36
2EJ

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

ASPECTOS METODOLOGICOS PARA EL DISEÑO DE BASES DE DATOS RELACIONALES

SEMINARIO DE INVESTIGACION INFORMATICA

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN INFORMATICA

P R E S E N T A

GERARDO TONATIUH TORRES TECOTL

ASESOR DEL SEMINARIO:
DRA. JUDITH ZUBIETA GARCIA

MEXICO, D.F. 1995

FALLA DE ORIGEN

DEDICATORIA

A mi madre ***CARMEN****, por los sacrificios tan grandes que ha hecho para ayudarme a salir adelante y a mi padre* ***HELIODORO****, a quien siempre recuerdo con admiración y cariño.*

A mis hermanos ***OSCAR****,* ***BLANCA****,* ***TERE*** *y* ***FABIOLA****, quienes siempre me han apoyado en todas mis metas.*

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México, a quien amo y respeto profundamente, en especial a la Facultad de Contaduría y Administración.

A la Dra. Judith Zubieta García, por compartir sus conocimientos y experiencia para la realización de este trabajo.

INDICE

INTRODUCCION

Anterior a la existencia del concepto de "Base de Datos", el procesamiento de información se realizaba mediante archivos de datos para aplicaciones específicas; el centro de atención eran las funciones del procesamiento, mientras que la estructura de los datos jugaba un papel menos importante. Cuando un sistema de procesamiento de datos era diseñado, la función del sistema debía especificarse primero y después los requerimientos de datos. Un resultado de este enfoque fue que cada aplicación tuvo sus propios archivos para resolver sus propias necesidades lo que implicaba alta redundancia de datos, inconsistencia, poca seguridad y era difícil compartir la información. Con el objeto de resolver estos inconvenientes, surgió el "enfoque de bases de datos", con lo que la administración de la información resultó más fácil de llevar a cabo. Una base de datos centraliza el control sobre todos los datos, reduce redundancia, previene inconsistencias, la información puede compartirse y pueden implantarse medidas de seguridad para su acceso.

El enfoque de base de datos dio como resultado la necesidad de contar con un software como vínculo entre los datos y los usuarios; lo que se conoce como Sistema Manejador de Bases de Datos, mejor conocido bajo las siglas DBMS (en inglés Data Base Management System). Este sistema realiza principalmente las funciones de agregar, recuperar, borrar y modificar información en las bases de datos. El primer DBMS fue Integrated Data Store (IDS), desarrollado en una computadora General Electric, por el Dr. Charles Bachman en 1962. En

1968 IBM anunció su primer DBMS bajo el nombre de Information Management System (IMS). Para ese tiempo todos los DBMS existentes adoptaban los modelos de datos jerárquico y de red. En 1970 Edgar F. Codd, investigador de IBM, propuso el **"modelo relacional para bases de datos"**. A partir de las investigaciones de Codd, científicos de esa misma compañía crearon el primer prototipo de DBMS relacional llamado System/R. Durante la década de los setenta, en el ámbito comercial predominaron las bases de datos con estructuras jerárquicas y de red, mientras que el nuevo modelo relacional sólo tuvo aceptación en instituciones académicas norteamericanas, con el propósito de seguir la línea de investigación y es hasta 1983 cuando IBM lanza al mercado DB2, su primer DBMS relacional para "mainframes".

La Importancia de un Buen Diseño de Base de Datos

Si bien el enfoque de base de datos en general reduce o elimina las desventajas de un sistema convencional de archivos, cuando las estructuras que contienen los datos no se diseñan correctamente, continúan presentes los problemas de redundancia y se corre el riesgo de modificar o borrar datos de tal forma que produzcan información inconsistente. Una base de datos mal diseñada arroja información incorrecta que crea desconfianza en sus usuarios.

Ya que el éxito o fracaso de una base de datos depende de su diseño y no exclusivamente del DBMS utilizado, es importante que quienes tienen asignada la tarea de diseñar una base de datos tengan a su alcance una metodología que les facilite esta labor.

En el entorno informático es evidente el uso de metodologías para el desarrollo de sistemas. Esta tesis propone, en específico, una metodología de diseño de bases de datos relacionales.

El proceso de diseño utilizado incluye la creación de tres modelos de base de datos: conceptual, lógico y físico. El conceptual, es un modelo expresado en términos de entidades, relaciones y atributos; el resultado de esta fase es el diagrama entidad-relación (capítulo 2). El modelo lógico es un modelo expresado en términos de relaciones, dominios, llaves primarias y llaves foráneas; el objetivo de esta etapa, revisada en el capítulo 3, es obtener relaciones normalizadas. Finalmente, el modelo físico es un modelo de la realidad, expresado en términos de archivos y estrategias de acceso tales como índices; el resultado de esta etapa es la especificación para la implantación física con un DBMS, como se expone en el capítulo 4.

El resultado final del diseño es el conjunto de los modelos mencionados que como un todo, debe considerarse como la especificación para la implantación de una base de datos.

Los capítulos anteriormente citados son la parte central de esta tesis, pero previo a estos, el capítulo uno presenta un marco de referencia, que contiene conceptos generales sobre las bases de datos, y de manera especial sobre las bases de datos relacionales. El último capítulo muestra un estudio de caso, en el que se aplica la metodología propuesta a un problema de diseño de bases de datos en la Unidad de Indicadores de Productividad Científica de la Universidad Nacional Autónoma de México.

1. MARCO DE REFERENCIA

1. MARCO DE REFERENCIA

El marco de referencia de esta investigación se divide en dos partes. La primera comprende los conceptos básicos sobre bases de datos. La segunda, cubre de manera específica las bases de datos relacionales.

1.1 Generalidades sobre bases de datos

En este primer apartado se exponen los conceptos básicos sobre bases de datos en general, como su definición, propiedades, los elementos de un sistema de base de datos, modelos y niveles.

1.1.1 Definición de base de datos

Normalmente se habla de una base de datos como un conjunto de archivos, donde cada archivo es un conjunto de datos. Una base de datos se puede definir simplemente como un conjunto organizado de datos. El objetivo de una base de datos es almacenar datos que representan características de hechos o entes que son relevantes para una organización. Por ejemplo, para una universidad es importante el registro de sus estudiantes inscritos. Por otra parte, existe el concepto de sistema de base de datos, que se refiere a la automatización de la base de datos, y que incluye a los datos, el hardware, el software y los usuarios.

1.1.2 Propiedades de las bases de datos

Toda base de datos debe mantener las siguientes propiedades:

(a) Datos compartidos

Una base de datos normalmente no se crea para uso exclusivo de una sola persona o aplicación, sino que es utilizada por un conjunto de personas que hacen uso de ella para

propósitos diferentes y tal vez al mismo tiempo. Por ejemplo, una base de datos de personal es compartida por el Departamento de Personal y el Departamento de Nómina.

(b) Integración de los datos

Una base de datos está integrada cuando existe una unificación de diferentes archivos, libres de datos redundantes o innecesariamente duplicados. La integración de los datos hace posible que puedan ser compartidos.

(c) Integridad de los datos

La integridad de los datos se refiere a su exactitud, completez o validez. Una base de datos conserva su integridad cuando refleja exactamente todos y cada uno de los hechos o entes que representa. Un ejemplo de falta de integridad podría ser que en un registro, un empleado aparezca adscrito a un departamento que no existe. Sin los controles de integridad apropiados, se corre el riesgo de ingresar información errónea o actualizar incorrectamente la base de datos.

(d) Seguridad de los datos

La seguridad se refiere a la protección de los datos contra accesos no autorizados. La restricción de acceso a los datos es una de las principales maneras de asegurar la integridad de la base de datos.

(e) Independencia de datos

Cuando las necesidades de información de los usuarios cambian, la mayoría de las veces es necesario reestructurar la base de datos, lo cual no debe implicar cambios en los programas existentes que operan sobre ella. C. J. Date define la independencia de datos

como "la inmunidad de las aplicaciones a los cambios en las estructuras de almacenamiento y estrategias de acceso"[1]. Cuando hay independencia de datos, existe la libertad de realizar los cambios necesarios sin tener que modificar las aplicaciones existentes. Existen dos niveles de independencia de datos: independencia lógica y física. La independencia lógica se refiere a que la estructura lógica de los datos es independiente a la implantación física y puede modificarse sin tener que cambiar drásticamente las aplicaciones existentes. La independencia física se refiere a que la estructura física de los datos puede modificarse sin tener que cambiar la estructura lógica de los datos y las aplicaciones existentes. La estructura física de los datos debe ser transparente para las aplicaciones que los utilizan.

1.1.3 Elementos de un sistema de base de datos

Como se mencionó anteriormente, un sistema de base de datos se refiere a la automatización de la base de datos, e incluye cuatro componentes: datos, hardware, software y usuarios.

(a) Datos

Un dato es una característica de un hecho o ente que por sí solo no es significativo. Sólo cuando los datos están en conjunto adquieren su completo significado; es decir, se convierten en información.

---

[1] Date, C. J. *An Introduction To Database Systems*. The Systems Programming Series: Addison Wesley. Estados Unidos de Norteamérica. 4ª edición, 1988. p. 16.

(b) Hardware

El hardware de una base de datos consiste básicamente en el medio en que reside de manera física. Normalmente se utilizan medios magnéticos de acceso directo como son los discos duros.

(c) Software

Entre la base de datos física y los usuarios se encuentra el denominado Sistema Manejador de Bases de Datos (Data Base Management System, **DBMS**), que es un conjunto de programas que realiza principalmente las siguientes funciones:

1. Crea la base de datos
2. Permite el acceso a los registros
3. Extrae información de acuerdo a los requerimientos de los usuarios
4. Permite la actualización de los registros.

Para realizar las funciones mencionadas, un **DBMS** tiene como mínimo los siguientes componentes:

1. Un Lenguaje de Definición de Datos (Data Definition Language, **DDL**) que permite crear la estructura de cada registro.

2. Un Lenguaje de Manejo de Datos (Data Manipulation Language, **DML**) que permite agregar, borrar o modificar registros de la base de datos. Los lenguajes de consulta y los generadores de reportes forman parte del DML.

3. Un Diccionario de Datos (DD) que permite mantener la integridad de la base de datos, pues contiene datos de los datos, es decir, información referente a los formatos de los registros, derechos de acceso de los usuarios, etc. No debe confundirse este concepto con el diccionario de datos como herramienta de análisis de sistemas.

4. Un lenguaje de programación, para construir programas o reportes complejos. Actualmente algunos manejadores de bases de datos ofrecen generadores de aplicaciones como parte integral del mismo.

(d) Usuarios

Se considera que hay tres grandes tipos de usuarios: el administrador de la base de datos, el programador de aplicaciones y el usuario final.

El administrador de la base de datos es quien tiene a su cargo el diseño de la base de datos y el control centralizado tanto de los datos, como de los programas que hacen uso de ellos. Entre otras funciones, el administrador de la base de datos define la estructura y métodos de acceso a la base de datos, los tipos de autorización de accesos para los usuarios y los controles de integridad, además de establecer las estrategias de respaldo y recuperación de la información.

El programador de aplicaciones es quien se encarga de escribir los programas que hacen uso de la base de datos.

El usuario final es quien accesa la base de datos por medio de una aplicación escrita por los programadores, o bien a través de una interfase proporcionada por un DBMS.

1.1.4 Modelos de bases de datos

La estructura lógica o esquema de una base de datos puede describirse utilizando cualquiera de los tres modelos de datos más utilizados: jerárquico, red o relacional.

1.1.4.1 El modelo jerárquico de bases de datos

La estructura del modelo jerárquico de base de datos asemeja un árbol que comienza con un nodo raíz que se ramifica en múltiples sub-árboles de nivel inferior. Cada nivel se conecta con los niveles superiores por medio de relaciones padre-hijo, donde un padre puede tener cero o más hijos, y a la vez cada hijo puede ser padre y tener sus propios hijos. Existe una relación "un padre-*n* hijos" (1:N). Para llegar a un nodo en un nivel *n*, el acceso se inicia desde el nodo raíz y luego se recorre el árbol a través de las ligas que unen padres con hijos, hasta que el nodo en el nivel *n* es alcanzado. Cada nodo en el árbol representa un tipo de registro. El manejo de datos en un modelo jerárquico se hace a través de un lenguaje de manejo de datos que permite hacer los recorridos de nodo a nodo dentro de un mismo árbol o a otro diferente para realizar la recuperación o bien actualización de datos. Una desventaja de este modelo es que permite duplicidad de información.

1.1.4.2 El modelo de red de bases de datos

El modelo de red fue creado como una mejora al modelo jerárquico. La estructura es similar, pero difiere en cuanto al número de padres que puede tener un hijo. El modelo en red permite varios padres para cada hijo, es decir existe una relación muchos a muchos (M:N) entre padres e hijos. En el modelo jerárquico se accesa un registro siguiendo la secuencia jerárquica

hacia el registro deseado. En el modelo de red, es posible definir apuntadores desde el nodo raíz hacia cualquier registro tipo, de tal manera que pueda ser accesado sin navegar por toda la secuencia jerárquica.

1.1.4.3 El modelo relacional de bases de datos

El modelo jerárquico y de red comparten ciertas similitudes; ambos usan nodos o registros de almacenamiento para representar objetos. La asociación entre los objetos en ambos modelos se representa físicamente por medio de ligas. El modelo relacional adopta un enfoque muy diferente, y es mucho más sencillo que los anteriores. Su estructura es la llamada relación o tabla; la organización de los datos es vista por los usuarios simplemente como tablas. El modelo relacional utiliza llaves primarias y llaves foráneas para relacionar dos o más tablas; no existe el concepto de ligas. Los operadores relacionales permiten hacer consultas sobre las tablas sin tener que conocer rutas de acceso.

Por la importancia de este modelo para esta tesis, éste se trata con mayor detalle en el apartado 1.2 de este capítulo.

1.1.3 Niveles de un sistema de base de datos

La arquitectura de un sistema de base de datos se divide en tres niveles generales: interno, conceptual y externo.

(a) El nivel interno

Es el nivel más bajo y se refiere a la manera en que los datos están físicamente almacenados. Este nivel es llamado también nivel físico. Una base de datos tiene un solo nivel interno.

(b) El nivel externo

Es el nivel más cercano a los usuarios y se refiere al modo en que cada usuario ve los datos para su propio uso. Así, cada usuario accesa un nivel externo individual.

(c) El nivel conceptual

Es la representación del contenido global de la base de datos. Una base de datos puede tener varios niveles externos, pero sólo un nivel conceptual y un nivel interno.

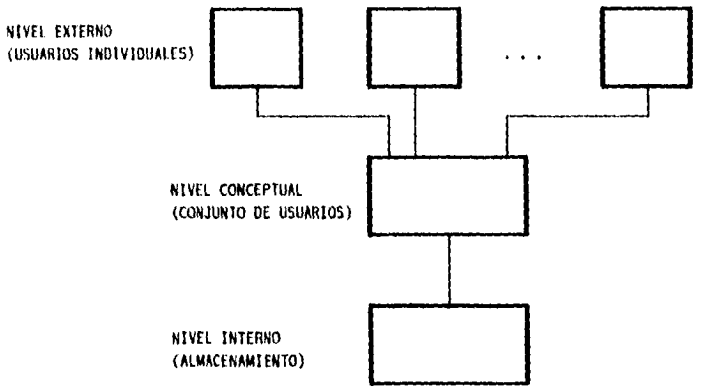

Figura 1.1 Niveles de la arquitectura de un sistema de base de datos.

1.2 El modelo relacional de bases de datos

1.2.1 Antecedentes

Anterior a los años setenta, lo modelos de bases de datos adoptaron estructuras arborescentes, tales como el modelo jerárquico y de red. El modelo relacional surgió en 1970, a partir de las investigaciones del Dr. Edgar F. Codd, quien trabajaba como investigador de la

IBM. Codd propuso el nuevo modelo mediante un documento titulado "A Relational Model of Data for Large Shared Data Banks"[2].

En un principio, la línea de investigación propuesta por Codd sólo causó interés en algunas instituciones académicas de los Estados Unidos. Mientras tanto, IBM construía el primer prototipo de manejador de base de datos relacional, llamado "System R". Pero aún durante los años setenta los modelos de red y jerárquico continuaron siendo los de mayor uso.

Es hasta 1983 cuando IBM lanza al mercado "DB2", su primer DBMS relacional para "mainframes". Desde entonces otras compañías han comercializado sus propios manejadores de bases de datos adoptando el modelo relacional, ahora universalmente aceptado.

1.2.2 Elementos del modelo relacional de bases de datos

El modelo relacional de bases de datos consta de una estructura de datos, un conjunto de operadores relacionales y dos reglas de integridad. Estos tres componentes están vinculados con la definición, la manipulación y la integridad de los datos, respectivamente.

La estructura relacional es la denominada relación o tabla. El conjunto de operadores relacionales es lo que Codd definió como "álgebra relacional": tales operadores producen nuevas relaciones a partir otras existentes. Las dos reglas de integridad son la de integridad de entidad y de integridad referencial.

---

[2] Codd, Edgar F. *A Relational Model Of Data For Large Shared Data Banks*. Communications of the ACM. Volumen 13. Número 6. Estados Unidos de Norteamérica. 1970.

1.2.2.1 La estructura relacional

Una de las mayores ventajas del modelo relacional es la simplicidad de su estructura, la llamada relación. La manera más simple de definir una relación es como una tabla de datos. Una relación está formada por tuplas y cada tupla tiene elementos llamados atributos. Una tupla equivale a un renglón dentro de una tabla y un atributo a una columna. La palabra relación es más apropiada en términos matemáticos formales, mientras que el término tabla es más utilizado en la práctica. En esta tesis ambos términos se utilizan indistintamente.

Toda relación o tabla debe cumplir las siguientes reglas:

(a) Toda relación o tabla debe tener un nombre único y distinto a otras dentro de la misma base de datos.

(b) Cada atributo o columna debe tener un nombre propio, único y distinto a otras, dentro de la misma tabla.

(c) Cada celda o intersección renglón-columna debe aceptar exactamente uno y sólo un valor, nunca un conjunto o arreglo de datos.

(d) Los valores en una misma columna deben ser del mismo tipo.

(e) Cada renglón de la relación debe ser distinto. No debe haber renglones idénticos.

(f) El orden de los renglones no es significativo.

(g) El orden de las columnas no es significativo.

El número de atributos en una relación es el grado de la relación y el número de tuplas es la cardinalidad. Una relación de grado uno es llamada unaria, una relación de grado dos es llamada binaria, una relación de grado tres es llamada ternaria, y una relación de grado *n* es llamada *n*-aria. El grado de una relación normalmente no cambia después de su creación, en cambio la cardinalidad varía según se agregan nuevas tuplas o se eliminan las existentes. En la figura 1.2 se muestra un ejemplo de una relación denominada PROFESORES, que contiene los atributos nombre, domicilio y teléfono.

PROFESORES

nombre	domicilio	teléfono
Enrique Guzmán	Oaxaca 72	6-15-12-89
Amparo Torres	Tepeyac 124	7-56-30-80
Dolores Mendoza	Cuauhtémoc 185	5-79-45-90
Isabel Espinosa	Ciruelos 23	3-93-78-45

Figura 1.2 Ejemplo de una relación de grado tres con cardinalidad cuatro.

El conjunto de valores de donde un atributo obtiene sus valores es el dominio del atributo. Los dominios son conceptuales en su naturaleza y pueden o no estar explícitamente contenidos en la base de datos.

Ya que en una tabla no está permitido tener renglones duplicados, cada tabla debe tener una llave primaria, es decir, un atributo o una combinación de estos que identifique de manera única cada renglón dentro de la tabla.

Las llaves primarias son de fundamental importancia en el modelo relacional, ya que para localizar un renglón dado en una base de datos, es necesaria la combinación del nombre de la tabla y el valor de la llave primaria.

