SISTEMAS DE BASES DE DATOS

1. SISTEMAS DE ARCHIVOS – INCONVENIENTES.

Consideremos un sistema de información de un banco que mantiene información sobre todos los clientes y sus cajas de ahorro. Los registros sobre los clientes y las cajas de ahorro están en archivos. Además, el sistema cuenta con algunos programas de aplicación que permiten manipular estos archivos:

- un programa para acreditar o debitar de una cuenta.
- un programa para agregar una nueva cuenta.
- un programa para averiguar el saldo de una cuenta.
- un programa para generar los resúmenes de cuenta mensuales.

Estos programas de aplicación han sido escritos por programadores en respuesta a las necesidades de la empresa.
Se agregan nuevos programas de aplicación al sistema a medida que surgen nuevas necesidades. Por ejemplo, supongamos que el banco empieza a operar con cuentas corrientes. Como resultado de esto, se crearán nuevos archivos para contener la información sobre las cuentas corrientes, y se escribirán nuevos programas de aplicación.
A medida que pasa el tiempo, más archivos y más programas son agregados al sistema. Dado que estos archivos y programas son creados a lo largo de un período extenso, presumiblemente por diferentes programadores, es muy probable que los archivos tengan formatos diferentes y que los programas estén escritos en varios lenguajes de programación.

El ambiente descripto anteriormente es un típico sistema de procesamiento de archivos, soportado por un sistema operativo convencional. Los registros permanentes son almacenados en varios archivos, y diferentes programas de aplicación son escritos para extraer registros de y agregar registros a los archivos apropiados.
Este esquema tiene ciertas desventajas:

REDUNDANCIA E INCONSISTENCIA DE DATOS

Dado que los archivos y programas de aplicación son creados por diferentes programadores a lo largo de un período extenso, la misma información puede estar replicada en varios lugares (archivos).
Por ejemplo, el domicilio y el número de teléfono de un cliente en particular puede aparecer en un archivo que contiene todos los registros de las cuentas corrientes, y en un archivo que contiene todos los registros de las cajas de ahorro. Esta redundancia conduce a un costo mayor de almacenamiento y de acceso, así como a la posible inconsistencia de los datos. Por inconsistencia de los datos, se entiende que las diferentes copias de un mismo dato no coinciden. Por ejemplo, si un cliente cambia de domicilio, y este cambio solamente se refleja en los registros de cuentas corrientes, los datos están inconsistentes.

DIFICULTAD PARA ACCEDER A LOS DATOS

Supóngase que uno de los oficiales del banco necesita encontrar los nombres de todos los clientes que viven en un área de la ciudad cuyo código postal es 1405. El oficial llama al departamento de procesamiento de información y les pide que le generen esa lista. Como este es un pedido inusual que no había sido anticipado cuando se diseñó el sistema original, no existe un programa de aplicación que genere esa lista. Existe, sin embargo, un programa de aplicación que lista todos los clientes. El oficial del banco tiene ahora dos alternativas. Puede obtener la lista de todos los clientes y pedirle a una de sus secretarias que extraiga manualmente la información requerida, o puede pedirle al departamento de procesamiento de información que uno de sus programadores escriba un programa de aplicación que lo haga. Ambas alternativas son obviamente insatisfactorias. Supóngase que se escribe un programa así y que, varios días después, ese mismo oficial necesita una lista que incluya solamente aquellos clientes que tienen un saldo de $ 1.000,00 o más. Tal como es de esperar, no existe un programa para generar esa lista. Otra vez, el oficial tiene las dos alternativas vistas, ninguna de las cuales es satisfactoria.
Lo que queremos resaltar es que este ambiente no permite obtener información en una forma conveniente y eficiente.

AISLAMIENTO DE LOS DATOS

Dado que los datos están “desparramados” en varios archivos, y que los archivos pueden tener diferentes formatos, se hace difícil escribir nuevos programas de aplicación para obtener los datos necesarios.

MULTIPLES USUARIOS

Para mejorar la performance global del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que múltiples usuarios actualicen los datos simultáneamente.

En un ambiente así, la interacción de las actualizaciones concurrentes puede resultar en datos inconsistentes. Por ejemplo, consideremos una cuenta bancaria A con un saldo de $ 5.000,00.

Si dos clientes extraen fondos (digamos $ 500,00 y $ 1.000,00 respectivamente) de la cuenta A al mismo tiempo (suponiendo que ambos están habilitados para extraer dinero de esa cuenta por ser una cuenta conjunta a nombre de los dos), el resultado de las ejecuciones concurrentes puede dejar como saldo $ 4.500,00, o $ 4.000,00, en vez de $ 3.500,00.

El sistema debería supervisar que esto no suceda. Como los datos pueden ser accedidos por diferentes programas de aplicación que no han sido coordinados previamente, esta supervisión se hace muy dificultosa.

PROBLEMAS DE CONFIDENCIALIDAD

No todo usuario de un sistema debería poder acceder a todos los datos. Por ejemplo, en un sistema bancario, una persona que prepara los cheques de sueldo solo necesita acceder a información de los empleados del banco. El o ella no debe poder acceder a información de los clientes del banco.

Análogamente, los cajeros solo necesitan acceder a información de las cuentas. No necesitan acceder a información de los empleados.

Cuando los programas de aplicación son agregados al sistema en una forma ad hoc, se hace difícil imponer estas restricciones de acceso.

PROBLEMAS DE INTEGRIDAD

Los valores de datos almacenados deben satisfacer ciertas restricciones de consistencia. Por ejemplo, el saldo de una cuenta bancaria nunca puede estar por debajo de una cantidad pre-especificada (por ejemplo, $ 100,00). Estas restricciones deberían ser impuestas por el sistema.

La imposición puede llevarse a cabo agregando código apropiado en los diferentes programas de aplicación. Sin embargo, cuando se agregan nuevas restricciones, se hace difícil cambiar los programas para imponerlas. Esto se complica más aún cuando las restricciones involucran varios ítems de datos de diferentes archivos.

Estas dificultades, entre otras, han llevado al desarrollo de sistemas de bases de datos.

