

Tema 6

Sistema de gestión de base de datos (SGBD)

Contenido

- 0 Definición
- 1 Propósito
- 2 Objetivos
- 3 Ventajas
- 4 Inconvenientes
- 5 Productos SGBD disponibles en el mercado
 - 5.1 SGBD libres
 - 5.2 SGBD gratuitos
 - 5.3 ejemplos SGBD comerciales

→ 6.5 Clasificación de los S.G.B.D.

Componentes y Arquitectura de los SGBD

6.7 S.G.B. Datos Distribuidos

6.8 Lenguajes SGBD

6.1 Los sistemas de gestión de base de datos (SGBD); (en inglés: *Database management system*, abreviado **DBMS**) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Propósito

Su función es la de gestionar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos.

6.2 Objetivos

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información.** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios *niveles de abstracción*.
- **Independencia.** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Consistencia.** En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería

- aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.
- **Seguridad.** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad.** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada. Los SGBD proveen mecanismos para garantizar la recuperación de la base de datos hasta un estado consistente (ver Consistencia, más arriba) conocido en forma automática.
- **Respaldo.** Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- **Control de la concurrencia.** En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- **Manejo de Transacciones.** Una Transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que el estado luego de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.
- **Tiempo de respuesta.** Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

5-3 Ventajas

- Proveen facilidades para la manipulación de grandes volúmenes de datos.:
 - Simplifican la programación de chequeos de consistencia.
 - Manejando las políticas de respaldo adecuadas garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores en el disco, o hay muchos usuarios accediendo simultáneamente a los mismos datos, o se ejecutaron programas que no terminaron su trabajo correctamente, etc.
 - Permiten realizar modificaciones en la organización de los datos con un impacto mínimo en el código de los programas.
 - Permiten implementar un manejo centralizado de la seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.

- Las facilidades anteriores bajan drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.
- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

6.4 Inconvenientes

1. Típicamente, es necesario disponer de una o más personas que administren de la base de datos, en la misma forma en que suele ser necesario en instalaciones de cierto porte disponer de una o más personas que administren de los sistemas operativos. Esto puede llegar a incrementar los costos de operación en una empresa. Sin embargo hay que balancear este aspecto con la calidad y confiabilidad del sistema que se obtiene.
2. Si se tienen muy pocos datos que son usados por un único usuario por vez y no hay que realizar consultas complejas sobre los datos, entonces es posible que sea mejor usar una planilla de cálculo.
3. Complejidad: los SGBD son software muy complejos y las personas que vayan a usarlo deben tener conocimiento de las funcionalidades del mismo para poder aprovecharlo al máximo.
4. Tamaño: la complejidad y la gran cantidad de funciones que tienen hacen que sea un software de gran tamaño, que requiere de gran cantidad de memoria para poder correr.
5. Coste del hardware adicional: los requisitos de hardware para correr un SGBD por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero.

6.5 Clasificación de los Sistemas de Gestión de Base de Datos

Los sistemas de Gestión de Base de Datos se clasifican según:

Modelo lógico en el que se basan:

- Modelo Jerárquico.
- Modelo de Red.
- Modelo Relacional.
- Modelo Orientado a Objetos.

Número de usuarios

- Monousuario.
- Multiusuario.

Número de sitios

- Centralizados.
- Distribuidos: Homogéneos, Heterogéneos.

Ámbito de aplicación

- Propósito General.
- Propósito Específico

6.6 Productos SGBD disponibles en el mercado

SGBD libres

- PostgreSQL (<http://www.postgresql.org> Postgresql) Licencia BSD
- Firebird basada en la versión 6 de InterBase, Initial Developer's PUBLIC LICENSE Version 1.0.
- SQLite (<http://www.sqlite.org> SQLite) Licencia Dominio Público
- DB2 Express-C (<http://www.ibm.com/software/data/db2/express/>)
- Apache Derby (<http://db.apache.org/derby/>)

SGBD no libres

- MySQL: Licencia Dual, depende del uso. No se sabe hasta cuándo permanecerá así, ya que ha sido comprada por Oracle. Sin embargo, existen 2 versiones: una gratuita que sería equivalente a la edición "express" SQL server de Microsoft Windows, y otra más completa de pago.
- Advantage Database
- dBase
- FileMaker
- Fox Pro
- gsBase
- IBM DB2: Universal Database (DB2 UDB)
- IBM Informix
- Interbase de CodeGear, filial de Borland
- MAGIC
- Microsoft Access
- Microsoft SQL Server
- NexusDB
- Open Access
- Oracle
- Paradox
- PervasiveSQL
- Progress (DBMS)
- Sybase ASE
- Sybase ASA
- Sybase IQ
- WindowBase
- IBM IMS Base de Datos Jerárquica
- CA-IDMS

