UNIFY 2000

OSCAR HERNANDO GUZMAN C.
JAIME ALBERTO MARIN J.
RAUL ANDRES CASTILLO M.
Estudiantes

AGRADECIMIENTOS

En este trabajo hemos tenido la valiosa colaboración del ingeniero de sistemas Juan Carlos Castillo, hermano de Raúl Andrés Castillo, quien nos ayudó con la recolección de información y con la explicación de algunas partes de este trabajo. Agradecemos a él su tiempo dedicado a nosotros.

Queremos agradecer también a nuestros profesores de Sistemas Operacionales I, Pedro Enrique Gil Peñalosa y Gustavo Adolfo Osorio, por sus enseñanzas y consejos. Esto nos ha ayudado mucho en la elaboración de este trabajo y en la presentación y desarrollo de la exposición del tema a él referido.

INTRODUCCION

En la primera parte se presenta un esquema introductorio al tema para explicar qué es una base de datos y al definir un esbozo general de la arquitectura de un sistema de bases de datos. En la presentación de qué es un sistema de base de datos veremos cada uno de los cuatro componentes principales de la base de datos: datos, hardware, software y usuarios; así como su función dentro de la base de datos. Además, presentaremos aspectos importantes de por qué utilizar un sistema de base de datos. La presentación de la arquitectura sirve de marco de referencia para la exposición de nuestro tema.

Termina esta primera parte con un breve comentario sobre ¿qué es una base de datos distribuida?, presentando también sus ventajas y desventajas.

La segunda parte trata sobre un problema central para el diseño de cualquier sistema de bases de datos, a saber, de qué manera habrán de concebir los usuarios la base de datos, es decir, el problema de a qué debería parecerse la base de datos para el usuario. Por tanto en la segunda parte se introducen los tres métodos principales para resolver este problema: el enfoque jerárquico, el enfoque de red y el enfoque relacional.

En la tercera parte se habla acerca del control de integridad en los sistemas de bases de datos relacionales, presentando el concepto de integridad así como la distribución de su control.

A partir del capítulo cuatro se va a tratar el tema principal de este trabajo:

UNIFY 2000. Se responderá primero a la pregunta: ¿qué es UNIFY 2000?, seguida de las características de rendimiento, configuración, portabilidad y seguridad de UNIFY 2000. Después de esto, se mostrará su terminología y sus componentes para poder entender mejor todos los términos usados en este trabajo.

El capítulo quinto trata de la arquitectura de UNIFY 2000, al igual que de su ingeniería. Así se podrá comprender su estructura y la manera cómo funciona.

Después de haber dado una idea general de UNIFY 2000 se entrará, en el capítulo sexto, a explicar cómo opera la administración de recursos, para después continuar con una explicación sencilla, en el capítulo siguiente, del SQL/A (Structured Query Language).

El capítulo octavo trata acerca de la operación de la base de datos, junto con la explicación de sus esquemas y de la integridad de datos. Y al final, en el último capítulo, se esboza un tema importante, los métodos de acceso a la información que utiliza UNIFY 2000: hash, links, b tree, direct key y secuencial.

Esperamos, entonces, que el trabajo sea agradable para todos sus lectores y que aprendan algo nuevo acerca de este tema.

1. CONCEPTOS BASICOS

1.1. ¿Qué es un sistema de bases de datos?

La tecnología de las bases de datos se ha descrito como "una de las áreas de la ciencia de la computación y la información de más rápido desarrollo". El campo rápidamente ha cobrado importancia práctica y teórica. La cantidad total de datos encomendados a las bases de datos se mide en varios miles de millones de *bytes*; la inversión financiera al respecto alcanza una cifra igualmente enorme; y no es exagerado afirmar que muchos miles de organizaciones dependen de la operación continuada y eficaz de un sistema de bases de datos.

¿Qué es exactamente un sistema de bases de datos? En esencia, no es más que un sistema de mantenimiento de registros basado en computadores y controlados centralmente, es decir, un sistema cuyo propósito general es registrar y mantener información. Tal información puede estar relacionada con cualquier cosa que sea significativa para la organización donde el sistema opera, cualquier dato necesario para los procesos de toma de decisiones inherentes a la administración de la organización. En la Figura 1 se muestra una representación muy simplificada de un sistema de bases de datos. Con esto se pretende indicar que un sistema de bases de datos incluye cuatro componentes principales: datos, hardware, software y usuarios.

1.2 Componentes principales de un sistema de bases de datos

1.2.1. Datos

Una base de datos en un repositorio de datos almacenados es tanto *integrada* como *compartida*.

Por *integrada* se entiende que la base de datos puede considerarse como una unificación de varios archivos de datos independientes, donde se elimina parcial o totalmente cualquier redundancia entre los mismos.

Por *compartida* se entiende que partes individuales de la base de datos pueden compartirse entre varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos (y utilizarla con propósitos diferentes). Como consecuencia del hecho de que la base de datos es integrada, se advierte que cualquier usuario específico, por lo general, tendrá acceso tan sólo a algún subconjunto de la base de datos completa. En

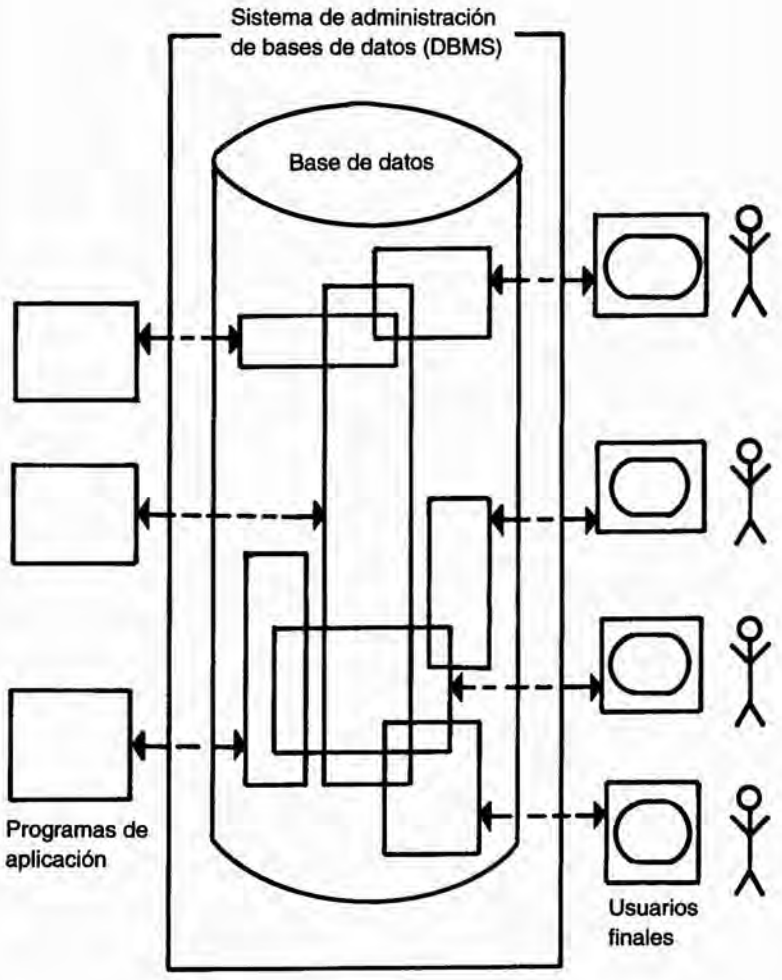


Figura 1
Representación simplificada de un sistema de bases de datos

otras palabras, diferentes usuarios percibirán de modos muy distintos una base de datos específica.

La palabra *compartida* no sólo abarca el compartimiento antes descrito, sino también el compartimiento *concurrente*: es decir, la oportunidad de que diversos usuarios accesen en realidad la base de datos (tal vez la misma parte de la base de datos) al mismo tiempo.

1.2.2 Hardware

El hardware se compone de los volúmenes de almacenamiento secundario donde reside la base de datos, junto con dispositivos asociados como las unidades de control, los canales, etc. Esta parte no se tocará en profundidad debido a que los problemas inherentes a esta área no son peculiares de los sistemas de bases de datos.

1.2.3. Software

Entre la base de datos física (el almacenamiento real de los datos) y los usuarios del sistema existe un nivel de *software*, que a menudo recibe el nombre de *sistema de administración de bases de datos* o *DBMS*. Este controla el almacenamiento y maneja todas las solicitudes de acceso a la base de datos formuladas por los usuarios.

1.2.4 Usuarios

Se consideran tres clases generales de usuarios:

— *Programador de aplicaciones:* es el encargado de escribir programas de aplicación que utilicen bases de datos. Estos programas de aplicación operan con los datos de todas las maneras usuales: recuperan información, crean información nueva, suprimen o cambian información existente, etc.

— *Usuario final:* es el que accesa la base de datos desde una terminal. Un usuario final puede emplear un lenguaje de consulta proporcionado como parte integral del sistema o recurrir a un programa de aplicación escrito por un usuario programador que acepte órdenes desde la terminal y a su vez formule solicitudes al DBMS.

— *Administrador de bases de datos o DBA:* es la persona cuya responsabilidad central es la de controlar los datos de operación de la base de datos.

1.3. ¿Por que utilizar bases de datos?

La respuesta general a esta pregunta es que un sistema de bases de datos proporciona a la empresa un control centralizado de sus datos de operación. Esto para que los datos de operación no se hallen muy dispersos y no sean muy difíciles de controlar.

1.3.1 Ventajas de los sistemas de bases de datos

— *Puede reducirse la redundancia*

En sistemas que no usan bases de datos, cada aplicación tiene sus propios archivos privados y esto genera a menudo gran redundancia en los datos almacenados, así como desperdicio resultante del espacio de almacenamiento. Por lo tanto, en un sistema de bases de datos debe controlarse y reducirse la redundancia mediante la integración de archivos individuales.

— *Puede evitarse la inconsistencia*

Esto es corolario del punto anterior. Por ejemplo, el hecho de que el empleado E8 trabaja en el departamento D8, se representa por dos entradas distintas en la base de datos, y el sistema no está al tanto de esta duplicación (no se ha controlado la redundancia). Habrá un momento en que las dos entradas no concuerden (cuando una de ellas se ha actualizado y la otra no). En tales circunstancias se dice que la base de datos es *inconsistente*. Y una base de datos en este estado puede suministrar información incorrecta o contradictoria.

— *Los datos pueden compartirse*

No sólo significa que las aplicaciones existentes pueden compartir los datos de la base de datos, sino también que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados.

— *Pueden hacerse cumplir las normas establecidas*

Con un control central de la base de datos, el DBA puede garantizar que se cumplan todas las formas aplicables a la representación de los datos. Esto puede comprender las normas de la compañía, de instalación, departamentales, industriales, nacionales o internacionales. Esto permite unificar los formatos de los datos almacenados como ayuda para el intercambio de datos entre sistemas.

— *Pueden aplicarse restricciones de seguridad*

El DBA puede asegurar que el único medio de accesar la base de datos sea a través de los canales establecidos y definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles.

— *Puede conservarse la integridad*

El problema de la integridad es garantizar que los datos de la base de datos sean exactos. La inconsistencia entre dos entradas que representan el mismo "hecho" es un ejemplo de falta de integridad.

— *Pueden equilibrarse los requerimientos contradictorios*

El DBA puede estructurar el sistema de bases de datos para brindar un servicio que sea el mejor para la empresa, en términos globales, al conocer los requerimientos globales de una empresa.

— *Provisión de independencia de datos*

Este constituye más un objetivo que una ventaja. Por esto se explicará un poco más a fondo.

