UNIDAD 1. INTRODUCCIÓN

El apunte de la presente unidad está basado en el libro "Fundamentos de Bases de Datos" de Abraham Silberschatz, Henry F. Korth y S. Sudarshan. A los fines de los contenidos de las materias de las tecnicaturas, el apunte debe complementarse con los ejemplos prácticos presentes en las transparencia y ejemplos dados en teoría.

La gestión de bases de datos ha evolucionado desde una aplicación informática especializada hasta una parte esencial de un entorno informático moderno. Como tal el conocimiento acerca de sistemas de bases de datos se ha convertido en una parte esencial en la formación en informática.

Un sistema de gestión de bases de Datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información acerca de una empresa particular. El primer objetivo de un SGBD es proporcionar un entorno que sea tanto práctico como eficiente de usar en la recuperación y el almacenamiento de la información de la base de datos.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. En suma, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos.

La importancia de la información en la mayoría de las organizaciones –que determina el valor de las bases de datos- ha conducido el desarrollo de una gran cantidad de conceptos y técnicas para la gestión eficiente de los datos. En esta unidad se presenta una breve introducción a los principios de los sistemas de bases de datos.

1. SISTEMAS DE ARCHIVOS

Considérese parte de una empresa de cajas de ahorro que mantiene información acerca de todos los consumidores y cuentas de ahorro. Una manera de mantener la información en una computadora es almacenarla en archivos del sistema. Para permitir a los usuarios manipular la información, el sistema tiene un número de programas de aplicación que manipula los archivos, incluyendo:

- Un programa para efectuar cargos o abonos en una cuenta.
- Un programa para añadir una cuenta nueva.
- Un programa para calcular el saldo de una cuenta.
- Un programa para generar las operaciones mensuales.

Estos programas de aplicación se han escrito por programadores de sistemas en respuesta a las necesidades de la organización bancaria.

Si las necesidades se incrementan, se añaden nuevos programas de aplicación al sistema. Por ejemplo, supóngase que las regulaciones de un nuevo gobierno permiten a las cajas de ahorro ofrecer cuentas corrientes. Como resultado, se crean nuevos archivos permanentes que contengan información acerca de todas las cuentas corrientes mantenidas por el banco y puede ser necesario escribir nuevos programas de aplicación para tratar situaciones que no existían en las cuentas de ahorro, tales como manejar descubiertos. Así, sobre la marcha, se añaden más archivos y programas de aplicación al sistema.

El sistema de procesamiento de archivos típico que se acaba de describir se mantiene mediante un sistema operativo convencional. Los registros permanentes son almacenados en varios archivos y se escriben diferentes programas de aplicación para extraer registros y para añadir registros a los archivos adecuados. Antes de la llegada de los SGBD, las organizaciones normalmente han almacenado la información usando tales sistemas.

Mantener información de la organización en un sistema de procesamiento de archivos tiene una serie de inconvenientes importantes:

- **Redundancia e inconsistencia de datos**: Debido a que los archivos y programas de aplicación son creados por diferentes programadores en un largo período de tiempo, los diversos archivos tienen probablemente diferentes formatos y los programas pueden estar escritos en diferentes lenguajes. Más aún, la misma información puede estar duplicada en diferentes lugares (archivos). Por ejemplo, la dirección y número de teléfono de un cliente particular puede aparecer en un archivo que contenga registros de cuentas de ahorro y en un archivo que contenga registros de cuenta corriente. Esta redundancia conduce a un almacenamiento y coste de acceso más altos. Además, puede conducir a inconsistencia de datos; es decir, las diversas copias de los mismos datos pueden no coincidir. Por ejemplo, un cambio en la dirección del cliente puede estar reflejado en los registros de cuentas de ahorro pero no estarlo en el resto del sistema.

- **Dificultad en el acceso a los datos**: Supongamos que uno de los empleados del banco necesita averiguar los nombres de todos los clientes que viven en el distrito postal 28733 de la ciudad. El empleado pide al departamento del procesamiento de datos que genere dicha lista. Debido a que esta petición no fue prevista cuando el sistema original fue diseñado, no hay un programa de aplicación para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de todos los clientes. El empleado del banco tiene ahora dos opciones: bien obtener la lista de todos los clientes y obtener la información que necesita manualmente, o bien pedir al departamento de procesamiento de datos que haga que un programador de sistemas escriba el programa de aplicación necesario. Ambas alternativas son obviamente insatisfactorias. Supongamos que se escribe tal programa y que, varios días más tarde, el mismo empleado necesita arreglar esa lista para incluir solo aquellos clientes que tienen una cuenta con saldo de 100.000$ o más. Como suponemos, un programa para generar tal lista no existe. De nuevo, el empleado tiene que elegir dos opciones, ninguna de las cuales es satisfactoria. La cuestión aquí es que el entorno de procesamiento de archivos convencional no permite que los datos necesarios sean obtenidos en forma práctica y

eficiente. Se deben desarrollar sistemas de recuperación de datos más interesantes para un uso general.

- **Aislamiento de datos**: Debido a que los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevo programas de aplicación para recuperar los datos apropiados.

- **Problemas de integridad**: Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de restricciones de integridad. Por ejemplo, el saldo de una cuenta bancaria no puede ser más bajo de una cantidad predeterminada (por ejemplo 10$). Los desarrolladores hacen cumplir estas restricciones en el sistema añadiendo el código apropiado en diversos programas de aplicación. Sin embargo, cuando se añaden nuevas restricciones, es difícil cambiar los programas para hacer que se cumplan. El problema es complicado cuando las restricciones implican diferentes elementos de datos de diferentes archivos.

