41

UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGON

SISTEMA DE BASE DE DATOS PARA EL MANEJO DE PROPUESTAS EN LA CARRERA DE INGENIERIA EN COMPUTACION

T E S I S

QUE PARA OBTENER EL TITULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

J U A N M O Y L A R A

ASESOR: ING. GLADIS EMILIA FUENTES CHAVEZ.

MÉXICO 2002

DEDICATORIAS Y AGRADECIMIENTOS

Gracias **Dios** por estar conmigo a cada paso, por la oportunidad de un nuevo día, por darme siempre fuerzas para seguir adelante y la maravillosa vida que no cambiaría por nada.

A mis **Padres** gracias por su amor, su confianza tan grande en mi, por su guía, apoyo y ejemplo, gracias por la hermosa vida que me han dado, me siento feliz y orgulloso de ser su hijo, todo mi amor y respeto para ustedes.

A mis hermanos y amigos **Lic. Cristina, Lic. Francisco, Lic Gustavo**, mis compañeros de juegos y desvelos, gracias por su ayuda, compañía y amistad, los quiero y admiro cada día más.

A **Gabriela Ayala** mi amiga y compañera, gracias por todo tu apoyo y entusiasmo, por ser presencia a pesar de la distancia, tu amor es inspiración y aliento en mi vida, te amo.

A mi grandes amigos **Julia**, **David** y **Noé**, con ustedes he compartido sueños, alegrías, tristezas, hemos crecido juntos, como personas y profesionistas, los mejores recuerdos que tengo de la UNAM son al lado suyo, gracias por su amistad.

A mi asesora y amiga **Ing. Gladis Fuentes Chávez,** gracias por toda tu amistad, disposición y apoyo, por encauzar mis esfuerzos y tu gran aportación en mi trabajo, tienes todo mi cariño y admiración.

A mi revisor y profesora **Ing. Silvia Vega Muytoy** gracias por su gran apoyo, amistad y su enorme disposición para la conclusión de este trabajo, le doy mi más sincero agradecimiento.

A mi revisor y profesor **Mat. Luis Ramírez Flores** gracias por su apoyo y disposición, siempre le tendré presente con mucho respeto, cariño y admiración.

A mi revisor y amigo **Ing. Rodolfo Vázquez Morales**, muchas gracias por todo tu apoyo y comprensión, por enriquecer con tus observaciones mi trabajo de tesis, pero sobre todo gracias por tu mistad.

A mi profesor y amigo **Ing. Martín Ordóñez**, muchas gracias por todo su apoyo y ayuda para la culminación de este trabajo.

A mi gran amiga **Elvia Duque**, gracias por tu amistad y tu cariño, me siento orgulloso y feliz de considerarme amigo tuyo, te quiero mucho.

A mi amigo **Ing. Roberto Márquez,** tu que me acompañaste en todo el proceso de mi tesis, entiendes el esfuerzo que ello significó. amigo gracias por tu amistad y apoyo, no pude haber encontrado mejor amigo que tú en la ENEP.

A mis amigos **Luis Andrés Miranda y Enrique Gutiérrez,** son tantos años y experiencias juntos, dentro y fuera de la escuela, que no puedo menos que considerarlos dentro de mi familia, muchas gracias por su amistad.

INDICE

Sistema de Base de Datos para el manejo de propuestas en la Carrera de ingeniería en computación.

Sería imposible concebir el entorno social y económico actual, sin el manejo oportuno y eficaz de grandes volúmenes de información, la noción que se tenía de ésta, como conocimiento transmisible, ha evolucionado a *recurso fundamental* en la toma de decisiones, conforme nos damos cuenta del poder real que nos da la información, se torna imperativa la necesidad de establecer mecanismos, para almacenar, actualizar y salvaguardar la integridad de los datos, sin entorpecer la libre circulación de la información.

Las bases de datos, surgen como una solución, o al menos como parte principal en la solución a esta necesidad, por ello es de suma importancia, para todo aquél que esté involucrado en el manejo de base de datos, conocer las técnicas relativas a la creación y modelado de las mismas, pues hay que tener en cuenta que las bases de datos, ayudan a resolver un problema del mundo real y aún cuando existe software que facilita la definición de la base de datos y el manejo de los datos contenidos en ella, no pueden auxiliar al usuario en la transformación de un problema real, a un modelo lógico entendible por los sistemas de gestión de bases de datos.

El sistema de gestión de bases de datos (SGBD) es el conjunto de programas que permiten la implantación, acceso y mantenimiento de la base de datos. El SGBD, junto con la base de datos y con los usuarios, constituyen el sistema de base de datos.

Este trabajo de tesis, pretende ayudar al usuario en los pasos previos a la construcción de la base de datos, que son de vital importancia para garantizar el buen funcionamiento del sistema que se pretende construir, por supuesto existen diferentes tipos de usuarios, y la profundidad en el conocimiento del diseño de bases de datos, dependerá de las necesidades de cada uno. El presente, aporta una obra de consulta a conceptos básicos, para aquellos usuarios que tienen apenas un primer contacto con las bases de datos, así como las bases teóricas en las que se soportan las tecnologías de los SGBD, esto para usuarios más avanzados, aún así, el contenido no deja de ser introductorio, más pretendo reforzarlo incluyendo la implantación de una solución, a un problema del mundo real, por lo que trataré este texto en dos partes claramente diferenciadas, en la primera, abordaré los fundamentos para el modelado de base de datos y en la segunda, la implantación de un sistema de base de datos, para la solución de un problema específico.

CAPÍTULO 1. SISTEMAS DE INFORMACIÓN Y BASES DE DATOS.

Las bases de datos van mas allá de ser un conjunto estructurado de datos, son piezas clave en el desarrollo y mantenimiento de un Sistema de Información (S.I), y tienen su razón de ser en la misma existencia de este. Por tanto es conveniente, antes de definir lo que es una base de datos, analizar algunos conceptos de relativos a la información y a los sistemas de información.

1.1 LA INFORMACIÓN COMO RECURSO INDISPENSABLE EN LA SOCIEDAD.

Nadie pondría en duda la importancia que la información tiene en nuestros días, ya que el concepto que se tenía de información, como un elemento de entrada/salida relacionado con la actividad de investigación, en donde el círculo de beneficiarios estaba integrado únicamente por los científicos e investigadores que elaboraban y por tanto poseían dicha información, evolucionó a un conocimiento transmisible, que beneficia a todos aquellos sectores que estén interesados en hacerse herederos de dichos conocimientos, esto abrió la posibilidad de que la información alcanzara espacios dentro de actividades socioculturales como la educación, la medicina y hasta la comunicación en masa, como consecuencia, se le comenzó a dar mayor importancia, a tal grado que llega a ser indispensable en la planificación y toma de decisiones, convirtiéndose así en un recurso fundamental, de nuestra sociedad actual.

SYNNOT[1] describe las oportunidades y las estrategias para los sectores de la información, y sugiere como se debe administrar éste recurso, e insiste en que en las próximas generaciones, solo sobresaldrán aquellas organizaciones que administren la información como recurso fundamental.

Si admitimos esta premisa como verdadera, entonces cobra verdadero significado la importancia de la libre circulación de la información, que responde a una necesidad de conocer el entorno social, cultural, económico y político en el que nos desenvolvemos con fines de investigación y toma de decisiones.

Ahora bien, nuestra necesidad por la información y la mayor disponibilidad de este recurso, nos han llevado a producir tal cantidad de datos, que perdemos la noción de las características básicas que debe tener la información que generemos.

[1] SYNNOT es autor del libro *Gestión de la información como recurso*, (1981)

1.2 CUALIDADES DE LA INFORMACIÓN

Las cualidades que debe de tener la información para ser considerada un recurso fundamental son básicamente: precisión, oportunidad, significado e integridad. Todas ellas en el grado que exija cada sistema en concreto.

La **precisión** se refiere al porcentaje de información correcta respecto a la información total del sistema, los resultados que arroje el sistema deben ser coherentes con las entradas que introdujo el usuario, se debe reducir al mínimo el grado de error, pues una precisión baja, lleva a una falta de credibilidad en el usuario hacia la información que le es proporcionada por el sistema.

La **oportunidad** se refiere al tiempo transcurrido desde el momento en que se produjo el hecho que originó el dato, hasta el momento en que la información llega a manos del usuario que la requiere. Hay que tener en cuenta que en general la información pierde valor con el tiempo, e incluso en algunos casos llega a perder total relevancia después de cierto tiempo, la pérdida del valor de la información será mas o menos rápida dependiendo del tipo de información.

La información que se suministra al usuario debe ser **significativa**, es decir, debe ser comprensible e interesante para quién la solicita, la información suministrada, tiene que ser fácilmente interpretable, sólo la necesaria y suficiente para que se cumplan los fines propuestos. Sin redundar o exponerse en grandes masas, pues esto dificulta su rápida asimilación.

Toda información contenida en el sistema debe ser coherente en sí misma, y consistente con las reglas semánticas del mundo real al que han de representar lo más fielmente posible, ésta cualidad es conocida con el nombre de **integridad**.

Los puntos anteriores son los requisitos que se deben tomar en consideración al implantar un sistema de información, teniendo en cuenta que entre más cualidades reúna la información, más elevado será el costo de implantación y mantenimiento, teniendo en cuenta también que algunas cualidades de la información pueden resultar incompatibles con otras, por ejemplo, una gran precisión requiere de un mayor tiempo en el proceso de generar la información, lo que trae consigo una reducción en la oportunidad. Por ello es importante que el diseñador, busque el punto de equilibrio entre las diferentes cualidades dependiendo de las necesidades del receptor de la información.

1.3 CICLO BÁSICO DE LA INFORMACIÓN

La información es el eslabón que une a todos los componentes de una organización, desde el momento en que se genera por primera vez, entra en un círculo continuo, cuya actualización debe ser permanente, los datos son procesados mediante modelos para crear la información, ésta información es enviada al receptor, quién la evalúa y parte de ella para tomar decisiones, las decisiones generan resultados, que a su vez son capturados y sirven como entradas de datos que serán procesados, repitiéndose así nuevamente el ciclo.
(Véase figura. 1.1)

Los resultados a evaluar, tienen que ser presentados de manera agradable y objetiva, para que de esta forma estén disponibles y accesibles para cualquier persona que funja como receptor.

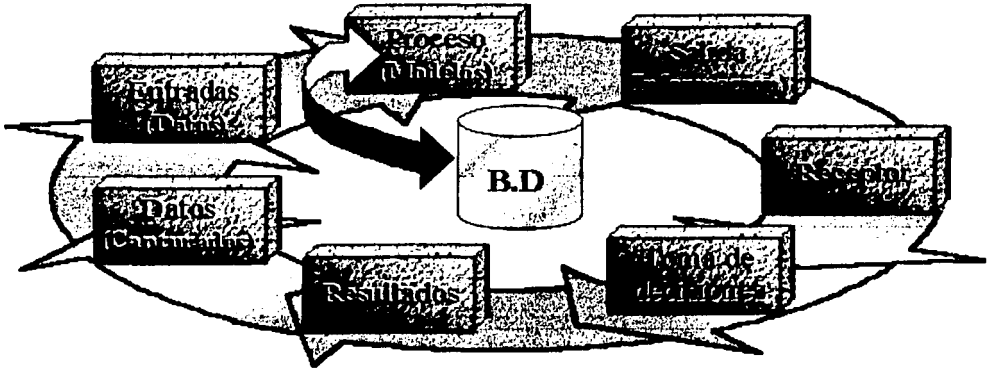


Figura 1.1 Ciclo Básico de la Información.

1.4 CONCEPTO DE SISTEMA DE INFORMACIÓN.

La información que es generada en toda organización[2], necesita ser transmitida a todas las personas que la componen, generalmente se hace desde y hacia el exterior del sistema, una parte de esa comunicación se realiza directamente entre los empleados, a esta forma de interacción se le conoce como *sistema de información informal*, pero cuando se trata de organismos complejos, esta práctica resulta inadecuada e incluso llegan a costar graves errores que se traducen en pérdidas económicas, siendo preciso entonces disponer de un *sistema de información formal*, también llamado *organizacional* o simplemente *sistema de información*, que integrado a un sistema de orden superior que es el organismo, aporta a éste la información necesaria, de manera que se cumplan los objetivos del sistema.

Antes de definir a los sistemas de información, analicemos los sistemas en general.

El vocablo *sistema* es definido como: "Conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a determinado objeto"[3].

Esta definición puede servir para nuestros fines, siempre y cuando precisemos los siguientes parámetros.

[2] La Organización está formada por personas que se unen para lograr un objetivo común, crear y ofrecer un producto o servicio.
[3]Diccionario de la Real Academia Española

El término "cosa", es definido como "*todo lo que tiene entidad, ya sea corporal o espiritual, natural o artificial, real o abstracta*"[4]. Entonces el sistema puede estar constituido por actividades, personas, seres inanimados, ideas, símbolos etc.
Llamaremos, desde ahora, *elementos* a las cosas que integren al sistema.

Visto desde este enfoque, a excepción del universo, que representa el todo, cualquier sistema siempre formará parte de otro sistema que lo engloba por ejemplo, un sistema de base de datos se puede considerar como un subsistema de un sistema de información y éste a su vez es un subsistema de la organización.

Los sistemas están natural o artificialmente limitados, llamándose a todo lo que se encuentra fuera de esta periferia entorno o ambiente del sistema, y es de ahí donde el sistema toma los elementos de entrada que después de ser procesados regresan a él como salidas (Véase figura 1.2).

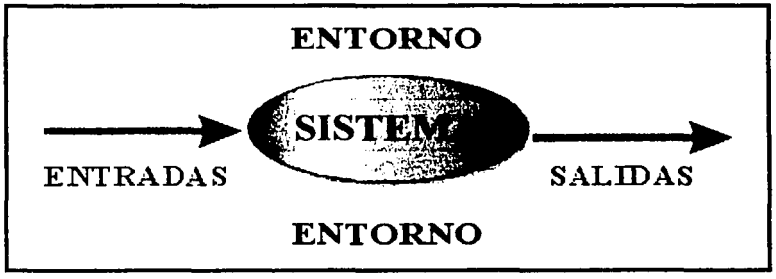


Figura 1.2. El sistema y su entorno

Ya definimos lo que es un sistema en general, ahora de una forma particular, definiremos a un *sistema de información* como el conjunto de todos los procedimientos y dispositivos implicados en el proceso, almacenamiento y distribución de la información en una organización.

El S.I, ha de tomar los datos del entorno, que es la propia empresa y quizá algunas fuentes externas, los resultados son la información que la empresa necesita para la toma de decisiones, este sistema estará regulado por las normas y directrices que formulen los directivos de la organización.

El S.I está integrado por un conjunto de módulos relacionados entre sí que funcionan como subsistemas del S.I y cada uno de ellos se ocupa de una tarea en específico. Uno de los módulos fundamentales para facilitarle al S.I el cumplimiento de las funciones de recuperación, elaboración y presentación de la información es el Sistema de Base de Datos.

[4]Diccionario de la Real Academia Española.

Las características de un S.I, según BUBENKO[5] (1980), pueden agruparse en:

- **Tecnológicas,** que afectan al rendimiento y seguridad del sistema, desde el punto de vista del equipo.
- **Funcionales y semánticas,** que se refieren a si el sistema hace lo que debe de una forma correcta (eficacia) y si es capaz de adaptarse a requisitos cambiantes.
- **Económicas,** que ponen el énfasis en el coste del sistema y en la eficiencia[6] con que responde a los objetivos.
- **Sociales,** que son las que tienen un impacto sobre el entorno social (interno o externo) en que se desenvuelve el sistema.

Podemos comparar al S.I con un motor que impulsa la información, haciéndola circular por el organismo, distribuyéndola y aportándola a aquellas áreas donde es necesaria. Para llevar a cabo ésta tarea, primero se debe recabar los datos que serán utilizados para elaborar información útil, es aquí donde resulta importante recordar, que los datos son recabados de usuarios que esperan una fácil interacción con el sistema, los fracasos en una aplicación, se encuentran en mayor medida relacionados a los aspectos sociales y humanos y no en el diseño tecnológico.

Recordemos también que la salida entregada por el sistema, es decir la información procesada, representa una solución del mundo real, generada para que usuarios del sistema tomen decisiones, aquí es donde se pone a prueba, la capacidad del diseñador del sistema, para crear una interfaz amigable, que responda adecuadamente a las necesidades del usuario, pero desde el punto de vista operacional.

Hay que señalar que los S.I pueden no estar informatizados, ya que no dependen forzosamente de computadoras para funcionar, pero en la mayoría de los casos, se apoyan en técnicas informáticas y la manipulación de la información se hace generalmente por medio de sistemas de administración de bases de datos.

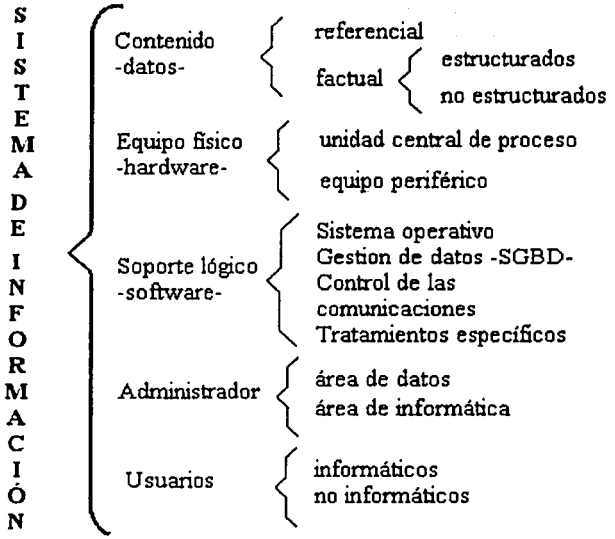
---

[5] Bubenko, J. Computer-Aided Information Systems Analysis and Dedign, Stundentlitteratur, Lund, Suecia, 1971

[6] La eficiencia está enfocada hacia operaciones y relaciones internas del sistema, y mide el grado de optimización de los recursos disponibles.

1.4.1 Componentes de un sistema de información

El siguiente esquema, resume los elementos que constituyen un S.I.



El *contenido* del S.I es el conjunto de todos los datos, texto, imágenes, archivos, Base de Datos, etc. almacenados de manera permanente. Estos datos tienen que adaptarse a los objetivos que se quieran alcanzar por la organización.

Los datos almacenados en el S.I, pueden ser de dos tipos: *referencial* o *factual.* El contenido referencial almacena referencias bibliográficas de los documentos donde se puede encontrar la información, mas no la información en sí, de modo ésta referencia sólo nos indica donde está la fuente, en cambio los tipos de datos factuales, devuelven la información buscada, sin necesidad de una búsqueda adicional.

Otra clasificación, pero aplicable a los datos factuales, es con respecto a su formato, según la cuál, los datos pueden estar estructurados o no estructurados, decimos que los datos están estructurados cuando han sido almacenados con algún formato, por ejemplo, en el caso de los datos de un empleado, decimos que tienen un formato, si las primeras posiciones del archivo son para el nombre, enseguida se reservó otro espacio para almacenar el apellido paterno y por último reservamos un espacio más para el apellido materno. Por el contrario, existen datos, que no podemos darles un formato fijo, como es el caso de datos multimedia (voz, imágenes, videos, etc.), éstos entran en la categoría de no estructurados.

El *equipo físico* es conformado por toda la tecnología de la hacemos uso en el S.I, para el tratamiento de los datos (computadoras, almacenamiento auxiliar, telecomunicaciones etc.).

El *soporte lógico* se compone de todo el conjunto de programas, documentación, lenguajes de programación, etc. En fin, todo el software necesario para que el sistema que administra la base de datos pueda controlar las comunicaciones en el sistema y dar respuesta a los tratamientos específicos en los datos, por supuesto, siempre apoyado en el sistema operativo.

Otro componente importante es el *administrador*, o mejor dicho la unidad de administración, ya que se trata de una acción y no de una persona, cuyo fin es asegurar permanentemente el uso correcto de los datos, así como garantizar la seguridad de los mismos.

El último componente del sistema son los *usuarios*, puede estar conformado desde una persona hasta un grupo de personas que han de acceder al S.I. Éstos usuarios pueden o no tener conocimientos informáticos, por lo que se debe usar un lenguaje muy sencillo para que interactúen con el S.I. o bien procedimientos de ágil interacción con el sistema.

1.4.2 Niveles de administración de datos dentro de un S.I

En toda organización los datos son manejados en distintos niveles, generalmente suelen distinguirse tres (operacional, táctico y estratégico), por lo cuál el sistema de información estará compuesto por tres subsistemas estructurados jerárquicamente y que corresponden con cada uno de los mencionados con anterioridad, éstos, gestionarán la información basados en las actividades que se llevan a cabo en el plano operacional de la organización, no se debe olvidar, que la información que generamos, caracteriza de una u otra forma las actividades llevadas a cabo por una empresa y dependiendo del nivel en el que se soliciten los datos, serán las exigencias de los usuarios, de no tomar en cuenta estas necesidades, la información producida sería inadecuada y en el peor de los casos nuestro sistema inoperante, al menos para ese nivel de administración de datos.

La figura 1.3 representa los tres niveles de administración en una organización, donde podemos observar que mientras la información fluye en sentido ascendente, las órdenes y los planes se mueven en sentido descendente.

Figura 1.3 Niveles de administración de las organizaciones

1.5 CONCEPTO DE BASE DE DATOS

Hasta ahora se ha enfatizado la importancia de la información en una empresa y el papel que juegan los sistemas de información en una organización, ahora toca el turno a las bases de datos.

El concepto de base de datos surge a finales de la década de los 60's, como propuesta para dar solución a los problemas técnicos y administrativos derivados del manejo de sistemas de archivos. Conforme los sistemas de información se volvían más complejos y abarcaban cada vez un mayor número de áreas operativas y administrativas dentro de las empresas, resultaba más difícil mantener a estos sistemas funcionando, eso sin mencionar los elevados costos que ello implicaba.

Las bases de datos, surgen como una solución al planteamiento de mejorar las formas de manipulación y almacenamiento de los datos dentro de los sistemas de información, las bases de datos son un instrumento que está orientado a mejorar la calidad en la administración de los datos, pero el éxito o el fracaso dependerá del buen uso que sepamos dar a éstas.

Definición de base de datos

Son muy numerosas las definiciones que existen referentes a las bases de datos, analizándolas con detenimiento, podemos encontrar que existen muchas similitudes entre ellas, por ejemplo: la mayoría de las definiciones coinciden en que una base de datos es un conjunto o colección de datos almacenados de forma no volátil, además de estar estructurados y relacionados entre sí.

A continuación presento algunas definiciones, que a mi consideración, reúnen las características esenciales de lo que es una base de datos.

"Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias, su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados."[7]

"Colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones."[8]

"Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos, y su definición (estructura de la base de datos) única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cuál ha de permitir captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitarán la seguridad del conjunto de los datos."[9]

De las definiciones anteriores, se pueden extraer las características principales de una base de datos.

- ***Almacenamiento no volátil,*** la base de datos guarda un conjunto de *datos*, que además de estar *relacionados entre si de manera estructurada*, deben poder quedar guardados de forma permanente, hasta que el usuario decida hacer alguna consulta o proceso con ellos.

- ***Redundancia controlada***, no debe haber duplicidades perjudiciales en los datos, ni repeticiones innecesarias. En caso de existir redundancias físicas, por ser convenientes para agilizar algún proceso, éstas serán tratadas directamente por el sistema.

---

[7] Organización de las Bases de Datos, J. Martín, Prentice-Hall (1982).
[8] Conference des Statisticiens Européens, (1977).
[9] Fundamentos y modelos de bases de datos, autores Adoración de Miguel y Mario Piattini, Ed. Alfaomega.

La redundancia física es permitida, siempre que sea justificada por motivos de eficiencia. En consecuencia, un dato se actualizará *lógicamente* por el usuario de forma única, y el sistema se encargará de actualizarlo *físicamente* en todos aquellos campos en los que estuviese repetido, en caso de existir redundancia física.

Uno de los problemas graves que se tiene con el enfoque tradicional de archivos es la redundancia en la información, cuando una empresa decide almacenar cierto tipo de información para ser utilizada por una aplicación específica, es muy probable que dicha información sea tarde o temprano requerida para otra u otras aplicaciones. Usando el enfoque tradicional de sistemas de archivos, veríamos a cada aplicación como un proyecto separado, lo que seguramente nos llevará a duplicar la información en varios archivos, puesto que no hay una coordinación que nos permita identificar los requerimientos de información comunes entre las distintas aplicaciones; la redundancia resultará entonces en gastos de almacenamiento mayores a los necesarios.

A finales de los 60's el problema del almacenamiento era mucho mayor que ahora, incluso podemos suponer que hoy en día, el almacenar grandes cantidades de información no representa un problema serio. Sin embargo existe un problema mucho más grave aún con los sistemas de archivos, este problema es que la redundancia introduce la posibilidad de inconsistencia, esto sucede cuando un dato es actualizado en ciertos archivos pero no en otros.

Si los datos son actualizados y almacenados por separado, ¿cómo garantizar que esa información es confiable?, Si existe alguna modificación en un archivo, ¿cómo garantizo que todos los sistemas que emplean esa información se enteren de los cambios y los lleven a cabo?

Por tanto, no debe existir redundancia lógica, esto con la finalidad de evitar inconsistencias en el sistema.

- Otro aspecto importante en las bases de datos es la *independencia* tanto física como lógica entre datos y tratamientos entre los mismos. De ésta manera pueden servir a diferentes aplicaciones, esto es lo que las hace diferentes con respecto de los archivos, pues los archivos son creados para una aplicación en específico, mientras que las bases de datos pueden ser usadas por *múltiples aplicaciones* y *múltiples usuarios*.

El diseño de un sistema tradicional de archivos parte generalmente de la suposición de que los datos serán empleados de ciertas formas y no de otras, esas suposiciones se hacen pensando en los requerimientos iniciales para lo que es diseñado ese sistema, pero también se harán supuestos basados en la forma como la aplicación organizará físicamente los datos. Por ejemplo, la lógica de búsqueda de un registro no es la misma si el archivo está almacenado en forma secuencial o accesible mediante otro archivo indexado.

La dificultad derivada de éste modelo, es que los sistemas se vuelven altamente sensibles a los cambios en su entorno, si hay que cambiar la forma de organización en los archivos o cambiar el tamaño de los campos, el programa de aplicación será afectado y en algunos casos de forma dramática.

Para tratar el problema de dependencia de datos, la tecnología de bases de datos propone tomar toda la información que es relevante para el funcionamiento de una empresa y colocarla de forma organizada y codificada en un solo depósito llamado base de datos, ningún programa tendrá acceso directo a los datos contenidos en la base, tendrán que solicitar la información a través de un nuevo nivel de software llamado Database Management System (DBMS).

El DBMS (traducido al español como sistema de gestión de bases de datos SGBD) provee acceso a la información mediante un alto nivel de abstracción, permitiendo una total independencia entre los datos y la aplicación.

- La ***descripción*** o ***definición*** del conjunto de datos contenidos en la base (llamada también esquema de la base de datos o diccionario de datos) debe ser *única* y estar integrada en los mismos datos.
  Por ejemplo, en la misma base de datos, debe estar documentado el tipo de datos que usarán los campos, si algún tipo cambia, la base de datos almacenará el cambio dentro de la misma base, de manera que los datos estén *autodocumentados.*

- La **actualización y recuperación** de los datos debe realizarse **mediante procesos bien determinados** e incluidos en el *sistema gestor de base de datos*, el cuál debe contener mecanismos que faciliten la disponibilidad, actualización y confidencialidad del conjunto de datos.

1.6 Sistemas de Base de datos

Los Database Systems o Sistemas de Base de Datos (como se ha traducido al español), son el modelo que precede al tradicional enfoque de los sistemas de archivos, y su principal objetivo es lograr la independencia entre estructuras lógicas y físicas.

Un sistema de base de datos está constituido por el Database Management System, que es el conjunto de programas que permiten la implantación, acceso y mantenimiento a la base de datos, los usuarios, programas de aplicación y por supuesto la base de datos

Existen varias características que distinguen al enfoque de los sistemas de base de datos con respecto de los sistemas de archivos; en el procesamiento de archivos tradicional, cada usuario define e implementa los archivos requeridos para una aplicación, aún cuando dos usuarios empleen la misma información, cada uno mantiene sus archivos por separado

y cada uno tiene un programa especial para manipular sus archivos, porque requieren datos que no se pueden obtener de los archivos del otro, o bien porque el tratamiento entre archivos fue diseñado de manera distinta. Esta redundancia es reducida al máximo empleando bases de datos, pues se tiene un solo contenedor para almacenar los datos y que se define una sola vez. Ese repositorio de información, sirve para todos los usuarios que requieran tener acceso a el.

Otra característica que distingue al enfoque de los sistemas de base de datos con respecto al del tratamiento de archivos, es la naturaleza autodescriptiva de las bases de datos, ya que no solo se almacena la base de datos misma, sino que también una definición completa de la estructura de la base de datos. Esta definición, denominada esquema de la base de datos, diccionario de datos o metadato, almacena información sobre el tipo y formato de almacenamiento de cada elemento en la base, así como de las restricciones que se aplican a los datos.

El diccionario de datos es consultado por el DBMS[10] y hasta por los mismos usuarios cada vez que se necesite información de cómo está estructurada la base de datos.

1.7 Personajes en un ambiente de bases de datos

En una base de datos pequeña, lo común es que una sola persona la diseñe, construya y administre. En cambio, en una base de datos grande que proveerá de información a cientos de usuarios, es de suponer que muchas personas intervendrán en el diseño, uso y mantenimiento.

Los actores que forman parte del entorno del sistema de base de datos se clasifican de acuerdo al interés que tengan sobre la base y a las acciones que realicen para mantener funcionando la base de datos.

1.7.1 Administrador de la base de datos (DBA)

En un entorno de base de datos, el recurso primario es la propia base de datos, el recurso secundario es el DBMS junto con todo el software relacionado con él, y es necesario que haya una persona encargada de administrar esos recursos, tal responsabilidad recae sobre el administrador de la base de datos (DBA: database administrator). El DBA se encarga de mantener la armonía en el sistema de base de datos, el autoriza los permisos de acceso a la base de datos, coordina y vigila el empleo de la misma, también es quién se encarga del proceso de tuning o afinación de la base de datos, esta acción es para optimizar la respuesta del sistema, otra responsabilidad importante del DBA es resolver los problemas de violación a la seguridad del sistema.

En una organización grande, el DBA cuenta con apoyo de personal que le ayuda a desempeñar estas funciones, pero la responsabilidad de mantener a salvo la base de datos y

[10] Database Management System, software que permite manipular los datos almacenados en la base de datos.

de que siempre esté disponible para el buen funcionamiento del sistema recae totalmente sobre la figura del DBA.

El DBA es el usuario con todos los privilegios dentro de la base de datos, en él se centraliza el control de la base de datos. A continuación se describen con más detalle las funciones que desempeña un administrador de base de datos.

- **Definición del esquema de la base de datos,** esta tarea se lleva a cabo antes de implementar físicamente la base de datos y tiene como finalidad el diseñar la estructura lógica de la base, aún cuando el DBA no sabe exactamente como van a funcionar los programas que solicitarán información a la base, trata de que las estructuras que seleccione satisfagan los requerimientos presentes y futuros, dependiendo de la complejidad del sistema, el DBA puede desarrollar éste análisis conjuntamente con un grupo de personas a los que se les llama diseñadores de base de datos, y entre todos aplicar modelos destinados a ayudar en el proceso de diseño de bases de datos.

