

43
ZET



UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE CIENCIAS

DISEÑO LOGICO DE LAS BASES DE DATOS
RELACIONALES Y DE LAS BASES DE DATOS
ORIENTADAS A OBJETOS

T E S I S

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A :

HECTOR GUEVARA AVILA



FACULTAD DE CIENCIAS
UNAM

MEXICO, D. F.



FACULTAD DE CIENCIAS
SECCION ESCOLAR

1995

FALLA DE ORIGEN

TESIS CON
FALLA DE ORIGEN



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



UNIVERSIDAD NACIONAL
AUTÓNOMA DE
MÉXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo de Tesis:
"DISEÑO LOGICO DE LAS BASES DE DATOS RELACIONALES Y DE
LAS BASES DE DATOS ORIENTADAS A OBJETOS"
realizado por HECTOR GUEVARA AVILA
con número de cuenta 8115990-1 , pasante de la carrera de ACTUARIA
Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis	DRA. MARIA DEL CARMEN LOPEZ LAISECA
Propietario	M. EN C. VIRGINIA ABRIN BATULE
Propietario	M. EN C. ELISA VISO GUROVICH
Suplente	M. EN C. ALEJANDRO MINA VALDES
Suplente	ACT. ROSA MA. CRUZ SALCEDO

Ma. del Carmen Lopez

Virginia Abrin Batule

Elisa Viso

Alejandro Mina Valdes

Consejo Departamental de Matemáticas
MAT. CESAR GUEVARA BRAVO

[Signature]

DEDICO ESTA TESIS A:

La memoria de Diana Karina, mi hija.

Mi esposa, porque con su amor, apoyo y comprensión me motivó a culminar este trabajo.

Mis hermanos y tíos, por creer y confiar en mí.

Mi padre, por haberme dado la vida y enseñarme con su ejemplo el camino del bien.

Mi madre, por lo que significó en mi vida y por el aliento que me dio para ser mejor cada día y aunque ya no esté conmigo sigue siendo mi guía.

AGRADECIMIENTOS

A través de estas líneas expreso mi más sincero agradecimiento a la Dra. Ma. del Carmen López Laiseca, por su valiosa dirección para la culminación de este trabajo.

Doy las gracias al Dr. Gabriel Rivero González, jefe de la oficina de Capacitación del INFONAVIT y al grupo de profesores de la Facultad de Ciencias que con entusiasmo y visión hicieron posible la creación del seminario de tesis, con el cual fue posible titularme.

También agradezco a todas las personas que de manera desinteresada me brindaron su apoyo e influyeron positivamente para la terminación de mi tesis.

INDICE

INTRODUCCION.....	1
CAPITULO 1	
SISTEMAS TRADICIONALES DE ARCHIVOS Y SISTEMAS DE BASES DE DATOS.....	2
1.1. Sistemas tradicionales para procesamiento de archivos.	2
1.1.1. Problemas y limitaciones de los archivos tradicionales.....	3
1.2. Antecedentes de las bases de datos.....	4
1.3. Sistema de base de datos.....	6
1.3.1. Niveles de especialización.....	7
1.3.2. Responsable del manejo del sistema de base de datos.....	7
1.3.3. El administrador de la base de datos.....	7
1.4. Consideraciones en el manejo de los datos.....	8
1.4.1. Seguridad.....	8
1.4.2. Integridad de los datos.....	11
1.4.3. Respaldo y recuperación.....	12
1.4.4. Concurrencia.....	13
1.4.5. Capacidad de auditoría.....	14
1.5. Problemas para la implantación de una base de datos.....	14
1.5.1. Conflictos organizacionales.....	14
1.5.2. Fallas en el desarrollo del proyecto.....	14

CAPITULO 2

REQUERIMIENTOS Y HERRAMIENTAS DE LAS BASES DE DATOS....	16
2.1. Sistema de base de datos.....	16
2.2. Software en los sistemas de bases de datos.....	16
2.3. Lenguajes de definición y de manipulación de datos..	17
2.3.1. Lenguaje de definición de datos.....	17
2.3.2. Lenguaje de manipulación de datos.....	18
2.4. Diccionario de datos.....	18
2.5. Proceso de desarrollo de la base de datos.....	21
2.5.1. Planeación preliminar.....	22
2.5.2. Estudio de factibilidad.....	22
2.5.3. Definición de requerimientos.....	23
2.5.4. Diseño lógico de la base de datos.....	23
2.5.5. Implantación de la base de datos.....	25
2.5.6. Evaluación y mantenimiento.....	25
2.6. Arquitectura de las bases de datos.....	25
2.7. Dirección actual de los sistemas de bases de datos..	27

CAPITULO 3

DISEÑO LOGICO DEL MODELO DE BASE DE DATOS RELACIONAL... 28	28
3.1. Antecedentes de las bases de datos relacionales.....	29

3.2.	Objetivo del diseño de la base de datos relacional..	30
3.3.	Requerimientos del modelo de base de datos relacional.....	30
3.4.	Datos no normalizados y normalización de los datos..	30
3.4.1.	Datos no normalizados.....	31
3.4.2.	Normalización de los datos.....	31
3.5.	Propiedades de las tablas en una base de datos.....	33
3.6.	Acceso a la base de datos.....	33
3.6.1.	Las llaves y su definición.....	33
3.7.	Dependencia funcional.....	35
3.8.	Formas normales.....	37
3.8.1.	Primera forma normal.....	38
3.8.2.	Segunda forma normal.....	40
3.8.3.	Tercera forma normal.....	42
3.8.4.	Forma normal Boyce and Codd y Cuarta forma normal.....	49

CAPITULO 4

	DEFINICION Y CONCEPTOS FUNDAMENTALES DE LOS MODELOS Y LAS BASES DE DATOS ORIENTADAS A OBJETOS.....	57
4.1.	Antecedentes de las bases de datos orientadas a objetos.....	58
4.2.	Necesidades en el diseño de la base de datos orientada a objetos.....	59
4.3.	Diferentes niveles del modelo orientado a objetos...	59

4.4. Modelos y vistas particulares.....	61
4.5. Conceptos básicos.....	62
4.5.1. Objeto y encapsulación.....	64
4.5.2. Clases, instancias y herencia.....	67
4.6. Relaciones y cardinalidad.....	78
4.6.1. Cardinalidad.....	80
4.6.2. Relación funcional.....	80

CAPITULO 5

DISEÑO LOGICO DEL MODELO DE BASE DE DATOS ORIENTADO A OBJETOS.....	83
5.1. Descripción y características de diseño.....	83
5.2. Modularidad.....	84
5.3. Método de diseño orientado a objetos.....	86
5.4. Definición de clases y de objetos en el diseño.....	87
5.5. Pasos del diseño.....	89
5.6. Un caso práctico.....	91
CONCLUSIONES.....	98
BIBLIOGRAFIA.....	100

INTRODUCCION

El objetivo de esta tesis es presentar los conceptos y principios fundamentales de las bases de datos y en particular de los modelos relacional y orientado a objetos, haciendo énfasis en el diseño lógico de ambos modelos. Esto, debido a que un diseño lógico adecuado garantiza la confiabilidad de la información que se maneja en una base de datos, elimina redundancia y permite reducir costos de mantenimiento. Gran parte del éxito o fracaso en la implantación de un sistema de base de datos depende de un diseño lógico apropiado.

Por lo tanto, se destacarán los pasos para la identificación y definición de los elementos que deben ser incluidos en una base de datos, desde el inicio del sistema, es decir, desde la etapa del diseño lógico, donde se crea el esquema lógico o conceptual de la base de datos, en su totalidad.

Se eligen los modelos relacional y orientado a objetos por lo que representan. El modelo relacional revolucionó, en su tiempo, el concepto que se tenía entonces sobre las bases de datos existentes, además de estar fundamentado sobre fuertes bases matemáticas, mientras que el modelo orientado a objetos está considerado como uno de los modelos de vanguardia y la tendencia parece indicar que será de los más utilizados en la década de los noventa.

En el capítulo uno, se hace un breve recorrido de qué son los sistemas basados en archivos tradicionales y el surgimiento de las bases de datos con antecedentes, necesidades y elementos que intervienen en su manejo y operación. Así como los problemas que se pueden presentar al momento de la implantación de un sistema de base de datos.

El capítulo dos estará dedicado a detallar los requerimientos y las herramientas imprescindibles para la implantación y mantenimiento de una base de datos. En los siguientes capítulos, se abordará el tema del diseño lógico de las bases de datos relacionales y de las bases de datos orientadas a objetos.

En el capítulo tres, se hace la descripción del modelo relacional, sus antecedentes, objetivos y elementos que intervienen. Además, se presenta el concepto de normalización, apoyándose para ello en ejemplos que permiten ir mostrando las formas normales básicas de este modelo.

El cuarto y quinto capítulos, estarán dedicados a la definición y presentación de los conceptos fundamentales de las bases de datos orientadas a objetos. A la descripción del diseño lógico y determinación de los elementos básicos que lo conforman. Al final del quinto capítulo, se presenta un ejemplo sencillo y práctico de la aplicación de este modelo y la manera como se definen y caracterizan los objetos del mundo real dentro del modelo abstracto de la base de datos.

CAPITULO 1

SISTEMAS TRADICIONALES DE ARCHIVOS Y SISTEMAS DE BASES DE DATOS

Los avances científicos, tecnológicos y administrativos que se han hecho hasta nuestros días precisan de mejores sistemas de información en todas las actividades de nuestra sociedad.

Por eso es importante contar con sistemas de información modernos y eficientes y que además cuenten con dispositivos de almacenamiento confiables y seguros que permitan actualizaciones y consultas fáciles y de rápido acceso.

En pocas palabras, se requiere contar con sistemas flexibles que se adapten con facilidad a los cambios, sin perder consistencia en los datos.

Los sistemas tradicionales de archivo para almacenamiento de datos presentan serias deficiencias, como son: redundancia, inconsistencia y multiplicidad de datos, entre otras. Esto los hace costosos y difíciles de mantener.

Los nuevos sistemas de bases de datos parecen responder a las necesidades de las empresas y la industria en general, ya que tienen como característica principal la centralización de los datos para eliminar la inconsistencia y multiplicidad de los datos y tener un control más eficaz de los accesos al sistema de tal forma que sólo lo puedan hacer los usuarios que tengan la autorización para ello. Además de tratar de evitar hasta donde sea posible la redundancia.

1.1. SISTEMAS TRADICIONALES PARA PROCESAMIENTO DE ARCHIVOS

Los ambientes tradicionales para el procesamiento de archivos son sistemas orientados al proceso. Los datos fluyen de un programa a otro. Los archivos se crean para satisfacer necesidades específicas de procesamiento. Por ejemplo, si el departamento de compras desea hacer un seguimiento de las órdenes de compras, entonces se desarrollará un archivo de órdenes de compra solamente con los datos necesarios para generar la salida requerida por el departamento de compras. Cada programa o sistema de aplicación que se desarrolle contendrá los datos necesarios para cubrir sólo las necesidades de un departamento en particular o un grupo de usuarios.

En un sistema tradicional de archivos, la recuperación de datos debe hacerse registro-por-registro mediante el empleo de un lenguaje de tercera generación como COBOL, BASIC, FORTRAN o PASCAL, entre otros.

1.1.1. PROBLEMAS Y LIMITACIONES DE LOS ARCHIVOS TRADICIONALES

Los sistemas basados en archivos tradicionales muestran una serie de desventajas como son:

1. Redundancia de datos
2. Un control pobre de los datos
3. Capacidad inadecuada de manipulación de los datos
4. Esfuerzo excesivo de programación

Redundancia de datos. La mayor dificultad en los sistemas basados en archivos tradicionales es que cada aplicación usa su propio archivo especial de datos. Por lo cual, la información se encuentra repetida en diferentes aplicaciones. En un banco, por ejemplo, el mismo nombre de un cliente puede aparecer en el archivo de cuentas de cheques, en el archivo de ahorros y en el archivo de préstamos.

Además, a pesar de que el nombre del cliente es el mismo, el campo relacionado frecuentemente tiene un nombre diferente en los archivos. Así, Cname puede ser el nombre con el que se identifica el campo donde se almacena el nombre del cliente en el archivo de cuentas de cheques, Sname el nombre del mismo campo pero en el archivo de cuentas de ahorro e Iname en el archivo de préstamos. Es posible también que el mismo campo llegue a tener diferentes tamaños en cada archivo. De esta forma, Cname podría ser de hasta veinte caracteres, mientras que Sname e Iname tener un límite de quince caracteres. Esta redundancia incrementa los elevados costos de mantenimiento y almacenamiento. Por otro lado, la redundancia de los datos también incrementa los riesgos de inconsistencia entre las versiones de datos comunes de los diferentes archivos.

Por ejemplo, suponga que el nombre de un cliente fue cambiado de Carol T. Jones a Carol T. Smith, el campo del nombre puede ser inmediatamente actualizado en el archivo de cuentas de cheques, la siguiente semana ser actualizado en el archivo de cuentas de ahorros y actualizarse incorrectamente en el archivo de préstamos, con el tiempo, tales discrepancias pueden provocar serias degradaciones en la calidad de la información contenida en los archivos.

La inconsistencia de los datos puede afectar la veracidad de los reportes. Si se desea generar un reporte para la gerencia en el que se muestre a todos los clientes que tengan ya sea una cuenta de cheques o de ahorros y una de préstamos, Carol T. Jones sería omitida erróneamente del reporte porque su nombre aparece como Carol T. Smith en el archivo de préstamos o aparecer con los dos nombres como cliente distinto.

Control pobre de los datos. En los sistemas de archivos tradicionales, no hay control centralizado a nivel de dato. Por lo mismo es común que un dato tenga múltiples nombres, dependiendo del archivo donde se encuentre.

En un nivel más fundamental hay siempre la posibilidad de que varios departamentos de una compañía no sean consistentes en su terminología. Un banco, por ejemplo, puede usar el término de cuenta y querer decir una cosa cuando se aplica a ahorros y otra muy diferente cuando se aplica a prestamos. Un término que tiene diferentes significados en diferentes contextos es llamado un **homónimo**. Contrariamente, diferentes palabras pueden significar la misma cosa. En una compañía de seguros se pueden referir a una póliza o a un contrato y referirse a la misma cosa con ambas palabras. Dos términos que significan la mismo son llamadas **sinónimos**.

Capacidad inadecuada de manipulación de los datos. En los archivos tradicionales es relativamente fácil obtener algún registro en particular, pero la situación se complica cuando se quiere extraer un conjunto de registros relacionados. Por ejemplo, si se desea saber las características de una lámpara y se tiene el número de identificación se puede acceder el registro del producto dentro del archivo de productos directamente, esto es adecuado cuando se quiere consultar un registro solamente.

Sin embargo, el acceso a un conjunto de registros relacionados puede no ser fácil. Por ejemplo, supóngase que se quiere saber sobre las ventas que se le hicieron a un cliente "x", podría ser necesario conocer el número total de ventas, los precios que obtuvo y qué productos fueron comprados, sería muy difícil de obtener tal información si no se tiene la identificación completa de todo lo que se está buscando.

Esfuerzo excesivo de programación. Un nuevo programa de aplicación frecuentemente requiere de un nuevo conjunto de definiciones de todo el archivo de datos. Aun cuando un archivo existente pueda contener algunos de los datos que se necesitan, la aplicación frecuentemente requiere de otros datos sueltos, como resultado, el programador tiene que recodificar las definiciones de todos los datos del archivo existente, así como de las definiciones de los datos nuevos. En los sistemas tradicionales de archivos existe una fuerte interdependencia entre los programas y los datos.

1.2. ANTECEDENTES DE LAS BASES DE DATOS

En las primera épocas del procesamiento de datos la mayor parte del tiempo y la atención en el desarrollo de una aplicación se invertía en los programas, en vez de dedicarlo a los datos y las estructuras en que éstos debían almacenarse. La mecanización parcial del proceso de programación, o aun la estandarización de

estilo se desconocían. En este medio ambiente, el tratamiento de los datos difícilmente era la preocupación de mayor prioridad.

Con el desarrollo del software los programas tomaron una forma más estandarizada y "estructurada", y se acumularon muchas aplicaciones nuevas que debían implantarse, provocando que la gran cantidad de tiempo dedicado a mantener los programas existentes resultara cada vez menos aceptable. Se hizo evidente entonces, que "la forma en que se manejaban los datos en el pasado era uno de los principales factores en el problema de mantenimiento de programas con que se encontraban los programadores. Esto debido principalmente a las siguientes causas:

- 1) Los datos se almacenaban en diferentes formatos en distintos archivos.
- 2) Con frecuencia los datos no podían compartirse entre programas diferentes que los necesitaran, lo que hacía que se requirieran archivos redundantes.
- 3) A menudo los datos no eran recuperables ni eran seguros.
- 4) Por lo general los programas se escribían de manera tal que si se modificaba la forma en que los datos se almacenaban, era necesario modificar el programa para seguir trabajando."¹

Con las condiciones descritas era común encontrar que los archivos almacenaban datos inexactos, inconsistentes o que no estaban actualizados y por si fuera poco, cualquier modificación en los métodos de acceso requerían cambios en la programación. Bajo estas condiciones, surgió la necesidad de sistemas que facilitaran el manejo de los datos, que evitarán la redundancia innecesaria, la modificación de programas cada vez que hubiese algún cambio en la información y que además fueran seguros y confiables.

Los sistemas de bases de datos fueron específicamente desarrollados para hacer las interrelaciones de los datos de diferentes archivos en forma mucho más sencilla.

Los sistemas de bases de datos pueden eliminar la redundancia de los datos ya que todas las aplicaciones comparten un conjunto común de datos. La información esencial como el nombre del cliente o la dirección aparecerá sólo una vez en la base de datos. Así, cuando se tenga que hacer cualquier cambio se hará una sola vez y con ello las aplicaciones estarán accediendo a datos consistentes.

¹ GUILLENSON, Mark L.; Introducción a las BASES DE DATOS, Mc Graw Hill, 1985, pág. 79.

Un sistema de base de datos apoya en el control de datos centralizados y ayuda a eliminar la confusión provocada por homónimos y sinónimos.

Las bases de datos permiten la separación entre programas y datos, de modo que los programas pueden ser en cierta forma independientes de los detalles de la implantación de los datos. Al dar acceso a un conjunto de datos compartidos y al soportar los poderosos lenguajes de manipulación de datos, los sistemas de bases de datos eliminan una gran cantidad de programación inicial y de mantenimiento.

1.3. SISTEMA DE BASE DE DATOS

En las empresas más pequeñas, encontramos casi siempre una colección de registros organizados para una aplicación determinada. La idea básica en la implantación de una base de datos es la de que los mismos datos deben ser aprovechados para tantas aplicaciones como sea posible

Un sistema de base de datos está compuesto de una base de datos y software de propósito general, que sirve como intermediario entre las necesidades del usuario y los servicios que proporciona el sistema y ayuda también al personal asignado a manejar la base de datos y el hardware adecuadamente.

DEFINICION

Una base de datos es una colección de datos interrelacionados que pueden ser procesados por una o más aplicaciones del sistema.

En otras palabras "la base de datos puede definirse como una colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es la de 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 datos nuevos y para modificar o extraer los datos almacenados"²

El software para manejar la base de datos es usualmente comprado a un vendedor de software y permite el acceso a la base por medio de programas de aplicación o por representaciones particulares de la base de datos llamadas **vistas** (una vista puede mostrar todo o parte de la base de datos).

² MARTIN, James; Organización de las Bases de Datos, trad. Adolfo Di Marco, 1er ed., México, Prentice - Hall Hispanoamericana, S. A., 1988, pág. 19.

1.3.1 NIVELES DE ESPECIALIZACION

Los sistemas de bases de datos, en general, se diseñan para almacenar grandes bloques de información. Esto requiere de un buen manejo, tanto en la información como en la implementación de mecanismos para el empleo de los datos, por lo que son necesarios nuevos niveles de especialización en el trabajo, lo que en realidad es otra forma de estandarización.

1.3.2. RESPONSABLE DEL MANEJO DEL SISTEMA DE BASE DE DATOS

Para estandarizar todas las actividades que se realizan en un sistema de base de datos, evitar que se dupliquen las tareas y que sean hechas sin una planeación adecuada, es necesario contar con algo o alguien que supervise cada una de ellas. Para esto se debe disponer de una forma común de tratar la información. Por ejemplo, si un programador ya generó las rutinas que se encargan de la seguridad del sistema habría un gasto innecesario en tiempo y económico si otro programador generara otras rutinas para un fin similar. Por ésta y otras varias razones, surge la necesidad de un responsable que administre, respalde, asegure y audite los datos y accesos a la base de datos, además de que servirá de interlocutor entre las necesidades de los usuarios y los servicios que pueda brindar el sistema de base de datos.

Las actividades mencionadas generalmente se le encargan a un grupo de personas (puede ser sólo una persona pero es raro que esto suceda) al que se le llama **Administrador de la Base de Datos (DBA** por sus siglas en inglés, Database Administrator). De no existir el DBA, sería fácil imaginar la anarquía y los problemas que ocasionaría la falta de control.

Dentro del grupo administrador de la base de datos se encuentran personas que se especializan en el diseño, seguridad, integridad, respaldo y recuperación, concurrencia, etc., de la base de datos.

1.3.3. EL ADMINISTRADOR DE LA BASE DE DATOS

El administrador de la base de datos (DBA) ocupa una posición estratégica y ventajosa con respecto a la definición y a la imposición de estándares de la compañía.

Con el diseño de la base de datos centralmente controlado, el DBA puede definir los estándares para el nombramiento y formato de los datos, pantallas, reportes así como el formato de los archivos. Esto simplifica la documentación y el entrenamiento requerido por los usuarios y da una mayor integración dentro de la compañía.

1.4. CONSIDERACIONES EN EL MANEJO DE LOS DATOS

Para el manejo de datos en cualquier medio ambiente de base de datos se deben considerar los siguientes puntos:

Seguridad

Integridad

Respaldo y Recuperación

Concurrencia

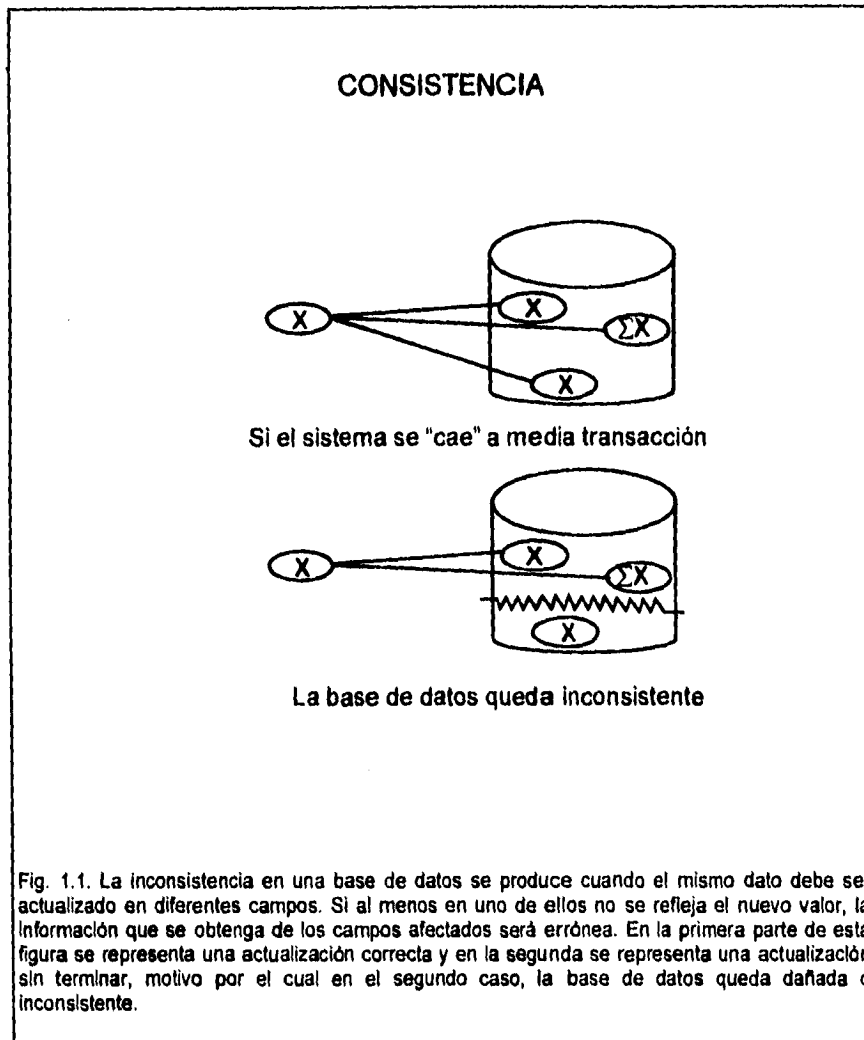
Capacidad de auditoría

1.4.1. SEGURIDAD

Los sistemas de bases de datos deben mantener también la seguridad de la información almacenada para prevenir caídas en el sistema o accesos no autorizados. Además, en los casos en que los datos vayan a ser compartidos, se debe evitar al máximo la duplicidad o la falta de actualización porque puede provocar inconsistencia en los datos (Fig. 1.1)³ con las consecuencias que esto pueda acarrear, de aquí que "los dos puntos clave para la seguridad son impedir que vean los datos aquellas personas que no deban hacerlo y evitar que se modifique información que no deba cambiarse"⁴.