2. QUE ES UNA BASE DE DATOS ?

“Conjunto de datos operativos almacenados, a los que acceden los sistemas de aplicación de una empresa”. R.W.Engles

“Colección de datos interrelacionados, archivados juntos, sin redundancias perjudiciales o innecesarias; su finalidad es la de servir a una o más aplicaciones de la mejor manera posible; los datos están almacenados de modo que resultan independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados”. James Martin

Debido a la importancia de la información en la mayoría de las organizaciones, la base de datos es un recurso valioso

Un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas que permiten administrar (gestionar) una base de datos.
También se utilizan las siguientes siglas:
DBMS (Data Base Management System)
SCBD (Sistema de Control de Bases de Datos)
SABD (Sistema Administrador de Bases de Datos)
SMBD (Sistema Manejador de Bases de Datos)

La administración de los datos involucra:

- definición de las estructuras de almacenamiento de la información.
- la provisión de mecanismos para la manipulación de la información.
- la protección de la información almacenada en la base de datos contra accesos no autorizados.
- la posibilidad de recuperar la información almacenada en la base de datos después de una falla del sistema (hardware o software).
- la protección de la información almacenada en la base de datos para evitar posibles resultados anómalos cuando los datos son compartidos entre varios usuarios concurrentemente.

El objetivo principal de un SGBD es el de proveer un ambiente que sea tanto conveniente como eficiente para obtener información de y almacenar información en la base de datos.

El sistema de base de datos provee a la empresa con la posibilidad de control centralizado de sus datos operativos.
De esta noción de control centralizado surgen las siguientes posibilidades:

- minimizar la redundancia.
- minimizar los datos inconsistentes.
- compartir los datos.
- cumplimiento de estándares (formato de campos, nombres de campos, documentación) lo que facilita el compartir los datos.
- cumplimiento de las restricciones de seguridad.
- balance de los requerimientos conflictivos de las distintas aplicaciones, pudiendo decidirse por estructurar el sistema para lograr la mejor solución para la empresa y no para una aplicación en especial.

3. ESQUEMAS E INSTANCIAS.

El conjunto de información almacenada en la base de datos en un instante en particular se denomina instancia de la base de datos. La descripción de una base de datos se denomina el esquema de la base de datos. Las bases de datos cambian a lo largo del tiempo a medida que se agrega información a la base de datos y se borra información de la misma. Los esquemas de una base de datos cambian con menor frecuencia que las instancias.
Existen varios esquemas en una base de datos. En general, los SGBD soportan un esquema físico, un esquema conceptual y varios esquemas externos para cada base de datos.

4. ARQUITECTURA A TRES NIVELES.

El objetivo de un sistema de base de datos es el de simplificar y facilitar el acceso a los datos. Los usuarios del sistema no deberían preocuparse innecesariamente con los detalles físicos de la implementación del sistema. El sistema debe ocultar ciertos detalles de cómo los datos son almacenados y mantenidos.
Esto se logra definiendo varios niveles de abstracción en los que puede visualizarse una base de datos.

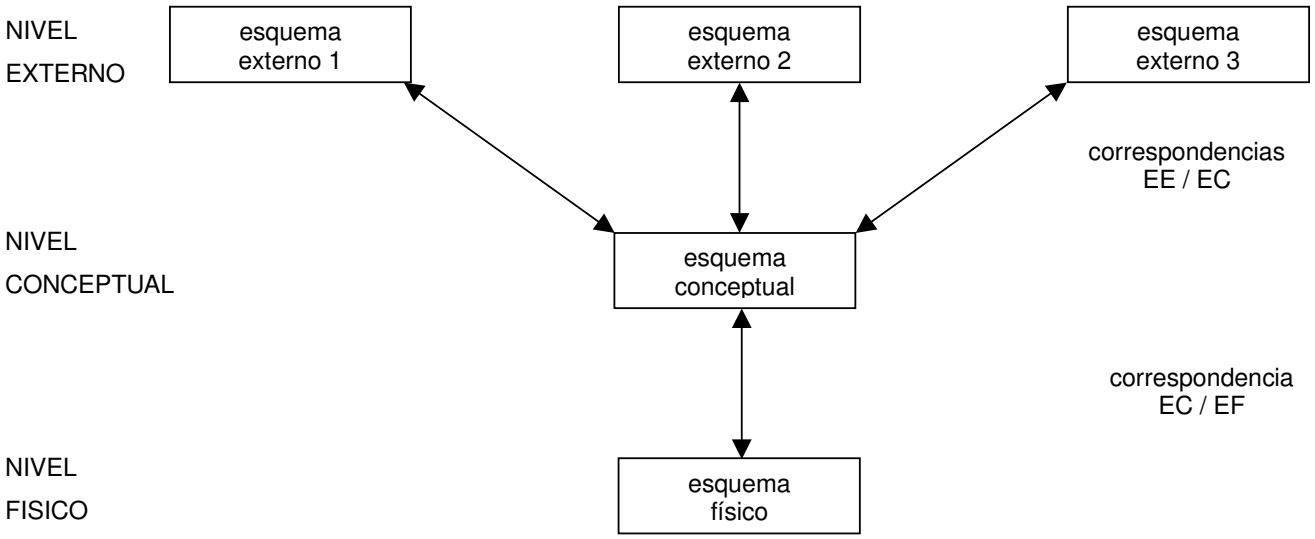
NIVEL FISICO O INTERNO
Este es el menor nivel de abstracción. En este nivel se describe la forma de almacenamiento de los datos.

NIVEL CONCEPTUAL O GLOBAL
Este es el siguiente nivel de abstracción. En este nivel se describe qué datos están almacenados en la base de datos, y las vinculaciones entre esos datos.
Este nivel describe la base de datos en su totalidad. Aunque la implementación de las estructuras del nivel conceptual implique estructuras complejas en el nivel físico, el usuario del nivel conceptual no necesita estar al tanto de ello.

NIVEL EXTERNO O DE APLICACIÓN O DE USUARIO
Este es el mayor nivel de abstracción. En este nivel se describen subconjuntos de la base de datos total. A pesar de que en el nivel conceptual se utilizan estructuras más simples, aún permanece cierto grado de complejidad debida al gran tamaño de la base de datos. Muchos usuarios del sistema de base de datos no están interesados en toda la información.
En el nivel externo se definen visiones para simplificar la interacción de los usuarios con el sistema. Pueden existir muchas visiones diferentes para la misma base de datos.