- **Definición de las estructuras de almacenamiento,** buscando siempre el mejor desempeño en la base de datos, el DBA debe de elegir entre la variedad de estructuras físicas para almacenamiento provistas por el software encargado del manejo de la base de datos. Por ejemplo, el DBA debe de saber elegir cuando usar índices en vez de búsquedas secuénciales para operaciones de consulta y/o actualización.

- **Modificación de estructuras Físicas y de almacenamiento,** el entorno para el que sirve el sistema de bases de datos es dinámico, por lo que es de esperarse futuros cambios, mismos que en lo posible deben plantearse a la hora del diseño, de llevarse a cabo cambios a la estructura lógica o física de la base, serán hechos por el DBA, por supuesto con previo análisis de dichos cambios tanto por él como por los diseñadores de la base de datos.

- **Autorización de acceso a la información,** como se dijo con anterioridad, la filosofía de las bases de datos es la de compartir datos entre múltiples usuarios, esos datos en el mundo real representan hechos conocidos que se traducen en información, gracias a las bases de datos, la información ahora tiene un alto grado de disponibilidad, pero no por eso significa que vamos a permitir a todos los usuarios de la base tener acceso total a los datos, hay que darle entonces a cada usuario, solamente la parte de información que le corresponde, esta es una tarea más para el DBA, quién debe de velar porque la información sea tratada solo por los usuarios correctos, coordinando y vigilando el buen empleo de la información.

- **Salvaguardar la integridad de la información,** esta tal vez sea la tarea más difícil para el DBA, porque es responsable cuando surgen problemas como violaciones a la seguridad y pérdida parcial o total de la información. El DBA debe implantar esquemas de recuperación y respaldo de información, preparándose para enfrentar cualquier tipo de eventualidad. Algo muy importante es que un buen DBA más que resolver muchos problemas, trabaja para prevenirlos.

1.7.2 Diseñadores de bases de datos

El trabajo de los diseñadores de bases de datos es identificar los datos que se almacenarán en la base y elegir cuales son las estructuras apropiadas para representar y almacenar dichos datos. Una base de datos actúa como un modelo de la realidad, su objetivo es relacionar hechos y situaciones, los datos que arroja la base se traducen en información que refleja la realidad de una empresa o institución para la cuál sirve el sistema. Por lo tanto, la tarea del diseñador de bases de datos no es nada fácil, de hecho las tareas de diseño y modelado de la base se deben hacer antes de que se implemente la base de datos, y una de las responsabilidades de los diseñadores es la de entrevistarse con los futuros usuarios de la base, a fin de comprender sus necesidades y presentar un diseño que satisfaga sus requerimientos.

1.7.3 Usuarios finales

Las bases de datos y los programas que con ella interactúan, solo son piezas del rompecabezas que forman los sistemas de bases de datos, que por supuesto no tendrían ningún sentido sin la parte principal que son los usuarios, cada vez que se diseña un sistema de bases de datos, se piensa en ¿quién lo va a utilizar?, una base de datos existe primordialmente para que alguien la use. Los usuarios finales son las personas que consultaran la base de datos, actualizaran la información y generarán reportes a partir de los datos. Según la frecuencia y la forma en que accedan a la base. a los usuarios finales se les puede clasificar de la siguiente forma:

- Usuarios finales simples: Son personas cuya función principal radica en consultas y actualizaciones a la base de datos empleando para ello operaciones previamente definidas, estas operaciones son conocidas como transacciones programadas y cada una ha sido definida con mucho cuidado. Este tipo de usuarios son la mayoría entre la población que requiere interactuar con la base de datos, como ejemplos de estos usuarios tenemos a los encargados de hacer reservaciones en líneas aéreas o los cajeros que hacen transacciones bancarias.

- Usuarios finales esporádicos: Son quienes requieren acceso a la base solo de vez en cuando y tal vez necesitando información diferente en cada ocasión. Para ello emplean un lenguaje avanzado para consulta, SQL en el caso de una base de datos relacional.

- Usuarios finales avanzados: Son aquellos usuarios que conocen ampliamente los recursos que les brinda su manejador de bases de datos y hacen uso de ellos para satisfacer sus complejos requerimientos.

- Usuarios finales autónomos: Estos usuarios emplean bases de datos personalizadas, usando software sencillo de utilizar y basado en menús e interfaces gráficas.

  Los usuarios autónomos generalmente adquieren una gran destreza en el manejo de la aplicación que les ayuda a guardar su información en una no muy compleja base de datos.

1.8 Niveles de abstracción en una base de datos.

Una característica fundamental en el enfoque de las bases de datos, es que proporciona un nivel de abstracción en los datos que permite ocultar los detalles de almacenamiento y recuperación de la información, mismos que no les son necesarios conocer a la mayoría de los usuarios. Dicha abstracción es presentada en forma de modelos de datos, estos modelos son conceptos que sirven para describir la estructura[11] de una base de datos. En todo modelo de datos es importante distinguir entre la descripción de la base de datos y la base de datos misma. La descripción de la base de datos es conocida como esquema de la base de datos (o metadatos) y es especificada durante el diseño de la base.

Existen tres formas de abstracción en un sistema de base de datos, es decir, tres maneras de estructurar un sistema de base de datos.

- **Estructura lógica del usuario,** diseña la BD desde el punto de vista del usuario.
- **Estructura lógica global**, conceptualiza la BD desde un punto de vista general, es decir, pensando en la interrelación usuario/sistema.
- **Estructura física,** analiza la forma en que serán almacenados los datos, pero físicamente.

La estructuración de una BD en estos tres niveles, tiene la finalidad de conseguir la independencia entre datos y aplicaciones, el manejo de múltiples vistas de usuarios y el empleo de un catálogo para almacenar la descripción (esquema) de la base de datos.

Esta arquitectura a tres esquemas es propuesta por el comité ANSI[12] / SPARC[13] como una alternativa de estandarización para los sistemas de gestión de bases de bases (SGBD), cabe señalar que los tres esquemas no son más que descripciones de los datos, los únicos datos que existen realmente están a un nivel físico, en un SGBD basado en la arquitectura a tres capas, cada grupo de usuarios hace referencia exclusivamente a su

[11] El concepto de estructura de una base de datos hace referencia a los vínculos y las restricciones que deben de cumplirse para esos datos.
[12] *American National Standards Institute*
[13] *Standard Planning and Requirements Committee*

propio esquema externo, por lo tanto el SGBD debe de transformar una petición basada en términos de un esquema externo a una petición basada en términos del esquema conceptual para finalmente expresarla en términos del esquema interno que se procesará sobre la base de datos (Véase figura 1.4).

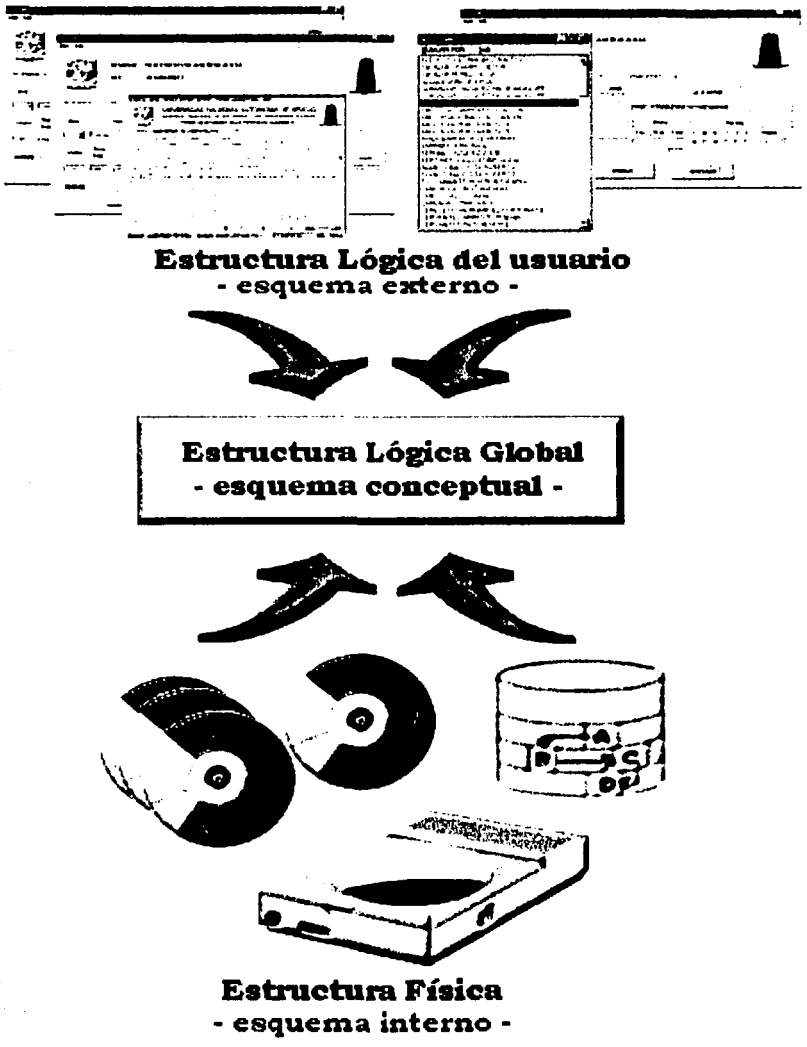


Figura 1.4 Las tres estructuras de los sistemas de bases de datos

1.8.1 Estructura lógica del usuario (esquema externo)

Es la visión que tiene un usuario en particular de la base de datos, y en él deberán encontrarse reflejados sólo aquellos datos e interrelaciones que necesite el correspondiente usuario, poniendo especial interés en las restricciones de uso, como son el derecho de insertar y borrar o el acceso a los mismos datos. Los caminos de acceso a los datos serán solo a nivel interno y por lo tanto no visibles para el usuario. Existirán tantos esquemas externos como exijan las diferentes aplicaciones, aunque un mismo esquema puede ser utilizado en más de una aplicación.

1.8.2 Estructura lógica global (esquema conceptual)

Por ser la visión global de los datos, deberá incluir la descripción de todos ellos y las interrelaciones entre cada uno, así como las restricciones de integridad y de confidencialidad, procurando que un cambio en alguna de estas restricciones no conduzca a una nueva definición del esquema conceptual.

1.8.3 Estructura física (esquema interno)

Define las estrategias de almacenamiento, éstas estrategias incluyen la asignación de espacios de almacenamiento para los datos, también la estrategia de emplazamiento de datos utilizada en la optimización de tiempos de respuesta y espacio de memoria utilizada, además se contemplan los caminos de acceso a datos (especificación de claves, punteros e índices), técnicas de compresión de datos, codificado de datos y los controles de acceso que permiten definir las reglas para proteger la confidencialidad de la información. Generalmente, estas especificaciones y estrategias son manejadas por medio de software especial para el manejo y administración de bases de datos.

Además de los aspectos citados, hay que mencionar que el administrador de la base de datos habrá de especificar:

- Dispositivos de memoria: Tamaño de la página, número de páginas asignadas a cada área de almacenamiento y tamaño de las áreas de entrada/salida (buffers).
- Correspondencia entre esquemas (mapping): Por omisión, se suele suponer que existe una correspondencia uno a uno entre los registros del esquema conceptual y los registros almacenados, en caso contrario el administrador debe indicar la relación existente entre ellos.
- Organizaciones físicas: Para mejorar la recuperación y los tiempos de acceso, el sistema debe dar facilidades para que el administrador defina el tipo de organización (dispersión, agrupamientos, índices, etc.) dependiendo del sistema de gestión de base de datos podrá también definir punteros entre registros, privilegiando así determinados caminos de acceso.

El tema de la arquitectura a tres esquemas será tratado con mayor profundidad en éste capítulo, en la parte concerniente a la estandarización de la arquitectura de los SGBD, sin embargo, he querido dar un primer acercamiento a los distintos niveles de abstracción en las bases de datos con la finalidad de que haya mayor entendimiento al explicar el concepto de independencia con respecto a los datos que expongo a continuación.

1.9 Independencia con respecto a los datos

La independencia con respecto de los datos es el objetivo primordial que se persigue con la arquitectura a tres esquemas y se puede definir como la capacidad de modificar el esquema en un nivel sin tener que modificar el esquema del nivel inmediato superior.
Por ejemplo, es posible modificar el esquema externo, digamos cambiando la interfaz del usuario sin tener por ello que alterar en lo absoluto el esquema conceptual.

La independencia con respecto a los datos puede dividirse en dos tipos, la independencia lógica con respecto a los datos y la independencia física con respecto a los datos.

La Independencia lógica con respecto a los datos, es la capacidad para modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación.

Por ejemplo, podemos modificar el esquema conceptual de la base al agregar un nuevo tipo de registro o al reducir la base de datos eliminando un tipo de registro y si el SGBD cuenta con independencia lógica respecto de los datos, solo será necesario modificar la definición de las vistas y las correspondencias, pero los programas de aplicación que hagan referencia a los elementos del esquema externo, deberán funcionar igual que antes de hacer las modificaciones.

La independencia física con respecto a los datos es la capacidad para modificar el esquema interno sin necesidad de alterar el esquema conceptual o alguno de los esquemas externos. Puede suceder que con el propósito de mejorar el rendimiento de la base de datos, algunos archivos sean reorganizados de manera física, un ejemplo concreto puede ser la creación de índices o de cualquier otra estructura de acceso adicional. Si la base aún contiene los mismos datos, no será necesario modificar el esquema conceptual.

Una arquitectura a tres niveles facilita en gran medida el poder lograr la independencia tanto lógica como física entre los datos, pero esta arquitectura tiene también su desventaja, porque la correspondencia entre niveles implica una serie de operaciones extras a la hora de las consultas a la base de datos, y que finalmente afectan la eficiencia del SGBD, sin embargo al analizar los beneficios que trae consigo esta arquitectura, la desventaja con respecto a la eficiencia no resulta tan grande, aunque no por ello despreciable.

1.10 SISTEMAS DE GESTIÓN DE BASES DE DATOS (SGBD)

Hasta este momento, se ha hecho referencia a las bases de datos como información almacenada que cumple toda una serie de características y restricciones, también se dijo que es uno de los componentes principales en un sistema de base de datos.

Pero para que la información pueda ser almacenada como se ha descrito y el acceso a la misma sea de forma satisfactoria, es necesario que existan una serie de procedimientos que garanticen las características que hasta ahora se le han atribuido a las bases de datos, estas acciones son llevadas a cabo por medio de un conjunto de programas, a los cuales se les denomina Database Management System (DBMS), traducido al español como Sistema de Gestión de Bases de Datos (SGBD), en lo sucesivo me referiré a este concepto por su siglas en español.

Dicho lo anterior, es posible definir a un SGBD como: una colección de programas de aplicación que proporcionan al usuario de la base de datos, los medios necesarios para realizar las siguientes tareas:

- Definición de los datos a los distintos niveles de abstracción (lógico, conceptual y físico).
- Manipulación de los datos en la base de datos. Es decir, la inserción, modificación, borrado y acceso o consulta de los mismos.
- Mantenimiento de la integridad[14] de la base de datos. Integridad en cuanto a los datos en sí, valores y las relaciones entre ellos.
- Control de la privacidad o seguridad de los datos en la base de datos.
- Y en general, los medios necesarios para el establecimiento de todas aquellas características exigibles a una base de datos.

El SGBD, junto con la base de datos y los usuarios, conforma lo que se conoce como sistema de base de datos, destinaré lo que resta de éste capítulo para describir las características del software generalizado para implementar y mantener una base de datos (Véase figura 1.5).

[14] La Integridad se refiere a que los datos deben de ser coherentes consigo mismos y consistentes con las reglas semánticas propias del mundo real al que han de representar lo más fielmente posible.

(*Figura 1.5 Elementos que conforman a un* ***Sistema de Base de Datos.***)

1.10.1 Componentes de los SGBD

El sistema de gestión de base de datos, está compuesto por una serie de elementos que en conjunto realizan todas las tareas mencionadas con anterioridad, y algunas otras más.

1.10.1.1 El Lenguaje de definición de datos

El lenguaje de definición de datos - *Data Definition Language* (DDL) -, es un lenguaje artificial, basado en un determinado modelo de datos que permite la representación lógica de los mismos, pues como dijimos, se necesita definir la base de datos desde el punto de vista del usuario, del almacenamiento físico y de un esquema global de la base de datos. Estas definiciones se realizan por medio de una serie reducida de morfemas simples, basados en una gramática sencilla que garantice que estas expresiones estén libres de ambigüedades. Las definiciones deben ser compiladas para dar una representación que pueda entender la computadora y que es la que utiliza el SGBD en tiempo de ejecución.

1.10.1.2 El lenguaje de definición del almacenamiento de los datos

Haciendo uso del lenguaje de definición del almacenamiento de datos – *Storage Definition Languaje* (DSDL) – es como describimos el nivel de abstracción lógico (cómo son manipulados los datos por la computadora) y el nivel físico (esquema interno).

Las características que definimos a un nivel de descripción lógico son:

- Cada una de las clases de objetos que formen parte del SGBD y sus propiedades.
- Cada una de las relaciones que existen entre los objetos.
- Todas las restricciones en los objetos y las restricciones en las relaciones.

Las características que definimos desde un punto de vista físico u operacional son:

- Las unidades físicas en donde serán almacenados los datos.
- Los archivos utilizados y los volúmenes de los mismos.
- Las características físicas y lógicas de los medios de almacenamiento.
- Los métodos de acceso a los datos (índices, tablas, etc.).

1.10.1.3 El lenguaje de manipulación de datos

Este es un lenguaje artificial que nos permite manipular los datos a nivel externo o de usuario de los datos. El término manipulación de datos se refiere a insertar, borrar, modificar y recuperar los datos almacenados en la base de datos. Al igual que el DDL[15] y el DSDL[16], el lenguaje de manipulación de datos – Data Control Lenguaje (DCL) – está basado en un modelo de datos (relacional, jerárquico, orientado a objetos, etc.). La gramática empleada por el DCL debe ser completa, sencilla y fácil de entender por usuarios no expertos.

Como se comentó con anterioridad el DCL tiene también la tarea de describir las vistas[17]externas de los datos.

Estas vistas pueden mostrase de dos formas:

- La primera: Haciendo uso exclusivamente del lenguaje de manipulación de datos, empleando las sentencias orientadas al usuario final.

- La segunda: Cuando se requieren medios más potentes y flexibles para definir y administrar los datos, se suele recurrir a lenguajes de programación tales como: "C", Pascal, Basic, etc. De esta manera, el lenguaje de programación, funge como lenguaje anfitrión y desde ahí se pueden mandar llamar las instrucciones del lenguaje de manipulación de datos, así que el DCL se convierte ahora en lenguaje huésped del lenguaje de programación.

1.10.1.4 El diccionario de datos

El diccionario de datos es un archivo o un conjunto de archivos en donde se almacena información acerca de la base de datos. Esta información hace referencia al tipo de datos que se pueden almacenar en la base, así como el esquema lógico y el esquema físico definido por el lenguaje de definición de datos. Es decir, los niveles de abstracción en la base de datos.

---

[15] Lenguaje de definición de datos *Data (Definition Language)*
[16] Lenguaje de definición del almacenamiento de datos *(Storage Definition Languaje)*
[17] Una vista es una visión parcial que los usuarios tienen de los datos almacenados en la base de datos.

Pero además de esta información, en el diccionario de datos se almacenan también las restricciones de acceso a los datos y los mecanismos que servirán para garantizar que dichas restricciones sean respetadas.

1.10.1.5 El gestor de la base de datos

El gestor de la base de datos también llamado monitor, es un componente software que se encarga de vigilar que el acceso a la base de datos se haga en forma segura y eficiente, además del íntegro almacenamiento de los datos.

Este componente sirve de interfaz entre los datos almacenados y los programas de aplicación que los utilizan. Toda operación que se quiera llevar a cabo directamente con la base de datos, deberá ser aprobada primero por el gestor de la base de datos, una vez interpretada y aprobada, se realiza la operación y se devuelve el resultado al programa o aplicación que lo solicitó, de no ser aprobada se rechaza la operación.

Como es el usuario quien solicita por medio de los programas de aplicación, acceso a la información contenida en la base de datos, entonces podemos visualizar al gestor de la base de datos como un intérprete entre el usuario y los datos. Así pues, el gestor de la base de datos es responsable de:

- Garantizar la confidencialidad en los datos permitiendo solo el acceso a usuarios autorizados.
- Proteger la integridad en los datos, vigilando que los mismos sean almacenados conforme a las restricciones definidas según el esquema de la base de datos.
- Asegurar el acceso concurrente a la base de datos, de modo que varios usuarios puedan acceder a los mismos datos sin que ello implique una pérdida en la integridad de estos.
- Garantizar la seguridad de los datos con procedimientos que permitan su recuperación tras un fallo que ocasione una pérdida o deterioro de los mismos.

1.10.1.6 El administrador de la base de datos

Es una persona encargada de resolver problemas relacionados con la base de datos, el administrador de la base de datos tendrá la responsabilidad de administrar, asegurar y vigilar la integridad de la información, así como de supervisar el desempeño del SGBD en el tratamiento de la misma.

1.10.2 EL SGBD COMO INTERFAZ ENTRE EL USUARIO Y LA BASE DE DATOS

Como se ha dicho anteriormente, son tres los niveles de gestión o administración que se distinguen en una empresa, estos niveles jerárquicos, buscan como objetivo común alcanzar las metas fijadas por los directivos de dicha empresa, la información que se utiliza para la toma de decisiones, debe estar contenida en una base común, siendo ésta el único depósito de datos para toda la organización, el SGBD debe ser capaz de integrar a todos los niveles atendiendo a las necesidades individuales de cada uno de ellos, pues es éste, quien suministra la interfaz entre los usuarios que componen los tres niveles de administración y la base de datos. (Véase figura 2.1).

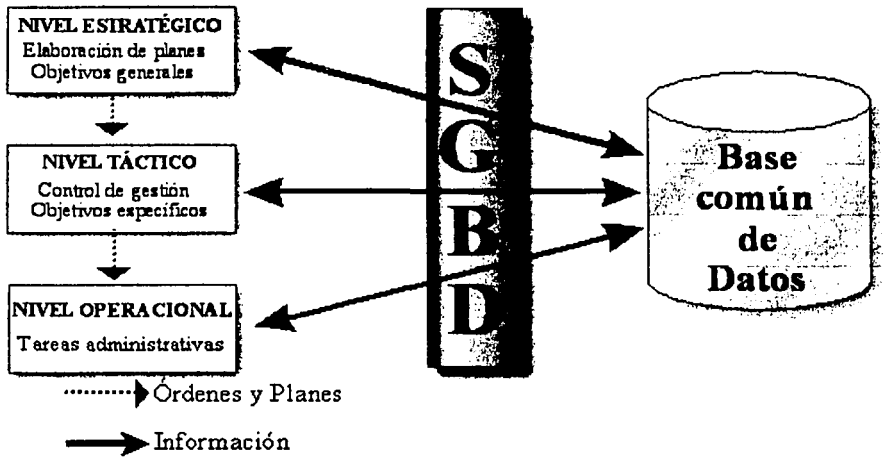


Figura 2.1 El SGBD como interfaz entre la base de datos y los niveles de gestión de la organización.

Si se tiene en cuenta la diversidad de usuarios que harán uso del sistema, todos con múltiples necesidades (que incluso pueden variar a lo largo del tiempo), entonces cobra una gran importancia dotar de una adecuada flexibilidad al SGBD, para que de ésta manera, pueda atender todas las exigencias de los usuarios y sea capaz de adaptarse a los distintos cambios que se puedan suscitar.

Las operaciones típicas que debe realizar un SGBD pueden resumirse en dos conjuntos, el que afecta a la totalidad de los datos y el que afecta a registros específicos.

Sobre la totalidad de los datos

- Creación
- Reestructuración
- Consulta a la totalidad

Sobre registros concretos

- Inserción
- Borrado
- Modificación
- Consulta selectiva

Los usuarios de la base de datos, ya sean el diseñador de la base, el administrador o el usuario final, deben contar con mecanismos que les permitan trabajar con la base de datos al nivel que cada uno requiera.

Al usuario final que consulta o actualiza la base por medio de un lenguaje anfitrión, no le incumbe la descripción física de los datos ni la descripción lógica global. De hecho, no se le debe permitir que tenga acceso a éstas, limitándolo sólo a las necesidades de información especificadas de antemano.

El administrador de la base, debe disponer de instrumentos que le ofrezcan facilidades para la creación, reorganización y respaldos de la información, mediante un conjunto de procedimientos que varían dependiendo del SGBD en concreto.

1.10.3 LA ESTANDARIZACIÓN DE LA ARQUITECTURA DE LOS SGBD

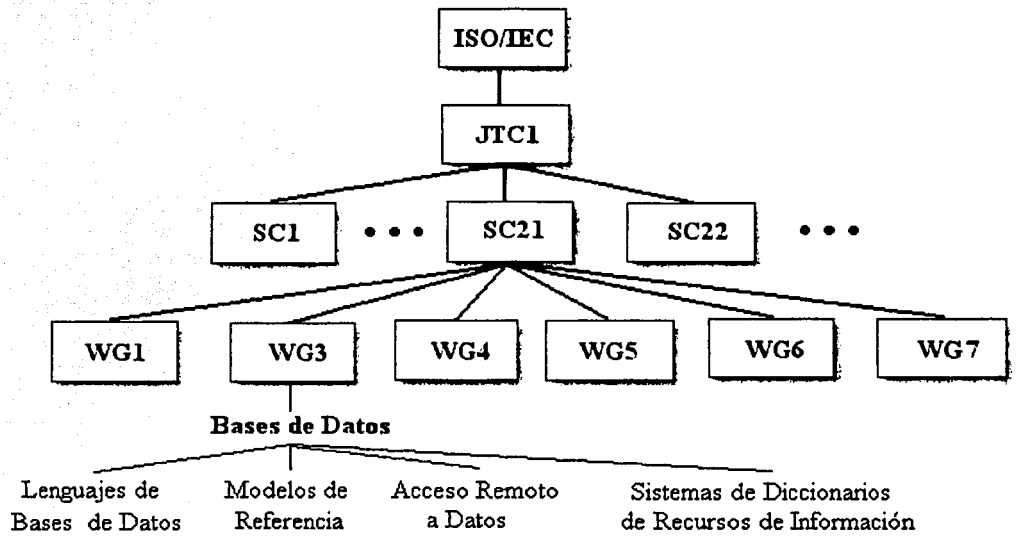
La estandarización de los SGBD nos permite que una vez que hayamos terminado un sistema de base de datos e implementado un SGBD en específico, podamos cambiar a otro SGBD comercial sin mayor problema, es decir, sin tener que volver a diseñar de nueva cuenta la base de datos ni las aplicaciones que acceden a esta.

La estandarización tiene como objetivo proteger las inversiones de las empresas, pero toda estandarización debe realizarse con suma cautela, puesto que una fijación prematura de reglas, puede coartar el advenimiento de nuevas tecnologías y limitar un desarrollo tecnológico, y en el caso del área informática es preciso continuar con la inercia tecnológica que cada vez parece cobrar mayor fuerza.

Esto parece situarnos en un gran dilema, sin embargo, algunas instituciones como ANSI/X3/APARC, e ISO creen que la tecnología en bases de datos está lo suficientemente

madura como para admitir una estandarización, misma que se ha implementado con éxito, aún cuando no ha sido aceptada por toda la comunidad informática.

Los organismos de estandarización ISO (Intenational Organization for Standarization) e IEC (International Electrotechnical Commision) han establecido una serie de comités encargados de las tecnologías de información, entre éstos podemos destacar al subcomité WG3 dedicado a las bases de datos.



Estos grupos de trabajo son integrados por representantes de organismos oficiales de estandarización de distintos países y se dedican a los siguientes proyectos principalmente:

- **Lenguajes de Bases de Datos:** Este proyecto es encargado del lenguaje para bases de datos en red (NDL) y del SQL para bases de datos relacionales. También trabajan en proyectos para la estandarización de bases de datos multimedia conocidos como SQL/MM.

- **Modelos de Referencia:** Proveen de un conjunto de procedimientos, interfaces y funciones que sirven de referencia para el desarrollo de los sistemas de bases de datos, con la finalidad de coordinar una estandarización en este campo.

- **Acceso Remoto a Datos (RDA):** En éste proyecto ponen atención a los servicios y protocolos de comunicación para el correcto acceso de los clientes al servidor. Por tanto estudian también el comportamiento de los servidores de base de datos y la forma en que almacenan y procesan información.

1.10.3.1 La Arquitectura ANSI/X3/SPARC

El grupo ANSI/X3/SPARC es un grupo de estudio del *Standard Planning and Requirements Committee* (SPARC) que forma parte del ANSI (*American National Standards Institute*) y que propone un esquema conceptual en el que se distinguen tres niveles de abstracción de datos (externo, conceptual e interno) del cuál derivan una colección de esquemas externos, éstos esquemas definen la forma en como cada aplicación debe ver a la base de datos. La transformación de unos esquemas en otros (función de correspondencia) se lleva a cabo por el SGBD.

La figura 2.2 muestra gráficamente los tres niveles de abstracción de datos junto con los tres esquemas que componen la arquitectura ANSI.

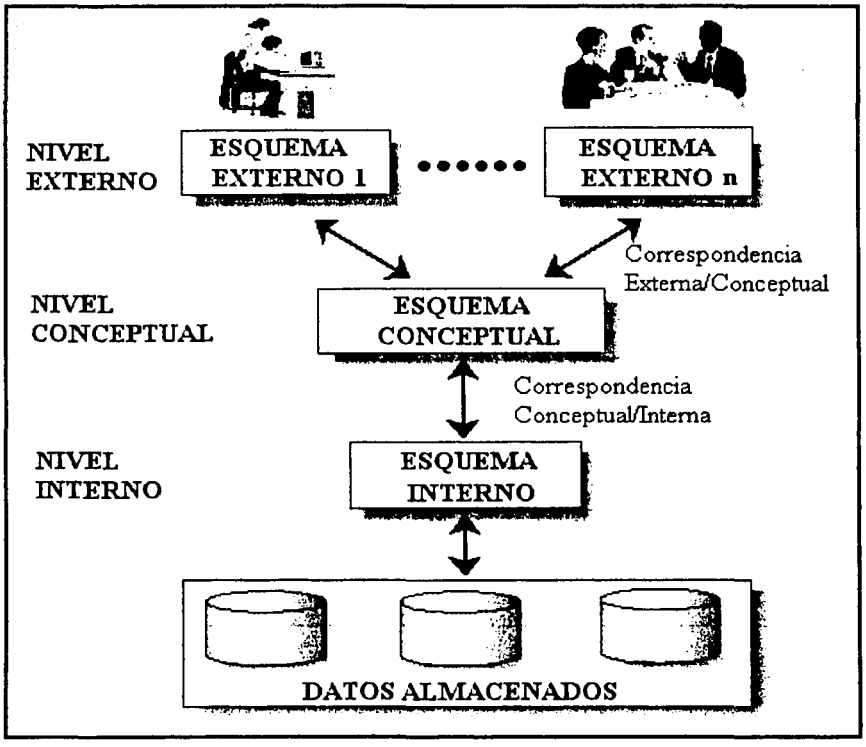


Figura 2.2 Los tres niveles de abstracción de los datos y la arquitectura triesquemática de ANSI

Esta arquitectura propone que los datos pasen a través de distintas capas separadas por interfaces, la razón de emplear distintas capas, es aislar los diversos componentes del sistema logrando así independencia entre ellos.

