UAEM | Universidad Autónoma del Estado de México

CUTex
Centro Universitario UAEM Texcoco

INGENIERÍA EN COMPUTACIÓN

FUNDAMENTOS DE BASES DE DATOS

APUNTES

PERIODO 2015A

Prof. Joel Ayala de la Vega.

Contenido

PRESENTACIÓN

La intención de la realización de los apuntes “Fundamentos de Bases de Datos” es poder otorgar un apoyo didáctico a los estudiantes de la Licenciatura de “Ingeniería en Computación” del C. U. UAEM Texcoco permitiéndoles una mayor comprensión del tema que se considera de gran relevancia para la vida profesional de un Ingeniero en Computación.

Los apuntes tratan de llevar la secuencia que solicita el plan de estudios de dicha Unidad de Aprendizaje. La distribución se realizó de la siguiente forma:

Los tres primeros capítulos de éste escrito están relacionados con la primera unidad de competencias del plan de estudios. Los primeros dos capítulos nos dan las bases mínimas para comprender lo que es una Base de Datos y poder hacer la diferencia con los Sistemas Gestores de Bases de Datos. El tercer capítulo muestra un modelo conceptual para poder analizar un requerimiento y hacer las abstracciones necesarias desde un punto de vista Entidad - Relación con la capacidad de poder sub clasificar o jerarquizar a un tipo de entidad. Al final del capítulo se muestra, paso a paso, el ejemplo del análisis de un requerimiento, esto le da al estudiante una mayor comprensión de la teoría antes vista, además de colocar otros requerimientos como parte de los ejercicios de tal capitulo.

El cuarto capítulo de este escrito está relacionado con la segunda unidad de competencias del plan de estudios. Éste capítulo muestra los dos modelos de implementación desarrollados en las décadas de los 60’s y de los 70’s, siendo éstos los modelos jerárquico y red. El modelo relacional fue conceptualizado por el Dr. Edgar Frank Codd en 1970 (Codd, Junio, 1970), y se implementó a principios de los 80’s, siendo el modelo de implementación más utilizado en este momento en las empresas. Se explican sus restricciones de integridad y se muestra la algorítmica necesaria para transformar el modelo Entidad Relación al modelo jerárquico, al modelo red y al modelo relacional, se da un ejemplo de transformación del modelo Entidad Relación a estos tres modelos y se explica la transformación del modelo Entidad Relación a los modelos jerárquico, red y relacional.

El capítulo quinto y el capítulo sexto de este escrito están relacionados con la cuarta unidad de competencias del plan de estudios En el capítulo cinco se muestran los aspectos teóricos (algebra relacional y cálculo relacional) para poder contestar consultas de una Base de Datos modelada bajo el paradigma relacional. Y el capítulo seis de este escrito muestra el lenguaje de consulta más utilizado por los Sistemas Gestores Relacionales (SQL). Al final del capítulo seis se coloca un ejercicio donde se muestra el requerimiento, el modelo conceptual, el modelo de implementación, una extensión de la Base de Datos y 68 consultas para que el estudiante pueda responderlas en algebra relacional, en cálculo relacional y en SQL. Es fundamental que el estudiante comprenda que, si no conoce plenamente los requerimientos y la forma de modelado, puede responder consultas sin sentido.

El capítulo siete de este escrito se relaciona con la tercera unidad de competencia del plan de estudios y está dedicado a la normalización. Se explican las posibles fallas que se pueden tener al momento de modelar una Base de Datos. Se explican, paso a paso, las formas normales de la primera a la quinta forma normal, incluyendo la forma Boyce Codd. Se explica un método para poder realizar en forma adecuada la normalización, se da un ejemplo de requerimiento y se explica con todo detalle su normalización, formando un árbol de normalización. Al final se colocan 4 requerimientos como ejercicio.

1. CONCEPTOS BÁSICOS DE BASES DE DATOS

1.1 REPRESENTACIÓN DEL CONOCIMIENTO

Según el Diccionario de la Real Academia Española, **representar** significa "hacer presente algo con palabras o figuras que la imaginación retiene", **Conocer** significa "Averiguar por el ejercicio de las facultades intelectuales la naturaleza, cualidades y relaciones de las cosas". (Real Académia Española)

La forma más adecuada para representar el conocimiento es mediante símbolos. Un símbolo es un número o cadena de caracteres que representan a un objeto o una idea. Nuestro lenguaje natural es bastante complejo para poder representar con precisión el conocimiento, entre otras cosas, por la *ambigüedad* inherente del mismo. Es muy difícil plasmar las diferentes interpretaciones de un enunciado en un algoritmo formal dado para que una máquina pueda interpretar en forma adecuada tal enunciado.
Es decir, la representación matemática y lógica han sido dos de las primeras formas de representación del lenguaje natural, muy cercanas a las máquinas, y por lo tanto, bastante alejadas del lenguaje natural.
Por otro lado, hay distintos tipos de conocimiento que se puede representar: desde los hechos simples hasta complejas relaciones, fórmulas matemáticas o reglas que siguen la sintaxis del lenguaje humano, asociaciones entre conceptos relacionados, inferencias y deducciones, establecimiento de jerarquías entre clases de objetos, etc. Cada tipo de conocimiento requiere una especificación tanto para su representación como para su tratamiento. En consecuencia, la elección de la forma de representación del conocimiento es una tarea crucial que deberá facilitar tanto la interpretación del mismo por parte de los humanos como su tratamiento por parte de las máquinas.
Una buena representación del conocimiento debería:

- Ser fácil de modificar por procedimientos manuales o mediante técnicas automáticas.
- Permitir la incorporación de nuevo conocimiento en forma sencilla.
- Facilitar la detección de incoherencias y falta de conciencia
- Posibilitar la reutilización de sentencias, procedimientos, etc.

Algunos de los paradigmas principales que representan el conocimiento en las máquinas se enumeran a continuación. En concreto, se van a enunciar tres grandes aproximaciones generales al tratamiento del conocimiento que están que están relacionados con distintos grados de relación:

1. Representación procedural y declarativa.
2. Representación relacional
3. Representación jerárquica.

1.2 CONOCIMIENTO PROCEDURAL Y DECLARATIVO

(Pagares Martìnez & Santos Peñas, 2006)
La representación procedural no sólo codifica hechos (constantes o variables acotadas), sino que también permite definir secuencias de operaciones para usar y manipular esos hechos. Por lo tanto, el algoritmo es fundamentalmente una forma de representar el

conocimiento. En la representación procedural, el conocimiento y su manipulación están indisolublemente unidos, no se pueden separar ni delimitar sus acepciones. Esto que podría ser considerado como una desventaja a la hora de formalizar los procedimientos, separándolos del conocimiento, ha venido a ser en parte subsanado por la programación declarativa. La representación declarativa del conocimiento permite expresar hechos, reglas y relaciones de forma independiente de su manipulación o procedimiento, y realmente representa conocimiento puro en este sentido.
La estructura tradicional conviene cuando la lógica del proceso está perfectamente consolidada, es decir, cuando como respaldo del programa está una teoría y un método de resolución perfectamente comprobados, que conduce a la solución a través de un proceso sistemático. En cambio, existen muchos ámbitos de la investigación, toma de decisiones, etc., en los que ese grado de certidumbre en los procedimientos no está presente. En este caso de ambigüedad en donde, por un lado, no es viable operar todas las opciones por la excesiva combinatoria y, por otro, no existe un conocimiento tan refinado que permita construir un proceso sistemático para obtener directamente la opción adecuada, resulta de interés una estructura "con conocimiento variable". Es decir, aquella en que puedan ensayarse distintas opciones de conocimiento para resolver uno o varios problemas relacionados y, en donde, en función del rendimiento y los resultados puede irse refinando el conocimiento, que puede llegar a alcanzar un grado de eficiencia comparable a un algoritmo específico de resolución.

CONOCIMIENTO RELACIONAL
Otra forma de representar el conocimiento es mediante **relaciones.** De hecho, se asemeja a la forma de almacenar la información de los seres humanos para algunas de sus aplicaciones, sobre todo en ámbitos comerciales, de negocios, etc. Es fuertemente dependiente del tipo de información con el que se trabaje.
El conocimiento se representa mediante tuplas de información de cada elemento (suceso, objeto, aplicación, proceso, etc.). Cada tupla contiene un número determinado de campos que define atributos específicos y valores de ese elemento, Almacenando una colección de información en una tabla, se pueden utilizar cálculos relacionales para manipular los datos, basándose relaciones definidas entre ellos, y así buscar información en la tabla.
Estas bases de datos suelen ser bastante flexibles: la incorporación de nuevo conocimiento se realiza en forma prácticamente inmediata, siguiendo una estructura predefinida. Sin embargo, no son muy adecuadas para la representación de relaciones complejas que se dan entre objetos o conceptos del mundo real. Por lo tanto se han propuesto otro tipo de conocimiento que se menciona a continuación.

CONOCIMIENTO JERÁRQUICO.
Los objetos o elementos que comparten una serie de características comunes se pueden asociar de forma natural en clases o grupos. Las relaciones y los atributos compartidos entre estos elementos son la base del conocimiento heredado. Es decir, es un tipo de conocimiento que trata con especificaciones comunes que se transmiten por un mecanismo de herencia, basado en una estructura jerárquica.
Este mecanismo de tratamiento de la información proporciona una forma muy compacta de representación del conocimiento y permite además algoritmos de razonamiento para procesar la información a distintos niveles de abstracción o granulidad.

Por ejemplo, un objeto general sería raqueta, se podría especificar como raqueta de ping-pong, raqueta de tenis, raqueta para la nieve, etc. A su vez se podría hablar de deportes con raqueta y de los atributos comunes de las raquetas, o a un nivel de detalle para especificar las características de un determinado deporte y raqueta utilizada en el mismo. La taxonomía o jerarquía de objetos o conceptos es realmente una forma muy útil y eficiente de organizar la información: reduce la complejidad y permite mayores niveles de abstracción.

Como se ha visto, existen diferentes formas de representar el conocimiento, puesto que hay diferentes tipos de conocimiento. Este curso se dedicará a analizar, modelar y manipular el conocimiento relacional.

1.3 DATO

(Luis, 2013)
El primer punto a tratar es el significado de dato. Dato, según el Diccionario de la Real Academia Española, es "el antecedente necesario para llegar al conocimiento exacto de algo o para deducir las consecuencias legitimas de un hecho". (Real Académia Española)

Según (Luis, 2013), dentro de Ciencias de la Computación los datos se dividen en tres categorías: En datos estructurados, datos semiestructurados y datos no estructurados.

La mayoría de las fuentes de datos tradicionales son datos estructurados, datos con formato o esquema fijo que poseen campos fijos. En estas fuentes, los datos vienen en un formato bien definido que se especifica en detalle. , y que conforma las bases de datos relacionales. Son los datos de las bases de dato relacionales, las hojas de cálculo y los archivos, fundamentalmente. Los datos estructurados se componen de piezas de información que se componen de antemano, vienen en formato específico, y se producen en un orden específico. Estos formatos facilitan el trabajo con dichos datos.

Los datos semiestructurados tienen un flujo lógico y un formato que puede ser definido, pero no es fácil su composición por el usuario. Datos que no tienen formatos fijos, pero contienen etiquetas y otros marcadores que permiten separar los elementos dato. La lectura de datos semiestructurados requiere el uso de reglas complejas que determinan cómo proceder después de la lectura de cada pieza de información. Un ejemplo típico de datos semiestructurados son los registros web logs de las conexiones a Internet. Un web log se compone de diferentes piezas de información, cada una de las cuales sirve para un propósito específico. Ejemplos típicos son el texto de etiquetas XML y HTML.

Los datos no estructurados son datos sin tipo predefinidos. Se almacenan como "documentos" u "objetos" sin estructura uniforme, y se tiene poco o ningún control sobre ellos. Datos de texto, video, audio, fotografías, documentos impresos, hojas electrónicas, imágenes digitales, formularios especiales, mensajes de correo electrónico y de texto, formatos de texto libre como correos electrónicos, mensajes instantáneos SMS, artículos, libros, mensajes de mensajería instantánea tipo WhatsApp, Line, Joyn, Vider, Line, Wechat, Spotbros. Al menos el 80% de la información de las organizaciones no reside en

las bases de datos relacionales, sino que se encuentran en espacios a lo largo y ancho de la organización, todos estos datos se conocen como datos no estructurados.

Dentro de la clasificación anterior, el curso tratará, principalmente, de un modelo estructurado. Por lo que el fin básico es dar a una base de datos un formato bien definido y que se comprenda plenamente el orden específico en que es creada.

1.4 Definición de Bases de Datos Estructurada.

(Elmasri & Navate, 2008)
(Date, 1993)
Bases de datos y la tecnología de bases de datos han tenido un gran impacto en el uso de las computadoras. Las bases de datos han jugado un rol muy importante en casi todas las áreas donde las computadoras han sido usadas, incluyendo el campo de los negocios, ingeniería, medicina, leyes, educación y biblioteconomía, por nombrar algunos. El mundo de las bases de datos hoy en día es de uso común y para introducirse en él es conveniente comenzar con una definición de este. La primera definición es en forma general.

Una base de datos es una colección de datos relacionados. Por datos, nosotros nos referimos a hechos conocidos que pueden ser recordados y que tienen un significado implícito. Por ejemplo, considere el nombre, número telefónico y dirección de los amigos que uno conoce. Usted puede tener registrados tales datos en un libro indexado, o en un disco, usando una PC y software tal como DBASE, PARADOX, EXCEL, ACCES, etc.

La definición precedente es muy general; por ejemplo, se pueden considerar la colección de palabras que forman este texto siendo tales palabras datos y de aquí que este texto constituye una base de datos. Sin embargo, el uso común del término de la base de datos tiene un uso más restringido. Una base de datos, según (Elmasri & Navate, 2008) tiene las siguientes propiedades implícitas:

- Una base de datos representa algún aspecto del mundo real, a veces llamado el mini mundo o el **universo de discusión**. Cambios al mini mundo se reflejan en la base de datos.
- Una base de datos es una colección lógicamente coherente de datos con un significado inherente.
- Una base de datos es diseñada, construida, y popularizada con datos para un propósito específico. Tiene un posible grupo de usuarios y alguna aplicación preconcebida en el cual estos usuarios están interesados.

En otras palabras, una base de datos tiene alguna fuente del cual los datos son derivados, algún grado de interacción con eventos en el mundo real, y una audiencia que está activamente interesada en el contenido de la base de datos.

Una base de datos puede ser de cualquier tamaño y de complejidad variable. Puede ser generada manualmente o por computadora.

Un sistema gestor de bases de datos (DataBase Management System-DBMS) es una colección de programas que permiten al usuario la creación y mantenimiento de una base de datos. El DBMS es un software de propósito general que facilita el proceso de definir, construir, y manipular bases de datos para varias aplicaciones. **Definir** una base de datos abarca especificar los tipos de datos, la estructura, y las restricciones de los datos que están almacenados en la base de datos. **Construir** una base de datos es el proceso de guardar los datos en un medio de almacenamiento que está controlado por el DBMS. **Manipular** una base de datos incluye cada una de las funciones que contiene el DBMS para recuperar datos específicos como las consultas, actualización de las bases de datos y generación de reportes.

No es necesario usar un DBMS de propósito general para implementar una base de datos en la computadora. Nosotros podemos escribir nuestro propio conjunto de programas para crear y mantener la base de datos. La figura 1.1 ilustra en forma general el contenido de un DBMS.

Un se denomina a **sistema de bases de datos** como la combinación de bases de datos y software DBMS. La figura 1.1 ilustra algunos de los conceptos que se han explicado hasta este momento.

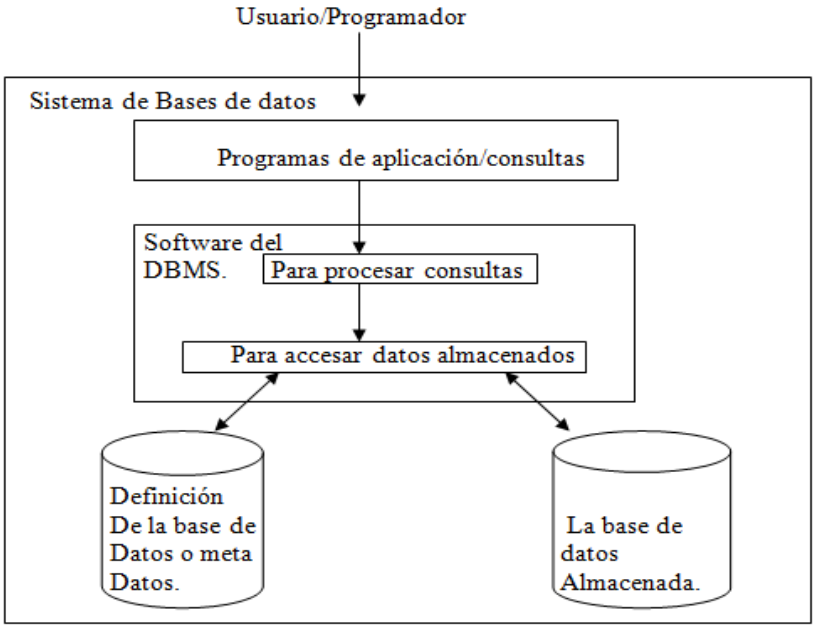


Fig. 1.1. Entorno de un sistema de bases de datos simplificado.

1.5 CARACTERISTICAS DE LA METODOLOGÍA DE LAS BASES DE DATOS.

(Elmasri & Navate, 2008)

Existen ciertas características que distinguen a una base de datos con respecto al enfoque tradicional de programación con archivos. En programación tradicional de archivos, cada usuario define e implementa las necesidades de los archivos de una aplicación específica. Por ejemplo, un usuario, la oficina de control escolar puede tener

un archivo de estudiantes y sus calificaciones. Se desarrollarán programas de aplicación para la impresión de las calificaciones de los estudiantes, introducir nuevas calificaciones o dar de baja a los estudiantes. Otro usuario. La oficina de contabilidad, puede tener un archivo con los pagos realizados por cada estudiante. Aunque ambos usuarios están interesados en los estudiantes, cada usuario tendrá archivos separados y programas para manipular tales datos. Ya que cada usuario almacenará datos que no son importantes para el otro usuario. Esta redundancia en definir y guardar los datos da como resultado una pérdida de espacio y un doble esfuerzo para mantener datos comunes actualizados.

Dentro de la filosofía de las bases de datos, existirá un solo almacén de datos. Las principales características de la metodología de una base de datos con respecto a un manejador de archivos es:

1. *Una base de datos se auto describe.* Una característica fundamental de una base de datos es que el sistema de la base de datos contiene no sólo los datos, sino también una completa definición o descripción de la base de datos. Esta definición se almacena en el catálogo del sistema, el cual contiene información tal como la estructura de cada archivo, el tipo y el formato de almacenamiento de cada dato, y varias restricciones de los datos. La información guardada en el catálogo se conoce como meta-dato.

   El catálogo es usado por el software del DBMS y ocasionalmente por el usuario de la base de datos. El software debe trabajar adecuadamente en aplicaciones diferentes como son una base de datos para una universidad, una base de datos bancaria, etc.,

   En programación de archivos tradicionales, la definición de los datos es parte del programa de aplicación. De aquí que estos programas están restringidos a trabajar con solo una base de datos específica, cuya estructura se declara en el programa de aplicación. Por lo que en la programación orientada a archivos sólo pueden tener acceso a bases de datos específicas, el software de un DBMS puede tener acceso a diversas bases de datos extrayendo la información de la base de datos en el catálogo.

2. *Diferencias entre programas, datos y abstracción de los datos.* La estructura de los datos en los programas tradicionales de archivos se define dentro del programa, por lo que cualquier cambio en la estructura del archivo puede requerir el cambio de todo el programa donde se requiera el acceso al archivo. Por el contrario, los programas de acceso en DBMS se escriben en forma independiente a los archivos. A esta propiedad se conoce como **independencia de datos con respecto al programa**. Por ejemplo, un programa donde se tiene acceso un archivo llamado estudiante en el cual solo tiene un registro que indica el nombre del estudiante con un tamaño de 40 caracteres. Si se desea introducir mayor información por estudiante como su día de nacimiento, el programa ya no funciona y debe ser cambiado. En contraste, en un ambiente de un DBMS, sólo se tiene que cambiar la descripción de los registros en el catálogo; sin que se requiera otro cambio dentro del programa.