- **Problemas de atomicidad**: Un sistema de una computadora, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallo. En muchas aplicaciones es crucial asegurar que una vez que un fallo ha ocurrido y se ha detectado, los datos se restauran al estado de consistencia que existía antes del fallo. Consideremos un programa para transferir 500$ desde la cuenta A a la B. Si ocurre un fallo del sistema durante la ejecución del programa, es posible que los 500$ fueran eliminados en la cuenta A pero no abonadas en la cuenta B, resultando un estado de la base de datos inconsistente. Claramente, es esencial para la consistencia de la base de datos que ambos, el crédito y el débito, tengan lugar, o que ninguno tenga lugar. Es decir, la transferencia de fondos debe ser atómica –debe ocurrir por completo o no ocurrir en absoluto-. Es difícil asegurar esta propiedad en un sistema de procesamiento de archivos convencional.

- **Anomalías en el acceso concurrente**: Conforme se ha ido mejorando el conjunto de ejecución de los sistemas y ha sido posible una respuesta en tiempo más rápida, muchos sistemas han ido permitiendo a múltiples usuarios actualizar los datos simultáneamente. En tales sistemas un entorno de interacción de actualizaciones concurrentes puede dar lugar a datos inconsistentes. Considere una cuenta bancaria A, que contiene 1.000$. Si dos clientes retiran fondos (por ejemplo, 100$ y 300$ respectivamente) de la cuenta A en aproximadamente el mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto (o inconsistente). Supongamos que los programas se ejecutan para cada retirada y escriben el resultado después. Si los dos programas funcionan concurrentemente, pueden leer ambos el valor de 1.000$ y escribir después, 900$ y 700$, respectivamente. Dependiendo de cual escriba el último valor, la cuenta puede contener bien 900$ o 700$, en lugar del valor correcto 600$. Para protegerse contra esta posibilidad, el sistema debe mantener alguna forma de supervisión. Sin embargo, ya que se puede acceder a los datos desde muchos programas de aplicación diferentes que no han sido previamente coordinados, la supervisión es difícil de proporcionar.

- **Problemas de seguridad**: No todos los usuarios de un sistema de bases de datos deberían poder acceder a todos los datos. Por ejemplo, en un sistema bancario, el personal de

nóminas necesita ver sólo esa parte de la base de datos que tiene información acerca de varios empleados del banco. No necesitan acceder a la información acerca de las cuentas corrientes. Como los programas de aplicación se añaden al sistema de una forma ad hoc, es difícil garantizar tales restricciones de seguridad.

Estas dificultades, entre otras, han dado comienzo al desarrollo de SGBD.

2. VISION DE LOS DATOS

El propósito principal de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantiene los datos.

2.1 Abstracción de los Datos

Para que el sistema sea útil, debe recuperar los datos eficientemente. Esta preocupación ha conducido al diseño de estructuras de datos complejas para la representación de los datos en la base de datos. Como muchos usuarios de sistemas de bases de datos no están familiarizados con computadoras, los desarrolladores esconden la complejidad a los usuarios a través de varios niveles de abstracción (ver Figura 1) para simplificar la interacción de los usuarios con el sistema:

- **Nivel físico:** El nivel más bajo de abstracción describe cómo se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas de bajo nivel.
- **Nivel lógico:** El siguiente nivel más alto de abstracción describe qué datos se almacenan en la base de datos y que relaciones existen entre esos datos. La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples. Aunque la implementación de estructuras simples en el nivel lógico puede involucrar estructuras complejas del nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad. Los administradores de base de datos, que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.
- **Nivel de Vistas:** El nivel más alto de abstracción describe sólo parte de la base de datos completa. A pesar del uso de estructuras más simples en el nivel lógico, queda algo de complejidad, debido al gran tamaño de la base de datos. A muchos usuarios del sistema de base de datos no les preocupará toda esta información. En su lugar, tales usuarios necesitan acceder solo a una parte de la base de datos. Para que su interacción con el sistema se simplifique, se define la abstracción del nivel de vistas. El sistema puede proporcionar muchas vistas para la misma base de datos. Por ejemplo, los cajeros de un banco ven únicamente la parte de la base de datos que tiene información de cuentas de clientes, no pueden acceder a la información referente a los sueldos de los empleados.