1.4. Independencia de los datos

Miremos primero el fenómeno opuesto a la independencia de los datos. Que las aplicaciones sean dependientes de los datos significa que la manera como los datos se almacenan en almacenamiento secundario y la manera como se accesan dependen de los requerimientos de la aplicación. Por ejemplo, digamos que un archivo particular se almacena en forma secuencial con índices. La aplicación de este archivo es *dependiente de los datos* porque es imposible cambiar la estructura de almacenamiento (la manera como están registrados físicamente los datos) o la estrategia de acceso sin afectar la aplicación.

En un sistema de bases de datos sería muy indeseable permitir que las aplicaciones fueran dependientes de los datos, al menos por las siguientes dos razones:

1. Aplicaciones diferentes requerirán vistas diferentes de los mismos datos.
2. El DBA debe tener libertad de modificar la estructura de almacenamiento o la estrategia de acceso en respuesta al cambio de necesidades, sin tener que alterar las aplicaciones existentes; por ejemplo, la empresa puede adoptar nuevas normas; las prioridades de las aplicaciones pueden cambiar; nuevos tipos de almacenamiento pueden aparecer en el mercado, etc.

Puede concluirse que la provisión de independencia de los datos es un objetivo esencial de los sistemas de bases de datos. Es posible definir la independencia de los datos como la *inmunidad de las aplicaciones a los cambios de la estructura de almacenamiento y de la estrategia de acceso.*

1.4.1. Representación de datos numéricos

Un campo numérico puede almacenarse en forma de aritmética interna o

como una hilera de caracteres. En cada caso, el DBA debe elegir una base adecuada (por ejemplo, binaria o decimal), una escala (punto fijo o flotante), el modo (real o complejo) y el nivel de precisión (número de dígitos). Cualquiera de estos aspectos puede cambiarse para mejorar el desempeño o para ajustarse a una nueva norma, o por otras razones.

1.4.2. Representación de datos de caracteres

Un campo de hilera de caracteres puede almacenarse en cualquiera de los diversos códigos de caracteres (por ejemplo, EBCDIC, ASCII).

1.4.3. Unidades de los datos numéricos

Las unidades de un campo numérico pueden cambiar (de pulgadas a centímetros, por ejemplo) durante el proceso de conversión al sistema métrico.

1.4.4. Codificación de los datos

En algunas ocasiones puede ser conveniente representar los datos en almacenamiento por medio de valores codificados; por ejemplo, un almacenamiento en forma de un dígito decimal puede interpretarse de acuerdo a la siguiente tabla: 1="rojo", 2="azul", etc.

Entonces, la base de datos debe ser capaz de crecer sin afectar las aplicaciones existentes. Permitir que la base de datos crezca sin perjudicar las aplicaciones existentes es la razón principal para proporcionar la independencia de los datos.

1.5. Arquitectura de un sistema de bases de datos

La arquitectura se divide en tres niveles generales: interno, conceptual y externo (Figura 2). El nivel *interno* es el más cercano al almacenamiento físico, es decir, el que concierne a la manera como los datos se almacenan en realidad; el nivel *externo* es el más cercano a los usuarios, es decir, el que atañe a la manera cómo cada usuario ve los datos; y el nivel *conceptual* es un nivel de "mediación" entre los otros dos. Si el nivel externo se relaciona con las vistas de los usuarios individuales, el nivel conceptual puede considerarse como el que define una vista de la comunidad de usuarios (recuérdese que a los usuarios sólo les interesa una parte de la base de datos). Asimismo, habrá una sola "vista interna", que representa la base de datos total tal como está almacenada.

La vista de un usuario individual se llama *vista externa*. Por tanto, una vista externa es el contenido de la base de datos tal como lo ve un usuario específico.

Figura 2
Los tres niveles de la arquitectura

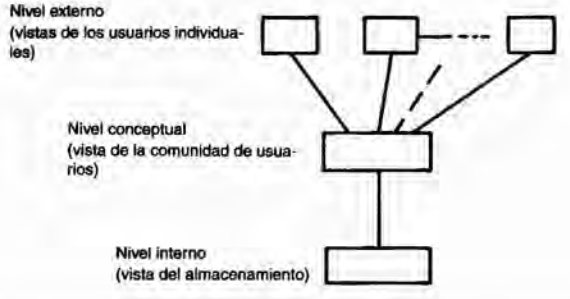
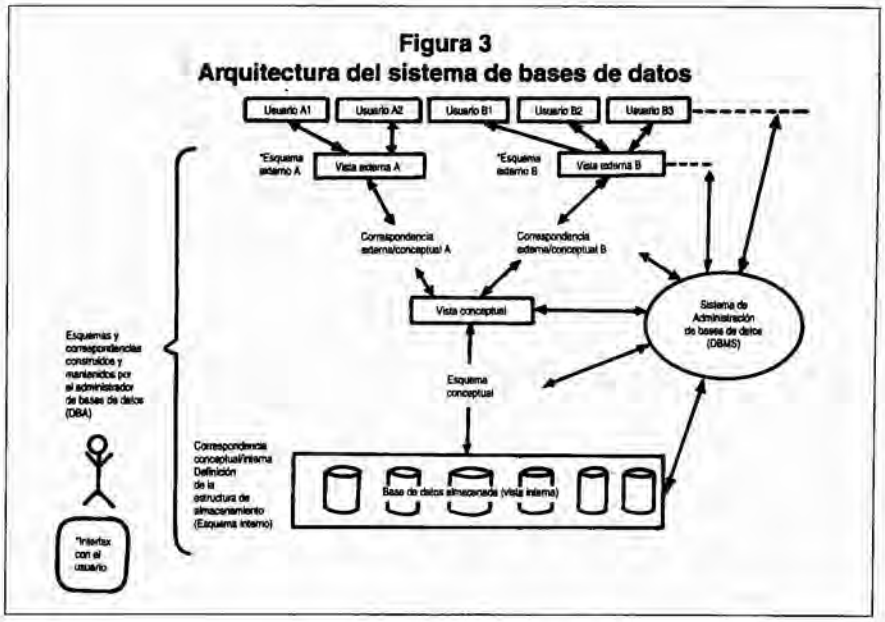


Figura 3
Arquitectura del sistema de bases de datos



La *vista conceptual* es una representación del contenido total de información de la base de datos, en forma relativamente abstracta en comparación con la forma como los datos se almacenan físicamente.

El tercer nivel de la arquitectura es el interno. La *vista interna* es una representación de un nivel muy bajo de la base de datos en su totalidad.

Los componentes de la arquitectura de una base de datos se muestran en la Figura 3. Considerando esta figura, vemos cómo el *administrador de la base de datos* es la persona (o grupo de personas) encargada del control general del sistema de bases de datos.

Por eso es importante que miremos las responsabilidades del DBA:

— *Decidir el contenido de la información de la base de datos*

Es trabajo del DBA decidir con exactitud qué información se mantendrá en la base de datos, es decir, identificar las entidades de interés para la empresa.

— *Decidir la estructura de almacenamiento y la estrategia de acceso*

El DBA también debe decidir de qué manera habrán de representarse los datos en la base de datos y especificar dicha representación.

— *Vincularse con los usuarios*

Es responsabilidad del DBA vincularse con los usuarios, garantizar que los datos que requieran estén disponibles.

— Definir los controles de autorización y los procedimientos de validación

— *Definir una estrategia de respaldo y de recuperación*

El DBA debe definir y poner en marcha una estrategia de recuperación adecuada, que incluya, por ejemplo, vaciado periódico de la base de datos en una cinta de respaldo y procedimientos para reponer las partes pertinentes de la base de datos desde la cinta más reciente. Todo esto para prevenir un daño de alguna parte de la base de datos (debido a un error humano, o a una falla en el hardware o en el sistema operativo de

apoyo, etc.) y para poder recuperar los datos con la mayor brevedad posible y reduciendo al mínimo las repercusiones en el resto del sistema.

— *Controlar el desempeño y responder a los cambios de requerimientos*

El DBA se encarga de organizar el sistema de tal manera que se logre un desempeño que sea "el mejor para la empresa", así como de hacer los ajustes adecuados a medida que los requerimientos cambian.

1.6. Lenguajes de bases de datos

Cada usuario tiene un lenguaje a su disposición. Para el programador de aplicaciones, se trata de un lenguaje convencional de programación como COBOL o PL/I; para el usuario de una terminal se trata de un lenguaje de consulta o de un lenguaje de propósito especial hecho a la medida de sus necesidades. Estos lenguajes se denominan regularmente lenguajes *anfitriones*. Cada lenguaje anfitrión incluye un *sublenguaje de datos (data sublanguage,* DSL), es decir, un subconjunto del lenguaje total que concierne a los objetos y a las operaciones de la base de datos.

En principio, cualquier sublenguaje de datos en realidad es una combinación de dos lenguajes: un *lenguaje de definición de datos (data definition language,* DDL), que permite la definición o descripción de los objetos de la base de datos (tal como los percibe el usuario), y un *lenguaje de manipulación de datos (data manipulation language,* DML), que apoya el manejo o procesamiento de esos objetos.

1.7. Bases de datos distribuidas

1.7.1. Introducción

La tecnología de sistemas de bases de datos distribuidas es uno de los más recientes desarrollos en el área de sistemas de bases de datos. Se espera que en los próximos años las bases de datos centralizadas sean "curiosas antigüedades" y más organizaciones se muevan hacia bases de datos distribuidas. El gran interés de parte de la comunidad y del comercio permite pensar en esto. La gran actividad de investigación en la última década ha generado resultados que ahora son la introducción de los productos comerciales en el mercado.

La tecnología de los sistemas de bases de datos distribuidas (DDBS) es la unión de dos áreas opuestas para el procesamiento de datos: *sistemas de bases de datos* y tecnologías de *redes de computadores.* Una de las mayores motivaciones para el uso de las bases de datos es el deseo de integrar los datos operacionales de una empresa y lograr provisión centralizada para el acceso controlado a los datos. Y la tecnología de las redes de computadores, por otro lado, promueve un modo de trabajo que va en contraste de todos los esfuerzos de centralización. Un primer vistazo acerca de esto podría ser la dificultad de entender cómo esos dos contrastes pueden ser sintetizados para producir una tecnología que es más poderosa y más promisoria que cada una de ellas por sí sola. La clave está en entender que el más importante objetivo de la tecnología de las bases de datos es la *integración*, y no la *centralización.* Es importante anotar que la realización de uno de estos dos términos no implica necesariamente la utilización del otro. Es posible alcanzar la integración sin centralización, y esto es exactamente lo que la tecnología de las bases de datos distribuidas intenta alcanzar.

1.7.2. Definición

Miremos primero la definición de un sistema de cómputo distribuido: es un número de elementos de procesamiento autónomos (y no necesariamente homogéneos) que están interconectados por una red de computadores y que

cooperan en la realización de sus tareas asignadas.

Una pregunta fundamental que necesita ser respondida es: ¿Qué está siendo distribuido? Una de las cosas que podría ser distribuida es el *procesamiento lógico*. Otra posible distribución está de acuerdo con la *función*. Varias funciones de un sistema de cómputo pueden ser delegadas a varias partes del software o del hardware. Un tercer modo de distribución está de acuerdo con los datos. Los datos usados por las aplicaciones podrían ser distribuidos en varios sitios de procesamiento. Finalmente, el *control* puede ser distribuido. El control de la ejecución de varias tareas podría ser distribuido en vez de ser realizado por un solo sistema de cómputo. Desde el punto de vista de los sistemas de bases de datos distribuidas, estos modos de distribución son todos necesarios e importantes.