La figura 1.3 muestra un ejemplo de una relación denominada VUELOS, en la que una línea aérea registra sus vuelos con su respectivo origen, destino y horarios de salida y llegada. Si partimos del hecho de que la línea aérea identifica sus vuelos con un número único y no pueden existir dos vuelos con el mismo número, la llave primaria de la relación VUELOS, sería el atributo "#vuelo".

VUELOS

#vuelo	origen	destino	hr_salida	hr_arribo
454	MEX	MAD	18:30	12:30
455	MAD	MEX	17:30	22:30
460	MEX	MTY	8:30	9:45
461	MTY	MEX	10:45	12:00

Figura 1.3 La llave primaria de la relación VUELOS es el atributo "#vuelo", ya que no puede haber dos vuelos con un mismo número.

El concepto de llave foránea es otro aspecto fundamental en las bases de datos relacionales. Una llave foránea es un atributo o una combinación de éstos, dentro de una relación ***R***, que es la llave primaria de otra relación ***S***.

En el ejemplo de la relación VUELOS se puede observar que los valores de los atributos origen y destino corresponden a las claves con los que la línea aérea identifican los aeropuertos.

Los datos de cada aeropuerto se registran en otra relación llamada AEROPUERTOS, cuya llave primaria es el atributo "cve_aeropuerto" (figura 1.4).

A partir de los conceptos anteriores, una base de datos relacional puede definirse simplemente como un conjunto de relaciones. Para Date, una base de datos relacional es "una base de datos que es percibida por sus usuarios como un conjunto de tablas (y nada más que tablas)"[3]. La expresión "percibida por los usuarios" utilizada por Date es importante, pues los conceptos del modelo relacional son aplicables únicamente a los niveles externo y conceptual de bases de datos. De hecho, el modelo relacional no propone nada a nivel interno, más bien se refiere a la manera en que los usuarios ven a la base de datos.

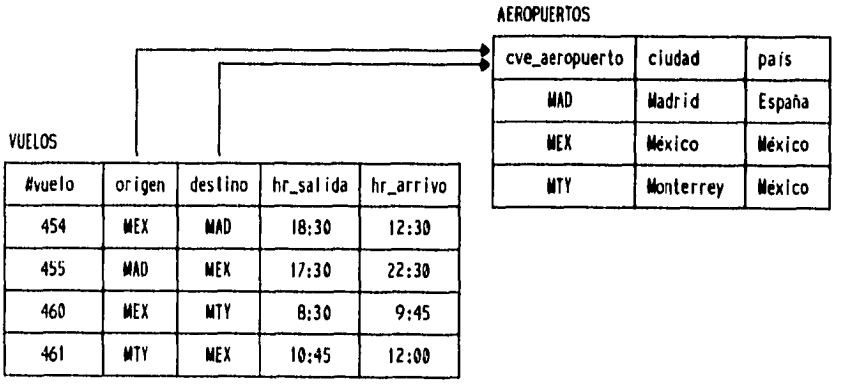

Figura 1.4 En la relación VUELOS los atributos "origen" y "destino" son cada uno llaves foráneas, ya que sus valores son utilizados como llave primaria en la relación AEROPUERTOS.

[3] Date, C. J. *An Introduction To Database Systems*. p. 96.

1.2.2.2 Operadores relacionales de datos

Codd propuso un conjunto de operadores de datos para manipulación de relaciones, a lo que llamó "álgebra relacional", que es la base fundamental de lenguajes tales como el Structured Query Language (SQL).

Los operadores de datos toman como entrada una o más relaciones y producen una nueva como salida. Los operadores relacionales de datos son los siguientes:

(a) **Operador de Selección (Select)**

Toma de la tabla de entrada únicamente aquellos renglones que cumplen una condición dada. Usando el operador de selección el grado de la relación de entrada es el mismo que la de salida. La cardinalidad de la relación de salida es menor o igual que la cardinalidad de la relación de entrada.

Este operador algebraico no debe ser confundido con la sentencia SELECT de SQL, la cual es más compleja, pues incluye la funcionalidad de los ocho operadores algebraicos.

(b) **Operador de Proyección (Project)**

Este operador extrae de una relación, únicamente aquellos atributos especificados. La relación resultante contiene todos los renglones de la relación de entrada, pero no todas sus columnas. La cardinalidad de la relación de entrada es la misma que la de salida. El grado de la relación de entrada es menor o igual al grado de la relación de salida.

(c) Operación de Producto (Product)

Toma dos relaciones y produce una nueva que contiene todas las posibles combinaciones de los renglones de ambas relaciones de entrada. La cardinalidad de la relación de salida es igual a la multiplicación de las cardinalidades de las relaciones de entrada. El grado de la relación de salida es igual a la suma de los grados de las relaciones de entrada.

(d) Operador de Yunción (Join)

Combina las tuplas de dos relaciones de acuerdo a un atributo en común, pero la relación de salida contendrá únicamente aquellas tuplas que cumplen una condición dada. El grado de la relación de salida es igual a la suma de los grados de las relaciones de entrada. La cardinalidad de la relación de salida es menor o igual a la suma de las cardinalidades de las relaciones de entrada.

(e) Operador de Unión (Union)

Toma dos relaciones que tienen la misma estructura y produce una combinación de las dos en una de salida. El grado de la relación de salida es el mismo que el de las relaciones de entrada. La cardinalidad resultante es la suma de las cardinalidades de las relaciones de entrada.

(f) Operador de Intersección (Intersection)

A partir de dos relaciones homogéneas, produce una nueva que contiene únicamente los renglones que aparecen ambas relaciones. El grado de la relación de salida es la misma que el de las relaciones de entrada. La cardinalidad debe ser menor o igual a la suma de las cardinalidades de las relaciones de entrada.

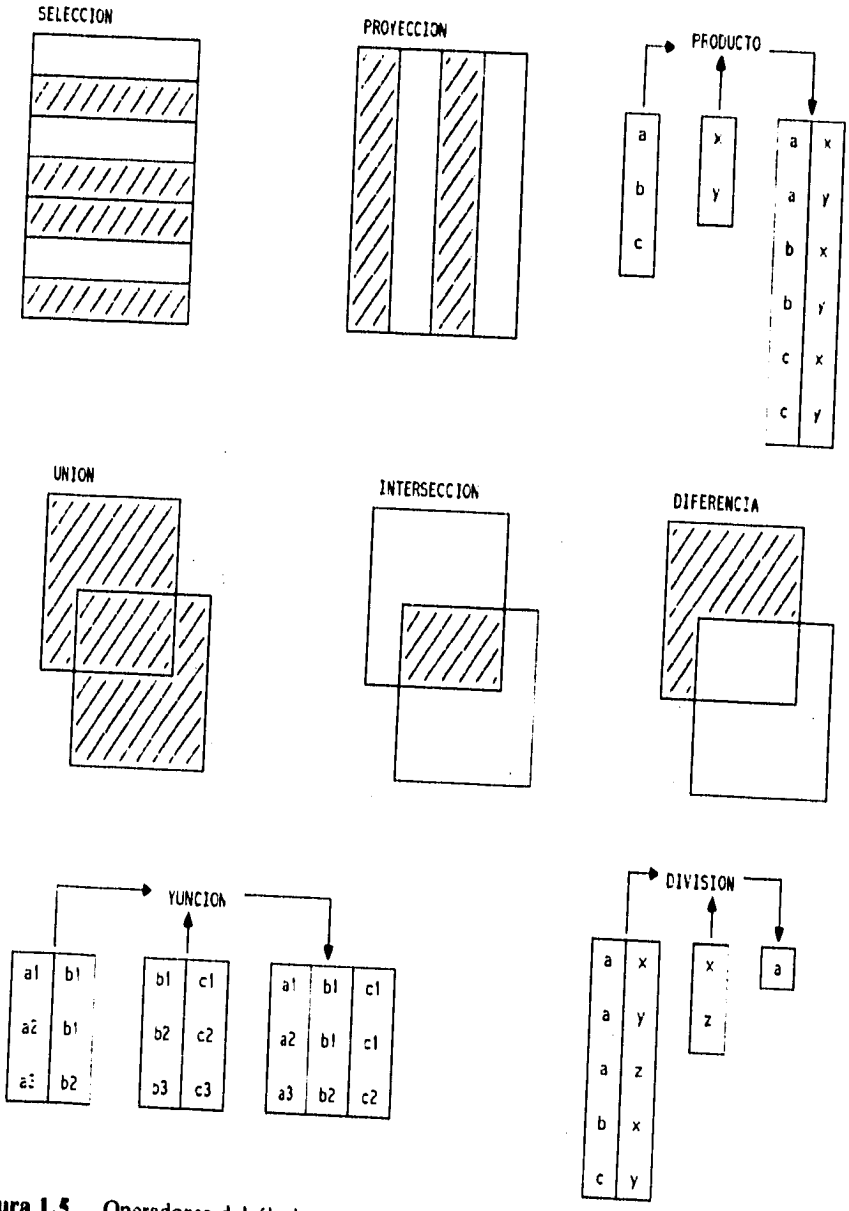

Figura 1.5 Operadores del álgebra relacional.

(g) Operador de Diferencia (Difference)

Toma dos relaciones homogéneas y construye una nu que aparecen en la primera pero no en la segunda. El grado de la relación de salida es el mismo que el de las relaciones de entrada. La cardinalidad debe ser menor o igual a la cardinalidad de la primera relación

(h) Operador de División (Divide)

Toma dos relaciones, una de ellas con un atributo y la otra con dos, y construye una nueva relación que contiene los valores del primer atributo de la relación binaria cuyos segundos atributos es igual a todos los valores de la relación unaria. La relación de salida es unaria y su cardinalidad es menor o igual a la cardinalidad de la relación binaria.

1.2.2.3 Integridad relacional

La integridad del modelo relacional se basa en dos reglas, una llamada de integridad de entidad y otra de integridad referencial. Estas dos reglas son de aplicación general para toda base de datos relacional, pues están directamente vinculadas con los conceptos de llave primaria y llave foránea.

(a) Integridad de entidad

Esta regla establece que cada relación debe tener una llave primaria y que el o los atributos que la forman no pueden contener valores nulos. La combinación de atributos que forma una llave primaria debe ser única.

Como consecuencia de esta regla, no pueden existir renglones duplicados en una tabla. Además, esta regla asegura una eficiente recuperación de información, toda vez que para localizar un renglón dado en una base de datos, es necesaria la combinación del nombre de la tabla y el valor de la llave primaria.

(b) Integridad referencial

Esta regla está relacionada con el concepto de llave foránea, y se define como a continuación:

Si una relación *R2* contiene una llave foránea *FK* vinculada a la llave primaria *PK* de una relación *R1*, entonces cada valor de *FK* en *R2*: (a) debe ser igual al valor de *PK* en alguna tupla de *R1*, o (b) debe ser totalmente nulo.

Un valor nulo en una llave foránea significa que no existe asociación alguna entre los objetos o que esa asociación se desconoce. Las relaciones *R1* y *R2* no necesariamente deben ser distintas, puede tratarse de una relación recurrente, es decir, sus tuplas se relacionan a través de llaves foráneas con otras tuplas dentro de la misma relación.

En el ejemplo de la figura 1.4, la relación VUELOS tiene como llave primaria al atributo "#vuelo" y además tiene dos llaves foráneas, los atributos "origen" y "destino", cuyos valores corresponden a la llave primaria de la relación AEROPUERTOS.

Los aspectos del modelo de datos relacional descritos anteriormente fueron resumidos por Codd en 1985 en un conjunto de reglas (Reglas de Codd) que él definió como condiciones que todo DBMS debe cumplir para que pueda ser considerado relacional (véase anexo A).

2. DISEÑO DEL MODELO CONCEPTUAL

2. DISEÑO DEL MODELO CONCEPTUAL

El diseño de bases de datos es un proceso de creación de modelos: un modelo conceptual, un modelo lógico y un modelo físico. El conceptual es un modelo del mundo real expresado en términos de entidades, relaciones y atributos. El resultado de esta fase es el Diagrama Entidad-Relación (E-R), en el que se describen entidades, relaciones y atributos, de una manera simple y comprensible.

2.1 El Diagrama Entidad-Relación

En 1976, Peter Chen propuso el modelo de datos denominado "Entidad-Relación", en el que combinaba las ventajas de los modelos de datos relacional y de red. En el documento denominado "The Entity-Relationship Model: Toward a Unified View of Data" [1], Chen presentó el nuevo modelo, acompañado de una técnica de diagramación como herramienta para diseño de bases de datos: el diagrama entidad-relación. El nuevo modelo propuesto por Chen no fue explotado comercialmente, pues no tuvo la aceptación esperada entre los usuarios de bases de datos. Sin embargo, el diagrama E-R es actualmente utilizado como una herramienta para representar los requerimientos de datos durante el diseño conceptual de bases de datos.

2.2.1 Entidades

Una entidad es un objeto o evento cuyos datos es necesario conocer y contener en una base de datos. Ejemplos de entidades son estudiantes, empleados, máquinas, contratos, etc.

---

[1] Chen, Peter Pin Shan. *The Entity-Relationship Model: Toward a Unified View of Data*. Communications of the ACM. Volumen 1. Número 1. Estados Unidos de Norteamérica. 1976.

Toda entidad tiene características que la hacen diferente a otras, es decir, tienen atributos propios. Un atributo es una propiedad de una entidad, es cualquier detalle que sirve para calificar, identificar, clasificar, cuantificar o expresar el estado de una entidad. Por ejemplo, para la entidad "Estudiantes" sus atributos serían el número de cuenta, nombre, sexo, fecha de nacimiento, domicilio, etc.

Una entidad es única y diferente a otras; así también sus ocurrencias. Para identificar de modo único una ocurrencia en una entidad se utiliza una llave de entidad como identificador, que puede estar formada por uno o más atributos de la entidad. En el caso de "Estudiantes", la llave de entidad sería el número de cuenta, pues este número es diferente para todos y cada uno de los estudiantes.

En un diagrama E-R, una entidad se representa con un rectángulo, en cuyo interior se escribe el nombre que identifica a la entidad. Para nombrar una entidad se debe elegir un nombre que represente claramente el hecho u objeto que identifica. Regularmente se utilizan sustantivos como nombres de entidades. En el diagrama se indica también cuáles son las llaves de las entidades, escribiendo debajo del recuadro de cada entidad, los nombres de los atributos que forman parte de la llave de entidad (figura 2.1).

Figura 2.1

Cabe mencionar también, que existen dos tipos de entidades: super-entidades y sub-entidades. Una entidad de la cual pueden derivarse otras, es llamada super-entidad. Una sub-entidad es aquella que se deriva de una super-entidad. Por ejemplo, "Estudiantes", "Profesores" y "Empleados" serían sub-entidades de la super-entidad "Personas" (figura 2.2).

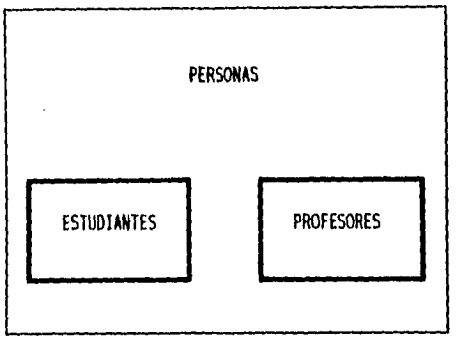

Figura 2.2

2.2.2 Relaciones

Una relación es una asociación entre entidades [2]. Por ejemplo, la entidad "Estudiantes" puede estar asociada con la entidad "Licenciaturas" a través de la relación "Estudian". Una ocurrencia de la relación "estudiantes *estudian* licenciaturas", sería por ejemplo, "Sánchez estudia Informática", donde "Sánchez" e "Informática" son ocurrencias de "Estudiantes" y "Licenciaturas" respectivamente.

---

[2] En este caso el término relación tiene el sentido de *asociación*; no se refiere a la estructura del modelo relacional de bases de datos revisado en el capítulo 1.

Para nombrar relaciones se usa regularmente un verbo. Por ejemplo,

Estudiantes	-	*Estudian*	-	Licenciaturas
Empleados	-	*Laboran en*	-	Departamentos
Tripulación	-	*Atiende*	-	Vuelos
Proveedor	-	*Suministra*	-	Materiales

En un diagrama E-R, la relación entre entidades se representa con un rombo o diamante, en cuyo interior se escribe el nombre de la relación.

El diagrama E-R para el ejemplo de "estudiantes estudian licenciaturas" sería como a continuación:

Figura 2.3

Las relaciones tienen una llave de relación, formada por las llaves de las entidades que relaciona. En el caso de "Estudiantes" y "Licenciaturas", las llaves de entidad son el número de cuenta y la clave de la licenciatura respectivamente. Ambas llaves de entidad conforman la llave de la relación "Estudian".

Para ver con mayor claridad el concepto de relación, véase la figura 2.4, en la que se muestra un diagrama de asociación entre las ocurrencias de las entidades mencionadas. Por un lado se presenta la entidad "Estudiantes" con las ocurrencias "Arciniega", "Colín", "Sánchez" y "Viveros". Por otra parte se presenta la entidad "Licenciaturas" con las ocurrencias "Administración", "Contaduría" e "Informática". La relación se describe mediante líneas que unen las ocurrencias de ambas partes.

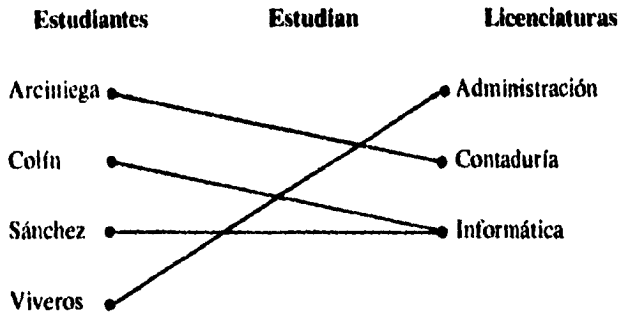

Figura 2.4 Diagrama de asociación de ocurrencias entre "Estudiantes" y "Licenciaturas".

La interpretación del diagrama anterior sería:

El estudiante Arciniega estudia la Licenciara en Contaduría,

El estudiante Colín estudia la Licenciatura en Informática,

El estudiante Sánchez estudia la Licenciatura en Informática

y el estudiante Viveros estudia la Licenciatura en Administración.

En la parte superior de los diagramas de asociación de ocurrencias se escriben los nombres de las entidades y entre ellas, el de la relación. El atributo utilizado para ejemplificar cada entidad debe ser tal que brinde una idea general de lo que es la entidad. En el ejemplo dado, para describir la entidad "Estudiantes" se utiliza parcialmente el atributo nombre, pues no sería claro utilizar el número de cuenta o sería confuso utilizar el atributo edad.

Los diagramas de asociación de ocurrencias son un complemento de los diagramas E-R pues muestran gráficamente, y a modo de ejemplo, la asociación entre ocurrencias de entidades, utilizando valores reales o ficticios de alguna propiedad de las entidades representadas.

2.2.2.1 Grados de relación entre entidades

El grado de una relación entre entidades se refiere al número de entidades asociadas [3]. Así, existen relaciones unarias, binarias, ternarias, ... , *n*-arias. En una relación unaria participa solamente una entidad; este tipo de relación es también conocida como recursiva, pues sus ocurrencias se asocian con otras dentro de la misma entidad. En las relaciones binarias participan sólo dos entidades; este tipo de relación se presenta con mayor frecuencia que los otros tipos. En las relaciones ternarias participan tres entidades y así, en las relaciones *n*-arias participan *n* entidades.

---

[3] Comparar con el concepto dado en el capítulo 1.

2.2.2.2 Cardinalidad de relación entre entidades

La cardinalidad de una relación se refiere al número de ocurrencias que pueden estar asociadas entre entidades [4]. Por lo tanto, existen relaciones uno a uno (1:1), uno a muchos (1:N) y muchos a muchos (M:N). Para distinguir la cardinalidad en un diagrama E-R, se anota un "1", "N" o "M" sobre las líneas de asociación y a un lado de las entidades.

