.

**Ejemplo de procesamiento distribuido de consultas**

* NODO1: EMPLEADO: 1.000.000 bytes
* NODO2: DEPARTAMENTO: 3500 bytes

Caso 1: *“Por cada empleado, obtener el nombre del empleado y el nombre del departamento al que pertenece”*

: ΠNombre,Apellido,NombreDPto(EMPLEADO ⋈ DEPARTAMENTO)

▪ La consulta se lanza desde el nodo3 (nodo respuesta) que no tiene datos implicados en la consulta.

▪ El resultado de ésta consulta constará de 10.000 tuplas. Cada tupla resultante será de una longitud de 40 bytes. El tamaño del resultado será por tanto de 400.000 bytes.

▪ Existen tres alternativas para resolver la consulta:

* Primera alternativa: Transferir, tanto la relación EMPLEADO, como la relación DEPARTAMENTO al nodo respuesta (nodo3) y realizar allí mismo la operación de join. En éste caso se transfieren: 1.000.000 + 3.500 = 1.003.500 bytes.
* Segunda alternativa: Transferir la relación EMPLEADO al nodo2, ejecutar el join en este nodo y enviar el resultado al nodo3. Esto implicaría transferir: 1.000.000 + 400.000 (resultado) = 1.400.000 bytes
* Tercera alternativa: Transferir la relación DEPARTAMENTO al nodo1, ejecutar el join en este nodo y enviar el resultado al nodo3. En este caso, los bytes transferidos serán: 3.500 + 400.000 (resultado) = 403.500 bytes.

Caso 2: *“Para cada departamento, obtener el nombre del departamento y el de su director”*

La consulta se lanza desde el nodo3. El resultado de ésta consulta constará de 100 tuplas (4.000 bytes).

* Opción 1: transferimos las relaciones DEPARTAMENTO y EMPLEADO al nodo3. Se transfieren: 3.500 + 1.000.000 = 1.003.500 bytes.
* Opción 2: transferimos la relación EMPLEADO al nodo2 y enviamos el resultado del join al nodo3. Se transfieren: 1.000.000 + 4.000 = 1.004.000 bytes.
* Opción 3: transferimos la relación DEPARTAMENTO al nodo1 y enviamos el resultado del join al nodo3. Se transfieren: 3.500 + 4.000 = 7.500 bytes.

**Procesamiento distribuido de consultas utilizando semijoin**

Reducción del número de columnas antes de transferir a otro nodo.

* Se envía la columna con la que se va a realizar el join de una relación R al nodo donde se encuentra la otra relación, allí se realiza el join con la otra relación S.
* Se envían las columnas implicadas en el resultado al nodo inicial y se vuelve a realizar el join con R.
* Sólo se transfieren las columnas de R que intervienen en la realización del join en una dirección y el subconjunto de columnas de S resultantes en la otra.