Una base de datos puede ser descripta en cada uno de estos tres niveles de abstracción, dando lugar a diferentes descripciones o esquemas de la base de datos. La descripción de la base de datos en el nivel físico constituye el esquema físico de la base de datos. El esquema conceptual es la descripción de la base de datos en el nivel conceptual. Los esquemas externos, subesquemas o visiones describen la base de datos en el nivel externo.

Normalmente varios esquemas externos se hacen corresponder a un esquema conceptual, que a su vez se hace corresponder a un esquema físico.
Esta correspondencia a dos niveles (EE / EC y EC / EF) es menos eficiente en términos de rendimiento pero provee mayor independencia de datos, pues los esquemas físico y externos pueden ser alterados independientemente uno de otros.



En un SGBD con tres niveles de esquemas, existirán dos niveles de transformación:

- transformación conceptual / físico que permite el paso de las ocurrencias de datos del formato conceptual al formato físico y viceversa.
- transformación externo / conceptual que permite el paso de las ocurrencias de datos del formato externo al formato conceptual y viceversa.

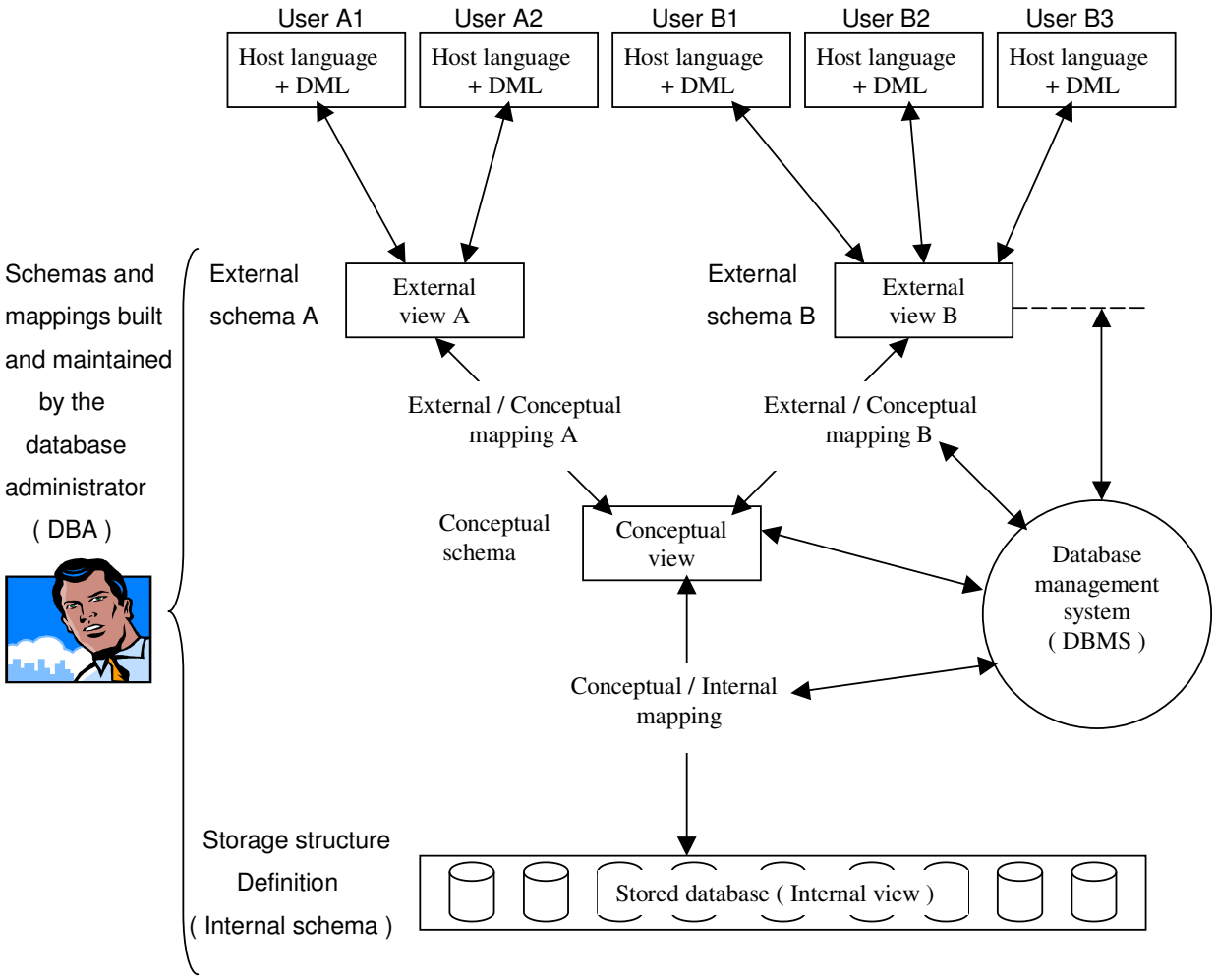
La correspondencia EC / EF define la correspondencia entre el esquema conceptual y el esquema físico; especifica la manera en que los registros y campos conceptuales están representados en la base de datos almacenada.

Una correspondencia EE / EC define la correspondencia entre un esquema externo en particular y el esquema conceptual ya que pueden existir diferencias entre estos dos niveles. Por ejemplo, los campos pueden tener tipos de datos diferentes, los nombres de registros y de campos pueden ser distintos, varios campos conceptuales pueden ser combinados en un único campo (virtual) externo, etc. Pueden existir muchos esquemas externos a la vez; varios usuarios pueden compartir un esquema externo; diferentes esquemas externos pueden solaparse.

Cada esquema externo cumple las siguientes funciones:

- selecciona un subconjunto lógico de los datos del esquema conceptual necesarios para una aplicación o conjunto de aplicaciones.
- presenta los datos de la forma más conveniente para el lenguaje de programación.
- restringe el acceso a un subconjunto de datos, contribuyendo así a la protección de la privacidad de los datos.

Algunos sistemas permiten la definición de un esquema externo en términos de otros esquemas externos (vía una correspondencia EE / EE), en vez de requerir una definición explícita de la correspondencia con el nivel conceptual.


User A1
User A2
User B1
User B2
User B3
Host language + DML
Host language + DML
Host language + DML
Host language + DML
Host language + DML
Schemas and mappings built and maintained by the database administrator (DBA)
External schema A
External view A
External schema B
External view B
External / Conceptual mapping A
External / Conceptual mapping B
Conceptual schema
Conceptual view
Database management system (DBMS)
Conceptual / Internal mapping
Storage structure Definition (Internal schema)
Stored database (Internal view)

5. INDEPENDENCIA DE DATOS.

La independencia de los datos en el contexto de una base de datos implica la independencia de las descripciones de los datos (esquemas) entre sí. Esto permite que se realicen cambios en un nivel sin afectar los demás niveles.