Observemos entonces: ¿qué es un sistema de bases de datos distribuida?:

Una base de datos distribuida es una base de datos no almacenada en su totalidad en un solo lugar físico, sino que se distribuye a lo largo de una red de computadores geográficamente separados que se conectan por medio de enlaces de comunicación. Otra definición es la siguiente: es una colección de múltiples y lógicamente relacionadas bases de datos que están distribuidas a través de una red de computadores. Un sistema de administración de bases de datos distribuidas (DDBMS) es definido entonces como el sistema de software que permite la administración del sistema de administración de bases de datos (DBMS) y hace la distribución transparente a los usuarios.

Un objetivo básico de un sistema distribuido es que el usuario lo perciba como un sistema centralizado; así pues, el hecho de que la base de datos esté distribuida debe ser importante tan sólo a nivel interno, y no a los niveles externo o conceptual.

Miremos un ejemplo muy simplificado: consideremos un sistema bancario donde la base de datos de las cuentas de los clientes se distribuye en todas las sucursales del banco, y donde el registro de la cuenta de cada cliente se almacena en la sucursal más cercana al domicilio o lugar de trabajo de éste. Los datos se almacenan en el sitio donde se usan con mayor frecuencia, pero también están disponibles, por medio de la red de comunicaciones, para los usuarios de otros lugares. Las ventajas de tal distribución son elocuentes: combina la eficiencia del procesamiento local (sin excesivos costos de comunicación) de la mayoría de las operaciones y todas las ventajas antes mencionadas (en particular, compartir datos) que ofrece un sistema centralizado. Pero también existen desventajas: los costos de comunicación pueden ser elevados y existen dificultades técnicas significativas para instrumentar un sistema así.

1.7.3. Ventajas y desventajas de los sistemas de bases de datos distribuidas

1.7.3.1. Ventajas

— *Autonomía local:* los datos están distribuidos, y hay un grupo de usuarios que comúnmente comparte algunos de estos datos. Por tanto, estos datos pueden estar localizados en el lugar de trabajo de los usuarios, teniendo así un control local.

— *Mejoría de la performance:* debido a que regularmente los datos usados están próximos a los usuarios y al paralelismo inherente en los sistemas distribuidos, es posible mejorar el desempeño en los accesos a las bases de datos.

— *Mejoría de la confiabilidad y disponibilidad:* si hay un daño en un sitio donde residen los datos o una falla de comunicación que hace a este sitio in-

accesible, no necesariamente significa que los datos sean imposibles de alcanzar.

— *Economía:* es posible ver esto desde dos puntos de vista. El primero está en términos de los costos de comunicación. Si la base de datos está geográficamente dispersa y las aplicaciones corren con una fuerte interacción de datos dispersos puede ser más económico partir las aplicaciones y hacer el procesamiento localmente en cada sitio. El segundo punto de vista es que normalmente son menores los costos de poner juntos un sistema de pequeños computadores con el equivalente poder de una sola gran máquina.

— *Expansibilidad:* en un medio distribuido, es mucho más fácil acomodar el incremento de tamaño de las bases de datos.

— *Compartimiento:* las organizaciones que tienen operaciones distribuidas geográficamente normalmente almacenan datos de una forma distribuida. Si los sistemas de información no están distribuidos, a veces es imposible compartir datos y recursos.

1.7.3.2. Desventajas

— *Complejidad:* los problemas del sistema de bases de datos distribuidas son más complejos que los de la administración de bases de datos centralizadas. Estos no solamente incluyen los problemas de un medio centralizado sino también un nuevo conjunto de problemas no resueltos.

— *Costos:* los sistemas distribuidos requieren hardware adicional, como mecanismos de comunicación, que han incrementado sus costos.

— *Distribución de control:* este punto fue establecido previamente como una ventaja, pero desafortunadamente la distribución crea problemas de sincronización y coordinación.

— *Seguridad:* uno de los mayores beneficios de las bases de datos centralizadas ha sido el control que se provee en el acceso a los datos. Y un sistema de bases de datos distribuidas involucra una red, que es un medio que tiene sus propios requerimientos de seguridad.

— *Dificultad de cambio:* muchos negocios ya han invertido fuertemente en sus sistemas de bases de datos que no son distribuidos. Y el problema se presenta en que no hay las herramientas o metodologías suficientes para ayudar a esos usuarios a convertir sus bases de datos centralizadas en sistemas de base de datos distribuidas.

2. ENFOQUES DE LAS BASES DE DATOS

Es conveniente clasificar los sistemas de bases de datos de acuerdo con el enfoque que adoptan. Los tres enfoques mejor conocidos son:

— el enfoque relacional;

— el enfoque jerárquico; y

— el enfoque de red.

2.1. El enfoque jerárquico

En el modelo jerárquico (Figura 4.), los elementos de información participan en una interrelación padre/hijo; cada padre puede tener varios hijos, pero cada hijo sólo puede tener un padre.

En la Figura 5 se muestra una posible vista jerárquica para la base de datos de proveedores y partes. En esta vista los datos se representan por una sencilla estructura de árbol, con las partes a un nivel superior a los proveedores. El usuario ve cuatro árboles individuales, uno para cada parte. Cada árbol se compone de un registro de la parte, junto con un registro de proveedor subordinado, uno para cada proveedor de la parte. Cada registro de proveedor incluye la cantidad enviada correspondiente.

Figura 4
Organización jerárquica de una base de datos.

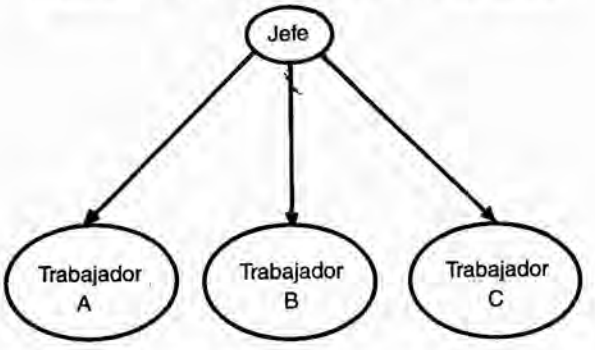
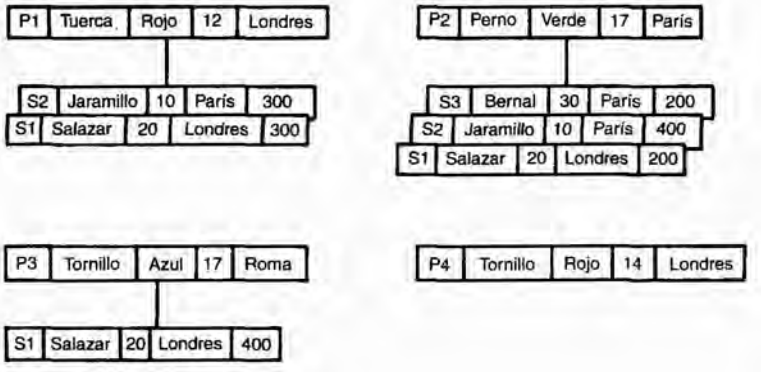


Figura 5
Datos muestra en forma jerárquica (las partes superiores a los proveedores).



El tipo de registro en el tope del árbol (en la Figura 5, el registro de la parte) se conoce en general como "raíz". La raíz puede tener cualquier número de dependientes; cada uno de éstos puede tener cualquier número de dependientes de nivel inferior, y así sucesivamente, hasta cualquier número de niveles.

El enfoque jerárquico conduce a complicaciones innecesarias para el usuario. La organización jerárquica dificulta la expresión de interrelaciones en las cuales los hijos se relacionan con más de un padre. En términos específicos, el usuario está obligado a dedicar tiempo y esfuerzo a resolver problemas introducidos por la estructura de datos jerárquica. Es claro que la situación empeorará con rapidez a medida que se introduzcan más tipos de registros en la estructura y la jerarquía se vuelva más compleja. Esto significa que los programas son más complicados de lo necesario, con la consecuencia de que su escritura, depuración y mantenimiento necesitarán más tiempo de programador del debido.

Consideremos tres funciones básicas: *insertar, suprimir y actualizar.*

Insertar: No es posible, sin introducir una parte ficticia especial, insertar datos relativos a un proveedor nuevo (por ejemplo, S4) hasta que ese proveedor suministre alguna parte.

Suprimir: Puesto que la información de las remesas está incorporada en el tipo de registro de proveedor, la única manera de suprimir una remesa es suprimir la correspondiente ocurrencia de proveedor. Se sigue que si suprime la única remesa de un proveedor específico, se pierde toda la información de ese proveedor. En el ejemplo, este caso se presentaría si se elimina S3.

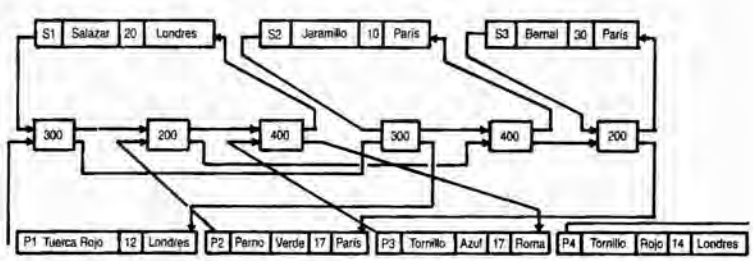
Por otro lado, surge un problema similar si se desea suprimir una parte que por casualidad es la única suministrada por algún proveedor, porque la supresión de cualquier ocurrencia de registro automáticamente suprime todas las ocurrencias dependientes, de acuerdo con la filosofía jerárquica.

Actualizar: Si se necesita cambiar la descripción de un proveedor, por ejemplo, cambiar la ciudad del proveedor S1 a Amsterdam, es preciso enfrentar el problema de buscar en toda la vista para hallar cada ocurrencia del proveedor S1, o la posibilidad de introducir una inconsistencia.

2.2. El enfoque de red

En la Figura 6 se muestra una vista de red para la base de datos de proveedores de partes. Una red es una estructura más general que una jerárquica por que una ocurrencia de registro específica puede tener cualquier número de superiores inmediatos (así como cualquier número de dependientes inmediatos), es decir, no está limitado a un máximo de un padre, como ocurre con una jerarquía. De esta manera, el enfoque de red permite modelar una correspondencia mucho más general que en el enfoque jerárquico.

Figura 6
Datos muestra en forma de red



Insertar: Para insertar datos relativos a un nuevo proveedor (S4, por ejemplo) sencillamente se crea una ocurrencia de registro de proveedor.

Suprimir: Para suprimir la remesa que conecta a P2 y S3, se suprime la ocurrencia del conector que liga a este proveedor y esta parte.

Actualizar: Se puede cambiar la ciudad del proveedor S1 a Amsterdam sin ningún problema de búsqueda y sin la posibilidad de la inconsistencia, porque la ciudad de S1 aparece precisamente en un sitio de la estructura.

Se afirma, por tanto, que la desventaja del enfoque de red es su excesiva complejidad, y una estructura así puede ser difícil de comprender, modificar o reconstruir en caso de falla. En ambientes muy dinámicos en los cuales se espera un crecimiento considerable de la base de datos, o cuando es muy probable la adición de nuevas características e interrelaciones, es mejor evitar el enfoque de red.

2.3. El enfoque relacional

En la figura 7 se muestra una vista relacional de los datos de proveedores y partes. Se puede observar que los datos se organizan en tres tablas: S (proveedores), P (partes) y SP (remesas). Las tablas como las de la Figura 7 se denominan *relaciones*, las filas de las tablas se denominan *tuplas*, las columnas *atributos o dominio*, y el primer campo de cada tupla *clave o llave primaria.* El número de dominios en una relación indica el grado de la relación y el número de tuplas de una relación se llama *cardinalidad* de la relación.

