TEMA 34

SISTEMAS GESTORES DE BASES DE DATOS. FUNCIONES. COMPONENTES. ARQUITECTURAS DE REFERENCIA Y OPERACIONALES. TIPOS DE SISTEMAS.

ÍNDICE

1. INTRODUCCIÓN

Una de las aplicaciones más significativas de los ordenadores es el almacenamiento, recuperación y mantenimiento de grandes cantidades de información utilizando entidades hardware/software conocidas como *sistemas de información*.

En una aplicación convencional con ficheros, éstos se diseñan de acuerdo con los programas. Esto es, un vez planteado el programa, se decide si debe haber ficheros, cuántos deben ser, qué organización tendrán, qué información contendrá cada uno, qué programas actuarán sobre ellos y cómo lo harán. Esto tiene la ventaja, en principio, de que los programas son bastante eficientes, ya que la estructura de un fichero está pensada para el programa que lo va a usar; sin embargo, conlleva graves inconvenientes. Por un lado, los programas que se realizan con posterioridad a la creación de un fichero pueden ser muy lentos, al tener que utilizar una organización pensada y creada a la medida de otro programa previo. Por otra parte, si se toma la decisión de crear nuevos ficheros para los programas que se han de realizar, se puede entrar en un proceso de degeneración de la aplicación, ya que:

- gran parte de la información aparecerá duplicada en más de un fichero (redundancia).
- Los procesos de actualización se complican, al existir la misma información en varios ficheros.
- Se corre el riesgo de tener datos incongruentes entre los distintos ficheros. Por ejemplo, tener dos domicilios diferentes de un mismo individuo en dos ficheros diferentes.

En una aplicación convencional con ficheros aparecen, pues, los siguientes *problemas*:

Dificultad de mantenimiento. Si hay ficheros con información duplicada, realizar las actualizaciones necesarias puede ser complejo y costoso. Incluso puede ser necesario utilizar archivos con diferentes organizaciones. Si la actualización no se realiza correctamente en todos los lugares se producirán inconsistencias en la base de datos.

Redundancia. Este problema consiste en tener datos que no aportan información, ya que se pueden deducir de otros datos. El caso típico de redundancia es mantener el mismo dato almacenado en varios sitios.

Rigidez de búsqueda. A cada fichero se le dará una determinada organización, de acuerdo con los tipos de acceso que se creían más frecuentes, pero puede ocurrir que se necesiten otros modos de acceso y sean difíciles de llevar a cabo sobre la organización ya existente.

Dependencia con los programas. Las relaciones entre los datos almacenados en los ficheros no están presentes en éstos. Es el programa que los utiliza el que se encarga de ello. Esto implica que cualquier modificación de la estructura de un fichero obliga a modificar todos los programas que lo usen.

Confidencialidad y seguridad. En general, es necesario impedir la consulta de determinados datos a determinados usuarios (confidencialidad) e impedir modificaciones efectuadas por usuarios no autorizados (seguridad). Si trabajamos convencionalmente con ficheros, el control deberá realizarlo el propio programa. En cualquier caso, no se podrá impedir que alguien construya un programa para modificar o ver el contenido de un fichero, si el sistema operatifvo le permite el acceso.

2. CONCEPTO Y ESTRUCTURA DE UNA BASE DE DATOS

Las bases de datos surgen como alternativa a los sistemas de ficheros, intentando solucionar o reducir sus inconvenientes.

Una **base de datos** *es un sistema formado por un conjunto de datos organizados de tal modo que: se controla el almacenamiento de datos redundantes (puede existir cierta redundancia para hacer más rápido el acceso o para asegurar la integridad de los datos), los datos resultan independientes de los programas que los usan, se almacenan los datos y las relaciones entre ellos y se puede acceder a los datos de diversas formas.*

Los requisitos que debe cumplir un buen sistema de base de datos son:

- **Acceso múltiple**. Varios usuarios pueden acceder simultáneamente a la base de datos, sin que se produzcan conflictos ni vistas incoherentes.
- **Utilización múltiple**. Cada usuario podrá tener una imagen o visión particular de la estructura de la base de datos.
- **Flexibilidad**. Se podrán utilizar distintos métodos de acceso.
- **Confidencialidad y seguridad**. Se controlará el acceso a los datos, impidiéndoselo a los usuarios no autorizados.
- **Protección contra fallos**. Deben existir mecanismos concretos de recuperación en caso de fallo del ordenador.
- **Independencia física**. Se puede cambiar el soporte físico de la base de datos sin que repercute en la base de datos ni en los programas que la usan.
- **Independencia lógica**. Se pueden modificar los datos contenidos en la base, las relaciones existentes entre ellos o incluir nuevos datos, sin que afecte a los programas que la usan.
- **Redundancia controlada**. Los datos se almacenan una sola vez, excepto en casos excepcionales.
- **Interfaz de alto nivel**. Existe una forma sencilla y cómoda de utilizar la base de datos desde un lenguaje de programación de alto nivel.
- **Interrogación directa** (**query**). Existe una utilidad que permite el acceso a los datos de forma conversacional.

2.1. Estructura general de una base de datos

En una base de datos se almacena información de una serie de objetos o elementos. Estos objetos reciben el nombre de **entidades**. Entidad es cualquier ente sobre el que se almacena información.

Así, en una base de datos académicos podrá haber información de las siguientes entidades: alumno, profesor, asignatura, centro, plan de estudios, curso, etc. En una base de datos comercial de una empresa podrán aparecer las siguientes entidades: cliente, producto, vendedor, etc.

De cada entidad se almacenan una serie de datos que se denominan **atributos** de la entidad. Puede ser atributo de la entidad cualquier característica o propiedad de ésta. Así, son atributos de la entidad alumno: DNI, apellidos y nombre, sexo, fecha de nacimiento, etc.

Entidades y atributos son conceptos abstractos. En una base de datos la información de cada entidad se almacena en **registros**, y cada atributo en **campos** de dicho registro, de forma análoga al almacenamiento en ficheros. Sin embargo, cada entidad necesita registros con una estructura específica, mientras que en un fichero todos los registros tienen la misma estructura. Es decir, en una base de datos hay **diferentes tipos de registros**, uno por entidad. Normalmente se reserva el nombre "registro" para especificar un "tipo de registro", usándose otras denominaciones para especificar cada una de las apariciones de ese registro en la base de datos, tales como: **elemento**, **valor actual de registro** u **ocurrencia de registro**. Con esta terminología se puede decir que en la base de datos comerciales antes mencionada, hay un registro de vendedor y tantas ocurrencias de dicho registro como vendedores tenga la empresa.

Por lo general, es suficiente con conocer el valor de uno o varios atributos de una entidad para identificar un elemento. Diremos que un conjunto de atributos de una entidad es una **superclave** de dicha entidad si el valor de dichos atributos permite identificar de forma única cada uno de los elementos de la entidad. Si de una superclave no se puede obtener ningún subconjunto que sea a su vez superclave, se dice que dicha superclave es una **clave candidata**. De todas las claves candidatas existentes en una entidad, el diseñador de la base de datos ha de escoger una, que se denominará **clave principal** o **primaria**. Al resto de las claves candidatas de una entidad (las que no han sido elegidas como primaria) se les llamará **claves alternativas**.

Ejemplo: En la entidad alumno con atributos: N_Matrícula, Nombre, Apellidos, DNI, Dirección, Teléfono, F_Nacimiento, encontramos como ejemplos de superclaves:

- N_Matrícula
- DNI
- Nombre + Apellidos
- Nombre+Apellidos+DNI
- N_Matrícula+DNI

Sin embargo, de éstas son claves candidatas:

- N_Matrícula
- DNI
- Nombre+Apellidos

Ya que, para el resto de las superclaves se puede encontrar un subconjunto de atributos que satisfacen la condición de ser, a su vez, superclaves.

Entre las claves candidatas escogemos como clave principal N_Matrícula, por ser la más discriminatoria. Las otras dos claves candidatas, DNI y Nombre+Apellido, pasarán a ser claves alternativas.

Decimos que existe una **referencia** entre dos entidades cuando un campo o un conjunto de campos de una de las entidades es clave de la otra; esto nos va a permitir localizar una entidad a partir de otra. A este conjunto de atributos que forman la clave de la segunda entidad se le denomina **clave ajena** de dicha entidad.