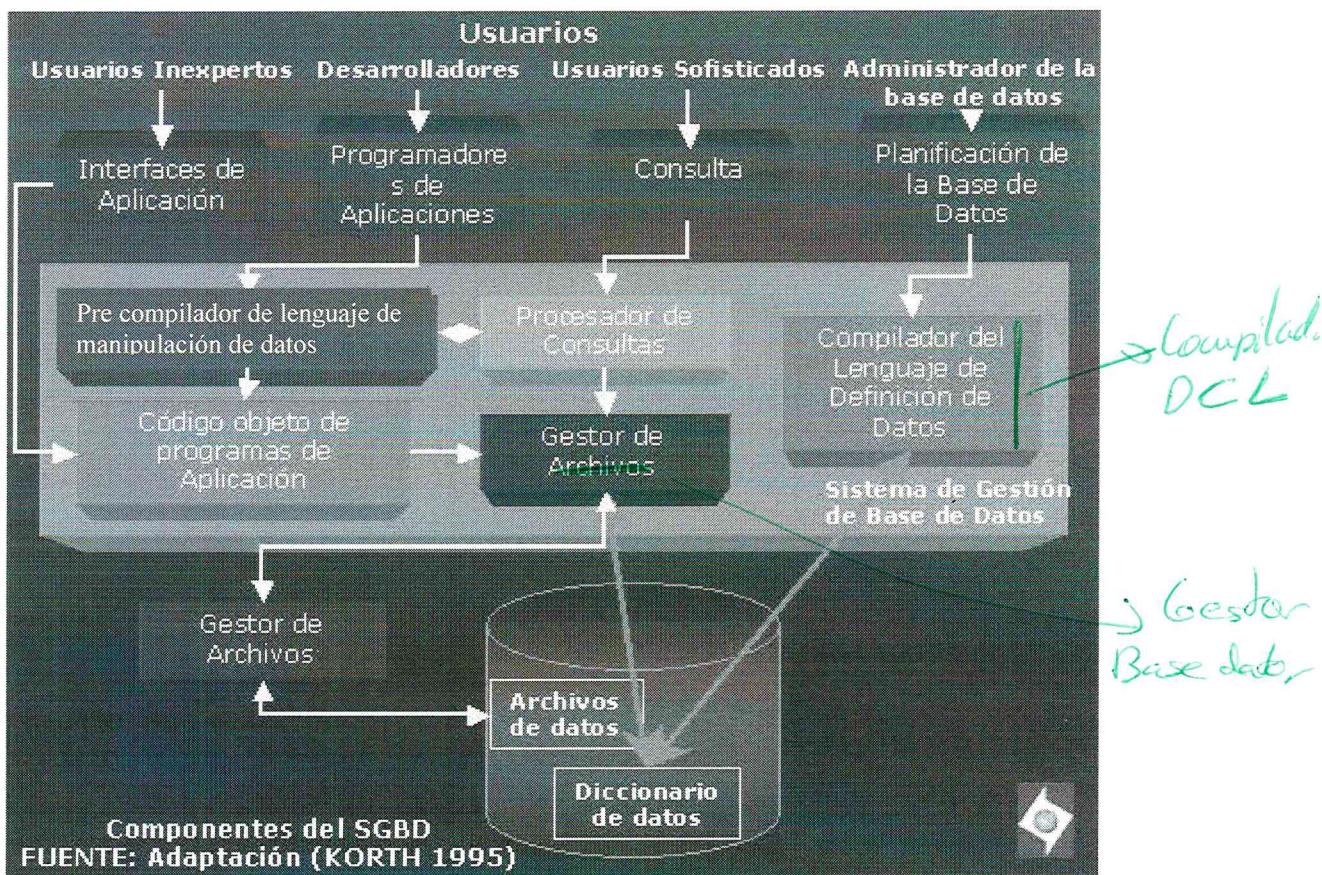
SGBD no libres y gratuitos

- Microsoft SQL Server Express
- Sybase ASE Express Edition para Linux (edición gratuita para Linux)

6.7 Componentes y Arquitectura de los SGBD

6.7.1 Componentes de un Sistema de Gestión de Base de Datos.

Un Sistema de Gestión de Base de Datos se divide en módulos que tratan cada una de las responsabilidades del sistema general. Los componentes funcionales de un SGBD (KORTH, 1995) incluyen:



- **Procesador de Consultas.** Traduce sentencias en un lenguaje de consultas a instrucciones de bajo nivel que entiende el gestor de la base de datos.
- **Gestor de la Base de Datos.** Proporciona la interfase entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas que se hacen en el sistema.
- **Gestor de Archivos.** Gestiona la asignación de espacio en la memoria del disco y de las estructuras de datos usadas para representar la información almacenada en disco.
- **Pre compilador del Lenguaje de Manipulación de Datos DML.** Convierte las sentencias en DML incorporadas en un programa de aplicación en llamadas normales a procedimientos en el lenguaje principal.
- **Compilador del Lenguaje de Definición de Datos DDL.** Convierte sentencias en DDL en un conjunto de tablas metadatos o "datos sobre datos".
- **Gestor del Diccionario de Datos.** Almacena metadatos sobre la estructura de la base de datos.

- **Gestor del Diccionario de Datos.** Almacena metadatos sobre la estructura de la base de datos.