Relación uno a uno (1:1)

Una relación uno a uno existe cuando las ocurrencias de las entidades participantes aceptan como máximo una ocurrencia asociada.

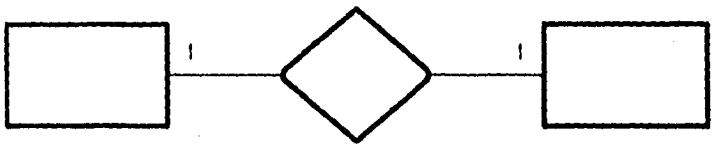

Figura 2.5 Diagrama E-R para una relación uno a uno.

Para ejemplificar una relación 1:1, supóngase que para el préstamo de libros a domicilio de una biblioteca, se establece que los usuarios pueden llevar consigo un libro como máximo. Para efectos de registrar el préstamo, pueden definirse las entidades "Libros", que se refiere a los libros disponibles en la biblioteca; y la entidad "Usuarios", referente a los usuarios de la biblioteca. Se define también la relación "En préstamo a" para asociar ambas entidades como "libros *en préstamo a* usuarios" (figura 2.6).

---

[4] Comparar con el concepto dado en el capítulo 1.

Figura 2.6

El ejemplo en cuestión puede verse con mayor claridad en el diagrama de ocurrencias de la figura 2.7, en el que se observa que un estudiante lleva uno y sólo un libro, y que un libro es prestado a sólo un estudiante a la vez.

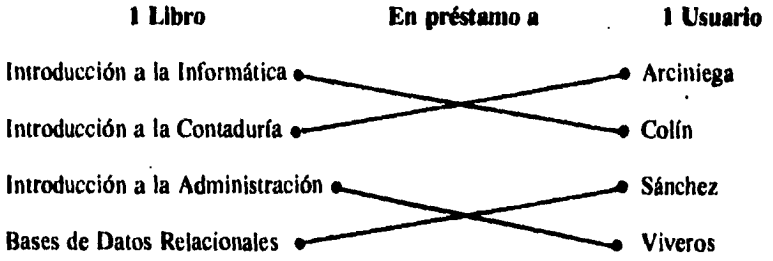

Figura 2.7

Relación uno a muchos (1:N)

En una relación uno a muchos, una ocurrencia de una entidad *A* puede asociarse a *n* ocurrencias de una entidad *B*, y las ocurrencias de *B* se asocian únicamente con una ocurrencia de *A*.

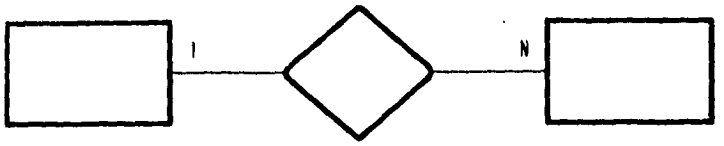

Figura 2.8 Diagrama E-R para una relación uno a muchos.

Para ejemplificar este tipo de relación, se utiliza nuevamente el caso de "Libros *en préstamo* a usuarios", considerando ahora que el préstamo a usuarios puede ser de varios libros a la vez (figura 2.9). En el diagrama de ocurrencias de la figura 2.10 se observa que un usuario puede llevar en préstamo más de un libro y que cada libro es prestado a sólo un usuario a la vez; véase que los títulos "Introducción a la Administración" y "Bases de Datos Relacionales" están "en préstamo a" "Sánchez".

Figura 2.9

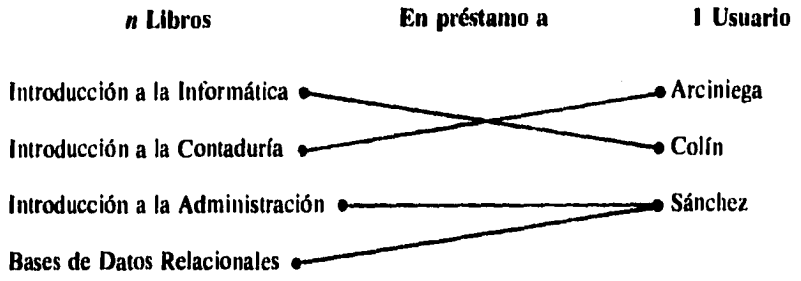

Figura 2.10

Relación muchos a muchos (M:N)

En una relación muchos a muchos, varias ocurrencias de una entidad pueden estar asociadas a varias ocurrencias de otra.

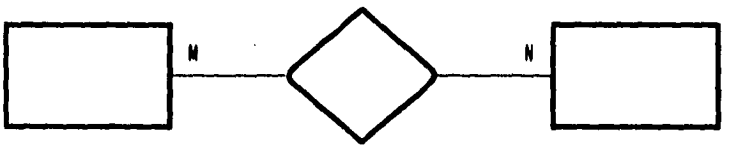

Figura 2.11 Diagrama E-R para una relación uno a muchos.

Supóngase que una universidad ofrece cursos extracurriculares a sus estudiantes. Cada estudiante puede tomar más de un curso a la vez y en cada curso puede inscribirse más de una persona. Para este ejemplo se definen las entidades "Estudiantes" (estudiantes inscritos en la universidad) y "Cursos" (Cursos ofrecidos). La relación para estas entidades sería "Inscritos en". El diagrama E-R se muestra en la figura 2.12.

Figura 2.12

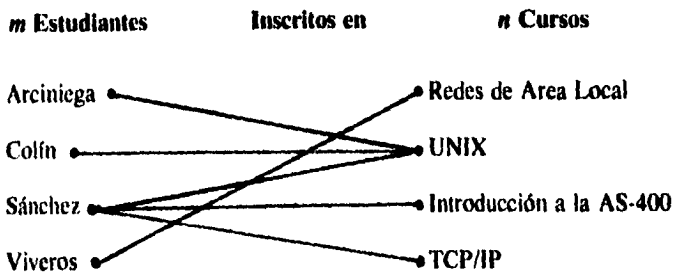

Figura 2.13

Nótese que, "Arciniega", "Colín" y "Sánchez" están "Inscritos en" "UNIX", y que además "Sánchez" toma "Introducción a la AS-400" y "TCP/IP".

2.2.2.3 Participación de entidades en una relación

Habrá ocasiones en que las ocurrencias de una entidad no deban forzosamente estar asociadas a las ocurrencias de otra, en este caso la participación de la entidad en la relación es opcional o parcial. Por otra parte, si es imperativo que una las ocurrencias de una entidad participe en una relación, entonces la participación es obligatoria o total. Para describir la participación obligatoria de una entidad en un diagrama E-R, se coloca un asterisco junto a la

entidad cuyas ocurrencias participan obligatoriamente en la relación. Para las entidades con participación opcional no se hace ninguna anotación especial (figura 2.14).

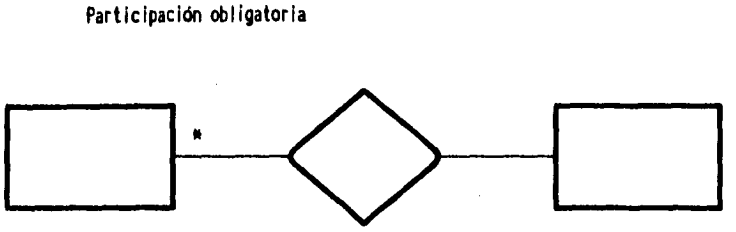

Figura 2.14

Tomando el ejemplo de "Estudiantes *inscritos en* cursos", si todos y cada uno de los estudiantes deben tomar al menos un curso, entonces la participación de "Estudiantes" en la relación es obligatoria. Por otro lado, habrá cursos en los que no haya alumnos inscritos, entonces la participación de "Cursos" en la relación es opcional (figura 2.15).

Figura 2.15

En el diagrama de ocurrencias de la figura 2.16 se advierte que todos y cada uno de los estudiantes están inscritos en al menos un curso y que hay cursos que no tienen alumnos inscritos, como es el caso de "DB2".

Las figuras 2.17 y 2.18 muestran cómo la participación de "Estudiantes" se convertiría en opcional, en el caso de que haya estudiantes que no tomen alguno de los cursos en cierto período y fuera necesario conservar su registro.

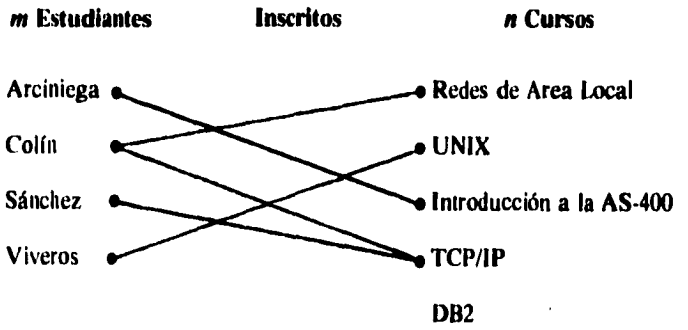

Figura 2.16

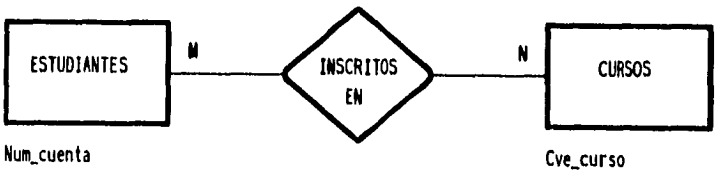

Figura 2.17

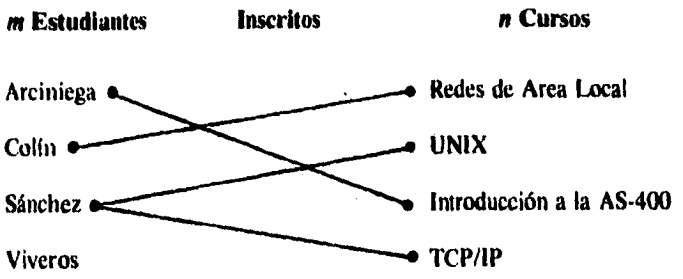

Figura 2.18

En el diagrama de ocurrencias anterior, la participación opcional de "Estudiantes" se ejemplifica con la ocurrencia "Viveros", que no se asocia con ninguno de los cursos ofrecidos.

Una vez construido el diagrama E-R, el siguiente paso es obtener las relaciones (estructuras) preliminares necesarias para contener los datos. Esta etapa se revisa en el siguiente capítulo.

3. DISEÑO DEL MODELO LOGICO

3. DISEÑO DEL MODELO LOGICO

Una vez definido el modelo conceptual de la base de datos, la siguiente etapa consiste en decidir las relaciones necesarias para contener los datos. Para C. J. Date, éste es el problema del diseño lógico de bases de datos[1].

El diseño lógico de bases de datos relacionales tiene como objetivo obtener relaciones normalizadas a partir de diagramas E-R. Paul Beynon define el modelo lógico de una base de datos como "un modelo del mundo real expresado en términos de relaciones, dominios, llaves primarias y llaves foráneas"[2].

Una vez construido un diagrama E-R, el siguiente paso es obtener las relaciones preliminares necesarias para contener los datos, aplicando un conjunto de reglas expuestas en este capítulo. Las relaciones obtenidas deben someterse a un proceso de normalización, que será presentado más adelante.

3.1 Deducción de relaciones preliminares a partir de diagramas E-R

El primer paso del diseño lógico es deducir relaciones preliminares necesarias para contener los datos, a partir de diagramas E-R. Es importante aclarar que en esta etapa el

---

1 Date, C.J. *An Introduction To Database Systems*. The Systems Programming Series: Addison Wesley. Estados Unidos de Norteamérica. 4ª Edición, 1988. p. 361.

2 Beynon-Davies, Paul. *Relational Database Design*. Blackwell Scientific Publications. Gran Bretaña. 1992. p. 6.

término "relación" se refiere a la estructura del modelo relacional, es decir tablas, mientras que el término asociación se usará para referirse al número de entidades involucradas en la relación.

3.1.1 Relaciones preliminares para asociaciones binarias uno a uno

Relaciones 1:1 con participación obligatoria de ambas entidades.

Supóngase que se desea deducir las relaciones necesarias para contener los datos del siguiente diagrama E-R.

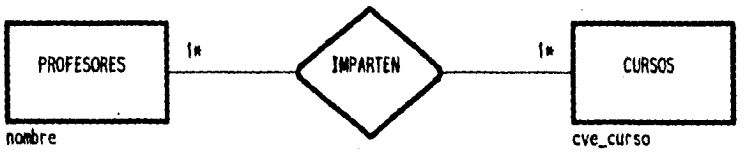

Figura 3.1 Diagrama E-R que representa la asociación 1:1 entre las entidades "PROFESORES" y "CURSOS".

Los atributos de "PROFESORES" son: nombre, domicilio y teléfono; mientras que los de "CURSOS" son: clave y nombre del curso. Para este ejemplo se eligió como llave de la entidad "PROFESORES" el atributo "nombre", y para la entidad "CURSOS", la clave del curso, "cve_curso". El diagrama de ocurrencias para este ejemplo se muestra en la figura 3.2.

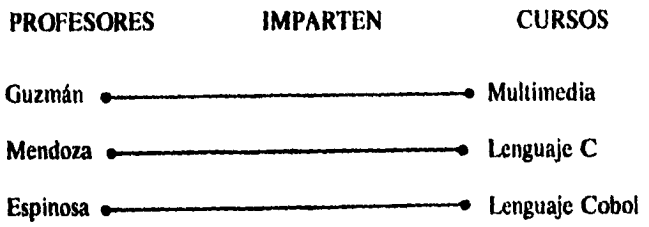

Figura 3.2 Diagrama de ocurrencias.

Para obtener las relaciones preliminares a partir del diagrama E-R de la figura 3.1, se aplica la siguiente regla:

REGLA 1. Cuando el grado de una asociación binaria es 1:1 y la participación de ambas entidades es obligatoria, sólo una relación es necesaria. La llave primaria de esta relación puede ser la llave de cualquiera de las entidades.

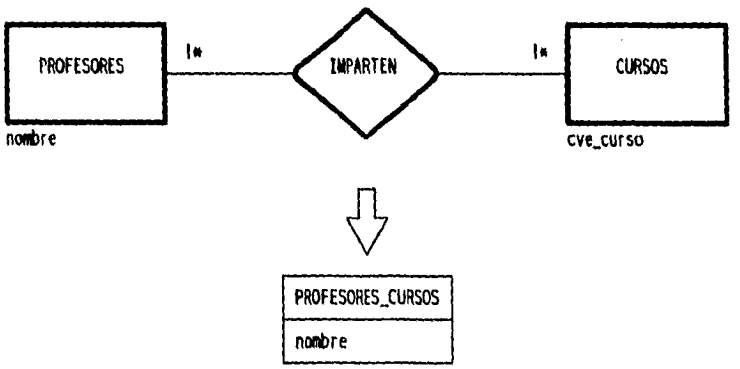

Figura 3.3 Derivación de relaciones de una asociación 1:1 con ambas entidades obligatorias, aplicando la regla número uno.

El nombre de la relación resultante puede tomarse de alguna de las entidades o hacer una combinación de ellos, como se hizo en este ejemplo. Como llave primaria de la relación resultante se eligió la llave de la entidad de "PROFESORES" (figura 3.3). En la figura 3.4 se presenta un ejemplo de la derivación de relaciones aplicando la regla número uno.

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

CURSOS

cve_curso	curso
MM	Multimedia
LC	Lenguaje C
LCBL	Lenguaje COBOL

PROFESORES_CURSOS

nombre	domicilio	teléfono	cve_curso	curso
Guzmán	Oaxaca 72	6-15-12-89	MM	Multimedia
Mendoza	Cuauhtémoc 185	5-79-45-90	LC	Lenguaje C
Espinosa	Ciruelos 23	3-93-78-45	LCBL	Lenguaje COBOL

Figura 3.4 Aplicando la regla número uno al ejemplo citado, se obtendría solamente una relación.

Relaciones 1:1 con participación de una entidad obligatoria y otra opcional

Considérese nuevamente el diagrama E-R de la figura 3.1, sólo que ahora la participación de "PROFESORES" es obligatoria y "CURSOS" es opcional (figura 3.5). El diagrama de ocurrencias para este ejemplo se presenta en la figura 3.6.

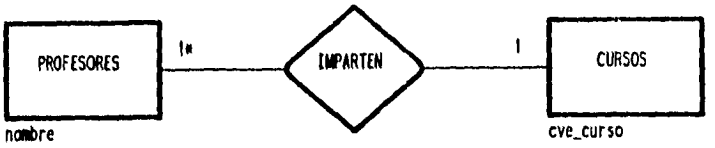

Figura 3.5 Relación 1:1 con participación obligatoria de "PROFESORES" y opcional de "CURSOS".

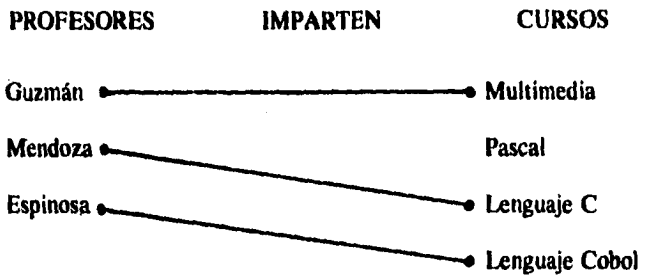

Figura 3.6 Diagrama de ocurrencias.

Si el grado de una relación entre dos entidades es 1:1 y la participación de una entidad es obligatoria y la otra es opcional, una relación no es suficiente para contener los datos, pues se presentarían valores nulos por cada instancia de una entidad que no está asociada a otra; por ejemplo, por cada curso que no es impartido por alguno de los profesores (figura 3.7). Para solucionar este problema, se aplicaría la regla número dos.

PROFESORES_CURSOS

nombre	domicilio	teléfono	cve_curso	curso
Guzmán	Oaxaca 72	6-15-12-89	MM	Multimedia
Mendoza	Cuauhtémoc 185	5-79-45-90	LC	Lenguaje C
Espinosa	Ciruelos 23	3-93-78-45	LCBL	Lenguaje COBOL
--------	--------	--------	LPAS	Lenguaje Pascal

Figura 3.7

REGLA 2. Cuando el grado de una relación binaria es 1:1 con la participación de una entidad obligatoria y la otra opcional, dos relaciones son necesarias. Debe haber una relación por cada entidad, con la llave de entidad como llave primaria para la correspondiente relación. Adicionalmente, la llave de la entidad con participación opcional debe agregarse como llave foránea en la relación de la parte obligatoria.

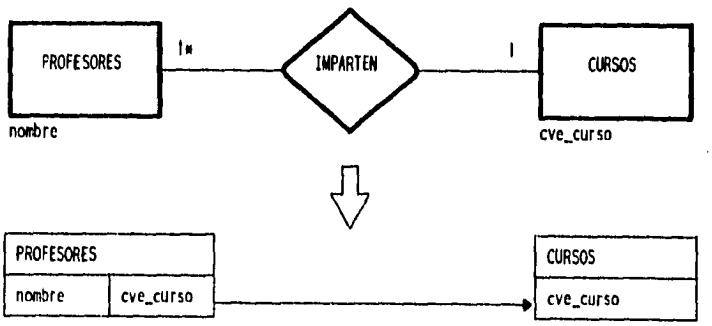

Figura 3.8

Aplicando esta regla al diagrama de la figura 3.5, las relaciones resultantes serían "PROFESORES" y "CURSOS", cada una teniendo como llave primaria la llave de la entidad de la cual se derivan. Adicionalmente "PROFESORES" tiene como llave foránea a "cve_curso" (figura 3.8). Un ejemplo de la derivación de relaciones aplicando la Regla 2 se presenta en la figura 3.9.