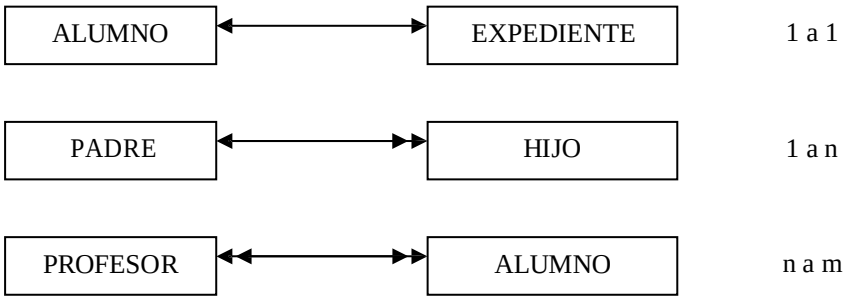
Las entidades, por sí solas, no describen la realidad de un sistema de información. No basta con identificar objetos, tenemos que establecer, además, las **relaciones** existentes entre ellos. En la base de datos se almacenan las entidades y las relaciones. Relación significa la existencia de algo común entre entidades. Así, por ejemplo, en una base de datos académicos existen relaciones entre diversas entidades: cursos y alumnos, alumnos y profesores, profesores y asignaturas.

El **grado de una relación** representa la participación en dicha relación de cada una de las entidades afectadas. Existen tres tipos posibles:

> **1:1** (**una a una**). A cada ocurrencia de una entidad le corresponde no más de una ocurrencia de la otra, y a la inversa. Son relaciones biunívocas. Por ejemplo: la relación entre Alumno y Expediente.
>
> **1:n** (**una a muchas**). A cada ocurrencia de la primera entidad le pueden corresponder varias ocurrencias de la segunda, y a cada ocurrencia de la segunda le corresponde no más de una ocurrencia de la primera. Un ejemplo es la relación Padre-Hijo, puesto que un padre puede tener varios hijos, pero un hijo sólo puede tener un padre.
>
> **n:m** (**muchas a muchas**). A cada ocurrencia de la primera entidad le pueden corresponder varias ocurrencias de la segunda, y viceversa. Un profesor da clase a muchos alumnos y un alumno puede tener varios profesores, por tanto la relación Profesor-Alumno pertenece a este tipo.

Las relaciones con grado uno a uno reciben el nombre de **relaciones simples**, mientras que las relaciones con grado uno a muchos y muchos a muchos se denominan **relaciones complejas**.

El **esquema de una base de datos** es la definición de su estructura lógica, es decir, la especificación de cada uno de los registros que la integran, indicando los campos que los componen y las relaciones que los ligan.



3. TIPOS DE BASES DE DATOS

Tradicionalmente, las bases de datos se clasifican en tres grupos*: jerárquicas, en red y relacionales*. Las bases de datos jerárquicas fueron las primeras en aparecer. La información se representa en forma de árbol. El problema con este tipo de estructura es que no todas las bases de datos se adaptaban a la estructura en árbol. En un intento de eliminar la rigidez de las bases de datos jerárquicas, se desarrolló la estructura en red, en la cual se permitían todo tipo de relaciones. Cualquier conjunto de información es representable mediante una base de datos en red. Por último, aparecieron las bases de datos relacionales, que pretendían obtener una mayor flexibilidad y rigor en el tratamiento de datos. Nacieron en los años 70 de la mano de E. F. Codd, quién planteó una alternativa a las bases de datos jerárquicas y en red existentes hasta el momento.

3.1. Bases de datos jerárquicas

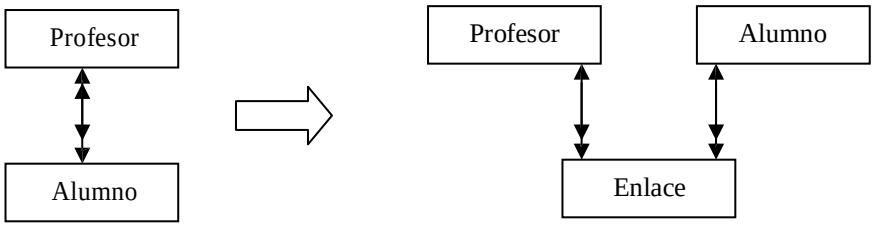
Sólo se pueden crear estructuras jerárquicas en forma de árbol. Están formadas por una colección de registros unidos a través de relaciones que pueden ser de uno a uno o de uno a muchos, no siendo posible definir relaciones de muchos a muchos. Las relaciones se implementan físicamente utilizando punteros (igual que una estructura en árbol).

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y tienen datos muy compartidos, permitiendo crear estructuras estables y de gran rendimiento.