El diccionario de datos

El **diccionario de datos** es el lugar donde se deposita información acerca de todos los datos que forman la BD. Es una guía en la que se describe la BD y los objetos que la forman.

El diccionario contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información.

En una BD relacional, el diccionario de datos proporciona información acerca de:

- La estructura lógica y física de la BD.
- Las definiciones de todos los objetos de la BD: tablas, vistas, índices, disparadores, procedimientos, funciones, etcétera.
- El espacio asignado y utilizado por los objetos.
- Los valores por defecto de las columnas de las tablas
- Información acerca de las restricciones de integridad.
- Los privilegios y roles otorgados a los usuarios.
- Auditoría de información, como los accesos a los objetos.

Un diccionario de datos debe cumplir las siguientes características:

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

6.7.2 Arquitectura de los sistemas de bases de datos

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

- **Nivel interno o físico:** el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la BD mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera.
- **Nivel externo o de visión:** es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la BD que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios.
- **Nivel conceptual:** describe la estructura de toda la BD para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento. Representa la información contenida en la BD. En la Figura 1.1 se representan los niveles de abstracción de la arquitectura ANSI.

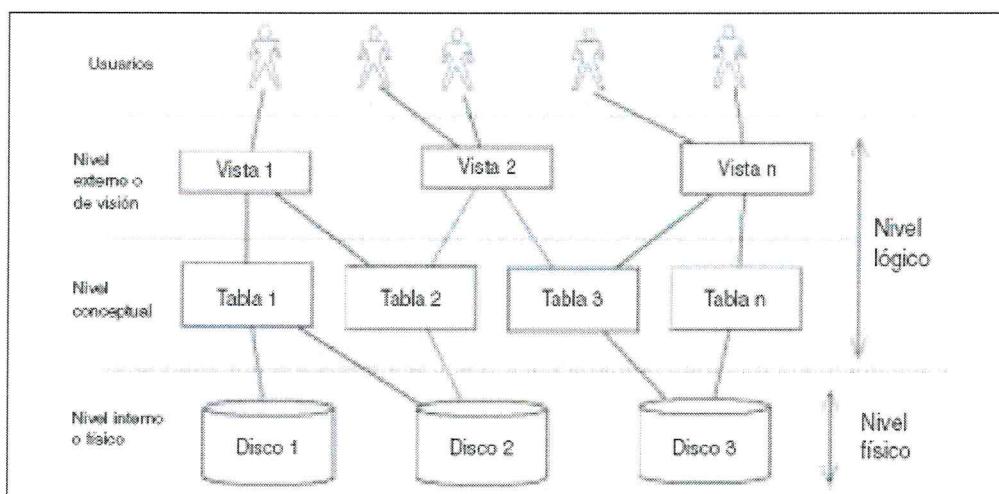


Figura 1.1. Niveles de abstracción de la arquitectura ANSI.

Esta arquitectura describe los datos a tres niveles de abstracción. En realidad los únicos datos que existen están a nivel físico almacenados en discos u otros dispositivos. Los SGBD basados en esta arquitectura permiten que cada grupo

de usuarios haga referencia a su propio esquema externo. El SGBD debe de transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos de esquema conceptual, para finalmente ser una petición expresada en el esquema interno que se procesará sobre la BD almacenada. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación, el SGBD es capaz de interpretar una solicitud de datos y realiza los siguientes pasos:

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

Para una BD específica sólo hay un esquema interno y uno conceptual, pero puede haber varios esquemas externos definidos para uno o para varios usuarios.

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

- Independencia lógica: la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se podrá modificar el esquema conceptual para ampliar la BD o para reducirla, por ejemplo, si se elimina una entidad, los esquemas externos que no se refieran a ella no se verán afectados.

- Independencia física: la capacidad de modificar el esquema interno sin tener que alterar ni el esquema conceptual, ni los externos. Por ejemplo, se pueden reorganizar los archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización, o se pueden añadir nuevos archivos de datos porque los que había se han llenado. La independencia física es más fácil de conseguir que la lógica, pues se refiere a la separación entre las aplicaciones y las estructuras físicas de almacenamiento.

En los SGBD basados en arquitecturas de varios niveles se hace necesario ampliar el catálogo o el diccionario de datos para incluir la información sobre cómo establecer las correspondencias entre las peticiones de los usuarios y los datos, entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el diccionario. La independencia de los datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios. Sólo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

6.8 SGBD - Distribuidos

6.8.1 Bases de datos distribuidas

Una **base de datos distribuida** (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas en diferentes espacios lógicos (pej. un servidor corriendo 2 máquinas virtuales) e interconectados por una red de comunicaciones. Dichas BDD tienen la capacidad de realizar procesamiento autónomo, esto permite realizar operaciones locales o distribuidas. Un sistema de Bases de Datos Distribuida (SBDD) es un sistema en el cual múltiples sitios de bases de datos 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.

Un sistema distribuido de bases de datos se almacenan en varias computadoras. Los principales factores que distinguen un SBDD de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben de estar comunicados por medio de algún tipo de red de comunicaciones para transmitir datos y órdenes entre los sitios.