El usuario podrá de esta manera manipular los datos, entendiendo por manipulación la inserción, modificación, consultas y el borrado de los mismos, utilizando una interfaz que podría ser un lenguaje de manipulación, por ejemplo SQL. La petición hecha por el usuario es ejecutada por los transformadores externo/conceptual, conceptual/interno e interno/almacenado, la acción que llevan a cabo los transformadores es convertir la petición en un formato adecuado para que las interfaces la procesen y devuelvan el resultado que será entregado al usuario.

Todas las interfaces las ha de suministrar el SGBD, pero ANSI no especifica la forma de implementación, por lo que queda a consideración del diseñador del sistema, lo que aumenta la flexibilidad y capacidad de evolución de éste esquema, al cuál se le pueden hacer los cambios necesarios (aunque ANSI recomienda que no sean muy drásticos) para adaptarlo al sistema de información de la empresa en la que se quiera implementar.

Otra ventaja de la independencia entre funciones es el poder introducir cambios en un esquema sin que afecte a los demás. Por ejemplo: Proporcionar nuevas funciones de correspondencia sin que lo note el usuario, afectando solamente a los procesadores de información del SGBD que mediante las correspondientes interfaces tendrán que modificar las funciones de correspondencia adaptándolas a las nuevas circunstancias. De igual forma es posible añadir nuevas vistas externas que le permitan al usuario satisfacer nuevas exigencias, lo que ayuda a una mejor administración de los recursos humanos, preservando las inversiones realizadas.

Esta primera parte, en la que se incluyen los conceptos básicos sobre Sistemas de Información, Bases de Datos y Sistemas Gestores de Bases de Batos, aporta la información necesaria para conocer el papel que juega cada uno de ellos en el ciclo básico de la información, que como mencioné, es el eslabón que une todos los elementos de una organización.

Por otra parte, tanto a los Sistemas de Información, las Bases de Datos y los SGBD, no se les debe concebir como unidades independientes, pues es en la interacción entre ellos, donde cobra significado la razón de ser de cada uno, de ahí el porque he querido manejarlos juntos en el primer capítulo, además de ser conceptos que se tocarán constantemente a lo largo de este trabajo.

En el siguiente capítulo, expondré sobre el modelo de datos relacional, que es un conjunto de reglas y conceptos que nos permiten describir los datos desde un punto de vista conceptual, analizando los datos como un todo, y no como fracciones independientes.

CAPÍTULO 2. EL MODELO DE DATOS RELACIONAL

En el mundo real se presentan un gran número de problemas a resolver, para darles solución, el hombre cuenta con una gran herramienta: su capacidad de abstracción, mediante la cuál es capaz de simplificar el número de parámetros y acciones que afectan al problema que se quiere solucionar. A todos los parámetros o restricciones que intervienen en la solución de un problema se les llama datos, y al conjunto de todas las relaciones o independencias entre éstos se les denomina información. En un principio, los datos son organizados, de manera que atiendan las necesidades de un proceso, cuando se cumple éste objetivo se intenta que los datos respondan a los requerimientos de un conjunto de procesos, para finalmente buscar una interpretación del mundo real. El modelo de datos sirve de soporte para ésta interpretación, ofreciendo un conjunto de reglas por medio de las cuales podemos describir cualquier fenómeno real o abstracto.

2.1 CONCEPTO DE MODELO DE DATOS

Los modelos de datos son abstracciones mediante las cuales podemos realizar una representación de los problemas a resolver, nos permiten describir un problema y a sus parámetros como un todo, sin que sea necesario describir particularmente cada elemento, ni cada uno de los estados de los parámetros con relación del tiempo.

Los modelos son utilizados en todas las áreas del conocimiento para la representación de los problemas en estudio, particularmente en las áreas que tienen que ver con la computadora, han sido propuestos diversos modelos para la representación y solución de diferentes problemas, específicamente nos interesan aquellos que nos ayudan a describir en distintos niveles de abstracción la estructura de una base de datos, la cuál se denomina esquema.

De un modo más preciso se define a un modelo de datos como: "*un conjunto de conceptos, reglas y convenciones que nos permiten describir y manipular (consultar y actualizar) los datos de un cierto mundo real que deseamos almacenar en la base de datos*"[18].

Aunque cada modelo de datos soporta una teoría para la representación y tratamiento de un problema, hay características con las que en general cuentan todos los modelos:

- Los modelos de datos presentan un conjunto de reglas mediante las cuales se puede representar gráficamente el problema a tratar, por medio de símbolos, son representados los objetos del sistema y las diferentes relaciones o dependencias que existen entre ellos. Esta simbología también es capaz de mostrar las restricciones concernientes a cada objeto y sus relaciones.

[18] Fundamentos y modelos de bases de datos, autores Adoración de Miguel y Mario Piattini, Ed. Alfaomega.

- Un pseudo lenguaje formado por un conjunto reducido de morfemas, y una sintaxis plenamente establecida para fijar las características estáticas y dinámicas del sistema.

- Así como de un conjunto de restricciones que limitan las características de los sistemas a representar.

Como fue señalado cuando se definió el término base de datos, existen distintos niveles de abstracción en una base, es decir, distintas maneras de ver a la base de datos, desde el punto de vista lógico, físico y conceptual. Por ello, los modelos de datos dan la oportunidad de representar el problema en éstos tres niveles, idealmente de forma independiente, tanto en representación como en tratamiento. Estas tres representaciones las hacen de la siguiente manera:

Nivel Conceptual: Muestra desde un punto de vista conceptual, los tipos de objetos que contiene el sistema y las relaciones que existen entre ellos. Para cada uno de esos tipos de objetos son descritas sus características, es decir, sus propiedades y restricciones.

En este nivel, podemos percibir al problema tal y como es en el mundo real, percibiéndolo sin tener en cuenta cómo será representando para que sea entendido por la computadora.

Un modelo conceptual de un problema es independiente de los procesos que serán utilizados para el tratamiento y mantenimiento de la información, por lo que en este nivel la atención está centrada en las características del problema a representar.

Nivel Lógico: Mientras que en el nivel conceptual el problema se representa tal y como es captado desde el mundo real, en el nivel lógico esta representación es modificada para poder adaptarla a los procesos de almacenamiento y tratamiento de información inherentes al software y hardware que será utilizado.

Por ejemplo, el uso de un sistema de gestión de base de datos relacionales, implica una serie de reglas para la representación de un problema del mundo real muy diferentes a las que emplearíamos en caso de utilizar algún otro sistema de base de datos, como el sistema de base de datos Jerárquico o el de Red.

Nivel físico: Podemos pensar en este modelo como el punto de vista que tendría la máquina de un problema del mundo real, es decir, describe las acciones elementales que se deben realizar para representar el comportamiento de los objetos. Por ejemplo, el criterio de búsqueda que empleará el sistema para recuperación de información.

También son descritos en este nivel los formatos de los documentos a usar, ubicación y archivos en los que serán almacenados.

Los modelos físicos, también llamados internos, no están estandarizados y son propios de cada uno de los productos comerciales que empleemos para el tratamiento de la información.

2.2 EL MODELO DE DATOS RELACIONAL

El modelo de datos relacional surge por la necesidad de una representación clara y sencilla del problema a tratar, por lo que este modelo, origina esquemas más simples, fáciles de manejar y por lo tanto más seguros y confiables. El Modelo de datos relacional es propuesto por el Doctor Edgar F. Codd a finales de los años sesenta, su modelo está basado en la teoría de las relaciones y unos de sus principales objetivos, es la independencia de la representación lógica de los datos con respecto a su almacenamiento interno.

La estructura básica del modelo es la relación, todos los datos dentro de una base son representados en forma de relaciones cuyo contenido varía con el tiempo. Expresado en terminología relacional, una relación es un conjunto de filas, (a quién Codd les llama tuplas) y que tienen determinadas características. En forma matemática una relación se define como un conjunto.

Codd también introduce en su modelo una serie de operadores que actúan sobre las relaciones y que en conjunto, conforman el álgebra relacional definida formalmente por él mismo en 1972.

En su propuesta Codd incluye su teoría de la normalización, compuesta inicialmente por tres formas normales, que son reglas para ayudar en el diseño de bases de datos y su finalidad es eliminar dependencias entre atributos (columnas de la relación) que originan anomalías en la actualización de la base de datos, proporcionando así una estructura más regular en la representación de relaciones.

Las consideraciones primordiales en las que se basa este modelo son:

- Generar esquemas representando fielmente la a información, a los objetos y a las relaciones existentes entre ellos.
- La información debe ser fácilmente entendida, aún por usuarios con poca experiencia en esta área.
- Posibilitar la ampliación del esquema de la base de datos sin modificar la estructura lógica existente y en consecuencia, los programas de aplicación.
- Máxima flexibilidad en la formulación de interrogantes, previstas o no sobre la información contenida en la base de datos.

2.2.1 TERMINOLOGÍA DEL MODELO RELACIONAL

El modelo de datos relacional representa la base de datos como una colección de relaciones, en términos informales, podemos equiparar una relación con una tabla, por ser similares en cuanto a estructura tabular.

Todos los datos de una base de datos serán representados en forma de relaciones cuyo contenido varía con el tiempo. La tabla sirve para una representación tabular y plana de la información, en su composición bidimensional, son presentados los objetos y las relaciones entre ellos.

Una Relación (tabla) es una matriz rectangular en la que podemos distinguir su nombre, un conjunto de columnas cuyas cabeceras son denominadas atributos, que representan las propiedades de la tabla y que también están caracterizadas por su nombre, y un conjunto de renglones llamados tuplas, que contienen los valores que toma cada uno de los atributos para cada elemento de la relación. (Véase figura 2.1).

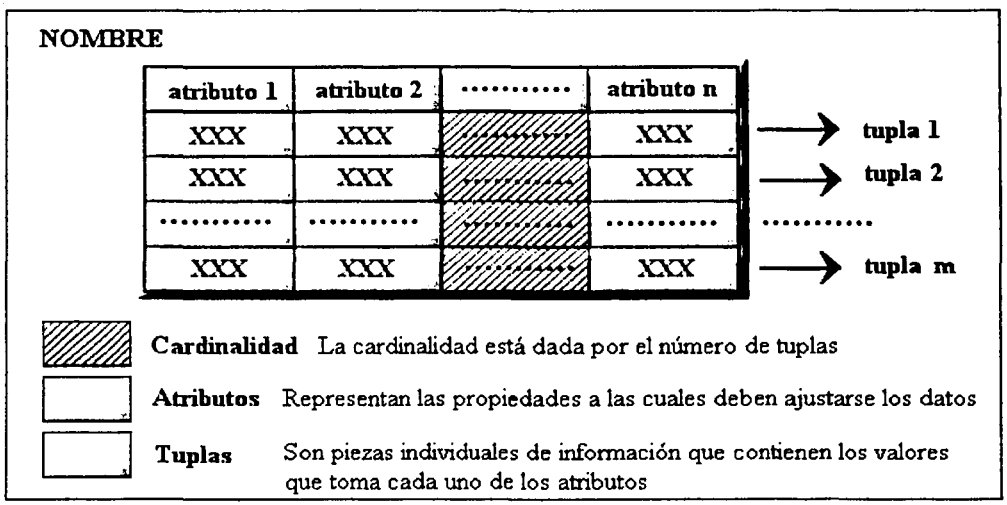


Figura 2.1 Representación de una relación en forma de tabla.

En la figura 2.2 se representa la relación LIBRO utilizando la estructura del modelo relacional, el nombre de la relación es LIBRO, los atributos son Titulo, Autor y Editorial; los dominios son de donde los atributos pueden tomar sus valores; las tuplas contienen los valores tomados por los atributos para cada determinado libro; el grado indica el número de atributos y la cardinalidad indica el número de tuplas en un momento dado.

Como podemos ver, una relación puede ser representada fácilmente en forma de tabla, aunque tiene unas características especiales que la distinguen de una tabla común, por ejemplo: No se admiten filas duplicadas, las filas y las columnas no están ordenadas y en el cruce de una fila con una columna solamente puede haber un valor, éstas son restricciones propias del modelo relacional.

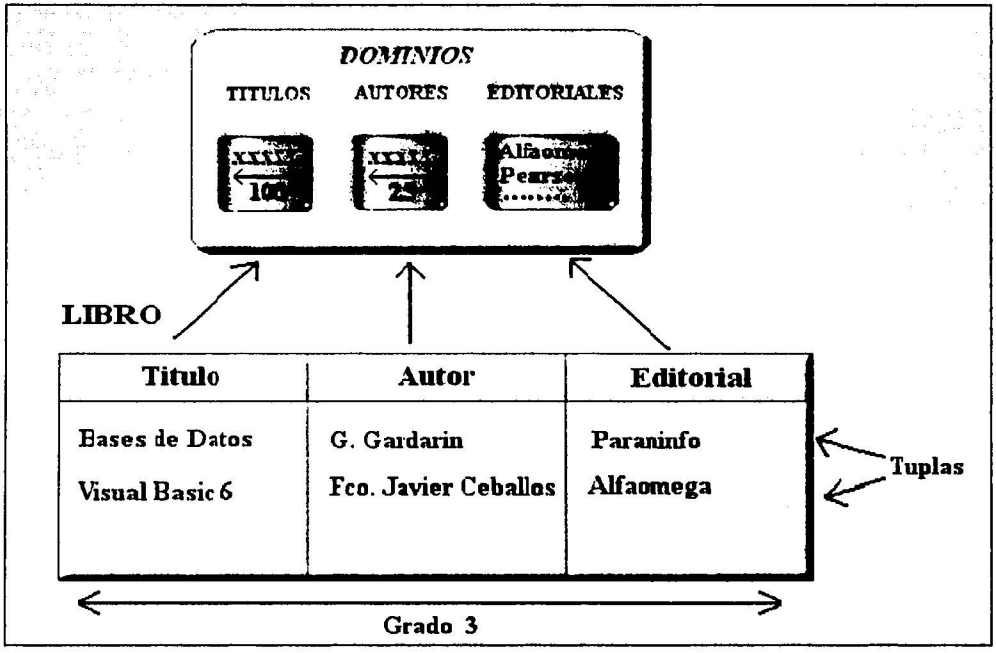


Figura 2.2 Representación de la relación LIBRO.

Esta representación es la razón del porqué habitualmente se utiliza el nombre de *tabla* para denominar a las *relaciones* y como consecuencia de ello se le llama *filas* a las *tuplas* y *columnas* a los *atributos*, aunque la terminología usada es irrelevante, ya que un modelo no dejará de ser relacional por utilizar una u otra terminología.

A continuación se presenta una comparación de la terminología relacional comparada con la usada en tablas y archivos.

RELACIÓN	*TABLA*	*ARCHIVO*
TUPLA	FILA	REGISTRO
ATRIBUTO	COLUMNA	CAMPO
GRADO	N° DE COLUMNAS	N° DE CAMPOS
CARDINALIDAD	N° DE FILAS	N° DE REGISTROS

2.2.2 Dominios

Un dominio D es un conjunto de valores atómicos, entendiéndose por atómico, en el sentido del modelo relacional, que cada valor del dominio es indivisible.

En términos más simples, un dominio se define como un conjunto de posibles valores que puede tomar cada atributo.

Un método común para especificar un dominio consiste en definir un tipo de datos al cuál pertenecen los valores que constituyen el dominio. También se suele especificar un nombre para el dominio que ayude a interpretar sus valores. Por ejemplo: El dominio del atributo *meses* es el conjunto del nombre de los doce meses del año.

Otros ejemplos de definiciones lógicas de dominios:

Numero_de_cuenta: El conjuntos de números de matrículas de alumnos formados por 10 dígitos.

Nombres: El conjunto de nombres de personas.

Materias: El conjunto de nombres de las materias impartidas en la Escuela Nacional de Estudios Profesionales Aragón.

También es posible asignar para cada dominio un formato y un tipo de datos, por ejemplo podemos decir que el dominio del atributo *edades* es el conjunto de los números enteros entre 16 y 80. Así pues, además de especificar el nombre del dominio, debe especificarse también un tipo de datos y un formato para los valores que contendrá dicho dominio.

2.2.3 Relaciones

Tomando el ejemplo de la tabla *Libro* mostrada en la figura 2.4, que tiene 3 atributos: Titulo, Autor y Editorial. Sea D_1 el dominio del atributo Titulo y D_2, D_3 los dominios de los atributos Autor y Editorial, entonces cada entrada a la tabla Libro, es decir cada tupla que insertemos se puede pensar como un elemento del producto cartesiano

$$D_1 \times D_2 \times D_3$$

Desde el punto de vista matemático, una relación sobre *n* dominios se define como un subconjunto del producto cartesiano de los dominios, es decir, como un conjunto de tuplas.

Existen algunas consideraciones que no debemos olvidar:

Primero: Por definición, sabemos que un conjunto no puede contener elementos duplicados, se dijo que una relación es un conjunto de tuplas, por lo tanto ninguna tupla aparecerá más de una vez en una relación.

Segundo: El orden del conjunto de los elementos es arbitrario, por lo tanto el orden en que las tuplas aparecen en una relación no está definido

Tercero: Todas las operaciones entre conjuntos, como: unión, intersección y diferencia son aplicables a las relaciones.

2.2.4 CLASES DE RELACIÓN

En principio las relaciones se clasifican en nominadas y sin nombre, las relaciones sin nombre se generan como resultado de consultas que no se materializan sino que se entregan al usuario cuando este ha realizado una consulta, los resultados que la relación sin nombre entrega pueden ser resultados intermedios o finales, de modo que siempre serán temporales.

Las relaciones nominadas, a su vez, se clasifican en Persistentes y Temporales;

Las relaciones persistentes son aquellas cuyo esquema de relación permanece en la base de datos, borrándose únicamente mediante acción explícita del usuario, en cambio las relaciones temporales desaparecen de la base de datos en cierto momento sin necesidad de una acción de borrado por parte del usuario.

Las relaciones persistentes se clasifican en:

- Relaciones base: Son las que existen por si mismas, sin depender de otras relaciones y se crean especificando explícitamente su esquema de relación, es decir, tanto el nombre de la relación, como el conjunto de atributos y dominios.

Hay que señalar que aunque en teoría, cuando se define una relación no sólo deben especificarse los nombres de los atributos, sino que también los de los dominios, en la práctica estos últimos suelen omitirse, debido a que la mayoría de los productos que se emplean para construir bases de datos relacionales no soportan ente concepto. Por supuesto deben de estar almacenadas también las relaciones entre las tablas.

- Vistas: Son relaciones derivadas de otras relaciones que se definen dando un nombre a una expresión de consulta. Se podría decir que son relaciones virtuales, en el sentido de que no almacenan datos, sino que almacenan su definición en términos de otras relaciones con nombre.

- Instantáneas: Son también llamadas *snapshot* y al igual que las vistas son relaciones derivadas que se definen en términos de otras relaciones nominadas, pero éstas si almacenan datos propios, que son derivados de ejecutar una consulta especificada o de guardar una relación base. Las instantáneas no se actualizan al cambiar la relación sobre la que están definidas, pero se actualizan cada cierto tiempo, de acuerdo a lo indicado por el usuario en el momento de su creación. Por esa razón, no pueden ser modificadas por el usuario, siendo entonces consideradas de solo lectura, pues solo pueden ser actualizadas por el sistema.

La figura 2.3 muestra gráficamente un resumen de la clasificación anterior.

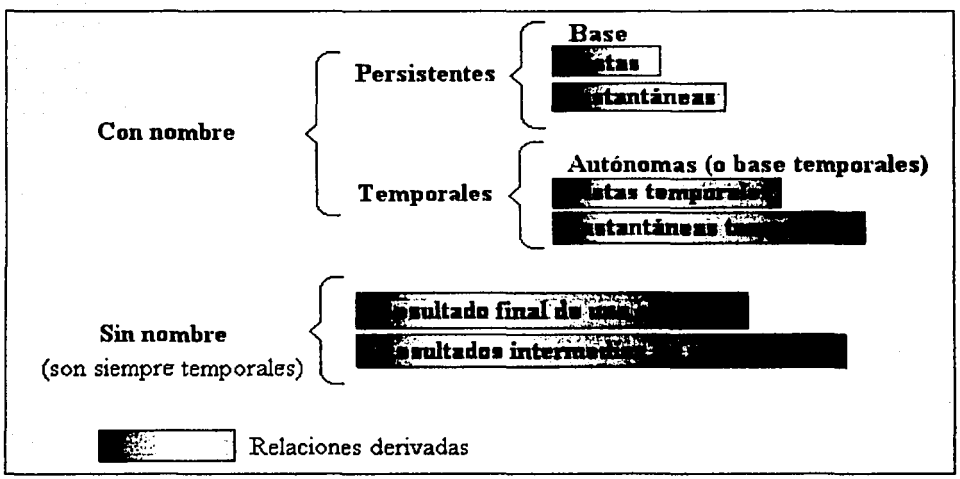


Figura 2.3 Clasificación de las relaciones.

2.2.5 CLAVES DE LAS RELACIONES

Lamamos Clave candidata de una relación, al conjunto de atributos que identifican de manera única y mínima a cada tupla de la relación y que por tanto, tiene la facultad de señalar sin ambigüedad y de forma única, a una, y solo a una, de las tuplas de esa relación.

En la figura 2.4 se ejemplifica el uso de una clave en una relación.

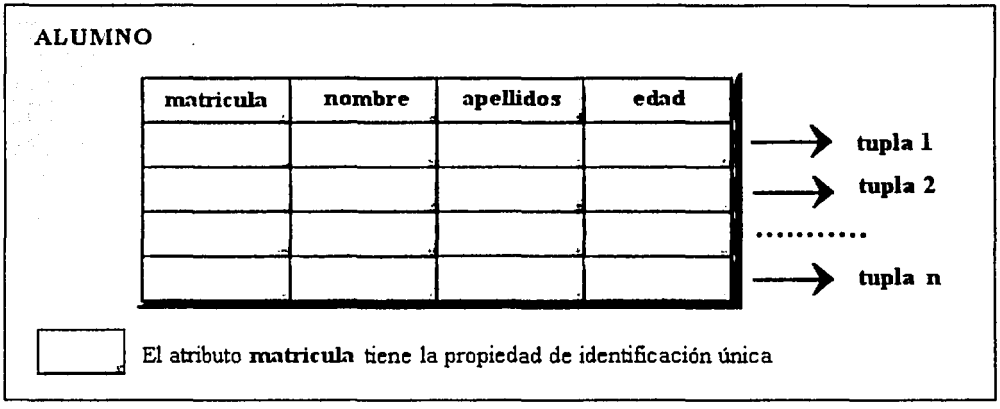


Figura 2.4 Representación de una clave en una relación.

En este ejemplo, al atributo matricula se le ha especificado la propiedad de identificación única, asegurando así que no existan dos tuplas iguales, por consiguiente, podemos identificar mediante de este atributo cualquier tupla de la relación ALUMNO

Una relación puede tener más de una clave candidata, de entre las cuales podemos distinguir tanto a la clave primaria, como a las claves alternativas.

- Una clave primaria es aquella clave que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación. En el caso de existir solo una clave, ésta será entonces considerada la clave primaria.

- Las claves alternativas son aquellas claves candidatas que aún no han sido escogidas como clave primaria.

Hay que recalcar que aun cuando una relación puede contener más de una clave candidata, sólo una de ellas será tomada como clave principal.

Otro aspecto importante es que toda relación, por definición debe contener al menos una clave candidata.

La distinción entre la clave primaria y la alterna va a tener influencia sobre la implementación física de la relación y sobre los procesos de acceso a la información mantenida en la misma, pero no sobre la semántica de la relación.

2.3 RESTRICCIONES

En cada modelo existen restricciones, es decir, estructuras o valores no permitidos, estas restricciones pueden ser inherentes al modelo, o bien de tipo semántico, y los datos almacenados en la base, deberán adaptarse a estas reglas impuestas por el modelo, (por ejemplo no tener tuplas duplicadas, o más de una clave principal), también han de cumplir con las restricciones que imponga el diseñador de la base de datos pensando en el usuario final, todo con el objeto de constituir solo peticiones válidas que encajen dentro del esquema.

2.3.1 RESTRICCIONES INHERENTES

Como se mencionó en el punto anterior, las restricciones inherentes son aquellas impuestas por el mismo modelo, en este caso se trata del modelo relacional, el cuál no admite ciertas estructuras, mismas que no son definidas por los usuarios, sino que son obligadas por el mismo modelo, no habiendo otra salida que llevarlas a cabo, esto llega a quitar flexibilidad a la hora de representar los problemas del mundo real.

A continuación se listan las restricciones inherentes al modelo relacional.

- No hay dos tuplas iguales (de donde se deduce la obligatoriedad de la clave primaria), esta característica es inherente al modelo relacional, sin embargo no lo es para la mayoría de los productos empleados para elaborar bases de datos relacionales, cuya flexibilidad para repetir una tupla dentro de una misma tabla, hacen de esta definición no obligatoria, sin embargo, las tablas que admiten filas duplicadas no son consideradas "conjuntos" (en el sentido matemático), sino "multiconjuntos".

- El orden de las tuplas no es significativo

- El orden de los atributos no es significativo

- Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo, lo que significa que no puede contener un valor desconocido o inexistente. A esta restricción se le conoce como: Regla de integridad de entidad.

- Cada atributo sólo puede tomar un único valor del dominio sobre el que está definido, no admitiéndose por tanto los grupos repetitivos. Se dice que una tabla que cumple con esta condición se encuentra normalizada.

2.3.2 RESTRICCIONES SEMÁNTICAS

Dentro del contexto relacional, nos referimos a las restricciones semánticas o de usuario como las facilidades que el modelo ofrece a los usuarios a fin de que éstos puedan reflejar en el esquema, lo más fielmente posible, la semántica del mundo real.

Estas restricciones semánticas, aún cuando son de gran utilidad, llegan a resultar insuficientes, por lo que los productos comerciales utilizados para la creación de bases de datos relacionales, suelen incluir ciertas facilidades que pueden programarse, otras sentencias para facilitar la manipulación de las bases de datos pueden ser incluidas (de forma embebida) mediante un programa de aplicación.

Las principales restricciones del modelo relacional son las siguientes:

Clave primaria: Nos permite declarar un atributo o un conjunto de atributos como *clave primaria* de una relación. Cuando un atributo ha sido señalado como clave principal, los valores que éste tome no podrán ser repetidos ni se admitirán valores nulos (también llamados valores ausentes), aún cuando se supone que la clave primaria de una relación es una restricción inherente por estar contemplada en el modelo relacional como obligatoria, la mayoría de los productos para crear bases relacionales no obligan a la declaración de este tipo de clave, por lo que se puede considerar como una restricción semántica que responde a la necesidad del usuario de imponer valores en conjunto que no se repetirán en la relación ni tomarán valores nulos, y así garantizar que no haya duplicidad en las tuplas.

Unicidad (UNIQUE). Indica que los valores en un conjunto de atributos (uno o más) no pueden repetirse en una relación, ésta restricción sirve para poder definir claves alternativas a la principal.

Obligatoriedad (NOT NULL), ningún atributo o conjunto de atributos puede quedarse vacío o contener caracteres no válidos.

Integridad de Referencia o Referencial. Sea *T1.a* un atributo de la tabla *T1* que forma parte de una clave ajena para la tabla *T2*. Es decir, que en *T2* existe un atributo definido con el mismo dominio, aunque no obligatoriamente con igual nombre, y que es parte de su clave primaria. Entonces, *T1.a* debe ser siempre igual a algún valor ya contenido en el atributo referenciado en la tabla *T2*, o bien tomar un valor nulo.

En figura 2.5, podemos observar como los valores del atributo codigo de la relación AUTOR, concuerdan con los de la clave primaria clave_libro de la relación LIBRO.
Hay que destacar el hecho de que los atributos que son clave ajena (en este caso clave_libro) de una relación, no tienen porqué coincidir con los nombres de la clave primaria (en el ejemplo la clave primaria es codigo) de la relación diferenciada.

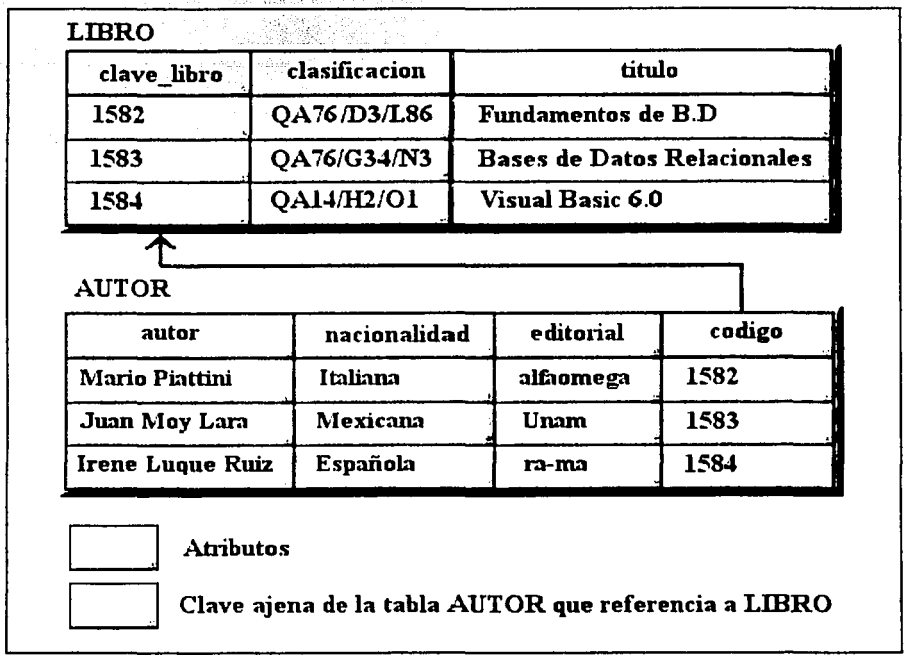


figura 2.5 ejemplo de relación referencial

La integridad referencial es una importante restricción que está impuesta por el mundo real, es el usuario quién la define al describir el esquema relacional, y el modelo la reconoce sin necesidad de programar o escribir algún procedimiento extra para obligar a su cumplimiento.

Además de definir las claves ajenas, hay que determinar que consecuencias pueden derivarse de ejecutar ciertas acciones (como el borrado y modificación), realizadas sobre tuplas de relación diferenciada. Las consideraciones que debemos hacer se definen como:

- **Operación restringida** (NO ACTION): Cuando el borrado de las tuplas que contienen clave referenciada o la modificación de dicha clave solo se permite si no existen tuplas con ese valor en la relación que contiene la clave ajena.

  Por ejemplo: Podemos borrar o modificar un número del atributo codigo, siempre y cuando en el atributo clave_libro de la tabla LIBRO, no exista ningun libro con dicho codigo. En caso contrario, el sistema debe impedir el borrado o modificación de dicho valor.

- **Operación con transmisión en cascada** (CASCADE): Es cuando el borrar una tupla de relación que contenga una clave candidata referenciada, implica un borrado en cascada de todas las tuplas a la cuales está relacionada dicha clave, obviamente, ocurre algo similar al cambiar el contenido de dicha clave, todos las tuplas referenciadas deben ser cambiadas también.

  Un ejemplo sería el siguiente: En el caso de modificar o borrar el código único de un libro en una tabla, entonces deben ser modificadas o eliminadas automáticamente, todas las tuplas relacionadas a ese código (ver figura 2.6).