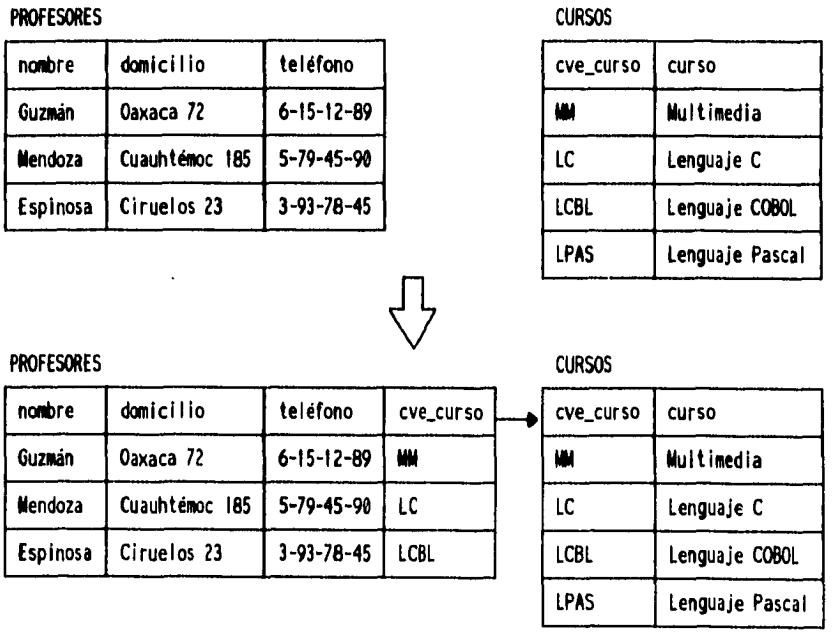

Figura 3.9 Instancias de la derivación de relaciones aplicando la regla número dos.

Relaciones 1:1 con participación opcional de ambas entidades

Considérese nuevamente el ejemplo de "PROFESORES" y "CURSOS", pero ahora ambas entidades participan de manera opcional en la relación (figura 3.10). En el caso de una relación binaria 1:1 con la participación opcional de ambas entidades, una relación no es suficiente. Si se utilizara una sola relación, se tendrían valores nulos por cada curso que no fuera impartido por alguno de los profesores y por cada profesor que no impartiera alguno de los cursos (figura 3.12a). Usar dos relaciones también da lugar a valores nulos por cada profesor que no imparta algún curso (figura 3.12b).

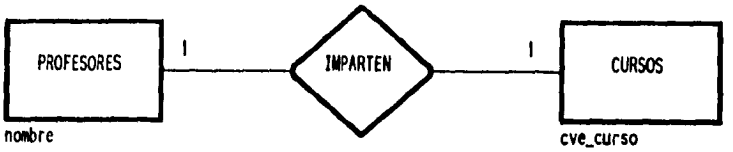

Figura 3.10

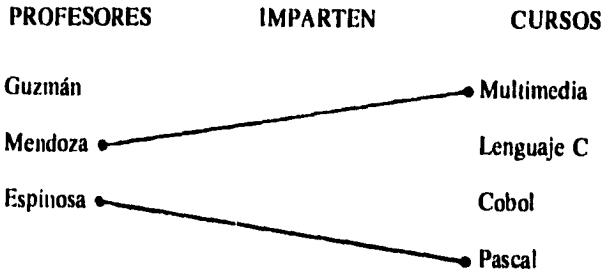

Figura 3.11 Diagrama de ocurrencias.

PROFESORES

nombre	domicilio	teléfono	cve_curso	curso
Guzmán	Oaxaca 72	6-15-12-89	-------	----------
Mendoza	Cuauhtémoc 185	5-79-45-90	MM	Multimedia
Espinosa	Ciruelos 23	3-93-78-45	PAS	Pascal
--------	--------	-------	LC	Lenguaje C
--------	--------	-------	CBL	Cobol

(A)

PROFESORES

nombre	domicilio	teléfono	cve_curso
Guzmán	Oaxaca 72	6-15-12-89	-------
Mendoza	Cuauhtémoc 185	5-79-45-90	MM
Espinosa	Ciruelos 23	3-93-78-45	PAS

CURSOS

cve_curso	curso
MM	Multimedia
LC	Lenguaje C
CBL	Cobol
PAS	Pascal

(B)

Figura 3.12

Para resolver este problema se aplica la siguiente regla:

REGLA 3. Cuando el grado de una relación entre dos entidades es 1:1 con participación opcional de ambas entidades, tres relaciones son necesarias: una por cada entidad, con la llave de entidad como llave primaria para la relación correspondiente, y una relación más, para la asociación, que tendrá entre sus atributos las llaves de cada entidad como llaves foráneas.

Aplicando esta regla al diagrama E-R de la figura 3.10, las relaciones resultantes serían "PROFESORES" y "CURSOS". La relación "IMPARTEN" tendría como atributos (llaves foráneas), las llaves de entidad "nombre" y "cve_curso" de las entidades "PROFESORES" y "CURSOS" respectivamente (figura 3.13). Un ejemplo de las instancias de las tres relaciones resultantes se presenta en la figura 3.14.

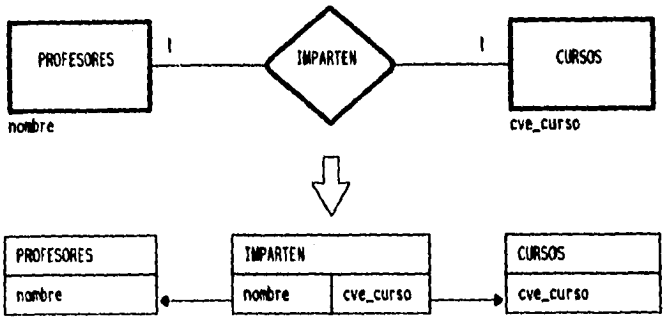

Figura 3.13 Deducción de relaciones a partir de una asociación binaria 1:1, donde ambas entidades participan de manera opcional.

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Cirueios 23	3-93-78-45

CURSOS

cve_curso	curso
MM	Multimedia
LC	Lenguaje C
CBL	Cobol
PAS	Pacal

IMPARTEN

nombre	cve_curso
Mendoza	MM
Espinosa	PAS

Figura 3.14

3.1.2 Relaciones preliminares para asociaciones binarias uno a muchos

Relaciones uno a muchos con participación obligatoria del lado "N"

En el caso de que la relación entre las entidades "PROFESORES" y "CURSOS" sea 1:N y que la participación de "CURSOS" sea obligatoria, los diagramas E-R y de ocurrencias serían los siguientes:

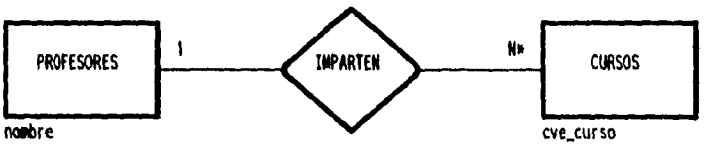

Figura 3.15 Asociación 1:N entre "PROFESORES" y "CURSOS". La participación de la entidad "CURSOS" es obligatoria.

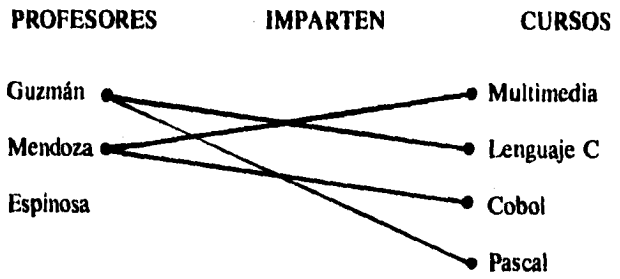

Figura 3.16 Diagrama de ocurrencias.

Bajo esta situación, utilizar una sola relación para contener los datos, daría lugar a la aparición de valores nulos en los atributos de la entidad "CURSOS" por cada profesor que no imparta algún curso. Por otra parte, habrá redundancia de datos en el caso de los profesores que imparten más de un curso. En la figura 3.17 se ejemplifican los problemas mencionados cuando se utiliza una sola relación. Se observa que hay valores nulos en los renglones que corresponden a los profesores que no dan alguno de los cursos. Además, los datos de algunos profesores se repiten por cada curso que imparten.

PROFESORES

nombre	domicilio	teléfono	cve_curso	curso
Guzmán	Oaxaca 72	6-15-12-89	LC	Lenguaje C
Guzmán	Oaxaca 72	6-15-12-89	LPAS	Lenguaje Pascal
Mendoza	Cuauhtémoc 185	5-79-45-90	MM	Multimedia
Mendoza	Cuauhtémoc 185	5-79-45-90	LCBL	Lenguaje COBOL
Espinosa	Ciruelos 23	3-93-78-45	---------	----------

Figura 3.17 En una asociación binaria 1:N, donde la participación del lado "N" es obligatoria, utilizar una relación genera dos problemas: redundancia de datos y valores nulos en la relación.

Para resolver esta situación puede emplearse la siguiente regla:

REGLA 4. Cuando el grado de una relación binaria es 1:N y la participación del lado N es obligatoria, dos relaciones son necesarias : una por cada entidad, con la llave de cada una

de ellas como llave primaria para la relación correspondiente. Adicionalmente, la llave de entidad de la parte opcional debe agregarse como llave foránea en la relación que corresponde a la parte obligatoria. La participación del lado "1" es indistinta.

La deducción de relaciones a partir del diagrama E-R de la figura 3.15, sería como se muestra en la figura 3.18. Se observa que la relación "CURSOS" tiene, de manera adicional, el atributo "nombre" como llave foránea. En la figura 3.19 se muestran ocurrencias de las relaciones resultantes.

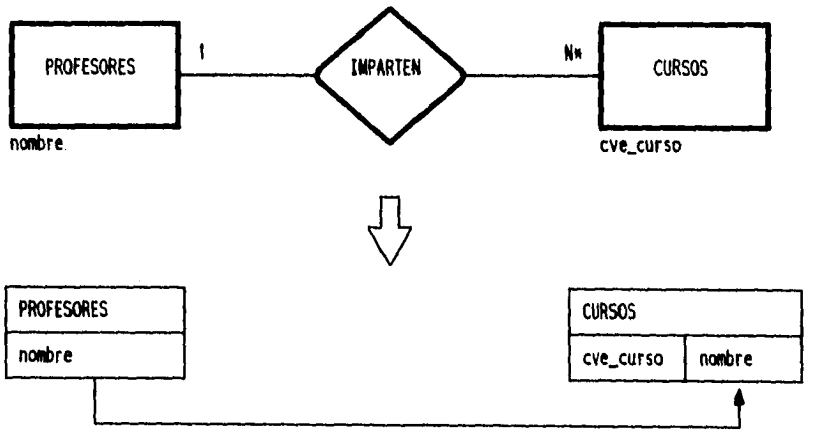

Figura 3.18 Deducción de relaciones.

Relaciones uno a muchos con participación opcional del lado "N"

Considérese que en el ejemplo de "PROFESORES" y "CURSOS" ambas entidades participan opcionalmente en una relación 1:N (figura 3.20).

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

CURSOS

cve_curso	curso	nombre
MM	Multimedia	Mendoza
LC	Lenguaje C	Guzmán
LCBL	Lenguaje COBOL	Mendoza
LPAS	Lenguaje Pascal	Guzmán

Figura 3.19 Ocurrencias de las relaciones resultantes al aplicar la Regla 4 al diagrama E-R de la figura 3.15.

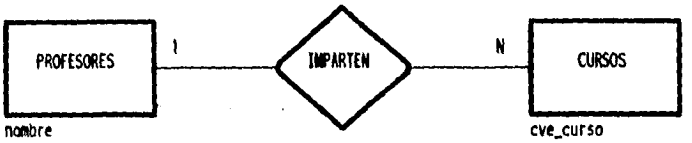

Figura 3.20 Asociación 1:N entre "PROFESORES" y "CURSOS". Ambas entidades participan opcionalmente.

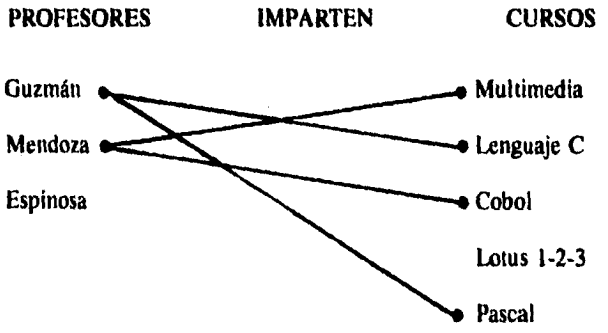

Figura 3.21 Diagrama de ocurrencias.

Si se utilizara una sola relación para contener los datos de este ejemplo, se presentarían valores nulos por cada curso que no es impartido por ninguno de los profesores, y por cada profesor que no imparte algún curso (figura 3.22a). Dos relaciones tampoco es recomendable, pues continuarían apareciendo valores nulos (figura 3.22b).

PROFESORES

nombre	domicilio	teléfono	cve_curso	curso
Guzmán	Oaxaca 72	6-15-12-89	LC	Lenguaje C
Guzmán	Oaxaca 72	6-15-12-89	PAS	Pascal
Mendoza	Cuauhtémoc 185	5-79-45-90	MM	Multimedia
Mendoza	Cuauhtémoc 185	5-79-45-90	CBL	Cobol
Espinosa	Ciruelos 23	3-93-78-45	-------	---------
--------	---------	-------	123	Lotus 1-2-3

(A)

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

CURSOS

cve_curso	curso	nombre
MM	Multimedia	Mendoza
LC	Lenguaje C	Guzmán
CBL	Cobol	Mendoza
PAS	Pascal	Guzmán
123	Lotus 1-2-3	-------

(B)

Figura 3.22 Utilizar una (a) o dos relaciones (b) para contener los datos de una asociación binaria 1:N, donde ambas entidades participan opcionalmente, da lugar a valores nulos.

Para solucionar este problema se aplica la Regla 5.

REGLA 5. Cuando el grado de una relación binaria es 1:N y la participación del lado N es opcional, tres relaciones son necesarias: una por cada entidad, con la llave de cada entidad como llave primaria para la relación correspondiente, y una relación más para la asociación, la cual tendrá entre sus atributos como llaves foráneas, las llaves de cada entidad. La participación del lado "1" es indistinta.

Al aplicar esta regla al diagrama de la figura 3.20, se obtienen tres relaciones: "PROFESORES", "CURSOS" e "IMPARTEN". Esta última tiene los atributos "nombre" y "cve_curso" como llaves foráneas, que forman la llave primaria de la relación (figura 3.23). Un ejemplo de las relaciones resultantes se presenta en la figura 3.24.

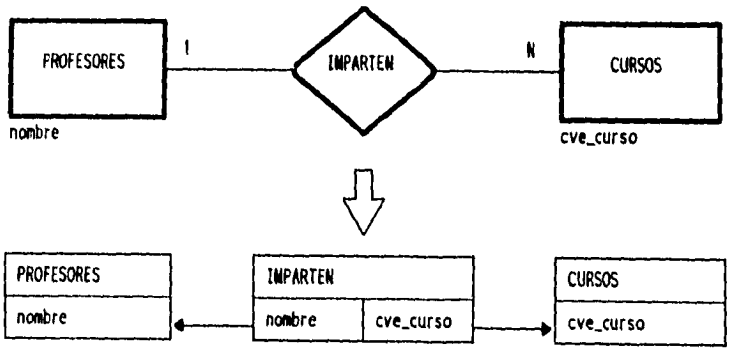

Figura 3.23 Deducción de relaciones a partir del diagrama de la figura 3.20, aplicando la Regla 5.

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

CURSOS

cve_curso	curso	nombre
MM	Multimedia	Mendoza
LC	Lenguaje C	Guzmán
LCBL	Lenguaje COBOL	Mendoza
LPAS	Lenguaje Pascal	Guzmán

Figura 3.24 Ocurrencias de las relaciones resultantes de aplicar la regla 5 al diagrama de la figura 3.20

3.1.3 Relaciones preliminares para asociaciones binarias muchos a muchos

En el diagrama E-R de la figura 3.25, se indica que varios profesores pueden dar varios cursos. Se trata de asociación M:N, donde ambas entidades participan opcionalmente.

Figura 3.25

Al utilizar solamente una relación para contener los datos del ejemplo en cuestión, se presentarían dos problemas: la redundancia de datos y valores nulos en la relación. En la figura 3.27 se observa que los datos de cada profesor se repiten por cada curso que imparte, igualmente, los datos sobre un curso se repiten por cada profesor que los imparte. Por otro lado,

se observa también que hay valores nulos por cada profesor que no imparte algún curso y por cada curso que no es impartido por alguno de los profesores. Si se utilizaran dos relaciones, se eliminaría la redundancia de datos, pero aún aparecerían valores nulos (figura 3.28).

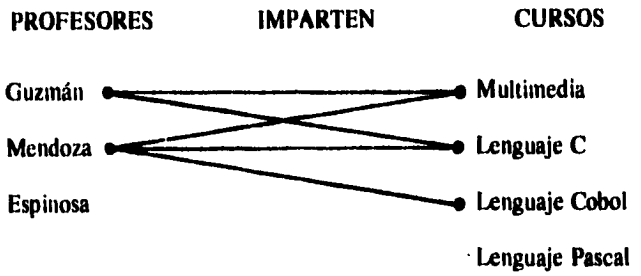

Figura 3.26 Diagrama de ocurrencias.

PROFESORES

nombre	domicilio	teléfono	cve_curso	curso
Guzmán	Oaxaca 72	6-15-12-89	MM	Multimedia
Guzmán	Oaxaca 72	6-15-12-89	LC	Lenguaje C
Mendoza	Cuauhtémoc 185	5-79-45-90	MM	Multimedia
Mendoza	Cuauhtémoc 185	5-79-45-90	LCBL	Lenguaje C
Mendoza	Cuauhtémoc 185	5-79-45-90	LCBL	Lenguaje COBOL
Espinosa	Ciruelos 23	3-93-78-45	--------	--------
--------	--------	--------	LPAS	Lenguaje Pascal

Figura 3.27 Al utilizar una sola relación para un problema de asociación M:N, se presentan dos problemas: la redundancia de datos y valores nulos en la relación.

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

CURSOS

cve_curso	curso	nombre
MM	Multimedia	Guzmán
MM	Multimedia	Mendoza
LC	Lenguaje C	Guzmán
LC	Lenguaje C	Mendoza
LCBL	Lenguaje COBOL	Mendoza
LPAS	Lenguaje Pascal	--------

Figura 3.28 Utilizar dos relaciones para un problema de asociación M:N, elimina la redundancia de datos, pero no los valores nulos.

Cuando el grado de una relación binaria es M:N, se requieren tres relaciones para almacenar los datos, sin importar el tipo de participación de las entidades. La regla para deducir relaciones a partir de asociaciones M:N es la siguiente:

REGLA 6. Cuando el grado de una relación binaria es M:N, se requieren tres relaciones para contener los datos: una por cada entidad, con la llave de cada entidad como llave primaria para la relación correspondiente, y una relación más para la asociación. La relación de asociación tendrá entre sus atributos las llaves de cada entidad. La participación de las entidades es indistinta.

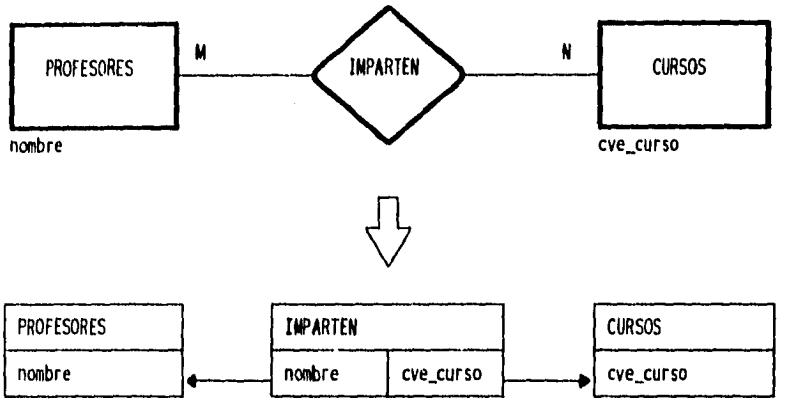

Figura 3.29 Deducción de relaciones a partir del diagrama E-R de la figura 3.25, aplicando la Regla 6.