3.2. Bases de datos en red

Son muy parecidas a las jerárquicas. Se permite cualquier tipo de relaciones, aunque, en principio, se distingue entre base de datos en red simple (no permite relaciones muchos a muchos y es la más usual) y base de datos en red compleja. Cualquier sistema es representable en una base de datos en red.

Tomemos, por ejemplo, la relación Profesor-Alumno, que es una relación de muchos a muchos. Dicha relación se puede incluir en una base de datos en red compleja, pero, en principio, no en una base de datos en red simple. Para lograrlo, en el caso simple, se debe expresar la relación muchos a muchos mediante dos relaciones uno a muchos. Para ello, se introduce un nuevo registro, del que existirá una ocurrencia por cada par Profesor-Alumno que estén relacionados. A estos registros que introducimos se les llama enlaces.



3.3. Bases de datos relacionales

Una base de datos relacional está formada por **tablas**. Una tabla es una estructura bidimensional formada por una sucesión de registros del mismo tipo. Si se imponen ciertas restricciones a las tablas, se pueden tratar como relaciones matemáticas. De ahí el nombre de este tipo de base de datos y el hecho de que a las tablas de una base de datos relacional se les llame relaciones.

Una tabla es la forma en que un usuario ve sus datos. Se divide horizontalmente en filas y verticalmente en columnas. Una fila representa un registro o un grupo de valores de campos, los cuales contienen toda la información necesaria sobre la entidad de la tabla. Una columna contiene información referente a un único campo o atributo.

Se define una clave como un atributo o conjunto de atributos que identifican una fila. Los requisitos que debe cumplir una clave son: identificación unívoca y no redundancia. Al conjunto de todas las claves posibles se les llama claves candidatas, y de entre ellas se elige una, que será la clave primaria.

Las tablas deben cumplir:

- Todos los registros de una tabla son del mismo tipo. Para almacenar registros distintos se usan tablas distintas.
- Cada columna se identifica mediante un nombre de columna.
- En ninguna tabla aparecen nombres de columnas repetidos.
- En ninguna tabla existen registros duplicados (filas idénticas).
- El orden de los registros en la tabla es indiferente. En cada momento se pueden recuperar los registros en un orden particular.
- En cada tabla hay una clave formada por uno o varios campos.
- El contenido de las tablas es independiente del almacenamiento físico de los datos.

El modelo relacional es uno de los más utilizados en el diseño y gestión de bases de datos, debido fundamentalmente a dos razones: por una parte, se basa en un reducido número de conceptos, que hacen de este sistema uno de los más fáciles de entender, diseñar, cambiar, administrar y acceder, y, por otra parte, posee un lenguaje de definición y manipulación de datos muy potente y flexible.

4. SISTEMAS GESTORES DE BASES DE DATOS (SGBD)

Se denomina **sistema de gestión de base de datos** al conjunto de software destinado a la creación, control y manipulación de la información sobre una base de datos. Concretamente, un SGBD debe permitir la realización de las siguientes tareas:

Definición del esquema de la base de datos. Una vez diseñado el esquema de la base de datos, hemos de describirlo mediante un conjunto de instrucciones. Esto se realiza mediante un lenguaje específico, denominado **lenguaje de definición de datos** (**DDL**).

Acceso a los datos desde un lenguaje de alto nivel. Esto se realiza también mediante un lenguaje específico, denominado **lenguaje de manipulación de datos** (**DML**). El DML puede ser utilizado de dos formas diferentes. Por una parte, se incluyen sentencias DML en programas escritos en lenguaje de alto nivel. Esto hace necesario incluir en el SGBD un precompilador que traduzca las instrucciones DML en instrucciones reconocibles por el compilador del lenguaje de alto nivel (lenguaje anfitrión). La otra forma de utilización del DML es mediante programas que contengan exclusivamente sentencias propias de este lenguaje.

Acceso a la información en modo conversacional. El SGBD debe incorporar una interfaz de usuario a través de la cual introducir sentencias directamente desde un terminal, para obtener información interactiva.

Gestión de ficheros. Función realizada por un módulo gestor de ficheros que se encarga de la comunicación con el sistema operativo. Además, realiza otras funciones, tales como control de usuarios, recuperar la información tras fallos del sistema, organización física de la base de datos, control de seguridad y privacidad, y gestión de accesos concurrentes.