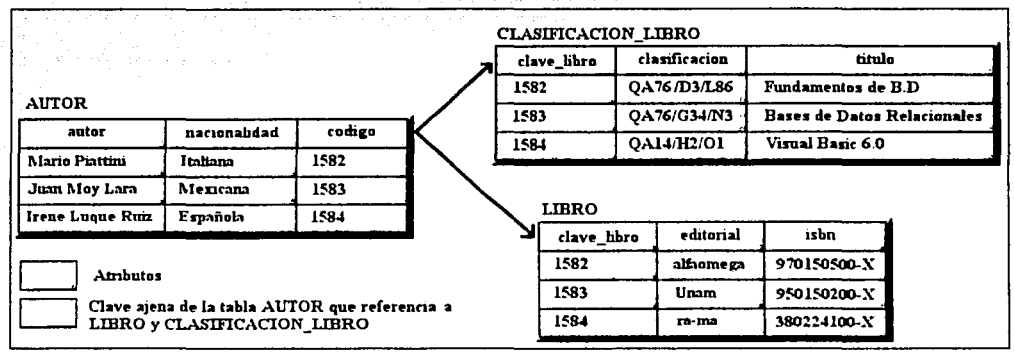


figura 2.6 ejemplo de relación que requiere transmisión en cascada.

- **Operación con puesta a valor por defecto** (SET DEFAULT): El borrado o modificación de las tuplas de la relación que contiene la clave candidata implica agregar un valor preestablecido, este valor ha sido agregado por defecto al crear la tabla correspondiente

Es importante recordar que la combinación de opciones de las restricciones semánticas que se usen, afectara a varias tablas, y si no se tiene un control adecuado de acciones a realizar se puede afectar seriamente la integridad de la base de datos, de modo que solo una especificación adecuada de las estructuras referenciales, puede garantizar la veracidad de la información.

Conjuntamente a las restricciones semánticas que se han descrito, existen otras que son llamadas, *restricciones de rechazo*, y sirven para evaluar las peticiones hechas por el usuario, dichas peticiones, se hacen por medio de predicados definidos sobre un conjunto de atributos, tuplas o dominios, los cuales son verificados por el sistema basándose en las restricciones de rechazo, de modo que esos predicados constituyan solo ocurrencias válidas del esquema, en caso de que dicha operación solicitada por el usuario intente violar alguna condición, el sistema impedirá que se lleve a cabo, es decir se produce un rechazo de la operación solicitada.

El modelo relacional describe dos tipos de rechazo, uno para cuando la condición afecta a un único elemento de la base de datos o afecta a más de uno.

- **Verificación** (CHECK). Esta restricción sirve para comprobar si el predicado (la petición hecha por el usuario) es cierto o falso, en el caso de que el predicado sea falso, la operación es rechazada. La operación de verificación se define sobre un único elemento (por ejemplo en una relación) y puede o no tener nombre.

- **Aserción** (ASSERTION). Esta restricción actúa de manera idéntica a la de verificación, diferenciándose solamente en que la de aserción puede afectar a varios elementos (por ejemplo a dos relaciones distintas) y siempre lleva un nombre.

Una regla de integridad (restricción semántica) está formada, además de su nombre, por dos componentes:

- La restricción propiamente dicha, que es quien dicta la condición que será evaluada y que deberán de cumplir los datos.

- La respuesta a la violación, formada por la especificación de la acción o acciones a ejecutar en caso de no cumplirse la restricción, éstas acciones van desde rechazar la petición e informar al usuario, hasta corregir el error con acciones complementarias.

2.4 El valor nulo en el modelo relacional

El valor *NULO* o *NULL* es una señal que se utiliza para representar información desconocida, inexistente, no válida o indefinida. Entre las razones que hacen del valor nulo una necesidad están:

- Insertar tuplas con ciertos atributos desconocidos en el momento de la inserción.

- Atributos inaplicables a ciertas tuplas, por ejemplo en nombre del cónyuge para una persona soltera o la profesión para un menor de edad.

- Añadir un nuevo atributo a una relación existente, ese atributo al momento de crearse será llenado con valores nulos.

El tratamiento de valores nulos ha llevado a especificar operaciones de comparación, operaciones aritméticas, operaciones algebraicas y funciones de agregación de forma específica para el caso de que alguno de los operandos tome valores nulos, e incluso obliga a introducir operadores nuevos.

Uno de los operadores nuevos es IS_NULL que toma el valor cierto si el operador es nulo y falso en caso contrario.

En cuanto a las operaciones aritméticas, es considerado como nulo el resultado cuando alguno de los operandos es nulo.

Por otro lado, en las operaciones lógicas de igualdad o desigualdad entre tuplas, solo se consideran a dos tuplas duplicadas si atributo por atributo ambos son iguales y no nulos o ambos nulos.

2.5 EL MODELO RELACIONAL Y LA ARQUITECTURA ANSI

Ya hemos hablado acerca de la arquitectura ANSI, y de los tres niveles que la describen (externo, conceptual, e interno), ahora examinaremos el modelo relacional, desde la perspectiva que nos proporciona la arquitectura ANSI.

Nivel Externo: El nivel externo está constituido por las vistas, las tablas virtuales, los lenguajes de manipulación y todos los programas que usarán los usuarios para el manejo, recuperación, visualización y actualización de la información contenida en la base de datos.

Nivel Conceptual: En el ámbito conceptual, se distinguen además de los dominios y restricciones entre los elementos de la base de datos (como son las dependencias entre ellos), las relaciones persistentes, las cuales en el modelo relacional son llamadas tablas reales, ya que tienen una representación directa en el almacenamiento interno.

Nivel Interno: Este esquema sólo contempla aspectos físicos y por lo que respecta al esquema interno, el modelo relacional no especifica nada al respecto, esto es natural, tratándose de un modelo lógico, haciendo una analogía, cada registro almacenado corresponde a una tupla, pero un registro puede estar constituido por varias tuplas y de diferentes relaciones. Además de que cada producto para crear bases relacionales ofrece elementos internos distintos, que ayudan a mejorar el rendimiento del sistema, como los son índices, peticiones, agrupamientos, etc. Por lo que en un esquema relacional, solo nos ocuparemos de los niveles externo y conceptual.

Podemos concluir entonces que el modelo relacional se adapta bastante bien a la arquitectura ANSI, exceptuando en que la arquitectura ANSI especifica que al usuario se le debe limitar a permanecer siempre dentro del esquema externo, utilizando solo las vistas que le proporciona el sistema, sin embargo, el modelo relacional autoriza al usuario tanto el acceso a las relaciones base (mostradas como tablas) como a las vistas (que no tienen representación directa en el almacenamiento), siendo éstas últimas, (según la arquitectura ANSI) responsabilidad exclusiva del diseñador o administrador de la base de datos.

Otra divergencia a notar, es el hecho de que ANSI señala a las vistas como elementos que pueden ser actualizados en todo momento, pero en el modelo relacional, no todas las vistas cumplen esta sentencia.

En la práctica, muchos productos relacionales no corresponden a la arquitectura ANSI a tres niveles, pero esto se debe a que las definiciones del esquema conceptual y del esquema interno no están claramente diferenciadas.

Aún así, el modelo relacional, es el que ha tenido más aceptación, comercialmente hablando, ya que ha proliferado de tal manera que prácticamente el 80% de las bases de datos que se construyen son de tipo relacional.

La figura 2.7 muestra una visión global del modelo relacional dentro del marco definido por la arquitectura ANSI.

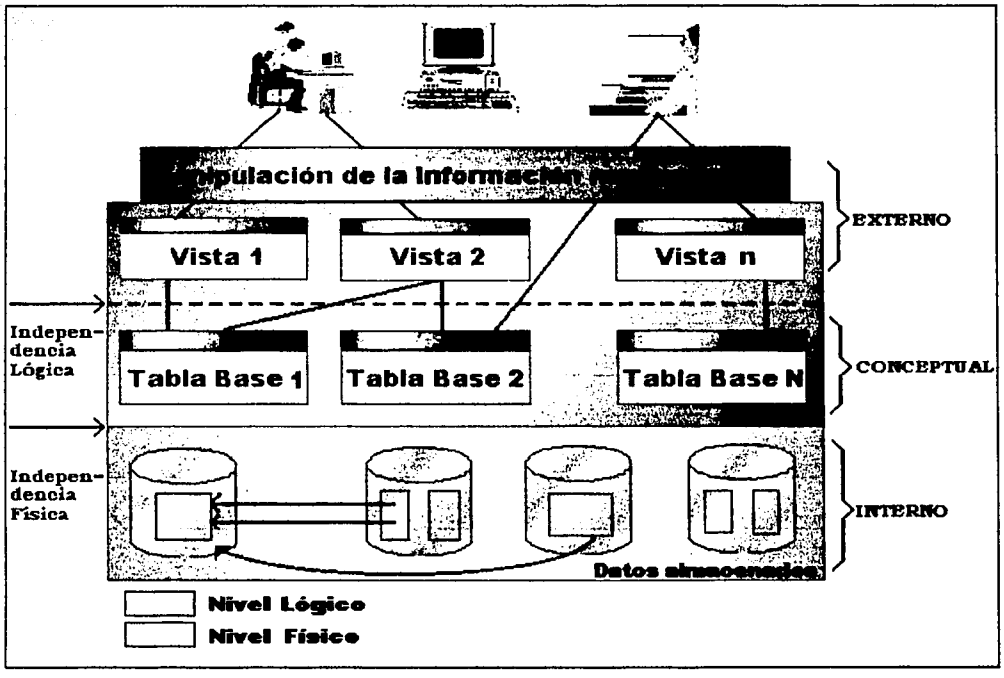


Figura 2.7 Representación de una arquitectura a tres niveles en un SGBD

En la figura 2.7, podemos observar que el nivel externo contempla la creación de vistas y la manipulación que harán de las mismas los usuarios por medio de un lenguaje de manipulación, en este caso el SQL.

En el nivel conceptual se contempla la creación de las tablas, dominios, relaciones entre las tablas etc.

El nivel interno plantea la forma en como los datos han de ser organizados físicamente (creación de índices, clusters, particiones etc.)

2.6 Las primeras 12 reglas de Codd para los sistemas relacionales.

El éxito del modelo de datos del Dr E.F. Codd desató consigo no solo la proliferación de manejadores de bases de datos relacionales, también originó una serie de debates acerca de cuál sería el mejor modelo de datos a seguir, comparando al modelo de datos relacional con sus antecesores que fueron el modelo Jerárquico y el modelo de Red.

El problema principal que enfrentaban el modelo jerárquico y de red, es que carecían de una base conceptual sólida, incluso no se tenía una noción clara del concepto de modelo. El mérito del Dr. Codd está en proponer una base matemática simple pero bien definida. En medio de tales debates y discusiones sobre la inclusión de características relacionales en productos no relacionales, el Dr. Codd publica en 1985 sus famosas 12 reglas que dictan que tan relacional es un producto, y llegando sorprendentemente a la conclusión de que en ese entonces no existía un solo producto que pudiera llamarse así mismo 100% relacional.

A continuación se enumeran dichas reglas:

1. Representación de la información: Toda información relacional debe representarse explícitamente a nivel lógico, y de manera única por medio de valores en tablas.

2. Acceso garantizado: Todo dato debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave y el nombre de una columna.

3. Tratamiento sistemático de valores nulos: Los valores nulos, información desconocida o inaplicable, han de ser tratados sistemáticamente por el sistema, el cuál ha de ofrecer las facilidades necesarias para su tratamiento.

4. Catálogo activo en línea basado en el modelo relacional: La representación de la metainformación[19] debe ser igual a la de los otros datos, y su acceso debe de poder realizarse por medio del mismo lenguaje relacional que se utiliza para los demás datos, es decir que el modelo de datos para la metainformación debe ser también relacional.

5. Sublenguaje de datos completo: Debe existir un lenguaje que permita un completo manejo de la base de datos (definición de datos, definición de vistas, manipulación de datos, restricciones de integridad, autorizaciones y manejo de transacciones).

6. Actualización de vistas: Toda vista teóricamente actualizable debe poder ser actualizada por el sistema.

7. Inserciones, modificaciones y eliminaciones de alto nivel: Todas las operaciones de manipulación de datos (consulta, inserción, modificación y borrado) deben operar sobre conjuntos de filas, es decir no debe de actuar registro a registro.

[19] Descripción de la base de datos.

8. Independencia física de los datos: El acceso lógico a los datos debe mantenerse incluso cuando cambien los métodos de acceso o la forma de almacenamiento,

9. Independencia lógica de los datos: Los programas de aplicación no deben verse afectados por cambios realizados en las tablas que estén permitidos teóricamente y que preserven la información.

10. Independencia de la integridad: Las reglas de integridad de una base de datos deben de ser definibles por medio del sublenguaje de datos relacional y habrán de almacenarse en el diccionario de datos (metabase) y no en la misma aplicación.

11. Independencia de la distribución: Debe existir un sublenguaje de datos que pueda soportar bases de datos distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos por primera vez o se redistribuyan éstos posteriormente.

12. Regla de no subversión: Si un SGBD soporta un lenguaje de bajo nivel que permite el acceso fila a fila, éste no puede utilizarse para saltarse las reglas de integridad expresadas por medio del lenguaje de más alto nivel.

De entre los modelos existentes para la descripción y manipulación de datos que se desea almacenar en una base de datos, no he dudado un momento en enfocarme a describir el modelo de datos relacional, que a diferencia de otros, como el jerárquico o el de red, genera esquemas muy simples a la comprensión y muy fáciles de manejar, sin embargo, la simpleza de sus esquemas, no resta poder a su capacidad de abstracción.

Hasta el momento de escribir este trabajo, puedo afirmar con seguridad, que no hay otro modelo de datos con el nivel de abstracción que permite el modelo relacional, esto gracias a la matemática simple, pero bien fundamentada del Dr. Codd.

Con el surgimiento del modelo relacional, aparece el lenguaje de cuarta generación llamado SQUARE que luego de varias modificaciones deriva en el lenguaje SEQUEL para finalmente convertirse en el tan famoso SQL que actualmente es el lenguaje de uso universal para la creación y manipulación de bases de datos relacionales, razón por la cuál, dedicaré el siguiente capítulo a exponer los fundamentos de este lenguaje.

CAPITULO 3 EL LENGUAJE SQL

El Lenguaje de Consulta Estructurado (SQL, por sus siglas en inglés) es un lenguaje de manipulación de datos empleado en los SGBD relacionales, y cuya semántica es muy parecida al lenguaje natural.

Las instrucciones SQL se encuentran organizadas en dos grupos o categorías:

- Las instrucciones para la manipulación de datos (DML), diseñadas para seleccionar, contar, clasificar y calcular la información almacenada en las tablas, es decir, recuperar registros de la base de datos.

- Y las instrucciones para la definición de datos (DDL), que están pensadas para utilizar sentencias SQL con la finalidad de crear componentes de la base de datos, por ejemplo, definir tablas, campos, índices y relaciones en la base de datos.

El SQL comenzó a ser diseñado en la década de los años sesenta en IBM para permitir a los usuarios el uso de instrucciones estandarizadas en diversas bases de datos.

La premisa era crear un lenguaje que no estuviera basado en ningún lenguaje de programación, pero sin embargo pudiera ser empleado de manera indistinta por todo lenguaje de programación, para recuperar y modificar el contenido de las bases de datos.

Partiendo de la idea original del modelo relacional, se planteó una forma matemática de acceso y manipulación de la información, este planteamiento derivó en una visión predicativa de los datos, que posteriormente originó un lenguaje basado en el álgebra y cálculo de predicados llamado SQUARE. Este lenguaje fue mejorado y ampliado, hasta obtener una nueva versión llamada SEQUEL, misma que fue enriquecida y en 1976 derivó en la primera versión del lenguaje SQL.

En 1979 surge el primer SGBDR comercial basado en SQL, llamado ORACLE, posteriormente surgen otros productos basados en SQL, como son: INTERBASE, INFORMIX, SYBASE e incluso otros productos que no estaban basados en este lenguaje, como INGRES, ADABAS Y SUPRA comenzaron a ofrecer interfaces SQL

Actualmente el SQL se ha convertido en un método estándar de uso universal para la manipulación de bases de datos. Los SGBDR, como ORACLE y SQL Server lo implementan de distintas maneras, normalmente lo emplean para la creación de consultas que extraigan datos de la base de datos, aunque tiene muchas otras funciones, como la creación de tablas y campos.

Las instrucciones de SQL son solo eso: instrucciones. Cada instrucción puede llevar a cabo operaciones en los objetos (tablas, columnas, índices etc.) de una o más bases de datos. SQL utiliza una serie de operadores algebraicos que operan sobre las relaciones o tablas de un esquema relacional, los operadores pueden operar sobre una tabla (unitarios) o dos tablas (binarios), pero siempre sobre la totalidad de la tuplas que conforman la extensión de la tabla.

Las operaciones básicas que realiza SQL son: Unión, diferencia, selección y producto cartesiano, también cuenta con operaciones avanzadas como la intersección, la reunión y la división.

A continuación se muestran algunos de los operadores más utilizados:

=	Operador de comparación de la igualdad
<> ó !=	Operador de comparación de la desigualdad
<	Operador de comparación que representa *menor que*
>	Operador de comparación que representa *mayor que*
<=	Operador de comparación que representa *menor o igual que*
>=	Operador de comparación que representa *mayor o igual que*
*, /	Operadores binarios que representan el producto y la división
+,-	Operadores binarios que representan la suma y la diferencia. Estos mismos operadores utilizados de forma unitaria, representan una expresión positiva o negativa.
IN	Operador de comparación que representa la igualdad a cualquier miembro de un grupo de valores en lugar de a un valor
BETWEEN x **AND** y	Operador de comparación que representa la pertenencia a un intervalo de valores
EXIST	Operador de comparación que representa la existencia en un objeto de valores
X **LIKE** y	Operador de comparación que representa la existencia de una subcadena en una cadena de caracteres
IS NULL	Operador de comparación representando la existencia de un valor nulo
NOT,AND, OR	Operadores lógicos que pueden acompañar a cualquier expresión correcta
Y algunas de las funciones más utilizadas son:	
UPPER (cadena)	Para convertir una cadena a mayúsculas
INITCAP (cadena)	Convierte a mayúscula sólo el primer carácter de la cadena

MONTHS_BETWEEN (d1,d2)	Devuelve el número de meses transcurridos entre las dos fechas señaladas.
SYSDATE	Devuelve la fecha y hora del sistema
TO_DATE (cadena [formato])	Convierte una variable de tipo cadena a tipo DATE, indicando el formato en el cual se desea representar la fecha (y hora)
AVG (lista_valores)	Obtiene el valor medio de la lista de valores numéricos
SUM (lista_valores)	Obtiene el sumatorio de la lista de valores numéricos
MAX (lista_valores)	Obtiene el valor máximo de la lista de valores numéricos
MIN (lista_valores)	Obtiene el valor mínimo de la lista de valores numéricos
STDEV (lista_valores)	Obtiene la desviación típica de la lista de valores numéricos
COUNT (expresión)	Obtiene el número de tuplas que satisfacen la expresión. Si la expresión es (*) obtiene el número de tuplas que satisfacen la consulta.

Fuente de los datos mostrados en la tabla: Diseño y uso de Bases de Datos Relacionales Ed. ra-ma autores. Irene Luque Ruiz, Miguel Ángel Gómez-Nieto.

3.1 EL SQL-ANSI ESTÁNDAR

La sintaxis utilizada en este lenguaje de manipulación de base de datos, es determinada por un comité que forma parte del Instituto Nacional Americano de Estándares (ANSI), Dicho comité está formado por profesionales informáticos que se encargan de establecer y hacer cumplir las normas establecidas. No obstante que existen tantas variantes en el lenguaje SQL, como productos relacionales en el mercado, la mayoría de ellos tratan de apegarse a los estándares fundamentales definidos por el comité ANSI-SQL.

El estándar más utilizado es el denominado SQL-89 (presentado en año 1989) y al cual le aprobaron las últimas modificaciones en el año 1992, por lo que denominaron a estas reformas SQL-92.

En esta última versión, es ampliada la capacidad semántica del esquema relacional, se mejora el tratamiento de errores y se incluyen normas para el SQL embebido.

Cada conjunto de normas SQL tiene tres niveles de conformidad. Un producto de bases de datos deberá cumplir al menos con el nivel 1 para poder ser considerado compatible con el estándar SQL. Los niveles 2 y tres son opcionales, aplicables a aquellos productos que quieran mejorar su compatibilidad con otros productos relacionales.

3.2 LOS ESQUEMAS RELACIONALES

El esquema de una base de datos relacional está formado básicamente por la definición de un conjunto de tablas[20] (relaciones). Cada tabla debe tener un nombre único en el esquema. Además de que la definición de cada tabla, deberá estar basada en la especificación de un conjunto de atributos (las columnas de la tabla), que representan las propiedades del objeto del mundo real representado en cada tabla.

Cada atributo debe tener un nombre único para una tabla y estará definido en un dominio de datos preestablecido. De entre todos lo atributos que forman la definición de una tabla, uno o un conjunto de ellos se elegirán como clave principal de la misma.

Opcionalmente, en la definición de una tabla pueden ser especificadas las claves foráneas, es decir, aquellos atributos de la tabla que están definidos en el mismo dominio y representan la misma propiedad que la clave principal de alguna otra tabla que forma parte del esquema de la base de datos.

3.3 ¿DÓNDE NOS ES ÚTIL EL LENGUAJE SQL?

Está claro que SQL, se utiliza como medio para recuperar los datos almacenados en las bases de datos relacionales, y es embebido por los SGBD como una de las facilidades que se le proporcionan al usuario para comunicar sus peticiones al sistema. Sin embargo, no siempre queda tan claro donde se incluyen las consultas necesarias para recuperar registros.

Pues bien, una instrucción SQL se utiliza en cualquier lugar donde se pueda emplear una referencia a una tabla, incluso es posible asegurar que si ha trabajado con algún sistema que emplee bases de datos como Microsoft Access o utilizado una aplicación desarrollada con Visual Basic que almacene datos, ha hecho uso de instrucciones SQL, con o sin su consentimiento.

3.4 MANIPULACIÓN DE LA INFORMACIÓN

El hacer referencia a manipulación de datos, supone operaciones de inserción, modificación, borrado y consulta de información almacenada en la base de datos. La solución que aporta SQL, es una serie de verbos asociados a cada una de estas operaciones, con una sintaxis clara y sencilla pero además potente, pues permite la anidación de sentencias en las que se requiera más de un verbo. De modo que SQL realiza ambas operaciones, o bien, distingue cuál de las dos llevar a cabo dependiendo si se cumple o no la condición evaluada.

---

[20] Una tabla es la combinación de campos y registros.

3.5 CONSULTAS

Una consulta es todo conjunto de instrucciones que recuperan registros de una base de datos.

Por medio de un solo predicado, se puede extraer datos de uno o más campos y de una o más tablas. Incluso los datos recuperados, pueden filtrarse sometiéndolos a una o más restricciones conocidas como criterios, de esta manera se limita la cantidad de registros recuperados.

Las instrucciones de SQL pueden crear grupos de registros a los que se les conoce como vistas.

Consultar información de una base de datos supone entonces, el acceso seleccionado a los datos de una o más tablas, obteniendo como resultado un valor, un conjunto de valores e incluso una nueva tabla.

3.5.1 La cláusula SELECT

SQL ofrece una estructura muy sencilla para la recuperación de datos, y el núcleo de esta consulta de recuperación es la cláusula SELECT.

El formato general para esta operación es:

SELECT [**DISTINCT** | **ALL**] *lista de atributos-1*

[**INTO**] *variable*

FROM *lista - tablas*

[**WHERE**] *Condición-1*

[**GROUP BY**] *lista atributos-2*

[**HAVING**] *Condición-2*

[**ORDER BY**] *lista atributos* [**ASC** | **DESC**];

Nota Importante: En lo sucesivo, para especificar las cláusulas del lenguaje SQL, será empleada la Forma Normal de Backus (BNF) donde:

[ ]	indica elementos opcionales
|	indica una alternativa
...	indica repetición

La palabra **DISTINC**, acompañada de algún atributo, indica que no podrán existir valores duplicados para dicho atributo (campo) en las tuplas (registros) que formarán la relación resultante.

La palabra **ALL**, obliga a devolver el conjunto completo de tuplas que coincidan con la petición, sin importar si hay valores repetidos para los atributos seleccionados. Por defecto la consulta devolverá valores duplicados.

La *lista de atributos-1* representa al grupo de campos, sobre los cuales se quiere realizar la operación de proyección. Pero si lo que se desea es que la operación de consulta incluya a todos los campos de las tablas que participan en la consulta, entonces se agrega solamente el modificador * , que hace mención de todos los campos, de todas las tablas que intervengan en la consulta.

El verbo **INTO**, permite precisar al usuario una variable para controlar el acceso a las tuplas recuperadas de la operación de consulta.

La cláusula **FROM** es necesaria para especificar cuales son las tablas que intervendrán en la consulta, enseguida de la cláusula **FROM** se coloca el nombre de cada una de las tablas, separando cada nombre por una coma.

En caso de ser especificado **WHERE** la operación de consulta realiza una selección de las tuplas obtenidas, filtrando los datos basándose en la condición especificada en *Condición-1*.

Otra cláusula opcional es **GROUP BY**, y se emplea para construir grupos de tuplas basadas en los campos dictados en lista de atibutos-2. Es decir, dividir una tabla en grupos (denominados tablas agrupadas), de acuerdo a los valores de ciertas columnas.

La cláusula **HAVING** se utiliza para restringir el conjunto de grupos que pueden ser formados por la cláusula **GROUP BY** (a la que va asociada), la función que realiza es análoga a la de **WHERE**, pero aplicado a tablas agrupadas.

Por último tenemos a la cláusula **ORDER BY** que tiene como función proporcionar un orden a las tuplas recuperadas en la consulta, este orden es especificado mediante los

parámetros **ASC** si queremos que sean ordenadas las tuplas en forma ascendente o **DES**, en el caso de preferir un orden descendente.

Ejemplos:

Una consulta **SELECT** básica

SELECT * FROM tblClientes;

Esta consulta recupera todos los registros y todos los campos de la tabla tblClientes (sin un orden determinado).

Para recuperar sólo el nombre y apellido de los clientes, podemos cambiar el predicado de la siguiente manera:

SELECT Nombre, Apellidos **FROM** tblClientes;

En el siguiente ejemplo, es utilizada la cláusula **WHERE** para indicar al motor de la base de datos que limite los registros que recupera en función de los criterios establecidos.

SELECT Nombre, Apellidos **FROM** tblClientes **WHERE** estado = 'D.F';

WHERE nos sirve para filtrar el resultado de la consulta, ya que sólo aparecerán las filas que cumplan con la condición de búsqueda.

El predicado de comparación, bien puede incluir símbolos como igual (=), distinto (<>), menor que (<), mayor que (>), menor o igual a (<=) y mayor o igual a (>=).
Como ejemplo tenemos la siguiente consulta:

SELECT Titulo, Autor, Editorial, Año **FROM** tblLibros **WHERE** Año >= 1999;

3.6 El Operador BETWEEN

Este operador nos permite obtener los valores que se encuentran dentro de un intervalo.

Por ejemplo, si quisiéramos una lista de todos los títulos de libros, junto con su autor y editorial que han sido publicados entre el año 1999 y 2001, entonces ejecutaríamos la siguiente consulta:

SELECT Titulo, Autor, Editorial, Año **FROM** tblLibros
WHERE Año **BETWEEN** 1999 **AND** 2001;

3.7 El Operador IN

Se utiliza para comparar un valor con una lista de valores, o bien con una subconsulta. Como ejemplo, podemos suponer que queremos consultar los títulos de libros publicados por las editoriales Ra-Ma y McGraw-Hill, lo podemos hacer por medio de la siguiente consulta:

```
SELECT Titulo, Editorial, FROM tblLibros
WHERE Editorial IN ( 'Ra-Ma', 'McGraw-Hill');
```

Podemos notar que es muy similar al operador igual (=), pero con la diferencia de que éste operador de comparación representa la igualdad a cualquier miembro de un grupo de valores en lugar de a un solo valor.

3.8 El Operador LIKE

Este operador nos da una gran ventaja a la hora de realizar las consultas, pues nos permite hacer uso de comodines, mediante el uso de un símbolo que generalmente es un "*" o bien un "%" que se utiliza para indicar cero o n caracteres en el patrón de búsqueda, y mediante el símbolo "_" se especifica exactamente un caracter.

Entonces, si quisiéramos hacer una consulta que devolviera un listado de todos títulos de libros que contengan la palabra "Historia" escribiríamos el siguiente predicado.

```
SELECT Titulo, Editorial, FROM tblLibros
WHERE Titulo LIKE '%Historia%';
```

Esta consulta también podría lucir de la siguiente manera:

```
SELECT Titulo, Editorial, FROM tblLibros
WHERE Titulo LIKE 'Historia*';
```

3.9 El Operador IS NULL

El operador **IS NULL** es empleado para determinar si una columna (campo) contiene valores nulos. Cabe mencionar, que es diferente almacenar en una columna, un espacio en blanco a almacenar un valor NULL; los espacios en blanco se consideran cadenas de caracteres almacenados, mientras que un valor NULL es considerado como un campo "vacío".

Una consulta que nos recuperaría los Títulos de los libros que carecen de ISBN es la siguiente:

```
SELECT Titulo, Editorial, FROM tblLibros WHERE ISBN IS NULL;
```

3.10 La Cláusula ORDER BY

La cláusula **ORDER BY** indica que las tuplas recuperadas serán ordenadas conforme a uno o varios campos especificados, ya sea en forma ascendente o descendente.

Para hacer uso de la cláusula **ORDER BY** escribimos una sentencia **SELECT** normal, incluyendo **ORDER BY** al final, y enseguida los campos que servirán de criterio para el ordenamiento.

Como ejemplo pondremos la instrucción que serviría para recuperar un listado de todos los pedidos de clientes que vivan en el Distrito Federal de la tabla llamada tblClientes ordenándolos por apellido.

```
SELECT IdPedido, Nombre, Apellido FROM tblClientes
WHERE ciudad = 'Distrito Federal'
ORDER BY Apellido;
```

Por omisión los datos son ordenados en forma ascendente, pero si lo que quiere es un ordenamiento en forma descendente, entonces debe especificarlo.

```
SELECT IdPedido, Nombre, Apellido FROM tblClientes
WHERE ciudad = 'Distrito Federal'
ORDER BY Apellido DESC;
```

Para ordenar por varios campos, se debe enumerar cada uno de ellos a continuación del otro, inmediatamente después de la cláusula **ORDER BY.** Por ejemplo, para ordenar el listado anterior, primero por apellido paterno y después por nombre utilizaríamos la siguiente instrucción:

```
SELECT IdPedido, Nombre, Apellido FROM tblClientes
WHERE total >= 1000
ORDER BY Apellido, Nombre;
```

3.4.2 Combinar tablas relacionadas en una consulta

Una combinación se utiliza para extraer información que está distribuida en más de una tabla. Obviamente una de las tablas debe contener una clave primaria[21] y estar relacionada por medio de este campo, con la otra tabla implicada en la búsqueda, de forma que podamos señalar sin ambigüedad la relación entre ambas tablas.

Por ejemplo, tomemos las dos tablas relacionadas que se muestran en la figura 3.1.