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtemoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

CURSOS

cve_curso	curso
MM	Multimedia
LC	Lenguaje C
LCBL	Lenguaje COBOL
LPAS	Lenguaje Pascal

IMPARTEN

nombre	cve_curso
Guzmán	MM
Guzmán	LC
Mendoza	MM
Mendoza	LC
Mendoza	LCBL

Figura 3.30 Ocurrencias de las relaciones resultantes de aplicar la Regla 6 al diagrama de la figura 3.25.

3.1.4 Relaciones preliminares para asociaciones *n*-arias

Considérese nuevamente el ejemplo de "PROFESORES" y "CURSOS". Supóngase ahora que los cursos son ofrecidos por las "FACULTADES" de una universidad. Los profesores imparten diferentes cursos en diversas facultades. Las facultades ofrecen diversos cursos.

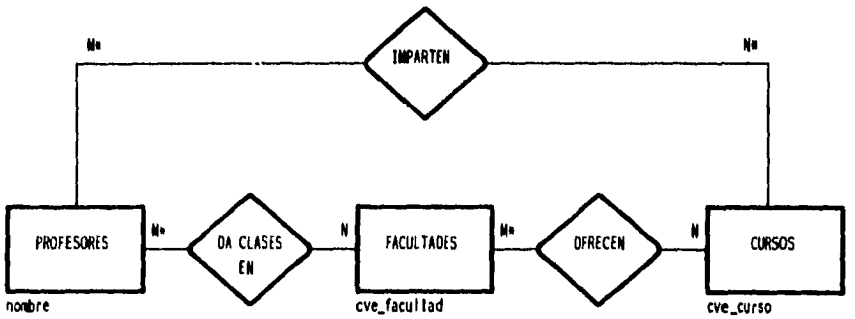

Figura 3.31 El grado de asociación entre "PROFESORES", "FACULTADES" y "CURSOS" es M:N.

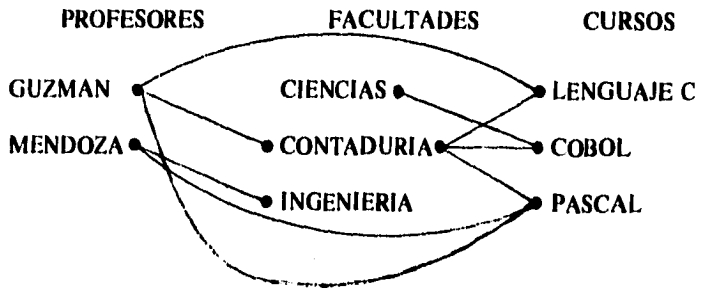

Figura 3.32 Diagrama de ocurrencias.

En el diagrama de ocurrencias anterior, se ejemplifica que la Facultad de Ciencias ofrece cursos de COBOL; la Facultad de Contaduría, de Lenguaje C, COBOL y Pascal; por su parte, la Facultad de Ingeniería imparte cursos de Pascal. En cuanto a los profesores, Guzmán da clases de Lenguaje C en la Facultad de Contaduría, y además da clases de Pascal en Ingeniería. Por otro lado, también Mendoza da clases de Pascal en Ingeniería.

Cabe destacar que aunque los casos mencionados pueden deducirse del diagrama de ocurrencias, puede afirmarse también que Guzmán imparte cursos de Pascal en la Facultad de Contaduría, lo cual puede no ser necesariamente cierto. Una relación entre tres entidades (ternaria) no puede modelarse utilizando relaciones binarias (diagramas 3.31 y 3.32), pues puede dar origen a confusiones. En las figuras 3.33 y 3.34 se muestra cómo modelar esta situación de un modo conveniente.

Cuando se presentan casos de asociaciones ternarias, las relaciones preliminares pueden deducirse utilizando la siguiente regla:

REGLA 7. Cuando tres entidades participan en una misma relación, cuatro relaciones preliminares son requeridas: una por cada entidad, con la llave de cada entidad como llave primaria para la relación correspondiente y una relación más para la asociación, la cual tendrá entre sus atributos las llaves de cada una de las entidades. Del mismo modo, cuando una relación es n-aria, se requieren $n+1$ relaciones preliminares para contener los datos. La participación de las entidades es indistinta.

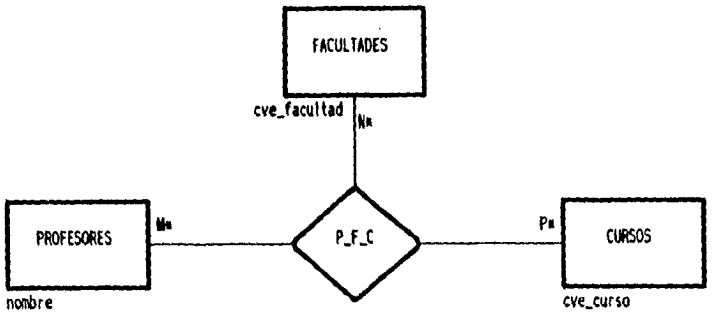

Figura 3.33 Diagrama E-R para la relación ternaria entre las entidades "PROFESORES", FACULTADES" y "CURSOS".

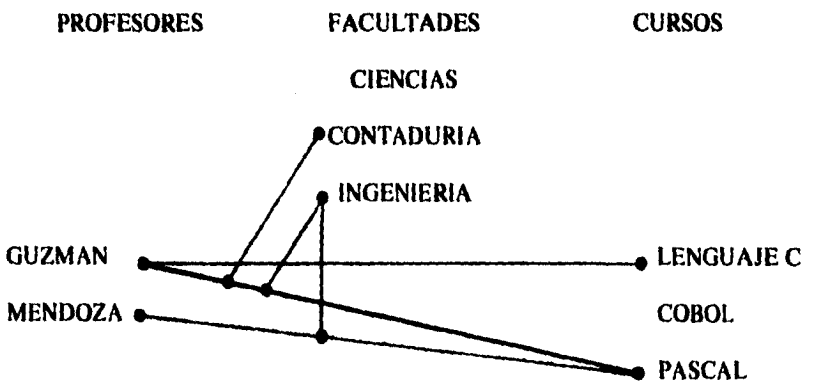

Figura 3.34 Diagrama de ocurrencias para la asociación entre "PROFESORES", "FACULTADES" y "CURSOS".

Aplicando esta regla al ejemplo presentado anteriormente, la conversión a relaciones preliminares sería como se muestra en la figura 3.35.

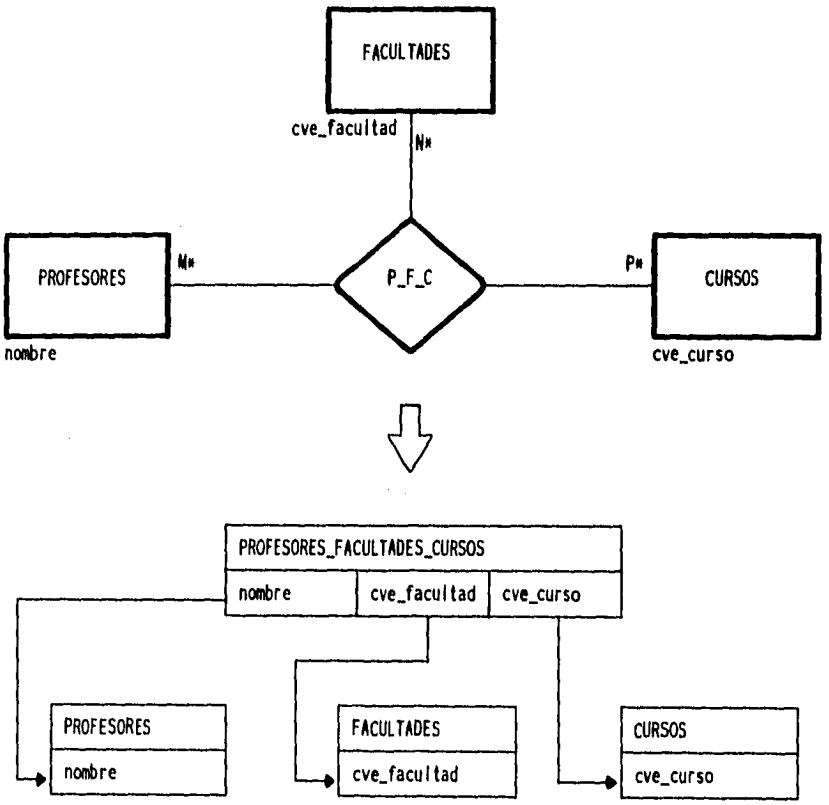

Figura 3.35

La llave primaria para P_F_C sería <nombre,cve_facultad>, si cada profesor impartiera un solo curso. Si cada profesor diera más de uno, entonces la llave sería <nombre,cve_curso,cve_facultad>. En la figura 3.36 se muestran ocurrencias de las relaciones resultantes de aplicar la Regla 7 al ejemplo expuesto.

PROFESORES

nombre	domicilio	teléfono
Guzmán	Oaxaca 72	6-15-12-89
Mendoza	Cuauhtémoc 185	5-79-45-90
Espinosa	Ciruelos 23	3-93-78-45

FACULTADES

cve_fac	nombre
FC	Facultad de Ciencias
FCA	Facultad de Contaduría y Administración
FI	Facultad de Ingeniería

CURSOS

cve_curso	curso
MM	Multimedia
LC	Lenguaje C
LCBL	Lenguaje COBOL
LPAS	Lenguaje Pascal
123	Lotus 1-2-3

IMPARTEN

nombre	cve_curso	cve_fac
Guzmán	LC	FCA
Guzmán	LPAS	FI
Mendoza	LPAS	FCA
Espinosa	LCBL	FC

Figura 3.36

Es importante señalar que en las bases de datos relacionales, la redundancia no se elimina completamente, pues siempre habrá repetición de datos cuando se trata de llaves primarias o foráneas.

3.2 Normalización de relaciones

Una base de datos relacional involucra cierta redundancia; sin embargo, ésta debe mantenerse al mínimo. Idealmente, un atributo de una entidad debe aparecer en un solo lugar, y solamente debe repetirse si es absolutamente necesario. La normalización es un proceso que forma parte integral del diseño lógico de bases de datos que se utiliza para asegurar precisamente un mínimo posible de redundancia.

El proceso de normalización se basa en el concepto de formas normales. Una relación está en una particular forma normal si satisface un conjunto específico de condiciones. Varias formas normales han sido definidas: Codd en 1972 establece la primera, segunda y tercera forma normal (1FN, 2FN, 3FN). Más adelante, en 1974, Codd y R. F. Boyce introdujeron una nueva versión de la tercera forma normal, conocida como "Forma Normal de Boyce y Codd" (FNBC). Después, en 1976, Fagin define la cuarta forma normal (4FN) y luego en 1979 la quinta forma normal (5FN).

Aun cuando existen seis formas normales, la normalización más allá de 3FN raramente se practica. Date comenta que la quinta forma normal "es un caso patológico y raro en la práctica"[3]. Aunque FNBC y 4FN son menos extraños, son aún de una naturaleza teórica. En esta investigación solamente se cubren la 1FN, 2FN, 3FN y FNBC.

---

[3] Date, C. J. *An Introduction To Database Systems*. p. 390.

3.2.1 Primera forma normal (1FN)

Una relación está en la primera forma normal, si cumple las siguientes condiciones:

1. Tiene una llave primaria, compuesta por uno o más atributos, que identifica de modo único una ocurrencia en la relación, y

2. Cada atributo en la relación sólo acepta un valor simple como valor del atributo.

El concepto de llave primaria es de fundamental importancia en el modelo relacional de bases de datos, ya que el único mecanismo de direccionamiento de tuplas o renglones es la combinación del nombre de la tabla o relación y el valor de la llave primaria.

El primer paso para definir qué atributos formarán una llave primaria es identificar llaves candidatas. Una llave candidata es uno o más atributos que pueden actuar como identificadores únicos de una tupla. La llave primaria se elige de varias llaves candidatas. Una llave candidata es, consecuentemente, una llave primaria.

Como segundo paso, se selecciona la llave candidata que será la llave primaria. Si hay varias llaves candidatas, deberá elegirse sólo una como llave primaria y considerar el resto como llaves alternas. Por ejemplo, para la relación "EMPLEADOS", se tienen como llaves candidatas el número de empleado y el número de seguro social, pues cada uno puede identificar uno y sólo un empleado. En este caso es mejor utilizar el número de empleado, pues es preferible

optar por aquellos atributos de longitud corta, toda vez que las llaves primarias generalmente aparecen más de una vez en una base de datos, lo que implica mayor ocupación de espacio de almacenamiento. Al elegir el número de empleado como llave primaria, el número de seguro social se convierte en llave alterna.

Quizá una de las mejores maneras de formar una llave primaria sea seleccionando atributos que acepten números consecutivos como valor de atributo. El atributo "número de empleado" en la relación "EMPLEADOS" es un número único dentro de una serie, que actúa simplemente como referencia.

Al formar llaves primarias deben considerarse los siguientes puntos:

1. El valor de una llave primaria debe ser único para cada tupla en una relación.

2. Los atributos que forman una llave primaria no deben aceptar valores nulos.

3. Debe existir una y sólo una llave primaria para una y sólo una ocurrencia de una entidad; es decir, no deben existir diferentes llaves primarias para un mismo objeto.

4. Si las características de las ocurrencias de una entidad cambian con frecuencia, la llave primaria debe evitar contener detalles de esas propiedades. Por ejemplo, considérese que se define que el número de empleado estará formado por cuatro dígitos, donde el primero

indicará el área de adscripción del empleado. Si un empleado cambiara de departamento, entonces tendría que cambiarse su número de empleado en todas las relaciones donde aparezca, a fin de conservar la integridad de la información de ese empleado. Esta manera de diseñar llaves primarias sólo funciona cuando las características de los objetos representados son estables o casi nunca cambian.

La segunda condición de la 1FN es que los valores de los atributos deben ser simples. Véase la relación "PROFESORES" en la figura 3.37. Esta relación no está en 1FN, ya que el atributo "idiomas" (idiomas que un profesor domina) acepta más de un valor, como es el caso del Profesor Arroyo, cuyo atributo idioma contiene "Inglés, Francés".

PROFESORES

nombre	domicilio	teléfono	idioma
Guillén	Vistaluz 134	6-66-67-63	Inglés
Arroyo	Olivo 28	5-56-03-88	Inglés, Francés
Martínez	Legaria 1243	7-96-05-90	Francés
Cuevas	Cecilia 25	8-36-40-13	Inglés, Alemán

Figura 3.37 Relación con atributos que contienen valores múltiples.

La solución a este problema es descomponer la relación en dos, creando una nueva. En este caso es necesario considerar una relación 1:N entre profesores e idiomas y crear una nueva

relación "IDIOMAS". De esta manera la relación "PROFESORES" puede llevarse a 1FN, como se muestra en la figura 3.38.

PROFESORES

nombre	domicilio	teléfono
Guillén	Vistaluz 134	6-66-67-63
Arroyo	Olivo 28	5-56-03-88
Martínez	Legaria 1243	7-96-05-90
Cuevas	Cecilia 25	8-36-40-13

IDIOMAS

nombre	idioma
Guillén	Inglés
Arroyo	Inglés
Arroyo	Francés
Martínez	Francés
Cuevas	Inglés
Cuevas	Alemán

Figura 3.38 Descomposición de la relación "PROFESORES" creando la relación "IDIOMAS".

3.2.2 Segunda forma normal (2FN)

La relación "VUELOS" de la figura 3.39 está en 1FN; tiene una llave primaria compuesta, formada por los atributos "cve_línea_aérea" y "#vuelo", además los atributos contienen valores simples; sin embargo, los nombres de las líneas aéreas se repiten de manera redundante. Una relación en 1FN puede presentar problemas de redundancia, por lo que es necesario transformarla a 2FN.

VUELOS

cve_línea_aérea	#vuelo	línea_aérea	destino
AMX	170	Aeroméxico	Tijuana
MXA	509	Mexicana	Acapulco
AMX	160	Aeroméxico	Guadalajara
MXA	517	Mexicana	Tijuana

Figura 3.39

Date señala que "Una relación *R* está en segunda forma normal (2FN) si y sólo si está en 1FN y cada atributo no-llave es totalmente dependiente de la llave primaria."[4]. Un atributo no-llave es aquel que no forma parte de la llave primaria.

La dependencia entre atributos puede entenderse de la siguiente forma: Dada una relación *R*, un atributo *y* depende de otro *x* si y sólo si cada valor de *x* tiene asociado exactamente un valor de *y*; dicho de otro modo, *x* determina funcionalmente a *y*. Por ejemplo, un número de empleado determina funcionalmente un nombre de un empleado, su domicilio, teléfono, etc., lo cual puede representarse como a continuación se indica:

num_emp --> nombre_emp, domicilio_emp, departamento, ...

---

[4] Date, C. J. *An Introduction To Database Systems*. p. 370.

La expresión "totalmente dependiente" utilizada para definir la 2FN se refiere a que si la llave primaria es compuesta (formada por más de un atributo), entonces los atributos no-llave deben depender de la llave primaria entera, es decir, de todas sus partes. Si un atributo no-llave depende parcialmente de la llave primaria, entonces la relación no está en 2FN y es necesario remover ese atributo de la relación.

Las dependencias funcionales a que hace referencia la segunda forma normal son entre atributos no-llave y llaves primarias compuestas, por lo que puede definirse también que una relación está en 2FN si su llave primaria está formada por un solo atributo o si no contiene atributos no-llave.

En el ejemplo de la figura 3.39 la redundancia se debe a que el atributo "línea aérea" depende parcialmente de la llave primaria, específicamente del atributo "cve_línea_aérea". La dependencia que existe entre los atributos de la relación "VUELOS" puede describirse de la siguiente manera:

cve_línea_aérea, #vuelo --> destino

cve_línea_aérea --> línea_aérea

Para transformar la relación "VUELOS" a 2FN y eliminar la redundancia existente, es necesario crear una nueva relación "LINEAS_AEREAS", que incluya el atributo "línea_aérea" y que tenga como llave primaria a "cve_línea_aérea" (figura 3.40).

VUELOS

cve_línea_aérea	#vuelo	destino
AMX	170	Tijuana
MXA	509	Acapulco
AMX	160	Guadalajara
MXA	517	Tijuana

LINEAS_AEREAS

cve_línea_aérea	línea_aérea
AMX	Aeroméxico
MXA	Mexicana

Figura 3.40

En resumen, una relación está en 2FN si está en 1FN y cumple alguna de las siguientes condiciones:

1. La llave primaria de la relación consta de un solo atributo.

2. No existen atributos no-llave en la relación.

3. Todos los atributos no-llave dependen totalmente de la llave primaria, cuando es compuesta.

3.2.3 Tercera forma normal (3FN)

Algunas relaciones en 2FN son aún susceptibles de contener datos redundantes. Por ejemplo, la relación "VUELOS" en la figura 3.41 está en 2FN, pues cumple con la condición

de que su llave primaria es un solo atributo, que en este caso es el número de vuelo; sin embargo la tarifas se repiten de manera redundante. La tarifa a un destino aparece más de una vez porque existe más de un vuelo a un mismo destino, como es el caso de Acapulco.

VUELOS

#vuelo	destino	hr_salida	hr_arribo	tarifa
170	Tijuana	8:00	10:17	930
509	Acapulco	8:20	8:55	210
160	Guadalajara	20:04	21:09	360
517	Acapulco	20:10	20:45	210

Figura 3.41

Date define la tercera forma normal de la siguiente manera: "Una relación *R* está en tercera forma normal (3FN) si y sólo si está en 2FN y cada atributo no-llave no depende transitivamente de la llave primaria."[5]. La dependencia transitiva es aquella que existe entre atributos no-llave y que no involucran a la llave primaria.