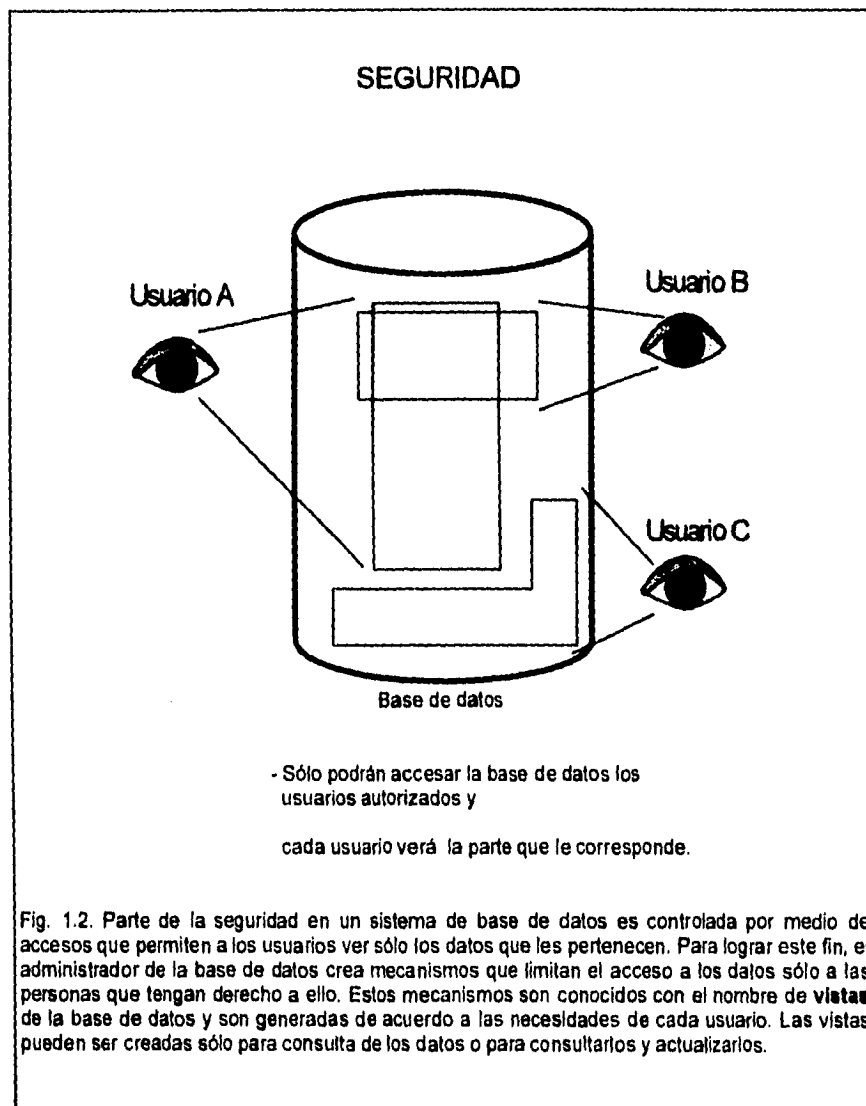
³ Las figuras 1.1, 1.2, 1.3, 1.4, 1.5 de este capítulo fueron tomadas de las páginas 10, 7, 9, 11 y 8 respectivamente del folleto de Bases de Datos. Conceptos, Diseño e Implantación de J. Carlos Talavera, Centro de Computo; División Académica de Computación, Instituto Tecnológico Autónomo de México (ITAM), 1993.

⁴ GUILLENSON, Mark L.; Introducción a las BASES DE DATOS, Mc Graw Hill, 1985, pág. 88.



El concepto de integrar los datos de una organización dentro de un conjunto común accesible a todos tiene ventajas y desventajas. Una ventaja es que áreas con funciones diferentes pueden beneficiarse con el uso de datos que ellos no crean o mantienen. La desventaja correspondiente, es que los datos pueden ser mal usados o dañados por usuarios que no tienen responsabilidad sobre ellos. El DBA suministra los procedimientos y controles para prevenir el abuso en el manejo de los datos.

El DBA crea vistas parciales de la base de datos (Fig. 1.2), de acuerdo a las necesidades de cada grupo y cada grupo se hace responsable de la parte (vista) de la base de datos que le corresponda. El grupo propietario puede entonces, otorgar el acceso a los datos dentro de la vista a otros grupos o personas que estén dentro de la organización. Este acceso puede ser restringido a parte de los datos, ya sea sólo con autorización para consulta o con autorización para actualización.



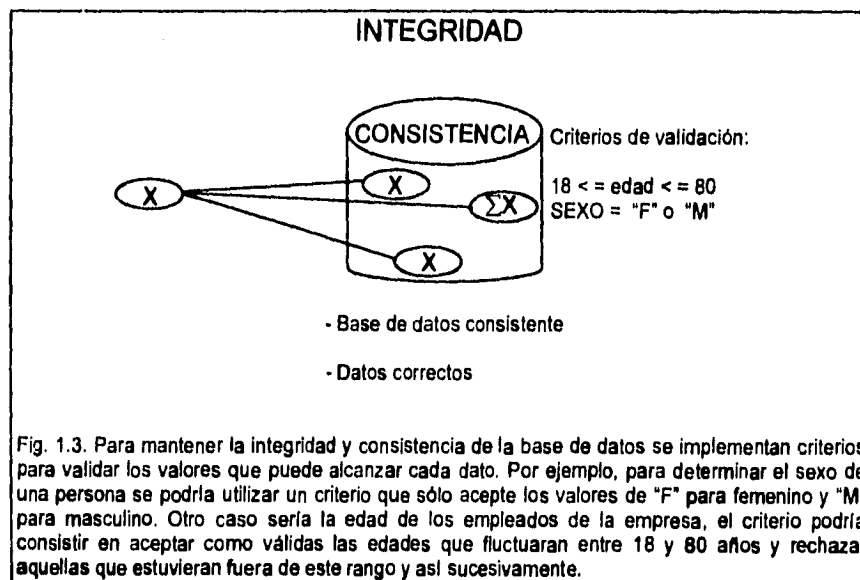
La información en cuanto a derechos de acceso y propiedad de los datos es mantenida bajo la dirección del DBA en el diccionario de datos.

Finalmente, el acceso a la base de datos se hace por medio de un mecanismo asignado por el DBA de contraseña o "password". Cualquier usuario que intente accederla debe proporcionar su password, mismo que es validado por el sistema. Después de que la contraseña pasa el proceso de validación, se le dan los derechos y privilegios que tiene registrados el sistema para el usuario en el diccionario de datos.

El administrador de la base de datos (DBA) es responsable de asignar los passwords y controlar los privilegios a que tienen derecho los usuarios. De este modo, el DBA puede reducir el riesgo de que un grupo de usuarios provoque daños a los datos de otro grupo. Cuando existan dos o más grupos que compartan los mismos datos, el DBA de acuerdo con los responsables de ellos, señalará quién o quiénes y cuándo los actualizarán.

1.4.2. INTEGRIDAD DE LOS DATOS

La integridad de los datos se refiere al problema de mantener la exactitud y consistencia de los datos en la base de datos. Los mecanismos de seguridad tales como passwords y vistas de los datos permiten al DBA evitar que se dañen o modifiquen los datos, debido a errores de actualización o provocado por gente con intereses personales que actúen de mala fe (Fig. 1.3).



1.4.3. RESPALDO Y RECUPERACION

El respaldo se refiere a la necesidad de crear copias de los archivos, en forma periódica, que contengan los datos de la empresa y se generan para prevenir cualquier tipo de desastre natural, como un terremoto, una inundación, etc., o provocado por intervención del hombre, ya sea por descuido o por una acción premeditada. Las copias (es conveniente tener más de una copia) deben ser enviadas fuera de las instalaciones de la empresa y aún a ciudades diferentes. Con esto se impide que una catástrofe destruya todas las copias de los datos. Los dispositivos que más se utilizan para los respaldos son: cintas y discos magnéticos.

Al proceso de utilizar la copia de la información respaldada se le llama recuperación y se realiza cuando alguna circunstancia daña la información (Fig.1.4).

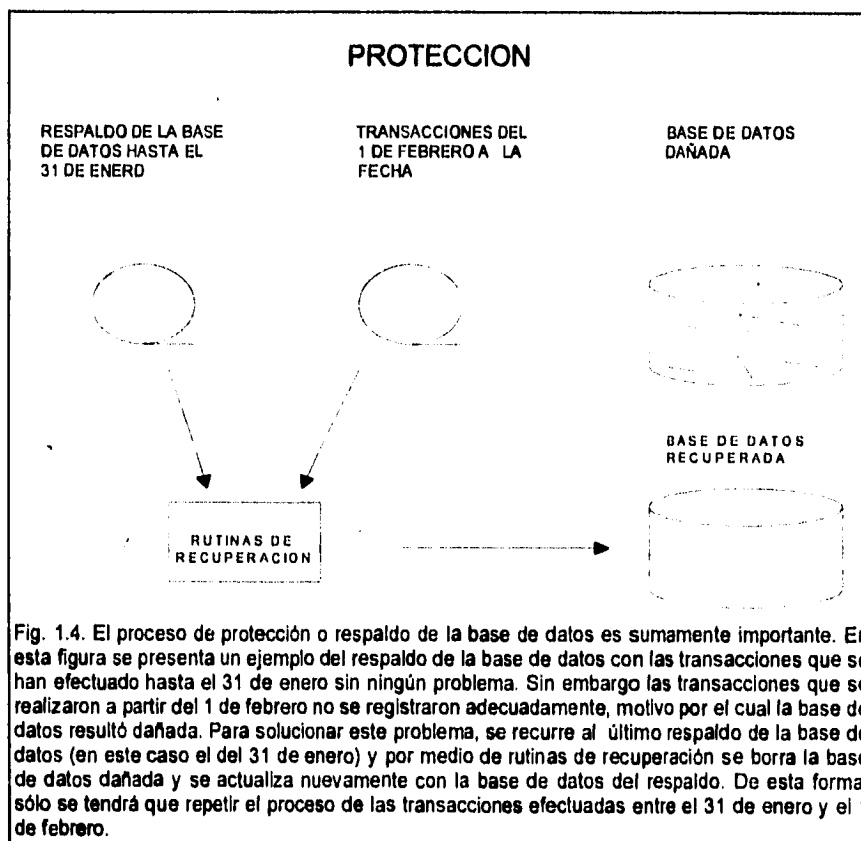
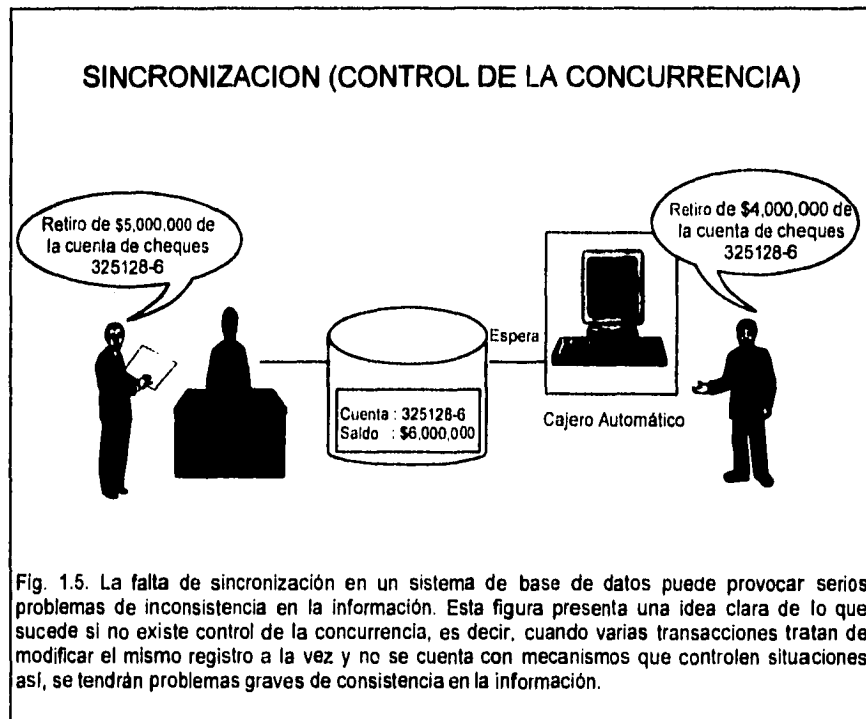


Fig. 1.4. El proceso de protección o respaldo de la base de datos es sumamente importante. En esta figura se presenta un ejemplo del respaldo de la base de datos con las transacciones que se han efectuado hasta el 31 de enero sin ningún problema. Sin embargo las transacciones que se realizaron a partir del 1 de febrero no se registraron adecuadamente, motivo por el cual la base de datos resultó dañada. Para solucionar este problema, se recurre al último respaldo de la base de datos (en este caso el del 31 de enero) y por medio de rutinas de recuperación se borra la base de datos dañada y se actualiza nuevamente con la base de datos del respaldo. De esta forma, sólo se tendrá que repetir el proceso de las transacciones efectuadas entre el 31 de enero y el 1 de febrero.

1.4.4. CONCURRENCIA

Es natural que una aplicación sea accesada por uno o más usuarios a la vez, pero si dichos usuarios actualizan simultáneamente el mismo registro puede acarrear graves complicaciones para alguno de ellos. Por ejemplo, supóngase que se tiene la cuenta bancaria 325128-6 con un saldo de \$6,000,000. Si a la misma hora se trata de retirar de la ventanilla de una sucursal y de un cajero automático \$5,000,000 y \$4,000,000 respectivamente y el sistema no tiene implementado un control de concurrencia, se habrán retirado \$9,000,000, lo cual implicaría un saldo negativo para la institución bancaria de \$3,000,000 (Fig. 1.5).



Para prevenir situaciones como la señalada, los sistemas de bases de datos implementan mecanismos de seguridad que consisten en retener el registro y no permiten que ninguna otra transacción lo utilice hasta que termine su actualización. Además, si el registro es solicitado en el mismo instante por dos o más usuarios, el sistema debe contar con elementos de discriminación que determinen cual usuario accederá el registro primero.

1.4.5. CAPACIDAD DE AUDITORIA

Es importante que cualquier empresa que cuente con un sistema de cómputo para el procesamiento de sus datos implemente las medidas necesarias para saber quién o quiénes tuvieron acceso o realizaron modificaciones y a qué datos. De esta manera se detectaría a quien tratara de sacar algún beneficio modificando los datos.

1.5. PROBLEMAS PARA LA IMPLANTACION DE UNA BASE DE DATOS

1.5.1. CONFLICTOS ORGANIZACIONALES

El manejo común de datos en una base de datos puede no ser políticamente factible en algunas organizaciones, cierto grupo de usuarios puede no estar dispuesto a ceder el control sobre sus datos a las necesidades de extender e integrar esos datos dentro de una base de datos. Además, el riesgo envuelto en los datos compartidos - por ejemplo, el que un grupo pueda dañar los datos de otro grupo - y los problemas potenciales del sistema que pueden limitar el acceso de otros grupos a sus propios datos pueden ser vistos más como desventajas que como ventajas.

Cuestiones como las mencionadas pueden evitar la adecuada implantación de un sistema de base de datos.

1.5.2 FALLAS EN EL DESARROLLO DEL PROYECTO

El proyecto para desarrollar un sistema de base de datos puede fallar por varias razones: algunas veces el responsable de la empresa no está totalmente convencido de las bondades de un sistema de base de datos y por lo tanto no lo autoriza; en otros casos, si el proyecto de la base de datos parece estar tomando demasiado tiempo puede provocar que se retire el apoyo, lo que implicaría no terminarlo. En otras ocasiones el proyecto de base de datos es muy grande en alcance y llega a ser casi imposible de realizar en un tiempo razonable, por lo que el responsable y los usuarios pueden desilusionarse y hacer que falle, antes de buscar otra alternativa que permita su culminación.

También puede suceder que durante el desarrollo del proyecto de la base de datos, el personal clave deje inesperadamente la compañía. Si no se consigue el personal adecuado para reemplazarlo se corre el riesgo de que el proyecto quede sin concluir.

Por tanto, se puede concluir que para superar los problemas mencionados y pensar en implantar un sistema de base de datos, es aconsejable que antes de

iniciar el diseño se haga un análisis que tome en cuenta los requerimientos de información de la empresa, así como sus posibilidades económicas.

CAPITULO 2

REQUERIMIENTOS Y HERRAMIENTAS DE LAS BASES DE DATOS

2.1. SISTEMA DE BASE DE DATOS

Los datos de una empresa acerca de sus productos, procesos de manufactura, clientes, proveedores, empleados, competidores y así sucesivamente, pueden, con almacenamiento y empleo adecuado ser controlados y accedidos rápidamente. Es necesario, entonces, contar con algún tipo de sistema uniforme que maneje los datos como lo que son: un recurso de la empresa.

El medio ambiente de base de datos, con su software de apoyo y las técnicas de manejo que le acompañan, proporcionan el marco para tratar los datos como un recurso estandarizado, administrable y compartible. Lo que permite manejar grandes volúmenes de información.

Un sistema de base de datos está formado por un conjunto de datos interrelacionados y un conjunto de programas que accesan esos datos. Los datos en un sistema de base de datos, deben cumplir con los siguientes atributos:

Estar interrelacionados

Tener capacidad de evolución

Ser accesibles a múltiples aplicaciones

Presentar redundancia mínima y controlada.

Una de las capacidades de los sistemas de manejo de base de datos es que proporcionan el software para facilitar el acceso a los datos de manera natural.

2.2. SOFTWARE EN LOS SISTEMAS DE BASES DE DATOS

Un sistema de base de datos incluye dos tipos de software:

1. Software de manejo de base de datos de propósito general, usualmente llamado **Sistema Manejador de Base de Datos, DBMS** por sus siglas en inglés (Database Management System).

2. Software de aplicación que usa el DBMS para manipular la base de datos y realizar una función específica de la empresa, tal como emitir un informe o el análisis de la tendencia de ventas.

El software de aplicación es generalmente escrito por empleados de la compañía para solucionar los problemas específicos de la compañía. El software de aplicación usa las facilidades que proporciona el DBMS para acceder y manipular los datos de la base de datos y generar los reportes y documentos que cubran las necesidades de información y proceso de la compañía.

El sistema manejador de bases de datos es software del sistema, similar al que utiliza un sistema operativo o un compilador y suministra un número de servicios para usuarios finales, programadores y otros. Como su nombre lo dice, el DBMS existe para facilitar el manejo de una base de datos y generalmente proporciona varios de los siguientes servicios:

Una definición centralizada que permite el control de los datos, a través del **diccionario de datos (DD)**.

Mecanismos de seguridad y de integridad de los datos

Mecanismos de control de concurrencias en el acceso a los datos por múltiples usuarios

Capacidad para reportar, manipular y consultar los datos orientados al usuario

Facilidades orientadas a los programadores para el desarrollo de sistemas de aplicación

2.3. LENGUAJES DE DEFINICION Y DE MANIPULACION DE DATOS

2.3.1. LENGUAJE DE DEFINICION DE DATOS

La forma como los datos se almacenan en la base de datos se define por medio de un lenguaje especial llamado **lenguaje de definición (o descripción) de datos (DDL, Data Definition Languages)** y es importante por dos razones: una la facilidad que requiere el programador al acceder los datos y la otra para aprovechar de manera completa la tecnología del hardware y alcanzar todos los beneficios de la base de datos.

De esta manera, el DDL debe tener la capacidad de proporcionar dos *visiones* de los datos:

"Se habla de la visión lógica de los datos como la forma en que el programador percibe que están, mientras que la visión física refleja la manera en que realmente están almacenados en disco.

Para que esta distinción funcione, debe haber un componente en el DBMS que pueda convertir las solicitudes de datos que provienen de las terminales o programas de aplicación basados en la visión original (lógica), a la forma de visión física, que es aquella en la que en realidad están almacenados en el disco los datos. Y para que el DBMS lo logre, debe tener a su disposición descripciones que pueda *comprender* sobre la manera en que los datos están almacenados en el disco, y la forma en que los usuarios o programadores los perciben. Tal información está almacenada en grupos especiales de registros conocidos como bloques de control⁵.

2.3.2. LENGUAJE DE MANIPULACION DE DATOS

Para *trabajar* con los datos se requiere de un lenguaje especial al que se le llama ***lenguaje de manipulación de datos*** y se refiere al mecanismo empleado para recuperar datos almacenados en una base.

Existen dos maneras básicas de hacerlo; la primera consiste en hacer que un programa de aplicación envíe una instrucción al DBMS para que encuentre ciertos datos en la base y los devuelva al programa. A los procedimientos de este tipo se les conoce con el nombre de **proposiciones integradas**.

La otra forma para acceder los datos se hace por medio de una terminal, en la que una persona emite un comando de manera directa al DBMS para que los encuentre y los despliegue en la pantalla; a esto se le conoce como **operación de consulta**.

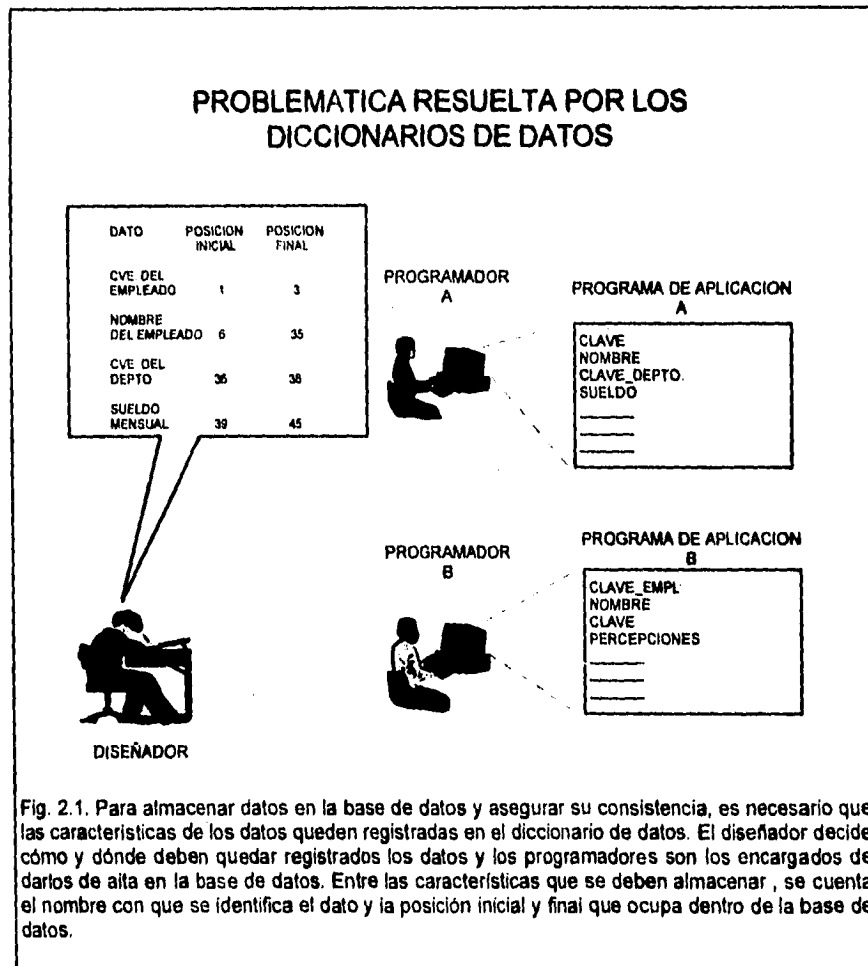
En un lenguaje de consulta existe un procesador de aplicación general capaz de buscar y recuperar cualquier dato, siempre y cuando esté almacenado de acuerdo con la estructura prescrita del DBMS, y la construcción de la misma consulta siga las convenciones adecuadas. De esta manera, cualquiera que emplee un lenguaje de consulta debe poder consultar la información disponible (siempre y cuando tenga la autorización para ello) aun cuando desconozca como están almacenados los datos.

2.4. DICCIONARIO DE DATOS

Los sistemas de bases de datos intentan, con diferentes grados de éxito, sobrepasar las limitaciones de los sistemas de archivos tradicionales. Para llegar a una integración, centralizan la estructura de datos, además los sistemas de bases de datos eliminan los problemas relacionados con la redundancia y control de datos. Una base de datos centralizada, está disponible para toda la compañía y si por ejemplo, el nombre de un cliente debe ser cambiado, el cambio está

⁵ GUILLENSON, Mark L. Introducción a las BASES DE DATOS, Mc Graw Hill, 1986, pág. 97.

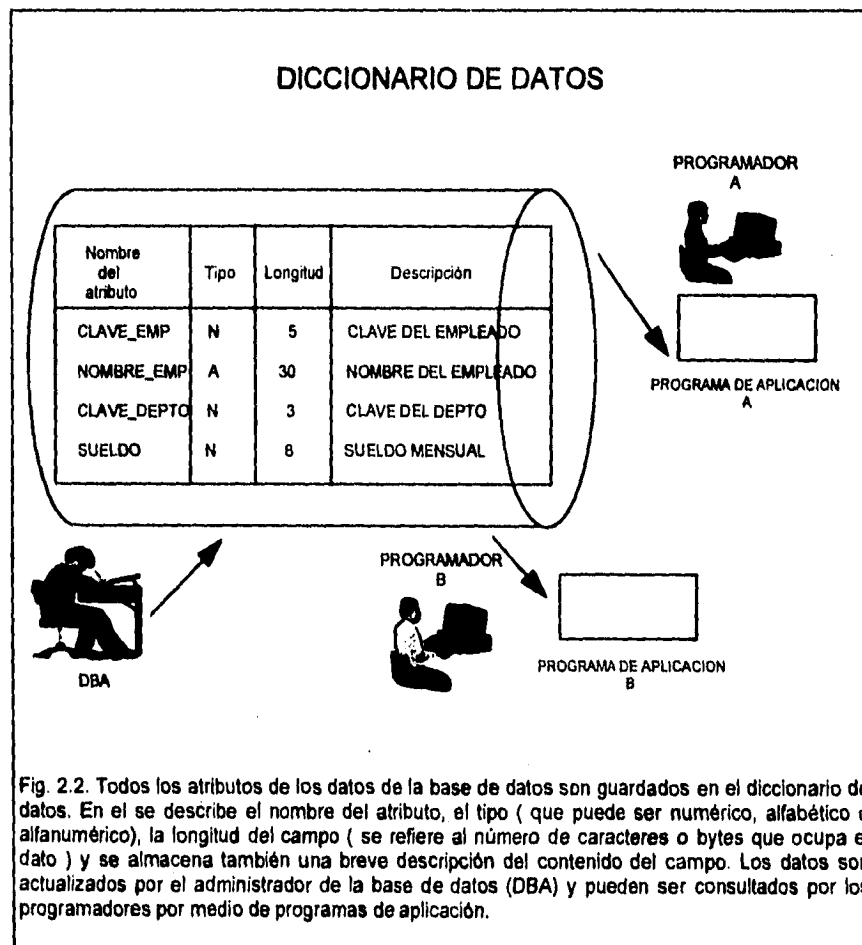
disponible para todos los usuarios. Los datos son controlados vía el Diccionario de Datos (ver figura 2.1)⁶, mismo que es manejado por un grupo de empleados de la compañía que saben cómo administrar la base de datos (DBA, Database Administrators). Los nuevos métodos de acceso simplifican grandemente los procesos relacionados, lo que facilita la manipulación de los datos. Todas estas características de los sistemas de bases de datos facilitan los esfuerzos de programación y reducen el mantenimiento de los programas.



⁶ Las figuras 2.1 y 2.2 de este capítulo fueron tomadas de las páginas 32 y 33 respectivamente del folleto de Bases de Datos. Conceptos, Diseño e Implantación de J. Carlos Talavera, Centro de Computo; División Académica de Computación, Instituto Tecnológico Autónomo de México (ITAM), 1993.

Dentro de la información que guarda el DD se encuentran: los nombres de los datos, archivos, pantallas y la manera como están constituidos los reportes.