   El desarrollo reciente en bases de datos orientados a objetos y lenguajes de programación permite a los usuarios definir operaciones sobre datos como parte de la definición de la base de datos. Una operación (también conocida como función) es especificada en dos partes. La interface de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación o método de la operación se especifica separadamente y puede ser cambiada sin

afectar la interface. Los programas de aplicación de los usuarios pueden realizar operaciones con respecto a algunos datos específicos invocando cualquiera de las operaciones invocándolas por su nombre y los argumentos necesarios, sin darle importancia de cómo han sido implementadas las operaciones. A esta característica se le conoce como **operaciones independientes de la programación.**

Las características que permiten tener independencia de datos e independencia de operaciones con respecto al programa se conoce como **abstracción de datos.** Un DBMS provee a los usuarios una representación conceptual de los datos donde no incluye los detalles de cómo los datos son guardados. Informalmente, un **modelo de datos** es un tipo de abstracción de datos que es usado para proveer esta representación conceptual. El modelo de datos usa conceptos lógicos, tales como objetos, sus propiedades, y su interrelación, que puede ser más fácil de entender para la mayoría de los usuarios que los conceptos del almacenamiento de los datos. De aquí, que el modelo de datos oculte los detalles de almacenamiento que no son de interés para la mayoría de los usuarios.

Dentro de las bases de datos, los detalles de la estructura y organización de cada archivo se almacenan en el catálogo. Los usuarios de las bases de datos manejan la parte conceptual de los archivos, y el DBMS extrae los detalles del archivo del catálogo cuando sean requeridos por el software del DBMS. Existen varios modelos de datos que proveen la abstracción de los datos para un usuario en una base de datos tales como el modelo jerárquico, el modelo red, el modelo relacional, el modelo orientado a objetos y el modelo deductivo.

3. *Soporte de vistas múltiples de datos.* Una base de datos típicamente tiene varios usuarios, cada usuario requerirá diferentes perspectivas o **vistas** de la base de datos. Una vista puede ser un subconjunto de la base de datos o puede contener **datos virtuales** que son derivados de los archivos de la base de datos pero no son guardados en forma explícita en los archivos. Un DBMS multiusuarios puede tener clientes con una variedad de aplicaciones y debe de proveer facilidades para definir vistas múltiples.
4. *Compartimiento de datos en un ambiente multiusuarios y manejo de transacciones.* Un DBMS multiusuarios, como su nombre lo indica, debe permitir acceso a la base de datos al mismo tiempo a varios usuarios. El DBMS debe incluir software para control de concurrencia para asegurarse de que varios usuarios al tratar de modificar los mismos datos, éstos se han modificados en forma correcta. Un ejemplo es cuando varias agencias a la vez tratan hacer reservaciones para un vuelo específico, el DBMS tendrá que asegurarse de que cada cliente tenga su asiento en forma adecuada y no sean vendidos dos boletos para el mismo asiento.

1.6 ACTORES DE LA ESCENA

(Elmasri & Navate, 2008)

En esta sección se identifica a las personas cuyo trabajo diario implica el uso diario de una base de datos empresarial, se les denominará *actores de la escena*. En la siguiente sección se hablará de los *trabajadores entre bambalinas* (los que trabajan en el mantenimiento el entorno del sistema de bases de datos pero que no están interesados en la propia base de datos).

1. *El administrador de bases de datos* (DBA, DataBase Administrator) es el responsable del acceso autorizado a la base de datos, de la coordinación y monitorización de su uso, y de adquirir los recursos software y hardware necesarios. También es responsable de problemas como la brecha de seguridad o de tiempos de respuesta pobres. En empresas grandes, el DBA está asistido por un equipo de personas que llevan a cabo estas funciones.
2. *Los diseñadores de la base de datos* son los responsables de identificar los datos que se almacenarán en la base de datos y de elegir la estructura apropiada para representar y almacenar los datos. Es responsabilidad de los diseñadores comunicarse con todo presunto usuario de la base de datos para conocer sus requerimientos, a fin de crear un diseño que satisfaga sus necesidades. Estos diseñadores interactúan con los grupos de usuarios potenciales y desarrollan **vistas** de las bases de datos que satisfacen los requisitos de datos y procedimientos de esos grupos. Cada vista se analiza y se integra con las vistas de los otros grupos de usuarios. El diseño final de la base de datos debe ser capaz de soportar los requisitos de todos los grupos de usuarios.
3. *Los usuarios finales* son personas cuyos trabajos requieren acceso a las bases de datos para realizar consultas, actualizaciones e informes; las bases de datos existen principalmente para ser utilizadas.
4. *Los analistas de sistemas y programadores de aplicaciones (ingenieros de software* determinar los requisitos de los usuarios finales, así como las especificaciones de desarrollo para las transacciones enlazadas que satisfacen esos requisitos. Los programadores de aplicaciones implementan esas especificaciones como programas; después, verifican, depuran, documentan y mantienen esas transiciones. Dichos analistas y programadores (normalmente conocidos como desarrolladores de software o ingenieros de software) deben familiarizarse con todas las posibilidades proporcionadas por el DBMS al objeto de desempeñar sus tareas.

1.7 TRABAJADORES ENTRE BAMBALINAS.

(Elmasri & Navate, 2008)

Además de los que diseñan, utilizan y administran una base de datos, hay otros usuarios que están asociados con el diseño, el desarrollo y funcionamiento de un entorno de software y sistema DBMS. Estas personas normalmente no están interesadas en la base de datos propiamente dicha y se dividen en la siguiente categoría:

- *Diseñadores e implementadores de sistemas DBMS.* Diseñan e implementan los módulos y las interfaces DBMS como un paquete software. Un DBMS está compuesto por módulos que incluyen: el catálogo, el lenguaje de consultas, la interfaz, acceder y almacenar los datos en un búfer, controlar la concurrencia y manipular la recuperación y la seguridad de los datos. El DBMS debe interactúa con otros software del sistema, como el sistema operativo, los compiladores y los diversos lenguajes de programación.
- *Desarrolladores de herramientas.* Diseñan e implementan herramientas. Las herramientas son paquetes opcionales que a menudo se compran por separado. Entre ellas se puede citar a los paquetes para el diseño de bases de datos, la

monitorización del rendimiento, las interfaces gráficas, el prototipado, la simulación, la generación de datos de prueba.

- *Operadores y personal de mantenimiento.* Son responsables de la ejecución y mantenimiento real del entorno hardware y software para el sistema de bases de datos.

Aunque estos trabajadores se encargan de que el sistema de bases de datos esté disponible para los usuarios finales, normalmente no utilizan la base de datos para sus fines propios.

1.8 VENTAJAS DE UTLIZAR METODOLOGÍA DBMS.

(Elmasri & Navate, 2008)

En esta sección se analizará alguna de las ventajas de utilizar un DBMS y que capacidades deben ofrecer.

1. *Control de redundancia.* En la creación tradicional de programas con procesamiento de archivos, cada grupo de usuarios mantiene sus propios archivos para manejar sus aplicaciones de procesamientos de datos. Una buena parte de datos se duplicaría, un dato almacenado por cada grupo de usuarios.
   A veces, y no pocas, esta **redundancia** en el almacenamiento de los datos provoca varios problemas. En primer lugar, es necesario realizar una misma actualización lógica varias veces. Esto implica una duplicación de trabajo. En segundo lugar, se desperdicia espacio de almacenamiento al guardar los mismos datos en varios lugares. En tercer lugar, es posible que los archivos que representan los mismos datos se tornen inconsistentes, quizá porque una actualización se haya aplicado a ciertos archivos pero no a otros.
   Con el enfoque de bases de datos, las vistas de los diferentes grupos de usuarios se integran durante el diseño de la base de datos. Para conservar la consistencia, debe crearse un diseño que almacene cada dato lógico en un solo lugar de la base de datos. Ello evita la inconsistencia y ahorra espacio de almacenamiento.

2. *Restricción de acceso no autorizado.* Cuando muchos usuarios comparten una misma base de datos, es probable que no todos tengan la autorización para tener acceso a toda la información que contiene. Además, es posible que sólo algunos usuarios tengan permiso para recuperar datos, en tanto que a otros se les permita obtenerlos y actualizarlos. Por lo que es necesario que se controle el tipo de operaciones de acceso (obtención y actualización). Por lo regular, a los usuarios o grupos de usuarios se les asignan números de cuenta protegidos con contraseña, mismos que sirven para tener acceso a la base de datos. El DBMS debe contar con un subsistema de **seguridad y autorización** que permita al administrador de la base de datos crear cuentas y especificar restricciones para ellos. El DBMS deberá entonces obligar automáticamente al cumplimiento de dichas restricciones. Cabe señalar que el mismo tipo de controles se puede aplicar al software del DBMS. Por ejemplo, sólo el personal administrativo del sistema tendrá autorización para utilizar cierto software **privilegiado,** como el que sirve para crear cuentas nuevas.

3. *Cumplimiento de las restricciones de integridad.* La mayor parte de las aplicaciones de base de datos tienen que cumplir con ciertas restricciones de integridad que deben cumplir los datos. El DBMS debe ofrecer recursos para definir tales restricciones y hacer que se cumplan.
   Es posible introducir erróneamente un dato sin violar las restricciones de integridad. Por ejemplo, si un estudiante obtiene una calificación de 10 pero se introduce 6 en la base de datos, el DBMS no podrá descubrir el error automáticamente, porque 6 es permitido en el tipo de datos calificación. Esta clase de errores sólo puede descubrirse manualmente (cuando el estudiante reciba su boleta de calificaciones y se queje) y corregirse después actualizando la base de datos. Por otro lado, el DBMS puede rechazar automáticamente una calificación de X, porque éste no es valor permitido para el tipo de datos de calificación.

4. *Respaldo y recuperación.* Todo DBMS debe contar con recursos para recuperarse de fallos de hardware o de software. Para ello está el subsistema de respaldo y recuperación del DBMS.

1.9 CUÁNDO NO USAR UN DBMS.

(Elmasri & Navate, 2008)
A pesar de las ventajas de usar un DBMS, hay algunas situaciones en las que su uso puede suponer unos sobrecostos. Los sobrecostos de utilizar un DBMS se deben a lo siguiente:

- Inversión inicial muy alta en hardware, software y formación.
- Costos derivados de las funciones de seguridad, control de la concurrencia, recuperación e integridad.

Es posible que surjan otros problemas si los diseñadores y el DBA no diseñan correctamente la base de datos o si las aplicaciones de sistemas de bases de datos no se implementan correctamente. Por tanto, no puede ser más deseable utilizar archivos normales en las siguientes circunstancias.

- Aplicaciones de bases de datos sencillas y bien definidas que no es previsible que cambien.
- Requisitos estrictos y en tiempo real para algunos programas que no podrían satisfacer debido al sobrecosto de un DBMS.
- Inexistencia de acceso multiusuario a los datos.

Algunas aplicaciones prefieren no utilizar DBMS de propósito general. Por ejemplo, las herramientas de diseño asistido por computadora (CAD) tienen archivos propietario y software de administración de datos destinados a la manipulación interna de dibujos y objetos 3D. Los sistemas de comunicación y conmutación, las implementaciones GIS a menudo implantan sus propios esquemas de organización de datos para el procesamiento de mapas.

Preguntas de repaso.

1. De una explicación de lo que es conocimiento.

2. Explique lo que es un dato.
3. Defina los siguientes términos: Bases de datos, DBMS, sistemas de bases de datos, catálogo de la base de datos, vista de usuario, DBA, usuario final, metadatos.
4. Comente la diferencia entre bases de datos y metabases de datos
5. Explique claramente el concepto de abstracción de datos
6. Compare la tecnología de archivos con la tecnología de bases de datos.
7. Explique la importancia de las restricciones de integridad.
8. Explique los diferentes tipos de bases de datos.
9. Explique los diferentes tipos de usuarios y sus actividades principales
10. Explique las capacidades que un DBMS debe proporcionar
11. Comente la diferencia entre redundancia y ambigüedad.

2. CONCEPTOS DE LOS SISTEMAS GESTORES DE BASES DE DATOS (DBMS. DataBase Management Systems).

Una característica fundamental del enfoque de bases de datos es que proporciona cierto nivel de abstracción de los datos al ocultar detalles de almacenamiento que la mayoría de los usuarios no necesitan conocer. Los modelos de datos son el principal instrumento para ofrecer dicha abstracción. Un modelo de datos es un conjunto de conceptos que pueden servir para describir la estructura de una base de datos. Con el concepto de estructura de una base de datos nos referimos a los tipos de datos, los tipos de relación y las restricciones que deben cumplir para esos datos. Por lo regular, los modelos de datos contienen además un conjunto de operaciones básicas para especificar lecturas y actualizaciones de la base de datos. Casi siempre el modelo de datos básico cuenta con operaciones genéricas para insertar, eliminar, modificar o recuperar un objeto.

2.1 CATEGORIAS DE LOS MODELOS DE DATOS.

(Elmasri & Navate, 2008)

Se han propuesto muchos modelos de datos, y podemos clasificarlos dependiendo de los tipos de conceptos que ofrecen para describir la estructura de las bases de datos. Los modelos de datos de alto nivel o **conceptuales** disponen de conceptos muy cercanos al modelo como la generalidad de los usuarios percibe los datos, en tanto que los modelos de bajo nivel o **físicos** proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el computador. Los modelos de datos de bajo nivel casi siempre están dirigidos a los especialistas en computación. Entre estos dos extremos hay una clase de modelos de datos de **implementación**, cuyos conceptos pueden ser entendidos por los usuarios finales aunque no están alejados de la forma en que los datos se organizan dentro del computador.

Los modelos de datos de alto nivel utilizan conceptos como entidades, atributos y tipos de relación. El modelo más popular es el modelo Entidad-tipo de relación que será explicado posteriormente.

Los modelos de datos de implementación son los más usados en los DBMS comerciales actuales, los modelos más populares son: jerárquico, red y relacional.

Los modelos de datos físicos describen cómo se almacenan los datos en el computador, al representar información como los formatos y ordenamientos de los registros y los caminos de acceso.

2.2 ESQUEMAS, INSTANCIAS Y ESTADO DE LA BASE DE DATOS.

(Elmasri & Navate, 2008)

En cualquier modelo de datos es importante distinguir entre la *descripción* de la base de datos y la *misma base de datos*. La descripción de la base de datos se denomina **esquema de la base de datos**, que se especifica durante la fase de diseño y no se espera que cambie con frecuencia.

Los datos reales de una base de datos pueden cambiar con mucha frecuencia y se denominan como **estado de la base de datos** o **instancia** de la base de datos. Cada esquema tiene su conjunto de instancias.

Esta distinción entre esquema y estado de una base de datos es muy importante. Cuando definimos una base de datos nueva sólo especificamos su esquema teniendo en este momento un estado vacío. El estado inicial es cuando se carga por primera vez con los datos iniciales. Desde este momento, cada vez que sobre la base de datos se aplique una operación de actualización, obtendremos otro estado de la base de datos. El DBMS es en parte responsable de garantizar que cada estado de la base de datos sea un estado válido; es decir, un estado que cumpla con la estructura y restricciones especificas en el esquema. Para esto el DBMS almacena las descripciones de las construcciones de esquema y las restricciones (también denominadas **metadatos**) en el catálogo del DBMS. En ocasiones, el esquema recibe el nombre de **intención** y el estado de la base de datos **extensión** del esquema.

2.3 ARQUITECTURA DE TRES ESQUEMAS.

(Elmasri & Navate, 2008)

El objetivo de la arquitectura de tres esquemas es separar las aplicaciones del usuario y las bases de datos físicas. Hay tres características importantes inherentes al enfoque de las bases de datos que son (a) la separación entre los programas y los datos (independencia con respecto a los programas y datos y con respecto a los programas y operaciones): (b) el manejo de múltiples vistas del usuario, y (c) el empleo de catálogos para almacenar la descripción o el esquema de la base de datos. Con estas tres características se formó una arquitectura que se conoce como la arquitectura de tres esquemas o arquitectura ANSI/SPARC, por el comité que la propuso.

El objetivo de la arquitectura de tres esquemas consiste en formar una separación entre las aplicaciones del usuario y la base de datos física. En esta arquitectura se pueden definir los tres niveles siguientes:

- El nivel interno describe la estructura física de almacenamiento de la base de datos. El esquema interno emplea un modelo físico de datos y describe todos los detalles para su almacenamiento, así como los caminos de acceso para la base de datos.
- El nivel conceptual tiene un esquema conceptual que describe la estructura de toda la base de datos para una comunidad de usuarios. En este nivel se puede usar un modelo de datos de alto nivel o uno de implementación.
- El nivel externo o vista del usuario incluye varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinado, y oculta a ese grupo el resto de la base de datos. En este nivel podemos usar modelos de alto nivel o implementación.

En un DBMS basado en la arquitectura de tres esquemas, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo; por lo tanto, el DBMS debe transformar una solicitud expresada en términos de un esquema externo a una solicitud expresada en términos del esquema conceptual, y luego a una solicitud en el esquema interno que se procesará sobre la base de datos almacenada. Si la solicitud es una obtención de datos, será preciso modificar el formato de la información extraída de la base de datos almacenada para que coincida con la vista externa del usuario. El proceso de transformar solicitudes y resultados de un nivel a otro se denomina **transformación** (mapping)

2.3.1 Independencia con respecto a los datos.

La independencia con respecto a los datos se puede definir como la capacidad para modificar el esquema en un nivel sin tener que modificar el esquema del nivel inmediato superior. Los dos tipos de independencia son:

1. Independencia lógica. Es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos (añadiendo un atributo a una relación), o para reducir la base de datos (eliminando un atributo o más a una relación). Si el DBMS cuenta con independencia lógica con respecto a los datos, sólo será preciso modificar la definición de la vista y las correspondencias. Además, las restricciones podrán modificarse en el esquema conceptual sin afectar los esquemas externos.
2. Independencia física. Es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual. Tal vez sea necesario modificar el esquema interno por la necesidad de reorganizar ciertos archivos físicos a fin de mejorar el rendimiento de las operaciones de obtención o actualización. Si la base de datos aún contiene los mismos datos, no deberá ser necesario modificar el esquema conceptual. Dado que la independencia física con respecto a los datos se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de lograr que la independencia lógica con respecto a los datos.

2.4 LOS TIPOS DE LENGUAJES USADOS EN LAS BASES DE DATOS.

(Elmasri & Navate, 2008)

Una vez que se ha completado el diseño de una base de datos y se ha elegido un DBMS para su implementación, el primer paso será especificar los esquemas conceptual e interno de la base de datos y cualquier correspondencia entre ambos. En muchos DBMS en los que no se mantiene una separación estricta de niveles, los diseñadores de la base de datos utilizan un mismo lenguaje.

- *Lenguaje de definición de datos.* (DDL: data definition language), el DBMS contará con un compilador de DDL cuya función será procesar enunciados escritos en DDL para identificar las descripciones de los elementos de los esquemas y almacenar la descripción del esquema en el catálogo del DBMS. Cuando en los DBMS se mantenga una clara separación entre niveles conceptuales e interno, el DDL servirá solamente para especificar el esquema conceptual.
- *Lenguaje de definición de almacenamiento*. (SDL: storage definition language), para especificar el esquema interno.
- *Lenguaje de definición de vistas*. (VDL: view definition language) Las correspondencias entre los esquemas se pueden especificar en cualquiera de los dos lenguajes. Para una verdadera arquitectura de tres esquemas, se requiere un tercer lenguaje, el lenguaje de definición de vistas, para especificar las vistas del usuario y sus correspondencias con el esquema conceptual.

- *Lenguaje de manipulación de datos*. (DML: data manipulation language) Una vez que se han compilado los esquemas de la base de datos y que en ésta se han introducido datos, los usuarios requerirán algún mecanismo para manipularla. Las operaciones de manipulación más comunes son la obtención, la inserción, la eliminación y la modificación de datos. En este caso, el DBMS ofrece el lenguaje de manipulación de datos para estos fines.

En los DBMS actuales no se acostumbra distinguir entre los tipos de lenguajes mencionados; más bien, se utiliza un amplio lenguaje integrado que cuenta con elementos para definir esquemas conceptuales, definir vistas, manipular datos y definir su almacenamiento. Un ejemplo representativo es el lenguaje de bases de datos relacional SQL, que representa una combinación de DDL, VDL, DML. SDL era un componente de las primeras versiones de SQL, pero se ha eliminado del lenguaje para mantenerlo únicamente en los niveles conceptual y externo.