La definición anterior significa que para que una relación esté en 3FN debe estar en 2FN y no debe existir dependencia entre sus atributos no-llave. Para que una relación esté en 3FN ninguno de sus atributos no-llave debe depender de otro atributo no-llave. Si el valor de un atributo no-llave puede obtenerse conociendo simplemente el valor de otro atributo no-llave, la relación no está en 3FN.

---

[5] Date, C. J. *An Introduction To Database Systems*, p. 373.

En el ejemplo de la figura 3.41, el problema radica en que las tarifas dependen del destino; tanto "destino" como "tarifa" son atributos no-llave. Las dependencias entre los atributos de la relación "VUELOS" es como a continuación se indica:

#vuelo --> destino, hr_salida, hr_arribo

destino --> tarifa

Por lo anterior, es necesario dividir la relación "VUELOS" en dos, creando una nueva relación "TARIFAS" que incluya las tarifas a cada destino y que tenga como llave primaria el atributo "destino" (figura 3.42). De esta manera se obtiene la 3FN de este ejemplo.

Dado que el énfasis de la 3FN está en que no debe existir dependencia entre atributos no-llave, puede identificarse en primera instancia que, una relación está 3FN cuando tiene solamente un atributo no-llave, o no existen atributos no-llave; es decir, todos los atributos forman la llave de la relación. Por supuesto, debe cumplirse también con la 2FN.

VUELOS

#vuelo	destino	hr_salida	hr_arribo
170	Tijuana	8:00	10:17
509	Acapulco	8:20	8:55
160	Guadalajara	20:04	21:09
517	Acapulco	20:10	20:45

TARIFAS

destino	tarifa
Tijuana	930
Acapulco	210
Guadalajara	360

Figura 3.42

3.2.4 Forma normal de Boyce y Codd (FNBC)

Las relaciones en tercera forma normal presentan algunas deficiencias, en el caso de que una tuvieran múltiples llaves candidatas, las cuales fueran compuestas y que tuvieran al menos un atributo en común. Por ejemplo, la relación "CURSOS" de la figura 3.43 contiene una programación de cursos de computación, que incluye el semestre, nombre del curso, nivel del curso, el nombre del profesor, y el horario de clase. Se asume que cada profesor imparte sólo un curso por semestre y cada curso tiene un solo profesor cada semestre, pero un profesor puede impartir más de un nivel.

CURSOS

profesor	semestre	curso	nivel	laboratorio
Guillén	95/1	Unix	1	A
Arroyo	95/1	Informix	1	B
Cuevas	95/2	Sybase	1	E
Guillen	95/1	Unix	2	D
Arroyo	95/1	Informix	2	C

Figura 3.43

La relación "CURSOS" tiene dos llaves candidatas: "profesor, semestre, nivel" y "curso, semestre, nivel". Además, la relación está en 3FN porque tiene únicamente un atributo no-llave, que depende totalmente de cada llave candidata; sin embargo, existe redundancia, toda vez que el nombre del profesor de un curso para un semestre dado aparece más de una vez. Esto se

debe a que el atributo "profesor" forma parte de una de las llaves candidatas y además depende de algunos atributos de otra de la cual no forma parte. Problemas como el expuesto fueron identificados por Boyce y Codd en relaciones en 3FN, para lo cual propusieron una nueva versión de la esa forma normal, conocida forma normal de Boyce y Codd (FNBC). Para eliminar la redundancia del ejemplo citado, es necesario convertir la relación "CURSOS" a FNBC.

Date define que "Una relación *R* está en forma normal de Boyce/Codd (FNBC) si y sólo si cada determinante es una llave candidata."[6]. Un determinante es un atributo del cual otros atributos dependen funcionalmente.

En el ejemplo anterior, la combinación "profesor, semestre" determina el curso, pero no es llave candidata. Igualmente la combinación "curso, semestre" determinan al profesor, pero tampoco es llave candidata de la relación. Estas dependencias se describen como a continuación se indica:

profesor, semestre --> curso

curso, semestre --> profesor

Ambas combinaciones de atributos son determinantes pero ninguna es llave candidata. Para obtener la FNBC de la relación "CURSOS" de la figura 3.43 es necesario dividir la

---

[6] Date, C. J. *An Introduction to Database Systems*. p. 374.

relación en dos, creando la relación "PROFESORES", la cual debe incluir a "profesor", "semestre" y "curso" (figura 3.44).

CURSOS

profesor	semestre	curso	nivel	laboratorio
Guillén	95/1	Unix	1	A
Arroyo	95/1	Informix	1	B
Cuevas	95/2	Sybase	1	E
Guillén	95/1	Unix	2	D
Arroyo	95/1	Informix	2	C

PROFESORES

profesor	semestre	curso
Guillén	95/1	Unix
Arroyo	95/1	Informix
Cuevas	95/2	Sybase

Figura 3.44

Dado que el énfasis de la FNBC está en relaciones con múltiples llaves candidatas, cuando una relación está en 3FN y tiene una sola llave candidata entonces está en FNBC.

4. DISEÑO DEL MODELO FISICO

4. DISEÑO DEL MODELO FISICO

El diseño físico de bases de datos se refiere al diseño de las estructuras físicas que contendrán la base de datos, es decir, los archivos, así como los índices utilizados para lograr acceso eficiente a la información. El modelo físico de una base de datos es un modelo de la realidad expresado en términos de archivos y estrategias de acceso tales como índices.

4.1 Almacenamiento y acceso físico a los datos

Antes de entrar de lleno a hablar sobre archivos e índices, es conveniente comprender primero cómo se almacenan físicamente los datos y el proceso general para accesarlos. En el proceso intervienen básicamente el manejador de la base de datos, el administrador de archivos y el administrador de disco. Estos dos últimos elementos forman parte del sistema operativo.

Puede decirse que un DBMS ve una base de datos a través del administrador de archivos, como un conjunto de archivos; a su vez, el administrador de archivos ve los archivos de una base de datos mediante el administrador de disco, como un conjunto de páginas o bloques de información. Finalmente, el administrador de disco percibe un disco como un conjunto de direcciones físicas. A continuación se describen, de una manera global, los pasos para recuperar un determinado registro.

1. Primero, un usuario o una aplicación solicita al manejador de la base de datos, un registro que cumpla con ciertas características. El DBMS determina el registro que reúne

los requisitos determinados y solicita al administrador de archivos, la recuperación de ese registro.

2. El administrador de archivos determina en qué página se localiza el registro requerido y solicita al administrador de disco que recupere esa página. Una página es la unidad de entrada-salida (E/S) de datos que pueden transferirse entre disco y memoria principal durante un acceso a disco[1]. Otros autores como Jeffrey Ullman se refieren a una página como un bloque[2]. Algunos tamaños de páginas abarcan 1024, 2048 ó 4096 bytes. El administrador de archivos ve al disco como una serie de conjuntos de páginas, formados por páginas de tamaño fijo. Cada conjunto de páginas se identifica con un identificador de conjunto ("page set ID"). A su vez, cada página se identifica con un número de página.

3. Finalmente, el administrador de disco determina la ubicación física de la página dentro del disco y ejecuta las operaciones de E/S necesarias.

Los procesos anteriormente descritos pueden variar en función del hardware y del software, pero tienen validez de carácter general.

---

1 Date, C. J. *An Introduction To Database Systems*. The Systems Programming Series: Addison Wesley. Estados Unidos de Norteamérica. 4ª Edición, 1988. p 47.

2 Ullman, Jeffrey D. *Principles Of Database Systems*. Computer Science Press. Estados Unidos de Norteamérica. 2º Edición, 1992. pp 36,37.

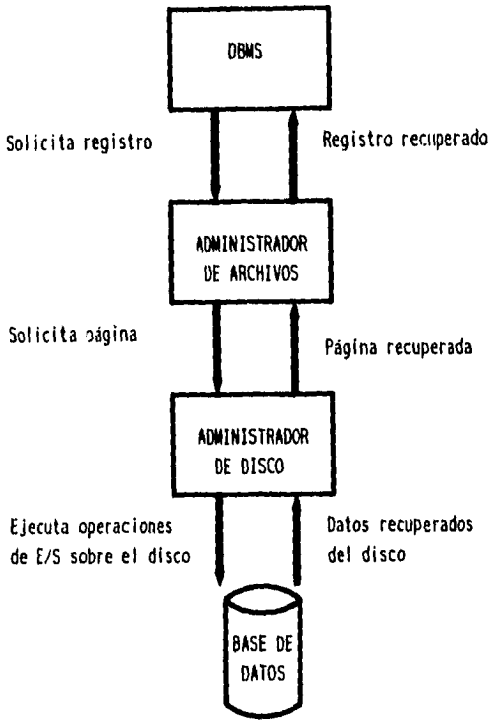

Figura 4.1 Para recuperar un registro dado, intervienen básicamente el DBMS, el administrador de archivos y el administrador de disco. Estos dos últimos forman parte del sistema operativo.

4.1.1 Manejo de páginas

La función principal del administrador de disco es hacer transparentes todos los detalles de las E/S's físicas en disco para el administrador de archivos, y permitirle una visión en términos (lógicos) de entradas y salidas de páginas. Esta función del administrador de archivos se conoce como administración de páginas. A continuación se presenta un ejemplo de esta

función. En el ejemplo se manejarán los archivos "PROFESORES" y "CURSOS", cuyos registros se identificarán con una "P" y una "C", respectivamente. Supóngase que cada registro requiere, para su almacenamiento, una página entera. Los eventos del ejemplo son los siguientes:

1. Inicialmente la base de datos está vacía. Sólo existe un conjunto de páginas libres, numeradas secuencialmente a partir de la unidad. La página cero es un caso especial que más adelante se detallará.

2. El administrador de archivos solicita la creación de un conjunto de páginas para registros de profesores, e inserta cinco registros (P1, P2, P3, P4 y P5). Ahora hay dos conjuntos de páginas: el conjunto de páginas libres y el correspondiente a registros de profesores.

3. Más adelante, el administrador de archivos solicita la creación de un nuevo conjunto de páginas para registros de cursos e inserta cinco registros de cursos (C1, C2, C3, C4, y C5), que abarcan las páginas 6 a 10.

4. Después, el administrador de archivos inserta un nuevo registro de profesores (P6). El administrador de disco localiza el primer espacio libre para guardarlo (página 11) y lo agrega al conjunto de páginas de profesores.

En este momento, el estado de las páginas es el siguiente:

Conjunto de Páginas:	Páginas:
Páginas libres "///"	12, 13, 14, ...
Registros de profesores "P"	1, 2, 3, 4, 5 y 11
Registros de cursos "C"	6, 7, 8, 9 y 10

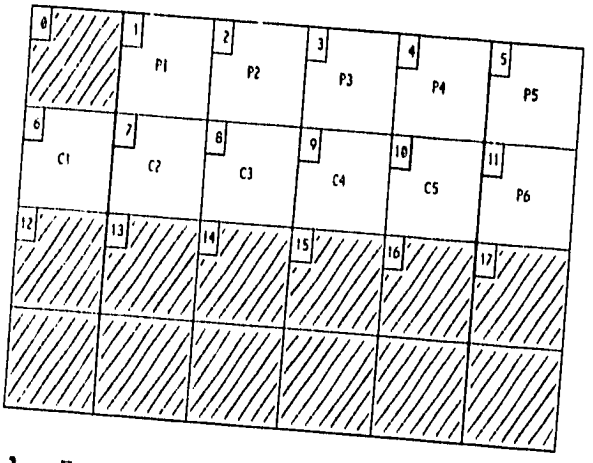

Figura 4.2 Estado de las páginas del disco hasta el evento cuatro.

5. Ahora se desea eliminar el registro P2. El administrador de archivos borra ese registro, contenido en la página número dos, y el administrador de disco agrega esa página al conjunto de páginas libres.

6. Se desea agregar un nuevo registro para un curso. El administrador de archivos inserta un nuevo registro (C6). El administrador de disco guarda ese registro en la primer página libre, en este caso es la número dos.

7. Ahora se desea borrar el registro P4. El administrador de archivos borra el registro P4, contenido en la página número cuatro, que queda vacía y pasa a formar parte del conjunto de páginas libres.

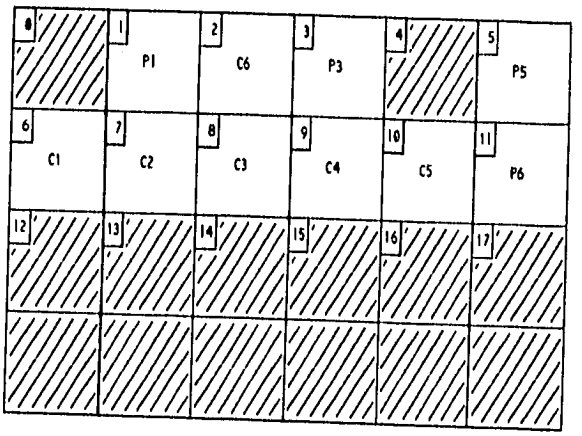

Figura 4.3 Estado final de las páginas del disco después de los eventos uno a siete, en los que se han borrado registros existentes y se han insertado nuevos.

Después de que el sistema ha estado insertando y borrando registros, no puede garantizarse que las páginas que están lógicamente juntas lo están físicamente. Por ejemplo, C6

no está físicamente junto a C5, ni P6 junto a P5. La secuencia lógica de las páginas es posible mediante el uso de apuntadores de página. Cada página tiene un encabezado de página, que contiene entre otras cosas la dirección física de la página que lógicamente le sigue. Los encabezados de página, y en particular los apuntadores de página, son manejados por el administrador de disco. Para localizar los diferentes conjuntos de páginas, es suficiente con encontrar la primer página del conjunto; las siguientes pueden localizarse siguiendo los apuntadores de cada una de las páginas. La dirección de la primera página de un conjunto se almacena en una página predefinida para tal efecto (página cero), la cual contiene normalmente una lista de los conjuntos de páginas que existen en el disco, con un apuntador hacia la primera página de cada conjunto. A esta página se le denomina tabla de contenido de disco o FAT (File Allocation Table).

4.1.2 Manejo de registros

Así como el administrador de disco tiene la función de administrar las páginas, el administrador de archivos tiene la función manejar registros, como a continuación se explica.

Supóngase que cada página puede contener varios registros (normalmente es así), en vez de uno como se ejemplificó anteriormente, y que el orden lógico de los registros se determina por los valores de su llave primaria. Considérense los siguientes eventos:

1. Los registros P1, P2, P3, P4 y P5 se insertan juntos en una misma página *p*, como se muestra en la figura 4.4. Véase que esta página aún tiene espacio libre para insertar más registros.

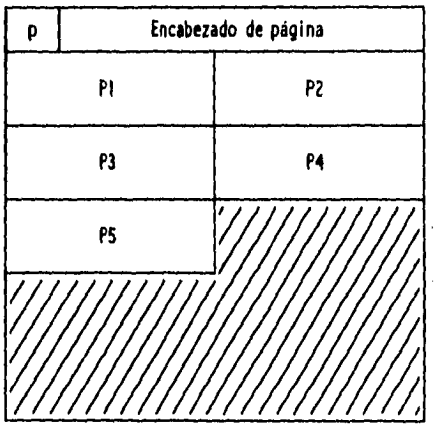

Figura 4.4 Estado de la página *p* después de insertar los primeros cinco registros.

2. Ahora se desea insertar el registro P9. El administrador de archivos guarda este registro en la página *p*, pues todavía hay espacio suficiente, y lo coloca inmediatamente después de P5.

3. Ahora se desea eliminar el registro de P2 de la página *p*. Al borrarlo, los registros P3, P4, P5 y P9 se recorren hacia arriba para no dejar espacio libre entre ellos.

4. Al insertar un nuevo registro por ejemplo P7, el administrador de archivos almacena tal registro en la página *p* y abre un espacio entre P5 y P9, recorriendo P9 hacia abajo y coloca P7 entre esos dos registros.

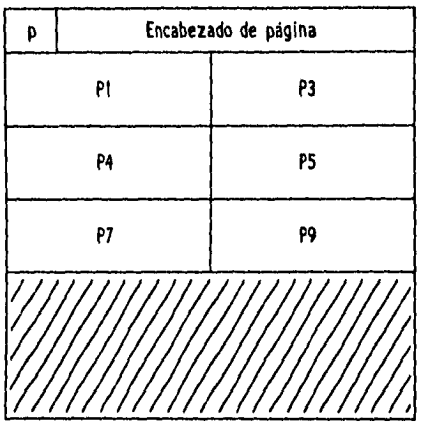

Figura 4.5 Estado de la página *p* después de insertar P9, borrar P2 e insertar P7.

La secuencia de los registros dentro de una página pueden guardar un cierto orden físico, determinado por la llave primaria de los registros.

4.1.3 Orden físico de registros

Otro concepto importante en el almacenamiento de datos es el orden o secuencia física que mantienen los datos que están lógicamente relacionados. Por ejemplo, durante una consulta, el último registro recuperado es *r*1, contenido en la página *p*1 y se desea recuperar *r*2, contenido

en *p*2. Dado que *p*1 y *p*2 son páginas contiguas, el tiempo de acceso a *r*2 es casi nulo, pues las cabezas de lectura y escritura del disco duro están inmediatas a la posición deseada. Si las páginas que contienen a *r*1 y *r*2 están separadas, el tiempo de acceso es mayor.

Hay dos formas de mantener juntos los registros en un archivo, una es la unión intra-archivo y otra es la unión inter-archivo. En una secuencia física intra-archivo, todos los registros de un archivo están almacenados juntos en una secuencia dada por un valor específico asignado por el usuario. Por ejemplo, número de empleado, número de cuenta, etc. El registro con número de empleado 101 está contiguo al 102, seguido del 103 y así sucesivamente. En una unión inter-archivo, los registros de múltiples archivos relacionados (relaciones 1:1, 1:N o M:N) se almacenan físicamente juntos.

4.2 Archivos

Una vez expuesto lo anterior, puede hablarse de la estructura física de almacenamiento de una base de datos. Los datos se almacenan en archivos que residen en medios de almacenamiento tales como discos magnéticos. Así como una tabla tiene renglones, un archivo se compone de registros. A su vez un registro consta de campos, equivalentes a las columnas de una tabla. Desde el punto de vista físico, un archivo es un conjunto de todas las ocurrencias un tipo de registro.

Al definir una base de datos con un DBMS los detalles de las estructuras de almacenamiento son transparentes para el usuario.

4.2.1 Registros

Un registro es un conjunto de datos interrelacionados, que hacen referencia a un mismo objeto (una ocurrencia de una entidad). Un registro es equivalente a un renglón de una tabla. En el nivel físico, los registros tienen una identificación de registro o RID ("record ID"), que indica la página en que se encuentra y su ubicación dentro de la página. Cuando se hace un acceso secuencial a un archivo, el orden que se sigue es el orden ascendente de los RID's.

En algunas ocasiones, los registros contienen datos de control además de los datos que el usuario define. Esa información se sitúa al inicio del segmento como un prefijo de registro, el cual es transparente para los usuarios y para las aplicaciones. Uno de los datos de control contenidos podría ser el tamaño del registro, si se trata de un registro de longitud variable.

4.2.2 Campos

Los datos que forman parte de un registro se almacenan en lo que se denomina campo. Un campo es equivalente a una columna de una tabla. Los términos atributo, columna y campo se utilizan frecuentemente de un modo intercambiable. No obstante, estos términos corresponden a contextos diferentes. Se utiliza el término atributo cuando se habla de relaciones. Se habla de columnas cuando se hace referencia a tablas. El término campo aplica cuando se hace referencia a un archivo de datos. Desde el punto de vista del nivel físico, un campo es un conjunto de bytes (caracteres) en el cual se almacena un dato. Al definir un campo por medio de un manejador de bases datos, se especifican tres características principales: nombre, tipo y longitud.