Figura 7
Datos muestra en forma relacional

S

S#	NOMS	ESTADO	CIUDAD
S1	Salazar	20	Londres
S2	Jaramillo	10	París
S3	Bernal	30	París

P

P#	NOMP	COLOR	PESO	CIUDAD
P1	Tuerca	Rojo	12	Londres
P2	Perno	Verde	17	París
P3	Tornillo	Azul	17	Roma
P4	Tornillo	Rojo	14	Londres

SP

S#	P#	CTD
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P3	200

Obsérvese que las relaciones S y SP tienen un dominio en común (el de los números de proveedor); también P y SP (el de los números de parte), y también S y P (el de los sitios). Una característica crucial de la estructura de datos relacional es que las asociaciones entre tuplas se representan únicamente por valores de datos en columnas sacadas de un dominio común. Otra característica del enfoque relacional es que toda la información de la base de datos se representa de una sola manera uniforme, a saber, en forma de tablas. Esta característica no es compartida por los enfoques jerárquico y de red.

Insertar: Dada la información acerca de un nuevo proveedor S4, simplemente se inserta esta información en la relación S.

Suprimir: Para suprimir la remesa que conecta la parte P2 con el proveedor S3, se suprime la tupla de SP donde P# = P2 y S# = S3.

Actualizar: El proveedor S1 se trasladó de Londres a Amsterdam, se cambia CIUDAD en la tupla de S donde S# = S1 a Amsterdam.

Considerando una operación de consulta, cualquier operación de recuperación puede considerarse una tabla. Por ejemplo: Halle S# y ESTADO de los proveedores de París. El resultado sería: S# ESTADO.

S2 10

S3 30

El resultado es una tabla, con dos renglones y dos columnas.

En general, pues, el resultado de cualquier recuperación es una tabla, derivada de alguna manera de las tablas de la base de datos; para formar el resultado se puede utilizar cualquier número de tablas, tanto al determinar la selección como al suministrar los valores reales del resultado. En otras palabras, *el proceso de recuperación es un proceso de construcción de tablas.* Al conocer este hecho, se puede definir un conjunto de operadores de construcción de tablas para usarlos en la recuperación. Tres de esos operadores son: Select (seleccione), Project (proyecte) y JOIN (reúna).

El operador Select construye una nueva tabla al tomar un subconjunto horizontal de una tabla existente, es decir, todos los renglones de una tabla existente que satisfagan una condición.

El operador Project, en contraste, forma un subconjunto vertical de una tabla existente al extraer determinadas columnas y suprimir renglones duplicados redundantes en el conjunto de columnas extraídas.

El operador JOIN combina tablas más pequeñas para formar otras más grandes y así producir relaciones más complejas.

La organización relacional de las bases de datos tiene muchas ventajas con respecto a los esquemas jerárquico y de red:

La representación de tablas empleada en el enfoque relacional es entendida con facilidad por los usuarios, y también es fácil de llevar a la práctica en el sistema físico de bases de datos.

Es relativamente fácil convertir casi cualquier otro tipo de estructura de bases de datos al esquema relacional. Puede considerarse al enfoque relacional como una forma universal de representación.

Se facilita la creación de nuevas relaciones requeridas para aplicaciones específicas.

La puesta en práctica del control de acceso a datos sensibles es sencilla. Basta colocar los datos delicados en relaciones separadas y controlar el acceso a estas relaciones mediante algún tipo de esquema de autorización o acceso.

Las búsquedas pueden ser más rápidas que en esquemas en los cuales se siguen cadenas de apuntadores.

Las estructuras relacionales son más fáciles de modificar que las estructuras jerárquicas o de red. Esto es vital en ambientes donde la flexibilidad es importante.

La claridad y visibilidad de los datos mejora con la estructura relacional. Es mucho más fácil examinar datos tabulares que desenmarañar interconexiones de complejidad posiblemente arbitraria entre elementos de información con un mecanismo de apuntadores.

3. CONTROL DE INTEGRIDAD EN LOS SISTEMAS DE BASES DE DATOS RELACIONALES

3.1. Introducción

La complejidad de las modernas aplicaciones de bases de datos requiere poderosas facilidades para controlar la exactitud de los datos en las bases de datos. Los requerimientos de control de semántica de datos han conducido a varios proyectos de investigación en el contexto de sistemas de bases de datos relacionales. El primer proyecto comenzó en los mediados de los setentas, junto con el desarrollo de los primeros sistemas de administración de bases de datos relacionales, como el System R y el INGRES. Los problemas básicos han llegado a ser claros ahora, pero las investigaciones continúan en nuevas áreas, como las especificaciones de restricciones y definición de complejos tipos de restricciones, verificación y optimización de especificaciones de restricciones, algoritmos de ejecución de restricciones eficientes y paralelismo en la ejecución de las restricciones.

3.2. El concepto de integridad

En los sistemas de administración de bases de datos, la exactitud y corrección de los datos es de gran importancia. Hay varias maneras en que los datos incorrectos pueden aparecer en la base de datos. Las siguientes disciplinas en la tecnología de las bases de datos tratan de prevenir ciertas clases de errores:

* ***Control de seguridad:*** se ocupa de prevenir a los usuarios de accesar y modificar los datos en la base de datos sin autorización; los subsistemas de control de seguridad de DBMS guardan un registro de las autorizaciones de los usuarios para realizar ciertas operaciones en ciertos datos y revisa esta autorización a través del acceso a la base de datos.

* ***Control de concurrencia:*** se ocupa de la prevención de inconsistencias causadas por el acceso concurrente de múltiples usuarios o múltiples aplicaciones a la base de datos; el subsistema de control de concurrencia orquesta el acceso a la base de datos, usando en la mayoría de los casos un locking o una técnica de tiempo de "sellado".

* ***Control de confiabilidad:*** se ocupa de la prevención de errores debido al mal funcionamiento del hardware o del software; el subsistema de control de confiabilidad usa una técnica de recuperación para reinstalar la base de datos después de que el sistema se haya caído, y técnicas como la réplica de datos para incrementar la confiabilidad del sistema.

* ***Control de integridad:*** se ocupa de la prevención de los errores semánticos cometidos por los usuarios debido a sus descuidos o falta de conocimientos; el subsistema de control de integridad usa reglas de integridad para verificar la base de datos y las operaciones realizadas en ella.

3.3. Distribución del control de integridad

Una importante pregunta para ser respondida es quién o qué es responsable de guardar la consistencia de la base de datos con un número de restricciones de integridad especificadas (control de integridad). En general, la tarea de control de integridad puede ser distribuida de tres maneras:

* ***Diseñador de aplicaciones:*** la tarea de control de integridad puede ser dejada al diseñador de aplicaciones o a un usuario ad hoc del sistema de base de datos. La falta de fuerza con respecto a la integridad se presenta en todo: las aplicaciones y transacciones requieren ser individualmente correctas respecto a las restricciones de integridad.

* ***Diseñador de transacciones:*** la tarea de control de transacciones puede ser responsabilidad de un diseñador de transacciones. Esto significa que las transacciones requieren ser correctas con respecto a las restricciones especificadas. En esta situación, las aplicaciones pueden hacer uso solamente de transacciones predefinidas.

* ***Sistema de administración de bases de datos:*** el sistema de administración de bases de datos puede tener la responsabilidad del control de integridad. Esto significa que las transacciones arbitrarias pueden ser ejecutadas en el sistema.

En el primer caso, la integridad de la base de datos no es garantizada por una especificación central de la misma base de datos, solamente por el diseño de las siempre numerosas y cambiantes operaciones de aplicaciones en la base de datos. Consecuentemente, la posibilidad de hacer un control de restricciones impropias es alta. Además, los cambios en algunas definiciones de restricciones requieren modificaciones de todas las aplicaciones incluyendo el control de integridad con respecto a esas definiciones. Esta es una situación indeseable, y esto puede ocurrir frecuentemente en la práctica.

En el segundo caso, la responsabilidad del control de integridad es parte del proceso de diseño de transacciones. De nuevo, los cambios en la definición de restricciones requieren modificaciones de todas las transacciones incluyendo control de integridad con respecto a esas definiciones. En una situación con muchas restricciones, las transacciones pueden llegar a ser muy complejas. Además, esta situación es difícilmente posible en una situación con una alta rata de transacciones correctas.

Si la tarea de control de integridad es localizada con el sistema de administración de bases de datos, la integridad de la base de datos es efectivamente regulada por un set central de definiciones de restricciones. Las aplicaciones y transacciones pueden ser totalmente desatendidas del control de integridad. Una desventaja de esta situación puede ser una reducida flexibilidad con respecto al manejo de restricciones.

4. UNIFY 2000

4.1. ¿Qué es UNIFY 2000?

El kernel de UNIFY 2000 es un sistema procesador de transacciones. El sistema procesador de transacciones de UNIFY 2000 y sus interfaces conforman una "próxima generación", 100% uptime RDBMS, para desarrolladores de software diseñando en línea y aplicaciones de bases de datos orientadas a transacciones. La Figura 8 muestra el UNIFY 2000 RDBMS.

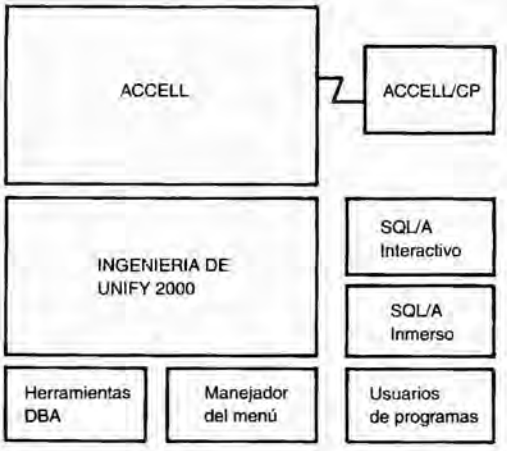
Para asegurar la alta flexibilidad y ejecución, UNIFY 2000 RDBMS incluye las siguientes características:

* Conformación estándar ANSI SQL.
* Seguridad de manejo.
* Transacciones completas de rollback y commit.
* Ejecución de transacción serializable.
* Control de concurrencia de transacciones en múltiples niveles.
* Recubrimiento de fallas en el sistema.
* Volumen y archivos grandes.
* Locking de múltiples niveles.
* Arquitectura expandible y portable.
* Múltiples interfaces.
* Múltiples métodos de acceso.
* Servidores de lock, caché y archivos.
* Facilidades de backup en línea.

4.1.1. Performance (Rendimiento)

* Manejo de buffers, o caché, en memoria compartida, que reduce el número de accesos a disco requeridos para accesar datos.

Figura 8.
UNIFY 2000 RDBMS



* Posee cinco métodos de acceso distintos (llave, hash, links, b-tree, secuencial).
* Manejo de frecuencia en los puntos de sincronización.
* Rápida recuperación después de fallas del sistema.

4.1.2. Configuración

Posee un archivo de configuración del sistema que hace de UNIFY 2000 un sistema altamente configurable y afinable. Este archivo es un archivo ASCII que almacena parámetros fácilmente modificables.

4.1.3. Portabilidad

* Escrito en programación C estándar
* Trabaja en ambiente Unix-Xenix
* El esquema de manejo de archivos asegura que sus sistemas de archivos sean independientes de los archivos físicos del sistema operacional sobre el cual corre UNIFY 2000.

4.1.4. Seguridad

El UNIFY 2000 provee un medio ambiente seguro para aplicaciones de bases de datos. Un ambiente seguro depende de la capacidad para permitir acceso selectivo a porciones específicas de la base de datos, dependiendo del papel del usuario en el desarrollo y mantenimiento de la base de datos.

También depende de la capacidad de permitir privilegios asociados con el acceso a niveles de usuario. Permite asegurar permisos sobre operaciones en tablas (selección, eliminación, consulta).