Hay dos tipos principales de DML. Se puede utilizar un DML de alto nivel o no procedimental para representar en forma concisa las operaciones de las bases de datos. Este lenguaje de alto nivel puede ser utilizado desde el monitor o incrustadas en programación de propósito general. Un lenguaje DML de alto nivel, como lo es SQL, puede especificar o recuperar registros con una sola sentencia.; por lo tanto se conocen también como ***set-at-time*** o ***set-oriented***. Una consulta de en un DML de alto nivel a menudo especifica los datos que hay que recuperar, en lugar de cómo recuperarlos; en consecuencia, dichos lenguajes también se conocen como **declarativos.** Un DML de bajo nivel o procedimental debe incrustarse en un lenguaje de programación de propósito general. Normalmente éste tipo de lenguaje recupera registros individuales de una base de datos, y los procesa por separado. Por consiguiente es preciso utilizar construcciones de un lenguaje de programación estructurado como, como bucles, para recuperar y procesar cada registro de un conjunto de registros. Los DML de bajo nivel también se conocen como ***record-at-a-time***.

Siempre que hay comandos DML, de alto o bajo nivel, incrustados en un lenguaje de propósito general conocido como ***lenguaje host***. Al DML se le conoce como ***sub lenguaje de datos.*** A un lenguaje de alto nivel utilizado en forma interactiva independiente se le conoce como ***lenguaje de consulta***.

2.5 Entorno de un Sistema de Bases de Datos.

(Elmasri & Navate, 2008)

En la figura 2.5.1 se ilustran los componentes típicos de un DBMS. En la mitad superior se refiere a los diversos usuarios de la base de datos y sus interfaces; la mitad inferior muestra las entradas del DBMS responsable del almacenamiento de los datos y el procesamiento de transacciones.

La parte superior de la figura muestra las interfaces para el personal del DBA, los usuarios casuales que trabajan con interfaces interactivas para formular consultas, los programadores de aplicaciones que programan utilizando algunos lenguajes *host* y los usuarios paramétricos que realizan las entradas de datos suministrando parámetros a las transacciones predefinidas. El personal del DBA trabaja en definir la base de datos y refinarla introduciendo cambios en su definición mediante el DDL y otros comandos privilegiados.

El compilador DDL procesa las definiciones de esquema, especificadas en el DDL y almacena las descripciones de los esquemas (metadatos) en el catálogo del DBMS. El catálogo incluye el nombre y tamaño de archivos, nombre y tipo de datos de los elementos de datos, los detalles de almacenamiento de cada archivo, la información que mapea entre esquemas y las restricciones, además de muchos otros tipos de información que los módulos del DBMS necesita.
Un compilador de consultas analiza las consultas lexicográficas y sintácticamente para garantizar la corrección de las operaciones de los modelos, consulta el catálogo del sistema para información estadística y física acerca de los datos almacenados, y genera un código ejecutable que lleva a cabo las operaciones necesarias para la consulta.

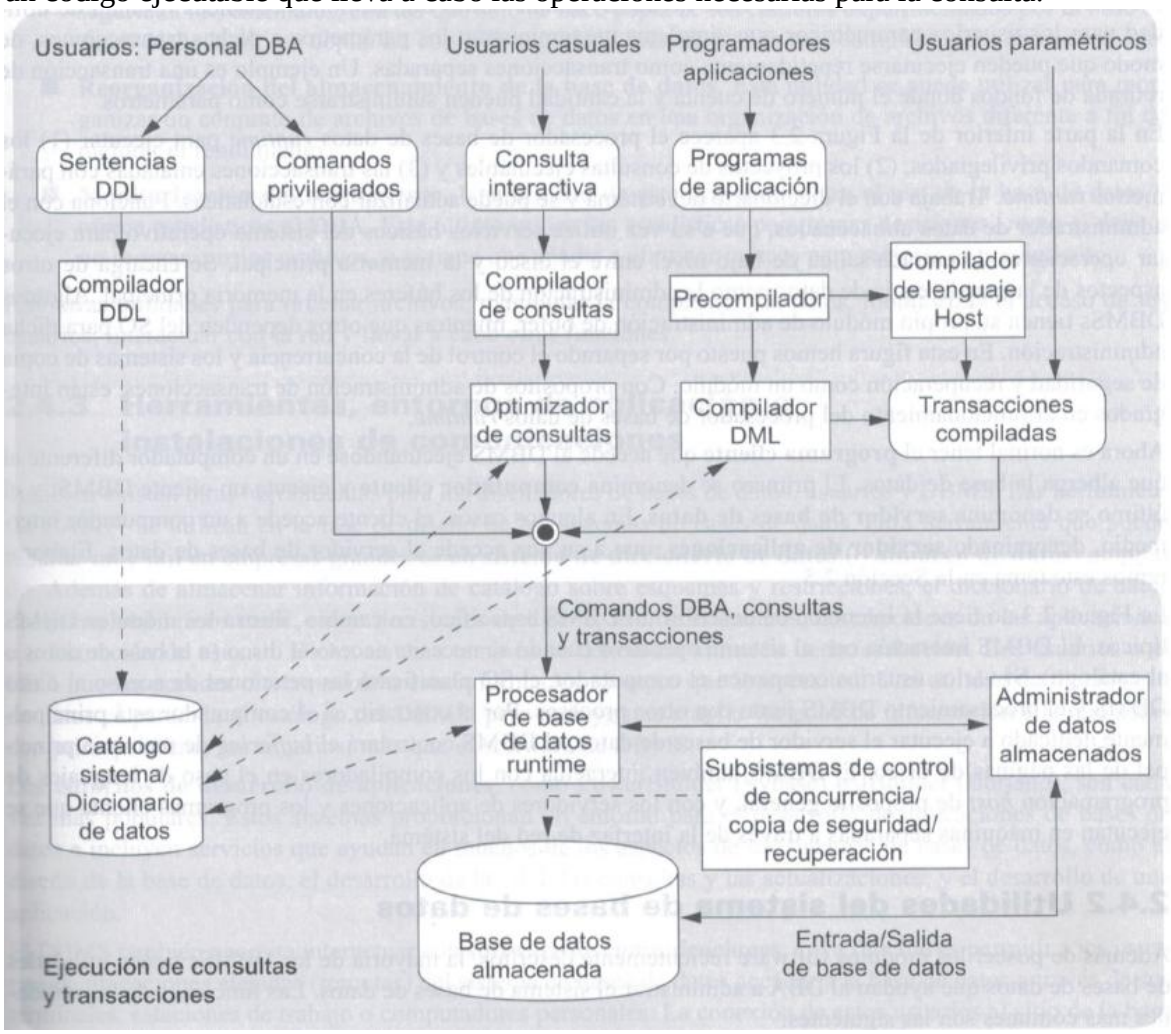


Figura 2.5.1. Módulos constituyentes de un DBMS y sus interrelaciones

Los programadores de aplicaciones escriben programasen lenguaje *host* como Java, C o COBOL., que son enviados a un pre compilador. Éste extrae los comandos DML de un programa de aplicación escrito en un lenguaje de programación *host.* Estos comandos se envían al compilador DML para su compilador e código objeto y así poder acceder a la base de datos.

2.6 Clasificación de los DBMS.

(Elmasri & Navate, 2008)

- Por el modelo de datos. Los modelos de datos empleados con mayor frecuencia en los DBMS comerciales son el relacional, el de red y el jerárquico. Algunos DBMS recientes se basan en el modelo orientado a objetos y bases de datos deductivas.
- Otro criterio para clasificar los DBMS es el número de usuarios a los que da servicio el sistema. Los sistemas mono usuarios sólo atiende a un usuario a la vez, y su principal uso se da en los computadores personales. Los sistemas multiusuarios, entre los que se cuentan la mayor parte de los DBMS, atienden a varios usuarios al mismo tiempo.
- Un tercer criterio es el número de sitios en los que está distribuida la base de datos. Casi todos los DBMS son centralizados; esto es, sus datos se almacenan en un sólo computador. Los DBMS centralizados pueden atender a varios usuarios, pero el DBMS y la base de datos en sí residen por completo en un solo computador. En los DBMS distribuidos la base de datos real y el software del DBMS pueden estar distribuidos en varios sitios, conectados por una red de computadores.
- Los DBMS homogéneos utilizan el mismo software de DBMS en múltiples sitios. Una tendencia reciente consiste en crear software para tener acceso a varias bases de datos autónomas preexistentes almacenadas en DBMS heterogéneos. Esto da lugar a los DBMS federados (o sistemas multibase de datos) en los que los DBMS participantes están débilmente acoplados y tienen cierto grado de autonomía local
- Otro criterio es el costo del DBMS. La mayor parte de los DBMS cuestan entre $10,000.00 y $100,000.00 dlls estadounidenses. Los sistemas mono usuario más económicos para computadoras cuestan entre $100.00 y $300.00 dlls. (en México se localizan desde gratis por la cultura a la piratería).
- Por último, se pueden clasificar por los tipos de camino de acceso del que disponen para almacenar los archivos. Los DBMS pueden ser de propósito general o de propósito especial. Cuando el rendimiento es de gran importancia, se puede diseñar y construir un DBMS de propósito especial, y pertenecer a la categoría de sistemas de procesamiento de transacciones en línea (OLTP: on-line transaction processing) que deben atender un gran número de transacciones concurrentes sin importar retrasos excesivos.

Preguntas de repaso

1. Defina los siguientes términos: Modelo de datos, esquema de bases de datos, esquema interno, esquema conceptual, esquema externo, independencia de datos, DDL, DML, SDL, VDL, arquitectura de tres capas.
2. Explicar las categorías de modelos de bases de datos.
3. Comentar la diferencia entre independencia física de datos e independencia lógica de datos
4. Comente la figura 2.5.1
5. Explique la clasificación de los DBMS.

3 UN MODELO CONCEPTUAL.

(Elmasri & Navate, 2008)
(de Miguel Castaño, Piattini Velthuis, & Marcos Martinez, 2000)
La metodología de diseño de bases de datos incluye cada vez más conceptos que sirven para especificar las operaciones con objetos de bases de datos, y como las metodologías de ingeniería de software especifican la estructura de las bases de datos que los programas utilizan y a las que accederán, es evidente que estas actividades están estrechamente relacionadas.
En este capítulo se muestra la metodología tradicional de concentrarse en las estructuras y las restricciones de la base de datos durante el diseño de esta última. Se presentan los conceptos del modelo Entidad- Relación (ER), siendo éste un modelo conceptual se describen los conceptos básicos de las estructuras de datos y las restricciones del modelo ER, así como su uso en el diseño de esquemas conceptuales para las aplicaciones de las bases de datos.

3.1 Uso de modelos de datos conceptuales de alto nivel

En la figura 3.1 se muestra una descripción simplificada del proceso del diseño de bases de datos.

- El primer paso es la recolección y análisis de requerimientos, durante la cual los diseñadores entrevistan a los futuros usuarios de la base de datos para entender y documentar sus requerimientos de información. El resultado de este paso será un conjunto de requerimientos del usuario redactado en forma concisa. Estos requerimientos deben especificarse en forma lo más detallada y completa como sea posible. En paralelo con la especificación de los requerimientos de datos, conviene especificar los requerimientos funcionales conocidos de la aplicación. Estos consisten en las operaciones definidas por el usuario (o transacciones) que se aplicarán a la base de datos, e incluyen la obtención de datos y la actualización. Se acostumbra usar técnicas como los diagramas de flujo de datos para especificar los requerimientos funcionales.
- Una vez recabados y analizados todos los requerimientos, el siguiente paso es crear un esquema conceptual para la base de datos mediante un modelo de datos de alto nivel. Este paso se denomina diseño conceptual de la base de datos. El esquema conceptual es una descripción concisa de los requerimientos de información de los usuarios, y contiene una descripción detallada de los tipos de datos, los tipos de relación y las restricciones; éstas se expresan mediante los conceptos del modelo de datos de alto nivel. Puesto que estos conceptos no incluyen detalles de implementación, suelen ser más fáciles de entender, de modo que pueden servir para comunicarse con usuarios no técnicos. También puede servir como referencia para asegurarse de satisfacer todos los requerimientos de los usuarios y de que no haya conflictos entre dichos requerimientos.
- Una vez diseñado el esquema conceptual, es posible utilizar las operaciones básicas del modelo de datos para especificar transacciones de alto nivel que correspondan a las operaciones definidas por el usuario que se hayan identificado durante el análisis funcional. Esto también sirve para confirmar que el esquema conceptual satisfaga todos los requerimientos funcionales identificados. Se puede modificar en este

momento el esquema conceptual si no resulta factible especificar algunos requerimientos funcionales en el esquema inicial.

- El siguiente paso en este proceso de diseño consiste en implementar, de hecho, la base de datos en un DBMS comercial. La mayoría de los DBMS disponibles hoy en el mercado utilizan un modelo de datos de implementación, así que el esquema conceptual se traduce del modelo de datos de alto nivel al modelo de datos de implementación. Este paso se denomina diseño lógico de la base de datos o transformación de modelos de datos, y su resultado es un esquema de base de datos específico en el modelo de datos de implementación del DBMS.
- El paso final es la fase de diseño físico de la base de datos, durante la cual se especifican las estructuras de almacenamiento internas y la organización de los archivos de la base de datos. En paralelo con estas actividades, se diseñan e implementan programas de aplicación en forma de transacciones de base de datos que correspondan a las especificaciones de transacciones de alto nivel.

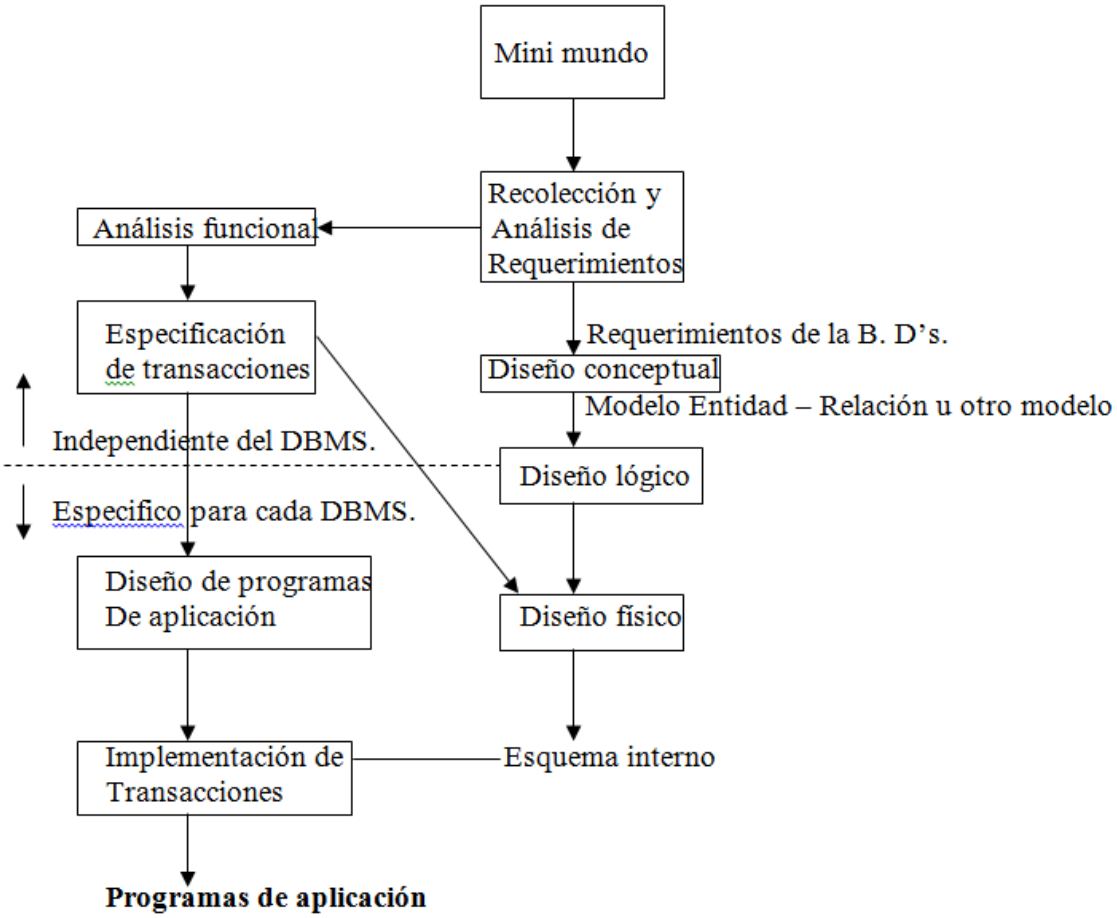


Fig. 3.1 Fases del diseño de una B. D.'s. (Elmasri & Navate, 2008)

3.2 El modelo Entidad – Relación (ER).

El modelo ER describe los datos en los tipos de entidades y los diferentes tipos de relación entre los tipos de entidades y atributos pertenecientes a los tipos de relación

3.2.1 Entidades y atributos.

El objetivo básico que se representa en el modelo ER es la **entidad**: un objeto del mundo real con existencia independiente. Una entidad puede ser un objeto con existencia física - una cierta persona, un automóvil, una casa o un empleado -, o un objeto con existencia conceptual, como una compañía, un puesto de trabajo o un curso universitario. Cada entidad tiene propiedades específicas, llamadas **atributos** que la describen. Por ejemplo, una entidad auto puede describirse por su marca, modelo, año, dueño, color, número de placas, número de registro y número de motor. Una entidad particular tendrá un valor para cada uno de sus atributos; los valores de los atributos que describen a cada entidad constituyen una parte decisiva de los datos almacenados en la base de datos.

Tipos de atributos.

- Atributos simples o atómicos. Son atributos no divisibles.
- Atributos compuestos. Se pueden dividir en componentes más pequeños, que representan atributos más básicos con su propio significado independiente. Por ejemplo el atributo dirección se puede subdividir en Domicilio, Ciudad, País, y C. P. Los atributos compuestos pueden forman una jerarquía; por ejemplo, domicilio aún se podría subdividir en tres atributos simples, Calle, Número exterior y Número interior. El valor de un atributo compuesto es la concatenación de los valores de los atributos simples que lo constituyen. Si no hay necesidad de referirse a los componentes individuales de una dirección, la dirección completa se designará como atributo simple.
- Los atributos mono valuados tienen un solo valor para una entidad en particular. Por ejemplo, Edad es un atributo simple mono valuado de persona.
- Los atributos multivalor son aquellos que tienen múltiples valores. Por ejemplo, el grado académico de una persona puede tener un sólo valor, dos o más valores. Otro ejemplo sería el atributo esposa para un hombre, éste podría tener sólo una esposa o varias si sus usos y costumbres se lo permiten.
- Los atributos almacenados son aquellos que se tienen que se localizan en la memoria física de una computadora u otro mecanismo físico como un archivero.
- Los atributos derivados son aquellos que se pueden deducir directamente de un atributo almacenado. Por ejemplo, el atributo derivado edad puede deducirse en forma directa del atributo almacenado fecha de nacimiento.
- El valor nulo del atributo se emplea cuando no se aplica el valor al atributo. Por ejemplo, en el atributo grado académico, si la persona no tiene estudios se colocaría el valor nulo. Otros casos serían si uno desconoce el valor del atributo. Por ejemplo, en el atributo teléfono, tal vez no se conozca el valor para una persona en específico.

Tipos de entidades.

Un tipo de entidad define un conjunto de entidades que poseen los mismos atributos. Cada tipo de entidad en la base de datos se describe con un nombre y una lista de atributos. Un tipo de entidad describe el esquema o intensión para un conjunto de

entidades que comparten la misma estructura. Las entidades individuales de un tipo de entidad se agregan en una colección o conjunto de entidades, que se conocen también como extensión del tipo de entidad.

Atributos llave o atributos clave de un tipo de entidad.
Los tipos de entidades casi siempre tienen un atributo cuyo valor es distinto para cada entidad individual. Los atributos de esta naturaleza se denominan atributos llave o atributos clave, y sus valores pueden servir para identificar de manera única a cada entidad (u objeto en particular).

Algunos tipos de entidad tienen más de un atributo clave. Por ejemplo, tanto el atributo registro federal del automóvil y el atributo número de placa en el tipo de entidad coche son atributos llave por derecho propio.