Además, en el diccionario de datos se lleva la cuenta de todos los datos de la base de datos, esto incluye el nivel elemental de los datos, grupos, estructuras de datos a nivel de registro y archivos o tablas relacionales (Fig. 2.2) . También guarda las relaciones que existen entre las estructuras de los datos. Adicionalmente, mantiene los índices que son utilizados para acceder rápidamente a los datos. Guarda las definiciones de los formatos de las pantallas y los reportes que son usados por varios programas de la aplicación y también controla el significado de los homónimos y de los sinónimos.



El DD puede ser visto como una parte de la base de datos. A la información que se encuentra en el DD se le llama **metadato** o **dato acerca del dato** (Fig. 2.3). El metadato está disponible para consulta y manipulación como cualquier otro dato en la base de datos. Sin embargo, la manipulación debe ser cuidadosamente controlada, ya que los datos en el DD son vitales para el funcionamiento adecuado de todo el sistema de la base de datos.


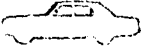
REALIDAD	DATOS (USUARIO)	METADATO
	12345 MARIA PEREZ SANCHEZ PATITO, S.A. GERENTE	NUM. DE IDENTIFICACION NOMBRE APELLIDOS COMPANIA PUESTO
	CHEVROLET CAVALIER BLANCO 1994	MARCA MODELO COLOR AÑO

Fig. 2.3. Un metadato da información acerca de un dato. Por ejemplo, MARIA es un dato que da información acerca de una persona y a su vez NOMBRE se refiere al dato (Maria) que identifica a esa persona (dato acerca del dato). Otro ejemplo, CAVALIER da referencia acerca de un AUTO y MODELO se refiere al nombre de ese AUTO (Cavalier), y así sucesivamente.

2.5. PROCESO DE DESARROLLO DE LA BASE DE DATOS

Aunque puede variar de acuerdo a los requerimientos y capacidad económica de la empresa, se considera que el ciclo de vida del desarrollo de una base de datos DDLC (por sus siglas en inglés, Database Development Life Cycle) se lleva al cabo en seis etapas:

- 1.- Planeación preliminar.
- 2.- Estudio de factibilidad.
- 3.- Definición de requerimientos.

- 4.- Diseño lógico.
- 5.- Implantación .
- 6.- Evaluación de la base de datos y mantenimiento.

2.5.1. PLANEACION PRELIMINAR

La organización estratégica del proyecto de la base de datos, tiene lugar dentro de la planeación preliminar. Durante este proceso, la empresa reúne información suficiente para contestar al menos las siguientes preguntas:

- 1.- ¿Cuántos programas de aplicación están en uso y qué funciones ejecutan ?
- 2.- ¿Cuáles archivos son utilizados por cada una de estas aplicaciones ?
- 3.- ¿Qué nuevas aplicaciones y archivos están en desarrollo ?

Esta información puede ser usada para establecer las interrelaciones entre las aplicaciones actuales y para identificar el uso que se le da a la información de la aplicación. Puede también ayudar a identificar futuros requerimientos del sistema y a estimar tiempo y el potencial beneficio económico de la instalación de un sistema de base de datos.

2.5.2. ESTUDIO DE FACTIBILIDAD

Un estudio de factibilidad involucra la preparación de un reporte en cada uno de los siguientes casos:

- 1.- Factibilidad tecnológica. ¿Se dispone de la tecnología para soportar el desarrollo de la base de datos?
- 2.- Factibilidad operacional. ¿Tiene la compañía el personal, el presupuesto y expertos internos para hacer exitoso un sistema de base de datos?
- 3.- Factibilidad económica. ¿Pueden ser identificados los beneficios? ¿Los costos y beneficios pueden ser medidos ?

Factibilidad tecnológica. En este estudio se analiza si la compañía cuenta con la capacidad y recursos tecnológicos para implantar la base de datos o tendrán que

ser adquiridos y si es necesario entrenamiento especial para su personal y cuánto tiempo se llevará la capacitación.

Factibilidad operacional. En esta etapa se hace un análisis de las habilidades con las que debe contar el personal operativo y el tiempo que será necesario para implantar el sistema de base de datos.

Factibilidad económica. Este estudio resulta un desafío por lo que implica, ya que en general los beneficios esperados de un sistema de base de datos no son fáciles de cuantificar, por eso es bueno plantearse las siguientes preguntas:

- a) ¿Qué tan pronto son esperados los beneficios ?
- b) ¿Es factible políticamente compartir los datos de diferentes departamentos ?
- c) ¿Qué riesgos están presentes en la implantación de un sistema de base de datos?
- d) ¿Qué aplicaciones serán implementadas y cuáles son los beneficios que se esperan ?

2.5.3. DEFINICION DE REQUERIMIENTOS

La definición de requerimientos involucra la definición del alcance de la base de datos, la identificación de los requerimientos de información de la gerencia, la determinación de los requerimientos de información por área y el establecimiento de los requerimientos de hardware y software.

2.5.4. DISEÑO LOGICO DE LA BASE DE DATOS

En la etapa del diseño lógico, se crea el esquema lógico o conceptual de la base de datos entera.

"El diseño lógico de la base de datos tiene dos componentes, el primero de los cuales significa organizar los campos de datos en agrupamientos no redundantes basados en las relaciones de los datos. El segundo abarca una organización inicial de los procedimientos lógicos en estructuras basadas en la naturaleza del software que maneja la base de datos y de las aplicaciones que utilizarán los datos"⁷.

⁷ GUILLENSON, Mark L. Introducción a las BASES DE DATOS, Mc Graw Hill, 1985, pág. 260.

El diseño lógico de la base de datos consiste primeramente de la identificación y definición de los elementos de los datos que serán incluidos en la base de datos, las relaciones que existen entre ellos, la restricción de valores que pueden llegar a tener y sus parentescos. Una restricción de valor señala los valores permitidos para un dato.

Para realizar la función del diseño lógico de la base de datos, el equipo técnico que administra la base de datos debe incluir personal que sea experto y entienda los conceptos del diseño y además tener habilidad en el trabajo con grupos de usuarios los cuales deben proporcionar la información básica necesaria para el diseño de la aplicación.

En un sistema de base de datos se tiene que contemplar tanto el aspecto físico de la base como el aspecto lógico.

El aspecto físico se refiere a la forma como quedan almacenados los datos en los discos de acuerdo al hardware que se tenga y el lógico se refiere a la forma como un usuario ve los datos de la base.

En algunos casos, el almacenamiento físico y lógico de los datos coinciden pero en general esto no es así.

Se puede decir que un diseño lógico apropiado es esencial para que un sistema de base de datos funcione adecuadamente. Esto permite obtener beneficios económicos y de información para la empresa.

Vistas de una base de datos. En la etapa del diseño lógico también se crean las vistas individuales de los usuarios y se registran todos los datos que serán mantenidos en la base de datos.

Los **administradores de la base de datos (DBA)** trabajan con los usuarios en varias áreas y diseñan partes de la base de datos. *La porción del diseño de la base de datos, que un grupo de usuarios dado tiene, es una vista de la base de datos.* Estas vistas deben ser entonces integradas dentro del esquema completo de la base de datos el cual define la estructura lógica de la base de datos entera.

El proceso del diseño lógico requiere la resolución de conflictos entre diferentes grupos de usuarios, por ejemplo, diferentes grupos pueden usar el mismo término en formas contradictorias. Más aun los grupos probablemente estén celosos de sus datos y se resisten a la posibilidad de permitir a otros accesarlos. Un control razonable debe ser establecido para definir cuáles grupos pueden acceder determinados datos y si sus accesos incluirán la capacidad de actualización de los valores de los datos. Los miembros del equipo técnico de administración de la base de datos debe poder negociar las resoluciones a conflictos como éstos.

2.5.5. IMPLANTACION DE LA BASE DE DATOS

Durante la etapa de implantación de la base de datos, se selecciona y adquiere el DBMS, se construye el diccionario de datos, la base de datos es poblada, los programas de aplicación son desarrollados y los usuarios son entrenados.

2.5.6. EVALUACION Y MANTENIMIENTO

La evaluación implica entrevistas y encuestas a los usuarios para determinar si las necesidades de información están cubiertas o hacen falta modificaciones. Los cambios a la base de datos se realizan de acuerdo a las necesidades de la empresa. Con el tiempo el sistema de base de datos es mantenido y mejorado con la adición de nuevos programas y datos, conforme los requerimientos cambian y se expanden.

El diseño inicial de la base de datos y la implementación deben ser seguidos por un mantenimiento continuo de las estructuras físicas y lógicas de la base de datos.

2.6. ARQUITECTURA DE LAS BASES DE DATOS

En general los estudiosos de las bases de datos, entre ellos C.J. Date⁸ coinciden en que *la arquitectura de una base de datos se divide en tres niveles, denominados niveles interno, conceptual y externo.*

Si el nivel externo se ocupa de las vistas individuales de los usuarios, puede considerarse que el nivel conceptual se ocupa de una vista comunitaria de los usuarios.

En otras palabras, existen muchas vistas externas distintas que representan, cada una de ellas, sólo una parte de la base de datos y responden a las necesidades particulares de cada usuario o grupo de usuarios (a la mayoría de los usuarios no les interesa toda la base de datos sino sólo una parte de ella).

Por otro lado, existe sólo una vista conceptual formada por una representación abstracta⁹ de la base de datos en su totalidad.

⁸ DATE, C. J. : Introducción a los sistemas de bases de datos, trad. Jaime Malpica; Américo Vargas Villazón, 3 ed., Wilmington, Delaware, E. U. A. Addison Wesley Iberoamericana, S. A.

⁹ En este contexto, una representación abstracta implica construcciones orientadas hacia el usuario, como son los registros y campos lógicos, y no construcciones orientadas hacia la máquina como los bits y los bytes.

De manera similar, habrá sólo una *vista interna*, la cual *representará a toda la base de datos tal como está almacenada físicamente*.

1) Nivel Físico o Interno.

El nivel interno es el que se ocupa de la forma como se almacenan físicamente los datos en la base.

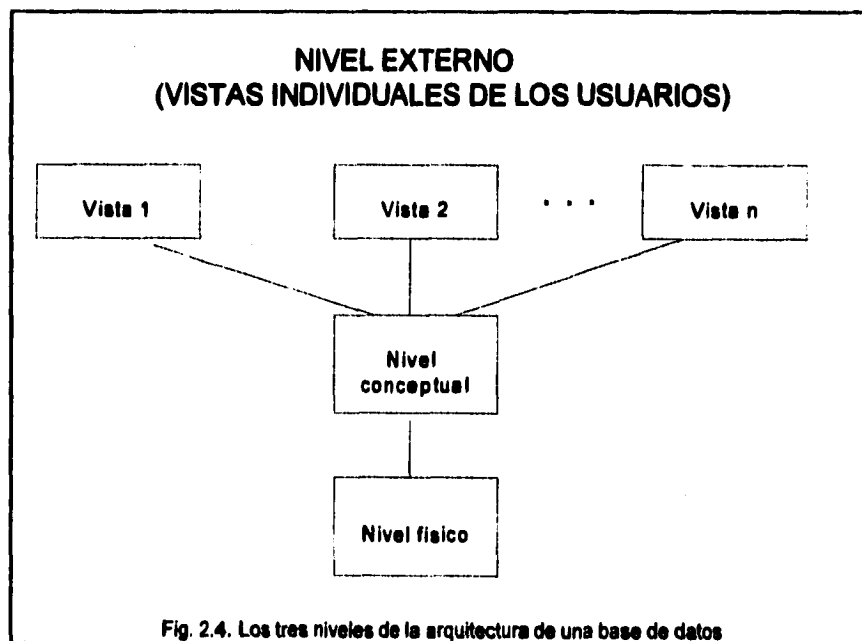
2) Nivel Conceptual.

El nivel conceptual es un nivel de mediación y en él se describen cuáles son los datos que realmente están almacenados en la base de datos y las relaciones que existen entre ellos. El usuario del nivel conceptual no se da cuenta realmente de la forma como están almacenados los datos en el nivel físico.

3) Nivel de Visión o Externo.

Este es el nivel más cercano a los usuarios, es decir, es el que se ocupa de la forma como los usuarios individuales perciben los datos.

La figura 2.4., representa los tres niveles de una base de datos.



Algunos modelos de bases de datos. Dentro de los modelos de bases de datos más conocidos se encuentran los jerárquicos, los de redes, los relacionales, los distribuidos y los orientados a objetos, entre otros.

Cada uno de ellos representa características propias para el almacenamiento y recuperación de los datos, así como de representación. Una vista jerárquica, por ejemplo, se representa por medio de estructura de árbol. Mientras que el modelo de redes se representa por medio de registros y ligas. La representación del modelo relacional se hace por medio de tablas y así sucesivamente.

2.7. DIRECCION ACTUAL DE LOS SISTEMAS DE BASES DE DATOS

A pesar de que los sistemas basados en manejo de archivos tradicionales son todavía abundantes y dan soluciones efectivas a un buen número de aplicaciones, hoy en día los sistemas relacionales están considerados lo último en los estándares de las operaciones de proceso de datos comerciales y se observa una clara tendencia de las compañías hacia la implantación de estos sistemas, siempre y cuando sea factible.

Las investigaciones adicionales prometen dar un entendimiento más completo de las necesidades del usuario con respecto a las bases de datos y en consecuencia continúa el desarrollo de nuevos modelos.

Los siguientes capítulos, abordarán el tema del diseño lógico de las bases de datos relacionales y de las bases de datos orientadas a objetos.

En el capítulo tres, se describe el modelo relacional y los elementos que intervienen, la importancia que tiene actualmente como modelo de base de datos y la evolución que ha mostrado, con cambios significativos en su concepción natural para permitir a los usuarios atacar problemas más complejos.

Los capítulos cuatro y cinco abordan el tema del diseño lógico desde el enfoque de las bases de datos orientadas a objetos. También se describen elementos como clase, objeto, entre otros, y la importancia que tiene este modelo de vanguardia.

CAPITULO 3

DISEÑO LOGICO DEL MODELO DE BASE DE DATOS RELACIONAL

"Con los primeros sistemas de bases de datos se hizo patente la necesidad de mayor independencia en los datos. La estructura lógica general de los datos se hizo más compleja en muchos casos y al crecer en tamaño las bases de datos se hizo inevitable el cambio de la estructura lógica general. Resultó importante que esa estructura general pudiese cambiar sin forzar el cambio de los programas de aplicación que la utilizaban"¹⁰.

Para tener una idea clara de lo que significa el **diseño lógico** en los sistemas de bases de datos, en este capítulo se hablará del **sistema de base de datos relacional**, elegido por sus fundamentos matemáticos que se basan en sólidos conceptos derivados de el álgebra y el cálculo relacional y porque ha servido de plataforma para la investigación y desarrollo de nuevos modelos de bases de datos.

El grado de complejidad de muchas Bases de Datos parece crecer sin límites previsible, motivo por el cual los diseñadores deben tener muy claro lo que se necesita del sistema para evitar que éste se transforme en una maraña de datos e interrelaciones. En los sistemas de bases de datos relacionales se recurre a una técnica denominada **normalización**, para prevenir situaciones que provoquen confusión en el manejo de los datos.

Aunque el concepto de normalización incluye "n" formas normales, es importante señalar que si bien la tercera forma normal no es la última, si se considera que un modelo de diseño lógico de base de datos relacional desarrollado en tercera forma normal puede ser implantado con bastante éxito en una empresa, de acuerdo a la concepción original enunciada por Codd.

Más adelante, en el capítulo correspondiente al modelo de bases de datos relacional, se explicará en qué consisten la primera, segunda y tercera formas normales y la metodología para llegar a ellas. Para aclarar tal concepto de normalización se recurrirá a un ejemplo que se irá desarrollando desde la forma no normalizada, hasta llegar a la tercera forma normal pasando desde luego por lo que implican la primera y segunda formas normales.

Las demás formas normales, aunque significativas, sólo contemplan casos especiales que se presentan en raras ocasiones, motivo por el cual sólo se

¹⁰ MARTIN, James, Organización de las Bases de Datos, trad. Adolfo Di Marco, 1er ed., México, Prentice - Hall Hispanoamericana, S. A., 1988, pág. 27.

describirán breves ejemplos de la cuarta forma normal y la forma normal llamada de Boyce and Codd, con lo que se concluirá este capítulo.

También, en otro capítulo aparte, se hablará del diseño de las bases de datos orientadas a objetos y su importancia dentro de los nuevos modelos de bases de datos.

Finalmente, es necesario señalar que este trabajo sólo estará dedicado al diseño del modelo lógico, sin que esto reste importancia al diseño físico de una base de datos.

3.1. ANTECEDENTES DE LAS BASES DE DATOS RELACIONALES

En 1970 E.F. Codd publicó un documento revolucionario que desafió el conocimiento convencional establecido de la base de datos. Codd argumentó que los datos debían ser relacionados de una manera lógica y natural y que estuviera inherente en los datos, más que a través de apuntadores o "pointers" físicos (un pointer representa una dirección física que identifica dónde puede ser encontrado un registro en el disco. Por ejemplo, cada registro de un cliente contendría un apuntador al primer registro de la factura para ese registro del cliente. El registro de la factura en turno contendría apuntadores a otros registros de la factura para facturar nuevas líneas de registros). Así, la gente podría combinar los datos de diferentes fuentes.

En el mismo documento, Codd propuso un modelo simple de datos en el cual todos los datos estarían representados en tablas construidas por renglones y columnas. A estas tablas se les dio el nombre matemático de relaciones, de aquí se tomó el nombre de modelo relacional. Codd también propuso dos lenguajes para la manipulación de los datos en las tablas, el álgebra relacional y el cálculo relacional

Para manejar los datos sobre bases conceptuales más que físicas, Codd introdujo otra innovación revolucionaria. En los sistemas de bases de datos relacionales la totalidad de los archivos pueden ser procesados con instrucciones sencillas y en contraste, los sistemas tradicionales requieren que los datos sean procesados un registro a la vez. La proposición de Codd mejoró significativamente la eficiencia conceptual en programación de la base de datos.

La forma de manipulación de los datos también hizo factible la creación de lenguajes de consulta más accesibles a usuarios sin conocimientos técnicos.

Si bien es muy difícil crear un lenguaje que pueda ser usado por toda la gente, sin tener en cuenta sus conocimientos y experiencia previa en computación, el

lenguaje de consulta relacional hace sencillos los accesos a las bases de datos para un gran número de grupos de usuarios.

3.2. OBJETIVO DEL DISEÑO DE LA BASE DE DATOS RELACIONAL

El objetivo del diseño de una base de datos relacional es generar un conjunto de tablas o relaciones que almacenen datos sin que haya redundancia y que a la vez permitan recuperarlos fácilmente.

Esto implica ordenar los campos de los datos que serán utilizados por una o más aplicaciones, poniéndolos en una estructura organizada en la que se propicien las relaciones necesarias entre los campos cumpliendo al mismo tiempo con las restricciones físicas del sistema específico de manejo de base de datos que se esté utilizando.

Para alcanzar este fin, es necesario entonces conocer a fondo la información real que maneja la empresa de donde se pretende hacer el diseño de la base de datos, porque esa información es el soporte para la creación de dicho diseño.

3.3. REQUERIMIENTOS DEL MODELO DE BASE DE DATOS RELACIONAL

En un sistema tradicional es fácil encontrar redundancia que puede crear una serie de problemas graves, empezando por el trabajo excesivo que se necesita para actualizar el valor de los campos en los archivos y finalizando por la enorme cantidad de datos, lo que implica grandes volúmenes de almacenamiento.

Las bases de datos relacionales ayudan a eliminar la redundancia en los datos y permiten accesarlos de una manera relativamente mucho más fácil y rápida, a través de tablas de n columnas por m renglones, el usuario está familiarizado con tal estilo de representación, lo comprende, visualiza y recuerda sin dificultad.

Las tablas deben organizarse de forma tal que no se pierda ninguna de las relaciones existentes entre los datos.

3.4. DATOS NO NORMALIZADOS Y NORMALIZACION DE LOS DATOS

*Tanto los datos no normalizados como los normalizados conforman una **relación** a la que se le denomina **universal** y en ella están contenidas todas las entidades y relaciones de un sistema.*

3.4.1. DATOS NO NORMALIZADOS

Los datos no normalizados pueden presentar situaciones como un campo redundante o multivaluado.

La Fig. 3.1 muestra una tabla sin normalizar con los datos que maneja una empresa ficticia a la que se llamará "La Gran Venta" y servirá de ejemplo para indicar los pasos que se siguen para normalizar una tabla que no lo está.

En la tabla se representan los datos, que de manera hipotética maneja la empresa. Allí se observa que número-producto, nombre-producto, cantidad-ordenada, precio-producto y total-producto, se repiten en diversas ocasiones. Esta forma de datos tiene ciertas desventajas, como la necesidad de alguna forma de registros de longitud variable y la complejidad adicional para el programador, lo que invariablemente llevará a un número mayor de errores de programación.

DATOS QUE MANEJA LA EMPRESA "LA GRAN VENTA"

ORDEN	FECHA-ORDEN	NÚMERO-CLIENTE	NOMBRE-CLIENTE	DIRECCION-CLIENTE	NÚMERO-PRODUCTO	NOMBRE-PRODUCTO	CANTIDAD-ORDENADA	PRECIO-PRODUCTO	TOTAL-PRODUCTO	TOTAL-ORDEN

Fig. 3.1.Registro sin normalizar (contiene varios campos con repetición o multivaluados como son : número-producto, nombre-producto, cantidad-ordenada, etc.)

3.4.2. NORMALIZACION DE LOS DATOS

Para realizar un buen diseño lógico de una base, sus datos deben ser normalizados. La normalización de datos es una metodología para "arreglar" campos en tablas que deben cumplir con ciertas propiedades, de manera que se elimine la redundancia entre los campos, es decir, es una técnica en el diseño de bases de datos mediante la cual se pueden agrupar los atributos en tablas (archivos en los sistemas tradicionales), logrando mínima redundancia y haciendo posible el uso del cálculo relacional o del álgebra relacional; la agrupación es tal que no existe pérdida de información.

Cada una de las tablas resultantes se ocupa de una sola área de conocimiento, por ejemplo *tabla de empleados*, *tabla de clientes*, *tabla de productos*, etc. La

* Todos las figuras que servirán para ejemplificar los pasos de la primera, segunda y tercera forma normal fueron tomadas de los apuntes del Diplomado de Base de Datos, impartido por el Instituto Tecnológico y de Estudios Superiores de Monterrey, en la ciudad de México en el año de 1992.

normalización es un proceso que permite paso a paso reemplazar tablas con datos redundantes o repetidos hasta llegar a tablas con n columnas y m renglones.

En general a una tabla de n columnas por m renglones se le llama relación. De esta manera la base de datos construida por medio de relaciones es una base de datos relacional.

El grado de una tabla se determina de acuerdo al número de columnas que contiene, es decir, si la relación tiene n columnas se dice entonces que la relación es de grado n . Así las relaciones de grado 2 se llaman binarias, las de grado 3 ternarias y las de grado n enearias. La Fig. 3.2 muestra un ejemplo de una tabla de grado 3 (ternaria).

EJEMPLO DE UNA TABLA DE GRADO 3 (TERNARIA).

# Nómina	Nombre-empleado	Salario
1221-2	Juan	N\$2450.00
1532-7	José	N\$3876.00
1658-6	Pablo	N\$8200.00
1354-3	Pedro	N\$5300.00

Fig. 3.2. Tabla compuesta por los atributos: #Nómina, Nombre-empleado, Salario.

Por definición, cada columna de la relación conforma un dominio. De esta manera, en el ejemplo de la figura IV.2 los números: 1221-2, 1532-7, 1658-6 y 1354-3 conforman el dominio de #Nómina; Juan, José, Pedro y Pablo conforman el dominio de Nombre-empleado y así sucesivamente. Con ello se tiene que la columna j -ésima conforma el dominio j -ésimo de la relación y debe contener valores del mismo tipo: si en la columna se almacenan nombres de empleados ésta debe contener sólo nombres de empleados, si son direcciones debe almacenar únicamente direcciones y así con los demás dominios de la tabla. Esto representa la **regla de integridad de valor**, la cual restringe a que un atributo¹¹ sólo pueda tomar valores que se encuentren dentro del dominio al que pertenece.

¹¹ Unidad de datos más pequeña que se puede definir; son ejemplos de atributo, el nombre, dirección, ciudad, número de cuenta, clave del estado, adeudo y pago en bruto. Un ATRIBUTO o DATO debe estar definido con precisión por el tipo, tamaño, intervalo de valores, etc. Otro nombre con el que se conoce al atributo es el de CAMPO. Técnicamente, el DATO es la definición LÓGICA de los DATOS, mientras que el CAMPO se refiere a los DATOS FÍSICOS contenidos en un REGISTRO. Por ejemplo, el DATO "nombre" define el tipo de DATO almacenado en el CAMPO "nombre" de 138 CARACTERES. Sin embargo DATO, CAMPO Y ATRIBUTO se usan indistintamente.

3.5. PROPIEDADES DE LAS TABLAS EN UNA BASE DE DATOS

Las tablas deben cumplir las siguientes propiedades:

1. Cada registro o renglón de la tabla representa un conjunto de datos y estos se dividen por columnas
2. Todos los elementos de una columna son de la misma clase y conforman un dominio
3. Cada columna de datos tiene nombre propio
4. Todos los registros o renglones de la tabla son únicos, es decir, no hay duplicados
5. El orden en que se tengan los registros no afectan su contenido

3.6. ACCESO A LA BASE DE DATOS

Para acceder a una tabla de la base de datos se requiere de un atributo o de un grupo de atributos que la identifiquen unívocamente. A este tipo de atributos se les conoce como llaves, cada tabla en una **base de datos relacional** posee una o varias llaves.

3.6.1. LAS LLAVES Y SU DEFINICION

Una **llave** es el conjunto mínimo de atributos que permiten acceder a cada registro o renglón de una tabla, por lo que una llave debe comprenderse como única y no puede haber una llave igual para otra tabla diferente.

Una llave puede estar formada por sólo un atributo o por la concatenación de varios atributos (si es este el caso entonces se dirá que es una llave compuesta, por dos o más atributos).

Definición de llaves. Para poder definir una llave es necesario conocer las reglas bajo las cuales operan los datos en la tabla, lo cual exige entender previamente y de una manera total los requerimientos que la base de datos debe cumplir.

Una tabla puede tener más de una llave, al conjunto de llaves que posee una tabla se les llama **llaves candidatas**, de entre las llaves candidatas se escoge una para trabajar con ella en la tabla y se le denomina **llave primaria**¹².