Introducción

La necesidad de almacenar datos de forma masiva dio paso a la creación de los sistemas de bases de datos. En 1970 Edgar Frank Codd escribió un artículo con nombre: "A Relational Model of Data for Large Shared Data Banks" ("Un modelo relacional para grandes bancos de datos compartidos"). Con este artículo y otras publicaciones, definió el modelo de bases de datos relacionales y reglas para poder evaluar un administrador de bases de datos relacionales.

6.8.2 Componentes

Hardware involucrado

El hardware utilizado no difiere mucho del hardware utilizado en un servidor normal. Al principio se creía que si los componentes de una base de datos eran especializados serían más eficientes y rápidos, pero se comprobó que el descentralizar todo y adoptar un enfoque "nada compartido" (*shared-nothing*) resultaba más barato y eficaz. Por lo que el hardware que compone una base de datos distribuida se reduce a servidores y la red.

Software

- **Sistema Manejador de Base de Datos Distribuida (DDBMS)**

Este sistema está formado por las transacciones y los administradores de la base de datos distribuidos. Un DDBMS implica un conjunto de programas que operan en diversas computadoras, estos programas pueden ser subsistemas de un único DDBMS de un fabricante o podría consistir de una colección de programas de diferentes fuentes.

- **Administrador de transacciones distribuidas (DTM)**

↑
el manejador
de transacc

Este es un programa que recibe las solicitudes de procesamiento de los programas de consulta o transacciones y las traduce en acciones para los administradores de la base de datos. Los DTM se encargan de coordinar y controlar estas acciones. Este DTM puede ser propietario o desarrollado en casa.

Manejador de transacciones distribuido (DTM)

Definición de transacciones

Una transacción es una secuencia de una o más operaciones agrupadas como una unidad. El inicio y el final de la transacción definen los puntos de consistencia de la base de datos. Si una acción de la transacción no se puede ejecutar, entonces ninguna acción dentro de la secuencia que conforma la transacción tendrá efecto.

Propiedades de las transacciones

- **Atomicidad:** Una transacción es una unidad atómica de procesamiento, esta se realiza o no se realiza.
- **Consistencia:** Si se ejecuta una transacción sobre un estado consistente, el resultado será un nuevo estado consistente.
- **Aislamiento:** Una transacción no hará visibles sus modificaciones a otras transacciones hasta que termine de ejecutarse completamente. Es decir, una transacción desconoce si otras transacciones se están ejecutando en el sistema.
- **Durabilidad:** Una vez una transacción se ejecuta exitosamente y realiza cambios sobre el sistema, estos cambios nunca se deben perder a causa de fallas en el sistema.

Tipos de transacciones

Una transacción puede clasificarse de diferentes maneras dependiendo básicamente de tres criterios:

1. **Áreas de aplicación.** En primer lugar, las transacciones se pueden ejecutar en aplicaciones no distribuidas. Las transacciones que operan en datos distribuidos se les conoce como transacciones distribuidas. Por otro lado, dado que los resultados de una transacción que realiza un commit son durables, la única forma de deshacer los efectos de una transacción con commit es mediante otra transacción. A este tipo de transacciones se les conoce como transacciones compensatorias. Finalmente, en ambientes heterogéneos se presentan transacciones heterogéneas sobre los datos.
2. **Tiempo de duración.** Tomando en cuenta el tiempo que transcurre desde que se inicia una transacción hasta que se realiza un commit o se aborta, las transacciones pueden ser de tipo batch o en línea. Estas se pueden diferenciar también como transacciones de corta y larga vida. Las transacciones en línea se caracterizan por tiempos de respuesta muy

cortos y por acceder un porción relativamente pequeña de la base de datos. Por otro lado, las transacciones de tipo batch toman tiempos relativamente largos y accedan grandes porciones de la base de datos.

3. **Estructura.** Considerando la estructura que puede tener una transacción se examinan dos aspectos: si una transacción puede contener a su vez subtransacciones o el orden de las acciones de lectura y escritura dentro de una transacción.

Función del manejador

El manejador de transacciones es el encargado de definir la estructura de las transacciones, mantener la consistencia en la base de datos cuando se ejecuta una transacción o se cancela la ejecución de una, mantener protocolos de fiabilidad, implementar algoritmos para el control de la concurrencia y sincronizar las transacciones que se ejecutan simultáneamente.

El manejador recibe solicitudes de procesamiento de transacciones y las traduce en acciones para el calendarizador.

La operación COMMIT señala el término exitoso de la transacción: le dice al manejador de transacciones que se ha finalizado con éxito una unidad lógica de trabajo, que la base de datos esta (o debería estar) de nuevo en un estado consistente, y que se pueden hacer permanentes todas las modificaciones efectuadas por esa unidad de trabajo.

La operación ROLLBACK, en cambio, señala el término no exitoso de la transacción: le dice al manejador de transacciones que algo salió mal, que la base de datos podría estar en un estado inconsistente y que todas las modificaciones efectuadas hasta el momento por la unidad lógica de trabajo deben retroceder o anularse.