ESQUEMA CONCEPTUAL

En principio, el esquema conceptual es independiente de los otros esquemas. Es especificado (al menos en teoría) antes que los demás y es compilado independientemente.

ESQUEMA FISICO

Este esquema no puede ser independiente del esquema conceptual, ya que su propósito es organizar el almacenamiento de los elementos descriptos en el esquema conceptual. Es compilado contra el esquema conceptual y normalmente requiere ser rediseñado después de algún cambio en el esquema conceptual.

Sin embargo, el esquema físico es independiente de los esquemas externos y puede ser independiente de las características de los dispositivos físicos si la alocación de almacenamiento se realiza en función de páginas en vez de pistas y cilindros, y si se usan punteros lógicos en vez de direcciones físicas para las referencias internas de los registros.

El contenido del esquema conceptual es más estable que el del esquema físico, por lo tanto es conveniente que la correspondencia entre los objetos de datos conceptuales y los objetos de datos almacenados sea declarada en el esquema físico. Así se evita cambiar el esquema conceptual ante cada reorganización del almacenamiento.

Si la estructura de almacenamiento de la base de datos almacenada es modificada, entonces la correspondencia EC / EF debe también ser modificada, para que el esquema conceptual permanezca invariante.

ESQUEMA EXTERNO

Aunque un esquema externo es una descripción lógica de un subconjunto del esquema conceptual, no tiene porque ser un subconjunto de las descripciones en el esquema conceptual; es posible que existan variaciones entre estas dos visiones:

- Formatos: debería ser posible declarar un ítem numérico del esquema conceptual como alfanumérico en el esquema externo.
- Tipos de registros: debería ser posible definir un tipo de registro externo como un subconjunto de los componentes del tipo de registro conceptual.
- Asociaciones: debería ser posible declarar sólo las asociaciones necesarias en el esquema externo.

Dado que el programa de aplicación procesa datos descriptos en un esquema externo, es natural que dependa de este esquema.

La posibilidad de modificar la definición de un esquema en un nivel sin afectar la definición de los esquemas en el nivel superior se conoce como independencia de datos.
Existen dos niveles de independencia de datos:

INDEPENDENCIA FISICA DE LOS DATOS

Es la habilidad para modificar el esquema físico sin causar la modificación del esquema conceptual (y por lo tanto, tampoco se ven afectados los esquemas externos ni los programas de aplicación).
Suele ser necesario modificar el esquema físico para mejorar el rendimiento de la base de datos.
El SGBD debería separar la organización física de los datos tan completamente como fuese posible de la descripción lógica de los mismos. El objetivo es que la organización física pueda ser cambiada parcial o totalmente sin que eso implique la necesidad de rehacer ni la descripción lógica global ni los programas de aplicación.
Nótese que si se modifica el esquema conceptual, es casi seguro que deberá modificarse el esquema físico, ya que éste es la implementación de aquel.

INDEPENDENCIA LOGICA DE LOS DATOS

Es la habilidad para modificar el esquema conceptual sin causar la modificación innecesaria de los esquemas externos (lo cual implica que tampoco sea necesario modificar los programas de aplicación). Es necesario modificar el esquema conceptual cada vez que se modifica la estructura lógica de la base de datos (por ejemplo, agregar un tipo de registro, agregar un campo a un registro), pero esto no debería afectar innecesariamente a los programas de aplicación ya existentes.

6. LENGUAJE DE DEFINICION DE DATOS (LDD).

Un esquema de una base de datos se especifica mediante un conjunto de definiciones expresadas en un lenguaje especial llamado lenguaje de definición de datos (LDD). El resultado de la compilación de las sentencias del LDD es un conjunto de información almacenada en un archivo especial denominado diccionario de datos. Un diccionario de datos es un archivo que contiene metadatos (datos sobre datos). Este archivo es consultado por el SGBD antes de leer o modificar los datos.
El esquema conceptual y los esquemas externos se definen con el LDD. El esquema físico (estructura de almacenamiento y los métodos de acceso usados para implementar la base de datos) también se especifica con el LDD (este sublenguaje del LDD suele denominarse “lenguaje de definición de almacenamiento”).

7. LENGUAJE DE MANIPULACION DE DATOS (LMD).

La manipulación de datos comprende:

- consultas
- actualizaciones (altas , bajas y cambios)

Un lenguaje de manipulación de datos (LMD) es un lenguaje que permite a los usuarios acceder y manipular los datos tal como están organizados por el modelo de datos.
Básicamente existen dos tipos de LMD:

- procedimentales: que requieren que el usuario especifique qué datos necesita y cómo obtenerlos.
- declarativos: que requieren que el usuario especifique qué datos necesita sin especificar cómo obtenerlos.

Los LMDs declarativos son usualmente más fáciles de aprender y usar que los LMDs procedimentales. Sin embargo, ya que el usuario no especifica la manera de obtener los datos, estos lenguajes suelen generar código menos eficiente que el producido por lenguajes procedimentales.
Esta dificultad puede remediarse a través del uso de varias técnicas de optimización ejecutadas por el SGBD.

8. SISTEMA DE GESTION DE BASE DE DATOS (SGBD).

El diseño de un SGBD puede, en teoría, basarse en cualquiera de las siguientes estrategias:

a) como parte integrada de la aplicación.
b) como parte integrada del sistema operativo.
c) como un sistema independiente que actúe como interfaz con el sistema operativo.

Como parte integrada de la aplicación, un SGBD sería un sistema más o menos “hecho a medida” generado como una extensión del lenguaje anfitrión proveyendo una tranferencia más rápida de datos de lo que sería posible de otro modo. Sin embargo, la desventaja mayor sería el costo de desarrollar tantos “sistemas hechos a medida”, cada aplicación teniendo su SGBD exclusivo.
Además, la protección de los datos sería problemática ya que sería dificil asegurar la ausencia de incompatibilidad entre esos sistemas. Finalmente, el uso concurrente también sería dificil de lograr porque un sistema no sabría lo que el otro sistema está haciendo, y el espacio en memoria requerido para acomodar tantos SGBD exclusivos sería demasiado grande.
Esta estrategia, por lo tanto, no es viable.