Para hacer la elección de una llave primaria se tiene que considerar tanto las necesidades de los usuarios como los requerimientos del sistema.

DEFINICION.

Una llave candidata es un conjunto de atributos que puede convertirse en llave primaria.

Propiedades de las llaves primarias:

Sea $R (A_1, A_2, A_3, \dots, A_n)$ un esquema¹³ de tabla, donde $A_1, A_2, A_3, \dots, A_n$ representan los n dominios que contiene dicha tabla.

Decimos que un subconjunto X de $(A_1, A_2, A_3, \dots, A_n)$ es llave primaria de R si cumple las propiedades:

- i) cada valor de X identifica de manera única a cada renglón de la tabla R
- ii) no existe X' subconjunto propio de X tal que: X' y X compartan la propiedad i)

Además de las propiedades descritas arriba la llave primaria no puede tener algún atributo con valor nulo, esto es, la llave primaria no puede tener algún valor en cero o blanco. A esta condición se le conoce como **regla de integridad de identidad**, es decir, una tabla debe poseer una llave única que no puede tener valores nulos.

Las **llaves secundarias** son las llaves candidatas que no fueron seleccionadas como llave primaria.

Las **llaves foráneas** se definen como :

Sea $R (A_1, A_2, A_3, \dots, A_n)$ un esquema de tabla, donde $A_1, A_2, A_3, \dots, A_n$ representan los n dominios que contiene dicha tabla.

¹² Una vez elegida la llave primaria no puede ser cambiada por otra, a menos que se modifique la tabla o el objetivo de la misma. Desde luego, una alteración de esta naturaleza puede implicar un gran costo en el mantenimiento del sistema.

¹³ Un esquema representa un renglón particular de una tabla o relación.

Decimos que un subconjunto Y de $\{A_1, A_2, A_3, \dots, A_n\}$ es una llave foránea de R si hace referencia a una llave primaria que puede estar en la misma o en otra tabla.

Un ejemplo puede aclarar esta situación:

Si se tiene la tabla 1 con los atributos :
#orden, fecha-orden, numero-cliente

donde **#orden** es la llave primaria de la tabla 1

Por otro lado se tiene la tabla 2 con los atributos
#orden, número-producto, nombre-producto

donde **#orden** y **número-producto** forman una llave primaria (compuesta) de la tabla 2 y a su vez es una llave foránea de la tabla 1, porque dentro de sus atributos se cuenta **#orden** y por lo tanto puede hacer referencia o acceder los registros de esa tabla.

Una llave foránea debe cumplir con la siguiente regla conocida como **integridad de referencia**:

Todos los valores no nulos de la llave foránea, deben estar contenidos dentro del conjunto de valores de la llave primaria a la que hace referencia.

Una tabla puede tener varias llaves foráneas. Las llaves primarias y foráneas forman las relaciones lógicas entre las tablas de las bases de datos.

3.7. DEPENDENCIA FUNCIONAL

Un concepto importante en el que se basa la normalización es el de la **Dependencia Funcional**.

La Dependencia Funcional involucra a dos columnas de la tabla; a través de esta idea se puede responder a la pregunta: ¿ Si el valor de la primera columna se repite, significa esto que el valor de la segunda columna también se repite?

Por ejemplo en el esquema:

CURSA (MATRICULA, NOMBRE-ALUMNO, FECHA-NACIMIENTO, CURSO,
 PROFESOR)

¿Si el valor de la MATRICULA se repite significa que el valor de la FECHA-NACIMIENTO, también se repite ?

La respuesta es SI. Entonces se dice que FECHA-NACIMIENTO es funcionalmente dependiente de MATRICULA, o bien, que MATRICULA determina funcionalmente a FECHA-NACIMIENTO y se especifica cómo:

MATRICULA \rightarrow FECHA-NACIMIENTO

DEFINICION:

Dada la relación R, el atributo Y de R es funcionalmente dependiente del atributo X de R si y sólo si, siempre que dos renglones de R coincidan en sus valores de X, también coincidan en sus valores de Y; especificándose cómo:

X \rightarrow Y

Lo cual se lee como: X determina funcionalmente a Y, o bien, Y depende funcionalmente de X.

DEFINICION.

Se le llama determinante (funcional) a un atributo (o conjunto de atributos) del cual depende funcionalmente en forma completa algún otro atributo.

Al continuar con el ejemplo del esquema CURSA se ve que :

Si el valor de MATRICULA se repite significa que:

- ¿El valor de NOMBRE-ALUMNO se repite? SI

entonces MATRICULA \rightarrow NOMBRE-ALUMNO

- ¿El valor de CURSO se repite ? NO

entonces MATRICULA \nrightarrow CURSO

- ¿El valor de PROFESOR se repite ? NO

entonces MATRICULA \nrightarrow PROFESOR

3.8. FORMAS NORMALES

Dado un conjunto de atributos que forman el conjunto central de una base de datos, existen muchas maneras en que los atributos pueden ser agrupados para formar el conjunto de tablas o relaciones de la base de datos, dichas formas de agrupación son llamadas **formas normales** (ver fig. 3.3).

La normalización es un proceso que parte de la relación universal y atraviesa por una serie de etapas conocidas como:

- primera forma normal (1NF)**
- segunda forma normal (2NF)**
- tercera forma normal (3NF)**
- forma normal de Boyce & Codd (BCNF)**
- cuarta forma normal (4NF)**
- ...

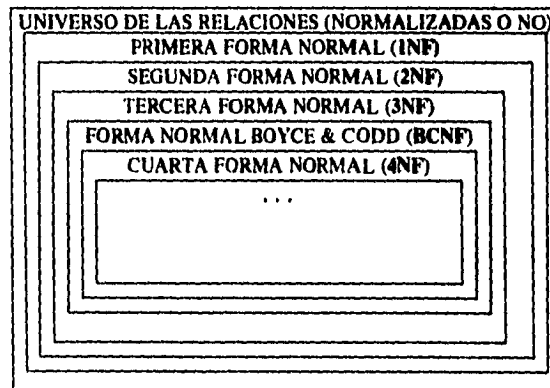


Fig. 3.3. Este cuadro presenta las diferentes contenciones de las formas normales, por ejemplo: si una forma está en n NF, también está en $(n-1)$ NF, $(n-2)$ NF, ..., 1NF.

Como ya se mencionó, para describir gráficamente la 1NF, 2NF, 3NF se utilizará como ejemplo la Figura 3.1, que aun no está normalizada. A través de ella se irá mostrando la evolución de las diferentes etapas de la normalización sin que pierda ninguno de sus atributos.

3.8.1. PRIMERA FORMA NORMAL

La primera forma normal es la menos restrictiva y se define de la siguiente manera :

Una relación está en *primera forma normal* (1NF) si y sólo si, todos sus atributos son atómicos ¹⁴

Obtención de la primera forma normal. Si se ve nuevamente la figura 3.1 y se observa después la figura 3.4, se puede notar la creación de nuevas tablas (tantas como sea necesario) con los campos repetitivos para obtener la primera forma normal.

Para este ejemplo esos campos son: número-producto, nombre-producto, cantidad-ordenada, precio-producto, total-producto y además #orden con los que se conformará otro registro.

DATOS QUE MANEJA LA EMPRESA "LA GRAN VENTA"

#ORDEN	FECHA-ORDEN	NUMERO-CLIENTE	NOMBRE-CLIENTE	DIRECCION-CLIENTE	NUMERO-PRODUCTO	NOMBRE-PRODUCTO	CANTIDAD-ORDENADA	PRECIO-PRODUCTO	TOTAL-PRODUCTO	TOTAL-ORDEN

Fig. 3.1.Registro sin normalizar (contiene varios campos con repetición o multivaluados como son : número-producto,nombre-producto,cantidad-ordenada,etc.)

Es importante señalar que aunque #orden no es un campo múltiple se incluye en el nuevo registro, además de conservarse en el registro origen para conseguir, de esta manera, relacionar ambas tablas a las que se identificará como ORDEN y ORDEN-PRODUCTO, con ello se evita perder información.

¹⁴ Un atributo atómico es aquel que no puede dividirse en otros atributos.

La 1NF no reduce redundancia (al contrario la incrementa), sin embargo, obliga a que las tablas queden en dos dimensiones (tablas planas).

PRIMERA FORMA NORMAL

Se eliminan los atributos no atómicos (campos repetitivos), creando varias nuevas tablas (tantas como sea necesario)

TABLA ORDEN

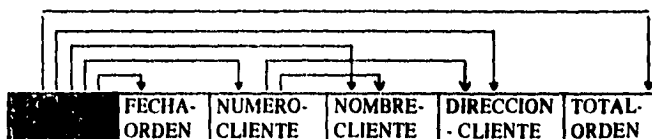


TABLA ORDEN-PRODUCTO

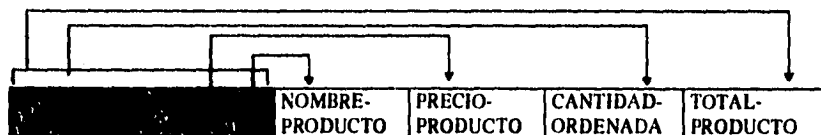


Fig. 3.4 Los cuadros sombreados representan las llaves primarias (ya sean simples o compuestas) de las tablas y las flechas señalan los atributos que determina cada llave. En el ejemplo de la figura, el registro ORDEN tiene como llave primaria #ORDEN y ésta determina a FECHA-ORDEN, NUMERO-CLIENTE, etc. En el registro ORDEN-PRODUCTO se tiene una llave primaria compuesta (#ORDEN y NUMERO-PRODUCTO) que determina a CANTIDAD-ORDENADA y TOTAL-PRODUCTO. Además NUMERO-PRODUCTO, por sí solo, determina a NOMBRE-PRODUCTO y a PRECIO-PRODUCTO.

3.8.2. SEGUNDA FORMA NORMAL

Antes de iniciar la explicación de lo que implica la segunda forma normal se harán las siguientes definiciones:

DEFINICION

Se le dice **determinante** de una relación al atributo (o grupo de atributos) por medio del cual se pueden señalar otros atributos de la relación de manera unívoca.

DEFINICION.

Un atributo es no primo si no forma parte de la llave primaria.

DEFINICION

Una relación está en segunda forma normal (2NF) si y sólo si está en 1NF y cada atributo no primo es funcionalmente dependiente de la llave primaria completa.

Las relaciones se transforman de primera forma normal a segunda forma normal al descomponer a las primeras en un conjunto de relaciones más pequeñas y eliminar los atributos no dependientes de la llave primaria completa, es decir aquellos atributos que no dependen totalmente de la llave. A las dependencias de este tipo se les llama **dependencias funcionales parciales**.

Una manera sencilla de lograr lo anterior se hace creando una nueva tabla para cada determinante en la relación en primera forma normal (1NF).

Los determinantes se convierten en las llaves primarias de las nuevas relaciones.

Los atributos restantes de la relación son todos aquellos atributos que son funcionalmente dependientes de la llave primaria, es decir, del determinante de la relación por el que se creó a la nueva relación.

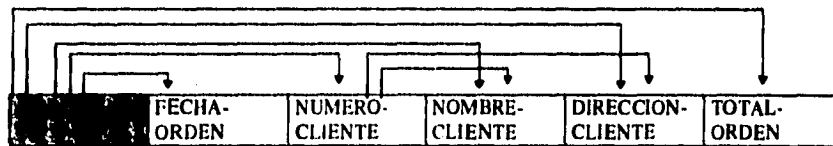
En la Fig. 3.4. se presentó la 1NF a que se llevó la tabla de la empresa "La Gran Venta". Allí se observa que la tabla ORDEN-PRODUCTO tiene atributos que no dependen de la llave completa (sólo de una parte de ella), dichos atributos son

NOMBRE-PRODUCTO y PRECIO-PRODUCTO, que están determinados por NUMERO-PRODUCTO.

Para eliminar la dependencia parcial de esa tabla se parte en dos su información lo que dará como resultado una tabla más, a la que se llamará PRODUCTO. Así en la tabla ORDEN-PRODUCTO tanto CANTIDAD-ORDENADA como TOTAL-PRODUCTO están totalmente determinados por la llave que forman #ORDEN y NUMERO-PRODUCTO, mientras que en la tabla PRODUCTO la llave NUMERO-PRODUCTO determina completamente a NOMBRE-PRODUCTO y a PRECIO-PRODUCTO.

De esta manera todas las tablas quedan en 2NF (ver Fig. 3.5).

ORDEN



ORDEN-PRODUCTO



PRODUCTO

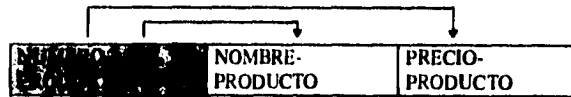


Fig. 3.5. Aquí se representan las tres tablas que quedaron después de haberse aplicado la segunda forma normal, cada una de ellas con su llave de acceso (los rectángulos sombreados).

3.8.3. TERCERA FORMA NORMAL

Las relaciones en 2NF no están exentas de anomalías en **altas, bajas y actualizaciones**, por lo cual no son la meta de una base de datos bien normalizada. Dichas anomalías se presentan cuando la relación en 2NF contiene dependencias transitivas entre sus atributos.

DEFINICION

Se dice que hay **dependencia transitiva** en una relación cuando la llave primaria *A* determina a los atributos *B* y *C*, pero *B* también determina a *C*

DEFINICION

Una relación está en tercera forma normal (3NF) si y sólo si está en 2NF y todo atributo no primo es dependiente no transitivamente de la llave primaria.

DEFINICION

Dos atributos son mutuamente independientes si ninguno es funcionalmente dependiente del otro.

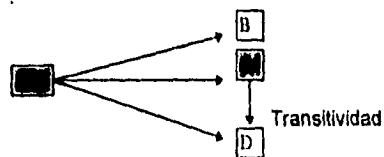
DEFINICION

Una relación está en 3NF si y sólo si está en 2NF y todos sus atributos no primos son mutuamente independientes.

A continuación se señala de manera esquemática las condiciones que se presentan cuando una tabla no está en tercera forma normal y las acciones que se deben ejecutar para obtenerla.

En la figura se observa como se da la transitividad ya que el atributo A determina a los atributos B,C y D pero además C también determina a D. Por lo tanto el esquema no está en 3NF.

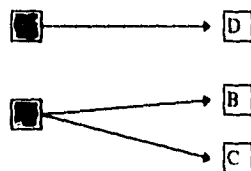
No está en 3NF :



Para eliminar la transitividad se separan los atributos que la ocasionan.

Al realizar dicha acción, C determina a D y A determina a B y a C, con lo cual se elimina la transitividad y se consigue llegar a la tercera forma normal. La siguiente figura ilustra dicha acción :

Está en 3NF :



Las anomalías en relaciones en 2NF pueden ser eliminadas dividiendo las relaciones en relaciones que no posean dependencias transitivas. Al dividir las dependencias transitivas en dos dependencias funcionales, cada determinante se convierte en la llave primaria de una nueva relación para llegar, de esta manera, a la tercera forma normal.

Además de las inconsistencias ya señaladas, existen relaciones que estando en 2NF, contienen redundancia que puede eliminarse.

Se hará un breve paréntesis con otro ejemplo para abordar lo que será la tercera forma normal, antes de continuar con el ejemplo de "La Gran Venta" el cual se ha venido trabajando para indicar los pasos de las formas normales.

Para el siguiente ejemplo se utilizará una tabla con transitividad entre sus atributos.

Ejemplo de una tabla que no está en 3NF:

TABLA INICIAL A LA QUE SE LLAMARA AUTOS

PLACAS	MODELO	COLOR	MARCA	PAIS
521-GBS	1988	BLANCO	CHEVROLET	E.U.A.
221-BBN	1990	AZUL	V.W.	ALEMANIA
857-FGN	1982	ROJO	FORD	E.U.A.
332-RST	1993	BLANCO	V.W.	ALEMANIA
451-LTA	1991	NEGRO	NISSAN	JAPON
874-HYS	1990	VERDE	NISSAN	JAPON
993-GFA	1992	GRIS	FORD	E.U.A.
378-AAT	1980	AZUL	V.W.	ALEMANIA
661-UAM	1988	CAFE	RENAULT	FRANCIA
953-JSO	1986	NEGRO	CHRYSLER	E.U.A.

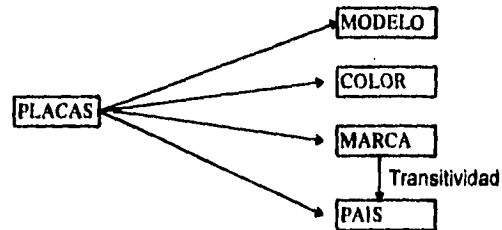
Tabla con los atributos Placas , Modelo , Color, Marca y Pais donde, si se analiza con cuidado, MARCA y PAIS son campos que presentan transitividad, ya que al conocer la marca del automóvil también se conoce el nombre del país origen. Por lo tanto, es innecesario almacenar ambos atributos cada vez que se da de alta un nuevo juego de placas.

Para eliminar la transitividad en la tabla y llevarla a 3NF se divide en dos la información: una parte con los atributos PLACAS, MODELO, COLOR y MARCA, la otra parte con MARCA y PAIS.

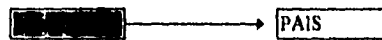
Si se ve el ejemplo bajo esquema se tiene:

AUTOS (PLACAS,MODELO,COLOR,MARCA,PAIS)

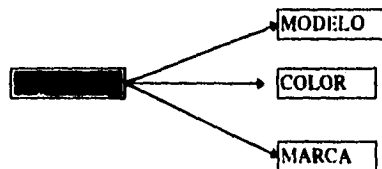
Donde PLACAS determina a los atributos MODELO, COLOR, MARCA y PAIS. Pero MARCA también determina a PAIS y aquí es donde se presenta la dependencia transitiva.



Para solucionar el problema se divide el esquema original en dos, uno con el atributo MARCA que determina a PAIS y el otro con el atributo PLACAS que determina a los atributos MODELO, COLOR y MARCA. Así se elimina la dependencia transitiva y se generan dos tablas : una, a la que se llamará FABRICANTES, con MARCA como llave y la otra, que conservará el nombre de AUTOS, con PLACAS como su llave principal.



MARCA determina a PAIS



PLACAS determina a MODELO, COLOR y MARCA

De esta forma, del esquema original AUTOS se generan los esquemas : FABRICANTES y AUTOS (ya sin relaciones transitivas).

FABRICANTES (MARCA, PAIS)

AUTOS (PLACAS, MODELO, COLOR, MARCA)

Al final quedan las siguientes tablas:

TABLA FABRICANTES

MARCA	PAIS
CHEVROLET	E.U.A.
CHRYSLER	E.U.A.
FORD	E.U.A.
NISSAN	JAPON
RENAULT	FRANCIA
V.W.	ALEMANIA

Si se conoce la marca del automóvil se conoce al país origen. Así MARCA determina a PAIS.

TABLA AUTOS

PLACA	MODELO	COLOR	MARCA
521-GBS	1988	BLANCO	CHEVROLET
221-BBN	1990	AZUL	V.W.
857-FGN	1982	ROJO	FORD
332-RST	1993	BLANCO	V.W.
451-LTA	1991	NEGRO	NISSAN
874-HYS	1990	VERDE	NISSAN
993-GFA	1992	GRIS	FORD
378-AAT	1980	AZUL	V.W.
661-UAM	1988	CAFE	RENAULT
953-JSO	1986	NEGRO	CHRYSLER

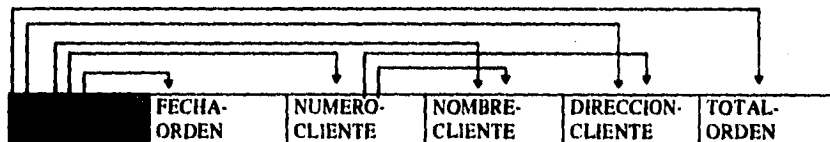
Una vez que se conocen las placas de un automóvil se puede conocer modelo, color y marca. Por tanto, PLACAS determina a los atributos MODELO, COLOR y MARCA.

De esta forma, al partir la información de la tabla AUTOS, se eliminó la transitividad y de una se generaron dos tablas: AUTOS (recortada) y FABRICANTES en tercera forma normal.

Se continuará con el ejemplo de "La Gran Venta", una vez que se ha comentado e ilustrado el procedimiento para llegar a la tercera forma normal.

De acuerdo a la definición para la tercera forma normal, se tiene entonces que la tabla ORDEN de la Fig. 3.5 aun tiene deficiencias : presenta transitividad ya que #ORDEN determina a FECHA-ORDEN,NUMERO-CLIENTE,NOMBRE-CLIENTE,DIRECCION-CLIENTE y TOTAL-ORDEN, pero a su vez NUMERO-CLIENTE determina también a NOMBRE-CLIENTE y a DIRECCION-CLIENTE.

TABLA ORDEN, TOMADA DE LA FIGURA 3.5



Para eliminar la transitividad de la tabla ORDEN, se parte en dos. Una tabla, que este caso seguirá llamándose ORDEN, con la llave #ORDEN que determina a FECHA-ORDEN,NUMERO-CLIENTE y TOTAL-ORDEN y otra tabla, a la que se dará el nombre de CLIENTE, con la llave NUMERO-CLIENTE que determina a NOMBRE-CLIENTE y a DIRECCION-CLIENTE.

A las tablas ORDEN-PRODUCTO y PRODUCTO no se les hace nada porque de acuerdo a la definición de 3NF ya se encuentran en ella.

La Fig. 3.6 presenta las tablas ORDEN, CLIENTE, ORDEN-PRODUCTO y PRODUCTO después de haberse aplicado los pasos para la normalización en 1NF, 2NF y 3NF.

TABLA ORDEN

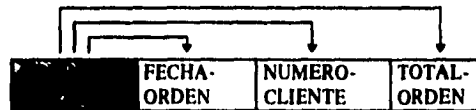


TABLA CLIENTE



TABLA ORDEN-PRODUCTO



TABLA PRODUCTO



Fig. IV.6. Después de haber sido aplicados los pasos de 1NF, 2NF y 3NF a la tabla original de la empresa "La Gran Venta" se generaron cuatro tablas cada una con su llave (la parte sombreada) y señalando (con las flechas) a los atributos que determinan.

Una vez que se ha conseguido llevar las tablas a tercera normal, el modelo está listo para continuar con el proyecto de implantación del sistema de base de datos.

3.8.4. FORMA NORMAL BOYCE AND CODD Y CUARTA FORMA NORMAL

A continuación se explicará en que consisten la forma normal conocida como Boyce and Codd (BCNF) y la cuarta forma normal (4NF) y se dará un ejemplo¹⁵ de los casos en que se utilizan cada una de ellas.

Forma normal BCNF. Una vez que Codd dio las bases para el desarrollo del modelo de datos relacional, los investigadores se dieron cuenta que la definición original de Codd para la tercera forma normal presentaba ciertas deficiencias. En términos más precisos, no manejaba de manera satisfactoria el caso de una relación en la cual:

- a) hay varias llaves candidatas,
- b) esas llaves candidatas son compuestas y
- c) las llaves candidatas se traslapan (es decir tienen por lo menos un atributo en común)

En otras palabras, existen relaciones en 3NF que todavía presentan redundancia y que puede eliminarse, la BCNF prevé casos como estos.

Antes de mencionar en qué consiste la forma normal de Boyce and Codd, se harán las siguientes definiciones:

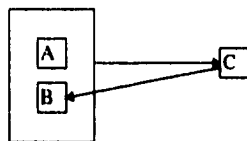
DEFINICION

Una relación está en BCNF si y sólo si cada determinante¹⁶ es una llave candidata.

Esta definición se representa como sigue:

Los atributos **A** y **B** determinan al atributo **C** y **C** a su vez determina a **B**

No está en BCNF

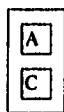


¹⁵ Los ejemplos fueron tomados del texto de J. Carlos Talavera, BASES DE DATOS, Conceptos, Diseño e Implantación. Centro de Cómputo, División Académica de Computación del Instituto Tecnológico Autónomo de México (ITAM).

¹⁶ Recuérdese que se le dice determinante de una relación al atributo (o grupo de atributos) por medio del cual se pueden señalar otros atributos de la relación de manera unívoca.

En casos como estos en que los atributos **A**, **B**, **C** se determinan mutuamente, **C** es un determinante pero no una llave candidata porque determina a **B** pero no determina a **A**.

Para solucionar situaciones de este tipo, se divide la información en dos: una parte en la que **C** determina a **B** y la otra en la que **A** y **B** son atributos únicos de la tabla y además ambos atributos forman parte de la misma llave primaria.



De esta forma, se obtienen las tablas T1 y T2:

Tablas en BCNF

T1 (C, B)

T2 (A, C)

Para este tipo de normalización es que se habla de llaves candidatas y no de llaves primarias. La motivación de la BCNF estriba en que la definición original de 3NF no maneja satisfactoriamente el caso de una relación que posea dos o más llaves candidatas compuestas y traslapadas.

Ejemplo de una relación que bajo las condiciones que se describen abajo no está en BCNF:

Tabla *GRUPOS*

ALUMNO	MATERIA	PROFESOR
10	300	LUIS
20	300	LUIS
30	300	LUIS
60	300	JOSE
70	300	JOSE
20	600	ANA
10	600	ANA
10	800	ORALIA

De la tabla *GRUPOS* se tiene el siguiente esquema:

GRUPOS (ALUMNO, MATERIA, PROFESOR)

Si en este ejemplo se cumple lo siguiente:

- Cada materia la enseñan varios profesores,
- cada profesor enseña sólo una materia,
- cada materia a cada estudiante de la misma es impartida por un mismo profesor,
- un estudiante cursa varias materias.

Entonces ALUMNO, MATERIA y PROFESOR pueden llegar a ser parte de las llaves candidatas de la relación, es decir las posibles llaves candidatas son:

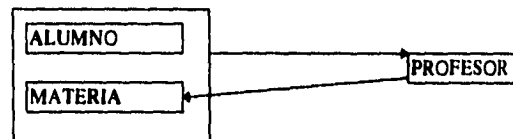
-- ALUMNO,MATERIA

-- ALUMNO,PROFESOR

-- PROFESOR

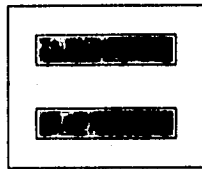
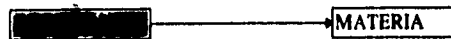
Pero en este caso, **PROFESOR** no cumple cabalmente la condición de una llave candidata, porque determina a **MATERIA** pero no determina a **ALUMNO**.

De la situación mencionada se presenta lo siguiente :



ALUMNO y MATERIA determinan a PROFESOR y a su vez PROFESOR determina a MATERIA, pero no determina a ALUMNO.