- **Sistema Manejador de base de datos (DBMS)**

Es un programa que procesa cierta porción de la base de datos distribuida. Se encarga de recuperar y actualizar datos del usuario y generales de acuerdo con los comandos recibidos de los DTM.

Nodos

Un nodo es una computadora que ejecuta un DTM o un DBM o ambos. Un nodo de transacción ejecuta un DTM y un nodo de base de datos ejecuta un DBM.

6.83 Objetivos de Implementación

Al implementar una base de datos distribuida se tienen ciertos objetivos comunes:

- **Transparencia de ubicación.** Permite a los usuarios tener acceso a los datos sin que tenga conocimiento de la ubicación de éstos. Se puede conseguir este nivel de transparencia al utilizar los administradores de transacciones distribuidas, los cuales son capaces de determinar la localización de los datos y de emitir acciones a los calendarizadores apropiados, lo cual puede ejecutarse cuando los administradores de transacciones distribuidas poseen acceso a los directorios de localizaciones de los datos.
- **Transparencia de duplicación.** Se debe manejar la base de datos, sin saber que elementos están duplicados.
- **Transparencia de concurrencia.** Cuando varias transacciones se ejecuten al mismo tiempo, los resultados de las transacciones no deberán afectarse. La transparencia de concurrencia se logra si los resultados de todas las transacciones concurrentes son consistentes de manera lógica con los resultados que se habrían obtenido si las transacciones se hubieran ejecutado una por una, en cualquier orden secuencial.
- **Transparencia de fallos.** Significa que a pesar de fallas las transacciones sean procesadas de un modo correcto. Frente a una falla, las transacciones deben ser atómicas, significa que se procesen todas o ninguna de ellas. Para este tipo de problemas es importante tener resguardo de la base de datos, y así poder restaurarla cuando sea conveniente. El sistema debe detectar cuándo falla una localidad y tomar las medidas necesarias para recuperarse del fallo. El sistema no debe seguir utilizando la localidad que falló. Por último, cuando se recupere o repare esta localidad, debe contarse con mecanismos para reintegrarla al sistema con el mínimo de complicaciones.
- **Localidad del procesamiento.** Los datos se deben distribuir lo más cerca posible de las aplicaciones que los usan para maximizar la localidad del procesamiento, este principio responde a minimizar el acceso remoto a los datos. Diseñar una distribución que maximice localidad del procesamiento puede hacerse añadiendo la cantidad de referencias locales y remotas correspondientes a cada fragmentación candidata y asignar la fragmentación eligiendo la mejor solución. Independencia de configuración. La independencia de configuración permite añadir o reemplazar hardware sin tener que cambiar componentes de software existentes en el sistema de base de datos distribuida.
- **Particionado de la Base de Datos.** La base de datos se distribuye de modo que no haya solapamiento o duplicación de los datos mantenidos en las diferentes localidades, como no hay duplicaciones de los datos, se evitan los costos asociados con el almacenamiento y mantenimiento de datos redundantes. Si un mismo segmento de datos se usa en más de una localidad se ve limitada la

disponibilidad de los datos. La fiabilidad también puede verse limitada cuando se produce un fallo en el sistema de cálculo de una localidad se afecta la disponibilidad de los datos de esa localidad no estén disponible para los usuarios en cualquier parte del sistema.

- **Fragmentación de datos.** Consiste en subdividir las relaciones y distribuirlas entre los sitios de la red, tiene como objetivo buscar formas alternativas de dividir una las instancias (tablas) de relaciones en otras más pequeñas. La fragmentación se puede realizar por tuplas individuales (fragmentación horizontal), por atributos individuales fragmentación vertical) o una combinación de ambas (fragmentación híbrida). El principal problema de la fragmentación radica en encontrar la unidad apropiada de distribución. Una relación no es una buena unidad por muchas razones. Normalmente las vistas de una relación están formadas por subconjuntos de relaciones. Además, las aplicaciones acceden localmente a subconjuntos de relaciones. Por ello, es necesario considerar a los subconjuntos de relaciones como unidad de distribución. Al descomponer de una relación en fragmentos, tratados cada uno de ellos como una unidad de distribución, permite el proceso concurrente de las transacciones. El conjunto de estas relaciones, provocará la ejecución paralela de una consulta al ser dividida en una serie de subconsultas que operará sobre los fragmentos. Cuando las vistas definidas sobre una relación son consideradas como unidad de distribución que se ubican en diferentes sitios de la red, podemos optar por dos alternativas diferentes: La relación no estará replicada y se almacena en un único sitio, o existe réplica en todos o algunos de los sitios en los cuales reside la aplicación. Las consecuencias de esta estrategia son la generación de un volumen de accesos remotos que pueden ser innecesarios con un mal manejo de estas replicas. Además, las réplicas innecesarias pueden causar problemas en la ejecución de las actualizaciones y puede no ser deseable si el espacio de almacenamiento está limitado. Los inconvenientes de la fragmentación están dados en que si las pueden estar definidas por fragmentos mutuamente exclusivos y al recuperar los datos de dos fragmentos situados en sitios diferentes es necesario trasmitir los datos de un sitio a otro y realizar sobre ellos la operación de unión (Join), lo cual puede ser costoso. El control semántico cuando los atributos implicados en una dependencia una relación se descompone en diferentes fragmentos y estos se ubican en sitios diferentes puede ser muy costos porque es necesario hacer búsquedas en un gran número de sitios.