El segundo enfoque puede representar la solución ideal desde el punto de vista del usuario si retiene el control centralizado y a la vez provee acceso eficiente a la base de datos. Sin embargo, el

problema es el diseño de un sistema operativo que, además de sus funciones normales, pueda manipular información estructurada tal como la requerida en una base de datos.
La factibilidad de este enfoque no ha sido demostrada todavía.

La tercera posibilidad es un compromiso entre los dos primeros enfoques, donde el SGBD actua como una interfaz entre el programa de aplicación y el sistema operativo. Pero esto es un tanto ineficiente porque los programas no pueden comunicarse directamente con el sistema operativo.
Por ahora, sin embargo, es el único enfoque que ofrece una solución viable.

Un SGBD es un conjunto de módulos de programas que provee la interfaz entre los datos almacenados en la base de datos y los programas de aplicación y consultas submitidas al sistema.
El SGBD es el responsable de las siguientes tareas:

- Interacción con el manejador de archivos: Los datos son almacenados en disco usando el manejador de archivos usualmente provisto por un sistema operativo convencional. El SGBD traduce las sentencias LMD a comandos de menor nivel del sistema de archivo. Por lo tanto, el SGBD es el responsable del almacenamiento, la obtención y la actualización de los datos en la base de datos.
- Control de la integridad: Los valores de datos almacenados en la base de datos deben satisfacer ciertos tipos de restricciones de integridad. Estas restricciones deben ser explicitadas por el ABD. Si se especifican estas restricciones, el SGBD debe ser capaz de verificar si las actualizaciones a la base de datos violan alguna de estas restricciones, y de ser así tomar la acción apropiada.
- Control de la seguridad: No todo usuario de la base de datos necesita acceder a la totalidad del contenido de la misma. Es tarea del SGBD controlar la autorización de acceso de cada usuario.
- Respaldo y restauración: Un sistema computarizado, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallas. Existe una variedad de causas posibles para esas fallas: fallas en los discos, cortes de energía eléctrica, y errores de programas. En cada uno de estos casos, es muy probable que se pierda información de la base de datos. Es responsaliblidad del SGBD detectar estas fallas y restaurar la base de datos a un estado existente previo a la falla. Esto se logra generalmente mediante procedimientos de respaldo (backup), bitácora (logging) y recuperación.
- Control de concurrencia: Cuando varios usuarios actualizan la base de datos concurrentemente, la consistencia de los datos puede verse afectada. Es responsabilidad del SGBD controlar la interacción de los usuarios concurrentes y evitar que la base de datos tenga datos inconsistentes.

Algunos SGBD, diseñados para ser usados en computadores personales pequeños, no cumplen varias de estas funciones. Esto hace que el SGBD sea más pequeño: con menos requerimientos de recursos físicos, especialmente memoria principal, y menos costoso para implementar. Por ejemplo, algunos SGBD pequeños sólo permiten el acceso a un usuario por vez. Otros dejan las tareas de respaldo y restauración al usuario.
Aunque este enfoque es suficiente para bases de datos personales, no es adecuado para satisfacer las necesidades de una mediana o gran empresa.

Se requiere una gran cantidad de módulos para la creación y operación de una base de datos. Cada uno de estos módulos se encarga de una de las responsabilidades del sistema total. El sistema operativo de la computadora realiza algunas de las funciones del sistema de base de datos, pero en la mayoría de los casos el sistema operativo sólo provee los servicios más elementales y el SGBD debe apoyarse sobre esos cimientos. Los detalles de los módulos dependen de las distintas implementaciones.
Nosotros presentaremos una descripción generalizada de los módulos funcionales:

COMPILADOR DEL LDD
Convierte las sentencias LDD en información que se almacena en el diccionario de datos.

GESTOR DE LA BASE DE DATOS
Provee la interfaz entre el manejador de archivos y los programas de aplicación y consultas submitidos al sistema. Básicamente realiza dos tipos de traducciones: traduce las operaciones sobre registros externos a operaciones sobre registros físicos; y traduce los registros físicos a registros externos.

PROCESADOR DE CONSULTAS
Traduce las sentencias del lenguaje de consulta en instrucciones entendibles por el gestor de la base de datos. Adicionalmente el procesador de consultas procura transformar la solicitud del usuario en una forma más eficiente, encontrando así una estrategia adecuada para ejecutar la consulta.

PRECOMPILADOR DEL LMD
Convierte las sentencias del LMD incluidas en un programa de aplicación en llamadas a procedimientos (CALLs) en el lenguaje anfitrión. El precompilador debe interactuar con el procesador de consultas para generar el código apropiado.

MANEJADOR DE INTEGRIDAD
Verifica que las actualizaciones a la base de datos satisfagan las restricciones de integridad.

MANEJADOR DE AUTORIZACIONES
Comprueba que el usuario esté autorizado para acceder a la información.

MANEJADOR DE RECUPERACIONES

Asegura que la base de datos permanezca en un estado correcto a pesar de que ocurran fallas en el sistema.

CONTROLADOR DE CONCURRENCIA

Garantiza que las interacciones concurrentes contra la base de datos se lleven a cabo sin conflictos entre ellas.

MANEJADOR DE ARCHIVOS

Se encarga de asignar el espacio físico dentro del disco y de operar sobre las estructuras de datos que se emplean para representar la información en los archivos en disco. En algunos sistemas, el manejador de archivos es un componente del sistema operativo subyacente, en otros forma parte del SGBD. Es usual que el manejador de archivos de propósito general provisto por el sistema operativo no está totalmente adecuado a los requerimientos especiales de un SGBD.

MANEJADOR DE BUFFERS

Se encarga de administrar los buffers del sistema de base de datos. Los buffers son áreas específicas de la memoria principal. En un sistema de base de datos existen al menos dos tipos de buffers:

- un buffer de E / S para transferir información entre el sistema operativo y el SGBD.
- un área reservada para contener las partes más usadas de la base de datos con el objeto de minimizar la cantidad de operaciones físicas de E / S.

Además, son necesarias varias estructuras de datos adicionales:

ARCHIVOS DE DATOS

Almacenan la base de datos en sí.

DICCIONARIO DE DATOS

Almacena información sobre la estructura de la base de datos. Aquí se conserva también la información de autorización de acceso. Guarda también información estadística acerca de los datos almacenados en la base de datos (tamaño de archivos, distribución de valores por campo, etc).