4.2. Terminología

Los modelos para definir el modelo relacional son:

Relación	=	tabla
Tupla	=	registro o fila
Atributo	=	columna o nombre del campo
Elemento	=	campo
Cardinalidad	=	número de filas en la tabla
Relación binaria	=	tabla de 2 columnas
Relación n-aria	=	tabla de n columnas

Dominio = rango de valores posibles en un campo

4.2.1. Componentes de la base de datos

UNIFY 2000 usa los términos tabla, registro, columna. En la Figura 9 se muestra la tabla de términos de base de datos equivalentes.

UNIFY 2000 incluye otro componente, el esquema. Un esquema es un subconjunto de la base de datos definida; éste consiste de una tabla especificada, de todas las columnas dentro de esta tabla, y de privilegios asociados. Una base de datos contiene al menos un esquema. En la Figura 10 se muestra la jerarquía de los componentes de UNIFY 2000.

Un ejemplo de una base de datos que contiene un esquema con varias tablas es el que se muestra en la Figura 11. El esquema contiene tres tablas, una tabla hash, un índice b-tree, y privilegios. En las tres tablas que existen, una de ellas tiene habilitados dos métodos de acceso (adicionales al secuencial que es implícito).

Figura 9
Tabla de términos equivalentes en bases de datos

Términos equivalentes en bases de datos		
RELACION	TABLA	TIPO DE REGISTRO
TUPLA	FILA	REGISTRO
ATRIBUTO	COLUMNA	CAMPO

Figura 10
Componentes jerárquicos de la base de datos

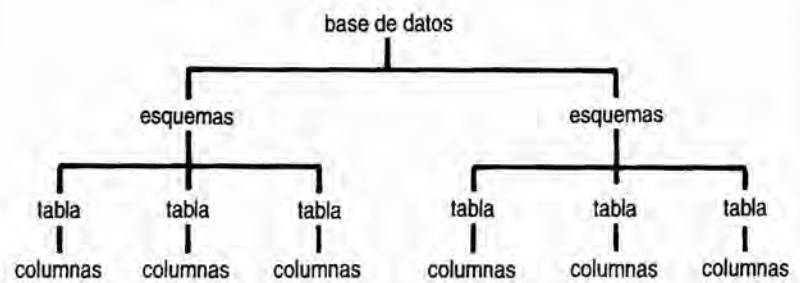
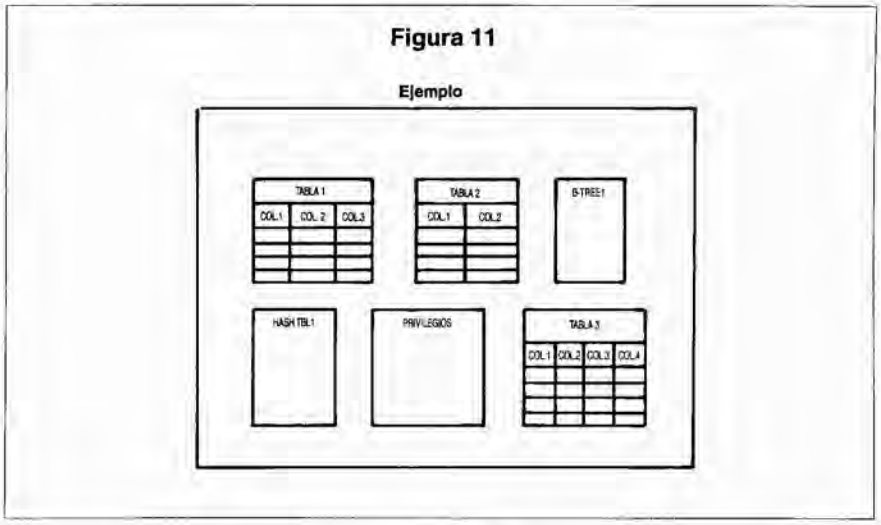


Figura 11



4.3. Construcción de una aplicación en UNIFY 2000

Haciendo caso omiso de las interfaces que se usen, construir una aplicación requiere unos pocos pasos básicos:

1. Desarrollo de un diseño lógico de base de datos.

Para desarrollar un sistema lógico de base de datos, se deben analizar los requerimientos en la aplicación de base de datos. Esto se necesita para determinar los tipos de información que se deben almacenar y cómo deberá estar organizada esta información.

2. Set up del ambiente de base de datos.

Se configura el ambiente del sistema operativo para correr aplicaciones eficientemente.

3. Crear la base de datos.

Para crearla, se usa una de las interfaces de UNIFY 2000, para dar un nombre y crear un archivo.

4. Decisión de los métodos de acceso.

Se debe determinar si se desea accesar datos por una columna llave, hash table, link, b-tree, o secuencialmente.

5. Definición de las tablas y columnas.

Para definir las tablas y columnas en la base de datos, se debe describir la tabla con su nombre y sus columnas especificando el tipo de datos, de las columnas y otras características incluyendo restricciones de integridad de datos.

6. Entrar o cargar datos en las tablas.

Para entrar o cargar datos se pueden usar estos métodos:

— Uso del SQL/A interactivo para insertar registros individualmente o desde un archivo ASCII.

— Uso de las funciones RHLI para insertar registros.

— Uso de las utilidades **dbld** para cargar registros dentro de las tablas.

— Uso de una aplicación ACCELL/SQL para entrar datos en las tablas.

7. Set up seguridad y adición de usuarios a la base de datos.

Se debe determinar cuáles de las bases de datos se desea que los usuarios accesen y cuáles de las operaciones se van a permitir que los usuarios ejecuten.

5. ARQUITECTURA DE UNIFY 2000

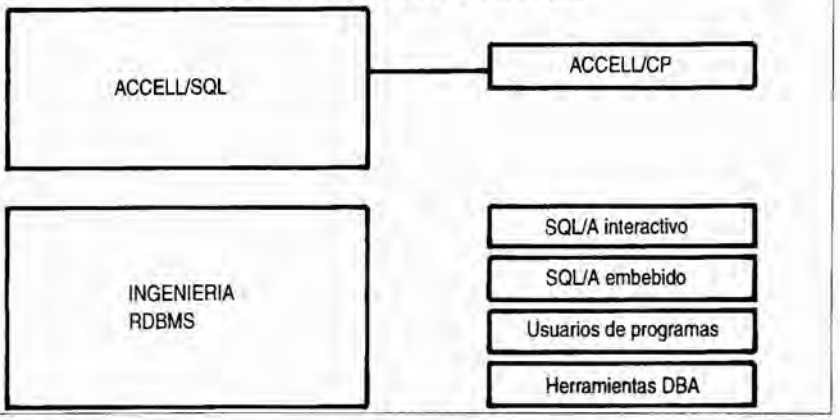
En la Figura 12 se muestra la arquitectura de UNIFY 2000.

El rango de interfaces de UNIFY 2000 de lenguajes de programación para manejar las siguientes utilidades:

ACCELL/SQL es una aplicación integrada de cuarta generación para desarrollo de sistemas.

ACCELL/SQL es un paquete de proceso cooperativo.

Figura 12
Arquitectura UNIFY 2000 RDBMS



SQL/A interactivo es un lenguaje interactivo de indagación de base de datos relacional que usa una sintaxis de palabras claves en inglés.

Embedded SQL/A es un recurso para incluir comandos SQL/A en programas escritos en C u otros lenguajes.

Los programas de usuarios son programas desarrollados que incluyen funciones para diseño, mantenimiento, y manejo de base de datos en UNIFY 2000.

Las herramientas DBA son utilidades shell-level del sistema operativo que ayudan al manejo y mantenimiento de la base de datos.

5.1. Ingeniería de UNIFY 2000 RDBMS

UNIFY 2000 kernel

Es el principal componente del sistema. Su función es coordinar las actividades de todos los demás componentes, a fin de asegurar la operación eficiente de UNIFY 2000.

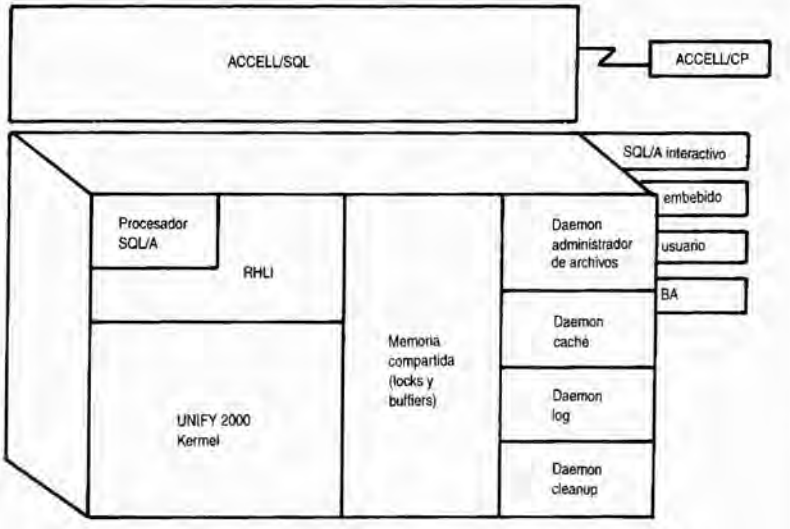
RHLI.

RHLI (Relational Host Language Interface) es una interface entre el UNIFY 2000 kernel y las interfaces de usuarios. Esta poderosa interface se compone de un conjunto de funciones y estructuras diseñadas a fin de permitir al desarrollador definir y manipular los diferentes objetos de la base de datos a través del lenguaje C.

Procesador SQL/A

Es la principal interface que UNIFY 2000 brinda al usuario, permitiéndole definir y manipular los diferentes objetos de las bases de datos, a través de un grupo de comandos fáciles de aprender, debido a su reducido número y a su orientación hacia el lenguaje común (inglés).

Figura 13
Arquitectura-ingeniería de UNIFY 2000



Memoria compartida (locks y buffers)

Se refiere al espacio en memoria donde UNIFY 2000 implementa y maneja los buffers y locks de aplicaciones. Cuando UNIFY 2000 accesa la base de datos, almacena la información de las transacciones y de la base de datos en buffers en la memoria compartida.

Debido a lo anterior UNIFY 2000 minimiza el número de accesos a disco requeridos para obtener la información. Esto implica, además, que UNIFY 2000 no mantiene buffers separados y múltiples copias de las páginas de la base de datos para cada usuario. Es decir, que si un usuario requiere de informacion que ya ha sido leída de la base de datos, UNIFY 2000 obtiene la información de la memoria compartida (caché) sin tener que accesar a través de disco.

Procesos de background o daemon

* **File manager daemon:** Participa en la administración de la base de datos física, así como del log físico. Inicializa procesos de sincronización. En los puntos de sincronización graba en el log físico las actualizaciones hechas en la memoria compartida y envía a la base de datos las páginas actualizadas del log físico. Los puntos de sincronización o sincronismo pueden ser datos en unidades de tiempo o en número de operaciones.

Para poder configurar UNIFY 2000 para una mejor adaptación y manejo de la base de datos física y del log físico se necesita de las siguientes especificaciones y operaciones características del *file manager daemon:*

— La unidad de medida para determinar la frecuencia de los puntos de sin-

cronización. Se pueden medir los puntos de sincronización como un número de operaciones (comienzo o fin de transacciones, actualizar una fila, o renombrar una columna). O se pueden medir los puntos de sincronización como un número de horas, minutos o segundos.

— La frecuencia de puntos de sincronización (el número de operaciones, horas, minutos o segundos entre puntos de sincronismo).

— El nombre del log físico.

— Estatus físico de los archivos logging, ya sea que los archivos estén prendidos o apagados.