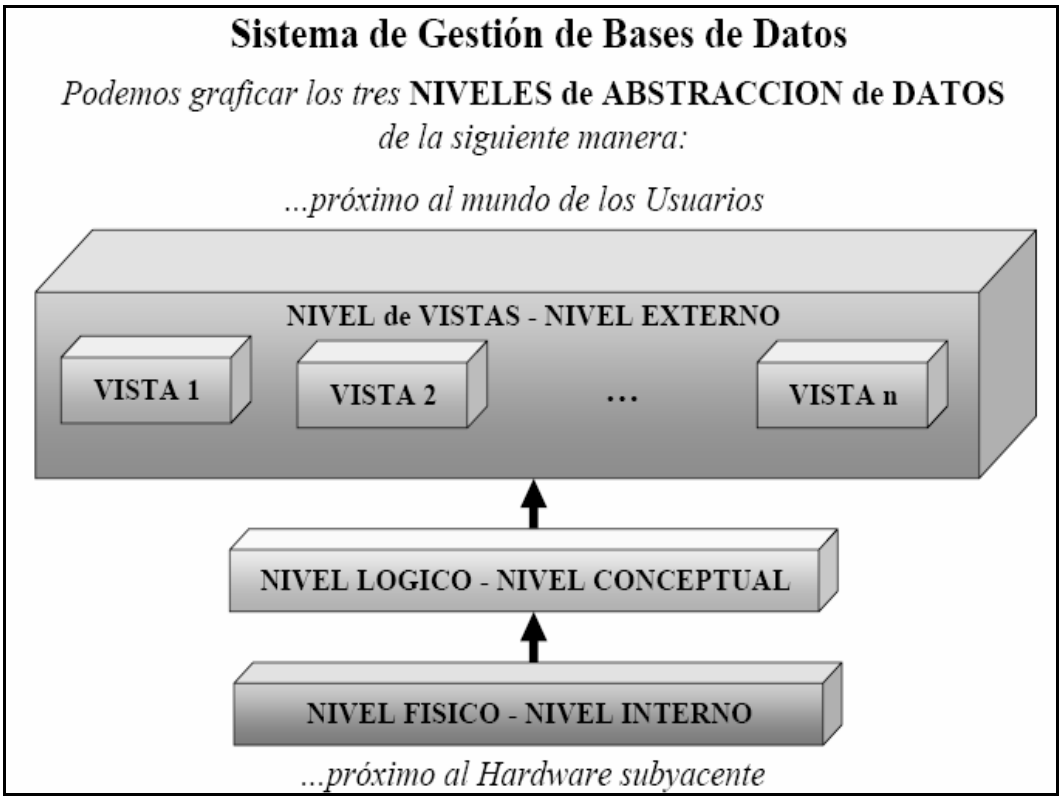

Figura 1: Niveles de Abstracción

2.2. Ejemplares y Esquemas

Las bases de datos van cambiando a lo largo del tiempo conforme la información se inserta y borra. La colección de información almacenada en la base de datos en un momento particular se llama *instancia* de la base de datos. El diseño completo de la base de datos se llama esquema de la base de datos. Los esquemas son raramente modificados, si es que lo son alguna vez.

2.3. Independencia de Datos

La capacidad para modificar una definición de esquema en un nivel sin que afecte a una definición de esquema en el siguiente nivel de abstracción se llama independencia de datos. Hay dos niveles de independencia de datos:

- **Independencia física de datos.** Es la capacidad para modificar el esquema físico sin provocar que los programas de aplicación tengan que reescribirse. Las modificaciones en el nivel físico son ocasionalmente necesarias para mejorar el funcionamiento, por ejemplo aumentar la performance del sistema gestor.

- **Independencia lógica de datos.** Es la capacidad para modificar el esquema lógico sin causar que los programas de aplicación tengan que reescribirse. Las modificaciones en el nivel lógico son necesarias siempre que la estructura lógica de la base de datos se altere (por ejemplo cuando se añaden a un sistema bancario cuentas del mercado de dinero).

La independencia de datos lógica es más difícil de proporcionar que la independencia física, ya que los programas de aplicación son fuertemente dependientes de la estructura lógica de los datos a los que ellos acceden.

3. MODELOS DE DATOS

La parte esencial de la estructura de base de datos es el modelo de datos.: una colección de herramientas conceptuales para describir los datos, las relaciones entre datos, la semántica de los datos y las restricciones de integridad. Los diferentes modelos de datos que se han propuesto se clasifican en dos grupos principales: modelos lógicos basados en objetos, y modelos lógicos basados en registros.

3.1 Modelos lógicos basados en objetos

Los modelos lógicos basados en objetos se utilizan para describir datos en los niveles lógicos y de vistas. Se caracterizan por el hecho de que proporcionan capacidades estructurales muy flexibles y permiten que las restricciones sean especificadas explícitamente. Hay modelos muy distintos que pertenecen a esta categoría. Dos de los más conocidos son:

- El modelo entidad relación
- El modelo orientado a objetos

A continuación se dará una breve descripción de ambos modelos.

3.1.1 Modelo entidad Relación

El modelo entidad relación (MER) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados *entidades* y de *relaciones* entre esos objetos. Una entidad es un concepto del mundo real, que es distinguible de otros conceptos. Por ejemplo, cada persona es una entidad, y las cuentas bancarias pueden ser consideradas entidades. Las entidades se describen en una base de datos mediante un conjunto de atributos. Por ejemplo, los atributos nro-cta y saldoCta describen una cuenta en particular de un banco. Una relación es una asociación entre varias entidades. Por ejemplo, una relación *tieneCta* asocia un cliente con cada cuenta que posee el cliente. Este pequeño modelo Entidad Relación se grafica en Figura 2. El conjunto de todas las entidades del mismo tipo y el conjunto de todas las relaciones del mismo tipo se denominan conjunto de entidades y conjunto de relaciones respectivamente.

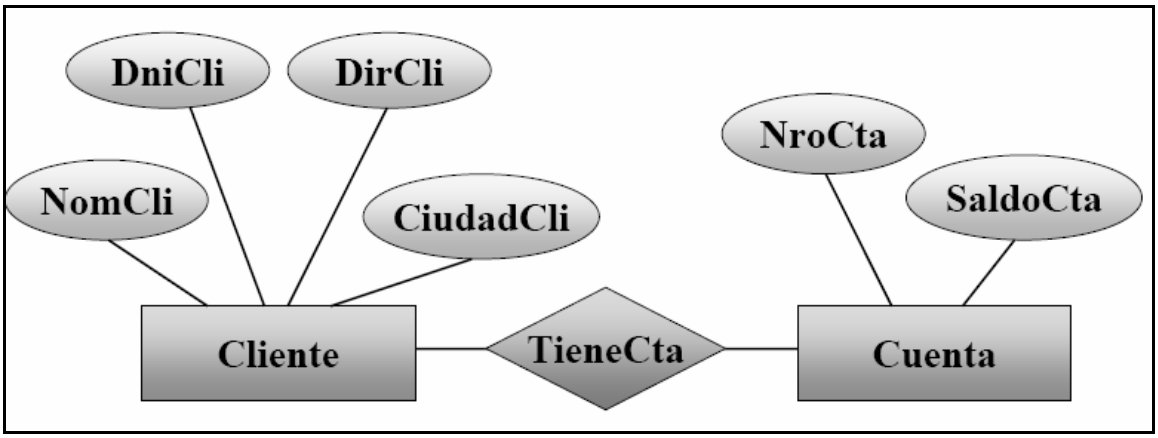

Figura 2: Ejemplo de Modelo Entidad Relación

Además de entidades y relaciones, el modelo Entidad Relación representa ciertas restricciones que los contenidos de la base de datos deben cumplir. Una restricción importante es la cardinalidad de una relación, que expresa el número de ejemplares de una entidad con las que otra entidad se puede asociar a través de una relación.