Para eliminar la redundancia se divide en dos partes la información: una en la que PROFESOR determina a MATERIA y la otra con ALUMNO y PROFESOR como atributos únicos y además formando parte de la misma llave.



Con esto se tiene dos nuevas tablas una en la que PROFESOR determina a MATERIA y en la otra tabla ALUMNO, PROFESOR como atributos y llave primaria (compuesta) a la vez.

G1(PROFESOR,MATERIA)

G2(ALUMNO,PROFESOR)

De la tabla original resultan dos tablas

PROFESOR	MATERIA
LUIS	300
JOSE	300
ANA	600
ORALIA	800

ALUMNO	PROFESOR
10	LUIS
20	LUIS
30	LUIS
60	JOSE
70	JOSE
20	ANA
10	ANA
10	ORALIA

Cuarta forma normal. Un breve ejemplo ilustrará la cuarta forma normal, pero antes se harán las siguientes definiciones¹⁷:

DEFINICION

Dada una relación R con atributos A, B y C la dependencia multivaluada (DMV)

$$A \twoheadrightarrow B$$

se cumple en R si y sólo si el conjunto de valores B que corresponden a un par (valor de A, valor de C) dado en R depende tan solo del valor de A y es independiente del valor de C. Como siempre A, B y C pueden ser compuestos.

La notación $A \twoheadrightarrow B$ se lee como : A multidetermina a B. Nótese que las DMV's, como se han definido, pueden existir sólo si la relación R tiene al menos tres atributos.

Para una relación R (A, B, C) , se observa que la DMV $A \twoheadrightarrow B$ se cumple si y sólo si también se cumple la DMV $A \twoheadrightarrow C$.

Las DMV'S siempre se presentan juntas de esta manera. Razón por la cual, es común expresarlas en una sola proposición, usando la notación:

$$A \twoheadrightarrow B \ C$$

Por ejemplo:

$$\text{CURSO} \twoheadrightarrow \text{PROFESOR} \ \text{TEXTO}$$

DEFINICION

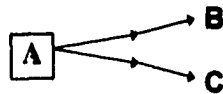
Una relación está en cuarta forma normal (4NF) si y sólo si esta en BCNF y no contiene dependencias multivaluadas:

La relación R (A, B, C) donde se cumplen las dependencias multivaluadas:

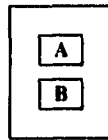
$$A \twoheadrightarrow B \ C$$

¹⁷ Estas definiciones fueron tomadas del folleto de Bases de Datos. Conceptos, Diseño e Implantación de J. Carlos Talavera, Centro de Computo; División Académica de Computación, Instituto Tecnológico Autónomo de México (ITAM), 1993.

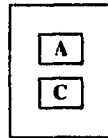
Se puede descomponer en sus dos proyecciones $R1(A,B)$ y $R2(A,C)$ sin pérdida de información :



No está en 4NF



Tablas en 4NF



El siguiente ejemplo presenta los casos en los que se utiliza la cuarta forma normal e indica el problema que se quiere resolver con ella.

Vamos a suponer que se da una relación no normalizada que tiene información sobre los cursos, profesores y textos.

Los grupos de FÍSICA son impartidos por REBECA, LUIS, JOSE y ADRIANA respectivamente y no importando de que profesor se trate, los textos para este curso son MECANICA BASICA y PRINCIPIOS DE OPTICA. Similarmente para MATEMATICAS y los otros cursos. Además un mismo texto (supóngase ALGEBRA LINEAL) puede ser utilizado en varios cursos distintos (como MATEMATICAS y ALGEBRA II) los cuales son impartidos por un mismo profesor (por ejemplo LAURA).

En la tabla se presenta la manera como están distribuidos los profesores en los grupos de FÍSICA, de MATEMÁTICAS y de ALGEBRA II.

CURSO	PROFESOR	TEXTO
FISICA	REBECA	MECANICA BASICA
	LUIS	PRINCIPIOS DE OPTICA
	JOSE ADRIANA	
MATEMATICAS	JOSE LAURA	ALGEBRA LINEAL GEOMETRIA ANALITICA
ALGEBRA II	LAURA MANUEL	ALGEBRA LINEAL

Presentación por renglón de curso, profesor y texto. En esta tabla, fácilmente se puede observar la redundancia que existe.

CURSO	PROFESOR	TEXTO
FISICA	REBECA	MECANICA BASICA
FISICA	REBECA	PRINCIPIOS DE OPTICA
FISICA	LUIS	MECANICA BASICA
FISICA	LUIS	PRINCIPIOS DE OPTICA
FISICA	JOSE	MECANICA BASICA
FISICA	JOSE	PRINCIPIOS DE OPTICA
FISICA	ADRIANA	MECANICA BASICA
FISICA	ADRIANA	PRINCIPIOS DE OPTICA
MATEMATICAS	JOSE	ALGEBRA LINEAL
MATEMATICAS	JOSE	GEOMETRIA ANALITICA
MATEMATICAS	LAURA	ALGEBRA LINEAL
MATEMATICAS	LAURA	GEOMETRIA ANALITICA
ALGEBRA II	LAURA	ALGEBRA LINEAL
ALGEBRA II	MANUEL	ALGEBRA LINEAL

Aplicando las formas normales anteriores (1NF, 2NF, 3NF, BCNF) no se llega a ninguna reducción. Sin embargo, si dividimos la información en dos tablas se reduce redundancia y se obtiene la cuarta forma normal (ver las tablas CURSO-PROFESOR y CURSO-TEXTO).

TABLA CURSO-PROFESOR

CURSO	PROFESOR
FISICA	REBECA
FISICA	LUIS
FISICA	JOSE
FISICA	ADRIANA
MATEMATICAS	JOSE
MATEMATICAS	LAURA

TABLA CURSO-TEXTO

CURSO	TEXTO
FISICA	MECANICA BASICA
FISICA	PRINCIPIOS DE OPTICA
MATEMATICAS	ALGEBRA LINEAL
MATEMATICAS	GEOMETRIA ANALITICA

De esta manera, la cuarta forma normal elimina redundancia y resuelve el problema que provocan las dependencias multivaluadas que se presentan en las tablas.

Con esto se concluye el capítulo del diseño lógico de los sistemas de bases de datos relacionales y se da paso al siguiente capítulo donde se desarrollará el tema del diseño lógico de los sistemas de bases de datos orientados a objetos. Sistemas a los cuales se han dirigido la mayor parte de las investigaciones en la década de los noventa.

CAPITULO 4

DEFINICION Y CONCEPTOS FUNDAMENTALES DE LOS MODELOS Y LAS BASES DE DATOS ORIENTADAS A OBJETOS

La idea fundamental de los sistemas orientados a objetos (OO) es que los usuarios no deberían tener que esforzarse con construcciones orientadas a la computadora tales como registros y campos, sino más bien deberían poder manejar objetos (y operaciones) que se asemejen más a sus equivalentes en el mundo real.

De esta manera, se presume que en la computadora se representen las entidades del mundo real como objetos que tienen atributos y participación en ciertos tipos de relaciones y no como los registros de los sistemas tradicionales orientados a los archivos.

Los modelos de datos orientados a objetos también son llamados **semánticos** porque describen (en lenguaje de máquina) las cosas de acuerdo al significado que se les da en la vida real. Esto es, capturan el **significado** de los datos y las relaciones que guardan en el modelo.

En particular, el propósito de un programa de aplicación es (por definición) resolver algún problema específico, en tanto que el de una base de datos es (por definición) resolver diversos problemas, algunos de los cuales quizá ni estaban previstos en el momento de establecerse la base de datos. Así, dotar de muchas características a los objetos de un lenguaje de programación es una buena idea: reduce la cantidad de código de manipulación que se necesita escribir, mejora la productividad de los programadores, facilita el mantenimiento de las aplicaciones, etc. En cambio dotar de muchas características semánticas a los objetos de una base de datos podría ser una buena o una mala idea: podría simplificar algunos problemas, pero al mismo tiempo podría dificultar otros o aun hacerlos imposibles de resolver.

El diseño del software se desliga de los detalles de representación de los objetos de datos que se usan en el sistema. Así, se pueden cambiar muchas veces esos detalles de representación, sin que ello afecte al sistema de software global.

Es prematuro decir que el diseño orientado a los objetos es un método maduro. Sin embargo, a medida que sigue creciendo la popularidad del enfoque orientado a los objetos, se impone la utilización de métodos de diseño para crear sistemas orientados a los objetos.

El enfoque que se denomina orientado a los objetos ha evolucionado a lo largo de los últimos 20 años. Los primeros trabajos sentaron las bases al establecer la

importancia de la abstracción, del ocultamiento de información y de la modularidad para la calidad del software.

Muchos investigadores creen que con la llegada del nuevo milenio, los métodos y los lenguajes de programación orientados a los objetos serán los que predominen y proponen a los sistemas orientados a objetos como el futuro en la tecnología de las bases de datos.

4.1. ANTECEDENTES DE LAS BASES DE DATOS ORIENTADAS A OBJETOS

Las bases de datos orientadas a objetos son el resultado de la convergencia de dos disciplinas de investigación: *El modelo de datos semántico y los lenguajes orientados a objetos*.

Estas disciplinas se desarrollaron independientemente pero en años recientes han empezado a mezclarse con importantes implicaciones para el procesamiento de las bases de datos.

El desarrollo original del modelo semántico de datos fue enunciado por Hull y King en 1987, con el propósito de incrementar la efectividad y la exactitud del diseño de la base de datos.

Los métodos del modelo semántico resultaron ser apropiados para manejar varios problemas de los usuarios y podían ser fácilmente convertidos a registros basados en la implementación de modelos tales como bases de datos jerárquicas, de redes y relacionales.

Mientras que el modelo de datos semántico estuvo interesado primeramente en las estructuras de los datos, los desarrolladores de los lenguajes de programación orientados a objetos estuvieron más interesados en la forma en que los lenguajes manipulaban los datos (consultas, cálculos, actualizaciones). Por lo que la estructura de los datos fue una preocupación secundaria para ellos.

La convergencia de estas dos áreas vino cuando los investigadores empezaron a aplicar los conceptos de los lenguajes orientados a objetos a las estructuras semánticas de datos.

El resultado es la noción de una *base de datos orientada a objetos*. En esta fusión de disciplinas, la terminología orientada a objetos ha tendido a predominar, es por eso que se habla de objetos más que de entidades, como si estuviéramos usando la terminología semántica.

4.2. NECESIDADES EN EL DISEÑO DE LA BASE DE DATOS ORIENTADA A OBJETOS

Para diseñar una base de datos es necesario dejar de pensar en la estructura de los sistemas de archivos. En otras palabras, se necesita cambiar de un enfoque que esté orientado físicamente a uno que esté orientado lógicamente. Esto se hace con un enfoque orientado a objetos.

En el proceso de definición de requerimientos, el diseño lógico necesita identificar los datos que manejan los usuarios y representarlos en un modelo bien definido. Para realizar esto, es necesario ver cuidadosamente la naturaleza de dichos datos y precisar el significado lógico de ellos para representarlos.

Durante esta etapa, los detalles de las vistas individuales de los usuarios son creados e integrados dentro de un modelo de datos orientado a objetos, registrando todos los elementos incorporados a los datos para ser mantenidos en la base de datos.

Hay varias formas en que un objeto puede ser establecido. La más común es el uso de un sistema generador de identificadores único. En este caso, la identidad de un objeto no dependerá de su posición ni de sus propiedades, porque éstas pueden cambiar una a una pero permanecerá el mismo objeto en todos aspectos. Un buen ejemplo es un río que, en el tiempo geológico, puede cambiar su fuente, ruta y contenido, pero aún sigue siendo el mismo río.

DEFINICION

"Un sistema de base de datos orientada a objetos (o modelo de datos) es un sistema de base de datos (o modelo) - en el sentido de que tiene todas las características de una base de datos e incluye lenguajes de acceso, capacidad para manejar grandes cantidades de datos, persistencia, integridad de transacciones y datos, concurrencia, seguridad y recuperación- que adicionalmente soporta abstracción y herencia en los objetos"¹⁸.

4.3. DIFERENTES NIVELES DEL MODELO ORIENTADO A OBJETOS

Al igual que en el modelo de base de datos relacional, en el modelo de base de datos orientado a objetos también se tienen tres diferentes niveles. Estos niveles son ilustrados en la Figura 4.1.¹⁹

¹⁸ GRAHAM, Ian, Object Oriented Methods, 1er. ed., Great Britain, Addison Wesley, 1992 (reimpresión).

¹⁹ Fig. tomada de HANSEN, Gary W., - HANSEN, James V. Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 80

FIGURA 4.1. LOS TRES NIVELES DE LOS MODELOS

NIVEL DE ABSTRACCION	ETAPA DEL MODELO	MODELO	CONSTRUCCION TIPICA
ALTO (DE VISION O EXTERNO)	DISEÑO DE LA METODOLOGIA	ORIENTADO A OBJETOS, RELACIONAL, ETC.	OBJETOS, RELACIONES, TABLAS, COLUMNAS
		UN CONJUNTO DE MODELOS DAN COMO RESULTADO UN ESQUEMA DE BASE DE DATOS	
MEDIO (CONCEPTUAL)	ESQUEMA DE LA BASE DE DATOS	ESQUEMA DE LA BASE DE DATOS	PERSONAS, NOMBRES, DIRECCIONES, ES EMPLEADO POR
BAJO (FISICO)	REALIDAD ACTUAL	EL CONJUNTO DE ESQUEMAS GENERAN EL MODELO PARA LA BASE DE DATOS FISICA	MARGARITA SANCHEZ, AV. PUENTE DE ALVARADO No. 845
		BASE DE DATOS	

El nivel más bajo corresponde al diseño físico de la base de datos, y se distingue porque en él se registran los hechos seleccionados acerca de la realidad que actualmente son verdaderos.

Por ejemplo, la base de datos puede registrar el hecho: "Margarita Sánchez vive en el 845 de la calle Puente de Alvarado ". Si la dirección de Margarita cambia, entonces el estado de la base de datos debe cambiar, para que la base continúe siendo un modelo exacto o preciso de la realidad.

En otra palabras, se dice que "el estado actual de una base de datos en particular es un modelo de la realidad, porque es un registro de hechos seleccionados acerca de la realidad y que son actualmente verdaderos"²⁰.

En el siguiente nivel (el nivel medio), el esquema describe la estructura de la base de datos con un inmenso rango de objetos, en los que se definen las características que tienen en común.

Como ejemplo, se puede mencionar que los objetos que "guardan" *nombre* y *dirección* en una base de datos particular, son registrados en el esquema como características que pueden cambiar con el tiempo y que son aplicables a diferentes personas.

²⁰ HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 80.

El nivel más alto corresponde al nivel de visión o externo. En este nivel, bajo la metodología de diseño de la base de datos, se describe la construcción y las reglas que pueden ser usadas en la formulación de un esquema o esquemas posibles para la base de datos.

En resumen, se habla del modelo orientado a objetos, como una metodología para crear esquemas de las bases de datos para una aplicación particular. Estos esquemas de bases de datos son, por sí mismos, modelos que dan la estructura lógica para capturar la información acerca de una porción particular de la realidad. Cuando esta información es capturada y registrada en un sistema de bases de datos computarizado, entonces la base de datos por sí misma es un modelo del estado actual de la realidad.

4.4. MODELOS Y VISTAS PARTICULARES

Lo mismo que en el modelo relacional, los usuarios del modelo orientado a objetos pueden tener su propia visión de la realidad y, en consecuencia, los datos son presentados de acuerdo a las necesidades de cada usuario. Por tal motivo, es pertinente regresar a hacer algunas consideraciones expuestas en el capítulo dedicado a las bases de datos relacionales.

Un modelo retiene únicamente los detalles seleccionados de la realidad. Por ejemplo, si se considera una transacción contable, tal como el depósito a una cuenta de cheques, Contabilidad quiere mantener ciertos detalles (número de cuenta, cantidad depositada, hora, fecha, número de transacción) e ignorar otras (palabras intercambiadas durante la transacción, número de gente en el banco, número de gente esperando en línea, música ambiental tocada por el sistema de sonido, condiciones climatológicas, etc.). La realidad envuelve una gran cantidad de detalles, pero Contabilidad considerará muchos de ellos irrelevantes para la transacción. Así, un modelo de contabilidad es una vista de la transacción que mantendrá únicamente aquellos detalles que se piensan relevantes para Contabilidad.

Por supuesto, alguno de los detalles considerados como irrelevantes por un usuario puede ser muy importante para otros usuarios. Por ejemplo, si se está desarrollando un sistema de base de datos para un restaurante de comida rápida, las condiciones del clima pueden ser un factor importante para la realidad del gerente de ese restaurante, ya que un día frío puede producir ventas muy diferentes a las que se tendrían si el día estuviera templado. Como resultado el gerente podría querer seguir los cambios climatológicos para tomar las medidas adecuadas al tiempo.

El número de gente esperando en la línea puede ser otro aspecto importante de la realidad de un gerente, ya que puede necesitar esta información para programar el número de trabajadores necesario para atender adecuadamente a los clientes y evitar que éstos pierdan mucho tiempo haciendo fila.

De lo dicho en los párrafos anteriores, se desprende que *los usuarios de los sistemas tendrán diferentes modelos o visiones de la realidad, conforme a sus necesidades.*

Como se mencionó antes, una base de datos representa un modelo de la realidad. Los sistemas manejadores de las bases de datos manejan la base de datos para que cada usuario pueda registrar, acceder y manipular los datos de la base de acuerdo a su modelo de la realidad. Para manipular los datos en una gran variedad de formas, los usuarios pueden obtener la información necesaria para tener una resolución exitosa. De este modo, *los modelos son herramientas poderosas para eliminar los detalles irrelevantes y entender la realidad individual de los usuarios.*

Modelar la realidad es, en muchas formas, como solucionar el problema de una historia. Ambas requieren que los detalles sean examinados minuciosamente para crear un modelo "correcto" de esa parte de la realidad. Esto significa que se deben asociar los elementos de la realidad a los elementos en el modelo. Si la asociación es hecha apropiadamente entonces el modelo puede ser usado para solucionar el problema. En caso contrario, dicho modelo no puede producir la solución correcta.

4.5. CONCEPTOS BASICOS

Aquí se presentan algunos términos y conceptos básicos del enfoque orientado a objetos (OO). Es decir, lo que se entenderá por objeto, clase, método (función u operación), mensaje y jerarquía de clase.

Para empezar, en la siguiente tabla²¹ se hace una correspondencia entre los términos de OO y los términos de programación tradicional.

²¹ Tabla tomada de: DATE, C. J. : Introducción a los sistemas de bases de datos, Vol. I, trad. Jaime Malpica; Américo Vargas Villazón, 5 ed., Wilmington, Delaware, E.U.A.; Addison Wesley Iberoamericana, S.A., 1993, pág. 676 .

TERMINO OO	TERMINO EN PROGRAMACION
Objeto	Variable
Clase	Tipo
Método	Función
Mensaje	Llamada
Jerarquía de clases	Jerarquía de tipos

Un objeto en la terminología OO corresponde (más o menos) a una variable en el sentido que tiene en programación.

El término de clase corresponde aproximadamente a la idea tradicional de tipo de datos; o quizá tipo abstracto de datos sería más preciso, porque lo importante es que:

- a) Los usuarios pueden definir sus propias clases.
- b) Todas las clases definidas o integradas por el usuario, llevan consigo un "conocimiento" de los operadores (o métodos) aplicables a los objetos de esa clase. De hecho, la única forma de operar sobre un objeto es mediante los operadores (métodos) definidos para la clase de ese objeto.

Un método es en esencia una operación o función que puede realizarse con los objetos de alguna clase específica. Se puede considerar que el conjunto de métodos aplicables a una clase dada se almacena junto con la definición de esa clase; de esta manera, se define la "interfaz pública" de los objetos de esa clase (en cambio, la representación interna de un objeto se considera "privada" con respecto al objeto en cuestión). Los objetos pueden actualizarse o examinarse sólo a través de su interfaz pública.

Para aplicar un método dado a un objeto determinado es necesario enviar un mensaje a ese objeto. Al recibir el mensaje, el objeto ejecuta la función (el método) solicitada por el mensaje y enseguida devuelve un resultado al remitente.

Se utilizará el siguiente ejemplo para explicar la jerarquía de clases. Si la clase B es una subclase de A, todo caso de B será en forma automática un caso de A, pero lo opuesto no se cumple. En otras palabras, si se tiene una clase llamada EMPLEADO y otra clase llamada ANALISTA entonces ésta es una subclase de EMPLEADO porque todo analista es un empleado, pero lo contrario no sucede, ya que es obvio que existen empleados que no son analistas. De aquí el concepto de jerarquía de clases (más adelante, en este capítulo, se hará una descripción más detallada del concepto de jerarquía de clase).

4.5.1. OBJETO Y ENCAPSULACION

4.5.1.1. OBJETO

El modelo orientado a objetos se basa en encapsular código y datos en una única unidad, llamada objeto. La interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes.

"En general, un objeto tiene asociado:

Un conjunto de *variables* que contienen los datos del objeto. El valor de cada variable es un objeto.

Un conjunto de mensajes a los que el objeto responde.

Un *método* (función), que es un trozo de código para implementar cada mensaje. Un método devuelve un valor como *respuesta al mensaje*.

El término *mensaje* en un contexto orientado a objetos no implica el uso de un mensaje físico en una red de computadoras, sino que se refiere al paso de solicitudes entre objetos sin tener en cuenta detalles específicos de implementación²².

Un objeto es una componente del mundo real que se ha hecho corresponder con funciones y datos utilizados en computación. En el contexto de un sistema basado en computadora, un objeto típicamente es un productor o un consumidor de información, o un elemento de información. Por ejemplo, objetos típicos pueden ser máquinas, órdenes, archivos, monitores, interruptores, señales, cadenas alfanuméricas o cualquier otro lugar, persona, cosa, ocurrencia, papel (de comportamiento) o suceso.

Hacer corresponder un objeto con el código de computadora que lo representa, implica elaborar una estructura de datos privada y crear procesos que se denominan operaciones o métodos que permitan controlar a dicho objeto.

Las operaciones contienen construcciones procedurales y de control que pueden ser invocadas mediante un *mensaje* -una petición al objeto para que realice alguno de sus métodos.

²² KORTH, Henry F. - SILBERSCHATZ, Abraham, Fundamentos de Bases de Datos, trad. Ma. Angeles Vaquero Martín; Antonio Vaquero García, 2 ed., Madrid, Mc Graw-Hill / Interamericana de España, 1993, pág. 459.

El objeto también tiene una parte compartida por la estructura de datos y los mensajes, que es su interfaz. Los mensajes se mueven por la interfaz y especifican la operación que se desea realizar sobre el objeto, aunque no cómo se ha de realizar dicha operación. Es el objeto que recibe el mensaje el que determina, mediante código escrito para ese propósito, cómo se implementa la operación requerida.

Al definir un objeto con parte privada y proporcionar mensajes para invocar el procesamiento adecuado, se consigue el ocultamiento de información, es decir, dejar ocultos para el resto de los elementos del programa los detalles de cómo está creado dicho objeto.

Los objetos con sus operaciones proporcionan una modularidad inherente. En otras palabras, los elementos del software (datos y procesos) están agrupados con un mecanismo de interfaz bien definido (en este caso, los mensajes).

Los objetos pueden almacenar relaciones representadas como ligas de otros objetos y métodos. Estas relaciones son objetos por su propio derecho y, por tanto, pueden tener atributos y métodos.

Los objetos representan elementos que son importantes para los usuarios en la porción de la realidad que queremos modelar. Ejemplos de objetos son la gente, los automóviles, los árboles, las lavadoras de platos, las casas, los martillos y libros. *Estos son objetos concretos. Los objetos conceptuales* serían las compañías, las habilidades, las organizaciones, los diseños de productos, las transacciones de negocios y las clasificaciones de trabajo.

4.5.1.2. OBJETO COMPLEJO

"Un objeto complejo es un dato que es visto como un simple objeto en el mundo real, pero que contiene otros objetos. Estos objetos pueden tener una estructura interna compleja arbitraria. A menudo los objetos están estructurados jerárquicamente, representando la relación entre ellos. El modelo de objetos complejos ha llevado al desarrollo de las bases de datos orientadas a objetos, las cuales están basadas en los conceptos de los lenguajes de programación orientados a objetos y a las bases de datos relacionales anidadas, en las que las relaciones pueden almacenarse dentro de otras relaciones".²³

Conceptualmente, un objeto en una base de datos orientada a objetos almacena tanto atributos, como métodos juntos. En una implementación actual, los objetos complejos y el código de los métodos pueden no estar físicamente almacenados.

²³ KORTH, Henry F. - SILBERSCHATZ, Abraham, Fundamentos de Bases de Datos, trad. Ma. Angeles Vaquero Martín; Antonio Vaquero García, 2 ed., Madrid, Mc Graw-Hill / Interamericana de España, 1993, pág. 457.

Es importante que cualquier lenguaje de consulta contemple esta cuestión de almacenamiento físico muy transparentemente, a fin de que el usuario o el diseñador de la aplicación de la base de datos pueda proceder como si el almacenamiento físico estuviera acorde con el modelo conceptual.

4.5.1.3. ENCAPSULACION

Cuando se habla de encapsulación en el modelo de datos orientado a objetos, se refiere al proceso de incluir los elementos que formarán parte de un objeto particular. En otras palabras, la *encapsulación* significa que existe información que está empaquetada bajo un sólo nombre y puede ser reutilizada como especificación o como componente de programa.

El concepto de encapsulación se resume como sigue:

"Un objeto proporciona encapsulación, en cuanto que pone en servicio una estructura de datos y un grupo de procedimientos para accederla, de tal forma que los usuarios de esa facilidad puedan ingresar a través de interfaces cuidadosamente documentadas, controladas y estandarizadas. Esas estructuras de datos encapsuladas, denominadas objetos, cuentan con datos activos a los que se les puede pedir que hagan cosas, mandándoles mensajes"²⁴.

A continuación se presenta una tabla²⁵ en la que se describen los elementos que constituyen (que se encapsulan) o forman parte de un objeto.

Un objeto encapsula	
Datos	Son los valores definidos para el objeto
Métodos (también llamados operaciones y/o funciones)	Acciones que se aplican para cambiar los atributos del objeto
Otros objetos	Se pueden definir objetos compuestos dentro del mismo
Constantes	Valores preestablecidos

²⁴ PRESSMAN, Roger S.. INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S.A. pág. 423.

²⁵ Tabla tomada de: DATE, C. J. : Introducción a los sistemas de bases de datos, Vol. I, trad. Jaime Malpica; Américo Vargas Villazón, 5 ed., Wilmington, Delaware, E:U:A., Addison Wesley Iberoamericana, S.A., 1993, pág. 676.