— El número mínimo de bloques disponibles en espacio del log físico.

* **Caché daemon:** participa en el manejo de la base de datos física y del log físico, administrando la memoria donde UNIFY 2000 almacena las páginas de información de la base de datos.

Esto garantiza el porcentaje de espacio libre en memoria caché. En los puntos de sincronización escribe las páginas de información de la base de datos almacenada en la memoria caché en el log físico.

El caché daemon ayuda a incrementar el rendimiento (performance) así:

— asegurando que existe una página disponible cuando un proceso de usuario la necesite.

— permite al usuario disponer de páginas leídas para evitar accesos a disco.

— limpia la memoria de procesos muertos y libera sus áreas utilizadas.

— provee estadísticas para su afinamiento.

Se puede, además, configurar UNIFY 2000 para manejo eficiente de la base de datos física y del log físico. Las siguientes operaciones son características del caché daemon:

— El mínimo porcentaje de páginas libres para la administración de caché.

— El óptimo porcentaje de páginas libres para la administración de caché.

— El número máximo de buffers caché.

* **Log daemon:** Participa en la administración de transacciones. Cuando éstas terminan (commit), UNIFY 2000 escribe (flushes) la transacción en el archivo de transacciones y en cada punto de sincronismo se escriben las transacciones completas o incompletas en este archivo a través del file daemon.

Adicionalmente, en un punto de sincronismo el log daemon transfiere los registros del archivo de transacciones al archivo de jornada.

* **Cleanup daemon (cleanup dead processes):** Administra los procesos conectados a la base de datos y garantiza suficiente memoria disponible para los usuarios de la base de datos, monitoreando el porcentaje de memoria compartida que está siendo usada y limpiándola cuando se requiere más espacio. Es decir, elimina procesos invertidos y revisa las transacciones que estos estaban generando.

6. ADMINISTRACION DE RECURSOS DEL SISTEMA OPERACIONAL

6.1. Almacenamiento físico

El almacenamiento físico del UNIFY 2000 se hace sobre un dispositivo o archivo dividido en bloques. Un bloque es una unidad atómica del acceso al disco dependiente del sistema operacional. El tamaño del bloque es hardware-dependiente.

A menudo un solo archivo lógico no es suficiente para retener todos los datos de una base de datos. Para que la base de datos pueda llegar a ser grande, UNIFY 2000 permite que se distribuya en volúmenes. Un volumen es un área contigua de espacio en disco en un dispositivo. Se pueden tener todos

los volúmenes que se necesiten, el número de volúmenes es limitado sólo por su capacidad de hardware.

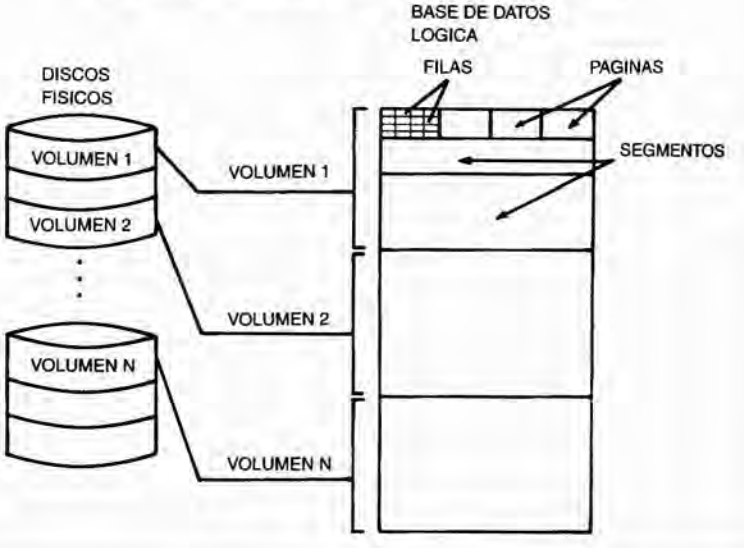
Al adicionar un segmento, si el nuevo segmento no puede ajustarse en el espacio permaneciendo en el actual volumen, UNIFY 2000 aloja el espacio del nuevo segmento en un volumen diferente. El número de segmentos en un volumen es limitado sólo por su hardware.

Un segmento contiene un número entero de páginas. Una página es la unidad de almacenamiento básica para UNIFY 2000. Una página no puede abarcar segmentos y tiene el mismo tamaño que un bloque del file system.

Una fila es la unidad de almacenamiento lógico básica. Una fila puede abarcar una página, pero no segmentos. Una tabla puede contener hasta dos billones de filas. El diagrama de la Figura 14 compara el almacenamiento físico y lógico de la base de datos.

Debido a que muchas aplicaciones deben correrse en diferentes sistemas operativos y sistemas de archivos nativos, la administración de archivos de UNIFY 2000 opera independientemente del sistema de archivos del sistema operativo huésped.

Figura 14
Almacenamiento de datos: físico versus lógico



La administración de archivos es extremadamente flexible porque se puede configurar dinámicamente, y almacenar la base de datos en varios tipos de archivos.

Se puede almacenar una base de datos UNIFY 2000 en tres tipos de archivos:

— Archivos regulares.
— Archivos contiguos.
— Dispositivos crudos.

Un *archivo regular* es un archivo ordinario del sistema operativo. Dinámica-

mente alojado, el archivo es referenciado, los bloques del archivo regular son localizados randómicamente alrededor del disco. El archivo regular es el tipo de archivo más fácil de usar, pero también tiene el peor rendimiento de los tres tipos de archivos.

Un *archivo contiguo* es un archivo de longitud fija, prealojado, que tiene bloques contiguamente localizados en el disco. El archivo contiguo es difícil de usar, pero su rendimiento es mejor que el del archivo regular.

Un *dispositivo crudo* es un archivo de longitud fija, prealojado, que existe como una partición en el disco. El dispositivo crudo es el tipo de archivo más difícil de usar, pero su rendimiento es el mejor.

En sistemas que no soportan archivos contiguos, UNIFY 2000 los emula. Esto implica que UNIFY 2000 puede prealojar cada archivo en su totalidad en la creación.

6.2. Escogiendo un tipo de archivo

Antes de escoger un tipo de archivo, se deben entender las ventajas y limitaciones de los tipos de archivos porque cuando es creada la base de datos, ya no se puede cambiar el tipo de archivo. Consideraciones importantes para el tamaño del archivo, velocidad y facilidad de uso:

*** Consideraciones del archivo regular:**

— No se necesita crear un archivo regular antes de usarlo porque UNIFY 2000 crea el archivo cuando es creada la base de datos.

— El tamaño de un archivo regular puede cambiarse dinámicamente, no restringiendo el crecimiento mientras el sistema de archivos corre fuera del espacio del disco.

— En sistemas operativos basados en Unix, un archivo regular es construido de una estructura jerárquica de bloques inodos. Como el archivo crece, el número de niveles inodos se incrementa, el cual a su vez incrementa el número de accesos físicos de E/S requeridos para accesar un bloque específico del archivo.

— Los archivos regulares son los tipos de archivos más lentos, pero los más fáciles de usar.

*** Consideraciones del archivo contiguo:**

— No necesita ser creado antes de usarlo, porque UNIFY 2000 lo crea cuando éste crea la base de datos.

— Un archivo contiguo es siempre prealojado, lo cual significa que se debe estimar cuán grande es el archivo que se espera crear. El tamaño del archivo no se puede cambiar. Para expandir un archivo contiguo se debe copiar el archivo original en un nuevo archivo más grande.

— Son más seguros que los archivos regulares.

— Pueden requerir de grandes cantidades de espacio contiguo en disco en tiempo de ejecución.

*** Consideraciones de los dispositivos crudos:**

— Se debe crear un archivo crudo antes de usarlo.

— Un dispositivo debe ser creado con un tamaño fijo, lo cual significa que se debe estimar cuán grande es el archivo que se espera crear. El tamaño del archivo no se puede cambiar. Para expandir un archivo crudo, se debe copiar el archivo original en un archivo nuevo más grande.

— Son los tipos de archivos más seguros, aunque su uso es el más difícil.

6.3. Volúmenes de la base de datos y sistemas de archivos

Se pueden usar volúmenes de disco para almacenar grandes aplicaciones de la base de datos, las cuales no pueden ser contenidas totalmente en un simple

archivo del sistema operativo. Ya que las bases de datos son a menudo grandes, UNIFY 2000 permite crear una base de datos multi-volumen.

UNIFY 2000 enumera volúmenes secuencialmente desde el número 1. La longitud de un volumen puede ser fija o variable.

Un path name del volumen es el path name del ínodo que identifica el dispositivo. Si el volumen no está en el volumen raiz, éste puede residir en cualquier lugar.

6.3.1. El volumen raiz

El volumen raiz es el primer volumen en un conjunto de volúmenes. Una base de datos tiene al menos un volumen raíz. No se puede borrar un volumen raíz, exceptuando cuando se borra toda la base de datos.

6.3.2. Creando volúmenes

Si se desea almacenar la base de datos en un archivo regular o contiguo, no se crea el volumen antes de haber creado la base de datos. UNIFY 2000 crea el volumen raíz cuando se crea la base de datos.

Cuando se está usando un archivo contiguo, sin embargo, se debe asegurar suficiente espacio contiguo en el disco para almacenar el volumen. La creación de la base de datos falla si no hay suficiente espacio en el disco para el archivo.

Si se desea almacenar la base de datos en el archivo crudo, se debe crear el volumen antes de haber creado la base de datos.

6.3.3. Borrando volúmenes

Un volumen debe ser vaciado antes que pueda ser borrado. Cuando se borra un volumen de un archivo multi-volumen de un dispositivo crudo, UNIFY 2000 no borra inmediatamente el archivo físico del sistema operativo que contiene el volumen. El archivo será automáticamente borrado (si es un archivo regular) cuando ocurra el próximo punto sincrónico.

6.4. Memoria compartida

La administración de transacciones y de archivos usan memoria compartida para el manejo de comunicación entre los componentes de software de la base de datos. La memoria compartida permite accesar a múltiples usuarios la misma base de datos y compartir la misma área de la memoria.

Cuando el primer usuario accesa la base de datos, UNIFY 2000 lee una página de información en el buffer caché en la memoria compartida. Si otro usuario desea acceder la misma página, la información está lista en caché. UNIFY 2000 no deja tener acceso de nuevo al disco.

Debido a que la memoria caché está en la memoria compartida, UNIFY 2000 no guarda copias separadas de la información de cada usuario. Esto lo hace más fácil guardando el resto de la base de datos actualizada.

Así mismo, un caché en memoria compartida puede ser más grande que un buffer individual. UNIFY 2000 puede guardar más páginas de datos en el buffer, el cual reduce el número de accesos al disco requeridos para recuperar información de la base de datos para muchos usuarios.

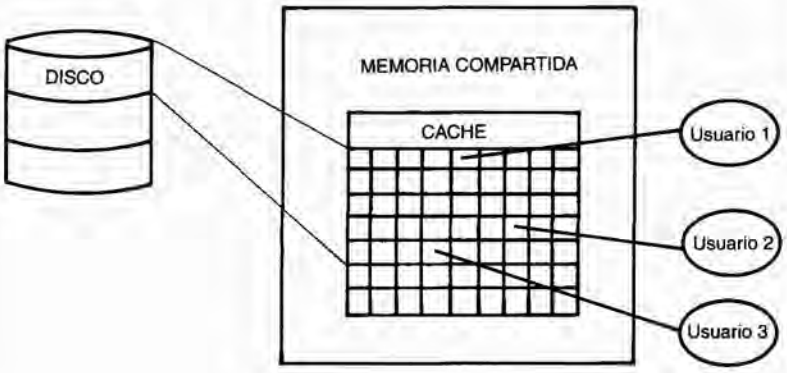
En la Figura 15 se muestra el acceso caché en memoria compartida.