La totalidad de estructuras lógicas de una base de datos se pueden expresar gráficamente mediante modelos ER, que constan de los siguientes componentes:

- **Rectángulos**, que representan entidades.
- **Elipses**, que presentan atributos.
- **Rombos**, que representan relaciones entre entidades.
- **Líneas**, que unen los atributos con las entidades y las entidades a las relaciones.

Cada componente se etiqueta con la entidad o relación que presenta.

3.1.2 Modelo Orientado a Objetos

Como el modelo ER, el modelo Orientado a Objetos (OO) está basado en una colección de objetos. Un objeto contiene valores almacenados en variables de instancia dentro de ese objeto. Un objeto también contiene fragmentos de código que operan en el objeto. Estos fragmentos de código se llaman métodos.

Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan juntos en *clases*. Una clase se pueden ver como una definición de tipo para los objetos.

La única manera de que un objeto pueda acceder a los datos de otro objeto es mediante la invocación de un método de ese otro objeto. Esta acción se llama envío de mensaje a otro objeto. Así, la interfaz de llamada de los métodos de un objeto define la parte visible (externa) del objeto. La parte interna del objeto, las variables de instancia y código de los métodos, no es visible externamente. El resultado es obtener dos niveles de abstracción de datos.

Para ilustrar el concepto, considérese un objeto que representa una cuenta bancaria. Tal objeto contiene variables de instancia número-cuenta y saldo. Asúmase que el banco ha estado pagando un

6% de interés en todas sus cuentas, pero ahora está cambiando su política para pagar un 5% si el saldo es menor que 5.000$, o un 6% si el saldo es mayor o igual a 5.000$. Para la mayoría de los modelos de datos, hacer este ajuste significaría cambiar el código en uno o más programas de aplicación. Para el modelo orientado a objetos, el único cambio se hace en el método *pago-interés*. La interfaz externa para los objetos permanece igual.

Al contrario que las entidades en el modelo ER, cada objeto tiene su propia identidad única, independientemente de los valores que contenga. Así, dos objetos que contienen los mismos valores son completamente diferentes. La distinción entre objetos individuales se mantiene en el nivel físico a través de la asignación de diferentes identificadores a los objetos.

3.2 Modelos lógicos basados en registros.

Los modelos lógicos basados en registros se usan para describir datos en los niveles lógicos y de vistas. En contrastes con los modelos de datos basados en objetos, se usan tanto para especificar la estructura lógica completa de la base de datos como para proporcionar una descripción de alto nivel de la implementación.

Los modelos basados en registros se llaman así debido a que la base de datos se estructura en registros de formato fijo de diferentes tipos. En cada tipo de registro se define un número fijo de campos o atributos, y cada campo tiene normalmente una longitud fija. El uso de registros de longitud fija simplifica la implementación en el nivel físico de la base de datos.

Los tres modelos basados en registros más ampliamente aceptados son el modelo relacional, el modelo de red, y el modelo jerárquico. El modelo relacional, ha prevalecido sobre los otros dos en los últimos años. A continuación se presenta una breve visión de cada modelo.

3.2.1 Modelo Relacional

En el modelo relacional se usa una colección de tablas para representar tanto los datos como las relaciones entre esos datos. Cada tabla tiene varias columnas, y cada columna tiene un nombre único. En la Figura 3 se presenta un ejemplo de una base de datos relacional consistente en dos tablas: una muestra los clientes de un banco y la otra muestra las cuentas que pertenecen a esos clientes. Esta figura muestra, por ejemplo, que el cliente Scholz, con número de DNI 9024931 vive en la calle San Martin 18 de la ciudad de Roca, y tiene dos cuentas: C-1000/1, con un saldo de 900 pesos y C-3000/1 con un saldo de 5000$. Nótese que los clientes Scholz y Vicario comparten la cuenta con número C-3000/1.

NomCli	DniCli	DirCli	CiudadCli
Scholz	9024931	San Martín 18	Roca
Vicario	14234848	Rivadavia 160	Cipolletti
Zahn	27470988	Las Acacias 51	Cipolletti
Martín	13537253	9 de Julio 754	Roca
Scholz	9024931	San Martín 18	Roca
Castañet	15219742	Las Heras 81	Allen
Giró	24602539	Villegas 1205	Allen
Vicario	14234848	Rivadavia 160	Cipolletti

DniCli	NroCta
9024931	C-1000/1
14234848	C-2000/2
27470988	C-3024/5
13537253	C-1567/1
9024931	C-3000/1
15219742	C-5342/4
24602539	C-9311/0
14234848	C-3000/1

NroCta	SaldoCta
C-1000/1	900
C-2000/2	1000
C-3024/5	525
C-1567/1	850
C-3000/1	5000
C-5342/4	2074
C-9311/0	728

Figura 3: Ejemplo de Modelo Relacional

3.2.2 Modelo de Red

Los datos en el modelo de red se representan mediante colecciones de registros, y las relaciones entre los datos se representan mediante vínculos o enlaces, que se pueden ver como punteros. Los registros en la base de datos se organizan como colecciones de grafos dirigidos. En la Figura 4 se presenta un ejemplo de base de datos en red que usa la misma información que la Figura 3.