INDICES

Proveen acceso eficiente a los ítems de datos que contienen un determinado valor.

Para operar una base de datos se requiere un gran número de utilitarios. Algunos de ellos son:

RUTINA DE CARGA (LOAD)

Para crear la versión original de la base de datos desde uno o más archivos secuenciales.

RUTINAS DE VUELCO Y RECARGA (DUMP Y RESTORE)

Para volcar la base de datos (total o parcialmente) a almacenamiento de respaldo y re-cargar la base de datos (total o parcialmente) desde esas copias de respaldo.

RUTINAS DE REORGANIZACION

Para reorganizar los datos de la base de datos por distintos motivos, por ejemplo: para agruparlos de una manera particular.

RUTINA DE RECOLECCION DE BASURA Y RE-ALOCACION

Para eliminar físicamente registros borrados (lógicamente) de los dispositivos de almacenamiento, para consolidar el espacio liberado y re-alocarlo donde fuese necesario. Algunos SGBD pueden realizar esta tarea automáticamente.

RUTINA DE REORGANIZACION DE INDICES

Para reorganizar índices, esta es una parte de la reorganización de la base de datos.

RUTINAS DE MONITOREO Y ANALISIS

Para recolectar estadísticas de las operaciones sobre la base de datos y analizarlas para la reorganización de la base de datos.

9. USUARIOS DE UN SISTEMA DE BASE DE DATOS.

Un sistema de base de datos está compuesto por distintos tipos de elementos: software, hardware, datos, personas.

DATOS

- base de datos propiamente dicha.
- diccionario de datos.
- bitácora.
- copias de respaldo.

HARDWARE

SOFTWARE

- Sistema de Gestión de Base de Datos.
- programas de aplicacion.

USUARIOS

- programadores de aplicaciones: son profesionales de la computación, que interactúan con el sistema a través de sentencias del LMD incluídas en un programa escrito en un lenguaje anfitrión o "host" (por ejemplo, Cobol, Pascal, Fortran, PL/1, C). Dado que la sintaxis del LMD usualmente es diferente a la del lenguaje anfitrión, las sentencias LMD son prefijadas por un caracter especial para que el preprocesador pueda reconocerlas y generar el código apropiado. Este preprocesador se denomina precompilador LMD. El precompilador LMD convierte las sentencias LMD a invocaciones de procedimientos (CALLs) en el lenguaje anfitrión. El programa fuente resultante es compilado por el compilador del lenguaje anfitrión, que genera el código objeto apropiado.
- usuarios finales: son usuarios sofisticados que interactúan con el sistema sin escribir programas. Realizan sus consultas utilizando un lenguaje de consulta de base de datos (query language), con facilidades de representación gráfica, análisis estadístico, generación de reportes, etc. Suelen ser consultas no previstas, ad hoc.
- usuarios de aplicaciones: son usuarios no sofisticados que interactúan con el sistema invocando algunos de los programas de aplicación que han sido escritos previamente. Por ejemplo, un cajero que necesita registrar una transferencia de A500 de la cuenta C1 a la C2 llamará a un programa llamado transf. El programa le solicitará el ingreso de la cantidad de dinero a transferir, el número de cuenta de la cual se transfiere el dinero y el número de cuenta a la cual se transfiere el dinero.
- administrador de base de datos: es la persona (o grupo de personas) encargadas del diseño, implementación y mantenimiento de las bases de datos.

10. ADMINISTRACION DE BASES DE DATOS.

Una de las principales razones para tener un SGBD es el control centralizado de los datos y de los programas que acceden a esos datos. La persona (o grupo de personas) que ejerce el control centralizado del sistema se denomina ADMINISTRADOR DE BASE DE DATOS.

Las funciones del grupo de administración de base de datos (ABD) incluyen:

- Definición de esquemas, es decir la codificación y compilación del esquema conceptual y los esquemas externos de la base de datos. Este proceso se conoce generalmente como DISEÑO LOGICO de la base de datos.
- Definición de las estructuras de almacenamiento y métodos de acceso, es decir la creación del esquema físico de la base de datos. Este proceso se conoce generalmente como DISEÑO FISICO de la base de datos.
- Modificación de la organización lógica y física, es decir la modificación de los esquemas externos, conceptual y físico.
- Autorización para el acceso a los datos, el ABD otorga a los usuarios diferentes tipos de autorización para acceder a la base de datos (o partes de la base de datos).
- Especificación de las restricciones de integridad, el ABD especifica, con las facilidades que para ello provee el LDD, las restricciones de integridad que el SGBD consulta y verifica cada vez que ocurre una actualización en la base de datos.
- Definición de los procedimientos de respaldo y restauración, el ABD debe definir e implementar una estrategia de respaldo y restauración, que incluya, por ejemplo, vuelcos periódicos de la base de datos a cinta y procedimientos de carga de la base de datos desde la copia de respaldo más reciente.

Para llevar a cabo estas funciones el ABD utiliza los programas utilitarios provistos con el SGBD. Otra de las herramientas fundamentales del ABD es el Diccionario de Datos.

11. DICCIONARIO DE DATOS.

El diccionario de datos (DD) es el elemento que relaciona, integra y controla todos los componentes de un sistema de bases de datos. Se puede considerar al DD como una base de datos en sí mismo, solo que en vez de contener datos de la empresa, contiene datos sobre los datos de la empresa.

La clave para el uso efectivo de una base de datos es la documentación adecuada de los datos que administra. El concepto de “diccionario de datos” surgió como un medio para proveer esa documentación. Actualmente ha sido extendido para significar un “recipiente” de toda la información necesaria a los usuarios y al SGBD. En este recipiente se almacenan las definiciones de los datos, su significado, las restricciones de integridad y los niveles de autorización, los esquemas externos, conceptual y físico (en código fuente y objeto), el código fuente y objeto de los programas, los utilitarios, los procedimientos de base de datos, las estadísticas, etc.