6.4.1. Configurando la memoria compartida

Virtualmente todo en UNIFY 2000 es configurable, incluyendo la memoria compartida. Para ayudar al manejo de la memoria compartida, UNIFY 2000 permite especificar las siguientes características:

* El esquema de direccionamiento físico de la memoria compartida pue-

Figura 15
Acceso caché en la memoria compartida



de ser en orden ascendente o descendente.

- El sistema de direccionamiento de la memoria compartida, el cual UNIFY 2000 llama para asignar los segmentos de memoria compartida en el proceso de direccionamiento de espacio. Se debe especificar un método de búsqueda o el método de direccionamiento por default del sistema operativo.
- El porcentaje tope de reorganización.
- El límite superior o inferior de la búsqueda en bytes.
- El tamaño máximo y mínimo de los segmentos de la memoria compartida.
- El modo de acceso a los segmentos de la memoria compartida, expresado como modos de acceso del sistema operativo para "usuarios", "grupos", y "otros".
- El esquema de bloque de la memoria compartida propio de UNIFY 2000 o el esquema de bloqueo del sistema operativo.
- El path name del directorio de trabajo de la memoria compartida, aplicable solamente si UNIFY 2000 está usando un esquema de bloqueo que provee el sistema operativo.
- La administración de la memoria compartida de la llave de la memoria compartida especifica el segmento de la llave que el administrador de la memoria compartida debe usar para el control de las funciones.
- La llave de la memoria compartida del administrador de autorización, de caché, de la base de datos, del archivo, de bloqueo y de transacción.

7. SQL/A (STRUCTURED QUERY LANGUAGE)

7.1. Definición

Es un lenguaje estructurado de acceso a la base de datos, simple en sintaxis y poderoso en su manejo, ya que permite la creación y manipulación de objetos de la base de datos, así como la consulta y modificación de los datos que en ella han sido creados.

Características:

* DDL: Data Definition Language.
* DML: Data Manipulation Language.
* Modo interactivo y embebido.
* Acceso multi-usuario.

Comandos y ayudas del SQL/A interactivo:

* Una vez que se ha ejecutado el comando desde el shell, abre la base de datos default si ésta ha sido creada, si no da un mensaje indicando que no hay base de datos disponible.
* Provee ayuda al usuario para todos los comandos.
* Proporciona información acerca de los objetos de la base de datos (tablas, columnas, esquemas, etc.).
* Permite corrección de erorres sin tener que reescribir el comando y la escritura de consultas complejas por medio de un editor.
* Permite ejecución de scripts con sentencias SQL. Ej.: start "archivo".
* Permite la ejecución desde el shell.
* Permite salir al shell sin terminar la sesión.
* Permite especificar el número de líneas o eliminar encabezados.
* Permite accesar información acerca de la sesión.
* Permite culminar (commit) y revisar (rollback) transacciones.

7.2. DDL: Data Definition Language

7.2.1. Creando una base de datos

Se puede crear pública o privada.

— Pública: accesible a todos los usuarios del esquema público que se crea por default. Habilita una identificación de usuario llamada 'public' para permitir el acceso mencionado.

— Privada: crea un esquema público por default pero no crea la identificación de un usuario 'public', por lo tanto no es accesible sino para los usuarios a los que explícitamente se les autorice.

Para convertir una base de datos privada en pública se debe habilitar la identificación de usuario 'public'.

7.2.2. Creando esquemas

Un esquema es un nivel lógico de agrupamiento de permisos o averiguaciones sobre acceso a tablas o grupos de usuarios.

Los esquemas que se crean son privados, ya que el esquema público es único y se crea con la base de datos.

Al crear un esquema se pueden crear otros objetos si se especifican: tablas, índices, etc. También se puede asignar privilegios a otros objetos.

7.2.3. Creando una tabla

Para definir una tabla se debe estar localizado en el esquema correspondiente y tener, ya sea, autorización DBA o autorización sobre el esquema.

Al crear una tabla se permite:

* Crear columnas y sus características, si se especifica.
* Crear características de configuración de la tabla, si se especifican.
* Crear sinónimos para la columna, es decir, nombres alternos para un campo de la tabla.

7.2.4. Modificando tablas

Para modificar tablas se debe tener autorización DBA o autorización al esquema.

Alterar una tabla permite:

* Adicionar o borrar la característica 'unique' de una columna.
* Adicionar o borrar sinónimos.
* Cambiar el nombre de la columna.

7.2.5. Borrar una tabla

Antes de borrar una tabla se deberi eliminar las validaciones y referencias

a esta tabla del archivo DIS (Data Integrity Subsystem).

Al borrar una tabla se borran sus datos y los índices, privilegios y vistas asociadas a ella.

7.3. DML: Data Manipulation Language

7.3.1. Definición

* Permite seleccionar información de la base de datos.
* Permite direccionar el resultado a la terminal del usuario (default) o a un archivo.
* Permite ordenar la salida por una o más columnas ascendente o descendentemente.
* Permite agrupar registros por una o más columnas.

7.3.2. Adicionar registros a una tabla

Se pueden adicionar registros a tablas del esquema actual. Si se desea adicionar registros a tablas de otros esquemas es necesario tener privilegios de 'insert' esquema a esquema, privilegios sobre la tabla y sobre todas las columnas que la componen o simplemente tener autoridad DBA.

No es necesario tener datos para todas las columnas de una tabla. Para poder adicionar registros, se pueden insertar valores default o nulos para columnas no especificadas.

Se pueden adicionar registros desde el SQL/A, dándolos explícitamente o leyéndolos de un archivo plano.

7.3.3. Actualización de registros (UPDATE)

Cambia los datos de las columnas en una tabla especificada.

Permite la actualización de uno o más registros dependiendo del criterio de búsqueda.

Los nuevos valores pueden ser expresiones constantes.

Los registros seleccionados son bloqueados de acuerdo con el nivel de bloqueo de la transacción actual.

7.3.4. Eliminación de registros (DELETE)

Eliminar registros de una tabla específica.

Eliminar uno o más registros dependiendo del criterio de búsqueda.

Los registros seleccionados por 'delete' son bloqueados antes de que sean borrados.

8. ADMINISTRACION

8.1. Operación de la base de datos

8.1.1. Activación de la base de datos

El activar la base de datos consta de tres pasos:

* Abrir la base de datos.
* Activar procesos en background (daemons).
* Crear la memoria compartida.

Aunque la base de datos se puede activar con el primer proceso de usuario, es importante que el administrador de la base de datos sea quien la active, ya que es posible que se necesite responder correctamente a eventos que ocurren cuando se activa la base de datos, como por ejemplo: copiar el archivo de jornada a una revisión anterior e iniciarlo en cero.

8.1.2. Desactivación de la base de datos

Para desactivar la base de datos, normalmente se deben seguir tres pasos:

— Terminar todos los procesos de usuario que están accesando la base de datos.
— Desactivar los procesos en background (daemons).
— Limpiar la memoria compartida.

En caso que existan procesos muertos que no permitan desactivar la base de datos, se debe esperar a que estos

procesos sean limpiados por el clean daemon.

Al crear una base de datos, UNIFY 2000 graba la identificación del usuario como creador y habilita un esquema público el cual queda como default.

La diferencia entre crear una base de datos pública y una privada, es que al crearla pública se crea una identificación de usuario llamada 'public', la cual permite que cualquier usuario accese la base de datos. Esto es, cuando un usuario intenta entrar en la base de datos, UNIFY 2000 compara la identificación del usuario contra la tabla de usuario de la base de datos, y si el usuario no se encuentra en ésta, UNIFY 2000 permite el acceso al esquema público por medio de la identificación 'public'.

Las instrucciones de autorización dan a un usuario permisos para accesar una base de datos o esquemas, y para crear esquemas, tablas o vistas.

8.2. Esquemas

Existen dos tipos de esquemas, que son: públicos y privados. El esquema público se crea por default y es único (no importa si la base de datos es pública o privada).

Los esquemas pueden ser creados por el creador de la base de datos o por cualquier usuario que tenga autorización DBA (esta autoridad es asignada por el creador a cualquier usuario). Los usuarios que tienen acceso a un esquema privado tienen acceso al esquema público. Todos los usuarios que tienen acceso a la base de datos pueden accesar el esquema público pero no el privado.

Los privilegios *entre esquemas* permiten a los usuarios de un esquema manipular los objetos de la base de datos pertinentes a otro esquema.

8.2.1. Privilegios esquema a esquema

Son privilegios permitidos de un esquema a otro. Este permite a los usuarios de un segundo esquema manipular objetos de la base de datos que pertenecen a otro esquema. Por ejemplo, una base de datos tiene dos esquemas, ventas y empleados. Ventas, el cual contiene las tablas, precios, órdenes y clientes, es el esquema actual. El esquema de empleados tiene tres tablas, empleados, aumento de sueldo y evaluaciones. Supongamos que se desea permitir a los usuarios de ventas leer la información de la tabla de empleados, pero no la de las otras dos tablas del esquema de empleados. Se pueden conceder privilegios de empleados a ventas, y los usuarios de ventas pueden leer las columnas de nombre, número, y departamento en la tabla de empleados.

Cuando se conceden privilegios esquema a esquema a un segundo esquema, se pueden enlazar los privilegios para operaciones específicas sobre objetos específicos de la base de datos. Esto significa que usuarios del segundo esquema pueden ejecutar sólo operaciones específicas sobre el primer esquema, o sobre una tabla o columna en el primer esquema.

Cuando se tienen privilegios en objetos específicos, se permiten operaciones tales como select, insert, update y delete. Se pueden conceder también todos los privilegios del primer esquema al segundo.

8.2.2. Usuarios

Los privilegios de usuario permiten ejecutar diferentes operaciones sobre los objetos de la base de datos.

UNIFY 2000 provee tres niveles de privilegios de usuario:

— Autorización de *acceso* a la base de datos y a esquemas.

— Autorización DBA.

— Autorización de esquemas.

*** Autorización de acceso:**

— Permite dar acceso a diferentes usuarios.

— Este es automático si la base de datos es pública.

— Debe ser explícito si la base de datos es privada.

— Determina si una base de datos es pública o privada.

*** Autorización DBA:**

— Permite autorizar a determinados usuarios para ejecutar cualquier acción sobre objetos de la base de datos.

— Es automática para el creador de la base de datos y no le puede ser revocada.

— Puede ser otorgada por el creador de la base de datos o por cualquier usuario con autorización DBA.

*** Autorización del esquema:**

— Es diferente de la autorización de acceso a esquema.

— Permite creación de objetos de la base de datos (esquemas, tablas, privilegios).

— Permite todos los privilegios sobre los objetos privados.

8.2.3. Esquema Default

Es el esquema donde es localizado el usuario al abrir la base de datos. Puede ser cambiado por un usuario con autorización DBA.

8.2.4. Privilegios de grupo

Si se desean conceder los mismos privilegios a un grupo de usuarios, se le concede a todos la misma autorización de acceso. El grupo puede, entonces, accesar el mismo esquema específico, si se conceden privilegios de otro esquema para que el esquema sea accedido por estos usuarios. Los usuarios pueden ejecutar cualquiera de las operaciones definidas por el esquema.

8.3. Integridad de datos

8.3.1. Transacción y manejo de locks

La transacción se define como una o más operaciones sobre la base de datos que deben ser tratadas como una sola unidad de trabajo.

En el manejo de transacciones, UNIFY 2000 ejecuta las siguientes tareas:

— Inicia, termina y valida transacciones.

— Mantiene la consistencia de la base de datos y la concurrencia de transacciones usando diferentes tipos de bloqueo y manejo de múltiples transacciones.