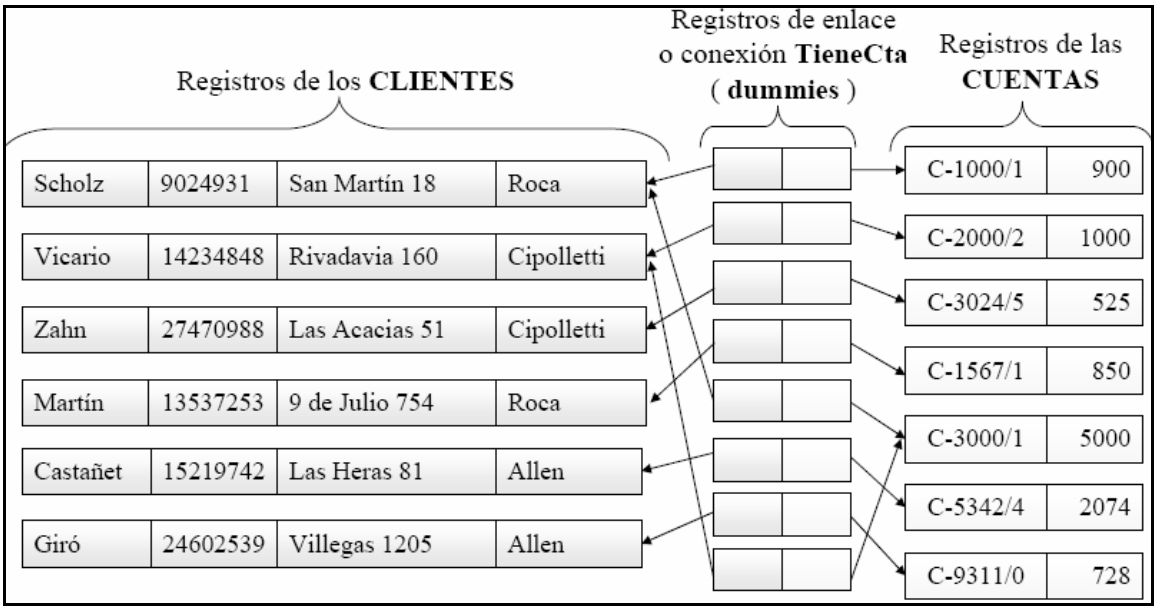

Figura 4: Ejemplo de Modelo de Red.

3.2.3 Modelo Jerárquico.

El modelo jerárquico es similar al modelo de redes, en el sentido en que los datos y las relaciones entre los datos se representan mediante registros y enlaces, respectivamente. Este se diferencia del

modelo de redes en que los registros se organizan como colecciones de árboles en lugar de grafos dirigidos. En la Figura 5 se presenta un ejemplo de base de datos jerárquica con la misma información que la Figura 3.

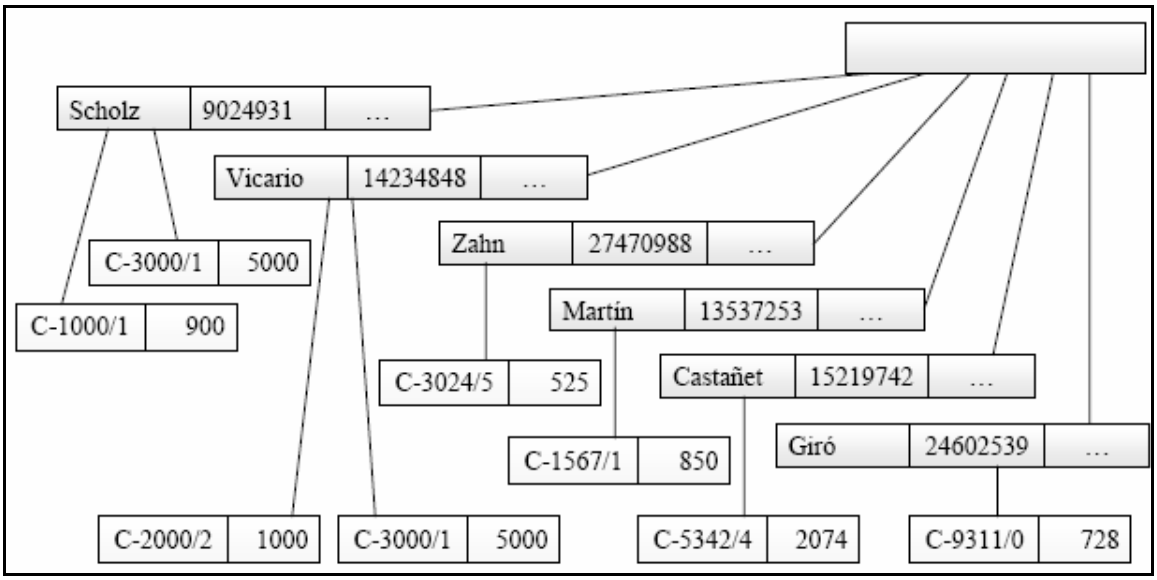

Figura 5: Ejemplo de Modelo Jerárquico.

3.2.4. Diferencias entre los modelos

El modelo Relacional se diferencia de los modelos de redes y jerárquico en que no usa punteros o enlaces. En su lugar, el modelo relacional relaciona los registros mediante los valores que ellos contienen.

4. LENGUAJES DE BASES DE DATOS

Un sistema de base de datos proporciona dos tipos de lenguajes diferentes: uno para especificar el esquema de base de datos y el otro para expresar las consultas y actualizaciones de la base de datos.

4.1 Lenguaje de definición de datos

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado lenguaje de definición de datos (LDD). El resultado de la compilación de las instrucciones en LDD es un conjunto de tablas que se almacenan en un archivo especial llamado diccionario de datos o catálogo de datos.

Un diccionario de datos es un archivo que contiene metadatos; es decir, datos acerca de datos. Este archivo se consulta antes de leer o modificar los datos reales del sistema de base de datos.

La estructura de almacenamiento y los métodos de acceso usados por el sistema de base de datos se especifican mediante un conjunto de definiciones en un tipo especial de LDD llamado un lenguaje de almacenamiento y definición de datos. El resultado de la compilación de estas definiciones es un conjunto de instrucciones para especificar los detalles de implementación de los esquemas de la base de datos. Estos detalles se ocultan a los usuarios.

4.2 Lenguaje de Manipulación de datos

Los niveles de abstracción se aplican no sólo a la definición o estructuración de los datos, sino también a la manipulación de los datos. Por manipulación de datos involucra:

- La recuperación de información almacenada en la base de datos.
- La inserción de información nueva en la base de datos.
- El borrado de información de la base de datos.
- La modificación de información almacenada en la base de datos.