Dominio de los atributos.
Cada uno de los atributos simples de un tipo de entidad está asociado a un conjunto de valores y su significado, que especifica los valores que es posible asignar a ese atributo para cada entidad individual. Por ejemplo, el intervalo de edad permitido para un empleado es de 16 a 70 años, por lo que el conjunto de valores permitidos para un empleado se localizará entre 16 y 70.
En términos matemáticos, un atributo A de un tipo de entidad E cuyo conjunto de valores es V se puede definir como una función de E al conjunto potencia de V:

$$A: E \rightarrow P(V)$$

La definición anterior abarca los atributos monovaluados y multivalor, además de los nulos. Un valor nulo se representa con el conjunto vacío. En el caso de atributos monovaluados, A (*e*) sólo puede ser un conjunto unitario para cada entidad *e* de E, pero no existe esta restricción para los atributos multivalor. En caso de un atributo compuesto A, el conjunto de valores V es el producto cartesiano de $P(V_1)$, $P(V_2)$, …, $P(V_n)$ son los conjuntos de valores de los atributos componentes simples que constituyen A.

$$V= P(V_1) \times P(V_2) \times P(V_3) \times P(V_4) \times \ldots \times P(V_n)$$

Cabe señalar que los atributos compuestos y multivalor pueden estar anidados de cualquier manera. A este tipo de atributo se les denomina ***atributos complejos.*** Podemos representar una anidación arbitraria agrupando componentes compuestos entre ( ) y separando los componentes con comas, y encerrando los atributos multivalor en { }. Por ejemplo, si una persona tiene tener más de una residencia y cada residencia puede tener varios teléfonos, se podría especificar un atributo DirecciónTeléfono para un tipo de entidad persona como sigue:

{DirecciónTeléfono ({teléfono(codárea, númteléfono)},
Dirección (domicilio (calle, númext, núminterior), ciudad, país, cp))}

3.2.2 Tipos de relación.
Un tipo de relación o vinculo "R" entre tipos de entidades E_1, E_2, E_3,…, E_n define un conjunto de asociaciones entre entidades de estos tipos de entidad. En términos

matemáticos, R es un conjunto de instancias de los tipos de relación r_i, donde cada r_i asocia n entidades ($e_1, e_2, \ldots, e_n$) y cada entidad e_j de r_i es miembro del tipo de entidades E_j, $1<=j<=n$. Por tanto, un tipo de relación es una relación matemática sobre $E_1, E_2, E_3, \ldots$, En. También puede definirse como un subconjunto de producto cartesiano $E_1 \times E_2 \times E_3 \times \ldots, \times E_n$. Se dice que cada uno de los tipos de entidades $E_1, E_2, E_3, \ldots, E_n$ participa en el tipo de relaciones R y, de manera similar, cada una de las entidades individuales $e_1, e_2, e_3, \ldots, e_n$ participa en la instancia del tipo de relación $r_i = (e_1, e_2, e_3, \ldots, e_n)$.
En términos informales, cada instancia del tipo de relación r_i de R es una asociación de entidades, donde la asociación incluye una y sólo una entidad de cada tipo de entidades participante. Cada una de estas instancias de relación r_i representa el hecho de que las entidades que participan en r_i están relacionadas entre sí de alguna manera en la situación correspondiente del minimundo. Posteriormente se podrá observar que, en una oración dentro de los requerimientos, un tipo de entidad es el sustantivo u objeto (como se sabe, el objeto lo definen sus atributos) y un tipo de relación es el verbo que une a los tipos de entidad.

Grado de un tipo de relación.
El grado de un tipo de relación es el número de tipos de entidades que participan en él. Los tipos de relación de grado dos se llaman binarios, y los de grado tres se llaman ternarios.
Como ejemplo del grado binario, el tipo de relación a emplearse es "pertenece a" en el cual permite relacionar al tipo de entidad empleado con el tipo de entidad departamento.

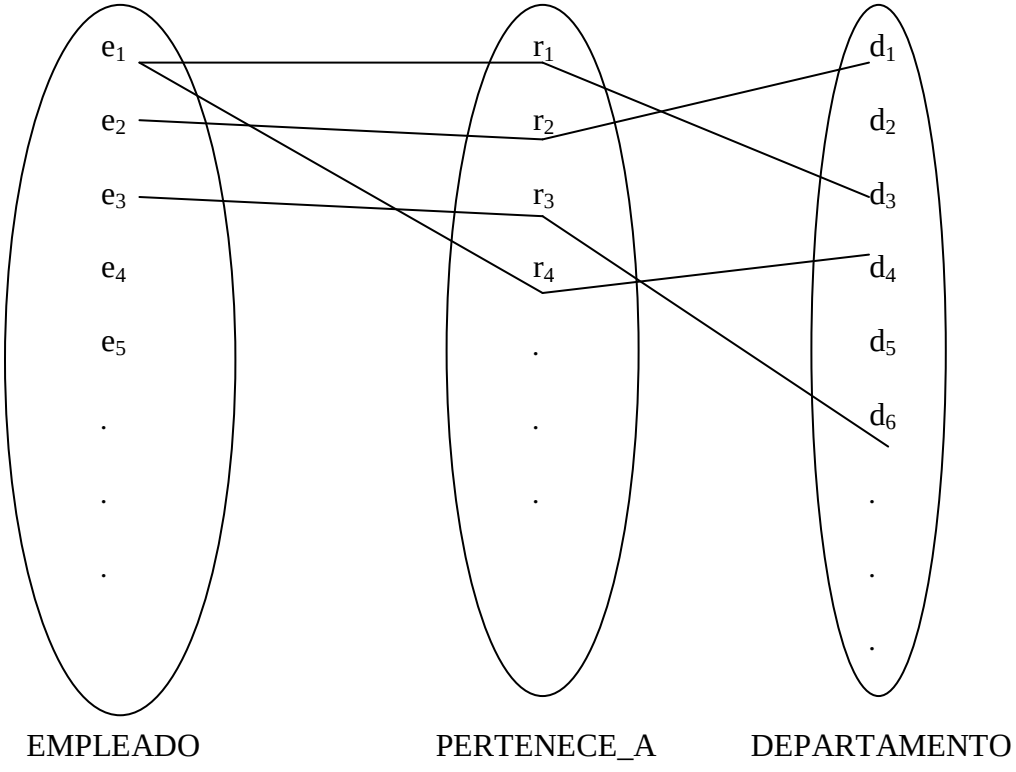


Fig. 3.2 Un ejemplo del grado de relación binario.

Como ejemplo de grado ternario, el tipo de relación usada es suministrar, donde cada instancia del tipo de relación r_i asocia a tres entidades - un proveedor v, componente c, proyecto p – siempre que v suministre el componente c al proyecto p.

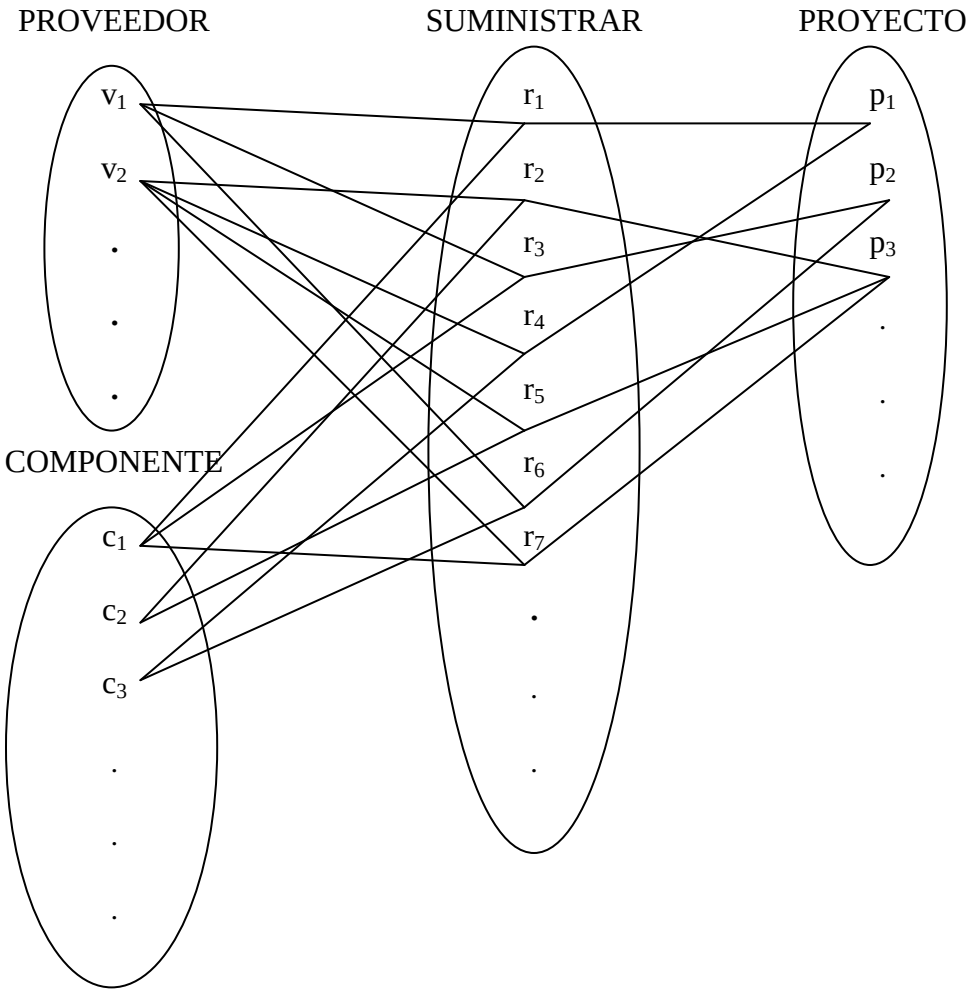


FIG 3.3 Ejemplo de un tipo de relación ternaria.

La oración dentro de los requerimientos puede escribirse así:

"El proveedor **x** suministra el componentes **y** al proyecto **z**".

Nombres de papeles y tipos de relación recursiva.

Todo tipo de entidad que participe en un tipo de relación desempeña un papel específico en el tipo de relación. El nombre del papel indica el papel que una entidad participante desempeña en la instancia del tipo de relación. Por ejemplo, en el tipo de relación PERTENECE_A, EMPLEADO desempeña el papel de empleado o trabajador y DEPARTAMENTO tiene el papel de departamento o patrón.

No son necesarios los nombres de papeles en los tipos de relación en los que todos los tipos de entidades participantes son distintos, ya que cada nombre de tipo de entidad se

puede usar como el nombre de papel. Sin embargo, en algunos casos el mismo tipo de entidad participa más de una vez en un tipo de relación con diferentes papeles. En tales casos el nombre del papel resulta indispensable para distinguir el significado de cada participación. Estos tipos de relación se conocen como recursivos, y la figura 3.4 muestra un ejemplo. El ejemplo trata de un rancho que tiene una base de datos en la cual un tipo de entidad pertenece a las vacas y toros existentes en el rancho. También se desea indicar los animales que ha procreado cada vaca. La vaca y sus retoños pertenecen a un tipo de entidad llamada VACAS_TOROS. Así, cada animal puede participar como mamá o hijo(a) de una determinada vaca.

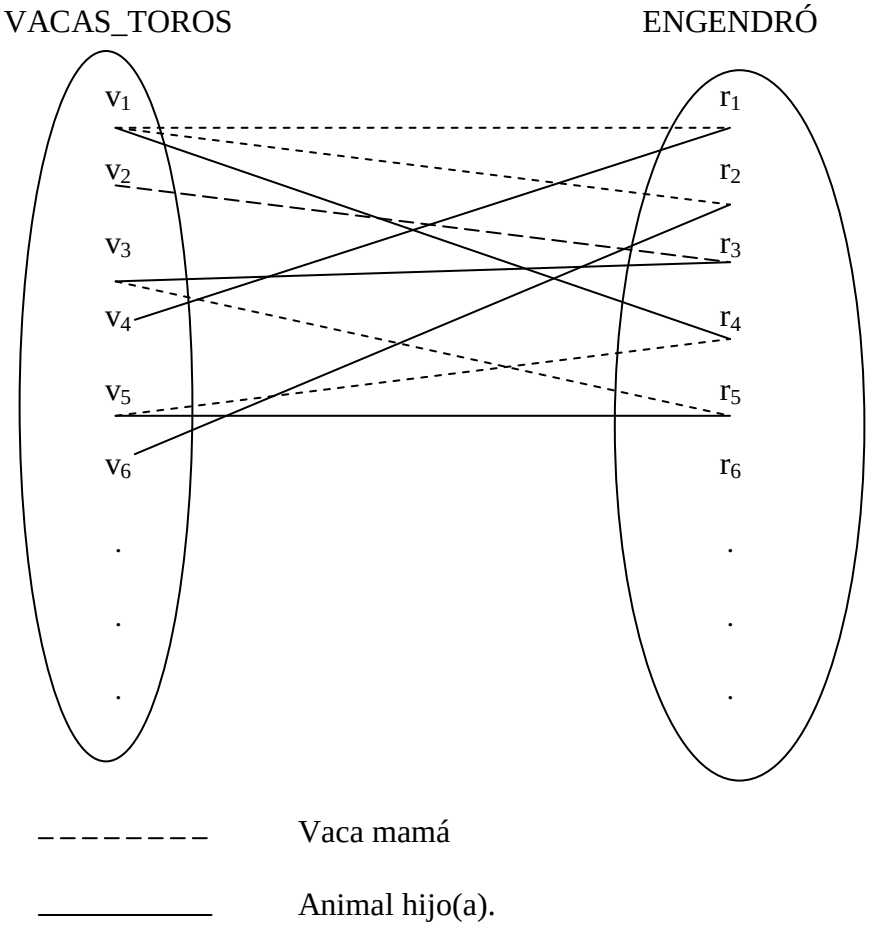
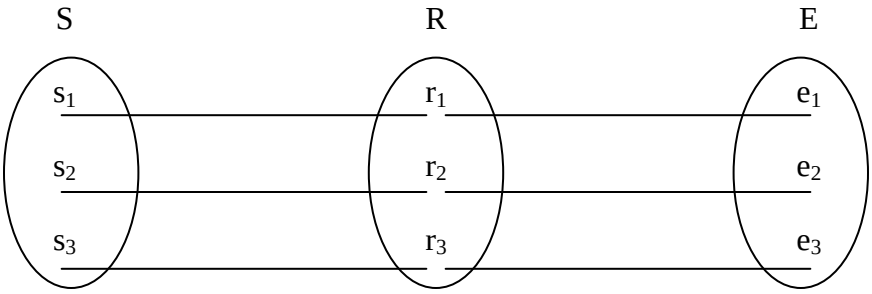


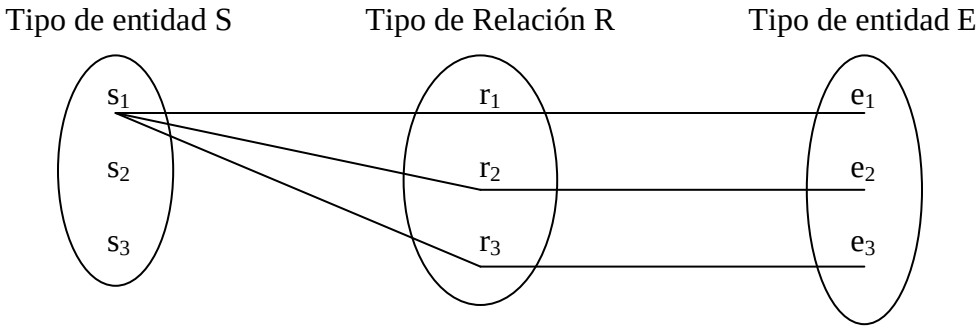
Fig. 3.4 Tipo de relación recursiva.

3.2.3 Restricciones sobre los tipos de relación.
Estas restricciones se determinan a partir de la situación del minimundo que los tipos de restricción representan. Se pueden distinguir dos tipos principales de restricción de los tipos de relación: razón de cardinalidad y participación.

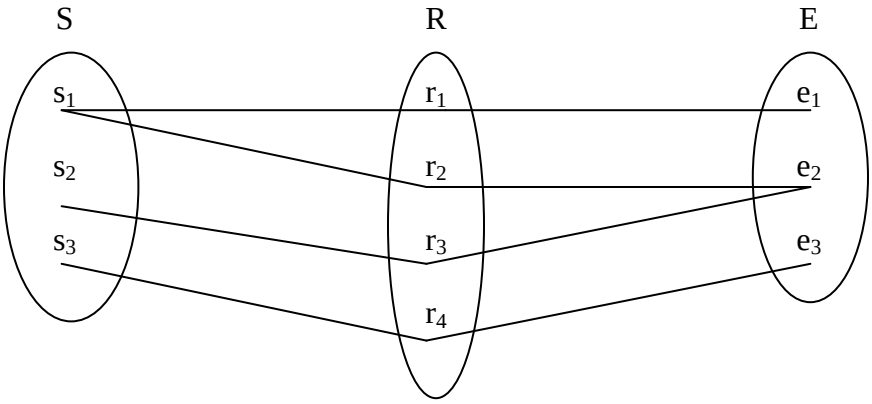
- Razón de cardinalidad. Especifica el número veces en los que una entidad puede participar en un tipo de relación. Las razones de cardinalidad más comunes son:



Razón de cardinalidad 1:1



Razón de cardinalidad 1: N



Razón de cardinalidad N:M

Fig. 3.5

3.2.4 Restricción de participación.

El tipo de participación puede ser total o parcial.

- Participación total. Es cuando toda entidad de un tipo de entidad tiene un tipo de relación con otro tipo de entidad.
- Participación parcial. Es cuando existen algunas (mas no todas) entidades en un tipo de entidad tienen un tipo de relación con otro tipo de entidad.

Ejemplo:

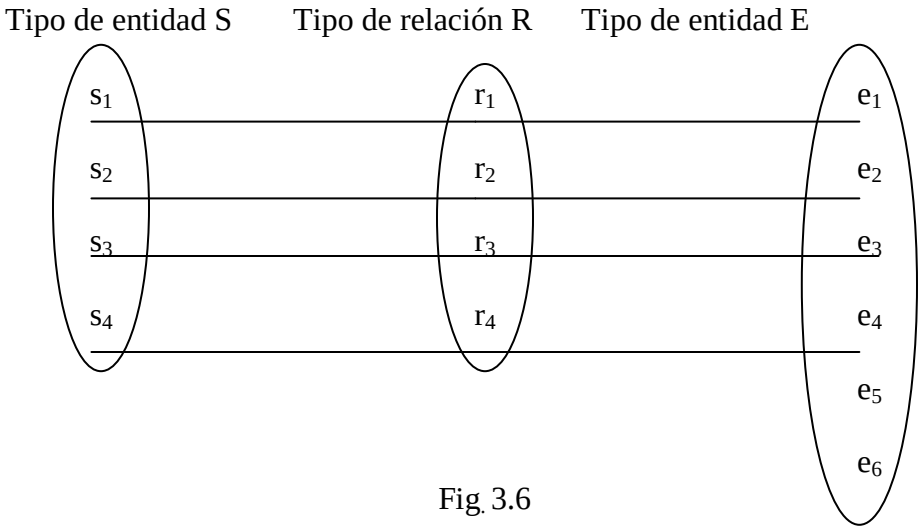


Fig. 3.6

En este ejemplo, la restricción de participación es total del tipo de entidad S al tipo de entidad E, ya que toda entidad s_j de S tienen una instancia en el tipo de relación R con una entidad e_i de E. El tipo de participación es parcial de E a S ya que las entidades e_5 y e_6 del tipo de entidad E no tienen una instancia en el tipo de relación R con el tipo de entidad S.

3.2.5 Los atributos en los tipos de relación.

Los tipos de relación también pueden tener atributos similares a los tipos de entidad. Cuando un atributo puede existir tanto en un tipo de entidad u otro tipo de entidad con un tipo de relación que existente entre ellos, el atributo es el tipo de relación. Cuando la razón de cardinalidad es 1:1 o 1: N el atributo se puede trasladar a uno de los tipos de entidad participantes en el tipo de relación (al ser traducido al modelo lógico). En el caso de que la razón de cardinalidad es de N: M, el atributo se especificará como atributo del tipo de relación (y será tratado en forma particular al ser traducido al modelo lógico).
También se puede comentar que el atributo pertenece al tipo de vínculo cuando el atributo explica la acción o vínculo entre tipos de entidades.