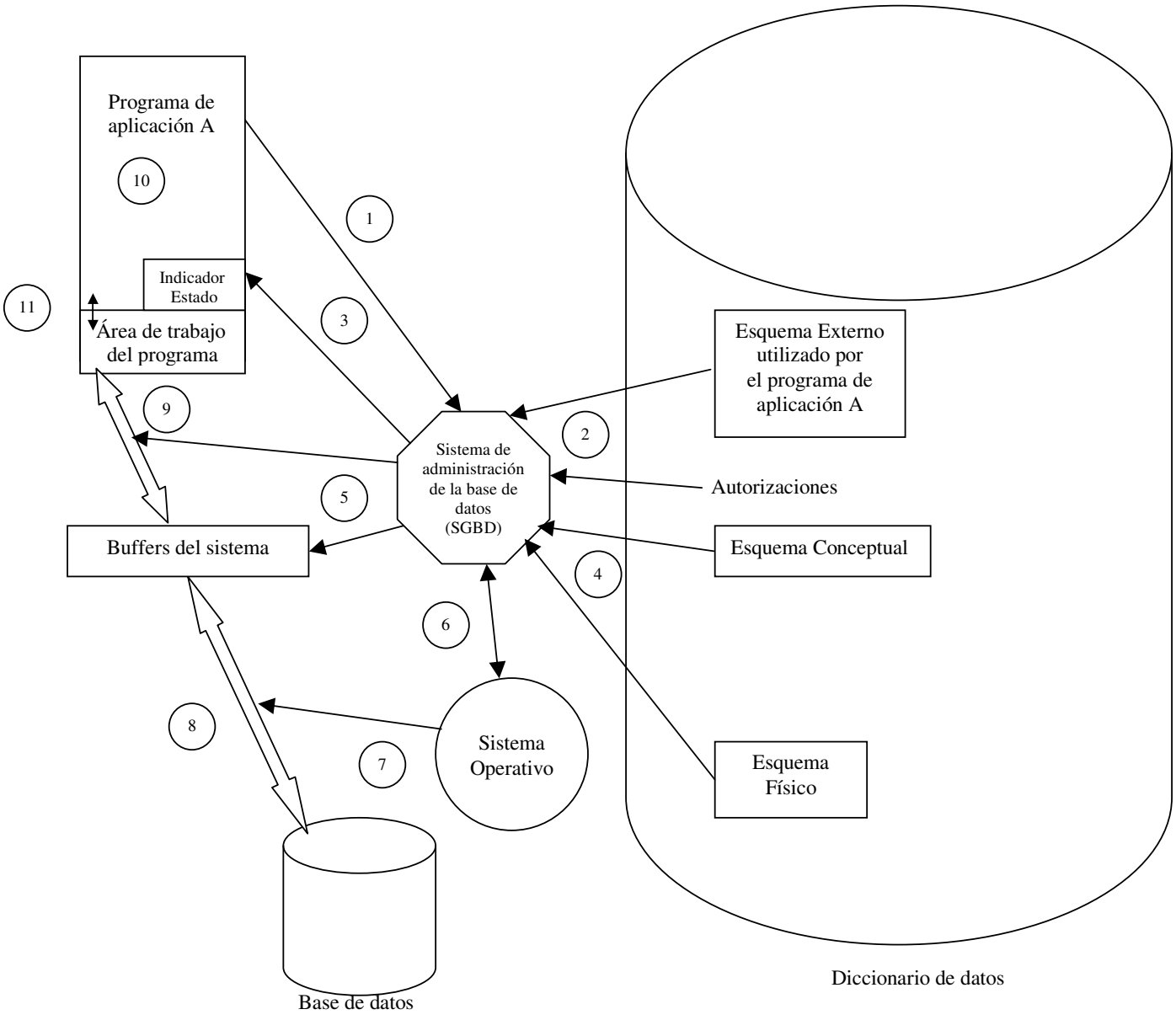
Un buen diccionario de datos también debería incluír información de referencias cruzadas que muestren, por ejemplo: qué programas usan qué datos de la base de datos, qué usuarios requieren qué informes, qué terminales están conectadas al sistema, etc.

El DD debe poder ser consultado como cualquier otra base de datos.
Así, por ejemplo, el ABD podrá saber qué programas y / o usuarios se verán afectados por algún cambio propuesto al sistema (análisis de impacto).

12. SECUENCIA DE EVENTOS PARA ACCEDER A UN REGISTRO.

El SGBD es el software que maneja todo acceso a la base de datos. Cuando un programa de aplicación requiere información de la base de datos, tienen lugar varias operaciones. Por ejemplo, consideremos los pasos involucrados en la consulta de un registro externo en particular.
Es probable que se requieran algunos campos de varios registros conceptuales. Cada ocurrencia de un registro conceptual puede, a su vez, requerir campos de varios registros físicos. Conceptualmente, entonces, el SGBD debe primero recuperar todas las ocurrencias de registros físicos requeridos, luego construir las ocurrencias de registros conceptuales requeridos y finalmente construir la ocurrencia del registro externo solicitado. En cada paso, puede ser necesario realizar conversiones de tipos de datos.

Veamos ahora en más detalle las operaciones que ocurren cuando un programa de aplicación solicita un registro al SGBD (los números entre paréntesis se corresponden con los números en la figura siguiente) :



(a) El programa de aplicación usando una sentencia del LMD solicita al SGBD uno o más registros del esquema externo especificando los arguemtos de selección requeridos (1).

(b) El SGBD analiza la solicitud contra la información presente en el esquema externo objeto (2). El SGBD también verifica las autorizaciones de acceso del usuario. Si la solicitud es inválida, devuelve el control al programa después de actualizar el indicador de error (3). Si la solicitud es válida, el SGBD rastrea las correspondencias EE / EC y EC / EF (4) para encontrar la información del almacenamiento y seleccionar un camino de acceso.

(c) Puede suceder que el registro requerido ya está en un buffer en memoria principal como consecuencia de una lectura previa. En ese caso, obviamente no es necesario volver a leer físicamente el registro. Por ello, el SGBD primero busca el registro requerido en el sistema de buffers (5). Si no está allí, el SGBD solicita al sistema operativo que se lo entregue (6).

(d) El sistema operativo localiza el registro deseado en la base de datos física (7), y se lo entrega en el buffer del sistema de base de datos (8) si la búsqueda es exitosa.

(e) Se repiten los pasos (c) y (d) hasta obtener todos los registros físicos necesarios.

(f) Con los registros hallados, el SGBD construye el registro externo y lo transfiere al área de trabajo del programa (9).

(g) El SGBD actualiza el indicador de estado (3) tanto si la búsqueda fue exitosa como si no, y devuelve el control al programa de aplicación.