Nombre de campo

Todo campo debe tener un nombre, el cual debe ser único dentro del mismo archivo. El nombre de un campo de un archivo es como el título de una columna dentro de una tabla.

El nombre debe ser descriptivo, es decir, debe dar una idea general del dato que representa. Preferentemente, los nombres de los atributos utilizados en el diseño conceptual de la base de datos deben conservarse como nombres para los campos. Aunque algunos DBMS's restringen el tamaño o caracteres permitidos para el nombre de un campo, éste debe se tan completo y autoexplicativo como sea posible.

Hay ocasiones en que es necesario utilizar abreviaturas en el nombre de un campo. Algunas veces estas abreviaturas sirven como prefijos o sufijos, por ejemplo "num" en "num_empleado". Es conveniente seguir un estándar de abreviaturas. Las abreviaturas siguientes son de uso común en nombres de campos:

num - número

cve - clave

cta - cuenta

ss - seguro social

rfc - registro federal de contribuyentes

cp - código postal

A continuación se presentan algunas reglas para abreviar nombres de campos.

1. Si las palabras en el nombre son demasiado largas, es conveniente abreviarlas y tener como máximo tres palabras abreviadas en todo el nombre.

2. Las letras iniciales de cada palabra siempre deben conservarse.

3. Debe preferirse conservar las consonantes sobre las vocales.

4. En abreviación, el principio de las palabras es más descriptivo que el final.

La palabra "número" es uno de los prefijos más comunes en nombres de campos. Usualmente se utiliza la abreviatura "num" o el signo de número ("#") como por ejemplo, "#empleado".

Cuando sea necesario utilizar dos o más palabras en un nombre de campo, es recomendable utilizar un guión bajo como separador. Por ejemplo, "num_empleado". Finalmente, debe usarse el singular en vez del plural, ya que un campo contiene un valor atómico. Por ejemplo, el campo "num_empleado" describe en cada una de sus ocurrencias uno y sólo un número de empleado.

Tipos de datos

Una vez que se ha definido el nombre de un campo, se debe evaluar el tipo de dato que contendrá. Un tipo de datos define las propiedades de los valores permitidos para un campo. Todos los datos dentro del mismo campo son del mismo tipo.

Básicamente existen dos tipos de datos: los numéricos y los no numéricos. Entre los datos numéricos están los números enteros y los reales.

Números Enteros	Números Reales
1	3.1416
-53	-89.12
17583	.05

Los datos no numéricos son conjuntos de caracteres alfanuméricos, que incluye los diez dígitos decimales[3], las 26 letras del alfabeto y un número de caracteres especiales, como por ejemplo: $, =, + y -. El registro federal de contribuyentes es una combinación de letras y números; por tanto, puede ser considerado alfanumérico.

Distintos manejadores de bases de datos soportan una gran variedad de tipos de datos, que son derivados de los dos tipos mencionados. Los tipos de datos más usuales son: números

---

[3] En este caso, un número se considera como un carácter; no pueden realizarse operaciones matemáticas con él.

reales, números enteros, alfanuméricos, fechas, horarios, monetarios, valores lógicos y campos de memoria.

Además de los tipos anteriores, existen campos para contener imágenes o sonidos digitalizados, que regularmente se definen como tipo objeto.

El tipo de dato de un campo depende básicamente de la naturaleza de los datos que va a contener; no se elegiría un tipo alfanumérico para contener el precio de un producto. Habrá ocasiones en que exista discrepancia en qué tipo de dato utilizar, por ejemplo, un código postal ¿se define como un valor numérico o como una cadena de caracteres?. Si se optara por el valor numérico se tendría el inconveniente que la algunos DBMS's especifican que los números enteros no pueden contener ceros precedentes; por ejemplo el código postal "09340" se aceptaría como "9340", lo cual provocaría más adelante la confusión de que si hay un error de captura y se trata del código "93400" ó "09340". Si esta es una desventaja del paquete, entonces sería conveniente aceptar los valores para este campo como una cadena de caracteres. Para definir el tipo de campo hay que considerar la naturaleza de los valores de los datos y los tipos de datos que pueden manejarse con cada DBMS.

Longitud de campo

La longitud de un campo es el número de espacios necesarios para contener los datos dentro de ese campo. Para un campo tipo caracter se debe especificar si contendrá un solo caracter o una cadena de 10 ó 30. La longitud de un registro es la suma de las longitudes de

sus campos. Cuando no se elige correctamente la longitud de un campo se presentan principalmente dos problemas:

1. Si la longitud es más grande de lo necesario, habrá espacio en disco ocupado inútilmente.

2. Si la longitud es demasiado pequeña, se corre el riesgo de que no acepte los datos de una manera completa.

Regularmente existen referencias que pueden dar una idea del tamaño de un campo. Por ejemplo, para una nómina, si se conocen los nombres de los empleados y se observa que el nombre más largo ocupa 35 caracteres, entonces sería conveniente designar una longitud de 40 para el campo "nombre_empleado", pues cabe la posibilidad de encontrar un nombre de mayor tamaño.

Es un hecho que la recuperación de datos es más eficiente cuando se almacena un mayor número de registros por página. Entre más registros se conservan en una página, es menor el número de accesos a disco. Si se tienen registros que ocupan 5 bytes y el tamaño de las páginas es de 15 bytes, únicamente pueden almacenarse 3 registros por página; en cambio, si los registros ocupan 3 bytes entonces cada página contendrá 5 registros. Por lo anterior, si la longitud de los registros de un archivo es muy grande, puede considerarse dividir el archivo en dos, creando una relación uno a uno entre los dos archivos, uno de los cuales contendrá los datos que se usan con mayor frecuencia y el otro contendrá los de menor uso. Se sugiere seguir

este procedimiento cuando es importante la rapidez en la recuperación de información en uno de los archivos. Si este factor no es importante y todos los campos de un archivo se utilizan con frecuencia, es mejor mantener el número de archivos al mínimo.

Cabe señalar que algunos DBMS's soportan registros de longitud variable; es decir, no es necesario definir el tamaño del registro. Este tipo de registro es útil cuando se ha decidido manejar más de una ocurrencia de un mismo tipo dentro de un campo. Por ejemplo, si se deseara guardar en un solo registro todos los cursos impartidos por un profesor, los registros se verían como se muestra en la figura 4.6. En este caso se trata de una relación 1:N entre profesores y cursos. En cada registro, el primer valor corresponde a la entidad "PROFESORES" y el resto son los valores de N ocurrencias de "CURSOS" relacionados. La llave primaria del registro es el registro entero.

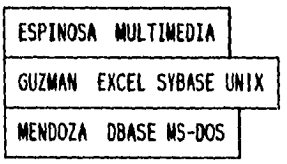

Figura 4.6 Registros de longitud variable que guardan los cursos que son impartidos por profesores.

Para cumplir con la Regla de Integridad de Entidad, vista en el capítulo 2, es importante definir, durante la creación de archivos, mediante un DBMS, que los campos que forman parte de la llave primaria no acepten valores nulos.

4.3 Indices

Cuando se dice que un archivo se accesa secuencialmente, significa que los registros son accesados uno después de otro de acuerdo al orden físico, hasta encontrar el registro deseado. El seguimiento de la secuencia se hace por medio de los identificadores de registro (RID's). Si el archivo está contenido en más de una página entonces serán necesarias más operaciones E/S sobre el disco. El objetivo principal de un índice es reducir el número de accesos al disco.
Un índice es una organización de datos que permite que ciertas consultas accesen uno o más registros de un archivo de datos una manera rápida. Conceptualmente, un índice es una tabla formada por dos columnas; una de ellas contiene los datos sobre el cual un archivo está indizado, la otra columna contiene apuntadores hacia los registros del archivo indizado. Físicamente, un índice es un archivo de datos con dos campos.

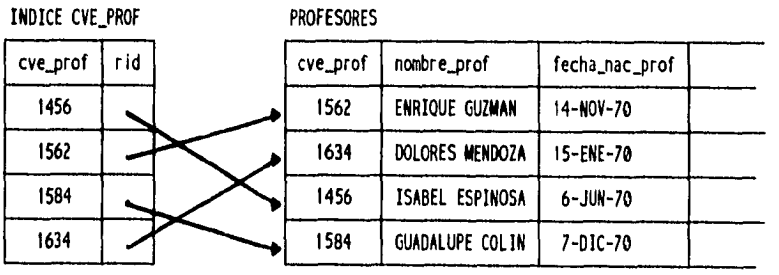

Figura 4.7 Indice del campo cve_prof para el archivo PROFESORES.

Cuando un índice es utilizado para accesen el índice en vez del archivo de datos. Si en el ejemplo de la figura 4.8 se quiere recuperar el registro de un profesor con una clave cualquiera, primero se hace una búsqueda de la clave en el índice de "cve_prof" y luego con la dirección obtenida se recupera la página que contiene el registro y éste se accesa directamente sin tener que recorrer la página. Para comprender mejor cómo funcionan los índices, puede hacerse una comparación con el procedimiento utilizado para localizar un libro en una biblioteca. Por una parte, existe un fichero o catálogo (índice) de libros (registros), en el cual se indica la clasificación (colocación) de cada libro en los estantes (archivos).

Un archivo puede tener uno o más índices. Un índice sobre algún campo de la llave primaria es llamado índice primario. Los índices sobre otros campos son llamados índices secundarios. Por ejemplo, el archivo "PROFESORES" puede tener un índice de las claves y otro sobre los nombres (figura 4.8).

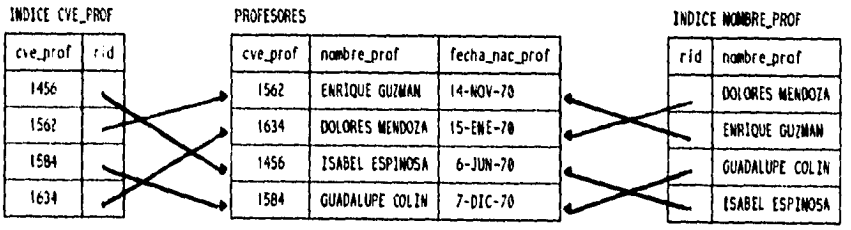

Figura 4.8 Archivo "PROFESORES" con doble índice, uno para "cve_prof" y otro para "nombre_prof".

Los índices pueden funcionar dos maneras, de acuerdo al orden físico del archivo indizado: cuando los archivos de datos están ordenados y cuando no lo están. Todo archivo tiene una orden físico, determinado por la secuencia de los registros en cada página y por la secuencia de las páginas. Los ejemplos de índices hasta ahora mencionados son para archivos no ordenados. Los índices de estos archivos son muy grandes, pues contienen una referencia para cada registro. Sin embargo, una gran ventaja es que pueden hacerse consultas como ¿existe algún profesor de nombre Guadalupe Colín?, sin tener que buscar en el archivo de datos.

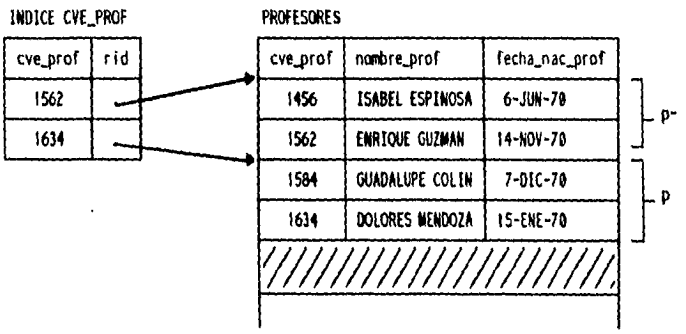

Figura 4.9 Ejemplo de un índice de un archivo ordenado.

Para comprender ahora cómo funciona un índice para un archivo ordenado, supóngase que el archivo "**PROFESORES**" está almacenado de tal modo que su orden físico corresponde a su orden lógico, determinado por el valor de alguno de sus campos, por ejemplo "cve_prof", y que se tiene un índice de ese campo. Bajo esta situación, no es necesario que el índice contenga un apuntador para cada registro del archivo indizado, basta con un apuntador para cada

página, que indique el valor más alto de "cve_prof" dentro de la página y el número de página. Véase la figura 4.9 donde el archivo indizado está ordenado de acuerdo a "cve_prof". Si se desea recuperar el registro con valor "cve_prof" igual a 1584, el sistema busca primero en el índice un valor mayor o igual a este. El sistema encuentra el valor 1634, cuyo RID apunta a la página *p* y recupera esa página. Finalmente, en la memoria de la computadora busca de manera secuencial el registro deseado. Una ventaja de manejar así los índices es que los archivos de índices ocupan menos espacio.

Una desventaja general de usar índices es que ocupan espacio de almacenamiento y por otra parte, hacen más lenta la actualización de registros, pues por cada uno que se inserta, modifica o borra en el archivo de datos, hay que realizar la misma operación también en los archivos de índices.

Un índice es realmente útil cuando se definen correctamente los campos sobre los cuales se indiza un archivo de datos. Si por ejemplo, hay un índice sobre un atributo que ninguna consulta emplea, entonces el índice no brinda ningún beneficio, sólo ocupa espacio y hace lento el sistema.

Las tareas que manejan la mayor cantidad de datos regularmente son las más lentas y si estas tareas son frecuentes, es conveniente crear índices de los campos determinantes de la información requerida. Para decidir correctamente los campos que deben estar indizados, puede utilizarse el cuadro de análisis de uso de campos descrito en la siguiente sección.

4.3.1 Análisis de uso de campos

Para determinar los campos sobre los cuales deben crearse los índices, se debe evaluar el uso de todos y cada uno de los campos en procesos donde hay recuperación de datos. Para tal efecto, se propone construir un cuadro de análisis de uso de campos, en el que se muestran, por un lado, los procesos más frecuentes sobre la base de datos tales como consultas y reportes, y por otro lado, los archivos con sus respectivos campos. Por ejemplo considérense las siguientes relaciones:

PROFESORES (cve_prof, nombre_prof, rfc_prof, fecha_nac_prof, domicilio_prof, num_tel_prof)

CURSOS (cve_curso, nombre_curso, horas_dura_curso, costo_curso)

ESTUDIANTES (num_cta, nombre_est, fecha_nac_est, domicilio_est)

Supóngase que la tarea más frecuente es la impresión de los siguientes reportes:

- Listado de profesores
- Listado de cursos ofrecidos
- Listado de profesores y cursos que imparten
- Listado de estudiantes
- Listado de estudiantes por curso

CUADRO DE ANALISIS DE CAMPOS

	PROCESOS				
	1	2	3	4	5
PROFESORES					
cve_prof	///		///		
nombre_prof	///		///		
rfc_prof					
fecha_nac_prof					
domicilio_prof					
num_tel_prof					
CURSOS					
cve_curso		///	///		///
nombre_curso		///			
horas_dura_curso		///			
costo_curso		///			
ESTUDIANTES					
num_cta				///	///
nombre_est				///	///
fecha_nac_est					
domicilio_est					
num_tel_est					

1. Listado de profesores
2. Cursos ofrecidos
3. Listado de profesores y cursos que imparten
4. Listado de estudiantes
5. Listado de estudiantes por curso que toman

/// Utilizado en el proceso

Figura 4.10

En el cuadro de análisis, deben señalarse los campos que participan de manera directa en cada una de las tareas listadas. Por participación directa debe entenderse que un campo es utilizado en procesos de búsqueda y comparación, estos campos son regularmente las llaves primarias. Por ejemplo, si se desea conocer el costo y las horas que dura un curso, por decir de Excel, los campos que intervienen en la consulta son "cve_curso", "costo_curso" y "horas_dura_curso"; sin embargo, la participación de "cve_curso" es fundamental, pues sin él no pueden determinarse los otros campos.

En el cuadro terminado (figura 4.10), puede apreciarse cuáles son los campos de mayor uso. Igualmente, pueden distinguirse las tablas que intervienen en un mismo reporte o consulta, como por ejemplo, "PROFESORES" y "CURSOS", que son necesarios para el reporte "Listado de profesores y cursos que imparten".

5. ESTUDIO DE CASO:
BASE DE DATOS DE INDICADORES DE PRODUCTIVIDAD CIENTIFICA

5. ESTUDIO DE CASO: BASE DE DATOS DE INDICADORES DE PRODUCTIVIDAD CIENTIFICA

Una de las actividades de mayor importancia que se llevan a cabo dentro de la Universidad Nacional Autónoma de México, es la investigación científica. Dentro de nuestra universidad se realizan trabajos que trascienden el contexto universitario y llegan a tener un fuerte impacto en el ámbito internacional y en el avance del conocimiento.

Dentro de la Secretaría Académica de la Coordinación de la Investigación Científica de la UNAM, existe la Unidad de Indicadores de Productividad Científica, que entre sus funciones tiene la tarea de llevar un registro de los trabajos publicados por los investigadores universitarios, así como medir el impacto de estas obras en la comunidad científica nacional e internacional. Para este último punto se considera como factor de medición del impacto, las citas que se hacen sobre un trabajo en un período de tiempo dado.

Resulta evidente, dados los niveles de producción de trabajos científicos, que este departamento requiere de un sistema de cómputo que pueda realizar las funciones anteriormente citadas, lo cual implica crear una base de datos en la que se puedan relacionar los datos personales de los académicos universitarios con sus trabajos y sobre todo con bases de datos extranjeras que contienen los factores de impacto de las revistas donde los trabajos son publicados, además de sus niveles de citación en diferentes períodos de tiempo, considerando las principales revistas científicas internacionales.

5.1 Problemática actual

La principal fuente de información de la Unidad de Indicadores, es el "Science Citation Index" (SCI). El SCI es una base de datos disponible en disco compacto que contiene información de las ci4/s que se hacen sobre las obras publicadas por diversos científicos en el mundo. La creación y mantenimiento de esta base de datos es una labor del Institute for Scientific Information de los Estados Unidos. El disco compacto de SCI existe desde 1980 y es actualizado trimestralmente.

Esta base de datos presenta algunos inconvenientes, siendo el mayor de los problemas la dificultad para relacionar los trabajos de los académicos con las citas que aparecen en el SCI.

En su estructura, SCI asigna a cada trabajo una clave de referencia, la cual a juicio del SCI debe considerarse como llave primaria. Al experimentar con esta base de datos se observó que existe más de una "llave primaria" para un mismo trabajo. Por ejemplo, la obra *"Chemical Evolution of the Galaxy", Serrano P.G. Thesis, University of Sussex, England, (1978)*, tiene las llaves:

SERRANO-A-1978-THESIS-U-SUSSEX y

SERRANO-A-1978-THESIS-U-SUSSEX ENGL

Si se recuperaran las citas relacionadas con la primera clave, se ignorarían todas las citas que están relacionadas a la segunda.

Por otra parte, si se desea extraer información de trabajos publicados por académicos de la UNAM, se presenta otro inconveniente, toda vez que se han identificado poco más de veinte maneras de reportar a esta institución, desde sus siglas "UNAM", hasta el nombre completo o por ejemplo "Univ. Nal. Aut. de Mex.", "Mexico National. A. Univ.", etc.

Otro grave problema es que la estructura utilizada por SCI arroja información que a menudo resulta confusa y poco confiable . El caso más común es que en un mismo campo se acepta más de un valor de campo. Por ejemplo, en el campo "autor" se acepta el nombre del primer autor, seguido de los coautores, separados por un caracter especial. Del mismo modo, en otro campo se presentan las nacionalidades de los autores. Es común que el contenido de los campos mencionados no sea consistente como se muestra a continuación:

AUTOR: LEQUEUX L; FOSTER H; MALAGNINI;

PAIS: FRANCE; BELGIUM

En este ejemplo, resulta imposible establecer la relación que existe entre los tres autores y los dos países mencionados. Posiblemente lo único que se podría afirmar es que el primer autor es de nacionalidad francesa.