3.2.6 Tipo de entidad débil.

Es posible que algunos tipos de entidades no tengan atributos llave propios; éstos se denominan tipos de entidades débiles. Las entidades que pertenecen a un tipo de entidades débil se identifican plenamente por su relación con entidades específicas de otro tipo de entidad, en combinación con algunos de los valores de sus atributos. Decimos que este otro tipo de entidad es propietario o identificador, y llamamos al tipo de relación "identificador" ya que relaciona a un tipo de entidad débil con su tipo de entidad propietario. Los tipos de entidades débiles siempre tienen una restricción de participación total (dependencia de existencia) con respecto a su tipo de relación identificador, porque una entidad débil no se puede identificar sin una entidad propietaria. Por lo regular, los tipos de entidades débiles tienen una llave parcial, que es el conjunto de atributos que

pueden identificar de manera única las entidades débiles relacionadas con la misma entidad propietaria.
Hay ocasiones en las que los tipos de entidades débiles se representan en forma de atributos multivaluados compuestos. El diseñador de la base de datos decide cuál representación se utilizará. Una estrategia es elegir la representación de tipo de entidades débil si tiene muchos atributos y participa de manera independiente en otros tipos de relación, aparte de su tipo de relación identificador. En general, es posible definir cualquier cantidad de niveles de tipos de entidades débiles; un tipo de entidades propietario puede ser él mismo un tipo de entidad débil. Además, los tipos de entidades débiles pueden tener más de un tipo de entidades identificador y un tipo de relación identificador de grado superior a dos.

3.3 Notación del modelo ER.

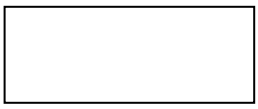
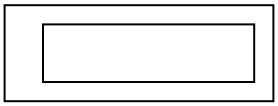
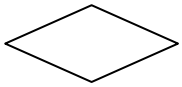
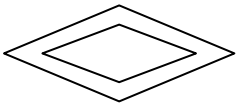
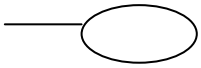
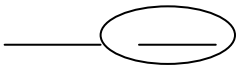
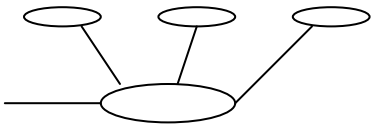
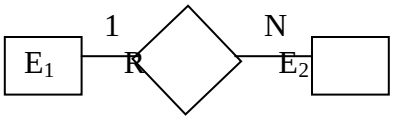
Símbolo	Significado
	Tipo de entidad
	Tipo de entidad débil
	Tipo de relación
	Tipo de relación identificador
	Atributo
	Atributo clave o llave
	Atributo compuesto
	Atributo multivaluado
	Atributo derivado
	Participación total
	Participación parcial
	Razón de cardinalidad.

Fig. 3.7

Un ejemplo de requerimiento y desarrollo del modelo conceptual.
En esta sección describiremos una base de datos de ejemplo, llamada COMPAÑIA, que servirá para ilustrar los conceptos del modelo ER y su uso en el diseño de esquemas. Aquí vamos a mencionar los requerimientos de información de esta base de datos, y en seguida crearemos su esquema conceptual paso a paso.

La base de datos COMPAÑIA se ocupa de los empleados, departamentos y proyectos de una empresa. Vamos a suponer que, una vez concluida la fase de recolección y análisis de requerimientos, los diseñadores de bases de datos redactaron la siguiente descripción del 'minimundo' (la parte de la compañía que se representará en la base de datos):

1. La compañía está organizada en **departamentos**. Cada departamento tiene un nombre único, un número único y un empleado que lo dirige, y nos interesa la fecha en que dicho empleado comenzó a dirigir el departamento. Un departamento puede estar distribuido en varios lugares.
2. Cada departamento controla un cierto número de **proyectos**, cada uno de los cuales tiene un nombre y un número únicos, y se efectuará en un solo lugar.
3. Se almacenará el nombre, número de matrícula, dirección, salario, sexo y fecha de nacimiento de cada **empleado**. Todo empleado está asignado a un departamento, pero puede trabajar en varios proyectos, que no necesariamente estarán controlados por el mismo departamento. Nos interesa el número de horas por semana que un empleado trabaja en cada proyecto, y también quién es el supervisor de cada empleado.
4. Se quiere mantener al tanto de los **dependientes** de cada empleado con el fin de administrar los términos de sus seguros. Se almacenará el nombre, sexo y fecha de nacimiento de cada dependiente, y su parentesco con el empleado.

Tipos de entidades y atributos en la base de datos COMPAÑIA.
Se pueden identificar cuatro tipos de entidades, uno para cada uno de los cuatro elementos (el sujeto que más destacan en cada uno de los cuatro puntos indicado en negrita.):

1. Un tipo de entidad DEPARTAMENTO con atributos Nombre, Número, Lugares. Lugares es un atributo multivaluado. Pueden ser el Nombre o el Número como atributo llave. En este caso se toma como atributo llave el Número y queda como atributo llave candidato a Nombre.
2. Un tipo de entidad PROYECTO con los atributos Nombre, Número y Lugar. Tanto Nombre como Número pueden ser atributo llave. Se escoge el Número como atributo llave, quedando el Nombre como atributo llave candidato.
3. Un tipo de entidad EMPLEADO con los atributos Nombre, Matricula, Sexo, Dirección, Salario, FechaNac y Departamento. Tanto Nombre como Dirección pueden ser atributos compuestos, aunque esto no se especificó en los requerimientos. Se debe de remitir a los usuarios para ver si alguno de ellos va a hacer referencia a los componentes individuales de Nombre – Paterno, Materno, NombPil- o de dirección.

1. Un tipo de entidad débil DEPENDIENTES con el tipo de entidad EMPLEADO, sus atributos son: NombreDependiente, Sexo, FechaNac y Parentesco (con el empleado). Ninguno de los atributos figuran como llave primaria, por lo que se escoge a un atributo como llave parcial, en este caso es el atributo NombreDependiente.

El diseño preliminar de los tipos de entidades para la base de datos se indica enseguida:

DEPARTAMENTO
Nombre, Número, {Lugares}

PROYECTO
Nombre, Número, Lugar

EMPLEADO
Nombre,(NombPila, Paterno, Materno), Matricula, sexo, Dirección, Salario
FechaNac, Departamento.

Siendo los atributos multivaluados indicados entre claves { }; los componentes de un atributo compuesto aparecen entre paréntesis ( ).

Tipos de relaciones y sus restricciones en la base de datos COMPAÑIA.

1. DIRIGE, un tipo de relación 1:1 entre EMPLEADO y DEPARTAMENTO. La participación de EMPLEADO es parcial, pero la de DEPARTAMENTO es total ya que cada departamento siempre debe tener un gerente.
2. PERTENECE_A, un tipo de relación 1: N entre DEPARTAMENTO y EMPLEADO. Ambas participaciones son totales.
3. CONTROLA, un tipo de relación 1: N entre DEPARTAMENTO y PROYECTO. La participación de PROYECTO es total; luego de consultar con los usuarios, se determina que la participación de DEPARTAMENTO es parcial.
4. SUPERVISION, un tipo de relación 1: N entre EMPLEADO (en el papel de supervisor) y EMPLEADO (en el papel de supervisado). Los usuarios nos dicen que no todo empleado es un supervisor y no todo empleado tiene un supervisor, de modo que ambas participaciones son parciales.
5. TRABAJA_EN que, después de que los usuarios indican que varios empleados pueden trabajar en un proyecto y que un empleado puede trabajar en varios proyectos, resulta ser un tipo de relación M: N. El atributo Horas pertenece al tipo de relación TRABAJA_EN ya que no pertenece al proyecto como tal ni al empleado, pertenece a la acción. Se determina que ambas participaciones son totales.
6. DEPENDIENTES_DE, un tipo de relación 1: N entre EMPLEADO y DEPENDIENTE, que también es el tipo de relación identificador del tipo de entidad débil DEPENDIENTE. La participación de EMPLEADO es parcial, en tanto que la de DEPENDIENTE es total.

Es importante tener el mínimo de redundancia posible cuando diseñemos en esquema conceptual de una base de datos.

El diagrama ER del esquema COMPAÑIA.

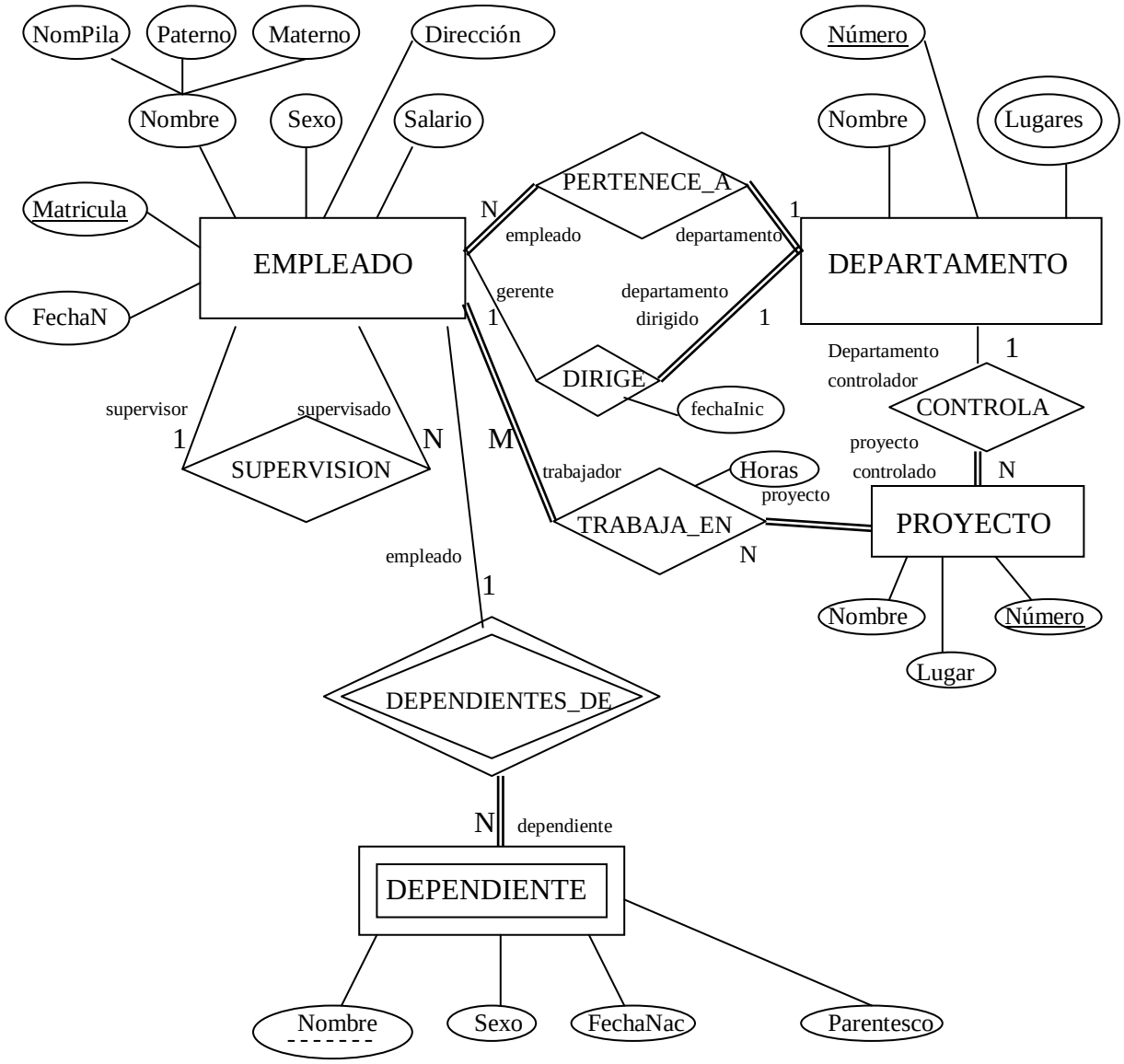


Fig. 3.8

Nombres apropiados para los elementos de esquema.

No siempre es trivial la elección de nombres para los tipos de entidades, los atributos, los tipos de relación y sobre todo los papeles que desempeñan. Debemos elegir nombres que comuniquen, hasta donde sea posible, los significados conferidos a los distintos elementos del esquema. Optamos en usar nombres en singular para los tipos de entidades, y no en plural, porque el nombre del tipo de entidades se aplica a cada una de las entidades individuales que pertenecen a ese tipo. En nuestros diagramas ER aplicamos la convención de que los nombres de los tipos de entidades y tipos de relación van en

mayúscula, los nombres de atributos comienzan con mayúscula, y los nombres de papeles van con minúsculas.
Como práctica general, dada una descripción narrativa de los requerimientos de la base de datos, los sustantivos que aparezcan en la narración tenderán a originar nombres de tipos de entidades, y los verbos tenderán a indicar nombres de tipos de relación. Los nombres de los atributos generalmente surgen de los sustantivos adicionales que describen a los sustantivos correspondientes a los tipos de entidades. Otra consideración en lo tocante a los nombres es que los de los tipos de relación deben de elegirse de modo que el diagrama ER del esquema se pueda leer de izquierda a derecha y de arriba hacia abajo.

Tipos de relación con grado mayor que dos.
Ya se definió el grado de un tipo de relación como el número de tipos de entidades participantes, y dijimos que un tipo de relación de grado dos era binario y uno de grado tres era ternario. La notación de diagrama ER para un tipo de relación ternario se ilustra enseguida donde se muestra el esquema del tipo de relación SUMINISTRA. En general, un tipo de relación R de grado n tendrá n aristas conectadas en un diagrama ER, y cada una conectará a R con un tipo de entidad participante.

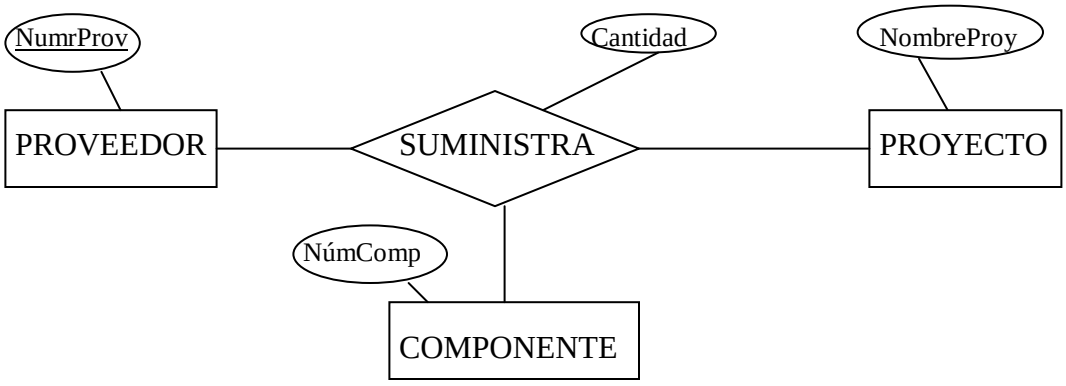


Fig. 3.9

En la siguiente figura se muestra un diagrama ER para los tres tipos de relación binarios PUEDE_SUMINISTRAR, UTILIZA y SUMINISTRA.

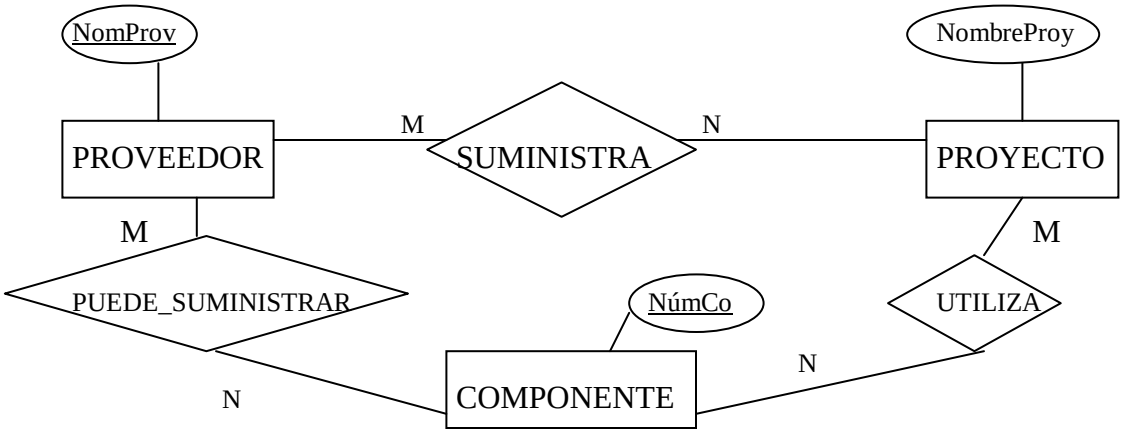


Fig. 3.10

En general, un tipo de relación ternario representa más información que tres tipos de relación binaria. A menudo resulta difícil decidir si un cierto tipo de relación es de grado n o si se debe descomponer en varios tipos de relación de menor grado. El diseñador debe basar su decisión en la semántica o significado de la situación específica que se va a representar. La solución más común es incluir el vínculo ternario junto con uno o más de los vínculos binarios, según se necesite.
Algunas de las herramientas que se emplean en el diseño de bases de datos tienen su fundamento en variaciones del modelo ER que sólo permiten vínculos binarios. En este caso, los tipos de relación ternarios como SUMINISTRAR se deben representar como tipos de entidades débiles, sin clave parcial y con tres vínculos identificadores. Los tres tipos de entidades participantes PROVEEDOR, COMPONENTE y PROYECTO, en conjunto, son los tipos de entidades propietarios; por tanto, una entidad del tipo de entidades débil SUMINISTRAR se identifica mediante la combinación de sus tres entidades propietarias provenientes de PROVEEDOR, COMPONENTE y PROYECTO.

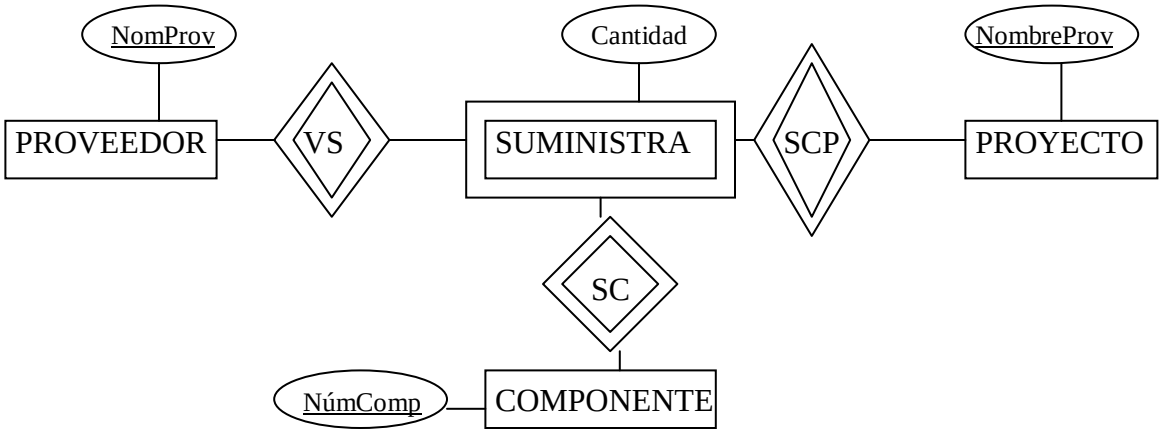


Fig. 3.11

3.4 El modelo ENTIDAD – RELACIÓN EXTENDIDO (ERE).

3.4.1. Introducción.

El modelo entidad – relación es suficiente para representar la mayoría de los esquemas de bases de datos tradicionales, las cuales incluyen las aplicaciones en los negocios y la industria. Sin embargo, desde 1970, existen nuevas aplicaciones de bases de datos (como CAD/CAM, bases de datos de imágenes y gráficas, cartografía y multimedia, incluyendo las bases de datos generadas en Inteligencia Artificial) en las cuales el diseñador requiere introducir semántica adicional. Se han propuesto varios modelos en la literatura. El que se revisará en este curso será el modelo ERE.

3.4.2. Conceptos del modelo ERE.

El modelo ERE incluye todos los conceptos del modelo ER. En adición, incluye los conceptos de subclase y superclase además de los conceptos de especialización y generalización. Otro concepto incluido en el modelo ERE es la categorización. Asociado

con estos conceptos se encuentra el mecanismo de herencia en el atributo. Desafortunadamente, no existe una terminología estándar para estos conceptos. Por lo que se usará la terminología más empleada hasta estos momentos.