6.8.4 Ventajas y Desventajas de las bbddd

Ventajas

- Refleja una estructura organizacional - los fragmentos de la base de datos se ubican en los departamentos a los que tienen relación.
- Autonomía local - un departamento puede controlar los datos que le pertenecen.
- Disponibilidad - un fallo en una parte del sistema solo afectará a un fragmento, en lugar de a toda la base de datos.

- 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~~

- 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 base de datos distribuida sin afectar a los demás sistemas (módulos).

Desventajas

- Complejidad - Se debe asegurar que la base de datos sea transparente, se debe lidiar con varios sistemas diferentes que pueden presentar dificultades únicas. El diseño de la base de datos se tiene que trabajar tomando en cuenta su naturaleza distribuida, por lo cual no podemos pensar en hacer joins que afecten varios sistemas.
- Economía - la complejidad y la infraestructura necesaria implica que se necesitará una mayor mano de obra.
- Seguridad - se debe trabajar en la seguridad de la infraestructura así como cada uno de los sistemas.
- Integridad - Se vuelve difícil mantener la integridad, aplicar las reglas de integridad a través de la red puede ser muy caro en términos de transmisión de datos.
- Falta de experiencia - las bases de datos distribuidas son un campo relativamente nuevo y poco común por lo cual no existe mucho personal con experiencia o conocimientos adecuados.
- Carencia de estándares - aún no existen herramientas o metodologías que ayuden a los usuarios a convertir un DBMS centralizado en un DBMS distribuido.
- Diseño de la base de datos se vuelve más complejo - además de las dificultades que generalmente se encuentran al diseñar una base de datos, el diseño de una base de datos distribuida debe considerar la fragmentación, replicación y ubicación de los fragmentos en sitios específicos

Consideraciones importantes

Detección de bloqueos y Concurrencia

- **Bloqueos**

Un bloqueo en general es cuando una acción que debe ser realizada está esperando a un evento. Para manejar los bloqueos hay distintos acercamientos: prevención, detección, y recuperación. También es necesario considerar factores como que hay sistemas en los que permitir un bloqueo es inaceptable y catastrófico, y sistemas en los que la detección del bloqueo es demasiado costosa.

En el caso específico de las bases de datos distribuidas usar bloqueo de recursos, peticiones para probar, establecer o liberar bloqueos requiere mensajes entre los manejadores de transacciones y el calendarizador. Para esto existen dos formas básicas:

- Autónoma: cada nodo es responsable por sus propios bloqueos de recursos.
 - Una transacción sobre un elemento con n replicas requiere $5n$ mensajes
 - Petición del recurso
 - Aprobación de la petición
 - Mensaje de la transacción
 - Reconocimientos de transacción exitosa
 - Peticiones de liberación de recursos
- Copia Primaria: un nodo primario es responsable para todos los bloqueos de recursos
 - Una transacción sobre un elemento con n copias requiere $2n+3$ mensajes
 - Una petición del recurso
 - Una aprobación de la petición
 - n mensajes de la transacción
 - n reconocimientos de transacción exitosa
 - Una petición de liberación de recurso

Podemos definir que dos operaciones entran en conflicto que debe ser resuelto si ambas acceden a la misma data, y una de ellas es de escritura y si fueron realizadas por transacciones distintas.

- **Concurrencia**

El ejemplo más común de un bloqueo mutuo es cuando un recurso A está siendo utilizado por una transacción A que a su vez solicita un recurso B que está siendo utilizado por una transacción B que solicita el recurso A. Entre los ejemplos específicos para las bases de datos distribuidas podemos destacar:

- Actualización perdida: cuando dos transacciones concurrentes borran el efecto una de la otra
- La extracción inconsistente: acceder a información modificada parcialmente por una transacción

- **Soluciones**

El control de concurrencia y detección y manejo de bloqueos es un área de mucho estudio en las bases de datos distribuidas, a pesar de esto no hay ningún algoritmo aceptado para solucionar el problema. Esto se debe a varios factores de los cuales se consideran a los siguientes tres los más determinantes:

1. La data puede estar duplicada en un BDD, por tanto, el manejador de la BDD es responsable de localizar y actualizar la data duplicada.
2. Si un nodo falla o la comunicación con un nodo falla mientras se realiza una actualización, el manejador debe asegurarse de que los efectos se reflejen una vez el nodo se recupere del fallo.
3. La sincronización de transacciones en sitios o nodos múltiples es difícil ya que los nodos no pueden obtener información inmediata de las acciones realizadas en otros nodos concurrentemente.