Otro caso también común es el siguiente:

AUTOR: LEQUEUX L; FOSTER H;

PAIS: FRANCE; BELGIUM; ITALY

En este ejemplo surgirían las siguientes preguntas: ¿Será que Lequeux o Foster tienen doble nacionalidad? ¿Será que el capturista omitió ingresar un nombre? o más bien ¿agregó un país de más?.

La falta de controles de integridad en el SCI la hacen una base de datos poco confiable, a pesar de ser la única en su género y que goza de gran prestigio internacional.

Sirva el caso del SCI como muestra de que aún en los grandes proyectos de bases de datos no se aplican las reglas más elementales de diseño de bases de datos.

El problema de diseño de la Base de datos de Indicadores de Productividad Científica es crear las estructuras que puedan contener los datos de académicos, trabajos y citas. Las estructuras creadas deben permitir una interacción con el SCI.

5.2 Diseño del modelo conceptual

Conforme al problema descrito anteriormente, se observa que existen tres entidades involucradas, las cuales son: ACADEMICOS, TRABAJOS y CITAS.

La entidad ACADEMICOS representa al conjunto de investigadores de quienes es necesario tener su registro. Esta entidad tiene los siguientes atributos:

Nombre

Sexo

Registro Federal de Contribuyentes

Nombramiento dentro de la UNAM

Disciplina, área y nivel en el Sistema Nacional de Investigadores (SNI)

Nivel de PEPRAC (Programa de Estímulos a la Productividad del Personal Académico).

La entidad TRABAJOS representa los artículos publicados por los académicos en revistas nacionales e internacionales. Esta entidad tiene los siguientes atributos:

Título del trabajo

Autores

Idioma en que está escrito

Nombre de la revista que lo publica

Año de publicación

Volumen.

La entidad CITAS comprende la citación que sobre los trabajos de los académicos hacen otros autores. La información sobre las citas se obtiene del SCI. Esta entidad tiene los siguientes atributos:

Indice del SCI

Título y autor del trabajo citado

Título y autor del trabajo citante

Idioma en que está escrito el trabajo citante

Nombre de la revista que lo publica

Año de publicación

Volumen.

Las relaciones que se identifican entre las entidades ACADEMICOS, TRABAJOS y CITAS son las siguientes:

1. "Académicos *publican* Trabajos" o "Trabajos *son publicados por* Académicos".
2. "Trabajos *son citados en* Citas".

Para construir el diagrama E-R deben considerarse los siguientes eventos:

1. Un investigador puede publicar cero o más trabajos.
2. Un trabajo puede ser publicado por uno o más investigadores.
3. Un trabajo puede aparecer en cero o más citas.

Construcción del Diagrama entidad relación.

Se observa que existe una relación M:N entre ACADEMICOS y TRABAJOS, así como una relación 1:N entre TRABAJOS y CITAS. En la relación "Académicos *publican* Trabajos", la participación de ACADEMICOS en su relación con TRABAJOS es opcional, toda vez que puede haber académicos que no publiquen trabajos, aunque es preciso conservar sus datos. Por otra parte, todo trabajo debe estar relacionado al menos a un autor, pues no puede existir un

trabajo sin autor; por lo tanto la participación de TRABAJOS es obligatoria. Por lo que refiere a la relación "Trabajos *son citados en* Citas", un trabajo puede no haber sido citado, mientras las ocurrencias de CITAS no pueden aparecer sin un trabajo al cual referirse. El diagrama entidad relación sería el siguiente:

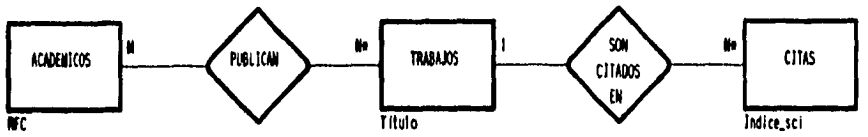

Figura 5.1

5.3 Diseño del modelo lógico

Relaciones preliminares

De acuerdo al diagrama E-R anterior y por tratarse de una relación M:N entre ACADEMICOS y TRABAJOS, se aplica la Regla 6 vista en el capítulo 3, para obtener las relaciones preliminares de esta asociación (figura 5.2).

REGLA 6. Cuando el grado de una relación binaria es M:N, se requieren tres relaciones para contener los datos; una por cada entidad, con la llave de cada entidad como la llave primaria para la relación correspondiente, y una relación más para la asociación. La relación de asociación tendrá entre sus atributos las llaves de cada entidad.

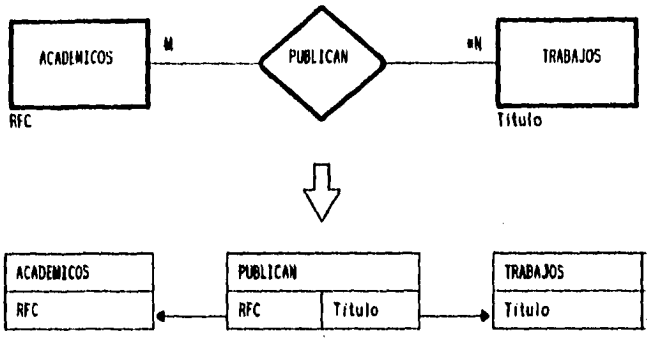

Figura 5.2

En cuanto a la relación "Trabajos *son citados en* citas", se trata una relación 1:N, donde la participación de CITAS es obligatoria, pues para que una cita exista debe existir primero un trabajo al que hace referencia. Para este caso se aplica la Regla 4 (figura 5.3).

REGLA 4. Cuando el grado de una relación binaria es 1:N y la participación del lado N es obligatoria, dos relaciones son necesarias: una por cada entidad, con la llave de cada una de ellas como llave primaria para la relación correspondiente. Adicionalmente, la llave de entidad de la parte opcional debe agregarse como llave foránea en la relación que corresponde a la parte obligatoria.

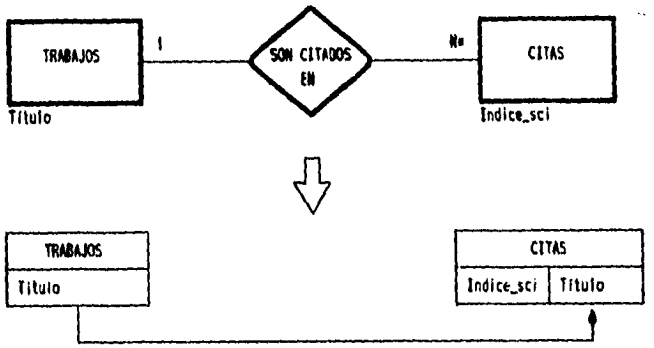

Figura 5.3

Normalización

Después de haber obtenido las relaciones preliminares, ahora es necesario normalizarlas.

Formas normales para la relación **ACADEMICOS**

1FN Tiene como llave primaria el atributo "RFC".

2FN Todos los atributos no-llave dependen totalmente de la llave primaria.

3FN No existe dependencia entre los atributos no-llave.

Formas normales para la relación **TRABAJOS**

1FN a) La relación TRABAJOS tiene como llave primaria el atributo "Título del trabajo".

b) Con el fin de evitar que el atributo "autores" acepte múltiples valores de campo, éste debe eliminarse de la relación. Además, al aplicar la Regla 6 se ha creado la relación "PUBLICAN" en la que se relacionan *m* autores con *n* trabajos.

2FN Todos los atributos no-llave dependen de la llave primaria.

3FN No existe dependencia entre los atributos no-llave.

Formas normales para la relación CITAS

1FN La relación CITAS tiene como llave primaria al atributo "Indice SCI".

2FN Todos los atributos no-llave dependen de la llave primaria.

3FN El atributo "autor del trabajo citante" debe eliminarse de la relación toda vez que su valor puede conocerse en la relación TRABAJOS por medio del atributo "título de trabajo citante".

La relación PUBLICAN, resultante de aplicar la Regla 6 a la asociación entre ACADEMICOS y TRABAJOS está en 3FN pues sus dos atributos forman la llave de la relación.

5.4 Diseño del modelo físico

Los archivos a crear son: ACADEMICOS, TRABAJOS, CITAS y PUBLICAN con la siguiente estructura de sus registros.

ACADEMICOS:

Atributo	Campos	Tipo	Longitud
Nombre	Apellido_mat	Caracter	20
	Apellido_pat	Caracter	20
	Nombres	Caracter	20
Sexo	Sexo	Caracter	1
RFC	RFC	Caracter	14
Disciplina SNI	Disciplina_sni	Caracter	6
Area SNI	Area_sni	Caracter	1
Nivel SNI	Nivel_sni	Caracter	1
Nivel PEPRAC	Nivel_peprac	Caracter	1

TRABAJOS:

Aunque el atributo "título del trabajo" puede ser usado como llave primaria se recomienda asignar un número único que identifique cada trabajo, el cual se utilizaría como llave primaria.

Atributo	Campos	Tipo	Longitud
Número de trabajo	#Trabajo	Numérico	5
Título del Trabajo	Título	Caracter	80
Idioma	Idioma	Caracter	15
Revista	Revista	Caracter	25
Año de publicación	Año_publicación	Caracter	4
Volumen	Volumen	Caracter	2

CITAS:

Atributo	Campos	Tipo	Longitud
Indice SCI	Indice_sci	Caracter	40
Titulo citante	Título_citante	Caracter	80
Idioma	Idioma	Caracter	12
Revista	Revista	Caracter	25
Año de publicación	Año_publicación	Caracter	4
Volumen	Volumen	Caracter	2
Núm. del trabajo citado	#trabajo_citado	Numérico	5

PUBLICAN

Atributo	Campos	Tipo	Longitud
RFC	RFC	Caracter	14
Número de Trabajo	#Trabajo	Numérico	5

Creación de Indices

Con el objeto de hacer más eficiente la recuperación de información, se recomienda crear índices para cada uno de los archivos sobre los campos que a continuación se indican:

Archivo:	Campo:
ACADEMICOS	RFC
TRABAJOS	#Trabajo
CITAS	#Trabajo citado
PUBLICAN	RFC y #Trabajo

El resultado de las tres etapas de diseño debe considerarse como un todo para la implantación del sistema. El diseño presentado atiende las necesidades de información que de cada entidad existen, pero cabe señalar que no garantiza una interacción eficiente con el SCI por la problemática antes expuesta. Una ventaja de esta estructura es que al utilizar como llave primaria de ACADEMICOS el RFC, puede establecerse una relación con la nómina de académicos y a través de ésta con el catálogo presupuestal de la UNAM. Al establecer esa relación, puede obtenerse información tal como gastos por investigación científica, por institutos, sedes, programas, etc.

CONCLUSIONES

CONCLUSIONES

Esta tesis no fue escrita con la intención de ser un compendio de todo lo referente al diseño de bases de datos relacionales. Gran parte del conocimiento sobre el tema no fue cubierto. Sin embargo, se trataron de presentar los aspectos fundamentales que resultaran útiles al lector, principalmente aquel con un cierto grado de conocimientos en cómputo, para que a partir éstos, pueda adoptar una metodología de trabajo.

Referente al diseño conceptual, la técnica de diagramación expuesta aquí es sumamente sencilla. Compárese por ejemplo con la utilizada por Leszek Maciaszec en "*Database Design and Implementation*"[1], donde se presenta una notación diferente y más compleja a la aquí expuesta. Por su parte, Edward Yourdon en "*Análisis Estructurado Moderno*"[2] recomienda el uso de diagramas E-R para casos de sistemas cuya complejidad no radica en los procesos, sino cuyo objetivo es servir de recipiente de una gran cantidad de información y donde ni siquiera es necesario preocuparse por construir un diagrama de flujo de datos (DFD).

La importancia de los diagramas E-R radica en que la estructura de los datos puede ser tan compleja que haga necesario enfatizarlas y examinarlas, independientemente de los procesos. La propuesta de diagrama E-R aquí expuesta fue tomada de la original de Peter Chen, agregando

---

[1] Maciaszec, Leszec A. *Database Design and Implementation*. Prentice Hall. Australia. 1990.

[2] Yourdon Edward. *Análisis Estructurado Moderno*. Prentice Hall. México. 1993.

la notación sobre el tipo de relación, es decir uno a uno, uno a muchos o muchos a muchos. Igualmente se agregó la indicación de la participación de las entidades, ya sea opcional u obligatoria.

Como herramienta fundamental de diseño lógico, se mostró el proceso de normalización, el cual forma parte integral del diseño lógico de bases de datos relacionales. Aunque existen seis formas normales, se considera óptimo llegar a la tercera forma normal, pues una relación en segunda forma normal es susceptible de presentar problemas para insertar, actualizar y borrar datos, mientras que una relación en tercera forma normal, no presenta tales anomalías. En cuanto a la forma normal de Boyce y Codd, así como la cuarta y quinta forma normal, se encuentra que son más bien teóricas y que raramente se practican.

El aspecto físico de una base de datos tiene que ver con el manejador de base de datos a utilizar, pero contar con herramientas de alta tecnología como los actuales DBMS no es suficiente si se desconocen los aspectos fundamentales sobre el nivel físico de una base de datos. Se considera que las nociones aquí presentadas son adecuadas para este fin.

Para esta investigación se consultó un gran número de autores que han hecho contribuciones sobre el tema. Pero quisiera destacar que son pocos los autores originales, toda vez que la mayoría hace referencia la obra de C.J. Date: "An *Introduction To Database*

Systems"[3]: obra que es considerada clásica en la materia, por lo que se recomienda ampliamente su lectura. Otros dos autores trascendentes sobre el tema son Edgar F. Codd y Peter Chen, cuyas publicaciones "*A Relational Model For Large Shared Data Banks*" y "*An Entity-Relationship Model: Toward a Unified View of Data*", respectivamente, dieron origen a la tecnología actual de bases de datos. Reproducciones de estos artículos pueden encontrarse en "*Readings in Database Systems*"[4].

Las bases de datos relacionales son usadas ampliamente en la actualidad y su uso continuará creciendo en el futuro. El éxito en un proyecto de bases de datos requiere sólidos conocimientos de los aspectos técnicos y no técnicos. En esta tesis pueden encontrarse ambos, y se pretende que sean útiles para el lector.

---

[3] Date C.J. *An Introduction To Database Systems*. The Systems Programming Series: Addison Wesley. Estados Unidos de Norteamérica. 4ª edición, 1988.

[4] Stonebraker, Michael. *Readings in Database Systems*. Morgan Kaufmann Publishers, Inc. Estados Unidos de Norteamérica. 1988.

ANEXO A

ANEXO A

Reglas de Codd

Las reglas siguientes fueron establecidas por E. F. Codd en 1985 como condicionantes para que un DBMS pueda ser considerado relacional.

Regla 0. El DBMS debe ser capaz de manejar la base de datos completamente a través de las propiedades relacionales. Esta regla se numera cero porque se considera como la regla fundamento.

Regla 1. Toda la información de la base de datos se debe representar como valores contenidos en tablas o relaciones.

Regla 2. El acceso a los datos deben garantizarse con la combinación del nombre de la tabla con el valor de la llave primaria de esa tabla.

Regla 3. El valor nulo representará información desconocida o inaplicable para un dato.

Regla 4. La descripción de la estructura de la base de datos (diccionario) debe estar contenida en tablas para el uso de un mismo lenguaje por parte del usuario en la manipulación y consulta de la misma.

Regla 5. EL DBMS debe poseer un lenguaje para realizar las siguientes funciones:

Definición de datos

Definición de vistas lógicas de usuario

Manipulación de datos

Controles de integridad

Controles de acceso a los datos

Controles de transacciones lógicas (inicio, final correcto, final erróneo)

Regla 6. Todas las vistas lógicas actualizables serán actualizables por el sistema.

Regla 7. La posibilidad de manipulación de una relación por un lenguaje de alto nivel será aplicable a la consulta, inserción, actualización y borrado de datos.

Regla 8. Los programas de aplicación y los usuarios no deben sufrir variaciones o interferencias por los cambios físicos en el almacenamiento de los datos o en las rutas de acceso a los mismos (independencia física de los datos).

Regla 9. Los programas de aplicación y los usuarios no deben sufrir variaciones o interferencias por los cambios lógicos en las tablas o relaciones que contienen la información (independencia lógica de los datos).

Regla 10. Los controles para garantizar la integridad de los datos deben poder ser definidos y almacenados en el diccionario y no en los programas de aplicación.

Regla 11. La distribución de la base de datos en diferentes nodos no debe afectar a los programas de aplicación o a los usuarios. La localización de la ubicación de la información debe ser resuelta por las funciones del lenguaje de datos.

Regla 12. Si un DBMS relacional posee un lenguaje de bajo nivel capaz de manipular la información registro a registro, se atendrá a las reglas y controles de integridad definidas por el lenguaje de alto nivel.

BIBLIOGRAFIA

BIBLIOGRAFIA

Beynon-Davies, Paul. *Relational Database Design.* Blackwell Scientific Publications. Gran Bretaña. 1992.

Beynon-Davies, Paul. Relational Database Systems. Blackwell Scientific Publications. Gran Bretaña. 1991.

Brathwaite, Ken S. *Relational Databases.* J. Ranade IBM Series: McGraw Hill. Estados Unidos de Norteamérica. 1991.

Chen, Peter Pin-Shan. *"The Entity-Relationship Model - Toward a Unified View of Data".* Communtcations of the Association for Computing Machinery. Volumen 1. Número 1. 1976. [En] STONEBRAKER, MICHAEL (1988).

Codd, Edgar F. *"A Relational Model of Data for Large Shared Data Banks".* Communications of the Association for Computing Machinery. Volumen 13. Número 6. Estados Unidos de Norteamérica, 1970. [En] STONEBRAKER, MICHAEL (1988).

Date, C. J. *An Introduction to Database Systems.* The Systems Programming Series: Addison Wesley. Estados Unidos de Norteamérica. 4a edición, 1988.

Date, C. J. Relational Database: Selected Writings. Addison Wesley. Estados Unidos de Norteamérica. 1989.

Diaz Barriga, Jesus y Guerrero, Ma. de Lourdes. *Computación, Temas Selectos.* Dirección General de Asuntos del Personal Académico de la UNAM. México. 1992.

Fernández, Eduardo B. y Summers, R.C. *Database Security and Integrity.* The Systems Programming Series: Addison Wesley. Estados Unidos de Norteamérica. 1981.

Gillenson, Mark L. *Introducción a las Bases de Datos.* McGraw Hill. México. Traducción de la primera edición en inglés de "Database Step By Step" por María de Lourdes Fournier G. 1988.

Hawryszkiewiks, Igor T. *Relational Database Design.* Prentice Hall. Australia. 1990.

Korth, Henry F. y Silberschatz, Abraham. *Database Systems Concepts.* McGraw Hill. Singapur. 1991.

Lopez-Fuensalida, Antonio. *Metodologias de Desarrollo.* Macrobit. México. 1991.

Maciaszec, Leszek A. *Database Design and Implementation*. Prentice Hall. Australia. 1990.

Martin, James. *Principles of Database Management*. Prentice Hall. Estados Unidos de Norteamérica. 1976.

Mittra, Sintasu S. *Principles of Relational Database Systems*. Prentice Hall. Estados Unidos de Norteamérica. 1991.

Shasha, Dennie Elliott. *Database Tuning*. Prentice Hall. Estados Unidos de Norteamérica. 1992.

Stonebraker, Michael. *Readings in Database Systems*. Morgan Kaufmann Publishers, Inc. Estados Unidos de Norteamérica. 1988.

Ullman, Jeffrey D. *Principles of Database Systems*. Computer Science Press. Estados Unidos de Norteamérica. 1992 4a edición.

Yourdon Edward. *Análisis Estructurado Moderno*. Prentice Hall. México. 1993.

Yourdon, Edward y Constantine, Larry L. *Structured Design*. Prentice Hall. Estados Unidos de Norteamérica. 1979.