(h) El programa de aplicación consulta el indicador de estado (10) y de acuerdo con el valor indicado manipula el registro en el área de trabajo con instrucciones del lenguaje anfitrión (11).

Para actualizar un registro de la base de datos, la secuencia de pasos es similar. Además, se realizan verificaciones adicionales de consistencia.

13. CICLO DE VIDA DE UNA BASE DE DATOS.

El ciclo de vida de una base de datos comprende tres fases: modelización , implementación y mantenimiento.

FASE 1 – MODELIZACION.

Existen dos posibles puntos de inicio en el ciclo de vida de una base de datos. Una base de datos puede ser creada o una base de datos ya existente puede ser reestructurada. En ambos casos el diseñador de la base de datos debe desarrollar modelos de cómo los datos son (o van a ser) procesados, especificando los tipos de entidades, las asociaciones entre ellos, los grupos funcionales de aplicaciones (o usuarios) y las restricciones de acceso, seguridad y privacidad.

Estos modelos serán discutidos con la comunidad de usuarios para asegurarse que satisfacerán las necesidades de las aplicaciones.

La modelización de una base de datos es un preceso iterativo.

FASE 2 – IMPLEMENTACION.

La implementación de una base de datos implica el diseño y la compilación de los esquemas, la carga de la base de datos y la compilación de los programas de aplicación que acceden a la base de datos.

2.1 Definición de esquemas.

Basándose en los modelos desarrollados, se codifican y compilan los esquemas de la base de datos. Primero, se codifica y compila el esquema global o conceptual de la base de datos usando un LDD de esquemas conceptuales. Segundo, se codifica y compila el módulo de correspondencia global / físico usando un LDD de esquema físico. El modulo objeto resultante define la estructura física de los archivos utilizados (factor de bloqueo, buffers, etc). Cada vez que el SGBD invoca al sistema operativo para abrir archivos, esta descripción física es usada.

Tercero, se codifican y compilan los esquemas externos usando un LDD de esquemas externos.

Cada módulo objeto resultante contiene las rutinas necesarias para derivar los tipos de objetos externos a partir de las definiciones de los tipos de objetos conceptuales y físicos.

Al finalizar esta tarea el diccionario de datos contiene información sobre los datos a ser almacenados en la base de datos. Todavía la base de datos no contiene datos.

2.2 Carga inicial.

Hay dos maneras de cargar una base de datos. La primera consiste en ingresar los datos interactivamente. Esto requiere programas que acepten datos ingresados desde una terminal y que los grabe en la base de datos.

El segundo método consiste en cargar la base de datos a partir de archivos ya existentes. Esto requiere programas que lean los datos de los archivos y los graben de acuerdo al esquema de la base de datos.

2.3 Programas de aplicación.

Los programas de aplicación que acceden a una base de datos tienen incluidas sentencias del LMD. Estos programas deben compilarse con un preprocesador que extrae las sentencias del LMD del programa fuente previo a su compilación, y las reemplaza con sentencias que invocan al SGBD. La salida del preprocesamiento es un módulo fuente que debe ser compilado para producir un programa objeto. Para trabajar correctamente el preprocesador debe obtener del programa información sobre el esquema externo a utilizar. El preprocesador usa las rutinas de traducción en el esquema externo objeto para generar correctamente las sentencias de acceso a la base de datos.

FASE 3 – MANTENIMIENTO.

Una vez cargada la base de datos entra en la fase de mantenimiento. Permanece en esta fase hasta que es reestructurada. De hecho, el mantenimiento, responsabilidad de la oficina de ABD, es la fase predominante en el ciclo de vida de una base de datos. Comprende dos clases de tareas: monitoreo de una base de datos en actividad y restauración de una base de datos fallada.

3.1 Monitoreo.

Es la responsabilidad del staff de ABD supervisar el rendimiento (performance) de un SGBD. Esto incluye el rol de intercomunicador entre la comunidad de usuarios y los operadores, y monitorear el rendimiento del SGBD en función de las distintas aplicaciones. El staff de ABD representa a los usuarios investigando los informes de fallas o anomalías del sistema durante la ejecución de los programas. Además el staff de ABD hace disponible información sobre la actual configuración de la base de datos, y los estándares de operación vigentes.

Para monitorear el rendimiento de la base de datos se utilizan programas utilitarios del SGBD que proveen estadísticas sobre la utilización del almacenamiento, tiempo de acceso y rendimiento (throughput) general del sistema.

Cuando la performance del acceso a la base de datos se degrada a un nivel inaceptable, el staff de ABD reorganiza la base de datos física e instala un nuevo módulo de correspondencia global / físico. Cuando surgen modificaciones a la estructura lógica de la base de datos, se reciclan las fases de modelización e implementación.

3.2 Recuperación ante fallas.

Cuando el SGBD falla (y existen numerosas posibles causas para que esto ocurra) la base de datos debe ser restaurada.

Una de las dificultades para restaurar una base de datos es determinar la causa de la falla. Es dificil generar un diagnóstico cuando se ejecutan varias transacciones concurrentemente sobre la base de datos. Esta es una de las principales desventajas de los SGBDs. Existen dos estrategias principales para la restauración de una base de datos: restauración hacia adelante (back-up y rollforward) y restauración hacia atrás (rollback).

Ambas estrategias requieren que el SGBD mantenga una bitácora (log) de todas las acciones que se ejecutaron sobre la base de datos.

RESTAURACION HACIA ADELANTE

En este método la base de datos es copiada (back-up) periódicamente. Cuando ocurre una falla, la base de datos es restaurada desde la copia de respaldo más reciente. A continuación, la porción apropiada de la bitácora puede ser aplicada (rollforward) a la base de datos restaurada. La desventaja de este enfoque es el tiempo que insume aplicar la bitácora a partir de la ultima copia de respaldo.

RESTAURACION HACIA ATRAS

Este segundo enfoque toma la base de datos fallada e intenta deshacer todas aquellas transacciones que no se completaron normalmente antes de la falla. Se deshace una transacción abortando la tarea que la generó y restaurando los registros afectados al estado que tenían antes de iniciarse la transacción. Una vez hecho esto, la base de datos debería estar en un estado válido. Las transacciones abortadas pueden volver a ejecutarse. Este enfoque es más rápido que el anterior porque se trabaja retrocediendo desde el momento de la falla.