Para el control de bloqueos mutuos no se ha desarrollado ninguna solución viable y la forma más simple y que la mayoría de productos utilizan es la implementación de un tiempo máximo de espera en las peticiones de bloqueos.

Causa de estas dificultades más de 20 algoritmos de control de concurrencia se han propuesto en el pasado, y aun así siguen apareciendo nuevos. Una revisión bibliográfica muestra que la mayoría de los algoritmos son variantes del 2PL (2-phase locking o bloqueo de dos fases) o el algoritmo de time-stamp. A continuación se explican estos dos algoritmos básicos.

- **Bloqueo de dos fases (2PL)**

El algoritmo 2PL utiliza bloqueos de lectura y escritura para prevenir conflictos entre operaciones. Consiste en los siguientes pasos para una transacción T:

1. Obtiene bloqueo de lectura para un elemento L (bloqueo compartido)
2. Obtiene bloqueo de escritura para un elemento E (bloqueo exclusivo)
3. Lee el elemento L
4. Escribe en el elemento E
5. Libera el bloqueo de L
6. Libera el bloqueo de E

6.8.5

Distribución de la base de datos

Una de las decisiones más importantes que el diseñador de bases de datos distribuidas debe tomar es el posicionamiento de la data en el sistema y el esquema bajo el cuál lo desea hacer. Para esto existen cuatro alternativas principales: centralizada, replicada, fragmentada, e híbrida. La forma centralizada es muy similar al modelo de Cliente/Servidor en el sentido que la BDD está centralizada en un lugar y los usuarios están distribuidos. Este modelo solo brinda la ventaja de tener el procesamiento distribuido ya que en sentido de disponibilidad y fiabilidad de los datos no se gana nada.

Replicadas

El esquema de BDD de replicación consiste en que cada nodo debe tener su copia completa de la base de datos. Es fácil ver que este esquema tiene un alto costo en el almacenamiento de la información. Debido a que la actualización de los datos debe ser realizada en todas las copias, también tiene un alto costo de escritura, pero todo esto vale la pena si tenemos un sistema en el que se va a escribir pocas veces y leer muchas, y dónde la disponibilidad y fiabilidad de los datos sea de máxima importancia.

Fragmentadas o Particionadas

Este modelo consiste en que solo hay una copia de cada elemento, pero la información está distribuida a través de los nodos. En cada nodo se aloja uno o más fragmentos disjuntos de la base de datos. Como los fragmentos no se replican esto disminuye el costo de almacenamiento, pero también sacrifica la disponibilidad y fiabilidad de los datos. Algo que se debe tomar en cuenta cuando se desea implementar este modelo es la granularidad de la fragmentación. La fragmentación se puede realizar también de tres formas:

- Horizontal: Los fragmentos son subconjuntos de una tabla (análogo a un restringir)
- Vertical: Los fragmentos son subconjuntos de los atributos con sus valores (análogo a un proyectar)
- Mixto: Se almacenan fragmentos producto de restringir y proyectar una tabla.

Para que una fragmentación sea correcta esta debe cumplir con las siguientes reglas:

- Debe ser **Completa**: Si una relación R se fragmenta en R_1, R_2, \dots, R_n , cada elemento de la data de R debe estar en algún R_i .
- Debe ser **Reconstruible**: Debe ser posible definir una operación relacional que a partir de los fragmentos obtenga la relación.
- Los fragmentos deben ser **Disjuntos**: Si la fragmentación es horizontal entonces si un elemento e está en R_i este elemento no puede estar en ningún R_k (para k distinto a i). En el caso de fragmentación vertical es necesario que se repitan las llaves primarias y esta condición solo se debe cumplir para el conjunto de atributos que no son llave primaria.

Criterios para escoger la distribución

- Localidad de la data: la data debería ser colocada donde ésta se accede más seguido. El diseñador debe analizar las aplicaciones y determinar como colocar la data de tal forma que se optimicen los accesos a la data locales.
- Fiabilidad de la data: Almacenando varias copias de la data en lugares geográficamente apartados se logra maximizar la probabilidad de que la data va a ser recuperable en caso de que ocurra daño físico en cualquier sitio.
- Disponibilidad de la data: como en la fiabilidad, almacenar varias copias asegura que los usuarios tengan a su disponibilidad los elementos de la data, aún si el nodo al que usualmente acceden no está disponible o falla.

- 
- Capacidades y costos de almacenamiento: a pesar de que los costos de almacenamiento no son tan grandes como los de transmisión, los nodos pueden tener diferentes capacidades de almacenamiento y procesamiento. Esto se debe analizar cuidadosamente para determinar donde poner la data. El costo de almacenamiento se disminuye significativamente minimizando la cantidad de copias de la data.
 - Distribución de la carga de procesamiento: una de las razones por la cual se escoge un sistema de BDD es porque se desea poder distribuir la carga de procesamiento para hacer este más eficiente.
 - Costo de comunicación: el diseñador debe considerar también el costo de usar las comunicaciones de la red para obtener data. Los costos de comunicación se minimizan cuando cada sitio tiene su propia copia de la data, por otro lado cuando la data es actualizada se debe actualizar en todos los nodos.
 - Uso del sistema: debe tomarse en consideración cual será el tipo principal de uso del sistema de BDD. Factores como la importancia en la disponibilidad de la data, la velocidad de escritura y la capacidad de recuperación de daños físicos deben tomarse en cuenta para escoger el esquema correcto.