4.5.2. CLASES, INSTANCIAS Y HERENCIA

Antes de iniciar el diseño de un sistema de base de datos se deben analizar las necesidades y los elementos que serán tomados en cuenta, para adecuarse a los requerimientos propios de cada usuario. De allí, que sean importantes los conceptos como clase, instancia y herencia.

Muchos objetos del mundo físico tienen características similares y realizan operaciones similares. Si observamos la planta de fabricación de una fábrica de equipos pesados, veremos máquinas fresadoras, taladradoras y perforadoras. Aunque cada uno de esos objetos es diferente, todos pertenecen a una **clase superior** denominada "máquinas herramienta". Todos los objetos de la **clase** máquinas herramienta tienen atributos comunes (por ejemplo, todos usan motores eléctricos) y realizan operaciones comunes (por ejemplo, corte, arranque, parada). Por tanto, clasificando un "torno" como miembro de la clase de máquinas herramienta, ya sabemos algo acerca de sus atributos y sus operaciones, incluso sin saber exactamente cuál es su función.

Las representaciones en software de objetos del mundo real se clasifican de forma muy parecida. Todos los objetos son miembros de una clase mayor y *heredan* la estructura de datos privada y las operaciones que han sido definidas para esa clase. Dicho de otra forma, una clase es un conjunto de objetos que tienen las mismas características. Así, un objeto es una *instancia* de una clase mayor.

Para entender el punto de vista orientado a los objetos es ilustrativo contar con un ejemplo del mundo real. Como ejemplo se tomará una silla. **Silla** es una instancia (también se utiliza el término "miembro") de una *clase* de objetos mucho mayor que denominamos **mueble**. Se puede asociar un conjunto de *atributos* genéricos a cada objeto de la clase **mueble**. Por ejemplo, todo mueble tiene **precio, dimensiones, peso, situación y color**, entre muchos atributos posibles, se aplican se esté hablando de una mesa o de una silla, de un sofá o de una cómoda. Dado que una silla es un *miembro* de la clase **mueble**, *hereda* todo los atributos definidos para la clase (ver Fig. 4.2)²⁶.

Por ejemplo, el objeto **silla** (y en general todos los objetos) *encapsula* datos (valores de los atributos definidos para la silla), *operaciones* (acciones que se aplican para cambiar los atributos de la silla), *otros objetos* (ya que se pueden definir objetos compuestos), *constantes* (valores preestablecidos) y otra información relacionada.

Con la clase ya definida, los atributos de ella se pueden reutilizar para crear nuevas instancias de la clase. Así por ejemplo, si se define un nuevo objeto al

²⁶ Figura tomada del libro de Roger S. Pressman, Ingeniería Del Software, un enfoque práctico, 3.ª ed., 1993, pag. 252.

que se le denomina *mella* (un cruce entre una mesa y una silla) que sea miembro de la clase *mueble*. *Mella* hereda todos los atributos de *mueble*.

Cada objeto de la clase *mueble* puede ser vendido y comprado, modificado físicamente (por ejemplo, se le puede quitar una pata o pintarlo de verde), o moverlo de un lugar a otro.

Cada una de esas *operaciones* (*servicios* o *métodos*) modifican uno o más atributos del objeto.

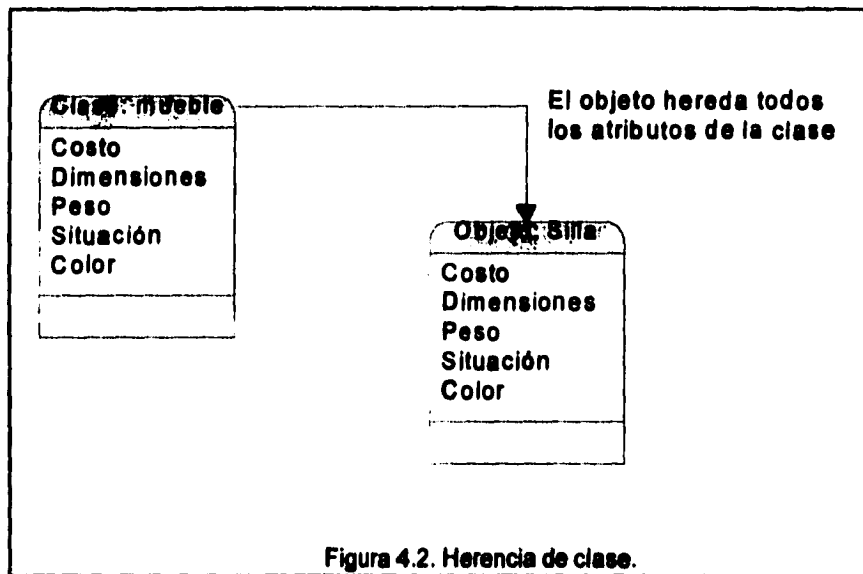


Figura 4.2. Herencia de clase.

Por ejemplo, si definimos el atributo *situación* como un elemento de datos compuesto como:

situación = edificio + piso + habitación

entonces una *operación* denominada *mover* modificará uno o más de esos elementos de datos (*edificio*, *piso* o *habitación*) que componen el atributo *situación*. Para hacerlo, *mover* ha de ser "*consciente*" de la existencia de esos elementos de datos. Se puede utilizar la operación *mover* para una mesa, una silla, mientras que sean instancias de la clase *mueble*. Todas las operaciones válidas (por ejemplo, *comprar*, *vender*, *pesar*) para la clase *mueble* están

relacionadas a la definición del objeto tal como muestra la figura 4.3²⁷, y son heredadas por todas las instancias de la clase.

Existen ocasiones en que la herencia no es completa. Esta situación se produce cuando una instancia candidata de una clase comparte la mayoría, pero no todos los atributos de la clase y requiere todas las operaciones de la clase, así como otras adicionales, que sólo son relevantes para dicha instancia.

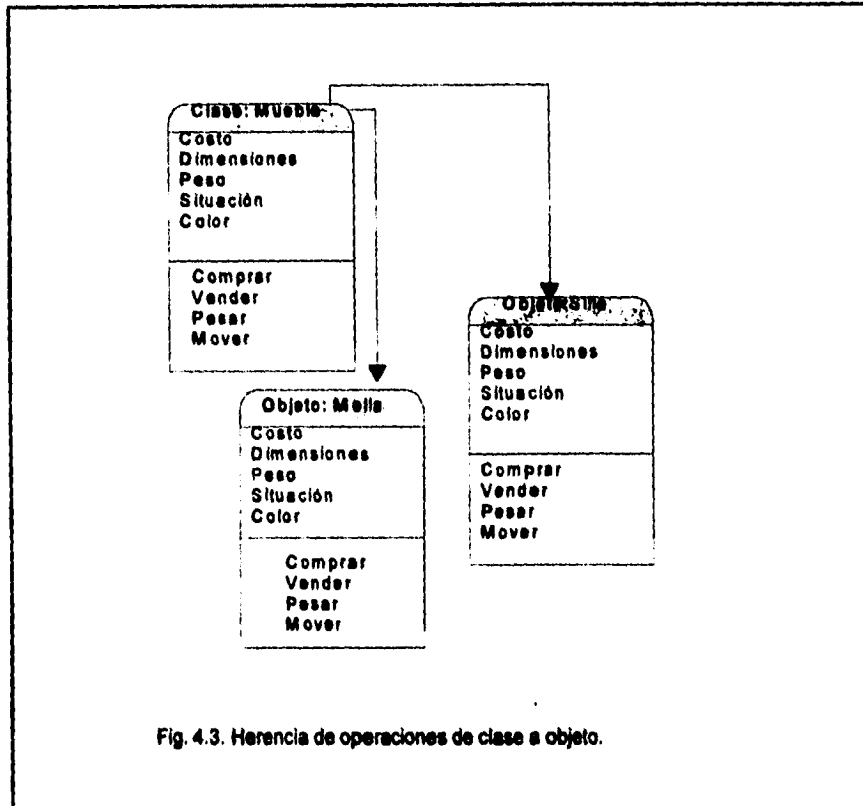


Fig. 4.3. Herencia de operaciones de clase a objeto.

El uso de clases, subclasses y herencia es de crucial importancia. La reutilización de componentes de software se lleva cabo creando objetos (instancias) que se forman sobre los atributos y las operaciones existentes heredadas de una clase o subclase. De esta manera, sólo es necesario especificar cómo son las diferencias entre el nuevo objeto y la clase, en lugar de definir todas las características del nuevo objeto.

²⁷ Figura tomada del libro de Roger S. Pressman, Ingeniería Del Software, un enfoque práctico, 3.ª ed., 1993, pag. 253.

4.5.2.1. JERARQUIA DE CLASES

Normalmente en una base de datos existen muchos objetos similares y se dice que son similares porque responden a los mismos mensajes, utilizan los mismos métodos y tienen variables del mismo nombre y tipo.

Al agrupar los objetos similares se forman las *clases*. A cada uno de los objetos pertenecientes a una clase se les llama *instancia* de su clase. Todos los objetos de una clase comparten una definición común, aunque difieran en los valores asignados a las variables.

Existen en el concepto de clase, aspectos que van más allá de los tipos abstractos de datos. Para representar dichos aspectos se trata a cada clase como si fuera un objeto. Así, un objeto clase incluye:

"Una variable con valores en un conjunto cuyo valor es el conjunto de todos los objetos que son instancias de la clase.

Implementación de un método para el *nuevo* mensaje, el cual crea una nueva instancia de la clase."²⁸

En general, un esquema de base de datos orientada a objetos requiere un gran número de clases. Pese a ello, se da el caso de que varias clases son similares.

"Por ejemplo, supóngase que tenemos una base de datos orientada a objetos para una aplicación bancaria. Sería de esperar que la clase de clientes del banco fuera similar a la clase de empleados del banco en la definición de las variables de nombre, dirección, número de teléfono, etc. Sin embargo existen variables específicas para los empleados (por ejemplo salario) y variables específicas para los clientes (por ejemplo tasa-credito). Sería deseable definir una representación para las variables comunes en un solo lugar,. Esto puede hacerse sólo si los empleados y los clientes están combinados en una clase.

Para permitir la representación directa de similitudes entre clases necesitamos colocar clases en una jerarquía de especialización. Los empleados y los clientes pueden representarse por clases que son especializaciones de una clase *PERSONA*. Las variables y los métodos específicos de los empleados se asocian a la clase *EMPLEADO*. Las variables y los métodos específicos de los clientes se asocian a la clase *CLIENTE*. Las variables y los métodos que se aplican tanto a los empleados como a los clientes se asocian a la clase *PERSONA*. La figura 4.4., muestra una jerarquía de clases que representa personas implicadas en la

²⁸ KORTH, Henry F. - SILBERSCHATZ, Abraham, Fundamentos de Bases de Datos, trad. Ma. Angeles Vaquero Martín; Antonio Vaquero García, 2 ed., Madrid, Mc Graw-Hill / Interamericana de España, 1993, pág. 459.

operación del ejemplo bancario. Las variables asociadas a cada clase en el ejemplo son:

PERSONA: número-seguro-social, nombre, dirección, número-teléfono-particular, fecha-de-nacimiento.

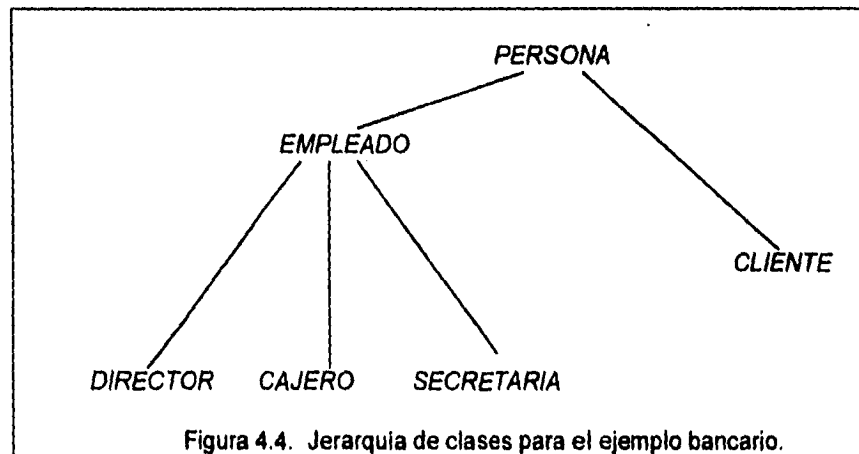
CLIENTE: tasa-crédito, estado-retención-impuestos, número-teléfono-trabajo.

EMPLEADO: fecha-de-contrato, salario, número-de-dependientes.

DIRECTOR: título, número-despacho, número-cuenta-de-gastos.

CAJERO: horas-por-semana, número-estación.

SECRETARIA: horas-por-semana, supervisor.²⁹



4.5.2.2. CLASIFICACION Y ASOCIACION

En un programa orientado a objetos, el énfasis está en la funcionalidad de la abstracción. En cuanto a la estructura concierne, significa que el énfasis está en la herencia de los métodos encapsulados. En un sistema de base de datos orientado a objetos, hay un mucho mayor requerimiento para modelar las propiedades estructurales de los datos y hacer explícita la herencia y el

²⁹ KORTH, Henry F. - SILBERSCHATZ, Abraham, Fundamentos de Bases de Datos, trad. Ma. Angeles Vaquero Martín; Antonio Vaquero García, 2 ed., Madrid, Mc Graw-Hill / Interamericana de España, 1993, págs., 459-460.

agregado de estructuras. Este tipo de estructuras pueden emerger en varias formas diferentes.

Los objetos pueden ser agrupados en clases acordes a su estructura común y propiedades funcionales; esto es **clasificación**. Por ejemplo, los empleados y clientes pueden ser agrupados como gente; a su vez gente, animales y plantas pueden ser agrupados como cosas vivientes. No hay una manera única de clasificar objetos. Los objetos pueden ser clasificados como pertenecientes a más de una clase.

Las *estructuras de clasificación (o redes)* representan generalización y especialización, vehículos son generalizaciones de automóviles y especializaciones de artefactos. Cercanamente a la clasificación está la **asociación**. Aquí las instancias concretas son agrupadas dentro de una clase acorde a algunas propiedades que comparten. Un ejemplo de una asociación de objetos que no es una clasificación de ellos puede ser el conjunto de todas las cosas que son rojas o el conjunto de cosas que pesan más de una tonelada

4.5.2.3. ESPECIALIZACION Y GENERALIZACION

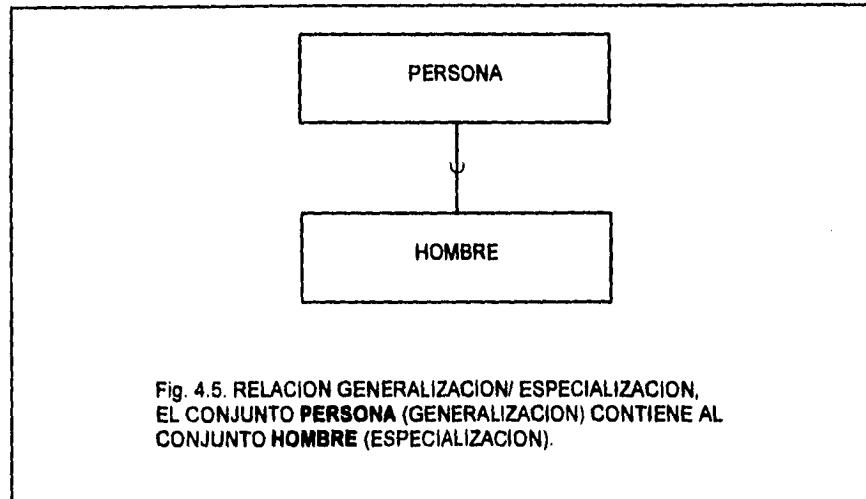
Como ya se dijo, algunas clases de objetos están contenidas dentro de otras clases de objetos. por ejemplo, *HOMBRE* (la clase de hombres) está contenido dentro de la clase *PERSONA*. Esto significa que cada hombre (cada instancia de la clase *HOMBRE*) es también una persona (una instancia de la clase *PERSONA*). Lo mismo sucede con la clase de *MUJER*, porque también está contenida dentro de la clase *PERSONA*.

Se dice que *HOMBRE* es una *especialización* de (o subclase de) *PERSONA* y se representa de la siguiente forma:

$$HOMBRE \subset PERSONA$$

PERSONA, por otro lado, es una *generalización* o *superclase* de *HOMBRE* (y de *MUJER*). Se designa la especialización / generalización relacionándolas como se muestra en la fig. 4.5³⁰.

³⁰ Figura tomada de: HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 84.



El símbolo en forma de \cup , apunta hacia la clase que contiene a la otra clase. Es decir, las "puntas" del símbolo \cup apuntan hacia la clase mayor. Por ejemplo, en el caso de las clases *HOMBRE* y *PERSONA* las puntas del símbolo \cup apuntan hacia la clase *PERSONA*, como se puede observar en la figura anterior.

Si un objeto es una especialización de otro objeto, entonces la especialización del objeto hereda todos los atributos y relaciones del objeto del cual es especialización.

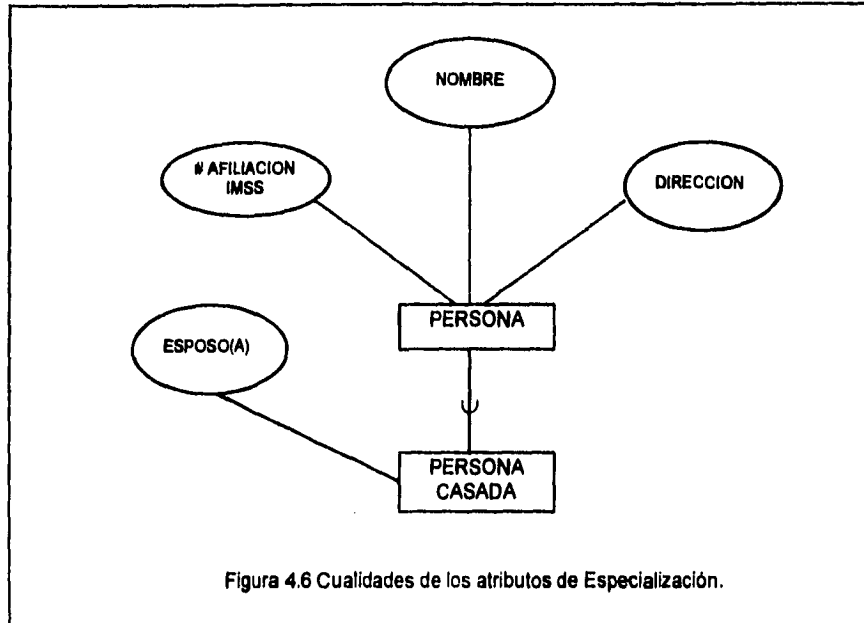
PERSONA-CASADA, por ejemplo, es una especialización de *PERSONA*. Como tal, una persona casada también tiene un nombre, un número de seguridad social, dirección, etc., y esto se debe a que es una persona.

Así la clase de objetos *PERSONA-CASADA* hereda estos atributos de la clase de objetos *PERSONA*.

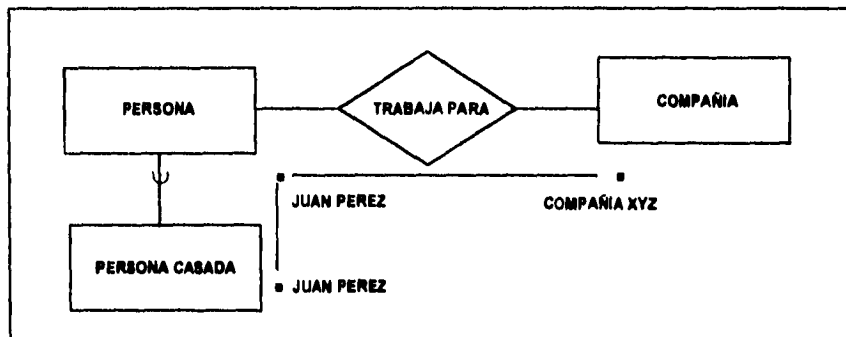
En adición, la especialización de la clase de objetos puede tener sus propios atributos. Por ejemplo, *ESPOSA* sería un atributo de *PERSONA-CASADA*, pero no de *PERSONA*.

Estos conceptos son ilustrados en la figura 4.6³¹.

³¹ Las Figuras 4.3 y 4.4 fueron tomadas de: HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 81.



Una especialización no únicamente hereda atributos, también hereda todas las relaciones. La figura 4.7 ilustra qué persona está relacionada a *COMPañIA* vía *TRABAJA-PARA*.



PERSONA-CASADA, siendo una especialización de *PERSONA*, está también relacionada a *COMPañIA* vía *TRABAJA-PARA*.

Supóngase que Juan Pérez es una persona casada que trabaja para la compañía XYZ. entonces hay un punto en *PERSONA-CASADA* que representa a Juan Pérez, un punto en *PERSONA* representando a Juan Pérez y un punto en *COMPañÍA* representando a la compañía XYZ. Juan Pérez en *PERSONA-CASADA* está relacionada a Juan Pérez en *PERSONA* que a su vez está relacionado a la compañía XYZ. Consecuentemente, Juan Pérez en *PERSONA-CASADA* está relacionado a la compañía XYZ.

La herencia de atributos y relaciones es un importante concepto, ya que permite definir subclases de clases de objetos que tienen atributos y relaciones de su propiedad pero además retienen todos los atributos y relaciones de la superclase. Esto hace posible modelar la realidad de una manera más precisa, que si no existiera el concepto de herencia.

La clasificación algunas veces representa un subconjunto de relaciones y una asociación una relación de miembros. Alguna cosa que es una instancia en una aplicación puede ser una clase en otra. Por ejemplo, en una base de datos de términos científicos, homínido es una instancia, mientras que en una base de datos antropológica homínido puede tener varias instancias, o claro subclases.

4.5.2.4. ATRIBUTOS

El valor del atributo está determinado únicamente por cada instancia del objeto. Por ejemplo, cada persona tiene una fecha de nacimiento y (para una base de datos particular) un número de seguridad social. Si una instancia particular de un objeto no tiene valor para uno de sus atributos, se dice que ese atributo tiene un valor nulo para la instancia del objeto.

Es importante destacar, que los atributos deben ser guardados conceptualmente separados de los objetos que describen. De esta manera, los valores de los atributos cambiarán frecuentemente mientras que los objetos asociados con ellos continúan siendo los mismos. Así, una persona cambiará de peso, estatura, nombre y color de pelo, pero seguirá siendo la misma persona. Esto no significa que todos los atributos cambian valores. De hecho, frecuentemente se trata de identificar atributos que no cambien, porque éstos pueden ser utilizados como llaves.

4.5.2.5. LLAVES

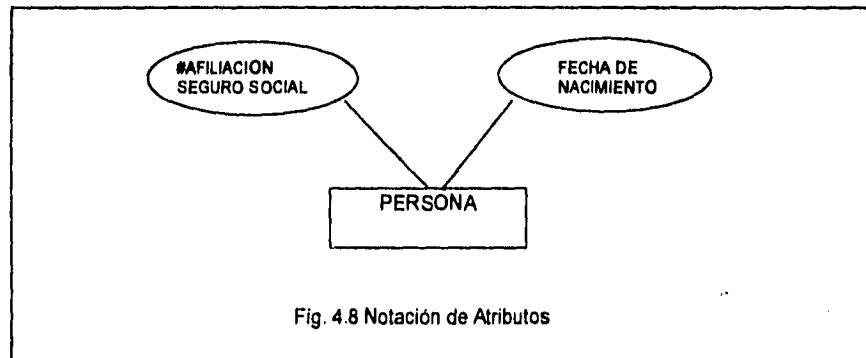
Una llave es una valor que se utiliza únicamente para identificar la instancia de un objeto.

Una llave externa es un atributo léxico o clase de atributos léxicos cuyos valores siempre identifican una instancia de objetos sencilla. Un atributo léxico es un

atributo formado utilizando una clase de objetos léxicos. Así, las llaves externas pueden ser impresas y leídas por los usuarios.

Dichas llaves sirven, por lo tanto, como significado con el cual instancias específicas de objetos pueden ser identificadas externamente al sistema de bases de datos.

Usualmente, a las llaves externas se les conoce simplemente como llaves. Por ejemplo, en la figura 4.8³², #AFILIACION SEGURO SOCIAL, podría ser una llave para persona, si se asume que cada número de seguridad social corresponde exactamente a una persona. En este caso, la cardinalidad mínima y máxima de #AFILIACION SEGURO SOCIAL para PERSONA es uno a uno. Fecha de nacimiento, por otro lado, no puede ser una llave, ya que cualquier fecha de nacimiento dada es la fecha de nacimiento de un gran número de personas.



Algunas veces son necesarios más de un atributo para formar una llave. Si por ejemplo, PERSONA, en la figura 4.8 está siendo usada en una base de datos genealógica, para delinear arboles familiares, pero donde mucha de la gente murió antes de que el número de seguridad social fuera introducido, se necesita algo más que el número de seguridad social. Puede ser suficiente el nombre, la fecha de nacimiento y el lugar de nacimiento. Si es así, la combinación de estos tres atributos formarían la llave para PERSONA. Si no lo es, entonces algún atributo más tendría que ser agregado.

Siempre que sea necesario, se puede componer un número de identificación cuya unicidad pueda ser forzada dentro del sistema.

³² Figura tomada de: HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 89.

En general se utiliza el término **conjunto de objetos** o **clase** para referirse a un conjunto de elementos del mismo tipo e **instancia de objeto** para referirse a un miembro en particular (o elemento) de esa clase o conjunto de objetos.

Los conjuntos de objetos o clases son representados con un rectángulo, mientras que las instancias (miembros de esa clase) son representados por medio de puntos. En la figura³³ 4.9., se observa gráficamente este concepto.

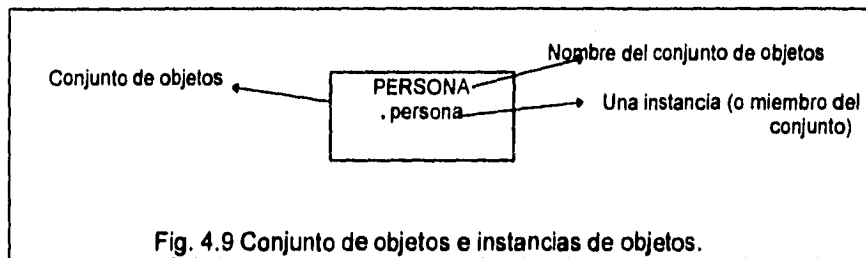


Fig. 4.9 Conjunto de objetos e instancias de objetos.