En un sistema tradicional de ficheros, cada una de las aplicaciones deberá realizar la descripción de los registros, así como determinar la organización de ficheros, los tipos de acceso, etc. En un entorno de base de datos dichas especificaciones las realiza el SGBD, y es el administrador de la base de datos el encargado de realizar éstas y otras funciones sobre el sistema.

4.1. Abstracción de la información

El SGBD debe proporcionar información a usuarios y desarrolladores a distintos niveles, representando cada uno de ellos una abstracción de datos:

Nivel de visión. A este nivel, cada grupo de usuarios posee conocimiento únicamente de aquella parte de la base de datos que le afecta. El usuario sabe de la existencia de los datos y su significado, pero ignora los detalles sobre su formato, tipo, estructura, y, en general, cualquier aspecto físico.

Nivel conceptual. La unión de todas las vistas da lugar a este nivel. En él se conoce la descripción de todos los datos y las relaciones existentes entre ellos.

Nivel físico. En este nivel se describe cómo se encuentran los datos almacenados físicamente en memoria secundaria. Es el nivel más cercano al hardware y se encuentra íntimamente ligado a él.

4.2. Componentes de un sistema de gestión de bases de datos ideal

Debería contar con:

- **Lenguaje de descripción del esquema conceptual**, fácil de usar y capaz de determinar la consistencia de los datos.
- **Lenguaje de definición de esquemas**, para especificar las restricciones de privacidad, la seguridad y vías de acceso.
- **Módulo de transformación lógica** a física.
- **Módulo de privacidad** de propósito general que proteja la base de datos contra accesos no autorizados.
- **Módulo de integridad** de propósito general para mantener la integridad de los datos.
- **Generador de programas de informes**.
- **Lenguaje de consulta** de propósito general.

4.3. Funciones del sistema de gestión de bases de datos relacionales

Un SGBD es un conjunto de programas que va a permitir insertar, modificar, borrar y buscar eficazmente datos específicos entre un volumen masivo de información compartida por todos los usuarios de la base de datos; pero también es una herramienta que va a permitir ordenar, buscar, reordenar y convertir datos.

Realiza una serie de *funciones* que pueden agruparse dentro de tres niveles:

1) Un **primer nivel**, en el que se incluyen todas las funciones relativas a la gestión de tablas o estructuras de datos en memoria secundaria. Estas funciones son misión del sistema operativo, pero hay casos en los que los SGBD también las realizan para conseguir mayor efectividad.

2) En un **segundo nivel**, gestiona los datos almacenados en las estructuras, la colocación y reagrupación de los datos, la gestión de los enlaces que se establecen entre ellos y las estructuras que van a permitir recuperarlos.

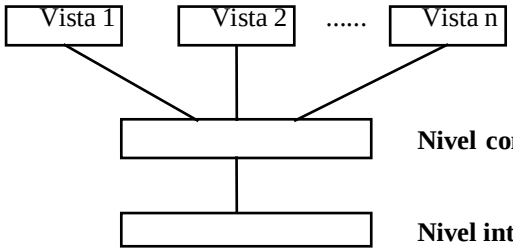
3) En un **tercer nivel**, se encuentran las funciones que permiten la interpretación y el análisis de las peticiones de los usuarios, así como la presentación de los datos solicitados por el programa de aplicación o por el propio usuario.

5. ARQUITECTURA DE UN SISTEMA DE GESTIÓN DE BASES DE DATOS

Un **SGBD** es un conjunto de procedimientos, ayudas de documentación, lenguajes y programas de software que administran los ficheros de la base de datos.

Uno de los *objetivos* de un SGBD es proporcionar a los usuarios una visión abstracta de la información, es decir, ocultar ciertos detalles referentes a la forma en que los datos se almacenan y mantienen, pero siempre permitiendo una recuperación eficaz de la información.

Existen varios paquetes de SGBD en el mercado con diferentes arquitecturas; una de ellas, la más estandarizada, es la que cumple los requerimientos de la normativa ANSI-SPARC, que en su división X3 establece: "la arquitectura de una base de datos debe poseer tres niveles: Interno, Conceptual y Externo". Cada uno de ellos pertenece a un tipo de vista diferente: almacenamiento físico, del usuario, y del programador.



Nivel externo.
(Vistas individuales de los usuarios).

Nivel conceptual (Vista comunitaria de los usuarios).

Nivel interno. (Vista de la forma de almacenamiento).