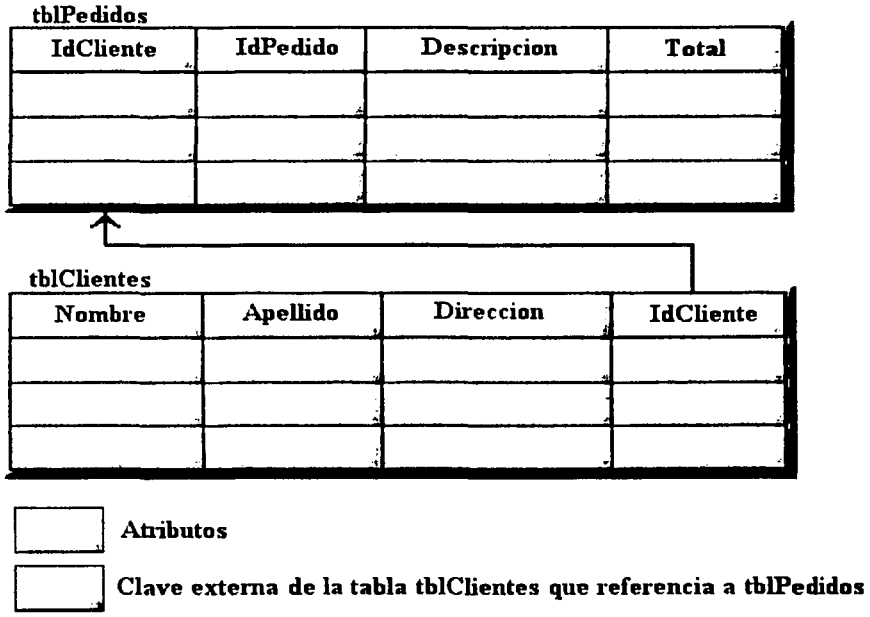


Figura 3.1 Ejemplo de dos tablas relacionadas mediante la clave primaria IdCliente, que es clave principal de la tabla clientes y está relacionada con la clave externa (foránea) de la tabla pedidos.

En la tabla pedidos se almacena la información referente a los pedidos hechos por los clientes, y en la tabla clientes, es almacena información concerniente a los datos personales de los clientes. Lo que se desea, es recuperar un conjunto de registros que muestren la siguiente información: Nombre, Apellido, descripción del pedido y total del importe.

[21] Conjunto de atributos que identifican de manera única y mínima a cada registro de la tabla

Para recuperar un conjunto de resultados como el que acabamos de mencionar, utilizamos una combinación, ésta operación resulta sencilla aún cuando los datos se encuentren en tablas distintas, sólo hay que poner atención en indicar en la consulta, la clave principal y la clave externa.

Mediante SQL, escribiremos la combinación como una expresión de equivalencia entre dos campos, obteniendo lo siguiente:

SELECT Nombre, Apellido, Descripcion, total **FROM** tblClientes, tblPedidos
WHERE tblClientes.IdCliente = tblPedidos.IdCliente;

Observe que si un mismo campo se presenta en dos tablas, en una misma consulta que incluye una combinación (como fue en el ejemplo anterior), entonces debe utilizar la sintaxis **NombreTabla.NombreCampo** en vez de solo el nombre del campo, de esta manera podemos ser explícitos a la hora de hacer mención a una tabla o campo.

Otra forma de expresar una combinación de tablas es por medio de la cláusula **INNER JOIN,** aunque la sintaxis es diferente, ambas consultas devuelven el mismo listado de datos. Utilizando **INNER JOIN** la consulta quedaría de la siguiente forma:

SELECT Nombre, Apellido, Descripcion, total **FROM** tblClientes
INNER JOIN tblPedidos **ON** tblClientes.IdCliente = tblPedidos.IdCliente;

Ésta sintaxis fue incluida en la última actualización del estándar SQL, por lo que no todas las bases de datos son compatibles con ella, sin embargo puede usar la primera expresión, que debe ser aceptada por toda base de datos compatible con el estándar ANSI.

La expresión **INNER JOIN** tiene el propósito de facilitar la lectura de la instrucción, pero la ventaja importante que tiene sobre **WHERE**, es que con **INNER JOIN,** si se pueden actualizar los datos que resultan de la consulta.

3.12 Agrupar datos por medio de una consulta

En algunas ocasiones es necesario obtener un resumen de los datos contenidos en la base, el motivo es contestar a preguntas como: ¿Cuantos pedidos se hicieron ayer?, en ese caso, no es importante saber con exactitud ¿quién hizo el pedido?, lo relevante es conocer el número de pedidos que entraron. Para responder a esta pregunta, puede utilizar consultas de grupo y funciones de agregación.

Lo que hacen las consultas de grupo, es resumir los datos en función de uno o más campos comunes. Y esto se logra mediante la cláusula **GROUP BY.**

Ejemplo: Se desea ver el número total de pedidos que fueron hechos el día de ayer, en ese caso se pueden agrupar los datos basados en el campo fecha, pero tal grupo deberá ser todavía filtrado, de manera que solo sean devueltos los datos que coincidan con la fecha de ayer.

Este último filtro se hace por medio de la cláusula **HAVING,** que es el equivalente a **WHERE,** pero es usado con **GROUP BY.**

Entonces, la consulta se haría mediante el siguiente predicado.

SELECT FechaOrden, **COUNT** (IdClientes) **AS** TotalPedidos **FROM** tblPedidos **GROUP BY** FechaOrden **HAVING** FechaOrden = #1/17/2002#

El conjunto de resultados para esta consulta seria algo parecido a esto:

FechaOrden	**TotalPedidos**
1/17/2002	45

Observe que los valores de fecha están delimitados por los símbolos (#), esto es atendiendo a la nomenclatura que utilizan motores de base de datos como Microsoft Jet SQL. Por otra parte, la Cláusula **AS** es empleada para asignar un nombre a la columna que contiene el resultado de la función de agregación, es necesario agregar un nuevo nombre al campo, debido a que el resultado de la función es calculado, más no almacenado en la base y por medio de éste "alias" podemos recuperar el resultado de la función.

3.13 Asignar alias a nombres de campos utilizando la cláusula AS

Como se vio en el ejemplo anterior, algunas veces es necesario cambiar el nombre de algún campo durante una consulta, por los siguientes motivos:

- Puede ser que la tabla contenga nombres muy largos, difíciles de manejar o simplemente no sean fáciles de recordar, y por ello convenga más darles un alias que nos permita trabajar con mayor comodidad.

- O bien, la consulta creada produce algún tipo de columna de cálculo, como es el caso de una suma, entonces debemos asignarle un nombre por el cuál podamos hacer mención a ese resultado.

Sin importar cual sea el motivo, es posible asignar un alias a un nombre de campo fácilmente, por medio de la cláusula **AS.**

Por ejemplo tomemos el siguiente caso: Supongamos que necesitamos hacer una serie de operaciones con el campo TotalOrden, y queremos referirnos a este campo como Subtotal, con la finalidad de recordar que estamos trabajando con sub totales y no un total general, entonces escribiríamos la siguiente instrucción.

SELECT FechaOrden, IdCliente, [TotalOrden **AS** Subtotal] **FROM** tblPedidos

Para el conjunto de datos que serán devueltos por esta consulta, el campo TotalOden ya no existe, pues ha sido sustituido por Subtotal, lo único que hemos hecho es cambiarle el nombre al campo, no en la tabla origen, sino en el conjunto de datos que nos es devuelto.

La cláusula **AS** nos da la facilidad de simplificar los nombres de los campos asignándoles un alias, pero la aplicación más común que se le da es cuando se realizan operaciones con los campos, por ejemplo, Si quisiéramos desglosar el 15% de impuesto que se paga en cada orden, lo haríamos de la siguiente forma:

SELECT FechaOrden, IdCliente, TotalOrden - (TotalOrden * 0.15) **AS** Subtotal,

(TotalOrden * 0.15) **AS** Impuesto **FROM** tblPedidos;

Es importante mencionar que los campos Subtotal e Impuesto no se almacenan en la base de datos, solo sirven para hacer referencia a los datos que son calculados en el momento. Y debido a que no se almacenan en la estructura de la base, es necesario un nombre para poder hacer referencia a ellos.

3.14 Consultas de acción

Mediante una consulta de acción, es posible modificar el contenido de una tabla, estamos hablando de cambiar el contenido de los registros directamente en la base de datos.

Las consultas de acción no devuelven conjuntos de datos, en vez de ello, realizan cambios permanentes al contenido de la base. Y son empleadas cuando se requiere hacer modificaciones a grandes volúmenes de datos en función de un criterio.

Por ejemplo, supongamos que se quiere aumentar un 10% al sueldo de todos los empleados de la tabla nomina, entonces el problema puede ser resuelto empleando una consulta de actualización (que es uno de los tipos de consultas de acción) para modificar todos los datos del campo Sueldo.

3.14.1 Consultas de actualización

Una consulta de actualización es utilizada para modificar un grupo de registros al mismo tiempo, y consta de tres partes:

- La cláusula UPDATE, mediante la cuál se especifica la tabla a actualizar
- La cláusula SET, nos sirve para indicar cuales serán los datos que cambiarán
- Y el parámetro WHERE, éste último es opcional y se emplea para delimitar el número de registros que serán afectados mediante la consulta de actualización.

Para ejemplificar la consulta de actualización, retomemos el supuesto anterior, en donde se quería aumentar un 10% al sueldo de todos los empleados de la tabla tblNomina, para lo cuál debemos modificar el todos los datos almacenados en el campo Sueldo.

UPDATE tblNomina **SET** Sueldo = Sueldo * 1.1;

De querer limitar el número de registros afectados, entonces agregamos la cláusula **WHERE** a la consulta. Por ejemplo si solo quiere aumentar un 10% al sueldo de los empleados que ganan por debajo de los 10,000 pesos entonces se agregaría lo siguiente a la consulta:

UPDATE tblNomina **SET** Sueldo = Sueldo * 1.1 **WHERE** Sueldo < 10000;

Mediante ésta consulta, solo serán afectados los registros que satisfagan la condición.

3.14.2 Consultas de eliminación

Este tipo de consultas de acción, tiene la capacidad de eliminar a todo un grupo de registros al mismo tiempo. La eliminación se hace empleando la cláusula **DELETE** y enseguida especificando los campos y la tabla origen de los datos que serán afectados.

Se debe poner especial cuidado en el manejo de éste tipo de consultas, pues los registros son borrados de forma permanente, ya que este tipo de acciones repercuten directamente en la en la base de datos.

Para prevenir posibles percances, una buena estrategias es agregar en la tabla un campo de validación, este campo solo almacenará un valor booleano (verdadero o falso) y servirá para indicar el estatus del registro. Así en vez de borrar el registro, podemos cambiar el estatus a falso, el registro no es borrado, pero puede ser ignorado en las consultas mediante un criterio de selección. Y de esta manera, podrá recuperar dichos registros si es que llega a necesitarlos en un futuro.

La siguiente consulta, ejemplifica como borrar todos los registros de la tabla tblPedidos que fueron hechos antes del 28 de enero del 2002

DELETE * **FROM** tblPedidos **WHERE** FechaPedido < #01/28/02#;

La cláusula **DELETE** nos permite borrar uno o varios renglones de una tabla, pero se deben de tener en cuenta las siguientes restricciones:

- No se pueden borrar renglones parcialmente, solo tuplas completas.
- La clúsula **WHERE** determina que tuplas serán eliminadas
- Si se omite la cláusula **WHERE** serán eliminadas todas las tuplas de una relación

3.14.3 Consultas de anexión

Como indica el nombre, las consultas de anexión se hacen pensando en anexar datos a la base, esto se hace de dos formas:

- Agregando un único registro a una tabla.
- Copiando uno o más registros de una tabla a otra.

La anexión de registros se hace mediante la cláusula **INSERT** de SQL, la sintaxis dependerá de si se quiere anexar un único registro, o bien, copiar un conjunto de datos de una tabla a otra.

Para anexar un único registro la sintaxis es la siguiente:

INSERT INTO NombreTabla (NombreCampo1, NombreCampo2 ...)
VALUES (ValorCampo1, ValorCampo2 ...);

Ejemplo: A continuación se escribe la sentencia SQL necesaria para agregar un nuevo registro a la tabla tblPedidos que cuenta con los campos IdPedido, FechaOrden, IdCliente y TotalOrden.

INSERT INTO tblPedidos (FechaOrden, IdCliente, TotalOrden)
VALUES (#28/01/02#, 110, 450.25);

En el momento de ejecutar la consulta anterior, es creado un nuevo registro en la tabla tblPedidos y son llenados los campos con los valores establecidos. Algo importante que seguramente habrá notado, es que en la consulta no fue incluido el campo IdPedido, la razón es que IdPedido es un campo tipo autonumérico, por lo que el contenido que almacenará este campo, será generado automáticamente por el SGBD, y el tratar de introducirlo por una vía diferente produciría un error.

Si lo que se quiere es copiar el contenido de datos de una tabla a otra, entonces se debe de agregar la cláusula **SELECT** a la instrucción. Imaginemos ahora que en vez de eliminar los registros antiguos de una tabla, los almacena en una nueva tabla llamada tblPedidosAntiguos, el contenido de esa tabla puede ser generado automáticamente con la siguiente instrucción:

```
INSERT INTO tblPedidosAntiguos SELECT * FROM tblPedidos
WHERE FechaOrden < #28/01/99# ;
```

3.14.5 Consultas de creación de tablas

Cuando es utilizada una consulta de anexión, se presupone que existe la tabla en donde serán copiados los datos, pero ¿que sucede si queremos guardar los datos en una nueva tabla cada vez?. Es decir, reproducir la estructura de una tabla para crear una nueva y llenarla con un junto de valores extraídos de una consulta. Existe una forma fácil de solucionar este problema mediante la cláusula **SELECT INTO**, que es similar a la consulta de anexión, excepto que en este caso puede crear una tabla nueva y copiar registros en ella, todo mediante una sola consulta.

La siguiente instrucción, sirve para copiar todos los registros de la tabla tblOrden a una nueva tabla llamada tblRespaldoOrden

```
SELECT * INTO tblRespaldoOrden FROM tblOrden;
```

Al ejecutar la consulta, es creada la tabla tblRespaldoOrden y llenada con todos los registros de la tabla tblOrden, pero de existir ya la tabla tblRespaldoOrden, entonces se generará un error, y el contenido de tblRespaldoOrden es reemplazado con los nuevos registros.

En caso de que desee limitar el número de registros a copiar, entonces puede añadir la cláusula **WHERE** para aplicar un criterio de selección, y de esta forma solo copiar un subconjunto de registros de la tabla original en la nueva tabla creada.

Ejemplo:

```
SELECT * INTO tblRespaldoOrden FROM tblOrden
WHERE FechaOrden < #28/01/99# ;
```

3.15 Consultas de Unión

Existen casos en donde será necesario mostrar el contenido de dos tablas como si fueran una sola, esto sucede cuando los datos se encuentran físicamente dispersos en dos tablas, pero requieren ser mostrados en un único resultado, en donde el contenido de ambas tablas se vea unificado mediante un único listado. Las consultas de unión son la solución para hacerlo.

Una consulta de unión fusiona el contenido de dos tablas que tienen estructuras de campos similares

Supongamos que necesita ver el contenido de la tabla tblRespaldoOrden en el mismo conjunto de resultados que la tabla tblOrden, para ello empleamos la siguiente instrucción:

```
SELECT * FROM tblOrden UNION SELECT * FROM tblRespaldoOrden;
```

En este caso, ambas tablas tienen la misma estructura, es decir, los campos tienen el mismo nombre y almacenan el mismo tipo de datos, así que lo que hace esta instrucción es combinar los registros de la tabla tblOrden con los de la tabla tblRespaldoOrden mostrando un solo conjunto de resultados.

Por omisión, las consultas de unión no devuelven registros duplicados, sin embargo puede agregar el operador **ALL** si su intención es mostrar el contenido total de las dos tablas, aún cuando existan registros repetidos.

```
SELECT * FROM tblOrden
UNION ALL
SELECT * FROM tblRespaldoOrden;
```

La condición para que la unión pueda llevarse a cabo es que exista compatibilidad entre dominios, no podemos unir un campo que almacene valores enteros con otro que almacene cadenas de caracteres.

En el caso de que solo algunos campos de las tablas coincidan en dominio, entonces podemos especificar una correspondencia entre los atributos que deseamos combinar.

En este ejemplo se especifica exactamente cuales son los campos que intervendrán en la unión, de esta manera podemos excluir los campos que no sean compatibles en dominio y hacer la unión sin ningún problema.

```
SELECT * FROM tblOrden
UNION CORRESPONDING (FechaOrden, TotalOrden)
SELECT  * FROM tblRespaldoOrden;
```

3.16 Definición de datos

Las cláusulas CREATE, ALTER Y DROP permiten respectivamente crear un esquema de relación (tabla), modificarlo o eliminarlo.

Para crear una relación es preciso especificar el nombre de la relación y los nombres y tipos de cada atributo.

La siguiente instrucción SQL define una relación llamada tblAlumno, con los atributos NumeroCuenta, Nombre, Direccion y Telefono. El atributo NumeroCuenta es un entero de 32 bits, Nombre es una cadena de 50 caracteres, Direccion es una cadena de 80 caracteres y Telefono es un entero de 16 bits. No se admiten valores nulos en el atributo NumeroCuenta.

```
CREATE TABLE tblalumno (NumeroCuenta int NOT NULL,
Nombre char (50), Direccion char (80), Telefono smallint);
```

El resultado de la sentencia anterior es la creación de una tabla cuyo esquema es apropiadamente reflejado en el diccionario de datos. Esta relación apenas creada no contiene aún ninguna tupla.

Pero ahora supongamos que necesitamos agregar un campo más a nuestra tabla, por ejemplo el campo CodigoPostal que será un entero de 16 bits, para ello utilizamos la siguiente instrucción.

```
ALTER TABLE tblAlumno ADD CodigoPostal smallint;
```

Algo muy útil es que se puede agregar un campo a la tabla aún cuando ésta ya contenga tuplas, en este caso todas la tuplas que ya existían tendrán asignado automáticamente el valor de nulo en el campo CodigoPostal.

Convenientemente está prohibido usar la instrucción **ALTER** para agregar un atributo restringido a valores nulos.

El comando **DROP** es empleado para eliminar por completo una estructura de la base de datos, si empleamos **DROP** para eliminar una tabla ésta se borra permanentemente, el cambio es registrado en el diccionario de datos y el comando no se puede deshacer una vez efectuado.

La siguiente sentencia SQL elimina de la base de datos la tabla tblAlumno

DROP TABLE tblAlumno;

3.16 Definición de Vistas

Una vista es una relación o tabla virtual definida por la aplicación y construida a partir de una tabla real. Es pues una expresión de consulta sobre otras vistas o tablas.

Puesto que una base de datos tiene múltiples usuarios con diversos requerimientos y privilegios, resulta muy conveniente la creación de vistas de datos, que nos dan la oportunidad mirar el contenido de la base de una manera más conveniente que el diseño original. Combinando las vistas con la posibilidad de definir que usuarios acceden a esa vista, se obtiene un mecanismo muy poderoso para controlar los privilegios de los usuarios.

La vista más simple es la que se define por la aplicación de una selección y una proyección de una única tabla de la base.

Ejemplo:

CREATE VIEW Pediatra **AS SELECT** Nombre, Hospital
FROM Medico **WHERE** Especialidad = "pediatra";

Esta instrucción da como resultado la creación de una tabla virtual llamada Pediatra con dos atributos que son Nombre y Hospital, cuyos tipos de datos corresponderán a la tabla real Hospital. En esta nueva relación solamente contendrá las tuplas cuyo atributo Especialidad coincida con la cadena de texto "pediatra".

La correspondencia entre las tablas Hospital y Pediatra es dinámica, por lo que si insertamos, eliminamos o modificamos una tupla en la tabla hospital, donde el atributo Especialidad sea igual a pediatra, el cambio se verá reflejado en la tabla Pediatra.

Una vez definida una vista, a los ojos de los usuarios esta se ve y se comporta igual que una tabla real, en cuanto a consultas se refiere. También es posible definir una vista sobre más de una tabla.

Ejemplo:

CREATE VIEW Hospitales-Medicos (Hospitales, Medicos)

AS SELECT Hospital.nombre, Medico.nombre

FROM Hospital, Medico

WHERE Hospital.numero = Medico.Hospital;

Para esta caso ha sido necesario asignar nombres a los atributos de la vista Hospitales-Medico (Hospitales y Medicos), esto para evitar confusión puesto que en las tablas reales ambos atributos se llaman Nombre.

Como resultado de la vista anterior obtenemos una tabla con un par de nombres, que son el nombre del Hospital y el nombre del médico que labora en el mismo.

3.17 Control de seguridad

Para poder controlar el acceso a las vistas y tablas reales SQL provee dos comandos que son **GRANT** y **REVOKE** para poder ejecutar cualquier sentencia SQL sobre alguna tabla o vista el usuario debe de tener la autoridad necesaria.

Por ejemplo, para ejecutar la siguiente consulta:

SELECT * FROM tblAlumno;

Para poder ver el resultado de la consulta, el usuario debe de tener permiso de **SELECT** sobre la tabla tblAlumno. Para que sea posible actualizar las tuplas de una relación se requieren los permisos de **INSERT, DELETE O UPDATE.**

Para otorgar la autoridad de **SELECT** sobre la tabla tblAlumnos se haría de la siguiente forma:

GRANT SELECT ON TABLE tblAlumnos **TO** Gaby;

Después de la sentencia anterior, Gaby tiene ahora la autoridad para hacer consultas sobre la tabla tblAlumnos.

Esa misma instrucción puede ser revocada mediante el comando **REVOKE**

Ejemplo:

REVOKE SELECT ON TABLE tblAlumnos **TO** Gaby;

Una opción que nos da el lenguaje SQL es la de además de otorgar a los usuarios autoridad sobre nuestras vistas y tablas el privilegio de pasar esa autoridad a otros usuarios.

Ejemplo:

```
GRANT SELECT, UPDATE
ON TABLE tblAlumnos, tblCalificaciones to Gaby
WITH GRANT OPTION;
```

La sentencia anterior le otorga a Gaby no solo la autoridad de consultar y modificar las tablas tblAlumnos y tblCalificaciones, sino también le da poder para a su vez pasar estos privilegios a otros usuarios.

No podía hablar de bases de datos relacionales, sin mencionar el tema de SQL, que aún cuando visto de forma somera en este capítulo, creo que el lector de este trabajo puede quedarse con una idea bastante clara del uso de este lenguaje.

Todas las instrucciones que explico en este capítulo, forman parte del SQL estándar, por lo que es posible emplearlas para recuperar y modificar el contenido de cualquier base de datos relacional.

Hasta este momento, se han descrito características, objetivos e instrucciones de manipulación y creación de bases de datos relacionales, suponiendo siempre que la base de datos ya estaba creada, o bien se diseñaba empleando tan solo el sentido común, en el siguiente capítulo se describe una metodología para modelar una base de datos a un nivel conceptual y cuyo modelo es independiente del SGBD.

Dicho modelo, puede emplearse para describir el problema a un nivel muy alto de abstracción y en una segunda etapa, ser adaptado a un esquema relacional.

CAPÍTULO 4. METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS

En los capítulos anteriores se han descrito varios aspectos del modelo relacional y tratado los fundamentos de las bases de datos, hasta el momento hemos supuesto que los atributos que forman las relaciones son agrupados empleando el sentido común del diseñador de la base de datos. Sin embargo, es posible contar con una medida formal que nos permita comparar si una agrupación de atributos para formar una relación, es mejor que otra.

Este capítulo tiene la finalidad de describir una metodología para el diseño de bases de datos, misma que comienza en su etapa inicial con la descripción del problema del mundo real mediante un modelo Entidad/Relación, de ese modelo se espera obtener un conjunto de relaciones empleando un algoritmo de transformación ER– Modelo Relacional.

Finalmente serán aplicados los principios de normalización a las relaciones que resultan de la transformación, cuyo objetivo apunta a evitar la redundancia y las anomalías que podrían suscitarse al momento de la actualización de información.

Con el proceso de normalización, descomponemos una relación que no está en una cierta forma normal, produciendo múltiples esquemas de relación hasta lograr un diseño acorde a la forma normal deseada.

4.1 Primera etapa, modelado de datos con el enfoque Entidad – Relación (E/R)

La primera etapa de esta metodología inicia con la descripción del problema empleando el modelo Entidad/Relación (E/R), es empleado éste modelo por que ofrece un alto nivel de abstracción y nos permite captar muy bien la semántica del mundo real, cosa que facilita la labor del diseñador ayudándole en su comunicación con el usuario.

Como su nombre lo indica, el modelo E/R se basa en entidades que se relacionan entre si. Entendamos entidades como cualquier objeto que exista dentro del universo descrito por el problema a resolver, es todo aquel objeto del cuál queramos almacenar información en la base de datos, puede tratarse de una persona, un lugar, una cosa, un concepto o suceso, ya sea real o abstracto pero de interés para la empresa a la que se va a resolver el problema.

Otro concepto que debemos tener claro antes de comenzar modelar datos, es el de *tipo de entidad,* que es la estructura genérica de una entidad, mientras que a cada instancia creada a partir de un tipo de entidad, se le denomina *ocurrencia de entidad* o *instancia de la entidad.*

Podemos imaginar al *tipo de entidad* como un molde para hacer galletas, mientras que las *ocurrencias de entidad o instancias de la entidad* serían las galletas hechas con la forma del molde.

Por ejemplo: El tipo de entidad Alumno se refiere a la estructura que describe las características de cada alumno, mientras que una ocurrencia de Alumno será cada uno de los alumnos en concreto.

Existen dos tipos de entidades:

- Entidad Regular: Las ocurrencias de un tipo de entidad regular tienen existencia propia, es decir, existen por si mismas.
- Entidades Débiles: Son ocurrencias de un tipo de entidad que para poder existir necesitan de la existencia de una entidad regular de la cual son dependientes, por ello si desaparece la entidad regular, también desaparece la entidad débil.

Un tipo de entidad débil se representa en el modelo E/R con dos rectángulos concéntricos (Véase figura 4.1).

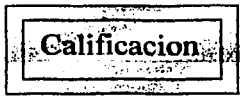


figura 4.1 Representación de un tipo de entidad

4.2 Interrelación

La interrelación es definida como la asociación o correspondencia entre entidades. A la estructura genérica del conjunto de interrrelaciones existentes entre dos o más tipos de entidad se le conoce como *tipo de interrelación.*

Un tipo de Interrelación contiene los siguientes elementos:

- Nombre: Por medio del cuál es identificada una interrelación de forma única. En la figura 4.2 el nombre de la interrelación parece como la etiqueta del rombo (Escribe).
- Grado: Este elemento indica el número de tipos de entidad que participan en la interrelación, en la figura 4.2 se puede observar que la relación Escribe es de Grado dos.

figura 4.2 Representación de un tipo de relación

Tipo de correspondencia: Número máximo de instancias que pueden derivar a partir de un tipo de entidad. El tipo de correspondencia es 1:1 (uno a uno) cuando en la interrelación solamente puede aparecer como máximo una instancia derivada de cada tipo de entidad. Pero si en cambio, de uno de los tipos de entidad pueden derivarse un número indefinido de ocurrencias o instancias del tipo de entidad, entonces el tipo de correspondencia será 1:N (uno a muchos). Y será un tipo de correspondencia N:M (muchos a muchos) si esto ocurre para ambos tipos de entidad (Véase figura 4.3).

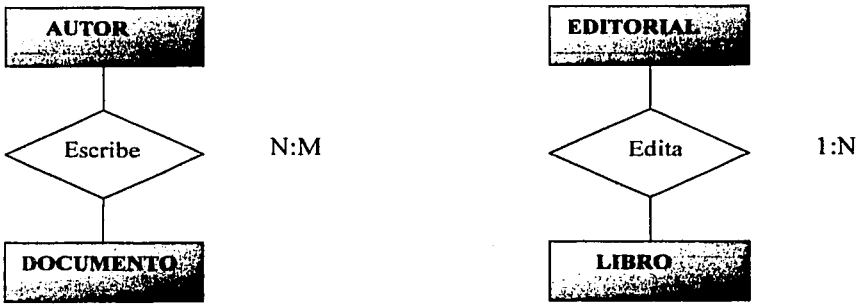


Figura 4.3 Representación de dos tipos de correspondencia

4.3 Atributo

Se le denomina atributo a cada una de las propiedades o características que tiene un tipo de entidad o de interrelación. Por ejemplo el tipo de entidad AUTOR tiene como atributos el Nombre, la Nacionalidad, Fecha de Nacimiento y en del tipo de entidad LIBRO podemos extraer los atributos Titulo, ISBN y Editorial. El atributo se representa con un óvalo en cuyo interior aparece el nombre del atributo, en ocasiones se suele poner el nombre del tributo fuera del óvalo.

De entre todos los atributos se debe de elegir alguno o varios que identificarán de manera unívoca cada una de las instancias creadas a partir de los tipos de relación.

Esos atributos serán llamados identificadores candidatos (Véase figura 4.4), y de los cuales se escoge uno como principal y los demás quedan como alternativos (Véase figura 4.5).

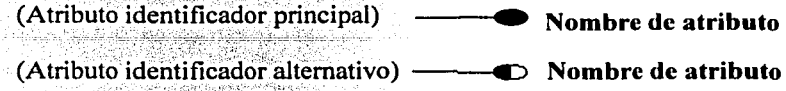


figura 4.4 Representación de atributos identificadores principal y alternativos

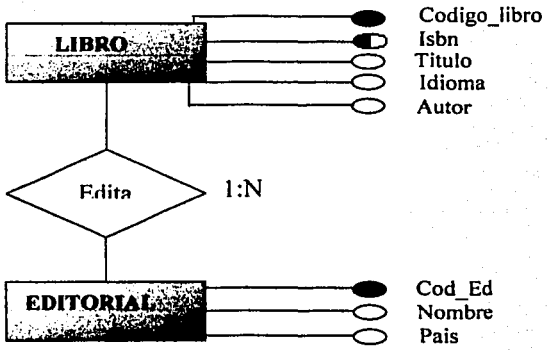


Figura 4.5 Representación de atributos de tipo entidad y de tipo interrelación

4.4 Cardinalidad en un tipo de entidad

La cardinalidad es el número de tuplas (renglones) que existen en una relación (tabla), en el modelo E/R es colocada una etiqueta del tipo (0,1), (1,1), (0,n) o (1,n), según corresponda, para indicar el número máximo y mínimo de instancias creadas a partir de ese tipo de entidad.

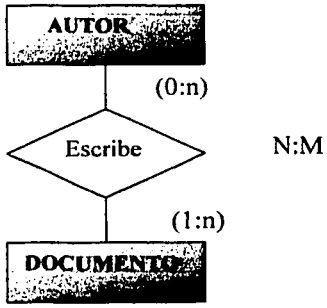


Figura 4.6 Ejemplo de cardinalidades máxima y mínima

Dicho términos más simples, esa etiqueta se coloca junto a cada tipo de entidad para especificar el número máximo y mínimo de renglones que derivarán de una interrelación.

En la figura 4.6, está ejemplificada la interrelación *Escribe*, en donde la cardinalidad (1,0) asociada al tipo de entidad *DOCUMENTO*, indica que una instancia *AUTOR* puede estar vinculada con 1,2,3,... o n instancias de *DOCUMENTO*, y la etiqueta (0,n) en *AUTOR* significa que una instancia de *DOCUMENTO* puede estar vinculada con 0,1,2,... o n instancias derivadas de *AUTOR*.

En otras palabras: Un autor escribe como mínimo 1 documento y como máximo n documentos, y un documento puede ser escrito por ningún autor (puede ser anónimo) o por muchos autores (n).

4.5 Dependencia en existencia y en identificación

Al igual que las entidades, las interrelaciones también son clasificadas en regulares y débiles, dependiendo del tipo de entidades que vinculan. Por lo que son llamadas interrelaciones regulares si asocian tipos de entidades regulares, y débiles si asocian un tipo de entidad débil con un tipo de entidad regular.
Una interrelación débil exige que las cardinalidades del tipo entidad regular sean (1,1).

Las interrelaciones débiles se dividen a su vez en dos tipos: Dependientes en existencia y dependientes en identificación.