De esta forma, el nombre de la clase o conjunto de objetos está dado en letras mayúsculas. En este caso, "PERSONA" es el nombre de una clase conjunto de objetos que representa gente. Una "persona" (en minúsculas) representa una instancia de la clase o conjunto de objetos PERSONA.

Los conjuntos de objetos o clases pueden ser **léxicos** o **abstractos**. Las instancias en los conjuntos de objetos léxicos pueden ser impresas, mientras que las instancias en los conjuntos de objetos abstractos no pueden ser impresas. Por ejemplo, NOMBRE sería un conjunto de objetos léxico, ya que las instancias en NOMBRE son nombres, los cuales están contruidos con cadenas de caracteres que pueden ser impresos. FECHA, CANTIDAD y NUMERO-DE-SEGURIDAD-SOCIAL son otros ejemplos de conjuntos de objetos léxicos, ya que fechas, cantidades y numero de seguridad social pueden también ser impresos.

PERSONA, como tal, es abstracta, porque una persona no puede ser impresa. Aunque es verdad que una persona puede ser representada por un objeto léxico tal como un nombre o un número de seguridad social.

Es necesario insistir, que una persona no es un nombre o un número de seguridad social. Por ejemplo, el nombre de una persona o el número de seguridad social pueden cambiarse, pero la persona continúa siendo la misma persona. Sin embargo para alcanzar un modelo más exacto de la realidad, se tiene que distinguir entre los conjuntos de objetos abstractos y los conjuntos de objetos léxicos.

³³ Figura tomada del libro de HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pag. 82.

4.6. RELACIONES Y CARDINALIDAD

Una relación liga dos clases de objetos. Si se consideran las clases de objetos *HOMBRES-CASADOS* y *MUJERES-CASADAS*. Se Puede definir la relación *ESTA-CASADO-CON* asociando estas dos clases, es decir, cada hombre casado con su esposa o inversamente cada mujer casada con su esposo, la relación *ESTA-CASADO-CON* consiste de una clase de parejas casadas, el esposo viene de la clase de *HOMBRES-CASADOS* y la esposa viene de la clase de *MUJERES-CASADAS*, se puede representar la relación entre estas dos clases de objetos como se muestra en la figura 4.10³⁴.

Además de las clases existen otro tipo de clases, como las que se mencionan enseguida. Una relación es en sí misma una clase de objetos, y consiste en parejas de instancias tomadas de las dos clases de objetos que se relacionan, esto es, cada instancia de una relación es un par de instancias de dos clases de objetos.

Si la clase *HOMBRES-CASADOS* es igual a {Juan, Pedro, Pablo} y la clase *MUJERES-CASADAS* es igual a {María, Linda, Mirna} y...

Juan está casado con María
Pedro está casado con Linda
Pablo está casado con Mirna

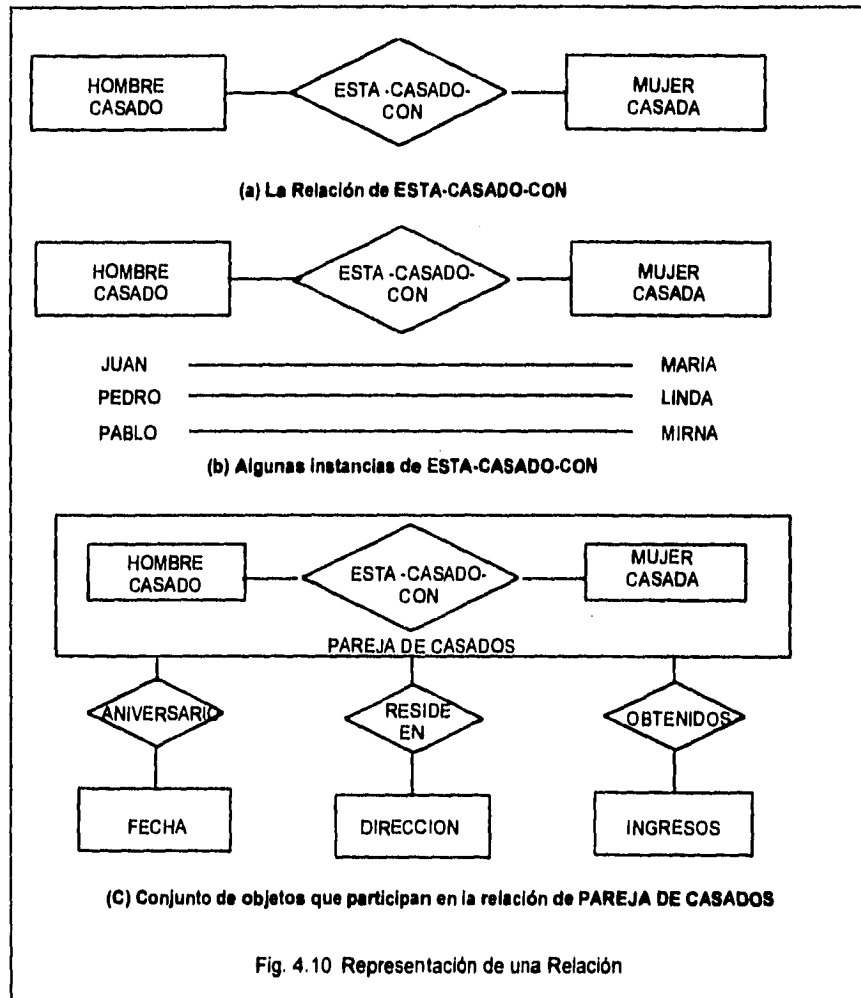
Entonces *ESTA-CASADO-CON* es igual a {(Juan, María), (Pedro, Linda), (Pablo, Mirna)}

Una clase de objetos como *ESTA-CASADO-CON* la cual se obtiene de una relación de otras clases de objetos, es llamada una clase de objetos agregada.

En otras palabras, "los objetos pueden ser agrupados como componentes de algún objeto complejo de dos formas, ya sea como parte de una estructura o como un conjunto de atributos. Esto es **composición o agregación**. Por ejemplo, un carro es un compuesto de ruedas, transmisión, carrocería, etc. A esta noción se le llama estructura de composición. La estructura de agregación es más general e incluye la posibilidad de la composición conceptual también como la composición física. Por ejemplo, una computadora es un concepto agregado de los conceptos que incluyen nombre, sistema operativo, otro tipo de software, ubicación, derechos de acceso, etc. Agregando los atributos y los métodos es como los objetos son formados conceptualmente en la programación orientada a objetos.

³⁴ Figura tomada de: HANSEN, Gary W., - HANSEN, James V. Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 84.

Un objeto que es un agregado de varios objetos contiene sus atributos y métodos. Un objeto que hereda atributos y métodos de un supertipo también hereda sus estructuras agregadas. Esta consecuencia no aparece frecuentemente en los programas orientados a objetos, pero es crucial en las bases de datos³⁵.



³⁵ GRAHAM, Ian, Object Oriented Methods. 1er. ed., Great Britain, Addison Wesley, 1992 (reimpresión), pág. 167.

**ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA**

4.6.1. CARDINALIDAD

La *cardinalidad* de una relación se refiere al número máximo de objetos que están relacionados a una instancia simple de otra clase de objetos. Por ejemplo, si se asume que cada persona casada tiene únicamente un esposo la cardinalidad de la relación *ESTA-CASADO-CON* es una en cada dirección, aunque normalmente el interés es sobre la cardinalidad máxima, algunas veces es útil especificar la cardinalidad mínima. Por ejemplo, si en la relación *ESTA-CASADO-CON* se toma en cuenta que puede haber parejas que no estén casadas entonces la cardinalidad mínima es cero en ambas direcciones.

Algunas relaciones no tienen un valor específico para la cardinalidad, ya que puede ir desde una cardinalidad de cero hasta una cardinalidad de "muchos". Por otro lado, si se considera una fábrica en la que un supervisor supervisa a uno o muchos trabajadores la cardinalidad puede ser uno a uno, o uno a muchos.

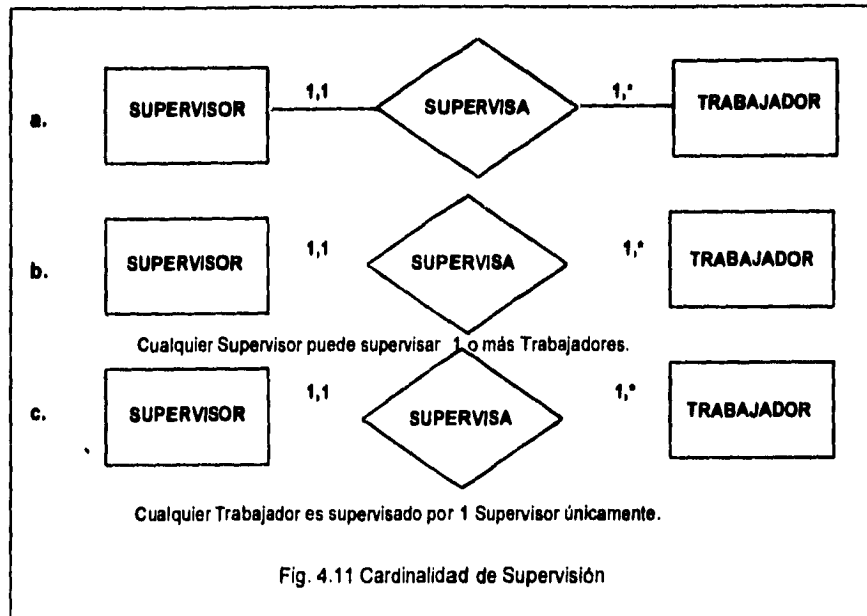
4.6.2. RELACION FUNCIONAL

Una cardinalidad máxima de uno en una dirección corresponde al concepto matemático de una función que forma una correspondencia de uno a uno o muchos a uno entre dos clases. Así, una relación de cardinalidad máxima de uno en una dirección es llamada *relación funcional* en esa dirección.

La relación supervisor/trabajador (figura 4.11³⁶) es funcional del trabajador hacia el supervisor, porque al saber el nombre de un trabajador se puede determinar quién es su supervisor. Esta relación no es funcional en la otra dirección, ya que un supervisor tiene muchos trabajadores.

Si la cardinalidad máxima en ambas direcciones de una relación es uno, se dice que la relación es 1:1 (uno a uno). Si es uno en una dirección y muchos en otra, se dice que la relación es 1:* (uno a muchos). Finalmente, si la cardinalidad en ambas direcciones es muchos a muchos entonces la relación es *: * (muchos a muchos).

³⁶ Figura tomada de: HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 87.



La tabla 4.1³⁷ resume las tres relaciones básicas de la cardinalidad.

CARDINALIDAD	NOTACION	EJEMPLOS
Uno a Uno	1:1 o 1-1	Un Esposo tiene una Esposa. Una Esposa tiene un Esposo. La relación de casados es de uno a uno.
Uno a Muchos	1:* o 1-*	Un empleado está en un Departamento. Un Departamento tiene muchos empleados. La relación de empleo es de uno a muchos.
Muchos a Muchos	*:* o *-*	Un estudiante toma muchos cursos. Un curso tiene muchos estudiantes. La relación de inscripción es de muchos a muchos.

Tabla 4.1 Tres relaciones básicas de Cardinalidad

³⁷ Tabla tomada de: HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992, pág. 88.

Una vez expuestos, en este capítulo, los fundamentos y conceptos básicos que envuelven el desarrollo del modelo de las bases de datos orientadas a objetos, el siguiente capítulo estará dedicado al aspecto del diseño lógico del modelo de bases de datos orientadas a objetos.

CAPITULO 6

DISEÑO LOGICO DEL MODELO DE BASE DE DATOS ORIENTADO A OBJETOS

Un mayor nivel de abstracción en los sistemas de cómputo es sin duda un objetivo deseable, y el paradigma de la orientación a objetos ha proporcionado valiosas herramientas para alcanzar ese objetivo en el campo de los lenguajes de programación. Es natural, entonces, preguntar si es posible aplicar el mismo paradigma en el área de las bases de datos, en vez de tener que preocuparse por las actualizaciones de registros, claves, etcétera. Esta idea es desde luego más atractiva, desde el punto de vista del usuario, al menos en lo superficial.

En esta metodología de diseño se piensa en términos de objetos en lugar de archivos. Los objetos pueden ser elementos como: persona, que a su vez contiene al objeto empleado, mismo que puede ser representante de ventas, cajero, secretaria. De esta forma, se pueden encontrar diversos objetos anidados dentro de otros haciéndolos más complejos. También se encuentran objetos como productos, directores, ventas, etc., que pueden no contener ningún objeto más definido dentro de sí, lo que los convierte en objetos simples. Sin embargo, lo más importante es que se piensa en términos de nombres que relacionan a los objetos. Por ejemplo, un director "dirige" a un representante de ventas. El director y el representante de ventas son los objetos. "Dirige" es una relación entre ellos. Como los problemas llegan a ser más complejos, el diseño orientado a objetos ayuda a estructurar más claramente las relaciones entre los objetos para almacenar un mayor número de éstas.

5.1. DESCRIPCION Y CARACTERISTICAS DE DISEÑO

Una vez que han sido definidas las características de los objetos y las clases, es necesario trabajar con sus descripciones.

La descripción de diseño de un objeto (una instancia de una clase o de una subclase) puede tener dos formas distintas:

La descripción del protocolo³⁸ que establece la interfaz del objeto, definiendo cada mensaje que puede recibir el objeto y las operaciones que realiza el objeto cuando recibe el mensaje.

³⁸ La descripción del protocolo no es otra cosa que un conjunto de mensajes con sus correspondientes comentarios asociados. Para grandes sistemas con muchos mensajes, a menudo se pueden crear categorías de mensajes.

La descripción de la implementación del objeto, que muestra los detalles de cada operación implicada en la recepción de un mensaje por el objeto. Los detalles de implementación incluyen la información sobre la parte privada del objeto, esto es, los detalles internos de la estructura de datos, y los detalles de procedimiento que describen las operaciones.

La descripción de la implementación de un objeto proporciona los detalles internos ("ocultos") requeridos para su implementación, aunque no necesariamente para su invocación. Es decir, el diseñador del objeto proporciona una descripción de la implementación del objeto y, consecuentemente, crea los detalles internos de dicho objeto. Sin embargo, otro diseñador o programador que utilice el objeto u otras instancias del objeto, sólo requiere la descripción del protocolo, no la descripción de la implementación.

La descripción de la implementación está compuesta por la siguiente información:

- 1) Una especificación del nombre del objeto y de la referencia a una clase.
- 2) Una especificación de la estructura de datos privada, con indicación de los elementos de datos y sus tipos.
- 3) Una descripción del procedimiento de cada operación o, alternativamente, referencias a dichas descripciones. La descripción de la implementación debe contener suficiente información para que se puedan manejar de forma adecuada todos los mensajes descritos en la descripción del protocolo.

"Las diferencias entre la información contenida en la descripción del protocolo y la contenida en la descripción de la implementación, son caracterizadas en términos de los *usuarios* y de los *suministradores* de los servicios. Un usuario de un *servicio* proporcionado por un objeto debe familiarizarse con el protocolo de invocación de ese servicio, es decir, con la forma de especificar qué es lo que desea. Al suministrador del servicio (el propio objeto), lo que le concierne es cómo proporcionar el servicio al usuario, es decir, los detalles de implementación"³⁹.

5.2. MODULARIDAD

El diseño orientado a objetos (DOO), al igual que otras metodologías de diseño orientadas a la información, crea una representación del campo del problema del mundo real y la hace corresponder con el ámbito de la solución, que es el

³⁹ PRESSMAN, Roger S., INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S.A., págs. 422-423.

software. El DOO produce un diseño que relaciona objetos de datos (elementos de datos) y operaciones de procesamiento de tal forma que modulariza la información y el procesamiento, en lugar de dejar aparte el procesamiento.

"La naturaleza única del diseño orientado a objetos queda reflejada en su capacidad de construir sobre tres pilares conceptuales importantes del diseño de software: Abstracción, ocultamiento de información y modularidad"⁴⁰.

Bertrand Meyer, sugiere cinco criterios para conseguir la modularidad (aunque desde luego puede haber otros más, se mencionarán sólo estos) y los relaciona con el diseño orientado a los objetos:

"Descomponibilidad: La facilidad con la que un método de diseño ayuda al diseñador a descomponer un gran problema en subproblemas más fáciles de resolver.

Componibilidad: El grado en que un método de diseño asegura que los componentes del programa(módulos), una vez diseñados y construidos, pueden ser reusados para crear otros programas.

Comprensibilidad: La facilidad con la que se puede comprender un componente de un programa sin tener que referenciar otra información ni otros módulos.

Continuidad: La capacidad de realizar pequeños cambios en un programa y que esos cambios se manifiesten por sí mismos con sólo unos cambios correspondientes en uno o en un número pequeño de módulos.

Protección: Una característica que reduce la propagación de efectos laterales para prevenir un error en un módulo dado."⁴¹

A partir de estos criterios, Meyer sugiere la derivación de cinco principios básicos de diseño para construcciones modulares:

1) **Unidades lingüísticamente modulares.**- Los módulos se definen como unidades lingüísticamente modulares cuando corresponden a unidades sintácticas del lenguaje utilizado. Es decir, el lenguaje de programación que se vaya a utilizar debe soportar directamente la modularidad.

⁴⁰ PRESSMAN, Roger S., INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S.A., pág. 415.

⁴¹ PRESSMAN, Roger S., INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S.A., pág.418.

2) Pocas interfaces.- El concepto de pocas interfaces se refiere al hecho de minimizar el número de interfaces entre módulos.

3) Interfaces pequeñas.- Al minimizar la cantidad de información que se mueve a través de una interfaz, se consigue lo que se llama interfaces pequeñas.

4) Interfaces explícitas.- Siempre que los módulos deban comunicarse, lo deben hacer de una forma obvia y directa.

5) Ocultamiento de información.- El principio de ocultamiento de información se consigue cuando toda la información de un módulo está oculta para un acceso desde el exterior, a menos que la información sea definida específicamente como *información pública*.

Los criterios y principios presentados se pueden aplicar a cualquier método de diseño, aunque el método de diseño orientado a objetos consigue cada uno de esos criterios de forma más eficiente que los otros enfoques.

5.3. METODO DE DISEÑO ORIENTADO A OBJETOS

El diseño orientado a los objetos permite a los desarrolladores indicar los objetos que se derivan de cada clase y las interrelaciones entre ellos.

"El diseño orientado a objetos debe comenzar con una descripción en lenguaje natural (por ejemplo, español) de la estrategia de solución, mediante una representación en software, de un problema del mundo real. A partir de esa descripción, el diseñador puede identificar los objetos y las operaciones"⁴².

El método de diseño orientado a objetos, está caracterizado por los siguientes pasos:

- 1) Definir el problema.
- 2) Desarrollar una estrategia informal (narrativa de procesamiento) para la representación en software del campo del problema del mundo real.
- 3) Formalizar la estrategia mediante los siguientes subpasos:
 - a. Identificar los objetos y sus atributos.
 - b. Identificar las operaciones que se les puede aplicar a los objetos.
 - c. Establecer interfaces que muestren las relaciones entre los objetos y las operaciones.

⁴² PRESSMAN, Roger S., INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S.A., pág. 423

d. Decidir aspectos del diseño detallado que proporcionen una descripción de la implementación de los objetos.

- 4) Volver a aplicar los pasos 2, 3 y 4 recursivamente.
- 5) Refinar el trabajo realizado en el análisis, buscando las subclases, las características de los mensajes y otros detalles de elaboración.
- 6) Representar la(s) estructura(s) de datos asociada(s) con los atributos del objeto.
- 7) Representar los detalles de procedimientos asociados con cada operación.

5.4. DEFINICION DE CLASES Y DE OBJETOS EN EL DISEÑO

Los métodos de análisis que mejor se ajustan a un sistema que va a ser implementado con un enfoque orientado a los objetos son los métodos de análisis orientado a los objetos.

La aplicación de los principios y métodos de análisis de requisitos permite al analista y al diseñador llevar a cabo dos subpasos necesarios:

- 1) Describir el problema.
- 2) Analizar y clarificar las limitaciones conocidas del sistema.

La representación en software del problema del mundo real, independientemente de su tamaño o de su complejidad, debe describirse con una frase sencilla y gramaticalmente correcta. Obviamente, el nivel de abstracción puede ser muy alto, pero la frase sencilla que describe el problema permite a los desarrolladores que trabajan en el proyecto comprender el problema de forma sencilla y uniforme.

La orientación a objetos enfatiza la importancia de identificar de forma precisa los objetos y sus propiedades que serán manipuladas por un programa, antes de comenzar a escribir los detalles de esas manipulaciones. Sin esa cuidadosa identificación, será casi imposible precisar las operaciones a realizar y sus efectos deseados.

Los nombres y las frases nominales de la estrategia informal son buenos indicadores de los objetos y sus clasificaciones en la solución del problema.

Así, el cometido del análisis del diseño orientado a objetos es el de aislar todos los nombres y frases nominales contenidos en la narrativa de procesamiento que describe lo que ha de hacer el sistema.

A menudo un nombre común es el nombre de una clase. Por ejemplo, sensor es un nombre común que describe una clase de dispositivos que se usan para detectar algún suceso. Los nombres propios son nombres de entes o cosas específicas. Por ejemplo, *Cavalier*, *Spirit* y *Jetta* son nombres propios o frases nominales de una clase que podríamos denominar **automóvil**. Un nombre abstracto o masivo es el nombre masivo que refiere a una colección de nombres propios de una clase referenciada por el nombre común **automóvil**.

Se puede utilizar esa categorización para ayudar a definir clases, subclases y objetos. A menudo, un nombre común representa una clase de objetos (una abstracción de datos). Un nombre propio representará una instancia de una clase. Un nombre abstracto o masivo (incluyendo unidades de medida) servirá para indicar características restrictivas o agrupamientos específicos del problema para los objetos o las clase. Sin embargo, es importante tener en cuenta que se deben utilizar el contexto y la semántica para determinar las categorías de los nombres. Una palabra puede ser un nombre común en un contexto, un nombre propio en otro y, en algunos casos, un nombre abstracto o masivo en un tercer contexto.

Es importante considerar todos los nombres y frases nominales, ya que serán de interés para la representación en software final de la solución. Algunos objetos residirán fuera de los límites del espacio de solución software. Otros objetos, aunque relevantes para el problema, podrán resultar redundantes o extraños al refinar la solución.

Una vez que se han identificado los objetos del espacio de la solución, el diseñador selecciona el conjunto de operaciones que actúan sobre los objetos. Las operaciones se identifican examinando todos los verbos de la estrategia informal (la narrativa de procesamiento).

Aunque existen muchos tipos distintos de operaciones, generalmente se pueden dividir en tres grandes categorías:

- 1) Operaciones que *manipulan* los datos de alguna forma (por ejemplo, adiciones, eliminaciones, cambios de formato, selecciones).
- 2) Operaciones que realizan algún *cálculo*.
- 3) Operaciones que *controlan* un objeto frente a la ocurrencia de algún suceso.

Los verbos connotan acciones u ocurrencias. En el contexto de la formalización del DOO, no sólo se consideran los verbos, sino también las frases verbales descriptivas y los predicados (por ejemplo: "es igual que"), como posibles operaciones.

El enfoque orientado a los objetos define el objeto como componente de programa que se encuentra enlazado con otros componentes (por ejemplo, datos privados, operaciones). Pero la definición de objetos y operaciones no es suficiente.

Durante el diseño, también debemos identificar las interfaces que existen entre los objetos y la estructura general (en sentido arquitectónico) de objetos.

Aunque un componente de programa es una abstracción de diseño, se puede representar dentro del contexto del lenguaje de programación que se va a utilizar para implementar el diseño.

Una vez identificados los componentes de programa, estamos listos para examinar la evolución del diseño y evaluar la necesidad crítica de cambio. Es probable que la revisión de la definición de "primera mano" de los paquetes produzca modificaciones para añadir nuevos objetos o para volver a pasos anteriores del DOO (es decir, a la estrategia informal), con el fin de valorar la exactitud de las operaciones que han sido especificadas.

5.5. PASOS DEL DISEÑO

El objetivo principal en el diseño orientado a objetos, es definir y caracterizar las abstracciones de una forma que resulte en una definición de todos los objetos, métodos (operaciones) y mensajes importantes.

Aunque puede haber otra metodología para definir los elementos que constituirán al sistema, en general se siguen los siguientes pasos:

"1) Identificar las abstracciones de datos para cada subsistema.

Esas abstracciones de datos son las clases del sistema. A menudo, las clases corresponden a objetos físicos del sistema que se modela. Si no es ese el caso, resulta útil utilizar analogías, obtenidas de experiencias del diseñador en diseños anteriores del sistema (o de otros sistemas). Este es, con mucho, el paso más difícil del proceso de diseño y la selección de las abstracciones tiene influencia sobre la arquitectura completa del sistema.

2) Identificar los atributos de cada abstracción.

Los atributos son las variables de instancia de cada clase. Muchas veces, cuando las clases corresponden a objetos físicos, las variables de instancia requeridas son obvias.

3) Identificar las operaciones de cada abstracción.

Las operaciones son los métodos (o procedimientos) de cada clase. Algunos métodos acceden y actualizan variables de instancia, mientras que otros ejecutan operaciones particulares de la clase. En este momento no se especifican los detalles de implementación de los métodos, sino sólo su funcionalidad. Si una nueva abstracción hereda de otra clase, hay que inspeccionar los métodos de esa otra clase para ver si alguno es redefinido en la nueva clase. El diseño detallado de los métodos queda postergado hasta la etapa del diseño detallado, en la que se usan técnicas de diseño más convencionales.

4) Identificar la comunicación entre objetos.

En este paso se definen los mensajes que se envían unos objetos a otros. Aquí se define la correspondencia entre los métodos y los mensajes que invocan a los métodos. Incluso, aunque no se piense en una implementación orientada a los objetos, los mensajes ayudan en la comunicación del equipo de diseño y se pueden utilizar en el siguiente paso para escribir guiones. El equipo de diseño ha de tomar decisiones sobre este protocolo teniendo como consideración primordial la consistencia en la denominación de los mensajes.

5) Probar el diseño con guiones.

Los guiones, consistentes en conjuntos de mensajes a objetos, prueban la habilidad del diseñador de satisfacer la especificación de requisitos del sistema.

6) Aplicar herencia donde sea apropiado.