En el nivel físico se deben definir algoritmos que permitan un acceso eficiente a los datos. En los niveles más altos de abstracción se enfatiza la facilidad de uso. El objetivo es proporcionar una interacción humana eficiente con el sistema.

Un lenguaje de manipulación de datos (LMD) es un lenguaje que permite a los usuarios acceder o manipular los datos organizados mediante el modelo de datos apropiado. Hay dos tipos básicamente:

- **LMD procedimentales:** Requieren que el usuario especifique que datos se necesitan y como obtener esos datos.
- **LMD no procedimentales:** Requieren que el usuario especifique que datos se necesitan, sin especificar cómo obtener esos datos.

Los LMD no procedimentales son más fáciles de aprender y usar que los LMD procedimentales. Sin embargo, como el usuario no especifica cómo conseguir los datos, estos lenguajes pueden generar código que no sea tan eficiente como el que generan los lenguajes procedimentales.

Una consulta es una instrucción de solicitud para recuperar información. La parte de un LMD que implica recuperación de información se llama lenguaje de consultas. Aunque técnicamente sea incorrecto, en la práctica es común que se usen como sinónimos los términos *lenguaje de consultas* y *lenguajes de manipulación de datos*.

5. TRANSACCIONES

Varias operaciones sobre la base de datos forman a menudo una única unidad lógica de trabajo. Un ejemplo es una transferencia de fondos, en el que se debita por ejemplo de una cuenta A y se acredita en una cuenta B. Claramente, es esencial que o bien tanto el debito como el crédito tengan

lugar, o bien no ocurra ninguno. Es decir, la transferencia de fondos debe ocurrir por completo o no ocurrir en absoluto. Este requerimiento de todo o nada se llama *atomicidad*. Además, es esencial que la ejecución de la transferencia de fondos preserve la consistencia de la base de datos. Es decir, el valor de la suma del saldo de las cuentas A y B (esto es A+B) se debe preservar. Este requisito de corrección se llama *consistencia*. Finalmente, tras la ejecución correcta de la transferencia de fondos, los nuevos valores de las cuentas A y B deben persistir en la base de datos, a pesar de la posibilidad de fallo del sistema. Este requisito de persistencia se llama *durabilidad*.

Una transacción es una colección de operaciones que se lleva a cabo como una función lógica simple en una aplicación de bases de datos. Cada transacción es una unidad de atomicidad y consistencia. Así, se requiere que las transacciones no violen ningún requisito de consistencia en la base de datos. Es decir, si la base de datos era consistente cuando la transacción comenzó, la base de datos debe ser consistente cuando la transacción termine con éxito.

Es responsabilidad del programador definir las diferentes transacciones apropiadamente, de tal manera que cada una preserve la consistencia de la base de datos. Asegurar las propiedades de atomicidad y durabilidad es responsabilidad del propio sistema de base de datos; específicamente del componente de gestión de transacciones. En ausencia de fallos, toda transacción completada con éxito y atómica se archivará fácilmente. Sin embargo, debido a diversos tipos de fallos, una transacción puede no siempre completar su ejecución con éxito. Si se asegura la propiedad de atomicidad, una transacción que falle no debe tener efecto en el estado de la base de datos. Así, la base de datos se restaura al estado en que estaba antes de que la transacción en cuestión comenzara su ejecución. Es responsabilidad del sistema de bases de datos detectar los fallos del sistema y restaurar la base de datos al estado que existía antes de que ocurriera el fallo.

Finalmente, cuando varias transacciones actualizan la base de datos concurrentemente, la consistencia de los datos puede no ser preservada, incluso aunque cada transacción individualmente sea correcta. Es responsabilidad del gestor *de control de concurrencia* controlar la interacción entre las transacciones concurrentes para asegurar la consistencia de la base de datos.

Los sistemas de bases de datos diseñados para uso sobre pequeñas computadoras personales pueden no tener las características vistas. Por ejemplo, muchos sistemas pequeños imponen la ligadura de permitir el acceso a un único usuario a la base de datos en un instante en el tiempo. Otros dejan las tareas de copia de seguridad y recuperación a los usuarios. Esta instalación permite una gestión de datos más pequeña, con menos requisitos de recursos físicos, especialmente de memoria principal. Aunque tales aproximaciones de bajo coste y con pocas prestaciones son suficientes para bases de datos personales pequeñas, son inadecuadas para satisfacer las necesidades de un desarrollo de media a gran escala.

6. GESTION DE ALMACENAMIENTO

Las bases de datos normalmente requieren una gran cantidad de espacio de almacenamiento. La totalidad de las bases de datos se mide normalmente en términos de megabytes o gigabytes, o para las bases de datos más grandes terabytes. Un gigabyte son 1.000 megabytes (1.000 millones de bytes) y un terabyte es un millón de megabytes (un billón de bytes). Debido a que la memoria

principal de las computadoras no puede almacenar esta gran cantidad de información, la información es almacenada en discos. Los datos son trasladados entre el disco de almacenamiento y la memoria principal cuando se necesita. Como la transferencia de datos a y desde el disco es lenta comparada con la velocidad de la unidad central de procesamiento, es necesario que el sistema de base de datos estructure los datos para minimizar la necesidad de movimiento entre el disco y la memoria principal.

El objetivo de un sistema de base de datos es simplificar y facilitar el acceso a los datos. Las vistas de alto nivel ayudan a conseguir este objetivo. Los usuarios del sistema no deberían cargar innecesariamente con los detalles físicos de implementación del sistema. Sin embargo, un factor principal de la satisfacción del usuario o la carencia de ellas con un sistema de base de datos es el funcionamiento del sistema. Si el tiempo de respuesta para una petición es demasiado largo, el valor del sistema disminuye. El funcionamiento de un sistema depende de la eficiencia con que se hayan usado las estructuras de datos para representar los datos en las bases de datos y la eficiencia con la que sea capaz de operar el sistema con estas estructuras de datos. Como es el caso de los sistemas informáticos, se debe hacer un compromiso no sólo entre espacio y tiempo, sino también entre la eficiencia de un tipo operación y otra.