3.4.3 Subclases, superclases y especialización.

Subclases y Superclases. En varios casos, un tipo de entidad tiene varios subgrupos de sus entidades y necesitan ser representados explícitamente porque así lo solicitan los requerimientos de la base de datos. Por ejemplo, los tipos de entidad que son miembros del tipo de entidad EMPLEADO pueden ser agrupados en SECRETARIA, INGENIERO, TECNICO, ADMINISTRADOR, etc., etc., el conjunto de tipos entidad es un subconjunto de los tipos de entidad que pertenecen a EMPLEADO. Cada entidad pertenece a uno de los subgrupos y además es un empleado. Se conocerá a éste subgrupo como SUBCLASE del tipo de entidad EMPLEADO, y EMPLEADO se conoce como SUPERCLASE de estas subclases.
La relación CLASE/SUBCLASE es a veces conocida como una relación ES UN porque nosotros decimos "un INGENIERO ES UN EMPLEADO" o "una SECRETARIA ES UNA EMPLEADA", etc.
Una entidad no puede existir en una base de datos sólo porque es miembro de una subclase, debe ser también miembro de la superclase. Sin embargo, no es necesario que toda entidad que pertenezca a una superclase sea miembro de alguna subclase.

La herencia en los atributos. Ya que una entidad de una subclase representa la misma entidad de la superclase, poseerá atributos específicos que pertenecen a la subclase y atributos que son parte de la superclase. Toda entidad que es miembro de una subclase hereda todos los atributos de la superclase. Observe que una subclase, junto con todos los atributos que hereda de la superclase, es ya un tipo de entidad por propio derecho.

Especialización. La especialización es el proceso de definir un conjunto de subclases de un tipo de entidad llamada la superclase. El conjunto de subclases que forma una especialización se define sobre la base de algunas características que lo distinguen de otras especializaciones dentro de la misma superclase. Por ejemplo, el conjunto de subclases {SECRETARIA, INGENIERO, TECNICO} es una especialización de la superclase EMPLEADO, otra especialización puede ser {EMPLEADO_ASALARIADO, EMPLEADO_HORAS}; esta especialización distingue a los empleados basado en el método de compensación.
Generalización. La especialización discutida en la sección anterior nos permite:

- Definir un conjunto de subclases de un tipo de entidad.
- Asociar atributos específicos a cada subclase.
- Establecer tipos de relación adicionales entre subclases y entre cada subclase y otros tipos de entidad u otras subclases.

Se puede pensar en un proceso de reversa en el cual se suprimen las diferencias sobre varios tipos de entidad y se identifican sus rasgos comunes, y se generalizan a una superclase del cual un tipo de entidad es una subclase en especial. Por ejemplo, considere los tipos de entidad CAMIONETA y AUTO, ambos pueden ser generalizados a un tipo de entidad VEHICULO. Se entiende como generalización al proceso de definir un tipo de entidad generalizado a partir de tipos de entidades que se pueden agrupar.

La generalización puede ser vista como la inversa de la especialización.

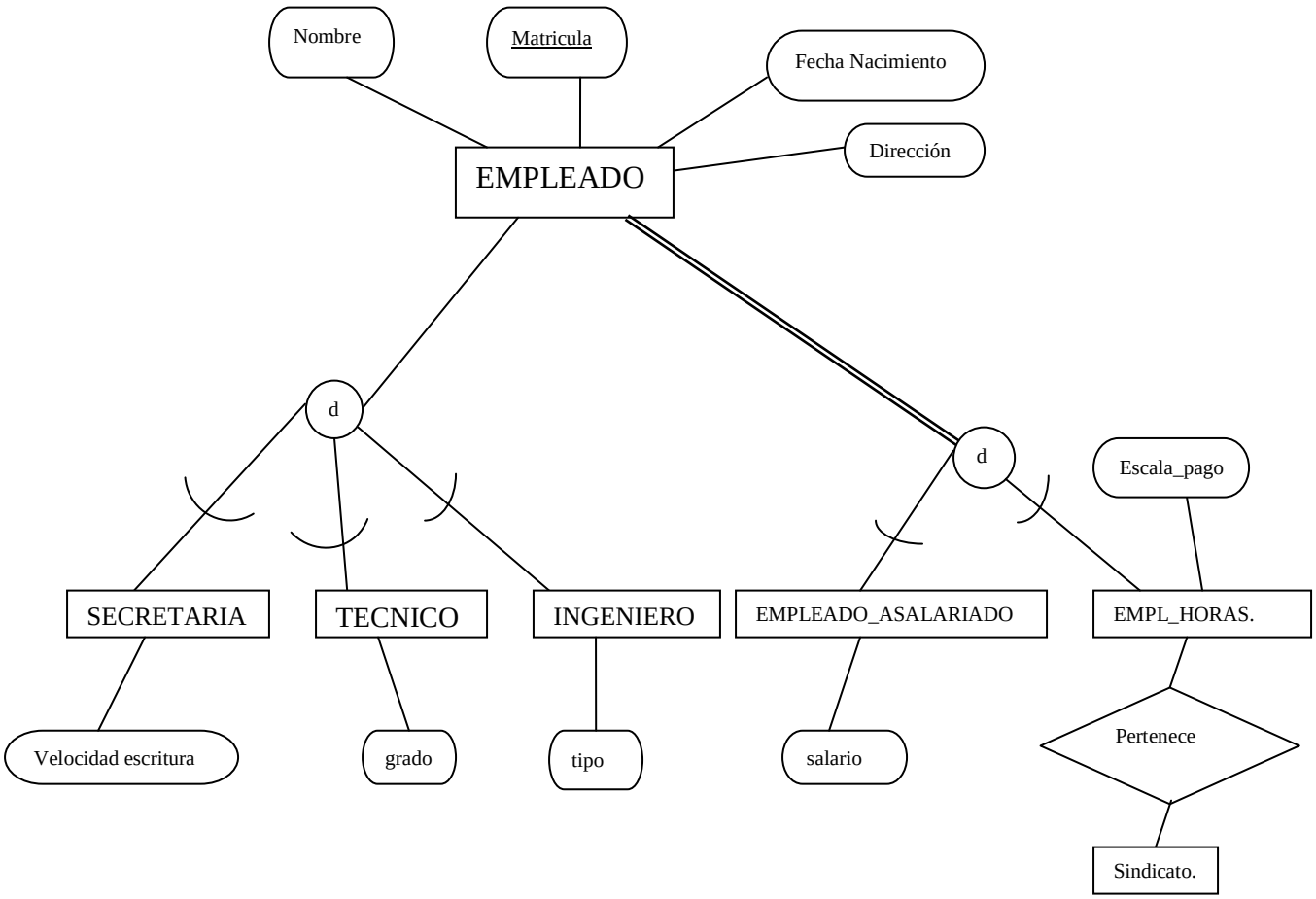


Fig. 3.12. Modelo ERE representando especialización.

La “d” en el círculo de la figura 3.12 indica “disjunto”, esto indica que una entidad solo puede pertenecer a una de las subclases ahí indicada. Si no existe la restricción de disyunción, entonces existe la restricción de sobre posición; esto es, la misma entidad puede pertenecer a más de una subclase de la especialización. En este caso se colocará una “o” en el círculo (overlap).

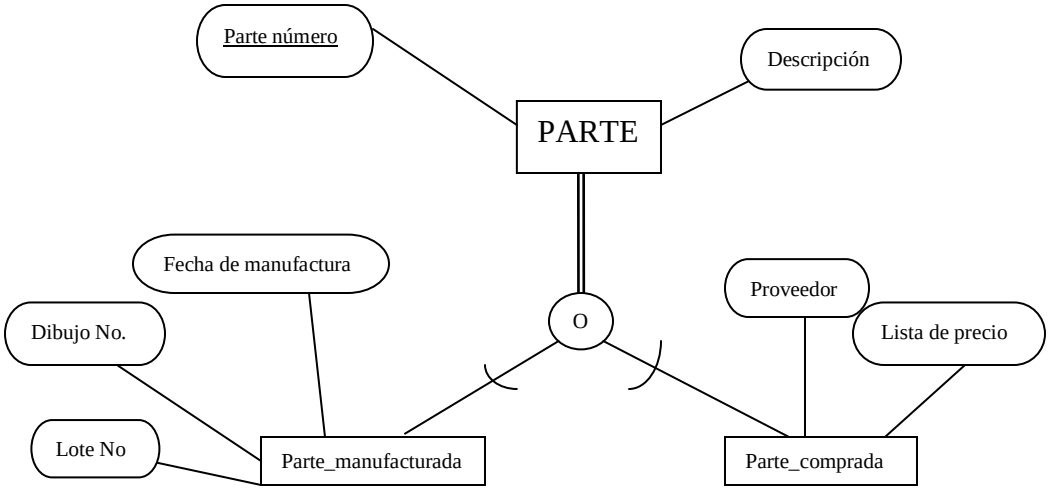


Fig.3.13. Especialización con subclases sobrepuestas

En una especialización, la participación puede ser parcial o total. La participación total indica que para toda entidad en la superclase debe ser miembro de una subclase dentro de la especialización. Por ejemplo, en la figura 3.12, todo empleado tiene que ser o asalariado o contratado por horas y se mostrará como doble línea en el modelo ERE. Una línea simple indicará especialización parcial. De esta forma, se tienen cuatro tipos de especialización:

- Disjunto, total
- Disjunto, parcial
- Sobrepuesto, parcial
- Sobrepuesto, total.

3.4.4. Especialización en malla (lattice) y herencia múltiple.

Una subclase puede tener ella misma subclases, formando una jerarquía o una malla de especializaciones. Por ejemplo, en la figura 3.14 INGENIERO es subclase de EMPLEADO y además es superclase de GERENTE_DE_INGENIEROS. Una jerarquía de especialización tiene la restricción de que toda subclase participa (como subclase) en una relación clase/subclase; en contraste, para una especialización tipo lattice una subclase puede ser subclase en más de una relación clase/subclase. Por lo que la figura 3.14 forma, dentro de su diagrama, un lattice.

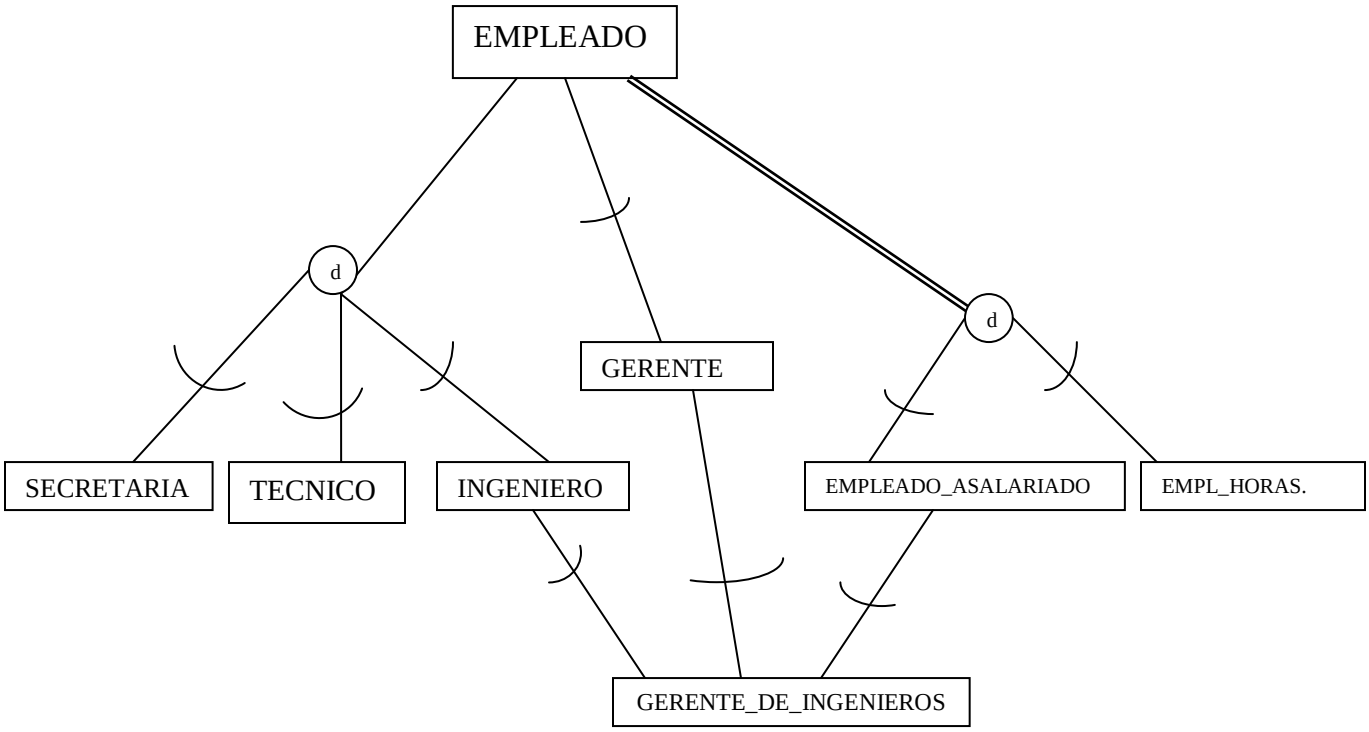


Fig. 3.14 Malla tipo especialización.

En la figura 3.15 se muestra otro lattice de especialización con más de un nivel. Esto puede ser parte de un esquema conceptual de una base de datos de una universidad. En tal especialización tipo lattice o malla, una subclase hereda los atributos de todos sus predecesores.

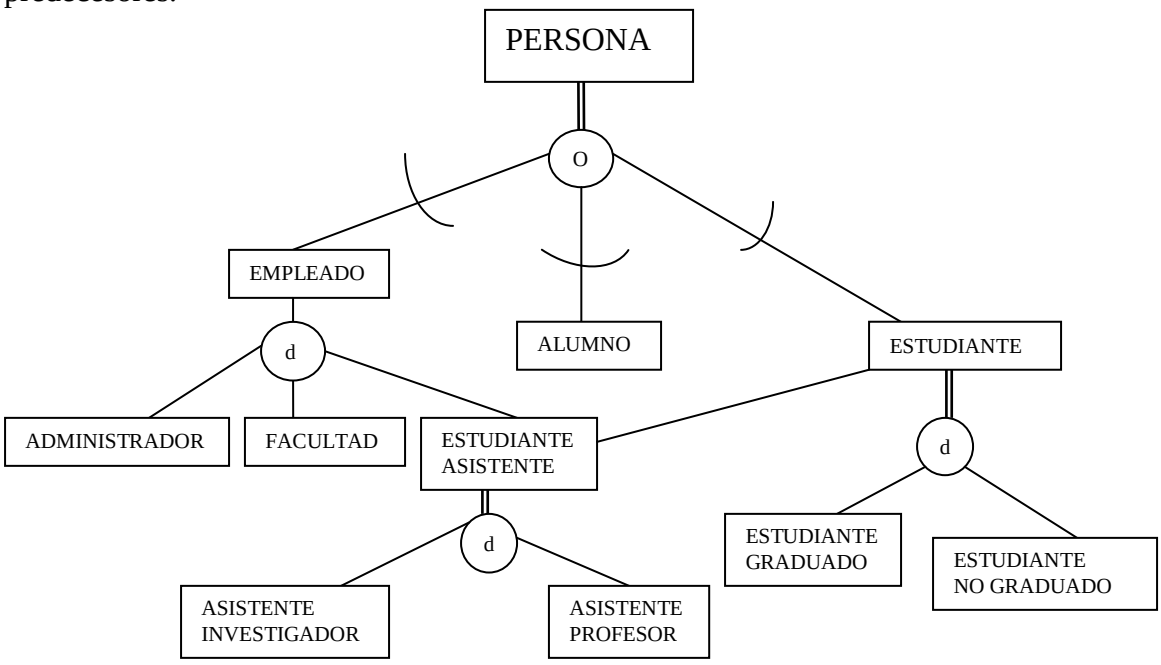


Fig. 3.15. Un lattice de especialización para una base de datos Universidad.

Una subclase con más de una especialización es conocida como subclase compartida. Por ejemplo, todo GERENTE_DE_INGENIEROS debe ser un INGENIERO pero también tiene que ser un EMPLEADO_ASALARIADO y un GERENTE, entonces el GERENTE_DE_INGENIEROS tiene que compartir todos los atributos de sus superclases. Esto introduce el concepto conocido como herencia múltiple, ya que la subclase GERENTE_DE_INGENIEROS directamente hereda atributos y relaciones de las múltiples superclases.

3.4.5. Categorías.

Una categoría tiene dos o más superclases que pueden representar distintos tipos de entidades, mientras que otras relaciones superclase/clase siempre tiene una simple superclase. Se puede comparar una categoría, como el tipo de entidad DUEÑO de la fig. 3.16 con GERENTE_DE_INGENIEROS de la figura 3.14. El último es una subclase de cada una de las tres superclases, de esta forma una entidad que es miembro de GERENTE_DE_INGENIEROS debe existir en las tres superclases (INGENIERO, GERENTE, EMPLEADO_ASALARIADO) por lo que un GERENTE_DE_INGENIEROS es un subconjunto de la intersección de las tres subclases. De la otra forma, una categoría es un subconjunto de la unión de sus superclases. De esta forma, una entidad que es miembro de DUEÑO debe existir en al menos en una de las superclases pero no tiene que ser miembro de todas ellas.

Los atributos que heredan trabajan más selectivamente en el caso de categorías. Por ejemplo, en la figura 3.16 cada entidad DUEÑO hereda los atributos de una COMPAÑIA, una PERSONA o un BANCO, dependiendo de la superclase a la cual pertenece. Esto se conoce como **herencia selectiva**. De la otra forma, una subclase tal como GERENTE_DE_INGENIEROS hereda todos los atributos de las superclases.

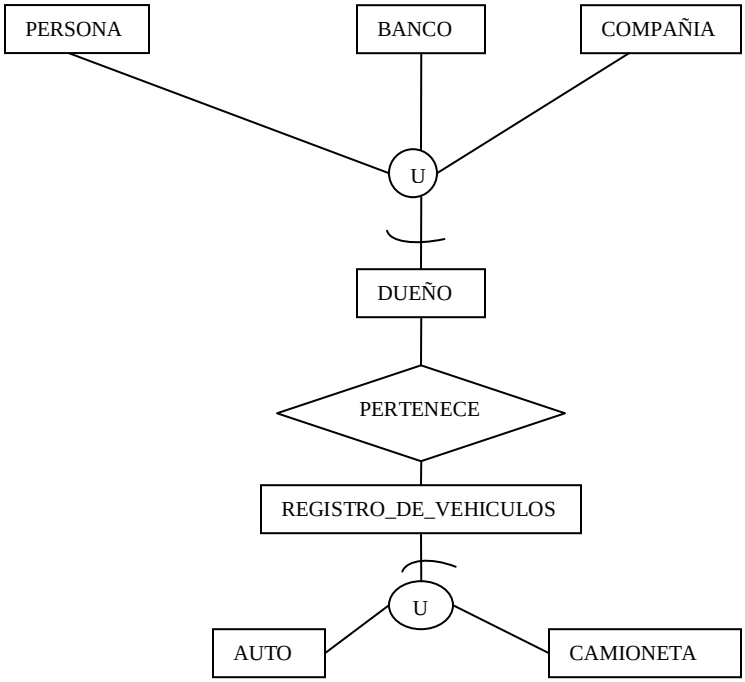


Fig. 3.16. Muestra de dos categorías. DUEÑO y REGISTRO_DE_VEHICULOS.

En este caso, la **U** dentro del círculo indica UNIÓN.

3.4.6 Ejemplo de un requerimiento:

CONFLICTOS BÉLICOS.

Una organización internacional pretende realizar un seguimiento de los conflictos bélicos que se producen en todo el mundo. Para ello creará una BD que responderá al siguiente análisis:

Se entiende por conflicto a cualquier lucha armada que afecte a uno o varios países y en el cual se produzcan muertos y/o heridos. Todo conflicto se identificará por un nombre que habitualmente hará referencia a la zona o causa que provoca el conflicto, aunque dado que este nombre puede cambiar con el paso del tiempo. Dentro de la BD cada conflicto se identificará mediante un código numérico sin significado alguno. Para cada conflicto se desea recoger los países a que afecta, así como el número de muertos y heridos contabilizados hasta el momento.

Los conflictos pueden ser de distintos tipos según la causa que lo ha originado clasificándose, a lo sumo, en cuatro grupos: territoriales, religiosos, económicos o raciales, en cada uno de estos grupos se recogerán diversos datos. En los conflictos territoriales se recogerán las regiones afectadas, en los religiosos las religiones afectadas, en los económicos las materias primas disputadas y en los raciales las etnias enfrentadas.