5.1. Nivel Interno

Es el nivel más bajo de abstracción de la información. Es la representación inferior de una base de datos, por ello, es el más cercano al almacenamiento físico. Permite describir los datos tal como están almacenados en el ordenador. Por ejemplo:

- Los ficheros que contienen (nombre, abstracción, ubicación)
- Los registros de estos ficheros (longitud, campos, ...)
- Las rutas de acceso a esos registros (índices, encadenamientos, ...)

El nivel interno es descrito por el SGBD por medio del **esquema interno** o **vista interna**.

Hay que recordar que para un programador un registro interno almacenado difiere de un registro lógico. Un registro físico puede estar formado por uno o más registros lógicos junto a elementos descriptivos del sistema: longitud y tipo de registro, banderas y punteros, y está almacenado en un dispositivo electrónico codificado como una combinación de ceros y unos.

La operación de transformar registros lógicos en físicos se denomina *transformación de datos o mapeo*.

Además, la implantación del esquema interno requiere de:

- **Archivos de datos**. Almacenan la base de datos.
- **Archivos de índices**. Permiten el acceso rápido a la información deseada.
- **Diccionario de datos**. Almacena la información relativa a la estructura de la base de datos.

5.2. Nivel conceptual

Es el siguiente nivel más alto de abstracción, el administrador de la base de datos del SGBD define el nivel conceptual por medio de un **esquema** o **vista conceptual**, al decidir qué información se guarda en la base de datos.

Este nivel corresponde a la estructura organizacional de los datos de la base, obtenida al reunir los requerimientos de todos los usuarios de una empresa, sin preocuparse de su organización física ni de las vías de acceso.

El esquema conceptual podría contener:

- Los datos elementales que definen los campos, **atributos** de los elementos de una empresa (NIF, nombre, concepto, ...).

- Los datos compuestos que permiten reagrupar los **campos** para describir los registros, las entidades del mundo real (persona, artículo, ...).
- Los datos compuestos que permiten reagrupar los campos para describir las **asociaciones** del mundo real (compras de artículos, propietarios de viviendas, ...).
- Algunas veces, las reglas que deberán seguir los datos en la empresa (que el stock mínimo esté comprendido en unos márgenes, que cada artículo corresponda con un solo proveedor, ...).
- Relaciones entre los datos, para conectar o relacionar registros en el procesamiento de ficheros múltiples.

En el nivel conceptual, la base de datos aparece como una colección de registros lógicos sin especificar su almacenamiento. En realidad, los ficheros conceptuales no existen físicamente.

A la parte del esquema conceptual que es de interés para un usuario o un grupo de usuarios finales se le llama **subesquema conceptual**.

5.3. Nivel externo

Es el nivel más alto de abstracción, y, por ello, el más cercano a los usuarios. El nivel externo representa la percepción individual de cada usuario o programador de la base de datos.

Si en los niveles interno y conceptual los esqumas describen toda una base de datos, el nivel externo describe únicamente la parte de datos de interés para un usuario o grupo de usuarios. Los usuarios pueden imaginar que los ficheros externos utilizados en sus programas existen tal y como ellos los perciben. Pero los ficheros externos tampoco existen físicamente.

El SGBD es el encargado de extraer los datos requeridos por los registros lógicos externos de uno o más registros físicos de la base de datos, cada vez que se ejecuta una operación de entrada/salida en un programa específico.

Por cada programa es necesario especificar un **esquema externo**, **subesquema** o **vista externa** para poder acceder de forma selectiva a un subconjunto de datos de la base integrada. Habrá usuarios que puedan acceder a más de un esquema externo, y un esquema externo puede ser compartido por varios usuarios; se protege, de esta manera, el acceso no autorizado a datos y el de usuarios mal intencionados.

A la hora de construir un subesquema hay que tener presente:

- Uno o más tipos de registros pueden omitirse en el esquema.
- Unos o más campos pueden omitirse en el registro conceptual.
- Puede cambiarse el orden relativo de los campos del registro.
- Una o más relaciones entre los datos pueden omitirse en el esquema conceptual.

Para una base de datos específica, habrá un esquema interno, y un único esquema conceptual, pero puede haber varios esquemas externos. Cada uno de los esquemas externos puede estar definido para un grupo de usuarios.