Un gestor de almacenamiento es un módulo del programa que proporciona la interfaz entre los datos de bajo nivel en la base de datos y los programas de aplicación y consultas suministradas al sistema. El gestor de almacenamiento es responsable de la interacción con la gestión de archivos. Los datos en bruto se almacenan en disco usando un sistema de archivos, que es habitualmente proporcionado mediante un sistema operativo convencional. La gestión de almacenamiento transforma las diferentes instrucciones de LMD a órdenes de un sistema de archivos de bajo nivel. Así, el gestor de almacenamiento es responsable del almacenamiento, recuperación y actualización de los datos en la base de datos.

7. ROLES ASOCIADOS A UN SISTEMA DE BASE DE DADOS

7.1 ADMINISTRADOR DE LA BASE DE DATOS

Una de las principales razones para usar un SGBD es tener un control centralizado tanto de los datos como de los programas que acceden a esos datos. La persona que tiene ese control central sobre el sistema se llama administrador de base de datos (ABD, ó en inglés DBA). Las funciones del ABD incluyen al menos las siguientes:

- Definición del esquema: El ABD crea el esquema original de la base de datos escribiendo un conjunto de definiciones que el compilador del LDD traduce a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.

- Estructuras de almacenamiento y definición del método de acceso. Los ABD crean las estructuras de almacenamiento apropiadas y los métodos de acceso escribiendo un conjunto de definiciones, que son traducidas por el compilador del lenguaje de definición y almacenamiento de datos.

- Esquema y modificación de la organización física. Los programadores llevan a cabo las relativamente escasas modificaciones sobre el esquema de base de datos o la descripción de la organización de almacenamiento físico escribiendo un conjunto de definiciones que son usadas bien por el compilador del LDD o bien por el compilador del lenguaje de definición y almacenamiento de datos para generar las modificaciones en las tablas correspondientes del sistema interno (por ejemplo, el diccionario de datos). Ver Figura 6.

- Concesión de la autorización para el acceso a los datos. La concesión u otorgamiento de diferentes tipos de autorización al administrador de la base de datos determinar que partes de la base de datos pueden acceder los diferentes usuarios. La información de autorización se mantiene en una estructura del sistema especial que el sistema de base de datos consulta cuando se intenta el acceso a los datos en el sistema.

- Especificación de las restricciones de integridad. Los valores de los datos almacenados en la base de datos deben satisfacer ciertas restricciones de integridad. Por ejemplo, quizás el número de horas que un empleado pueda trabajar en una semana no deba exceder de un límite especificado (por ejemplo, 80 horas). Tales restricciones deben ser especificadas explícitamente por el administrador de base de datos. Las restricciones de integridad se mantienen en una estructura especial que el sistema de base de datos consulta cuando tiene lugar una actualización en el sistema.

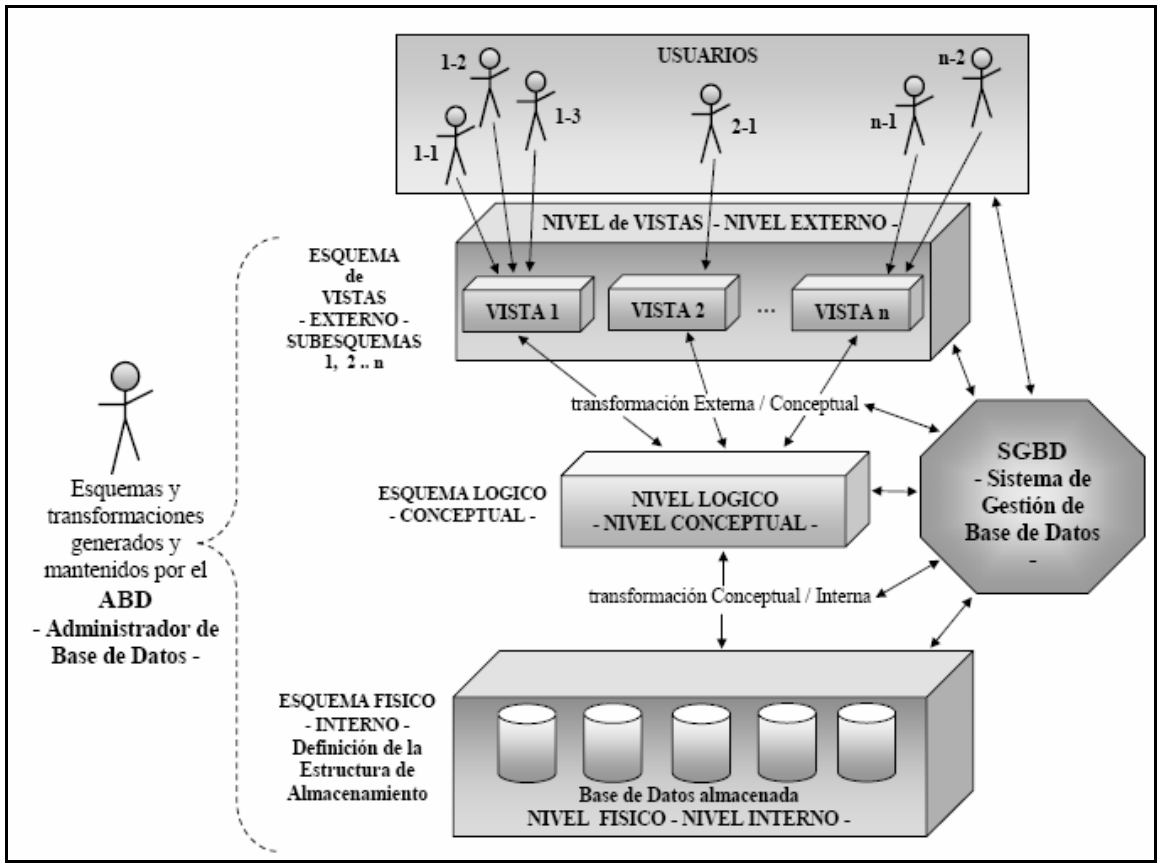

Figura 6: Actividades de Transformación del ABD.