Seguridad

Desde hace ya varios años las bases de datos son ampliamente utilizadas en departamentos de gobiernos, empresas comerciales, bancos, hospitales, etc. Actualmente se está cambiando el esquema bajo el cuál se utilizan las bases de datos, ya no son utilizadas únicamente de forma interna, sino que se tiene muchos accesos externos de tipos distintos. Estos cambios que se han introducido en el uso de las bases de datos ha creado la necesidad mejorar las prácticas de seguridad ya que el ambiente ya no es tan controlado como el esquema antiguo.

Conceptos

Los problemas de mayor importancia en seguridad son **autenticación, identificación, y refuerzo de los controles de acceso apropiados**. El sistema de seguridad de niveles múltiples. Éste consiste en muchos usuarios con distintos niveles de permisos para una misma base de datos con información de distintos niveles. En las bases de datos distribuidas se han investigado dos acercamientos a este modelo: data distribuida y control centralizado, y data y control distribuidos.

En el acercamiento de data distribuida y control centralizado se divide en dos soluciones: **particionado y replicado**. En el primero de estos lo que se tiene es un conjunto de nodos y cada uno de ellos opera a cierto nivel de seguridad, así el usuario con nivel de permisos X accede al servidor que maneja la data para X. El **replicado** surgió debido a que si alguien con altos derechos de seguridad deseaba consultar data con de bajo nivel de seguridad debía enviar su petición a un servidor de bajo nivel de seguridad por lo cual se podría divulgar información sensible. En el esquema replicado entonces la data se repite en cascada de tal forma que el nivel más alto tiene una copia entera de la base de datos, y el más bajo solamente la información de más bajo nivel. El otro acercamiento de data y control distribuido cada nodo contiene información de distintos niveles y está diseñado para aceptar peticiones de cualquier nivel de usuario.

El problema de inferencia

El problema de inferencia consiste en usuarios tratando de ejecutar consultas sobre la BD y estos infiriendo información sobre la respuesta legítima que la base de datos debe responder. Las herramientas para minería de datos hacen este problema aún más peligroso ya que hacen que sea más fácil para cualquier novato poder deducir patrones e información importantes de simplemente probar consultas.

6.8.6 ~~Tipos de arquitecturas/~~implementaciones ~~Factores~~

En un sistema de bases de datos distribuidas, existen varios factores que deben tomar en consideración que definen la arquitectura del sistema:

1. **Distribución:** Los componentes del sistema están localizados en la misma computadora o no.
2. **Heterogeneidad:** Un sistema es heterogéneo cuando existen en él componentes que se ejecutan en diversos sistemas operativos, de diferentes fuentes, etc.
3. **Autonomía:** Se puede presentar en diferentes niveles, los cuales se describen a continuación:
 - *Autonomía de diseño:* Habilidad de un componente del sistema para decidir cuestiones relacionadas a su propio diseño.
 - *Autonomía de comunicación:* Habilidad de un componente del sistema para decidir como y cuando comunicarse con otros SGBD (Sistema Gestor de Bases de Datos).
 - *Autonomía de ejecución:* Habilidad de un componente del sistema para ejecutar operaciones locales como quiera.

Multi Base de Datos Distribuida

Cuando una base de datos distribuida es muy homogénea se dice que es multi base de datos distribuida.

Sin embargo, los dos niveles de correspondencia implican un gasto de recursos durante la ejecución de una consulta o de un programa, lo que reduce la eficiencia del SGBD. Por esta razón pocos SGBD han implementado la arquitectura completa.

6.9 Lenguajes de los SGBD

Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores de aplicaciones y usuarios finales. Los lenguajes van a permitir al administrador de la BD especificar los datos que componen la BD, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios. Los lenguajes del SGBD se clasifican en:

- **Lenguaje de definición de datos (LDD o DDL):** se utiliza para especificar el esquema de la BD, las vistas de los usuarios y las estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la BD.
- **Lenguaje de manipulación de datos (LMD o DML):** se utilizan para leer y actualizar los datos de la BD. Es el utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones. Los hay procedurales, en los que el usuario será normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan. Las sentencias de un LMD procedural están embebidas en un lenguaje de alto nivel llamado anfitrión. Las BD jerárquicas y en red utilizan estos LMD procedurales.

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

- La mayoría de los SGBD comerciales incluyen **lenguajes de cuarta generación (4GL)** que permiten al usuario desarrollar aplicaciones de forma fácil y rápida, también se les llama herramientas de desarrollo. Ejemplos de esto son las herramientas del SGBD ORACLE: SQL Forms para la generación de formularios de pantalla y para interactuar con los datos; SQL Reports para generar informes de los datos contenidos en la BD; PL/SQL lenguaje para crear procedimientos que interactúen con los datos de la BD.