- *Dependencia en existencia*: Una interrelación tiene dependencia en existencia, cuando vincula un tipo de entidad regular con un tipo débil, en este caso, las instancias derivadas de la entidad débil, no pueden existir sin las instancias creadas a partir de la entidad regular. Una dependencia en existencia se representa en el modelo E/R, agregando la etiqueta "E" al rombo que representa la interrelación débil.

Así por ejemplo, en la figura 4.7 se ejemplifica un tipo de interrelación con una dependencia en existencia, puesto que asocia el tipo de entidad regular *Empresa* con el tipo de entidad débil *Departamentos*.

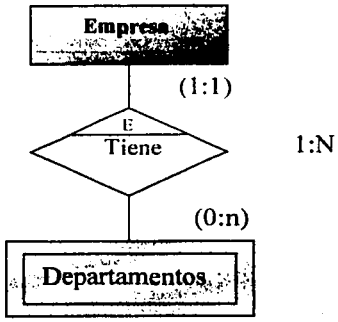


figura 4.7 Ejemplo de dependencia en existencia

En el caso de los datos concernientes a los departamentos, éstos solo tienen sentido si la empresa a la que pertenecen continúa en la base de datos. De ser necesario borrar los datos de una empresa, también se deben de borrar los datos de los departamentos que la componen, por no tener ningún sentido almacenarlos, eso es lo que hace que exista una dependencia en existencia.

- *Dependencia en identificación*: Existe una dependencia en identificación, cuando además de la dependencia en existencia, las instancias del tipo entidad débil, no se pueden identificar solo mediante sus propios atributos, sino que además se deben de agregar las claves de las instancias regulares de las cuáles dependen.

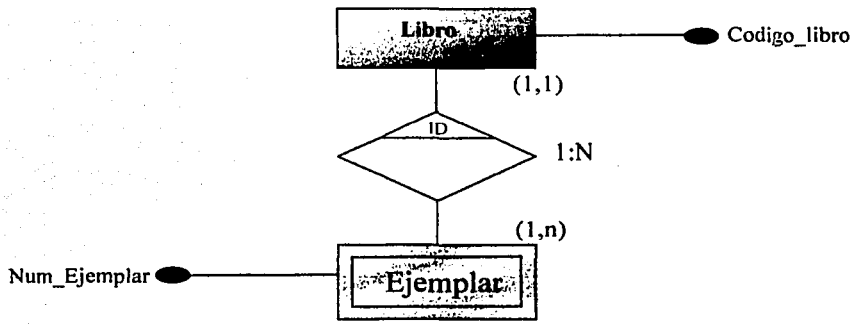


figura 4.8 Ejemplo de Dependencia en identificación

Como ejemplo tenemos la interrelación *Tiene* mostrada en la figura 4.8, que asocia al tipo de entidad regular *Libro* con el tipo de entidad débil *Ejemplar*, la razón por la que es dependiente en identificación, es que cualquier ejemplar, además de depender en existencia de un cierto libro, el atributo de identificación principal para cada ejemplar, está compuesto por el código del libro (Codigo_libro) mas el número del código propio (Num_Ejemplar).

4.6 Generalización y herencia en el modelo E/R

El modelo E/R permite representar relaciones jerárquicas, iguales a las que existen en el mundo real, creándose así una jerarquía de tipos de entidad, en donde existe un tipo de entidad que es un subtipo de otro tipo de entidad representada a un nivel de abstracción mayor. La generalización es una abstracción entre entidades a la que se le da el nombre de: **ES_UN.** (Véase figura 4.9)

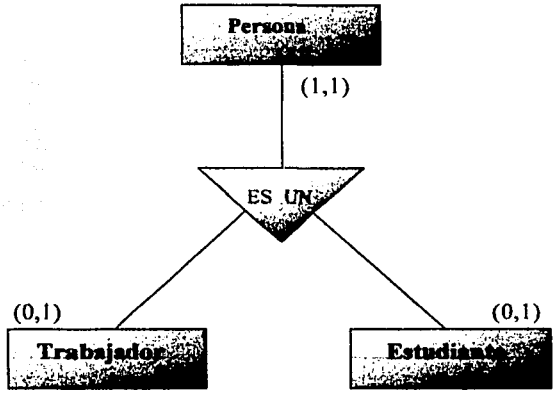


figura 4.9 Ejemplo de generalización

Es posible apreciar en la figura 4.9, que existe en esa jerarquía un supertipo (Persona) y que constituye la generalización de dos subtipos de entidades (Trabajador y Estudiante). Para la representación de este tipo de interrelación, se emplea un triángulo invertido, al cuál van conectados los subtipos.

Una característica importante de las jerarquías es la *herencia*, que permite que los atributos de un supertipo sean heredados por sus subtipos.

En la figura 4.9 se define un supertipo llamado *Persona,* del cuál se derivan dos subtipos: *Trabajador* y *Estudiante,* lo que esta representación quiere decir, es que tanto un trabajador como un estudiante son personas, por lo que los atributos de entidad *Trabajador* y *Estudiante* poseerán (heredarán) todos los atributos del tipo de entidad *Persona.*

En la generalización, los atributos comunes a los subtipos (incluidos los identificadores) se asignan al supertipo, mientras que los atributos específicos se asocian al subtipo correspondiente.

4.7 Segunda etapa, transformación del esquema conceptual al relacional.

Una vez que se han expuesto los preliminares del modelo E/R, estamos listos para comenzar a continuar los pasos de esta metodología de diseño de bases de datos, de la cuál se dijo que en su primera etapa, comprende la descripción del problema del mundo real, explicado en términos del modelo E/R

La primera etapa tiene la finalidad de elaborar el Diseño conceptual, en donde el objetivo perseguido es obtener una buena representación del problema, representando los recursos de información de la empresa y fuera de toda consideración sobre eficiencia de la computadora. Para elaborar este diseño conceptual, nos basaremos en el modelo E/R.

Una vez obtenido el esquema conceptual, iniciamos la segunda etapa de nuestra metodología, en donde buscaremos obtener el esquema lógico, lo anterior se hará transformando el esquema conceptual de manera que se adapte al modelo de datos relacional.

La transformación de un esquema E/R al esquema relacional está basado en los siguientes principios:

- Todo tipo de entidad se convierte en relación.
- Todo tipo de interrelación N:M se transforma en una relación.
- Todo tipo de interrelación 1:N se traduce en el fenómeno de propagación de clave o bien se crea una nueva relación.

A continuación se presentan una serie de reglas para la transformación de los esquemas conceptuales, en este caso E/R a objetos válidos en los esquemas lógicos relacionales. Estas reglas pretenden la transformación de un esquema conceptual a un tipo relacional sin pérdida de información, conservando el nivel de representación del problema.

La aplicación de las reglas se hace dependiendo del tipo de objeto del esquema E/R que se va a transformar, y de la cardinalidad de las relaciones que los objetos mantienen con otros objetos en el esquema conceptual.

4.7.1 Transformación de tipos de entidad

Regla 1. Todos los tipos de entidad presentes en el esquema conceptual, se transforman en relaciones (tablas) en el esquema relacional, pero conservando cada uno de sus atributos y los tipos de los mismos, así como la característica de identificador de estos atributos.

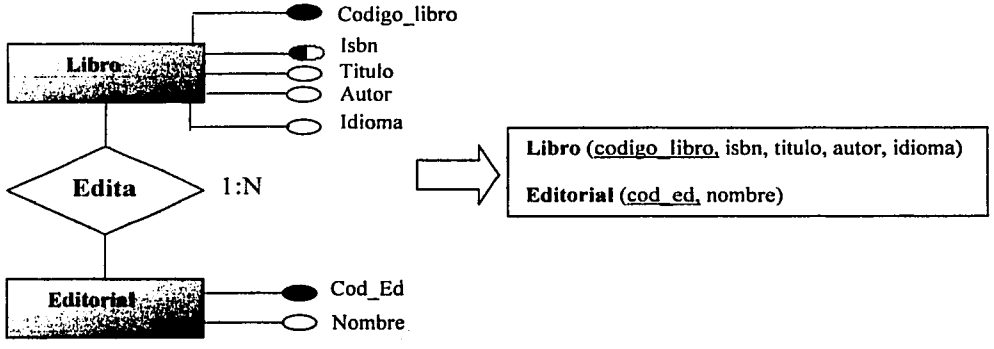


figura 4.10 transformación de tipos de entidad a Relaciones

En el ejemplo de la figura 4.10 se ha aplicado la primera regla de transformación dando como resultado las relaciones Libro y Editorial, note que aún falta una transformación, en este caso falta transformar la interrelación *Edita.* Este tipo de transformación se verá mas adelante.

4.7.2 Transformación de tipos de interrelación uno a uno

El proceso de transformación de los tipos de interrelación binarias, va a depender del valor de la cardinalidad mínima con la que participa alguno de los tipos de entidad, basándose en ese valor, son presentadas las siguientes reglas de conversión:

Regla 2. Si los dos tipos de entidad participan con cardinalidad máxima y mínima igual a uno, entonces:

a) Si los dos tipos de entidad tienen el mismo identificador:

   1. Los dos tipos de entidad se transforman en una única tabla formada por la agregación de los atributos de los dos tipo de entidad.

2. La clave de la tabla es el identificador de los tipos de entidad (es el mismo en ambas)

b) Si los tipos de entidad tienen diferente identificador, cada tipo de entidad se transforma en una tabla y además:

1. Cada tabla tendrá como clave principal el identificador de cada uno de los tipos de entidad de los cuales se deriva.

2. Cada tabla tendrá como clave foránea el identificador del otro tipo de entidad con el cuál está relacionado.

c) Si los dos tipos de entidad tienen la misma clave, pero uno de ellos es un tipo de entidad débil, entonces se procede de algunas de las dos formas expuestas anteriormente, dependiendo de la conveniencia para los requerimientos funcionales.

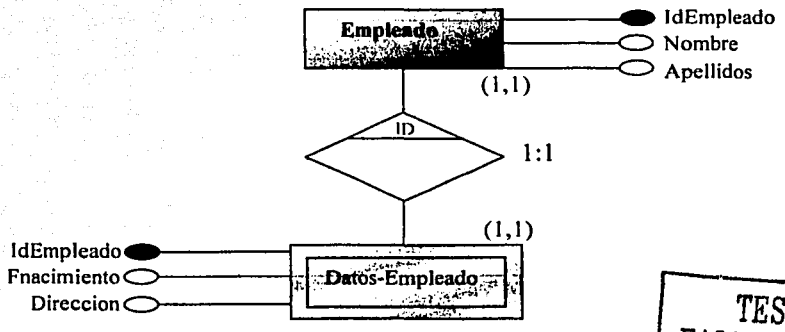


figura 4.11 Ejemplo de relaciones binarias

En el caso expuesto en la figura 4.11, se puede observar que se trata de un tipo de relación débil por identificación, por lo que no habrían, en este caso, datos personales de empleados, ni se podrían identificar, si no existieran primero empleados.
Aplicando la regla dos obtenemos el siguiente esquema relacional:

Empleado (IdEmpleado, Nombre, Apellidos, Fnacimiento, Direccion)

Puede darse el caso de que por motivos de procesamiento, el diseñador requiera la existencia de ambas tablas y no la composición que resultó en el esquema anterior, de acuerdo con la misma regla 2, el esquema resultante sería el siguiente:

Empleado (IdEmpleado, Nombre, Apellidos)

Datos-Empleado (IdEmpleado, Fnacimiento, Direccion)

Regla 3. Si en un tipo de interrelación binaria alguno de los tipos de entidad participa de forma parcial, es decir, con cardinalidad mínima cero, entonces cada tipo de entidad se transforma en una tabla y se procede de alguna de las siguientes formas:

1. El identificador del tipo de entidad que participa parcialmente, pasa a formar parte de los atributos de la tabla correspondiente a la transformación del otro tipo de entidad y podrá tomar valores nulos. Si se hace de esta forma, entonces no se generará ninguna tabla para el tipo de interrelación.

2. Se construye una nueva tabla correspondiente al tipo de interrelación formada por los dos atributos identificadores de cada entidad. El atributo que será considerado como clave de esa tabla al identificador del tipo de entidad que participa de forma parcial en el tipo de interrelación. Los atributos de esta tabla mantendrán referencias con las claves de las tablas correspondientes a la transformación de los tipos de entidad.

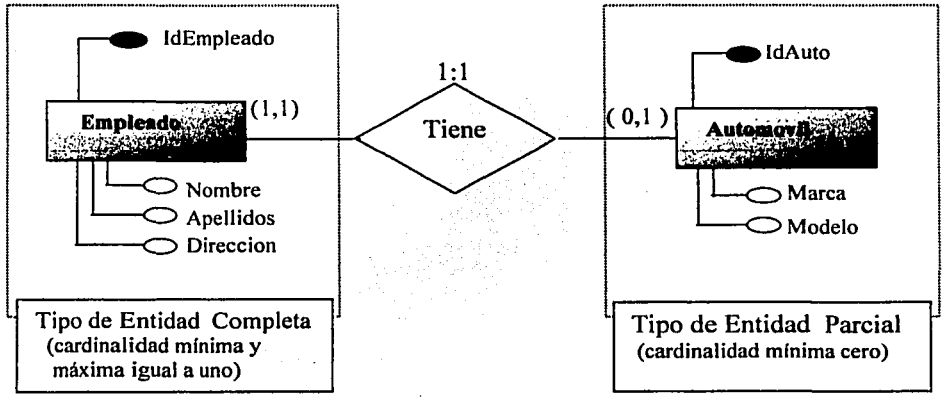


figura 4.12 Ejemplo de Relación Binaria 1:1

La relación ejemplificada en la figura 4.12, representa la abstracción de un supuesto en el que una empresa le asigna a algunos empleados, no a todos, un automóvil prestado, todos los autos son asignados y lo que se desea es saber ¿qué empleado, tiene asignado que auto?.

A continuación aplicamos a este esquema E/R, la regla 3 para convertirlo a un esquema relacional, obteniendo cualquiera de estos dos resultados:

1. No se construye una tabla para el tipo de interrelación, entonces:

Empleado (IdEmpleado, Nombre, Apellidos, Direccion)

Automovil (IdAuto, Marca, Modelo, IdEmpleado)

El identificador del tipo entidad *Empleado* pasa a formar parte de la tabla correspondiente a la transformación del tipo de entidad *Automovil.*

2. Se construye una tabla correspondiente a la transformación del tipo de interrelación:

Empleado (IdEmpleado, Nombre, Apellidos, Direccion)

Automovil (IdAuto, Marca, Modelo, IdEmpleado)

TieneAuto (IdAuto, IdEmpleado)

En esta segunda propuesta la clave será el identificador del tipo de entidad Automovil por participar de forma parcial en el tipo de interrelación e IdEmpleado queda solo como clave alterna.

Analizando las dos opciones anteriores, es posible llegar a la conclusión de que la primera transformación resulta más favorable, porque no existirán valores nulos en ninguna de las dos tablas. El atributo *IdEmpleado* que forma parte de la tabla *Automovil* siempre tendrá un valor, puesto que todos los automóviles son asignados a algún empleado.

La segunda transformación, en donde se crea una tabla que mantiene la relación entre los dos tipos de entidad, presenta el inconveniente de un esquema más grande que no por eso aporta una mejor representación del problema.

Regla 4. Si en un tipo de interrelación binaria *ambos* tipos de entidad participan de forma parcial, es decir, con cardinalidad mínima cero, entonces cada tipo de entidad se transforma en una tabla y se procede de alguna de las siguientes formas:

1. Los identificadores de cada uno de los tipos de entidad pasan a formar parte de los atributos de las tablas correspondientes al otro tipo de entidad. Ambos atributos actuarán como claves foráneas en esas tablas.

2. Se construye una nueva tabla correspondiente al tipo de interrelación formada por los dos atributos identificadores de cada entidad. El atributo identificador de esa tabla será el identificador de uno de los tipos de entidad y necesariamente se definirá como clave alterna el atributo identificador del otro tipo de entidad.

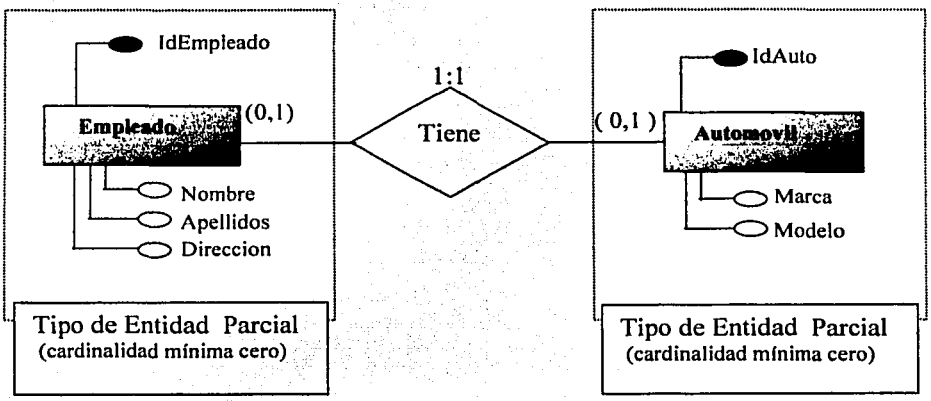


figura 4.13 Ejemplo de Relación Binaria 1:1

La relación ejemplificada en la figura 4.13, retoma el ejemplo anterior, pero para exponer la regla 4, daremos a ese problema una pequeña variación, entonces supongamos pues que una empresa le asigna a algunos empleados, no a todos, un automóvil prestado, pero *no* todos los autos son asignados y lo que se desea es saber ¿qué empleado, tiene asignado que auto?.

Aplicando la regla 4 obtenemos el siguiente esquema relacional:

Empleado (IdEmpleado, Nombre, Apellidos, Direccion, **IdAuto)**

Automovil (IdAuto, Marca, Modelo, **IdEmpleado)**

También podría haberse aplicado el criterio 2 de la regla 4 y construir la tabla correspondiente al tipo de interrelación *TieneAuto,* de manera que obtendremos el siguiente esquema:

Empleado (IdEmpleado, Nombre, Apellidos, Direccion)

Automovil (IdAuto, Marca, Modelo, IdEmpleado)

TieneAuto (IdAuto, IdEmpleado)

Para este ejemplo se ha tomado la decisión de que el atributo IdEmpleado sea la clave principal y el atributo IdAuto la clave secundaria de la tabla *TieneAuto,* pero bien pudo haberse tomado el criterio opuesto:

Empleado (IdEmpleado, Nombre, Apellidos, Direccion)

Automovil (IdAuto, Marca, Modelo, IdEmpleado)

TieneAuto (IdEmpleado, IdAuto)

4.7.3 Transformación de tipos de interrelación uno a muchos

A continuación se dictan los pasos a seguir en caso de tener esquemas conceptuales en los que estén presentes tipos de interrelaciones binarias donde un tipo de entidad participa con cardinalidad máxima uno y el otro tipo de entidad participa con cardinalidad máxima muchos, al igual que en los casos anteriores se aplican reglan en función de las cardinalidades mínimas con las cuales participa cada tipo de entidad en el tipo de interrelación.

> Regla 5. Si en un tipo de interrelación binaria 1:N (uno a muchos) ambos tipos de entidad participan de forma completa, o el tipo de entidad que interviene con cardinalidad máxima muchos participa de forma parcial, entonces, cada tipo de entidad se transforma en una tabla y el identificador del tipo de entidad de entidad que participa con cardinalidad máxima uno pasa a formar parte de la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima muchos. Este atributo será definido como clave foránea de esta tabla manteniendo una referencia con la tabla correspondiente al tipo de entidad que participa con cardinalidad máxima uno.

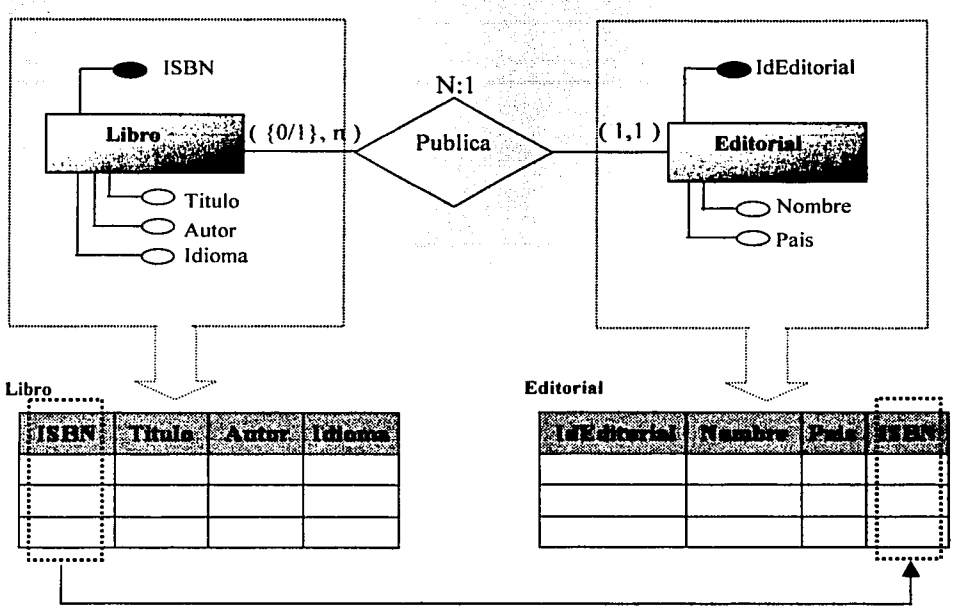


figura 4.14 Relación binaria uno a muchos 1:N

La figura 4.14 muestra la relación 1:N (uno a muchos) que existe entre los tipos de entidad Libro y Editorial. En este esquema conceptual, se representa la relación existente en el mundo real entre una editorial y los libros que esta edita. Una editorial puede publicar de cero hasta muchos libros, mientras que un libro solo puede ser publicado por una editorial. Aplicando la regla 5 obtenemos el siguiente esquema relacional:

Libro (ISBN, Titulo, Autor, Idioma)

Editorial (IdEditorial, Nombre, Pais, ISBN)

Regla 5.1 Si en un tipo de interrelación binaria 1:N (uno a muchos) ambos tipos de entidad participan de forma parcial, o el tipo de entidad que interviene con cardinalidad máxima uno participa de forma parcial, entonces, cada tipo de entidad se transforma en una tabla y se genera una tabla correspondiente al tipo de interrelación. Esta tabla estará formada por los identificadores de los tipos de entidad que intervienen en el tipo de interrelación, esos atributos serán definidos como claves foráneas y la clave principal de esta tabla será el atributo identificador correspondiente al tipo de entidad que interviene con cardinalidad máxima muchos.

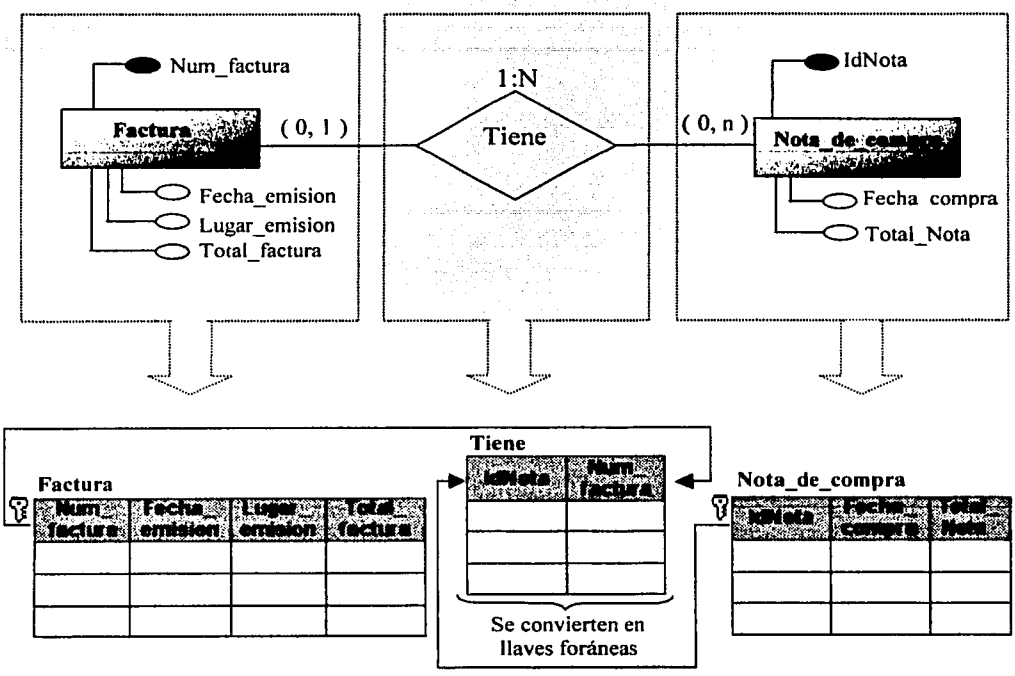


figura 4.15 Ejemplo de Relación binaria uno a muchos 1:N. Regla 5.1

El ejemplo de la figura 4.15 representa una relación existente entre los tipos de entidad *Factura* y *Nota_de_compra.* Esta interrelación indica que existen cero o muchas facturas expedidas por Nota de compra, mientras que una nota de compra es facturada en ninguna o una sola factura.

Es decir, que la lógica de negocio de la empresa representada en la figura 4.14 admite emisión de notas de compra sin factura, también es posible emitir facturas sin la realización previa de una nota de compra, pero una nota de compra sólo puede ser parte de una factura, nunca de dos.

Si aplicamos a este esquema conceptual la regla 5.1 obtendremos el siguiente esquema relacional:

Factura (Num_factura, Fecha_emision, Lugar_emision, Total_factura)

Nota_de_compra (IdNota, Fecha_compra, Total_nota)

Tiene (IdNota, Num_factura)

4.7.4 Transformación de tipos de interrelación muchos a muchos

El proceso de transformación al esquema relacional en este tipo de interrelaciones no está condicionado por la cardinalidad mínima, por lo que solamente es necesario seguir la siguiente regla para efectuar la transformación de un esquema al otro.

Regla 6. En un tipo de interrelación binaria N:N (muchos a muchos) cada tipo de entidad se transforma en una tabla y se genera una nueva tabla para representar al tipo de interrelación. Esta tabla estará formada por los identificadores de los tipos de entidad que intervienen en el tipo de interrelación y por todos atributos asociados al tipo de interrelación. La clave principal de esta tabla estará integrada por la agregación de los atributos identificadores correspondientes a los tipos de entidad que intervienen en el tipo de interrelación.

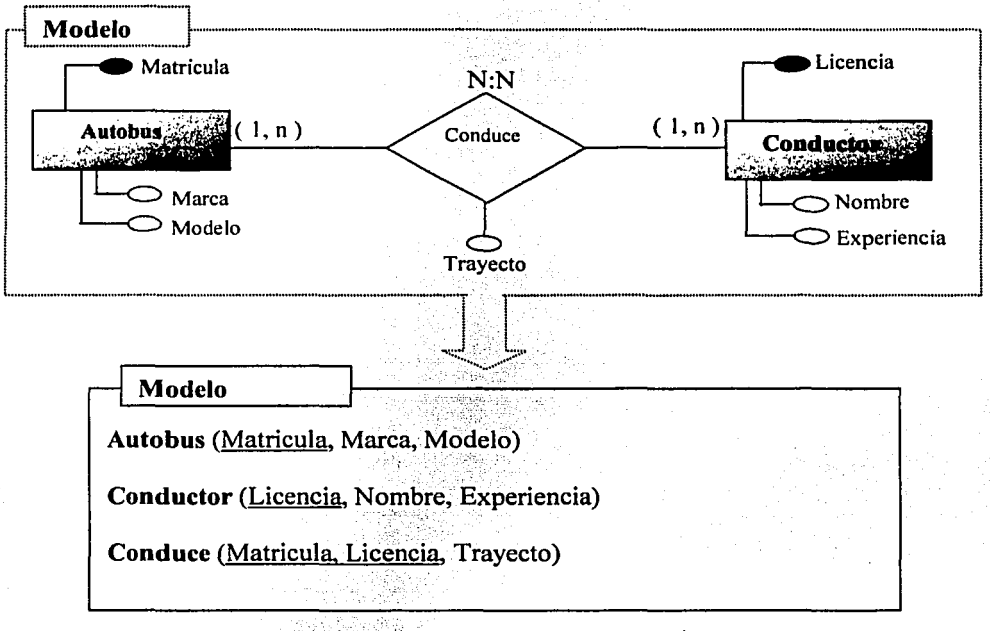


figura 4.16 Ejemplo de Relación binaria muchos a muchos N:N. Regla 6

La figura 4.16 muestra un ejemplo de conversión de un modelo E/R al modelo relacional. En este caso, se muestra una interrelación muchos a muchos derivada de un planteamiento en el cuál se existe una relación entre los autobuses y los conductores de los mismos, en la que un conductor puede conducir varios autobuses y un autobús puede ser conducido por varios conductores. El tipo de interrelación entre los tipos de entidad

Autobus y Conductor tiene asociado un atributo *Trayecto* que representa el recorrido realizado por un conductor con un autobús determinado.

Note que en la relación derivada llamada *Conduce* se ha definido como clave compuesta la formada por la agregación de los atributos identificadores de los tipos de entidad que participan en el tipo de interrelación.

4.8 Tercera etapa, aplicación de la teoría de normalización

El proceso de normalización es una propuesta hecha por el Dr Codd en el año de 1972, en donde se somete un esquema relacional a una serie de pruebas para certificar si dicho esquema pertenece o no a una cierta *forma normal,* la finalidad de esa forma normal es el conseguir una redundancia mínima en los datos que serán almacenados, así como evitar anomalías al momento de actualizar los datos en la base.

Cuando se implementa una base de datos sin realizar previamente el proceso de normalización, se vuelve muy ardua la tarea de realizar cambios en la estructura de la base, pues suelen presentarse incongruencias entre los datos, estas incongruencias rompen con la integridad[22] en la base de datos, y además de ser difíciles de identificar son difíciles de corregir.

El Dr. Codd propuso tres formas normales, a las cuales llamó primera, segunda y tercera forma normal. Posteriormente, junto con Boyce propusieron una definición más estricta de la tercera forma normal, llamada forma normal de Boyce-Codd.

Todas estas formas normales se basan en las dependencias funcionales entre los atributos de una relación. El proceso de normalización puede entenderse como un proceso mediante el cuál los esquemas de relación insatisfactorios se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que poseen propiedades deseables.

Más adelante se propusieron una cuarta forma normal y una quinta forma normal, dirigiendo estas reglas a las dependencias multivaluadas y dependencias de reunión.

El método de diseño de bases de datos propuesto en este trabajo, solo contempla hasta la tercera forma normal, ya que no es necesario llegar hasta formas normales más altas para conseguir un buen modelado de bases de datos, incluso la utilidad práctica de las formas normales, puede quedar en entre dicho cuando las restricciones en las que se basan son muy difíciles de detectar o de entender, y dado que ya en la tercera forma normal se obtiene el grado de normalización necesario y en la mayoría de los casos el deseable, aplicaremos solo la primera, segunda y tercera forma normal a nuestro modelo relacional derivado de la primera y segunda etapa de esta metodología.

[22] La integridad se refiere a que las reglas del negocio deben de cumplirse en la base de datos, de tal manera que los datos almacenados en la base, reflejen la realidad de la empresa.

4.8.1 Primera Forma Normal (1FN).

La primera forma normal establece que los dominios de los tributos deben incluir sólo valores atómicos, es decir, valores simples e indivisibles y que el valor en una tupla debe ser un valor individual proveniente del dominio de ese atributo.

Una relación R satisface la primera forma normal si, y sólo si, todos los dominios subyacentes de la relación R contienen valores atómicos.