En los conflictos intervienen diversos grupos armados (al menos dos) y diversas organizaciones mediadoras (podría no haber ninguna). Los mismos grupos armados y organizaciones mediadoras podrán entrar y salir del conflicto, en ambos casos se recogerá tanto la fecha de incorporación como la fecha de salida. Temporalmente, tanto un grupo armado como una organización mediadora podrían no intervenir en conflicto alguno.

De cada grupo armado se recoge el código que se le asigna y un nombre. Cada grupo armado dispone de al menos una división y es liderado por al menos un líder político. Las divisiones de que dispone un grupo armado se numeran consecutivamente y se registra el número de barcos, tanques, aviones y hombres de que dispone, asimismo se recoge el número de bajas como suma de bajas producidas en todas sus divisiones.

Los traficantes de armas suministran diferentes tipos de armas a los grupos armados. De cada tipo de arma se recoge un nombre y un indicador de su capacidad destructiva. De cada traficante se recoge un nombre, los diferentes tipos de armas que puede suministrar y cantidad de armas de cada uno de los tipos de arma que podría suministrar. Se mantiene el número total de armas de cada uno de los diferentes tipos de armas suministrado por cada traficante a cada grupo armado

Los líderes políticos se identifican por su nombre y por el código de grupo armado que lideran. Además se recoge una descripción textual de los apoyos que éste posee.

Cada división la pueden dirigir conjuntamente un máximo de tres jefes militares, aunque cada jefe militar no dirige más de una división. A cada jefe militar se le identifica por un código, además se recoge el rango que éste posee, y dado que un jefe militar no actúa por iniciativa propia sino que siempre obedece las ordenes de un único líder político de entre aquellos que lideran al grupo armado al que el jefe pertenece, se registrará el líder político al que obedece.

De las organizaciones mediadoras se recogerá su código, su nombre, su tipo (gubernamental, no gubernamental o internacional), la organización de que depende (una como máximo), el número de personas que mantiene desplegadas en cada conflicto y el tipo de ayuda que presta en cada conflicto que será de uno y solo uno de los tres tipos siguientes, médica, diplomática o presencial.

Con diversos fines, los líderes políticos dialogan con las organizaciones; se desea recoger explícitamente esta información. Así para cada líder se recogerán aquellas organizaciones con que dialoga o viceversa

ANALISIS DEL REQUERIMIENTO.

PARTE I.

> *Se entiende por conflicto cualquier lucha armada que afecte a uno o varios países y en el cual se produzcan muertos y/o heridos. Todo conflicto se identificará por un nombre que habitualmente hará referencia a la zona o causa que provoca el conflicto, aunque dado que este nombre puede cambiar con el paso del tiempo, dentro de la BD cada conflicto se identificará mediante un código numérico sin significado alguno. Para cada conflicto se desea recoger los países a que afecta, así como el número de muertos y heridos contabilizados hasta el momento.*

En el primer párrafo se define qué es un conflicto bélico, cómo se identifica y qué información se necesita guardar. Es bastante claro que CONFLICTO es un tipo de entidad que se identifica por un código numérico llamado "Código".
Como un conflicto puede afectar a varios países se podrían considerar dos posibilidades de modelado:

- Pensar en PAIS como un tipo de entidad que se relaciona con CONFLICTO mediante el tipo de relación **Afecta**
- Representar País como un atributo multivaluado.

Si se siguen leyendo las especificaciones del problema se observará que no se vuelve a mencionar ninguna característica del país y por lo tanto se puede concluir que no es importante como tipo de entidad. Por lo que se elige la segunda opción mencionada. Los números de Muertos y Heridos serán atributos obligatorios del tipo de entidad CONFLICTO, cuyo valor por defecto sería cero.

> *Los conflictos pueden ser de distintos tipos según la causa que lo ha originado, calcificándose, a lo sumo, en cuatro grupos: territoriales, religiosos, económicos o raciales, en cada uno de estos grupos se recogerán diversos datos. En los conflictos territoriales se recogerán las regiones afectadas, en los*

religiosos, las religiones afectadas, en los económicos las materias primas disputadas y en los raciales las etnias enfrentadas.

En el segundo párrafo se presenta una clasificación de los conflictos en varios tipos, considerando esta clasificación como una jerarquía total y exclusiva Ver fig. 1.

PARTE II.

En los conflictos intervienen diversos grupos armados (al menos dos) y diversas organizaciones mediadoras (podría no haber ninguna). Los mismos grupos armados y organizaciones mediadoras pueden intervenir en diferentes conflictos. Tanto los grupos armados como las organizaciones mediadoras podrán entrar y salir del conflicto, en ambos casos se recogerá tanto la fecha de incorporación como la fecha de salida. Temporalmente, tanto un grupo armado como una organización mediadora podrán no intervenir en conflicto alguno.

Además de los grupos armados, intervienen organizaciones mediadoras, las cuales se considerarán como tipos de entidades que se interrelacionan con Conflicto.

Cada una de las interrelaciones poseerá dos atributos multivaluados F. I. Cont, F. F. Conf, F. I. Media, F. F. Media, que nos indica las fechas de entrada y de salida en los conflictos de los grupos armados y las organizaciones mediadoras, respectivamente. Se puede observar que las fechas de fin de las intervenciones, tanto de los grupos armados como de

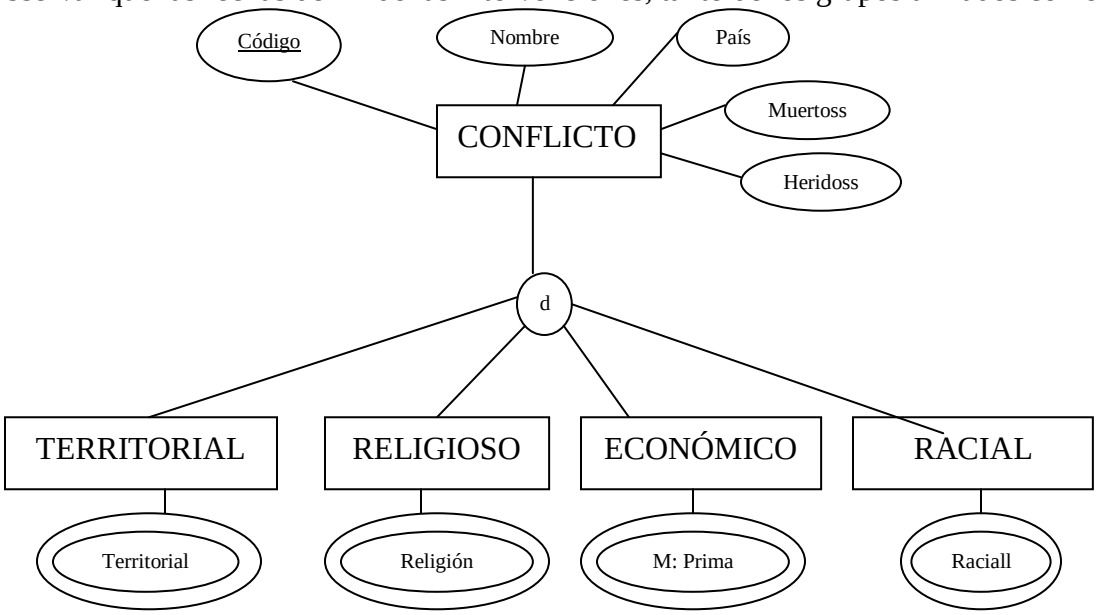


Fig. 3.17

Las organizaciones, son atributos opcionales, ya que puede darse el caso de no haber acabado con la intervención.

Para contemplar el hecho de que tanto un GRUPO ARMADO como una ORGANIZACIÓN pueden no participar en un conflicto de forma temporal, pondremos la cardinalidad mínima de ambas a cero. La cardinalidad máxima será n en las dos entidades.

También se indica en el texto que en un conflicto participan al menos dos grupos armados, con lo que la cardinalidad mínima de CONFLICTO en la interrelación **Interviene** es de 2 y la máxima de n.
Las cardinalidad de CONFLICTO en la interrelación **Media** son 0 para la mínima, ya que en las especificaciones se expone que puede haber conflictos en los que no participe ninguna organización, y n porque no existe limitación en el número de organizaciones que intervienen en un conflicto.

> *De cada grupo armado se recoge el código que se le asigna y un nombre. Cada grupo armado dispone de al menos una división y es liderado por al menos un líder político. Las divisiones de que dispone un grupo armado se numeran consecutivamente y se registra el número de barcos, tanques, avienes y hombres de que disponen, asimismo se recoge el número de bajas que ha tenido. Para los grupos armados se recoge el número de bajas como suma de las bajas producidas en todas sus divisiones.*

El tipo de entidad GRUPO ARMADO se identificará por un Código, el Nombre, suponiendo que este es único, será un atributo alternativo y los líderes políticos se podrían considerar como un atributo multivaluado, pero si leemos el problema hasta el final se comprobará que se necesita como tipo de entidad, ya que se relaciona con más tipos de entidad.
La cardinalidad mínima de GRUPO ARMADO en esta interrelación es 1, al tener al menos un líder político que lo lidera y su cardinalidad máxima es n, ya que no se tiene ninguna restricción acerca del número de líderes políticas asociados a un grupo armado en concreto. Las cardinalidades asociadas a LIDER POLÍTICO podrían ser 1 la mínima y n la máxima si se considera que sólo queremos en la B. D.'s aquellos líderes que se relacionen con algún grupo armado, como es el caso.
Por otro lado, detectamos el tipo de entidad débil DIVISIÓN que depende en identificación de GRUPO ARMADO. Todos los atributos que posee el tipo de entidad DIVISIÓN son obligatorios.
Aunque aparezca el número de bajas por cada división, en el enunciado se especifica que también se desea conocer el número de bajas total para cada grupo armado, y para reflejarlo consideremos un atributo en el tipo de entidad GRUPO ARMADO.

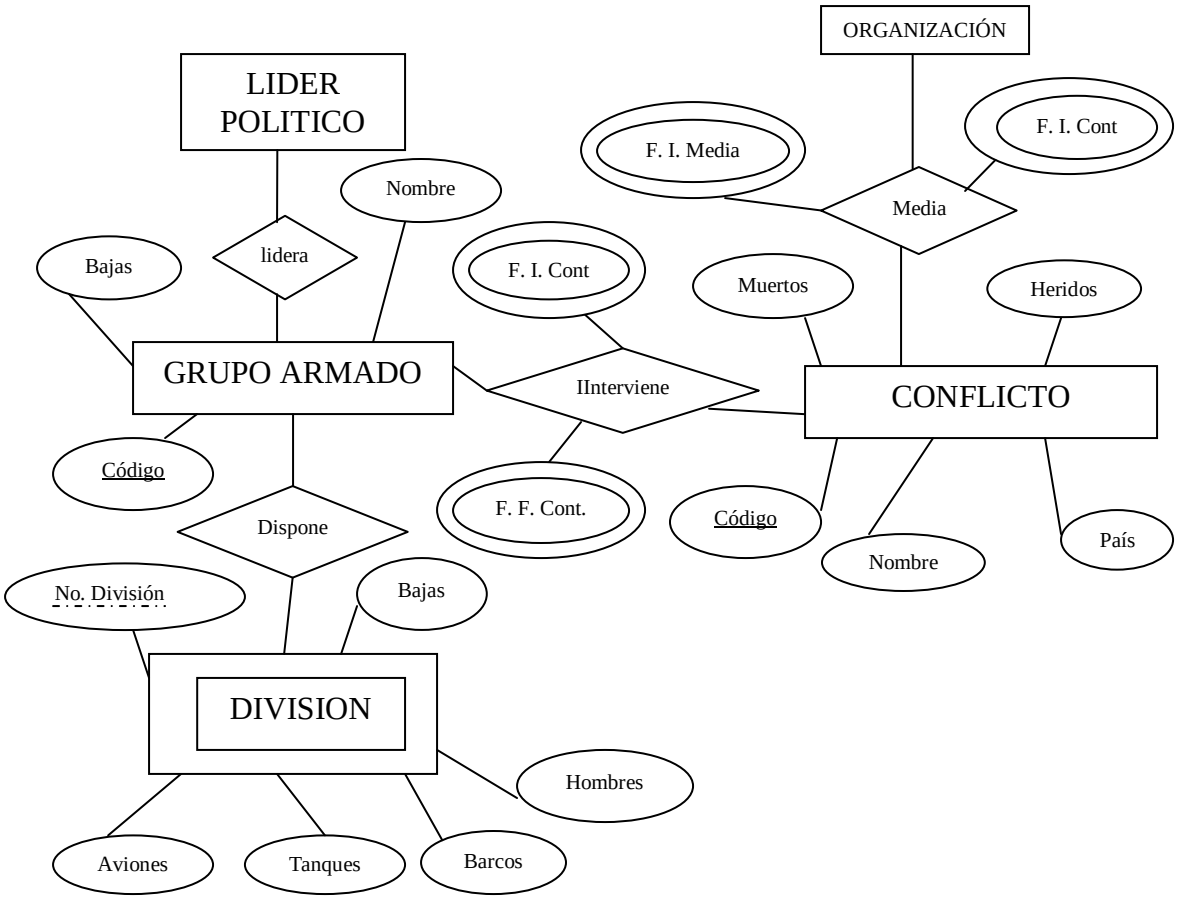


Fig. 3.18

PARTE III.

> *Los traficantes de armas suministran diferentes tipos de armas a los grupos armados. De cada tipo e armas se recoge un nombre y un indicador de su capacidad destructiva. De cada traficante se recoge un nombre, los diferentes tipos de armas que puede suministrar y cantidad de armas de cada uno de los tipos de arma que podría suministrar. Se mantiene el número total de armas de cada uno de los diferentes tipos de armas suministrado por cada traficante a cada grupo armado.*

En la frase "Los traficantes de armas suministran diferentes tipos de armas a los grupos armados" se presenta un tipo de relación entre tres tipos de entidad, TRAFICANTE, TIPO ARMA y GRUPO ARMADO, por lo que tenemos un posible tipo de relación ternario. Para hallar las cardinalidades fijamos dos tipos de entidad y vemos con cuántos ejemplares del otro tipo de entidad está relacionado, por ejemplo, fijamos Los tipos de entidad GRUPO ARMADO y TIPO DE ARMA, lo que significa que para un determinado grupo armado que posee se lo han podido suministrar 1 o n traficantes. El resto de las cardinalidades se hallarían de forma análoga y también serían (1, n). Podría haber dudas con la cardinalidad siguiente: un traficante que tiene un tipo de arma puede suministrárselo a varios grupos armados (cardinalidad 0,n)) que reflejaría las posibilidades de suministro), sin embargo, la anterior semántica se estará recogiendo con un tipo de relación binaria explícita **Puede Suministrar** que se analizará posteriormente, Se tiene por tanto que la semántica del tipo de relación **Suministra** considera únicamente los suministros efectivos, por lo que su cardinalidad es (1,n).

Este tipo de relación ternaria tendrá un atributo, N° de Armas, que indicará la cantidad de armas suministradas de cada tipo por un suministrador a los distintos grupos armados. Este atributo nos da una pista sobre la no utilización de tipo de relaciones binarias en lugar del anterior tipo de relación ternaria para considerar este supuesto, ya que no sería semánticamente equivalente.
Como llaves primarias de los dos nuevos tipos de entidades, TRAFICANTE y TIPO DE ARMA, se tendrán los nombres, suponiendo que éstos son únicos. Además, en el tipo de entidad TIPO DE ARMA se tendrá otro atributo Indicador que informará de la capacidad destructiva de cada ejemplar de dicho tipo de entidad, y que supone un valor obligatorio.
Para reflejar el tipo de arma que posee cada traficante y qué cantidad **Puede Suministrar** (es decir, su stock) se creará un tipo de relación binaria entre estos dos tipos de entidad y se indicará con un atributo Ex. Arma de dicho tipo de relación, la información sobre el stock de cada traficante.

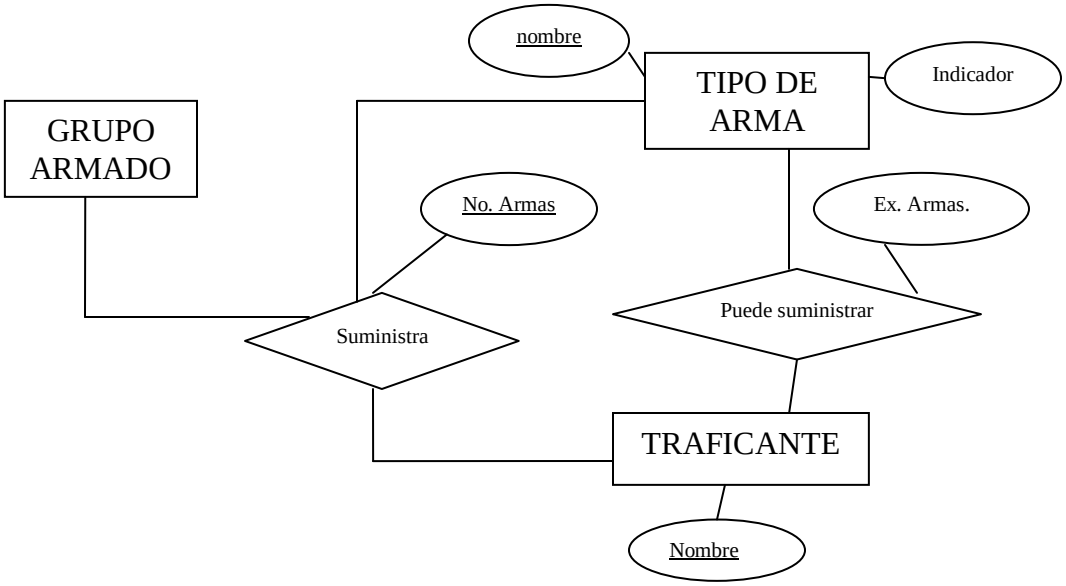


Fig. 3.19

PARTE IV.

> *Los líderes políticos se identifican por su nombre y por el código de grupo armado que lideran. Además se recoge una descripción textual de los apoyos que éste posee.*

Este párrafo es el que confirma la existencia del tipo de entidad LIDER POLÍTICO. Pero además, se añade que este tipo de entidad es débil respecto al tipo de entidad GRUPO ARMADO, por lo que se debe añadir al esquema. También se añade el tipo de relación **Lidera.**

> *Cada división la pueden dirigir conjuntamente un máximo de tres jefes militares, aunque cada jefe militar no dirige más de una división. A cada jefe militar se le identifica por un código, además se recoge el rango que éste posee y dado que un jefe militar no actúa por iniciativa propia sino que siempre obedece las órdenes de un único líder político de entre aquellos que lideran al grupo armado al que el jefe pertenece, se registrará el líder político al que obedece.*

Aparece un nuevo tipo de entidad JEFE MILITAR y dos nuevos tipos de relación, uno con DIVISIÓN para saber qué jefes militares dirigen cada una de las divisiones y otra con LÍDERES POLÍTICOS para conocer de quién recibe órdenes cada jefe militar.

El tipo de entidad JEFE MILITAR tiene su llave primaria que es el código y un atributo obligatorio que es Rango.
Las cardinalidades asociadas a JEFE MILITAR en el tipo de relación **Dirige**, que surge con el tipo de entidad DIVISIÓN, son (0,1) y en el sentido contrario de (1,3) ya que se especifica en el texto que una división la pueden dirigir como máximo tres jefes militares.
Siguiendo con el estudio de cardinalidades el tipo de relación **Obedece**, que asocia los tipos de entidades JEFE MILITAR y LÍDER POLÍTICO, tendrá (1,1) para JEFE MILITAR, ya que siempre obedece órdenes de un líder político y (0,n) para el otro tipo de entidad porque no todos los líderes políticos tienen el poder necesario para dar órdenes.