Si el proceso de abstracción de datos del paso 1 se ha realizado de forma descendente, se puede especificar la herencia allí. Sin embargo, si las abstracciones se han creado de abajo hacia arriba (a menudo porque los requisitos denominan directamente las abstracciones), la herencia se aplica aquí, antes de pasar a otro nivel de abstracción. El objetivo es reutilizar lo más posible los datos y/o los métodos que ya han sido diseñados. En este paso, a menudo se hacen evidentes semejanzas en datos y operaciones, pudiendo combinar las variables de instancia y los métodos comunes en nuevas clases. Puede que una nueva clase no tenga sentido por sí misma. Su único propósito es recopilar variables de

instancia y métodos comunes, por lo que se le denomina *clase abstracta*⁴³

Los investigadores y desarrolladores de las bases orientados a objetos, han atacado de diferente manera los problemas y deficiencias que presenta esta tecnología, por lo que sería iluso y atrevido señalar que lo expuesto aquí es aceptado como el único camino para el diseño del modelo lógico de dichas bases. Por ello, el presente capítulo sólo proporcionó los conceptos y elementos más utilizados en el diseño lógico de estos sistemas.

En la siguiente sección, se dará un ejemplo sencillo para la aplicación del diseño de base de datos orientada a objetos y la manera como se va definiendo el sistema de acuerdo a las necesidades de cada usuario.

5.6 UN CASO PRACTICO

Para finalizar este capítulo, se presenta un ejemplo práctico del diseño de bases de datos orientado a objetos, dentro de un ambiente bancario en el que existen clientes, cuentas de cheques y de ahorros, entre otros elementos. La idea es ir mostrando cómo se definen y caracterizan los objetos del mundo real, dentro de un modelo de base de datos orientado a objetos.

Anteriormente se dijo, que en un modelo lógico de base datos orientado a objetos se trata de capturar la realidad y representarla dentro de la base de datos.

En este ejemplo⁴⁴, se mostrará el modelo de un banco ficticio al que se le llamará BANCO CENTRAL (BC) y reflejará la realidad de su usuario, Roberto Gómez, Presidente del mencionado banco.

El banco, como se mencionó, maneja cuentas de cheques, cuentas de ahorros y clientes (fig. 5.1a). Al establecer las relaciones apropiadas entre estas clases se tiene la figura 5.1b. De esta manera, se está en posición de responder algunas preguntas:

¿Cuántas cuentas de cheques se tienen?

¿Cuántas cuentas de ahorros?

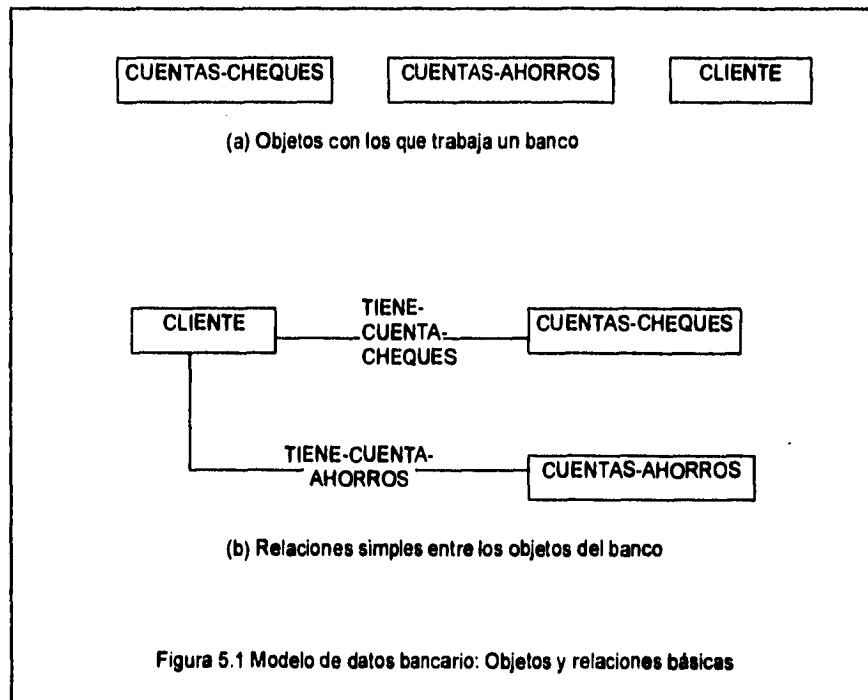
¿Cuántos clientes?

⁴³ PRESSMAN, Roger S., INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S.A., pág. 437.

⁴⁴ El ejemplo que aquí se desarrolla fue tomado de: HANSEN, Gary W., - HANSEN, James V, Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992

Las respuestas se obtienen contando las instancias en cada una de las clases o conjunto de objetos. Con el software apropiado, Roberto Gómez, podría solicitar a su personal de sistemas que respondiera a las preguntas en cualquier momento o incluso él podría recibir un reporte periódicamente.

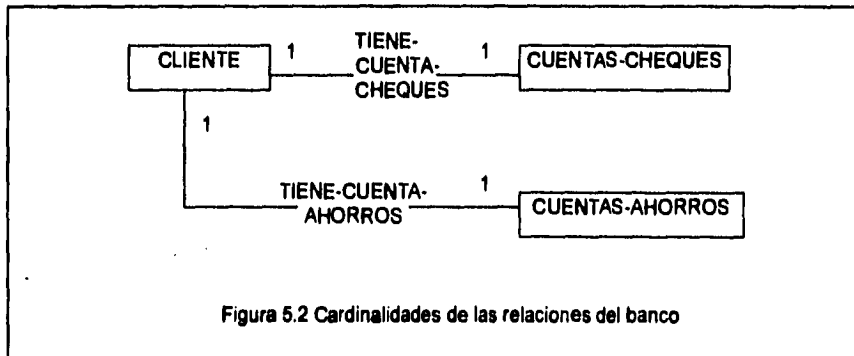
El manejo de una base de datos es más limpio que el de un sistema basado en archivos tradicionales. En sistemas basados en archivos, sin la conexión entre los archivos dado por la base de datos, bien pudiera haber sólo dos archivos (uno para las cuentas de cheques y uno para las cuentas de ahorros). En cada uno de estos archivos, la información de los clientes estaría incluida en un cierto número de campos (nombre del cliente, dirección, etc.). La tercer pregunta sería difícil de responder, ya que se tendría que extraer todos los datos del cliente de los dos archivos, clasificarlos y desechar los que estén duplicados. En una base de datos, sin embargo, estos datos del cliente pueden ser mantenidos separadamente al mismo tiempo que se preserva la conexión deseada con la información de las cuentas.



¿Cuáles son los clientes que tienen tanto cuenta de ahorros, como cuenta de cheques?

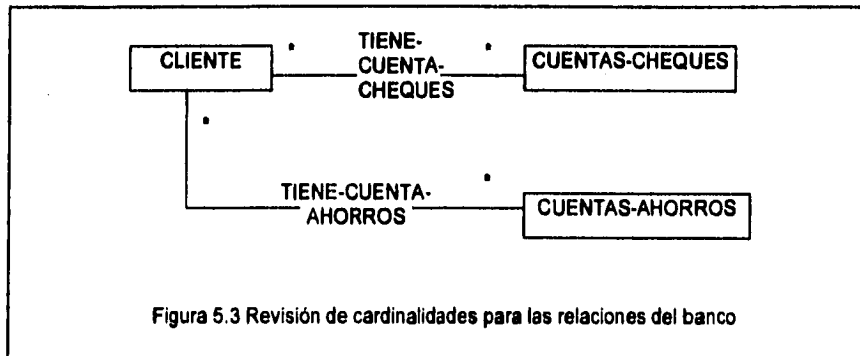
Esta pregunta puede ser contestada únicamente mirando las relaciones. Un cliente tiene una cuenta de ahorros si está relacionado vía TIENE-CUENTA-AHORROS a una instancia en CUENTAS-AHORROS. Similarmente, un cliente tiene una cuenta de cheques si está relacionado vía TIENE-CUENTA-CHEQUES a una instancia en CUENTAS-CHEQUES. Finalmente, un cliente tiene ambas cuentas si está relacionado vía instancias en CUENTAS-AHORROS y CUENTAS-CHEQUES. Para responder a las preguntas formuladas arriba únicamente se cuentan todos los clientes que están relacionados de esta forma.

Cardinalidad. La figura 5.1b, intencionalmente omite la cardinalidad de las relaciones. Estas serán tratadas ahora. Supóngase que se indican las cardinalidades como se muestra en la figura 5.2. Dichas cardinalidades indican que un cliente no puede tener más de una cuenta de cheques o una cuenta de ahorros. Es decir, para cada cuenta existe solamente un cliente.



Las cardinalidades pueden no ser copia fiel de la realidad que se está tratando de reflejar. Considérese la siguiente cardinalidad para CUENTAS-CHEQUES. ¿Puede un cliente tener únicamente una cuenta de cheques? BC, como muchos bancos, permite a un cliente tener más de una cuenta de cheques, pero las cardinalidades de la figura 5.2 no permiten esto.

Si se ven las otras cardinalidades, ¿Es realista suponer que una cuenta será asignada a no más que un cliente? Esto parece improbable, ya que unir las cuentas entre esposo y esposa y entre padres e hijos es lo más común. Para reflejar de una manera más precisa la percepción de la realidad que se tiene, se actualiza la figura 5.2 y queda, de esta forma, la figura 5.3.



El modelo en la figura 5.2 es incorrecto porque no refleja la percepción de la realidad de BC. Quizá una percepción diferente de la realidad pudiera hacer que el modelo de la figura 5.2 estuviera correcto. Un banco, por ejemplo, puede decidir que ningún cliente pueda tener más de una cuenta de algún tipo dado y que no haya unión de cuentas. En ese caso, la figura 5.2 representaría correctamente la realidad de ese banco. Un modelo puede estar correcto o incorrecto de acuerdo a las necesidades de la realidad que se quiera reflejar.

Regresando al modelo de la figura 5.3, ya se pueden contestar las siguientes preguntas adicionales:

¿Cuántos clientes tienen múltiples cuentas de cheques?

¿Cuántas cuentas de cheques unidas se tienen?

¿Cuántos clientes con cuentas múltiples de cheques tienen una cuenta de ahorros?

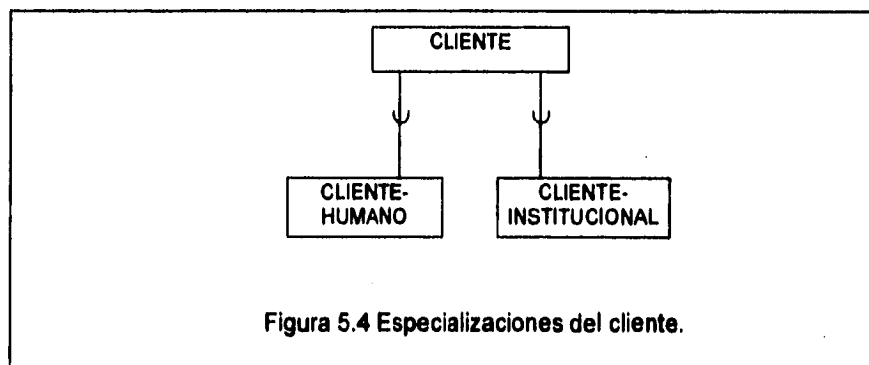
Para contestar a la primera pregunta, obsérvese lo siguiente: Un cliente tendrá múltiples cuentas de cheques si ese cliente está relacionado vía TIENE-CUENTA-CHEQUES en al menos dos diferentes instancias en CUENTAS-CHEQUES y examinando cada instancia en CLIENTE para ver si está también relacionado y contando las instancias que tienen esa característica; las otras dos preguntas se contestan de manera similar.

¿Son los clientes del banco siempre gente?

Puede ser que alguno de los clientes del banco sean organizaciones: De negocios, sin fines de lucro, iglesias, agencias de gobierno. ¿Desea Roberto Gómez, gerente del Banco Central, distinguir los diferentes tipos de cliente con que cuenta el banco?, la respuesta es sí, ya que diferentes tipos de clientes

tienen diferentes atributos y, en consecuencia, sus cuentas pueden tener características diferentes. La figura 5.4 muestra dos especializaciones de CLIENTE. La clase CLIENTE-HUMANO es, por supuesto, el conjunto de clientes que es gente. La clase CLIENTE-INSTITUCIONAL contiene aquellos clientes que son organizaciones.

Una de las principales ventajas de utilizar la especialización y la generalización es que se pueden crear diferentes atributos para la especialización de una clase de objetos, mientras que al mismo tiempo retiene los atributos comunes al nivel de generalización.



La figura 5.5 muestra que cada cliente tiene un número de cliente asignado, mismo que puede ser usado como una llave, sin embargo, los clientes humanos tienen diferentes atributos a los clientes institucionales.

Si se observa bien, la figura 5.6 es un compuesto de las figuras 5.3 y 5.5 en la que se refleja la especialización de CLIENTE. En la misma figura (5.6) se agregó el atributo BALANCE para cada clase de los objetos de cuentas.

De esta forma, se pueden responder algunas preguntas más.

¿Qué porcentaje de las cuentas de ahorro tienen un balance abajo de N\$1.000?

¿Cuál es el tipo de cliente que tiende a tener el promedio de balance más alto en sus cuentas de cheques?

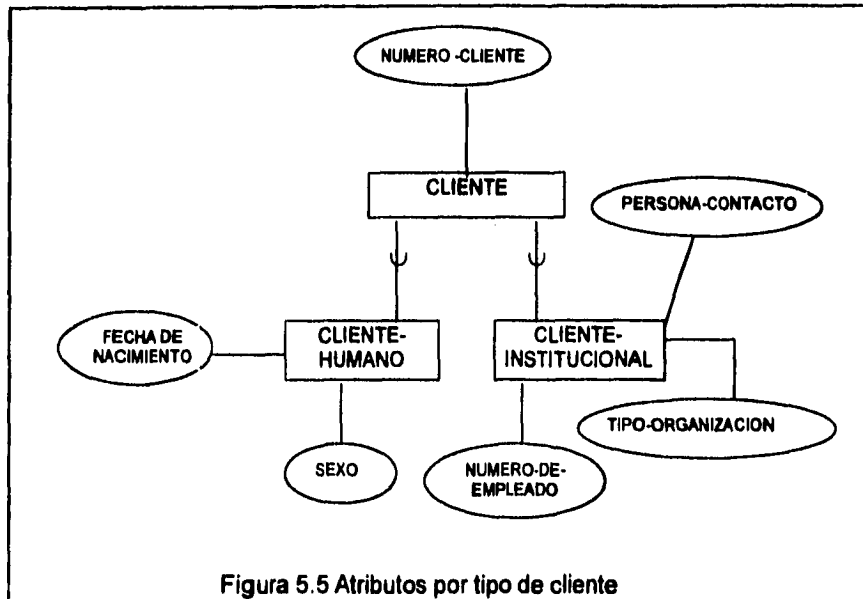


Figura 5.5 Atributos por tipo de cliente

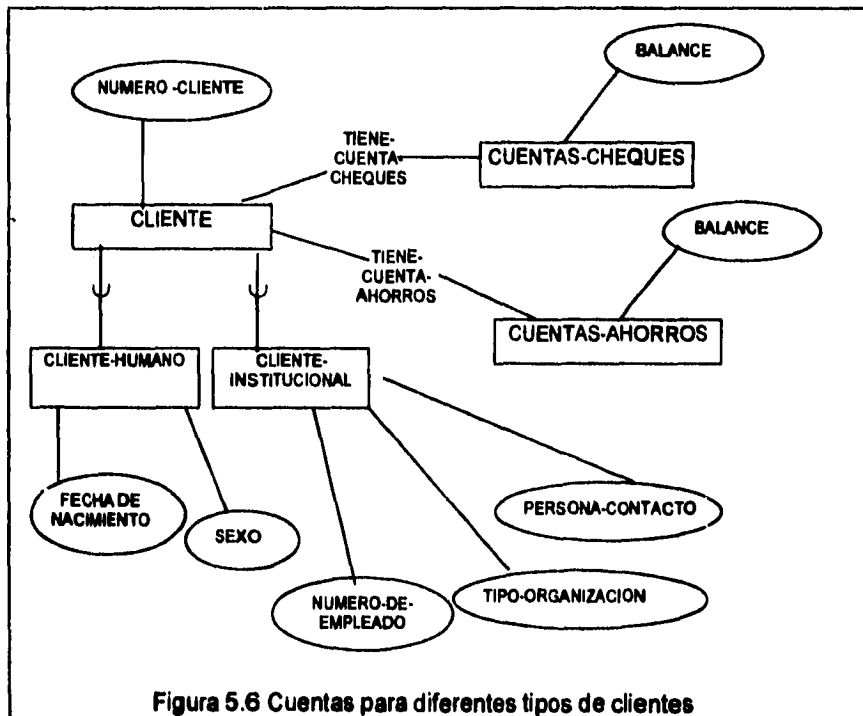


Figura 5.6 Cuentas para diferentes tipos de clientes

La respuesta a la segunda pregunta depende de qué entendemos por "tipos de clientes". El diseño de la base de datos nos permite distinguir entre clientes humanos y clientes institucionales. También, dentro de la clase CLIENTE-INSTITUCIONAL, se pueden determinar los tipos de clientes utilizando el atributo TIPO-ORGANIZACION. Por ejemplo, TIPO-ORGANIZACION podría ser de negocios, no lucrativa, iglesia o agencia de gobierno. Para responder a la segunda pregunta, iniciamos en la clase CLIENTE-HUMANO y continuamos a través de CLIENTE a CUENTAS-CHEQUES vía TIENE-CUENTA-CHEQUES. Se hace esto para cada cliente humano y se registra el balance. Una vez terminado, se suma el balance promedio para los clientes humanos. Después se hace lo mismo para la clase CLIENTE-INSTITUCIONAL. Para responder la pregunta comparamos los dos promedios y obtenemos, finalmente, la respuesta que se busca.

A través de este ejemplo, se fueron repasando conceptos esenciales del modelo orientado a objetos. Si observamos nuevamente la figura 5.6 (Esta figura representa el modelo bancario terminado y responde a los requerimientos planteados en un principio), veremos las clases, las instancias, los atributos que se fueron agregando a los objetos iniciales (fig. 5.1) y la manera como el modelo se fue ajustando a las necesidades del usuario.

El objetivo de este ejemplo fue presentar, de manera simple, el alcance que hoy en día tienen las bases de datos orientadas a objetos y la importancia que, al igual que en otros modelos, tiene el diseño lógico para estos sistemas.

Aunque los modelos orientados a objetos ofrecen ventajas, aun tienen mucho camino por recorrer, debido a la falta de madurez del mercado de las bases de datos orientadas a objetos, lo cual constituye una posible fuente de problemas, además de que la falta de estándares complica el avance en las investigaciones y el afianzamiento de esta tecnología.

Sin embargo, se espera que en la década de los noventa los modelos de bases de datos orientadas a objetos estén a la vanguardia y sean, a su vez, la base para la investigación de nuevos modelos.

CONCLUSIONES

Con el avance científico y tecnológico, se han requerido sistemas de información cada vez más eficientes, confiables y seguros, que permitan actualizaciones y consultas rápidas y de fácil acceso.

De aquí que los sistemas de cómputo basados en archivos se hayan quedado a la zaga al presentar serias deficiencias, entre las que se cuentan redundancia de datos, excesivo esfuerzo de programación y control pobre de los datos entre otras.

Los sistemas de bases de datos, han venido a resolver algunos de los inconvenientes que presentan los sistemas basados en archivos. Estos nuevos sistemas, tienen como característica principal la centralización de los datos, con la que se elimina en gran parte la inconsistencia y multiplicidad de los datos, mejorando con ello la seguridad de la información y el control de los accesos al sistema. Además de proporcionar las herramientas necesarias para el respaldo y recuperación de la base de datos, en caso de que resulte dañada por algún evento fortuito o provocado por la mano del hombre.

En cuanto al diseño lógico, se tiene que destacar la importancia que tiene para cualquier modelo de base de datos ya que de él depende, en gran parte, el éxito o fracaso de un sistema de base de datos, debido al impacto que tiene en las etapas posteriores. Así, cualquier error que se descubra en la etapa del diseño lógico, será menos costoso que si se descubre en una etapa más avanzada. Es decir, un sistema mal diseñado tendrá mayores gastos de operación y mantenimiento provocados por fallas en el sistema que pudieron ser previsibles, por redundancia en los datos o por manejo ineficiente de la información, entre otras causas.

Respecto a los modelos de las base de datos relacionales y de las bases de datos orientadas a objetos, se puede decir que ambos presentan ventajas y desventajas.

Por ejemplo, el modelo relacional presenta limitaciones en el manejo de datos complejos, en el manejo de consultas recursivas y no logra la eliminación total de la redundancia en los datos. Mientras que el modelo orientado a objetos no tiene una definición clara, de aceptación universal, de lo que son los sistemas orientados a objetos y sufre la falta de estandarización en los lenguajes que manipulan las bases de este modelo, además de que el manejo de un registro a la vez por parte de los lenguajes orientados a objetos es un retroceso a los días de los sistemas prerrelacionales.

Pese a ello, se puede afirmar que casi todas las investigaciones actuales sobre bases de datos tienen como fundamento (aunque en algunos casos en forma indirecta) el enfoque relacional y, en términos generales, no puede negarse que el enfoque relacional representa la tendencia dominante en el mercado actual y constituye uno de los avances más importantes en el campo de las bases de datos, ya que descansa sobre un fundamento matemático firme y esa es una de sus principales ventajas.

Por lo que se refiere a los beneficios que presentan las bases de datos orientadas a objetos, se cuentan la reusabilidad del código de programación, mayor modularidad y su facilidad para adaptar, conceptualmente, los objetos del mundo real dentro de la base de datos, además de la flexibilidad y el soporte que proporcionan para el manejo de tipos de datos complejos.

Es importante señalar que los modelos relacional y orientado a objetos no se contraponen sino que se complementan, aunque hayan seguido líneas distintas y como ejemplo se tiene lo siguiente: Aunque aparentemente el enfoque orientado a objetos elimina la necesidad de normalizar, en el sentido de que maneja en forma directa objetos no normalizados (jerarquías), no significa que elimine de manera automática los problemas causados por los objetos no normalizados. La normalización sigue siendo necesaria, a menos que se esté dispuesto a tolerar bases de datos mal diseñadas. Por otro lado, el modelo relacional podría aprovechar la flexibilidad en el manejo de los datos complejos que presenta el modelo orientado a objetos, entre otras posibilidades.

Resumiendo, no obstante que los sistemas basados en manejo de archivos tradicionales son todavía abundantes y dan soluciones efectivas a un buen número de aplicaciones, hoy en día los sistemas relacionales están considerados lo último en los estándares de las operaciones de proceso de datos comerciales y se observa una clara tendencia de las compañías hacia la implantación de estos sistemas. Sin embargo, las investigaciones adicionales prometen dar un entendimiento más completo de las necesidades del usuario con respecto a las bases de datos y en consecuencia continúa el desarrollo de nuevos modelos y al parecer los cambios más significativos están ocurriendo en el área de bases de datos orientadas a objetos. Se proyecta que en la década de los noventa la metodología de tales sistemas sea la más común.

BIBLIOGRAFIA

- 1.- BRATHWAIT, Ken S.; Relational Databases (Concepts, Design and Administration) 1er. ed., E. U. A., Mc Graw Hill, 1991.
- 2.- BURCH, Jhon G. - GRUDNITSKI, Gary; Diseño de sistemas de información (Teoría y práctica), trad. Roberto Pérez Vázquez, 5a. ed-- 1a ed. Español, México Megabyte (ed. Limusa), 1992.
- 3.- CHRISTIE, Linda Gail, Christie John, ENCICLOPEDIA DE TERMINOS DE MICROCOMPUTACION, primera reimpresión, Prentice-Hall Hispanoamericana, S. A.
- 4.- DATE, C. J.; Introducción a los Sistemas de Bases de Datos, trad. Jaime Malpica; Américo Vargas Villazón, 3 ed., Wilmington, Delaware, E. U. A., Addison Wesley Iberoamericana, S. A., 1986.
- 5.- DATE, C. J.; Introducción a los Sistemas de Bases de Datos, Vol. I, trad. Roberto Escalona García; Américo Vargas Villazón; Adoración de Miguel Castaño, 5 ed., Wilmington, Delaware, E. U. A., Addison Wesley Iberoamericana, S. A., 1993.
- 6.- DATE, C.J.; Relational Database, selected writings, 1er. ed., E. U. A., 1986.
- 7.- DATE, C.J; An Introduction to DATABASE SYSTEMS, 5ta. ed., E. U. A., Addison - Wesley, 1991 (reimpresión con correcciones).
- 8.- GRAHAM, Ian; Object Oriented Methods, 1er. ed., Great Britain, Addison Wesley, 1992 (reimpresión).
- 9.- GUILLENSON, Mark L.; Introducción a las BASES DE DATOS, Mc Graw Hill, 1985.
- 10.- HANSEN, Gary W., HANSEN, James V.; Database Management and Design, 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1992.
- 11.- KORTH, Henry F., SILBERSCHATZ, Abraham; Fundamentos de Bases de Datos, trad. Ma. Angeles Vaquero Martín; Antonio Vaquero García, 2 ed., Madrid, Mc Graw-Hill / Interamericana de España, 1993.
- 12.- LUCAS, Jr. Henry C.; Sistemas de Información (Análisis, puesta a punto), trad. Jaime de Montoto y De Simon, 2 ed., Madrid, Paraninfo, 1984.

- 13.- MARTIN, James; Information Engineering (Design & Construction) 1er. ed., Englewood Cliffs, New Jersey, Prentice - Hall, 1990.
- 14.- MARTIN, James; Organización de las Bases de Datos, trad. Adolfo Di Marco, 1er ed., México, Prentice - Hall Hispanoamericana, S. A., 1988.
- 15.- MURDICK, Robert G.; Sistemas de Información Administrativa, trad. Rosa Ma. Rosas Sánchez, 2 ed., México, Prentice - Hall Hispanoamericana, S. A., 1988.
- 16.- PRESSMAN, Roger S.; INGENIERIA DEL SOFTWARE, Un enfoque práctico 3 ed. España, Mc Graw Hill/Interamericana de España, S. A.
- 17.- SEEN, James A.; Sistemas de información para la administración., trad. Dr. Manuel Flores Bravo, 3 ed., E. U. A.-- México, Grupo Editorial Iberoamérica, 1990.
- 18.- SETRAG Khoshafian - RASMIK Abnous; Object Orientation (Concepts, Languages, Databases, User Interfaces) 1er. ed., E. U. A., 1990.
- 19.- TALAVERA, J. Carlos; Bases de datos, conceptos, diseño e implantación. México, ITAM, 1993.
- 20.- WILLIAMS, David; Systems Analysis and Design a Estructurado Approach, 1er., ed., Reading, Massachusetts-Menlo Park, California, Addison-Wesley, 1987 (reimpresión).