6. USUARIOS DE LA BASE DE DATOS

Se consideran tres clases de usuarios que manipulan un base de datos:

Programador de aplicaciones. Encargado de escribir programas de aplicación que utilicen las bases de datos. Estos programas se escriben en algún lenguaje de programación de alto nivel, y están diseñados para apoyar a un usuario final.

Usuario final. Accede a la base de datos desde un terminal empleando un lenguaje de consulta (DML) o a través de un programa de aplicación.

Administrador de la base de datos. Es la persona (o grupo de personas) encargada del control general del sistema de base de datos. Entre sus responsabilidades se incluyen las siguientes:

- Decidir el contenido de la información de la base de datos.
- Decidir la estrucutura de almacenamiento y la estrategia de acceso.
- Vincularse con el resto de usuarios de la base de datos.
- Definir los controles de autorización y procedimientos de validación.

- Definir una estrategia de respaldo y recuperación tras posibles fallos del sistema.
- Controlar el rendimiento y utilización de la base de datos.
- Responder a los cambios de requerimientos.

7. LENGUAJE DE DEFINICIÓN DE DATOS (DDL) Y LENGUAJE DE MANIPULACIÓN DE DATOS (DML) EN SQL.

Cualquier lenguaje de bases de datos está formado por un DDL y por un DML. Existe un lenguaje considerado como estándar para manipular bases de datos relacionales: SQL (Structured Query Language). Estudiaremos el DDL y el DML que utiliza el SQL. Las características principales de SQL son su sencillez, su carácter estándar y ser un lenguaje declarativo o no procedural (para que SQL realice una acción concreta no se le dice cómo tiene que hacerla, sino lo que queremos obtener).

El **lenguaje de definición de datos** (**DDL**) proporciona órdenes para definir, eliminar y modificar tablas, así como para crear índices y vistas. Las órdenes SQL más importantes del DDL son:

CREATE TABLE. Define el esquema de la base de datos. Crea una tabla con los campos y tipos indicados. Su sintaxis es la siguiente:

CREATE TABLE nombre_tabla (n_campo1, t_campo1, ..., n_campoN, t_campoN)

n_campo es el nombre de un campo y t_campo es su tipo

ALTER TABLE. Con esta orden se pueden añadir nuevos campos a una tabla ya existente, modificar o borrar los campos de la tabla. Su sintaxis es la siguiente:

ALTER TABLE nombre_tabla ADD (n_campo, t_campo)
ALTER TABLE nombre_tabla MODIFY (n_campo, t_campo)
ALTER TABLE nombre_tabla DROP n_campo

DROP TABLE. Borra el esquema de la base de datos, es decir, destruye tanto los datos contenidos en la tabla como la estructura de la tabla. Su sintaxis es:

DROP TABLE nombre_tabla

El **lenguaje de manipulación de datos** (**DML**) está basado en el álgebra relacional, e incluye órdenes para insertar, suprimir y modificar tuplas (filas) de la base de datos. Con el DML podemos trabajar de dos formas: de manera interactiva, a través de un terminal, o utilizándolo como un lenguaje huésped dentro de un programa escrito en otro lenguaje de alto nivel. Las órdenes DML son:

INSERT. Inserta nuevas filas en la base de datos. Su formato es:

INSERT INTO nombre_tabla VALUES (valor_campo1, ..., valor_campoN)

UPDATE. Modifica los valores de determinados campos. Si se especifica una condición, únicamente actualizará los valores de los campos de las filas que cumplan la condición. Su sintaxis es:

UPDATE nombre_tabla SET campo=valor_campo [WHERE condición]

DELETE. Borra filas de la tabla especificada. Si no se pone condición, se eliminarán todas las filas de la tabla. Si se especifica una condición, únicamente se borran los registros que satisfagan dicha condición. Su sintaxis es:

DELETE FROM nombre_tabla [WHERE condición]

SELECT. Se utiliza para consulta de tablas. Es la orden principal en SQL. Selecciona un conjunto de registro de una o más tablas que cumplan una condición. Su sintasis más usual es la siguiente:

SELECT campo1, ..., campoM FROM nombre_tabla [WHERE condición] [ORDER BY campo]

8. BIBLIOGRAFÍA

Gary W. Hansen
Diseño y Administración de Bases de Datos
Prentice Hall, 1998

Alberto Prieto
Introducción a la Informática
Mc Graw-Hill, 2ª edición, 1997

Alfonso Ureña López
Fundamentos de Informática
Ra-ma, 1997