La aplicación de esta regla consiste en descomponer aquellas tuplas en donde los atributos tengan más de un valor, en tantas tuplas como valores estén presentes.

Cuando una base de datos cumple con la primera forma normal, se puede establecer con mayor facilidad una clave primaria, misma que permite identificar a cada registro de forma única dentro de una misma tabla.

Una tabla con este primer nivel de normalización hace más sencillas las tareas de búsqueda y de cálculo. Ya que sólo se tendría que operar en un campo específico y no en varios con el mismo tipo de información.

Alumnos

IdAlumno	Nombre	Materias	Calificaciones
91291190	Juan	Física, Química	8,9
40004185	Christian	Química, Biología, Física	10,9,10

figura 4.17 Ejemplo de Relación sin normalizar

En este ejemplo (ver figura 4.17) tenemos una relación llamada *Alumnos,* la cuál no se encuentra en primera forma normal, debido a que los atributos *Materias* y *Calificaciones* no contienen valores atómicos, es decir, estos atributos tienen más de un valor.

Para solucionar este problema, se crea una nueva relación vinculada a la relación original mediante una relación 1:N

Alumnos

IdAlumno	Nombre
91291190	Juan
40004185	Christian

Materias

IdAlumno	Materia	Calificación
91291190	Física	8
91291190	Química	9
40004185	Química	10
40004185	Biología	9
40004185	Física	10

figura 4.18 Ejemplo de Relación normalizada a primera forma normal

4.8.2 Segunda Forma Normal (2FN)

Una relación R satisface la segunda forma normal si, y solo si, satisface la primera forma normal y cada atributo de la relación depende funcionalmente de forma completa de la clave primaria de esa relación.

La segunda forma normal se aplica en tablas que tengan claves primarias compuestas, o sea claves formadas por dos o más atributos. En tablas con ésta característica, se deben quitar los atributos (campos) que no dependan totalmente de la clave primaria y crear una o varias tablas con los atributos excluidos, dejando únicamente los datos dependientes. Es decir, debemos verificar que todos los atributos dependan completamente de la clave primaria.

A continuación, analizaremos un modelo relacional que no se encuentra en segunda forma normal, como podemos ver, nos encontramos con dos relaciones (Cuenta y Banco), si observamos la relación *Cuenta,* podremos notar que los atributos Balance y Fecha_de_Apertura dependen de la clave primaria IdCuenta, por otro lado, el atributo Direccion_Banco es independiente de la clave primaria, puesto que no depende del atributo IdCuenta, pero si depende de la clave primaria IdBanco (*ver figura 4.19 inciso A*), entonces, la acción lógica a realizar sería mover el atributo Direccion_Banco a la relación Banco, como se muestra en la figura 4.19 inciso B.

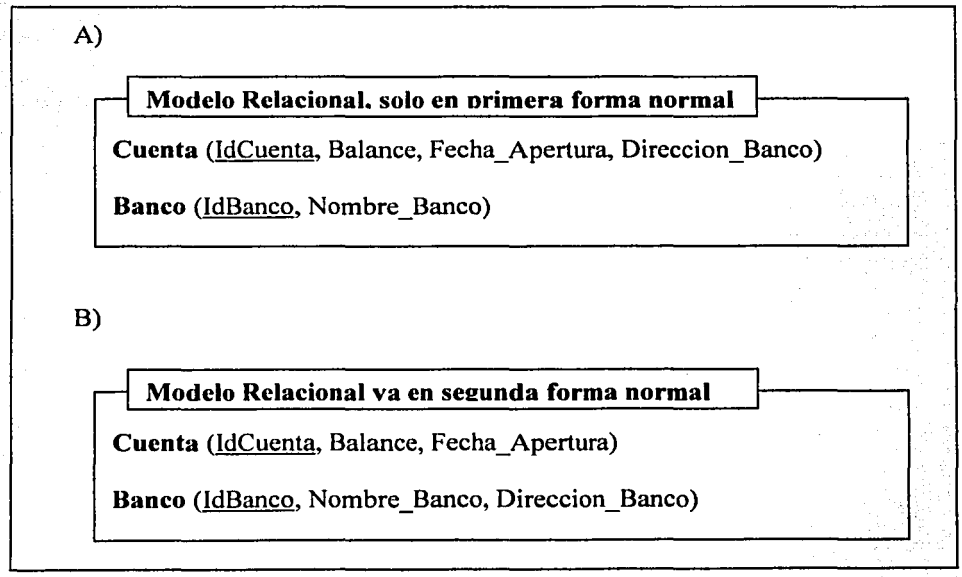


Figura 4.19 ejemplo de normalización (2FN)

Como ejemplo, supondremos que se han creado las tablas Cuenta y Banco, y que estas dos relaciones cuentan ya con algunos registros:

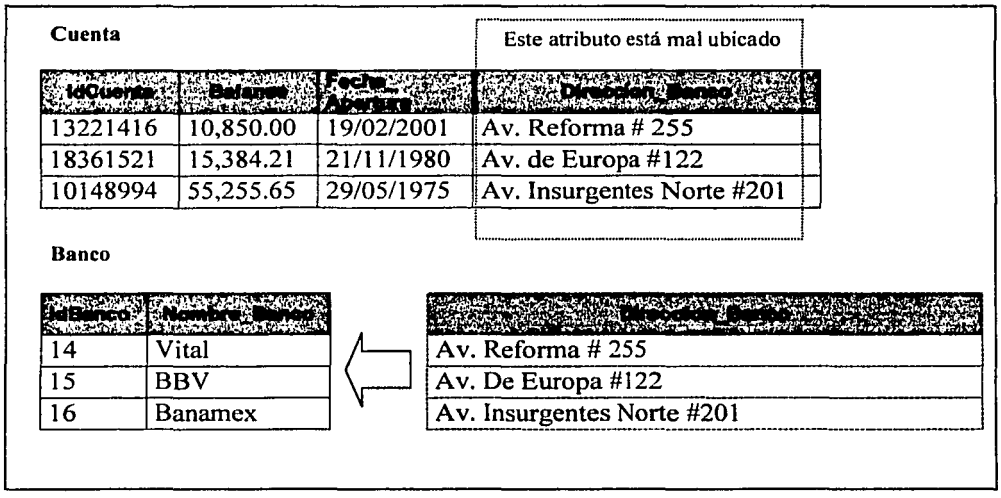


Figura 4.20 ejemplo de normalización (2FN)

El atributo de localización del banco está mal ubicado, puesto que la dirección del banco no depende del número de cuenta, por lo tanto ese no debe ser un atributo de la tabla Cuenta, para que las relaciones queden en segunda forma normal, se tiene que mover el atributo Direccion_Banco a la relación Banco.

A esto se refiere la regla de normalización (FN2) cuando dicta que cada atributo de la relación debe depender funcionalmente de forma completa de la clave primaria de esa relación. Por lo que si un atributo no es dependiente del la clave primaria completamente, está fuera de lugar y deberá ser movido.

4.8.3 Tercera Forma Normal (3FN).

Una relación R satisface la tercera forma normal si, y solo si, satisface la segunda forma normal y cada atributo no primo de la relación no depende funcionalmente de forma transitiva de la clave primaria de esa relación. Es decir, no pueden existir dependencias transitivas entre los atributos que no forman parte de la clave primaria de la relación R.

Esta forma normal indica que no debe existir una relación estrecha entre los campos o atributos de una tabla (no debe haber una dependencia transitiva entre los campos).

Si un atributo depende de otro que no es una clave primaria, entonces es necesario mover ambos a una nueva entidad y esa nueva entidad deberá estar relacionada con la entidad original.

Ejemplo:

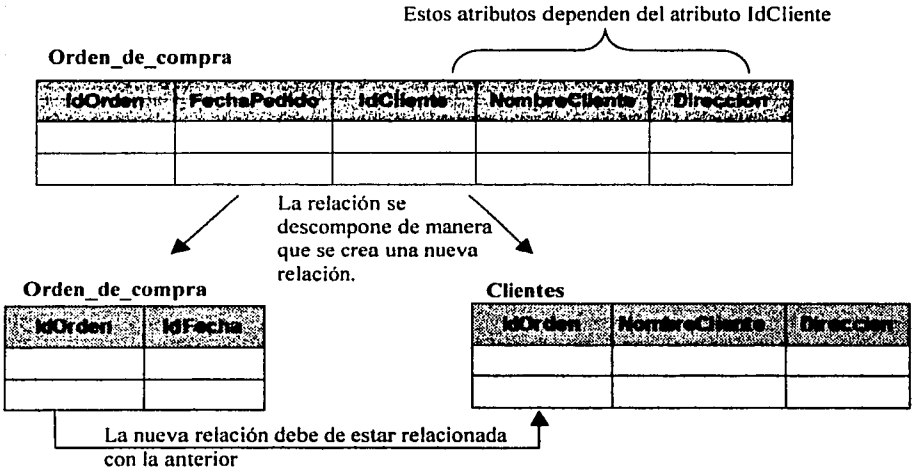


Figura 4.21 ejemplo de normalización (3FN)

En el ejemplo expuesto en la figura 4.21, es necesario descomponer la relación *Orden_de_compra* porque los atributos *NombreCliente* y *Dirección* dependen del atributo *IdCliente* y ese atributo no es una clave principal, por lo que se crea una nueva relación llamada *Clientes* que mantiene una relación con la tabla anterior por medio de la clave IdOrden.

Existen numerosas metodologías para diseñar bases de datos, la expuesta anteriormente, es conocida como Top Down porque parte de una idea general, que es la creación de un modelo conceptual, para después llegar a un modelo particular, que en este caso se trata del modelo relacional.

Esta metodología es parte de un proceso de diseño de base de datos que se compone de seis fases, mismas que veremos en su totalidad en el capitulo siguiente, esta introducción, se debe a que en el siguiente capítulo dedicaré más espacio al análisis del problema para la creación del esquema conceptual, que a la conversión del esquema conceptual en esquema relacional, razón por la cuál fue explicada dicha conversión con antelación en este capítulo.

CAPÍTULO 5. IMPLEMENTACIÓN DEL SISTEMA DE BASE DE DATOS PARA EL MANEJO DE PROPUESTAS EN LA CARRERA DE INGENIERÍA EN COMPUTACIÓN.

El presente trabajo de tesis, ha sido encaminado para apoyar un primer acercamiento a las bases de datos, así como a las técnicas relativas a la creación y modelado de las mismas, éste capítulo lo pensé en un inicio, como una metodología a seguir para implementar un sistema que utilizara una base de datos (de ahí el título del capítulo), describiendo en general los pasos a seguir desde al análisis de factibilidad, hasta la validación y prueba de aceptación del sistema.

Sin embargo, he preferido no concentrar la atención en el ciclo de vida del sistema de información y enfocarme al diseño e implementación de bases de datos.

La razón, es que deseo un trabajo orientado de principio a fin a las bases de datos y en específico a las bases de datos relacionales.

No es mi intención restar importancia a los pasos previos o subsecuentes al diseño de las bases de datos, de hecho, no es posible un buen diseño de la base de datos sin una adecuada recolección y análisis de requerimientos. Y por supuesto, la razón de diseñar una base de datos, es la de implantarla y validar la satisfacción de los requerimientos para los que fue diseñada.

Es por eso que comenzaré describiendo brevemente las fases de las cuales se compone el ciclo de vida de un sistema de información, para luego, dedicar lo que resta del capítulo a aplicar una metodología completa pero orientada al diseño de bases de datos.

La base de datos que utilizo de ejemplo para ilustrar los pasos del modelado de bases de datos, es parte de un sistema propuesto para resolver un problema del mundo real: El sistema de base de datos para el manejo de propuestas en la carrera de ingeniería en computación, que se encuentra al momento de escribir este trabajo en la fase de operación.

5.1 Ciclo de vida de un Sistema de Información

Un sistema de bases de datos es parte complementaria de un sistema mucho mayor que es el sistema de información, el sistema de información es el que a final de cuentas controla los recursos de información en una organización.

Un sistema de información engloba todos los recursos que dentro de la organización se encargan de la recolección, administración y uso de la información, si el sistema de información emplea mecanismos computarizados, entonces el SGBD, el hardware empleado, el personal que usa y administra los datos, etc. todo esto queda incluido en el universo del sistema de información, de tal manera que el sistema de base de datos es solo una parte del sistema de información (Véase figura. 5.1).

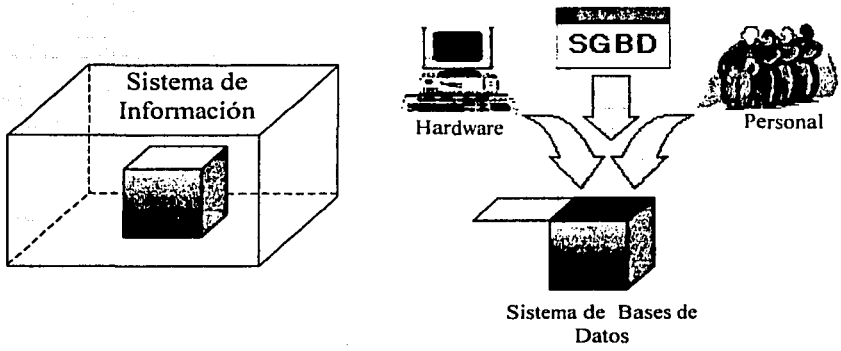


Figura 5.1 Sistema de Aplicación de Bases de Datos dentro de un Sistema de Información.

El ciclo de vida de un sistema de información incluye por lo general las siguientes fases:

1. *Análisis de factibilidad*: Se ocupa de analizar las posibles áreas de aplicación, en esta fase se llevan a cabo estudios preliminares en donde se comparan los costos contra los beneficios, de igual manera se establecen prioridades entre las aplicaciones que se contemplan desarrollar dentro del sistema.

2. *Recolección y análisis de requerimientos*: En esta fase se obtienen los requerimientos detallados de propia mano de los posibles usuarios, con la finalidad de identificar sus problemas y necesidades específicas.

3. *Diseño*: Esta fase está formada por dos partes, por un lado se debe trabajar sobre el diseño de la bases de datos y por el otro se trabaja sobre el diseño de los sistemas de aplicación, es decir, los programas que usan y almacenan la información contenida en la base de datos.

4. *Implementación*: En esta fase el sistema de información se instala, la base de datos se carga y se hacen las pruebas de transacciones en la base de datos, esta fase termina con el programa puesto en marcha y con las pruebas preliminares llevadas a cabo satisfactoriamente.

5. *Validación y prueba de aceptación*: Se valida el sistema en cuanto a los requerimientos establecidos en la fase dos, el sistema es probado contra los requerimientos y las especificaciones de comportamiento que se definieron junto con los futuros usuarios.

6. *Operación*: Esta fase operativa comienza después de que todas las funciones han sido probadas y validadas, el proceso de operación, viene acompañado de un monitoreo periódico al sistema para supervisar el rendimiento y estar preparados para dar mantenimiento en caso de posibles fallas. Es posible que en esta fase se requieran modificaciones, en cuyo caso se debe de pasar de nuevo por todas las etapas anteriores.

5.2 Ciclo de vida del sistema de base de datos.

El ciclo de vida del sistema de base de datos o sistema de aplicación de base de datos, son los pasos que tienen lugar dentro de la fase de diseño e implementación de un sistema de información, mismos que se hacen a la par del diseño e implementación de los programas que procesan y utilizan la información contenida en la base de datos.

Las fases por las que atraviesa del llamado ciclo de vida del sistema de aplicación de base de datos son las que enumero a continuación:

1. *Definición del sistema*: En esta fase se define el alcance del sistema de base de datos, sus usuarios y sus aplicaciones.

2. *Diseño*: En esta fase se pretende elaborar el diseño lógico completo del sistema de base de datos, obviamente, en esta fase se necesita elegir un modelo de datos que nos ayude a describir los datos que deseamos almacenar en la base, en este caso se empleará el modelo de datos relacional, que fue descrito con antelación.

3. *Implementación:* La fase de implementación, comprende el proceso de describir las definiciones conceptual, externa e interna de la base de datos, crear los archivos de la base de datos e instalar el software de aplicación.

4. *Carga o conversión de los datos:* Una vez que el sistema ha sido instalado y la base de datos está lista para recibir la información, entonces se comienza a alimentar a la base, ya sea cargando los datos directamente o importando unos ya existentes.

5. *Conversión de aplicaciones:* De existir software que se usaba en un sistema anterior y sea requerido ahora, entonces se procede a adaptarlo para que funcione con el nuevo sistema también.

6. *Prueba y validación*: En esta fase el nuevo sistema es probado y se valida para corroborar que satisface los requerimientos para los que fue diseñado.

7. Operación: El sistema y todas las aplicaciones que formen parte del mismo se ponen en operación.

8. *Supervisión y mantenimiento*: Inmediatamente después de que el sistema se pone en operación, comienza la fase de supervisión, conforme se verifique el buen funcionamiento del sistema, la supervisión al mismo será cada vez menor.

Ahora bien, el proceso de diseño de bases de datos que describiré en este capítulo, está contenido dentro del paso número dos del ciclo de vida del sistema de aplicación de bases de datos. Siendo un subproceso que está encaminado a diseñar la estructura lógica y física de una o más bases de datos para atender las necesidades de información que requieren lo usuarios de una organización.

El proceso de diseño de una base de datos va encaminado a satisfacer los requerimientos de información de los usuarios, lo que obliga a proveer una estructuración de la información natural y fácil de entender.

Este proceso de diseño de la base de datos, arroja como resultado un esquema cuya modificación resultará bastante difícil una vez implementada la base de datos.

Son seis la fases principales que componen a este proceso de diseño de base de datos :

1. *Recolección y análisis d requerimientos.* En esta fase se define el alcance del sistema de base de datos, sus usuarios y sus aplicaciones.

2. *Diseño conceptual de la base de datos.*

3. *Elección de un SGBD.*

4. *Transformación a modelo de datos (llamado también diseño lógico de la base de datos).*

5. *Diseño físico de la base de datos.*

6. *Implementación del sistema de bases de datos.*

Las seis fases mencionadas no tienen que efectuarse necesariamente en esa secuencia estricta, en muchos casos es posible verse obligado a modificar la fase anterior durante la fase que le precede, estos son los llamados ciclos de retroalimentación entre fases y son muy comunes durante el diseño de bases de datos.

Tanto la fase uno que implica recabar información sobre el uso que se piensa dar a la base de datos, y la fase seis en donde se pone en marcha la base de datos, son

consideradas fases que no forman parte del diseño de bases de datos, sino que son parte del ciclo de vida del sistema de información.

Aún cuando estoy de acuerdo con esa aseveración, he querido incluirlas en el proceso de diseño de bases de datos, porque en este capítulo no contemplo las fases previas que forman parte del ciclo de vida del sistema de información y que son de gran importancia para elaborar el diseño conceptual de la base de datos.

La parte medular del proceso de diseño de una base de datos, la constituyen las fases 2, 4 y 5, conforme avancemos en este capítulo, ahondaré más en cada una de las fases mencionadas.

En lo que va escrito en este capítulo, he mencionado las fases que componen el ciclo de vida de un sistema de información, el ciclo de vida de un sistema de base de datos y ahora las fases que componen el proceso de diseño de bases de datos, para asegurar que hasta este momento se tenga una idea cronológica correcta de donde está situado cada proceso, a continuación expongo una gráfica en donde preciso que paso antecede a cuál otro, en esa gráfica también delimito cuales son las fases entorno a las que trata específicamente este capítulo (Véase figura. 5.2).

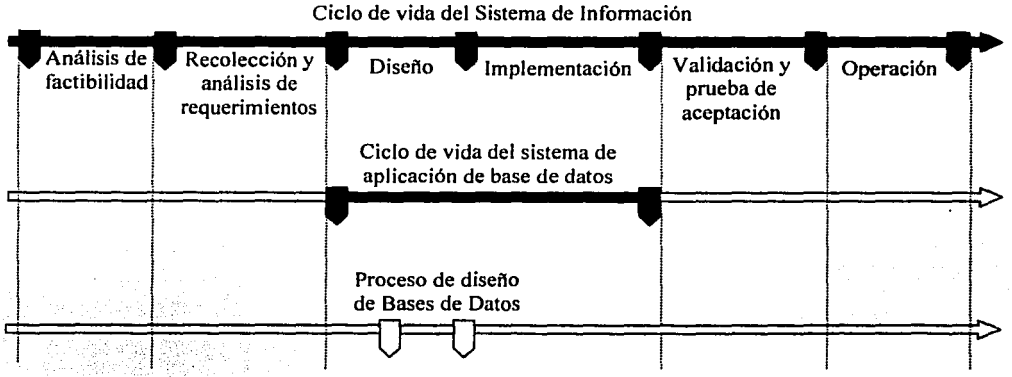


Figura 5.2 Diagrama cronológico de las fases que intervienen en el desarrollo de un sistema de información.

Este capítulo va dirigido hacia el proceso de diseño de Bases de Datos, conformado por las seis fases antes mencionadas (Recolección y análisis de requerimientos, Diseño conceptual de la base de datos, Elección de un SGBD, Transformación al modelo de datos, Diseño físico de la base de datos e Implementación del sistema de base de datos) que se llevan a cabo dentro del paso 2 del ciclo de vida del sistema de aplicación de base de datos (Véase figura. 5.3).

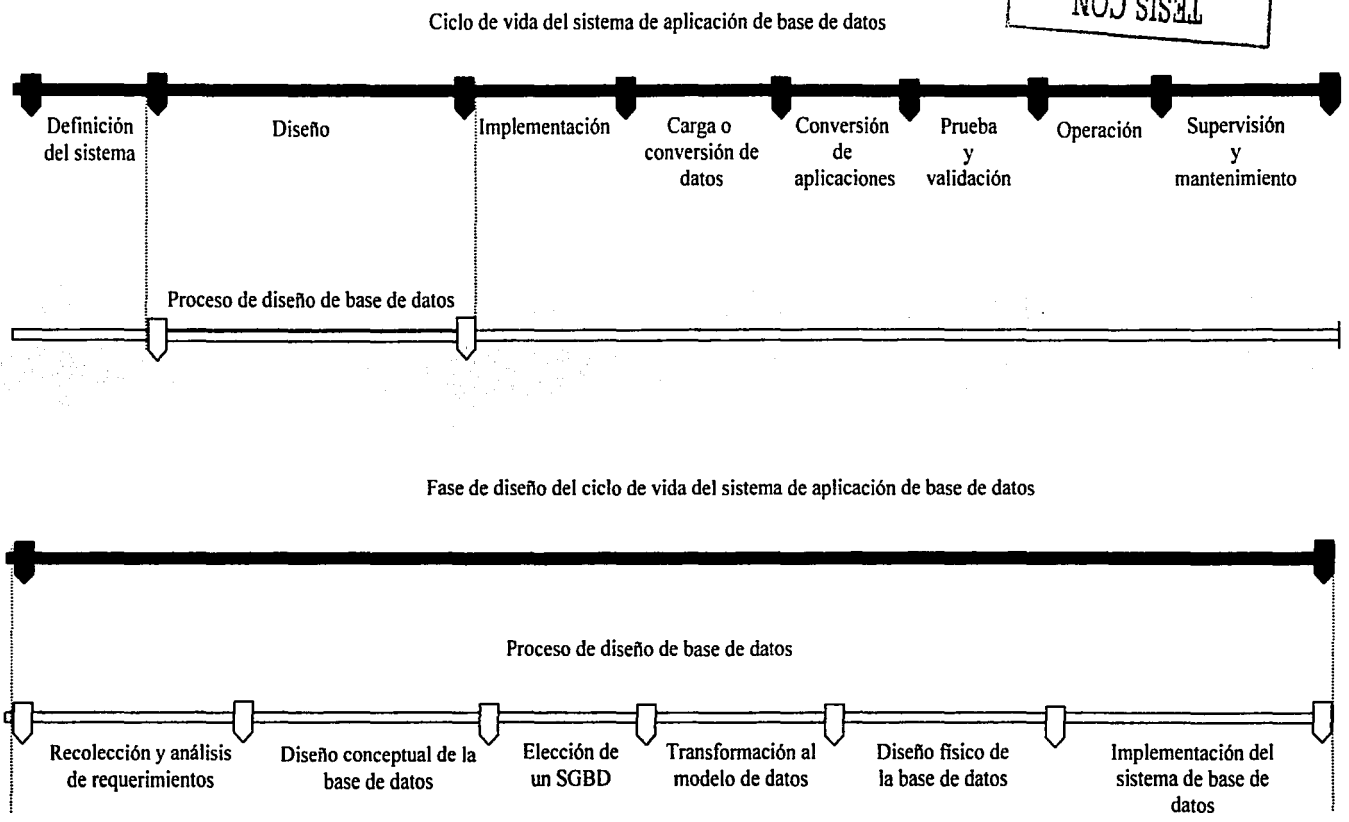


Figura 5.3 El proceso de diseño de base de datos, tiene lugar durante la fase de diseño del ciclo de vida del sistema de aplicación de base de datos.

5.3 Fase 1: Recolección y análisis de requerimientos.

Esta fase surge de la necesidad de conocer las expectativas de los usuarios y de los usos que se pretende dar a la base de datos, mismos que se deben de recabar con el mayor detalle posible.

Para especificar los requerimientos, se debe de identificar las demás partes del sistema de información que van a interactuar con el sistema de bases de datos, entre ellas a los usuarios.

Una vez recabados los requerimientos, éstos son reunidos y analizados mientras se llevan a cabo las siguientes actividades:

- Identificación de las principales áreas de aplicación y grupos de usuarios que utilizarán la base de datos.
- Se estudia y analiza la documentación existente relacionada a las aplicaciones que van a interactuar con el sistema de base de datos.
- Se estudia el entorno de operación actual y cuales son los planes de aprovechamiento de la información.
- Se analizan las prioridades de los posibles usuarios y la importancia que dan a diversas aplicaciones.

5.4 Fase 2: Diseño conceptual de la base de datos

En esta segunda fase de diseño de base de datos, se examinan los requerimientos obtenidos en la fase uno para obtener a partir de ellos, un esquema conceptual de lo que será la base de datos, dicho esquema deberá ser independiente de cualquier SGBD.

Dicho esquema conceptual, debe ser capaz de darnos un entendimiento completo de la estructura de la base de datos, así como de las relaciones entre los objetos que existan en la base y las restricciones que resulten de trasladar las reglas del negocio a la base de datos.

Algo importante es el poder lograr una independencia entre el esquema conceptual y el SGBD, porque todos los SGBD suelen tener peculiaridades y restricciones que no deben influir sobre el diseño del esquema conceptual.

Para diseñar un esquema conceptual, debemos identificar los componentes básicos del esquema que son los tipo de entidades, los tipos de relaciones entre entidades y sus atributos, también se deben de especificar los atributos clave, la cardinalidad y las entidades débiles, todo ello derivado de la fase uno de recolección y análisis de requerimientos.

5.5 Fase 3: Elección del SGBD

La elección del SGBD obedece a varios factores, algunos técnicos, otros económicos y algunos concernientes a políticas de la organización.

Los factores técnicos tienen que ver con la capacidad del SGBD para resolver la tarea en cuestión, lo que debemos considerar en primera instancia es el tipo de SGBD, si es relacional, de red, jerárquico, orientado a objetos o de alguna otra clase. Entre los factores técnicos a considerar están las estructuras de almacenamiento y caminos de acceso que maneja el SGBD, las interfaces de usuario y programador disponibles, los tipos de lenguajes de consulta disponibles y los tipos de lenguaje de consulta de alto nivel.

En cuanto a los factores económicos y de organización que influyen en la elección de un SGBD podemos mencionar las siguientes:

- Costo de adquisición del software: Es el gasto inicial que hace al comprar el SGBD y todo el software adicional que ello implique, como lenguajes de alto nivel para interactuar con el SGBD y diferentes aplicaciones que se emplearán de interfaz para facilitar la interacción con el usuario.

- Costo de mantenimiento: Éste es el gasto que se tiene que llevara cabo por concepto de servicio de manutención por parte del proveedor y de la actualización regular de la versión del SGBD.

- Costo de adquisición de hardware: Esta es una posibilidad que estará en función de los requerimientos necesarios para el buen desempeño del SGBD y del equipo con el que se cuente al momento de adquirir este software, puede ser que se requiere adquirir memoria adicional o unidades de disco duro con mayor capacidad de almacenamiento.

- Costo de creación y conversión de la base de datos: Es el costo que implica crear la base de datos desde un inicio o bien convertir un sistema ya existente al nuevo software del SGBD.

- Costo de personal: Cuando se adquiere un nuevo software en una organización, generalmente hay que emprender un proceso de reorganización en el departamento que se encargará de la base de datos, entre los cambios es posible que se tenga que contratar a nuevo personal, como caso específico tenemos el puesto de un DBA, en caso de que no haya quién administre la base de datos.

- Costo de capacitación: El costo de capacitación será necesario dependiendo de la dificultad para operar el nuevo SGBD, por lo general es preciso capacitar al personal para su uso y programación.

5.6 Fase 4: Transformación al modelo de datos (diseño lógico de la base de datos)

La fase cuatro de esta metodología, consiste en crear un esquema conceptual pero basado en el SGBD elegido, esto se hace transformando el esquema conceptual producido en la fase dos del modelo de datos de alto nivel al de datos del SGBD.

El resultado de esta fase debe consistir en enunciados DDL[23] escritos en el lenguaje del SGBD elegido que especifiquen los esquemas a nivel conceptual y externo del sistema de base de datos.

Existen muchas herramientas CASE[24] de diseño automatizado que pueden generar DDL para sistemas comerciales a partir de un diseño de esquema conceptual.

5.7 Fase 5: Diseño físico de la base de datos.

El diseño físico de la base de datos es el proceso de elegir estructuras de almacenamiento y caminos de acceso específicos para que los archivos de la base de datos tengan un buen rendimiento con las diversas aplicaciones que utilizan la base de datos.

La elección de las estructuras más adecuadas va a depender del SGBD con el que se esté trabajando, una vez que se ha elegido un SGBD, este proceso se resume a elegir de entre las opciones que nos presenta el software para el manejo de los archivos contenidos en la base de datos.

En la lección de las opciones de diseño físico se suelen considerar los siguientes criterios:

- Tiempo de respuesta: Es el tiempo que transcurre entre que una transacción va a ser ejecutada y la obtención de una respuesta. Un aspecto que influye mucho sobre el tiempo de respuesta, es el tiempo de acceso a la base de datos en donde tiene mucho que ver el SGBD.

- Aprovechamiento de espacio: Esto se refiere a la cantidad de espacio de almacenamiento que ocupan los archivos de la base de datos y sus estructuras de acceso.

- Productividad de las transacciones: Es el número promedio de transacciones que el sistema puede procesar por minuto y suele medirse en las condiciones de mayor tráfico en el sistema.

---

[23] Lenguaje de definición de datos - *Data Definition Language* (DDL)

[24] Ingeniería de Software asistida por computadora - Computer Assisted Software Engineering – (CASE)

5.7 Fase 6: Implementación del sistema de base de datos

Una vez completado el diseño lógico y físico de la base de datos, se compilan los enunciados escritos en el DDL del SGBD seleccionado y con ello se crean los archivos que forman los esquemas de la base de datos.

Ya creados los archivos, se encuentran vacíos y listos para almacenar la información, con lo cuál se procede a cargar la base de datos, si existe información que deba importarse de otra aplicación, entonces los programadores implementarán rutinas de conversión para trasladar los datos del antiguo sistema a la nueva base de datos.

Una vez que las transacciones estén listas y los datos hayan sido cargados a la base de datos, entonces la fase de diseño e implementación habrá terminado y comienza la fase de operación del sistema.