7.2 USUARIOS DE UNA BASE DE DATOS

Un primer objetivo de un sistema de base de datos es proporcionar un entorno para la recuperación de la información y el almacenamiento de nueva información en la base de datos. Hay cuatro tipos diferentes de usuarios de un sistema de base de datos, diferenciados por la forma en que ellos esperan interactuar con el sistema.

- **Programadores de aplicaciones**. Son profesionales informáticos que interactúan con el sistema a través de llamadas del LMD, que están incluidas en un programa escrito en un lenguaje anfitrión (por ejemplo, Pascal, Cobol, PL/I, C, etc.). Estos programas comúnmente se llaman programas de aplicación. Los ejemplos en un sistema bancario incluyen programas que cargan cuentas, abonan cuentas, o transfieren fondos entre cuentas.

- **Usuarios sofisticados**. Interactúan con el sistema sin programas escritos. En su lugar, ellos especifican sus consultas en un lenguaje de consulta de bases de datos. Cada una de estas consultas se envía al procesador de consultas, cuya función es transformar instrucciones LMD a instrucciones que el sistema gestor de almacenamiento entienda. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría.

- **Usuarios especializados.** Son usuarios especialistas que escriben aplicaciones de base de datos que no son las convencionales desarrolladas por programadores de aplicaciones. Entre estas aplicaciones están los sistemas de diseño asistidas por computadoras, sistemas de bases de conocimiento, sistemas expertos, sistemas que almacenan datos con los tipos de datos completos (por ejemplo, gráficos ó datos de audio), etc.

- **Usuarios normales.** Son usuarios no sofisticados que interactúan con el sistema mediante la invocación de alguno de los programas de aplicación permanentes que se han implementado previamente. Por ejemplo, si un cajero bancario necesita transferir una cantidad de dinero de una cuenta a otra, invocará a un programa llamado transferir. El programa solicita al cajero el importe de dinero a transferir , la cuenta de la que el dinero va a ser transferido y la cuenta a la que el dinero se transferirá.

8. ESTRUCTURA DEL SISTEMA COMPLETO

Un sistema gestor de base de datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. La estructura del SGBD se muestra en la Figura 7 la cual incluye diversos módulos previamente descriptos.

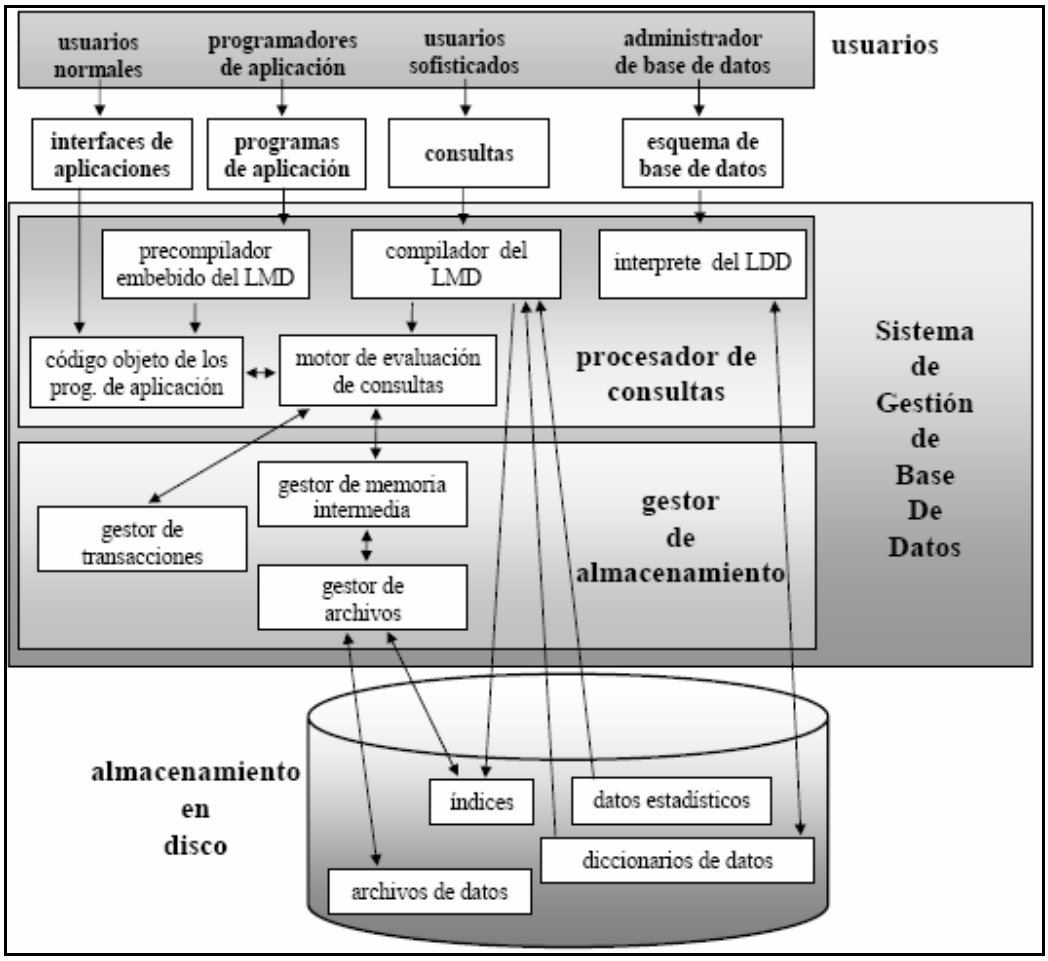

Figura 7: Estructura General del SGBD.