— Mantiene un lo de transacciones.

— Permite configurar el manejo de transacciones a través de parámetros tales como:

* Porcentaje de espacio en el lo de transacciones.

* Unidad de medida de punto de sincronismo.

* Número de unidades (frecuencia).

Una transacción puede iniciarse:

— al abrir la base de datos.

— al terminar una transacción anterior.

Una transacción termina:

— con la sentencia Commit.

— con la sentencia Rollback.

— al cerrar la base de datos.

— con la sentencia END del SQL/A.

Después de que una transacción termina, UNIFY 2000 libera los bloques adquiridos por ésta y revisa los efectos de transacción que terminaron mal (aborted transaction). Cuando una transacción termina normalmente, UNIFY 2000 asegura que todos sus efectos serán salvados en la base de datos.

*** Manejo de datos:** El manejador de UNIFY 2000 controla la concurrencia entre transacciones y previene conflictos entre éstos, a través de diferentes tipos de locks y niveles de bloqueo de transacciones.

*** Promoción de locks:** La promoción de locks conserva memoria y provee

mayor rendimiento en requerimientos de bloqueo, posiblemente sacrificando concurrencia. Puede ocurrir de un tipo de bloqueo a otro o de bloqueo de registros a bloqueo de tablas, es decir, que después de bloquear determinado número de registros se bloquea automáticamente la tabla a la que pertenecen.

* **Liberación de locks:** Al terminar la transacción.

Cuando ocurre una promoción de locks, (controlado por UNIFY 2000).

8.3.2. Logging físico y lógico

El proceso de logging consiste en almacenar operaciones físicas o lógicas (transacciones) ejecutadas sobre la base de datos, a fin de recuperar la consistencia de la misma después de alguna falla. Si no se tiene logging activo, las actualizaciones se efectúan directamente de la memoria compartida a la base de datos, por tanto, si el sistema falla y la memoria es borrada, la base de datos quedará en un estado inconsistente.

Los registros actualizados son escritos en la base de datos cuando se pasa la memoria caché (flush). No hay control de cuándo o en qué orden serán grabados (no hay sincronización).

* **Logging físico:** Consiste en grabar las páginas actualizadas de la memoria caché al log físico en cada punto de sincronización y posteriormente a la base de datos.

Características:

— Facilita la recuperación después de fallas que causan corrupción a la base de datos.

— En caso de una falla del sistema, cuando se está escribiendo en la base de datos, el log físico es usado para restaurarlo.

* **Logging lógico:** Consiste en mantener un archivo con la información acerca de cada transacción realizada en la base de datos para asegurar un estado consistente lógico después de alguna falla.

Características:

— Después de una falla se usa para rehacer transacciones terminadas después del último punto de sincronización y para revisar transacciones activas en el momento de la falla.

8.3.3. Fallas y recuperación

Se pueden presentar tres tipos de fallas:

— en programas.

— en el sistema.

— en los medios (daños físicos).

* **Fallas en el programa:** Un programa de usuario puede contener errores que causen que el programa falle o un usuario lo puede interrumpir. Si el logging está activo, UNIFY 2000 leerá el log de transacciones para revisar los efectos de la transacción abortada.

* **Fallas en el sistema:** En caso de caídas del sistema por fallas eléctricas, la información en memoria se pierde y la base de datos puede quedar en un estado inconsistente. UNIFY 2000 provee un manejador de recuperación integrado (Integrated Recovery Manager, IRMA) que se encarga de restaurar la base de datos a un estado consistente.

IRMA verifica el log físico, si éste no está vacío asume que el sistema ha fallado, y dependiendo de la información de log físico, IRMA la aplica a la base de datos (sincroniza) o desecha la información que contiene (no confiable). Una vez termina la recuperación física, si el logging está activo, IRMA procesa el log de transacciones para revisar las no terminadas y rehace las que terminaron normalmente.

* **Fallas en los medios:** Estas fallas son daños físicos. Cuando un dispositivo se daña, la información que contiene se pierde permanentemente. Por ejem-

plo, cuando se estrellan las cabezas de un disco.

Para poder recuperar la información de la base de datos se deben tener activos:

— el log de transacciones.

— el log físico.

— el log de jornada (journal).

Es posible recuperar a partir del backup y el log de jornada.

*** Actividades del logging:**

— Commit: escribe las transacciones terminadas en el log de transacciones.

— Syncpoint:

— Notifica a UNIFY 2000 que la sincronización comenzó.

— Suspende actualizaciones.

— Copia todas las operaciones del buffer de transacción al log de transacciones.

— Copia las páginas actualizadas de la memoria caché al log físico.

— Activa actualizaciones.

— Transfiere los cambios del log físico a la base de datos.

— Copia las transacciones completas al journal.

8.3.4. Backup de la base de datos

Por medio del programa **budb**, UNIFY 2000 permite hacer backup de la base de datos mientras los usuarios están trabajando en ella. El programa **budb** asegura la consistencia de la base de datos a través del backup, ya que al iniciarse, da a UNIFY 2000 dos instrucciones:

* Sincroniza la base de datos (fuerza un punto de sincronismo).

* Suspende los puntos de sincronismo hasta que el backup lo termine.

Mientras se hace el backup, todas las actualizaciones se hacen en el log físico, y en caso de que el log físico o el log lógico llegue a su límite mientras se hace el backup, UNIFY 2000 suspende las *actualizaciones hasta que termine.*

Cuando se corre el programa desde el shell, él envía un mensaje al operador para que desmonte el journal (si está en cinta) y monte la cinta del backup. En seguida el budb hace un backup completo de los volúmenes de la base de datos, b-tree, *DIS y los archivos de* datos tipo texto y binarios. Al terminar los archivos de jornada, hay que comenzar a actualizar el backup.

*** Restaurando la base de datos:** Una vez se tiene el backup de la base de datos, es posible recuperar el estado consistente de ésta después de una falla.

El programa de restauración sigue los siguientes pasos:

— Solicita al operador montar el primer volumen del backup (así con todos).

— *Lee al backup y deja la base de da*tos en un estado consistente.

— Solicita al operador montar el archivo de jornada a la base de datos (el primero después de haber hecho el backup).

— Aplica las transacciones de jornada a la base de datos.

— Aplica las transacciones terminadas del log de transacción después del último punto de sincronización antes de la falla.

8.3.5. Dis, Data, Integrity Subsystem

Esta herramienta permite mantener la integridad de los datos por medio de un archivo ASCII que contiene secciones de tablas y columnas, a fin de validar y asignar valores por defecto a columnas y chequear referencias entre tablas.

9. METODOS DE ACCESO

9.1. Hash

Permite acceso a una tabla por medio del valor exacto de alguna columna

o combinación de columna. El hash es un método de acceso que aplica un algoritmo al valor de una columna para determinar dónde será localizado el registro en tabla.

Características:

— Se puede definir hash a cualquier columna o columnas de una tabla que cumpla con la característica de ser única.

— Se debe definir explícitamente.

— No se puede definir hash en columnas texto o binario.

— Cada registro en la tabla tiene uno correspondiente en la tabla de hash.

— UNIFY 2000 automáticamente incrementa o decrementa el espacio localizado para una tabla hash, sin requerir que el usuario reorganice la base de datos.

— Provee mayor velocidad de acceso por llave completa.

— No devuelve los registros en orden.

9.2. Links

Permite relacionar dos tablas por medio de columnas comunes, una de las tablas (la relacionada) es la tabla padre y la otra tabla es la tabla hija. La tabla hija contiene una o más columnas que referencian una columna o columnas "únicas" en la tabla padre.

Los links aseguran la integridad referencial entre dos tablas ya que no permiten que un registro padre sea eliminado mientras existan referencias a él (registro hijos).

9.3. B-Tree (Arboles binarios)

Permite un rápido acceso por rangos o valores incompletos, a través de una técnica modificada de árboles binarios.

Dependiendo de la longitud de las columnas indexadas el b-tree usa un determinado número de búsquedas para hacer más efectivo el acceso. Es decir, no se limita a la búsqueda binaria (divisiones de dos en dos).

Características:

— Balanceo dinámico del árbol.

— Reorganización dinámica.

— Trae los registros clasificados.

— Uso de archivos externos a la base de datos.

— Requiere más espacio en disco que los demás métodos de acceso.

9.4. Direct key (Llave directa)

El valor de la llave es manejado directamente a una dirección de la base de datos.

Características:

— Se especifica en el momento de cerrarse la tabla.

— Debe ser de tipo numérico (mínimo cinco).

— Los valores de la llave pueden ser especificados por el usuario o calculados por UNIFY 2000.

— Provee el mejor tiempo de respuesta en acceso e inserciones, porque la localización de los registros está directamente relacionada al valor de la llave.

— El tamaño de la tabla no afecta el tiempo de acceso.

— Devuelve los datos en orden ascendente.

— Los valores de las llaves de registros borrados pueden ser reutilizados por nuevos registros a los que no se les especificó un valor para la llave en el momento de insertarlos.

9.5. Secuencial

Busca en todos los registros de la tabla, comenzando con el primero, de uno en uno hasta el final.

Características:

— No requiere espacio adicional en el disco.

— Si se requieren leer todos los registros de una tabla y el orden no impor-

ta, éste es el método más rápido para hacerlo.

— Para buscar determinados registros debe leerlos todos y, por tanto, es muy lento.

— Para leerlos en determinado orden se deben clasificar.

— Este es el método de acceso default.

CONCLUSIONES

Con este trabajo hemos comprendido la importancia de un sistema de base de datos en las organizaciones. Los sistemas de bases de datos sirven para la manipulación eficiente de la información útil para la administración y toma de decisiones de todo tipo en las empresas. Esto les va a permitir un mejor crecimiento y les permitirá tener una ventaja competitiva con respecto a las demás empresas.

De los tres enfoques de las bases de datos presentados en este trabajo, el principal es el enfoque relacional, que es el que ha venido dominando el mundo de las bases de datos. Sin embargo, estos no son los únicos, pues hoy en día existen otras tendencias como las bases de datos orientadas a objetos, que están en desarrollo y vislumbran un muy positivo panorama.

En particular, hemos aprendido un poco de UNIFY 2000, que presenta todas las ventajas del enfoque relacional y con características que han sido presentadas en este trabajo.

Así mismo vale la pena resaltar como conclusión de este trabajo no sólo su contenido sino también todo lo que tiene que ver con su correspondiente exposición. Ha sido muy importante para nosotros la gran experiencia vivida al enfrentar un auditorio y al aprender a manejarlo, lo mismo que el utilizar las ayudas visuales para la mejor comprensión del tema por parte de los asistentes.

Esto nos servirá para desenvolvernos de una mejor manera en nuestra futura vida profesional y además para enriquecernos como personas. En todo esto, la ayuda que nos prestaron nuestros profesores de Sistemas Operacionales I ha sido de gran provecho.

BIBLIOGRAFIA

Data and Knowledge engineering. Volumen 5, Nº 6, diciembre de 1993.

Data and Knowledge engineering. Volumen 11, Nº 3, diciembre de 1993.

Ozsu, M. Tamer, Patrick Valduriez. *Principles of distributed database system.* Prentice Hall, 1991.

Deitel, Harvey M. Introducción a los sistemas operativos. México, Addison-Wesley Iberoamericana, 1987.

Unify 2000, Manual de referencia. Unify Corporation, USA, 1989.

Manual, curso de UNIFY 2000 hecho por la C-NIX.

Date, C.J. Introducción a los sistemas de base de datos. México, Addison-Wesley Iberoamericana, 1986.

"*For Developers Only:* How to choose your Database Management System". Data Based Advisor, may 1993.