5.8 Un caso práctico: Planteamiento del problema

A continuación expongo un caso práctico, con lo cuál, deseo ejemplificar la metodología de diseño de bases de datos expuesta anteriormente, como mencioné en la introducción, la metodología está centralizada en la construcción de la base de datos y no en la elaboración de la aplicación.

El quehacer administrativo siempre ha sido un trabajo arduo, aún cuando no es una labor difícil, se vuelve extenuante por la enorme cantidad de información que se maneja en procesos que se repiten una y otra vez, facilitando así la aparición de errores y dificultando la adecuada organización de la misma.

Las computadoras son la solución idónea para administrar, almacenar y garantizar la integridad de nuestra información, así como un acceso pronto y eficaz a nuestros datos.

Como parte de mi trabajo de tesis, propongo el uso de un sistema para agilizar el proceso de elaboración de propuestas por medio de un análisis del banco de horas de la carrera.

El objetivo es entonces, implementar un sistema que permita al usuario realizar su trabajo en la mitad del tiempo que lo hacía antes, con un mínimo de error y en una plataforma sumamente amigable, esto con la finalidad de necesitar un lapso de tiempo muy corto en la capacitación para el uso correcto y óptimo del sistema.

5.9 Definición del sistema

Una tarea que es llevada a cabo todos los semestres, en cada una de las jefaturas de carrera en la ENEP Aragón, es la elaboración de propuestas para respaldar el banco de horas.

Estos documentos, fungen como renovación del contrato para el personal docente de cada jefatura, a grandes rasgos en el se exponen tanto los nombres de las asignaturas, como el número de horas, días en que serán impartidas, la fecha de inicio y fin del periodo semestral y por supuesto el nombre y firma del profesor que impartirá dichas asignaturas. También sirven de referencia para actualizar el pago de nómina al departamento de personal.

Es importante señalar que no existe un formato estándar para elaborar las propuestas, pero existen requerimientos indispensables para la elaboración de las mismas, y es en estos en los que se basa cada jefatura de carrera para elaborarlos, algunas personas han optado por seguir los lineamientos marcados por otras jefaturas, y hasta utilizar el mismo formato de otros departamentos, lo que ha propiciado cierta uniformidad, sin que se haya generalizado aún.

El formato original fue elaborado en Microsoft Excel y existe uno por cada profesor, el cuál es llenado uno por uno, como no se cuenta con alguna base de datos en la computadora, entonces información tal como: Claves de la materias, R.F.C, horas teoría y horas práctica son buscadas una por una en listas previamente impresas.

Lo anterior representa un problema de derroche de tiempo y esfuerzo humano que bien puede canalizarse hacia áreas más importantes.

5.9.1 Solución propuesta

Ahora bien, ¿cómo pretendo agilizar este proceso?, mi propuesta es la siguiente:

Utilizar la computadora para llevar a cabo la búsqueda y llenado de información específica como lo es: El R.F.C. del profesor, el nombre de la asignatura, las horas teóricas y prácticas, el horario, etc. Elaborar un formato único, que sea llenado e impreso fácilmente desde la computadora.

Todo esto en un ambiente amigable, es decir, de muy fácil comprensión y uso, además accesible a los equipos de cómputo con los que se cuenten en la ENEP Aragón y en específico para la carrera de Ingeniería en computación.

Para esto desarrollé e implanté un sistema que cumple los objetivos ya planteados.

5.9.2 Recolección y análisis de requerimientos

a) Identificación de los usuarios principales

Las figuras que intervienen en este sistema de base de datos son tres:

- La jefatura de carrera: Es quien elabora las propuestas teniendo conocimiento previo de el número de horas totales que tiene para asignar a los profesores, ese número de horas del que puede disponer para asignar, se le conoce como banco de horas.

  La figura de jefatura de carrera, está representada por una persona que operará el sistema, por lo que la o las aplicaciones que deriven de este desarrollo serán operadas por un solo usuario.

  Además del número de horas que tiene para asignar, la jefatura de carrera debe contar con todos los datos pertenecientes a la plantilla de profesores de los que puede disponer para asignar horas de profesor.

- El profesor es la persona por la cuál se elaboran las propuestas, de modo que toda información concerniente al profesor que deba estar contenida en la propuesta, deberá ser almacenada y actualizada en la base de datos, el profesor, no tendrá contacto directo con el sistema de base de datos, pero si recibirá uno de los productos finales que arrojará el sistema que es la propuesta, misma que firmará para que sea canalizada por medio de la jefatura de carrera al departamento de personal para que sea terminado el trámite.

- El departamento de personal es a quien se le entrega la propuesta ya elaborada y firmada por el profesor.

De las tres figuras anteriores, solo una de ellas (la jefatura de carrera) operará el sistema, tanto los profesores, como el departamento de personal, solo están interesados en el producto final que arrojará el sistema, que en este caso será un documento que contenga la información necesaria para extender, renovar o iniciar su contrato con la ENEP Aragón.

Para elaborar una propuesta, se requiere la siguiente información: Los datos personales del profesor, la categoría del profesor, nombre de la carrera en donde dará las clases, nombre de las asignaturas que impartirá al igual que el horario.

Cada propuesta elaborada debe incluir el tipo de movimiento que se realizó (alta, prorroga, baja o licencia), la fecha de elaboración y la fecha de inicio y límite para los que tiene vigencia esa propuesta.

También se consideran los siguientes supuestos:

a) Una "propuesta" es un documento que hace las veces de una extensión o revocación de contrato, por lo que un profesor tendrá una propuesta por cada tipo de movimiento autorizado en cada carrera.

b) Un profesor puede tener una categoría diferente por cada asignatura.

c) Las categorías sirven para clasificar jerárquicamente a los profesores, de modo que entre mayor jerarquía, mayores privilegios tiene, entre ellos mayor remuneración económica.

d) Un profesor puede impartir una o varias asignaturas en distinto horario, pero todas aparecerán en el documento llamado "propuesta".

e) Las asignaturas que imparte cada profesor, son agrupadas por categoría del profesor y por tipo de movimiento.

f) Todas las propuestas deben estar almacenadas en la base de datos, pero se entregarán al departamento de personal de forma impresa y firmada por el profesor.

g) Tanto los datos de los profesores como los datos de las asignaturas deberán estar almacenados de manera permanente.

5.9.3 Diseño Conceptual de la base de datos

Considerando el enunciado del problema y los supuestos semánticos anteriores, obtenemos la información necesaria para extraer del problema el siguiente grupo de tipos de entidades.

- **Tipo de entidad** *Profesor*: El cuál representa al objeto del mundo real "persona que imparte alguna asignatura en alguna carrera de la ENEP Aragón" y para la cuál son elaboradas las propuestas.

  Se van a considerar los siguientes atributos para esta entidad:

  Nombre :Representa el nombre o nombres del profesor

  aPaterno :Apellido paterno del profesor

  aMaterno: Apellido materno del profesor

  RFC: Representa el Registro federal de contribuyentes del profesor

nExpediente: Representa el número de expediente del profesor, mismo que se le asigna al ser contratado por la ENEP Aragón.

Telefono: El número telefónico del profesor

Domicilio: Dirección particular del profesor

CURP: Registro único de población

- **Tipo de entidad** *Propuestas*: Esta entidad representa al "conjunto de documentos donde se estipula la extensión o revocación del contrato para el personal docente de cada carrera"

  Se van a considerar los siguientes atributos para esta entidad:

  IdPropuesta :Representa un identificador único para identificar a cada propuesta

  Profesor :Heredado del tipo de entidad *Profesor*

  ClaveAsignatura: Heredado del tipo de entidad *Asignaturas*

  Periodo: Representa semestre en curso para el cuál se elabora la propuesta

  Grupo: Heredado del tipo de entidad *Grupos*

  Categoria: Heredado del subtipo concerniente a la entidad *Profesor*

  Movimiento: Representa al tipo de movimiento que se está llevando a cabo con esta propuesta (alta, baja licencia).

  Causa: Representa los motivos por los que se lleva a cabo el movimiento

  Carrera: Carrera de la ENEP Aragón a la que pertenece la prorroga

  fInicio: Representa la fecha de inicio a partir de la cuál entra en vigor la propuesta

  fLimite: Representa la fecha de termino a partir de la cuál la propuesta carece de valor

  HorasTeoria: Representa el número de horas teóricas que el profesor impartirá clase

  HorasPractica: Representa el número de horas que el profesor impartirá clase en algún laboratorio

Horario: Es el horario en el cuál el profesor impartirá la asignatura

uResponsaable: Nombre de la unidad responsable donde se revisó la propuesta

Observaciones: Texto que contiene alguna aclaración o comentario respecto de la propuesta

- **Tipo de entidad** *Asignaturas*: Representa al conjunto de todas las asignaturas que se imparten en alguna carrera de la ENEP Aragón.

  Para esta entidad, se han considerado los siguiente atributos

  ClaveAsignatura :Representa un identificador único para identificar a cada asignatura

  NombreAsignatura :Nombre de la materia

  Horas: Numero de horas a la semana que debe dedicarle un profesor a impartir esta asignatura

Avanzando un poco más en el análisis, podemos tomar en cuenta algunas consideraciones:

Aún cuando los requerimientos iniciales demandan solamente un sistema centralizado, en donde se requiere un entorno independiente con una sola computadora que acepte la entrada de datos, procese esos datos y después muestre los resultados de ese procesamiento en su mismo monitor, lo más sensato es preparar el camino para que la base de datos que resulte a partir del modelo Entidad Relación, pueda ser trasladada a un escenario de computación distribuida, pensando en que la tendencia creciente de hoy son las aplicaciones que residen y trabajan a lo largo de Internet o de una Intranet.

Hasta este momento, el problema es modelado desde el punto de vista conceptual, aquí es donde se definen los objetos que contiene el sistema y las relaciones entre ellos.

En esta etapa de diseño a nivel conceptual, se describe el problema tal y como es en el mundo real, percibiéndolo sin tener en cuenta como será representado para que lo entienda la computadora, por el momento resultan irrelevantes los procesos de almacenamiento y mantenimiento de la información, no nos interesa si estará la base de datos instalada en una computadora personal o en un servidor de base de datos en red.

Lo que si nos concierne y resulta importante es el número de usuarios que van a utilizar la base de datos, por el momento, la idea es que la base de datos contenga información de una sola carrera de la ENEP Aragón, pero pensando que en un futuro se tenga la necesidad de manejar las propuestas de más de una carrera en una misma base de datos, he optado por agregar los siguientes tipos de entidad:

- **Tipo de entidad** *Carrera*: El cuál representa al objeto del mundo real "carrera de la ENEP Aragón donde fue elaborada la propuesta".

  La razón de tener esta entidad es el poder agrupar las materias por carrera, en el supuesto caso de que en una misma base de datos coexistan los datos de dos o más carreras.

  Entonces, para esta entidad tendremos los siguientes atributos:

  IdCarrera :Representa un identificador único para identificar a cada carrera

  NombreCarrera :Nombre de la carrera

- **Tipo de entidad** *Grupos*: Esta entidad representa al conjunto de todos los grupos existentes en la carrera de la ENEP Aragón para la cuál se están elaborando las propuestas.

  Para esta entidad son considerados los siguientes atributos:

  IdGrupo :Representa un identificador único para identificar a cada grupo

  Grupo : Número del grupo

A continuación nos encontramos con una consideración más, la entidad *Profesor* puede tener las siguientes clasificaciones: Ayudante de Profesor "A", Ayudante de Profesor "B", Profesor de Asignatura "A" Interino, Profesor de Asignatura "B" Interino, Profesor de Asignatura "A" Definitivo, Profesor de Asignatura "B" Definitivo, la entidad profesor puede entrar en una o varias de las categorías antes mencionadas, incluso puede ser que en un futuro se agreguen nuevas categorías para clasificar a los profesores, para resolver este problema, se incluirá un tipo de entidad llamado *Categorías,* que tendrá la función de "catalogo" y contendrá todas las categorías existentes y de igual manera almacenará futuras categorías.

- **Tipo de entidad** *Categorias*: Esta entidad representa al conjunto de todas las posibles clasificaciones que es posible asignarle a un profesor.

  Para esta entidad son considerados los siguientes atributos:

  IdCategoria :Representa un identificador único para identifjcar a cada categoría

  Categoria : Nombre de la categoría

FALTA
PAGINA

|105|

Una vez analizado el problema e identificadas las entidades que participarán en el modelo, se procede a elaborar el esquema conceptual (Véase figura 5.4).

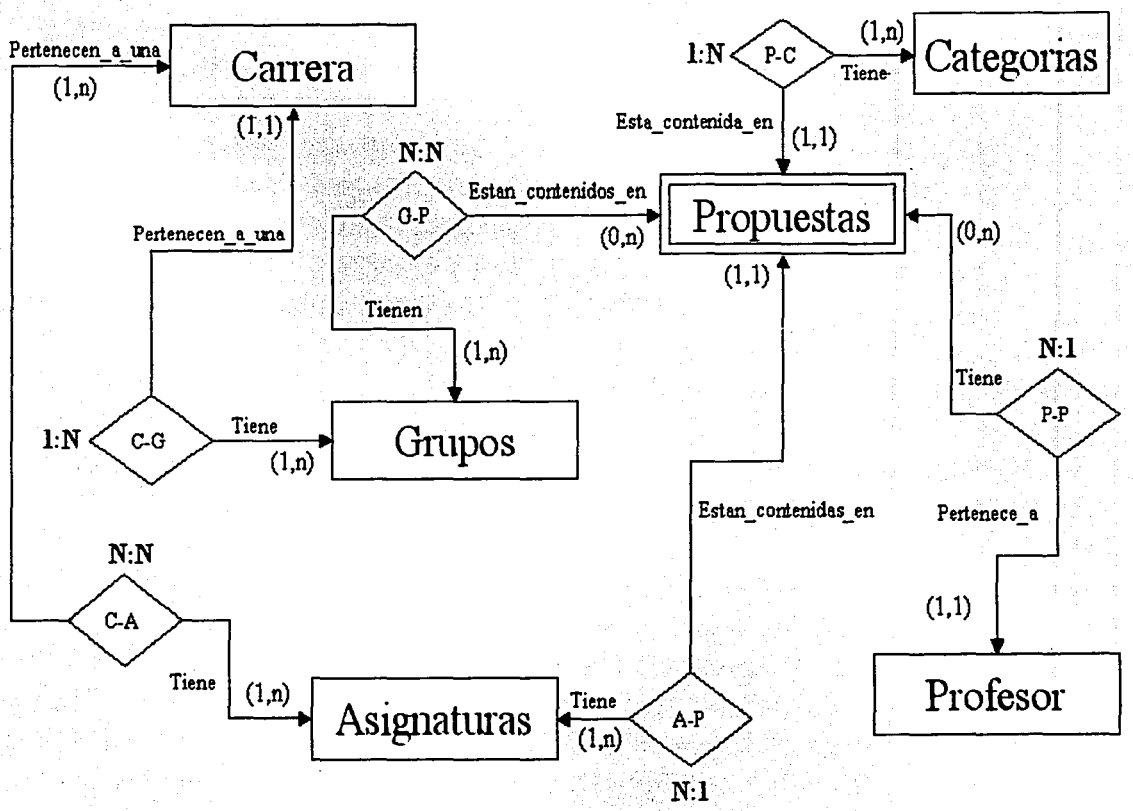


Figura 5.4 Esquema conceptual

Este esquema conceptual, es independiente del SGBD y antes de pasar al siguiente paso, es conveniente analizarlo junto con los probables usuarios del sistema, para asegurarse de que el modelo construido, describe correctamente el problema en cuestión.

5.9.4 Elección del SGBD

Como mencioné anteriormente, la elección del SGBD obedece a varios factores, de los cuales señalé algunos técnicos y otros económicos que llegan a influir.

Existen algunas veces, en los que desafortunadamente el factor económico no solo es la parte importante, sino que además es la parte decisiva en la toma de esa decisión, el sistema de base de datos que estoy tomando como ejemplo práctico, fue concebido como idea original en la secretaría técnica de la carrera de ingeniería en computación de la ENEP Aragón, aunque la idea es buena y el beneficio puede alcanzar a todas las carreras de la ENEP, no se cuenta con presupuesto para asignar a este proyecto.

La aclaración anterior, es porque en un caso como éste, el factor económico es quién juega el papel preponderante en la decisión de que SGDB se va a emplear, para este caso en específico, se hecho mano de los recursos con los que ya contaba la organización, descartando la posibilidad de adquirir nuevo software, es por eso que se empleó Microsoft Access para crear la base de datos.

Consideré pertinente exponer las razones por las que se empleó Microsoft Access, que aún cuando creo que cumple su papel de aplicación para la creación y manipulación de base de datos pequeñas, no entra a mi consideración, dentro de la definición que hemos manejado hasta este momento de SGBD.

Aún cuando Microsoft Access no es la solución idónea para resolver el problema, no por eso deja de ser una solución viable en este momento, ente las ventajas está obviamente el que no hay que hacer inversión alguna, ni de adquisición, ni de capacitación.

Por otro lado, se pierde la posibilidad de tener, a corto plazo, una base de datos que resida en un servidor de base de datos y contenga información de más de una carrera, e incluso haciendo algunos cambios al modelo conceptual, obtener también horarios por carrera y por grupo, mismos que podrían consultarte por Internet, es por eso que no resulta óptima la solución obtenida con Microsoft Access.

5.9.5 Transformación al modelo de datos

Como la base de datos será elaborada en el formato de Microsoft Access, entonces el esquema conceptual debe ser transformado a un esquema entidad – relación.

Una vez elaborado el esquema entidad-relación, emplearé la misma herramienta de diseño que incluye Microsoft Access para construir la base de datos.

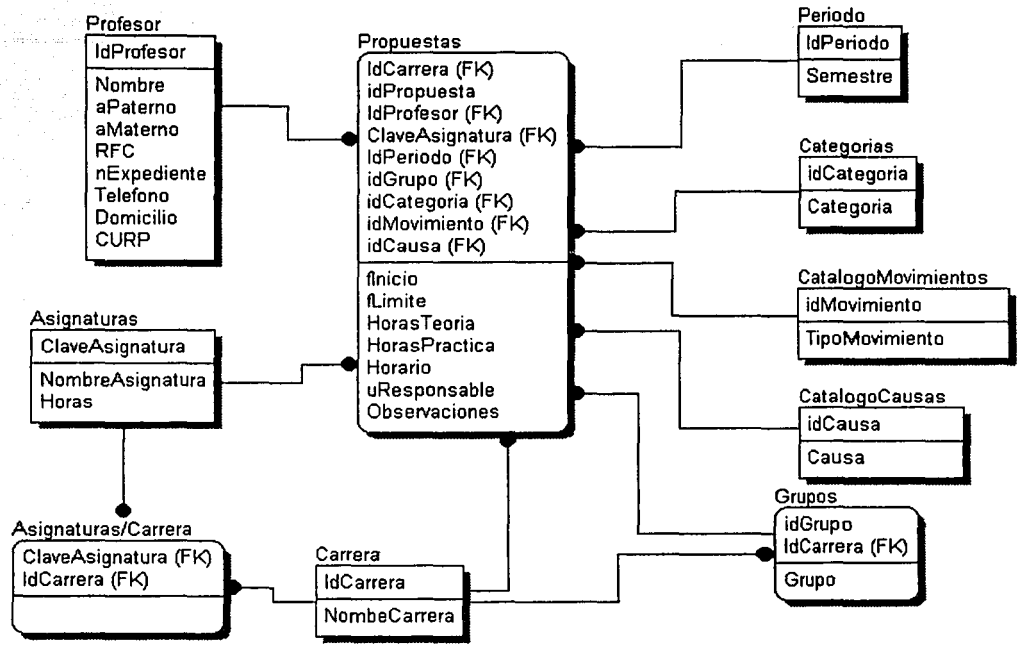


figura 5.5 Esquema Entidad -Relación

La figura 5.5 representa el esquema entidad - relación resultante del modelo conceptual, de haber empleado un SGBD como ORACLE, entonces deben de introducirse los enunciados DDL necesarios para definir cada una de las tablas, empleando SLQ como se vio en el capítulo 3.

Por ejemplo:

CREATE TABLE Carrera (IdCarrera int NOT NUL, NombreCarrera char(60));

5.9.6 Diseño físico de la base de datos

El diseño físico contempla la elección de las estructuras más adecuadas para mejorar el desempeño de la base de datos en cuanto a productividad en las transacciones, éstas estructuras que uno puede seleccionar, dependen del SGBD a utilizar, en este caso, Microsoft Access al no ser un SGBD, no ofrece una gran diversidad de recursos de los que podamos hacer una selección, entre la serie de recursos que ofrece es la creación de índices para los campos que uno le indique, esto puede mejorar los tiempos de respuesta al

momento de llevar a cabo consultas empleando ese campo, emplearé índices solo en los campos que sirven de identificador único.

Algo que proporciona Microsoft Access y resulta ser muy útil, es la posibilidad de actualización y borrado en cascada, lo que me da la certeza de que si hago un cambio a algún dato personal de la entidad profesor, ese cambio se verá reflejado simultáneamente en las propuestas, lo que me garantiza coherencia en la información generada a partir de la base de datos.

Además de algunas reglas de validación que Microsoft Access permite llevar a cabo directamente sobre los campos a la hora de insertar datos, no hay muchas opciones que se puedan configurar

5.9.7 Fase 6: Implementación del sistema de base de datos

Una vez que se ha completado el diseño lógico y físico de la base de datos y se obtiene, en este caso, el archivo .mdb, la base de datos queda lista y en espera de que sea cargada con los datos.

La elaboración de propuestas es una actividad previa al inicio del semestre, para cuando inició esta fase, nos encontrábamos a mitad del semestre, después de introducir durante una semana datos de prueba y depurar los errores en la interfaz, se procedió a cargar los datos por partes, siendo los datos que no presentaban grandes cambios, los primeros en ser ingresados a la base, por ejemplo, los datos personales de los profesores, los catálogos de materia, causas, categorías, periodo, carrera y asignaturas los primeros en ser almacenados, dejando temporalmente a un lado la elaboración de propuestas.

La aplicación que sirve de interfaz para el usuario, fue hecha en Visual Basic y más que programar las peticiones a la base de datos, el reto fue diseñar una interfaz que resultara muy amigable en su utilización, pues la idea es que una vez que se termine la etapa de pruebas, se comience a liberar el sistema paulatinamente en cada una de las carreras de la ENEP Aragón, por supuesto se prevé que la capacitación sea mínima, al igual que el mantenimiento.

Las pantallas de captura son apegadas al estándar impuesto por Windows, las ventanas, menús, botones, cajas de texto y menús contextuales son elementos que prácticamente todos los usuarios de estaciones de trabajo tipo PC conocen y les es natural su uso.

La figura 5.6 es un ejemplo de las pantallas que se emplean para captura de datos

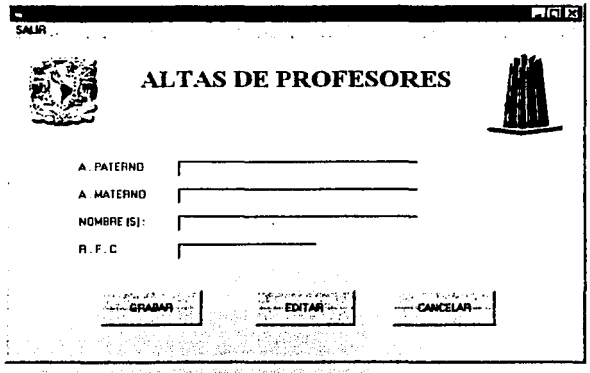


figura 5.6 Ejemplo de pantalla de captura

Para toda consulta a la base de datos, el usuario emplea la interfaz gráfica, como se carece de la figura de administrador de la base de datos, no está abierta la posibilidad de realizar consultas directas por medio de SQL.

Todas las consultas están junto con el código de Visual Basic y son las que fueron solicitadas por el usuario del sistema (véase figura 5.7).

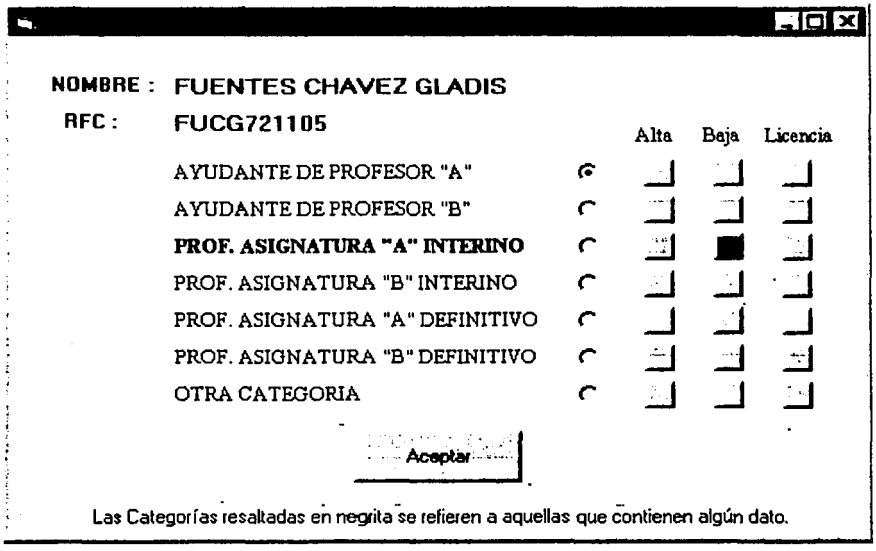


Figura 5.7, consulta de propuestas por profesor, clasificadas por categoría.

El sistema está pensado para que en lo posible, el usuario elija entre opciones en vez de teclear él mismo el dato, por ejemplo cuando se trata de asignar categorías, días de la semana, tipos de movimiento, RFC, semestre actual, etc.

Con esto se evita errores de sintaxis o los típicos errores de presionar una tecla por otra, de este modo se prevén inconsistencias en la información (véase figura 5.8).

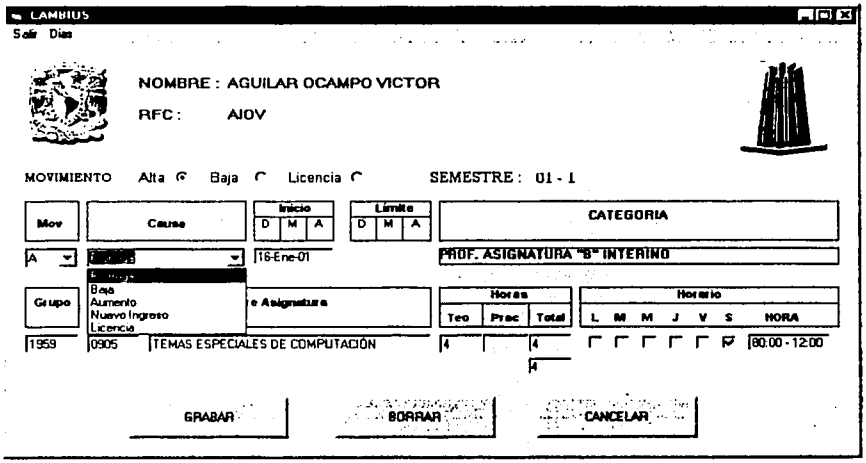


Figura 5.8, Pantalla de cambios hechos a una propuesta

Las pruebas han sido hasta ahora satisfactorias y se planea liberar el sistema gradualmente para otras carreras, aunque por ahora el sistema cumple con los requerimientos señalados por el usuario, se tienen pensados algunos cambios, por ejemplo, la migración de la información a un SGBD, tentativamente se ha pensado en SQL SERVER, la inclusión en el proyecto de un servidor para bases de datos y el cambio de la interfaz por una orientada a Web.

Obviamente el trabajo de análisis, la solución propuesta y el esquema conceptual, son trabajo que no se pierde, pues aún cuando cambia la interfaz o el SGBD, el problema sigue siendo el mismo, por lo tanto la descripción, mientras sea planteada a un nivel alto de abstracción, sigue empleándose sin cambio.

Conclusiones generales

Cuando comencé con este proyecto de tesis, la necesidad a satisfacer era optimizar un proceso administrativo en la carrera de ingeniería en computación de la ENEP Aragón, de manera que mi idea original fue describir la metodología que llevaría a cabo para elaborar el sistema con el cuál agilizaría el proceso de elaboración de propuestas.

Una vez que inicié el proceso de análisis de requerimientos y me dispuse a realizar el modelado del problema a nivel conceptual, me encontré con que carecía de una metodología sustentable para llevar a cabo un modelado tal, que pudiese describir de manera gráfica y sencilla el problema, que involucrase todas las entidades contenidas en el mismo y las relaciones entre ellas.

Me di cuenta que la necesidad que yo quería resolver, estaba precedida por mi necesidad de tener los elementos necesarios para ofrecer una solución óptima, el problema no radicaba en encontrar una solución, porque a final de cuentas se me ocurrían varias, y en la parte de programación no tenía mayor problema.

Pero si iba a exponer en un trabajo de tesis, una solución a un problema, ¿qué parámetro emplear para saber que la solución que estás presentando es la adecuada?.

Decidí entonces, satisfacer primero mi necesidad de obtener los conocimientos básicos para poder sustentar la solución que iba a presentar.

El trabajo que desarrollé es entonces, un trabajo de investigación sobre bases de datos, explico en él, los conocimientos que considero son necesarios para poder llevar a buen término, un análisis de un problema, representarlo a nivel conceptual y finalmente implementarlo en una base de datos.

El contenido lo he divido de la manera cronológica como a mi me hubiera gustado obtener la información, empezando por conocimientos generales, hice énfasis en la importancia que tiene en nuestros días la obtención pronta de información confiable en la toma de decisiones, porque finalmente ese resulta el móvil para que tengan razón de ser las bases de datos.

El contenido lo explico de una manera didáctica, pues está dirigido a quién en algún momento tuviera la misma necesidad o inquietud que yo, de obtener los conocimientos necesarios para iniciarse en el mundo de las bases de datos, no deja de ser un documento introductorio, mi investigación puede servir de apoyo para compañeros que inicien trabajos sobre bases de datos no partan de cero, creo que ahí es donde radica su valor.

BIBLIOGRAFIA

Irene Luque Ruiz, Miguel Ángel Gómez - Nieto
Diseño y uso de BASES de DATOS RELACIONALES
De la edición RA-MA, 1997, Madrid España.

Gio Wiederhold
Diseño de base de datos
Mc Graw-Hill, 1988, segunda edición, México.

Bob Reselman
Active Server Pages 3.0 con ejemplos
Prentice Hall,2000, primera edición, Buenos Aires.

Adoración de Miguel Castaño, Mario G. Piattini Velthuis
Fundamentos y modelos de bases de datos
Alfaomoga Grupo Editor, segunda edición, Madrid España

Elmasri Ramírez, B. Navathe Shamkant
Sistema de base de datos, conceptos fundamentales
Pearson Educación, segunda edición, México

Curtis Smith, Michael Amundsen
Aprendiendo programación de bases de datos con Visual Basic 6 en 21 días
Pearson Educación, 1999, México

Jeffrey P. McManus
Bases de datos con Visual Basic 6
Prentice Hall, 1999, Madrid España

Campderrich, B.
Técnicas de Bases de Datos
Editores Técnicos Asociados, segunda edición, 1989

Date, C.J.
Introducción a los sistemas de Bases de Datos
Addison Wesley Iberoamericana, quinta edición, 1990

Batini, Ceri, Navathe
Diseño Conceptual de Bases de Datos
Addison Wesley, 1994

J. Martin
Organización de las Bases de Datos
Prentice Hall, 1982

M.L. Gillenson
Introducción a las Bases de Datos
McGraw Hill, 1988

G. Wiederhold
Diseño de Bases de Datos
McGraw Hill, 1986