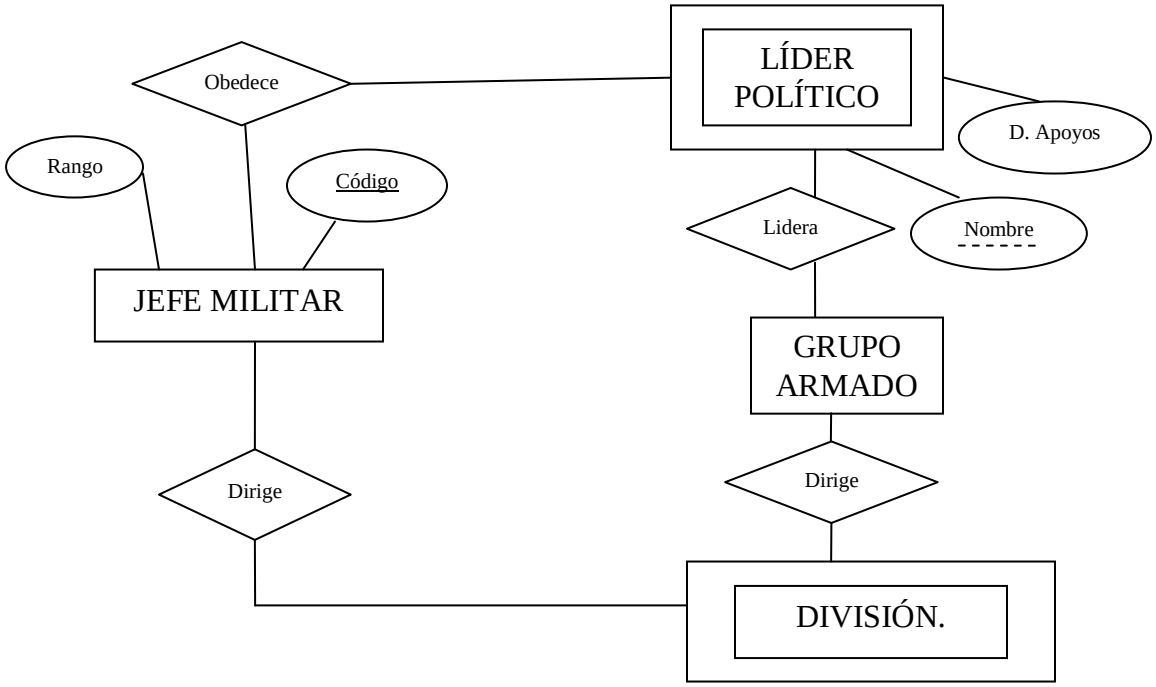


Fig. 3.20

PARTE V.

> *De las organizaciones mediadoras se recogerá su código, su nombre, su tipo (gubernamental, no gubernamental o internacional), la organización de que depende (una como máximo), el número de personas que mantiene desplegadas en cada conflicto y el tipo de ayuda que presta en cada conflicto que será de uno y sólo uno de los tres tipos siguientes, médica, diplomática o presencial.*

En este párrafo se indican los atributos del tipo de entidad ORGANIZACIÓN, la llave será Código, una llave alterna que será nombre y un atributo Tipo cuyos valores pueden ser gubernamentales, no gubernamentales o internacionales. Este último atributo se podría haber pensado como una clasificación de ORGANIZACIÓN, es decir una jerarquía, pero se puede prescindir de ella, ya que los subtipos ni tienen atributos propios ni se relacionan con otros tipos de entidad, con lo que se recoge la misma semántica definiendo un dominio para este atributo.
Como aparece en el texto que una organización puede depender de otra, se tendrá en el esquema un tipo de relación reflexivo, **Depende**, cuyas cardinalidades son (0,1) para

representar este hecho y (0,n) para recoger que una organización puede representar a cero o varias organizaciones.
También se desea guardar información sobre las unidades que despliega en cada conflicto, que se representará como atributo del tipo de relación **Media** al igual que el tipo de ayuda que se definirá dentro de un dominio cuyos valores serán los tipos de ayuda expuestos en el párrafo.
El que dentro de un conflicto una organización sólo pueda prestar un tipo de ayuda simplifica la solución. Una hipótesis más general supondría bien tener que introducir un atributo multivaluado, bien añadir un nuevo tipo de entidad y promocionar el tipo de relación **Media** a un grado superior a dos.

> *Con diversos fines, los líderes políticos dialogan con las organizaciones; se desea recoger explícitamente esta información. Así para cada líder se recogerán aquellas organizaciones con que dialoga y viceversa.*

Para recoger que los líderes políticos pueden mantener conversaciones con las organizaciones mediadoras y viceversa, crearemos un nuevo tipo de relación para completar el esquema.

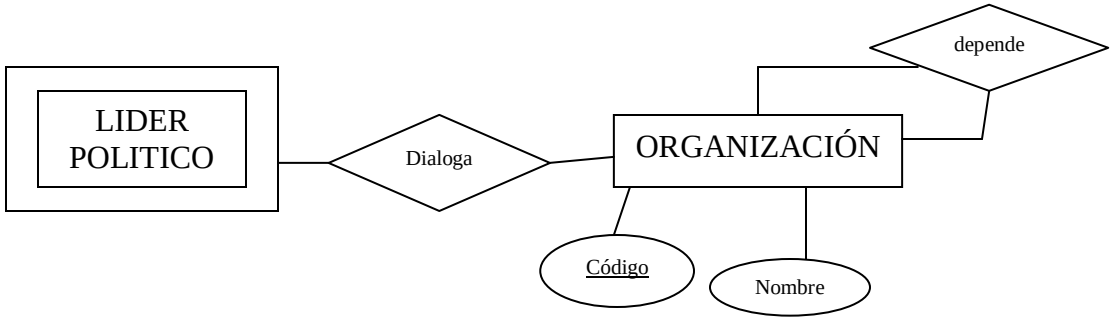


Fig. 3.21

SEMANTICA NO REFLEJADA.

- Secuencialidad del número de división de un grupo armado.

3.5 Preguntas de repaso:

1. Explique la diferencia entre entidad y tipo de entidad
2. Defina atributo
3. Defina los tipos de atributos
4. Comente que es el tipo de relación y explique la forma de descubrir el tipo de relación en los requerimientos
5. Explique en qué momento un atributo pertenece al tipo de relación y coloque un ejemplo.
6. Comente que es razón de cardinalidad y tipo de participación.
7. Comente sobre el grado del tipo de relación
8. Comente la diferencia entre llave primaria, llave candidata y llave parcial
9. Comente cuando un tipo de entidad es un tipo de entidad identificador.
10. Defina herencia simple, herencia múltiple y categoría
11. Comente la diferencia entre disyunción, sobre posición y unión.

3.6 EJERCICIOS

En cada uno de los requerimientos diagramar su modelo ERE.

A. CAMPEONATO DE AJEDREZ.

El club de ajedrez de Villatortas de Arriba, ha sido encargado por la Federación Internacional de Ajedrez de la organización de los próximos campeonatos mundiales que se celebrarán en la mencionada localidad. Por este motivo, desea llevar una base de datos toda la gestión relativa a participantes, alojamiento y partidas. Teniendo en cuenta que:
En el campeonato participan jugadores y árbitros; de ambos se requiere conocer el número de asociado, nombre, dirección, teléfono de contacto y campeonatos en los que han participado (como jugador o como árbitro). De los jugadores se precisa además el nivel de jugador en una escala de 1 a 10.
Ningún árbitro puede participar como jugador.
Los países envían al campeonato un conjunto de jugadores y árbitros, aunque no todos los países envían participantes. Todo jugador y árbitro es enviado por un único país. Un país puede ser representado por otro país.
Cada país se identifica por un número correlativo según su orden alfabético e interesa conocer además de su nombre, el número de clubes de ajedrez existentes en el mismo.
Cada partida se identifica por un número correlativo (Cód_P), la juegan dos jugadores y la arbitra un árbitro. Interesa registrar las partidas que juega cada jugador y el color (blancas o negras) con el que juega. Ha de tenerse en cuenta que un árbitro no puede arbitrar a jugadores enviados por el mismo país que le ha enviado a él.
Todo participante participa en al menos una partida.
Tanto jugadores como árbitros se alojan en uno de los hoteles en los que se desarrollan las partidas, se desea conocer en qué hotel y en qué fechas se ha alojado cada uno de los participantes. Los participantes pueden no permanecer en Villatortas durante todo el campeonato, sino acudir cuando tienen que jugar alguna partida alojándose en el mismo o distinto hotel. De cada hotel, se desea conocer el nombre, la dirección y el número de teléfono.

El campeonato se desarrolla a lo largo de una serie de jornadas (día, mes, año) y cada partida tiene lugar en una de las jornadas aunque no tengan lugar partidas todas las jornadas.

Cada partida se celebra en una de las salas de las que pueden disponer los hoteles, se desea conocer el número de entradas vendidas en la sala para cada partida. De cada sala, se desea conocer la capacidad y medios de que dispone (radio, televisión, video…) para facilitar la retransmisión de los encuentros. Una sala puede disponer de varios medios distintos.

De cada partida se pretende registrar todos los movimientos que la componen, la identificación de movimiento se establece en base a un número de orden dentro de cada partida: para cada movimiento se guardan la jugada (5 posiciones) y un breve comentario realizado por un experto.

B. ENERGÍA ELÉCTRICA.

Se pretende llevar a cabo un control sobre la energía eléctrica que se produce y consume en un determinado país. Se parte de las siguientes hipótesis.
Existen productores básicos de electricidad que se identifican por un nombre, de los cuales interesa su producción media, producción máxima y fecha de entrada en funcionamiento. Estos productores básicos lo son de una de las siguientes categorías: Hidroeléctrica, Solar, Nuclear o Térmica. De una central hidroeléctrica o presa nos interesa saber su ocupación, capacidad máxima y número de turbinas. De una central solar nos interesa saber la superficie total de paneles solares, la media anual de horas de sol y tipo (fotovoltaica o termodinámica). De una central nuclear, nos interesa saber el número de reactores que posee, el volumen de plutonio consumido y el de residuos nucleares que produce. De una central térmica, nos interesa saber el número de hornos que posee, el volumen de carbón consumido y el volumen de su emisión de gases.
Por motivos de seguridad nacional interesa controlar el plutonio de que se provee una central nuclear, este control se refiere a la cantidad de plutonio que compra a cada uno de sus posibles suministradores, (nombre y país), y que porta un determinado transportista (nombre y matricula), ha de tenerse en cuenta que el mismo suministrador puede vender plutonio a distintas centrales nucleares y que cada porte,(un único porte por compra), puede realizarlo un transportista diferente.

Cada día, los productores entregan la energía producida a una o varias estaciones primarias, las cuales pueden recibir diariamente una cantidad distinta de energía de cada uno de estos productores. Los productores entregan siempre el total de su producción. Las estaciones primarias se identifican por su nombre y tienen un número de transformadores de baja a alta tensión y son cabecera de una o varias redes de distribución.

Una red de distribución se identifica por un número de red y sólo puede tener una estación primaria como cabecera. La propiedad de una red puede ser compartida por varias compañías eléctricas, a cada compañía eléctrica se identifica por su nombre.

La energía sobrante en una de las redes puede enviarse a otra red. Se registra el volumen total de energía intercambiada entre dos redes.

Una red está compuesta por una serie de líneas, cada línea se identifica por un número secuencial dentro del número de red y tienen una determinada longitud. La menor de las líneas posible abastecerá al menos a dos subestaciones.

Una subestación es abastecida sólo por una línea y distribuye a una o varias zonas de servicio, a tales efectos, las provincias (código y nombre), se encuentran divididas en tales zonas de servicio, aunque no puede haber zonas de servicio que pertenezcan a más de una provincia. Cada zona de servicio puede ser atendida por más de una subestación.

En cada zona de servicio se desea registrar el consumo medio y el número de consumidores finales de cada una de las siguientes categorías: particulares, empresas e instituciones.

C: GESTIÓN DE NÓMINA.

Enunciado.
Una empresa decide informatizar su nómina. Del resultado del análisis realizado, se obtiene la siguiente información:

A cada empleado se le entregan múltiples justificantes de nómina a lo largo de su vida laboral en la empresa y al menos uno mensual.
A cada empleado se le asigna un número de matrícula en el momento de la incorporación a al empresa, y éste es el número usado a efectos internos de identificación. Además se registran el RFC del empleado, nombre, número de hijos, porcentaje de retención para Hacienda, datos de cuenta corriente en la que se le ingresa el dinero (banco, sucursal y número de cuenta) y departamentos en los que trabaja. Un empleado puede trabajar en varios departamentos y en cada uno de ellos trabajará con una función distinta.
De cada departamento se mantiene su nombre y cada una de sus posibles sedes.
Son datos propios de un justificante de nómina el ingreso total percibido por el empleado y el descuento total aplicado. La distinción entre dos justificantes de nómina se hará, además de mediante el número de matrícula del empleado, mediante el ejercicio fiscal y número de mes al que pertenece y con un número de orden en el caso de varios justificantes de nómina recibidos el mismo mes.
Cada justificante de nómina consta de varias líneas (al menos una de ingresos) y cada línea se identificará por un número de línea del correspondiente justificante. Una línea puede corresponder a un ingreso o a un descuento. En ambos casos, se recoge la cantidad que corresponde a la línea (en positivo si se trata de un ingreso o en negativo si se trata de un descuento); en el caso de los descuentos, se recoge la base sobre la cual se aplica y el porcentaje que se aplica para el cálculo de éstos.
Toda línea de ingreso de un justificante de nómina responde a un único concepto retributivo. En un mismo justificante, puede haber varias líneas que respondan al mismo concepto retributivo. De los conceptos retributivos se mantiene un código y una descripción.
De cara a la contabilidad de la empresa, cada línea de un justificante de nómina se imputa al menos a un elemento de coste. Al mismo elemento de coste pueden imputársele varías líneas. Para cada elemento de coste, se recoge un código, una descripción y su saldo.
Entre los elementos de coste se establece una jerarquía, en el sentido de que un elemento de coste puede contener a otros elementos de coste, pero un elemento de coste sólo puede estar contenido en, a lo sumo, otro elemento de coste.
En determinadas fechas, que se debe recoger, cada elemento de coste se liquida con cargo a varios apuntes contables (código y cantidad) y a una o varias transferencias bancarias, de las que se recogen los datos de cuenta corriente (banco, sucursal y número de cuenta) y la cantidad. Por cada apunte contable y transferencia bancaria se pueden liquidar varios elementos de coste.

4. MODELOS DE IMPLEMENTACIÓN

(Elmasri & Navate, 2008)
(Reinosa, Maldonado, Nuñoz, Damiano, & Abrustsky, 2012)

LOS MODELOS LOGICOS PRIMITIVOS.
(Los modelos lógicos antes del modelo relacional)

Los sistemas de gestión de bases de datos organizan y estructuran los datos de tal modo que puedan ser recuperados y manipulados por usuarios y programas de aplicación. Las estructuras de datos y las técnicas de acceso proporcionadas por un DBMS particular se denominan su *modelo de datos*. El modelo de datos determina la <<personalidad>> de un DBMS, y las aplicaciones para las cuales está particularmente bien conformado. En este capítulo se describe el modelo de datos relacional y lo compara con otras estrategias de organización de bases de datos.

MODELOS DE DATOS PRIMITIVOS.
Cuando la gestión de datos se popularizó durante los setenta y los ochenta emergieron un puñado de modelos de datos populares. Cada uno de estos primeros modelos de datos tenía ventajas y desventajas que jugaron papeles importantes en el desarrollo del modelo de datos relacional. En muchos sentidos el modelo de datos relacional representó un intento de simplificar los modelos de datos anteriores.

Antes de la introducción de los sistemas de gestión de bases de datos, todos los datos permanentemente almacenados en un sistema informático, tales como la nómina y los registros de contabilidad, se almacenaban en archivos individuales. Un sistema de gestión de archivos, generalmente proporcionado por el fabricante del computador como parte del sistema operativo, llevaba la cuenta de los nombres y ubicación de los archivos. El sistema de gestión de archivos básicamente no tenía un modelo de datos; no sabía nada acerca de los contenidos internos de los archivos. Para el sistema de gestión de archivos, un archivo que contuviera un documento de procesamiento de textos y un archivo que contuviera datos de nómina aparecían igual.
El conocimiento acerca del contenido de un archivo –qué datos contuviera y cómo estuvieran organizados- estaba incorporado a los programas de aplicación que utilizaba el archivo, tal como lo muestra la figura 4.1. En esta aplicación de nóminas, cada uno de los programas COBOL que procesaban el archivo maestro de empleados contenía una descripción de archivo (DA) que describía la composición de los datos en el archivo. Si la estructura de los datos cambiaba –por ejemplo, si un ítem adicional de datos fuera a ser almacenado por cada empleado- todos los programas que accedían al archivo tenían que ser modificados. Como el número de archivos y programas crecía con el tiempo, todo esfuerzo de procesamiento de datos de un departamento se perdía en mantener aplicaciones existentes en lugar de desarrollar otras nuevas.
Los problemas de mantener grandes sistemas basados en archivos condujeron a finales de los sesenta a desarrollar sistemas de gestión de bases de datos. La idea detrás de estos sistemas era sencilla; tomar la definición de los contenidos de un archivo y la estructura de los programas individuales, y almacenarla, junto con los datos, en una base de datos. Utilizando la información de la base de datos, el DBMS que la controlaba podría tomar

un papel mucho más activo en la gestión de los datos y en los cambios a la estructura de la base de datos.

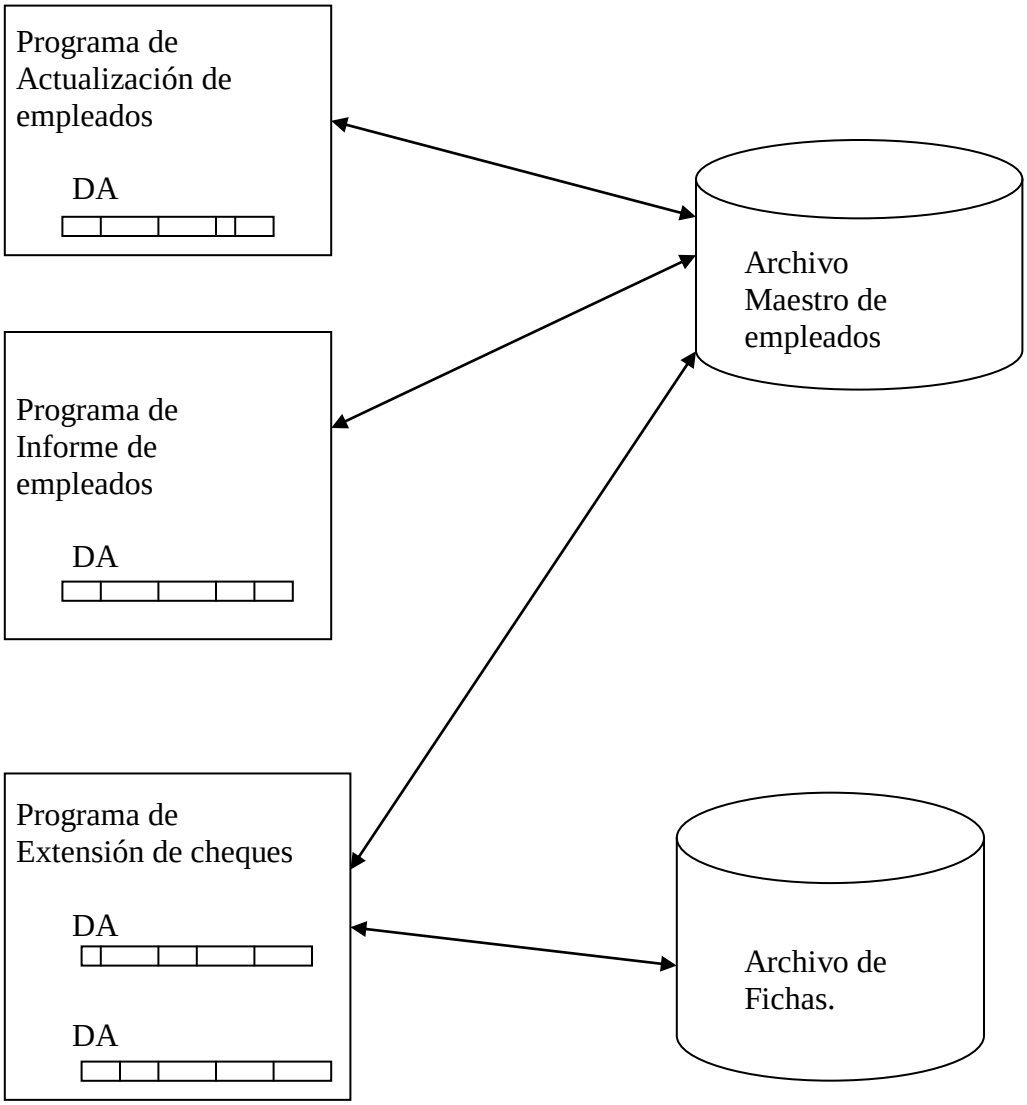


Fig 4.1. Aplicación de nómina utilizando un sistema de gestión de archivos.

4.1 ENFOQUE JERÁRQUICO DE LA BASE DE DATOS.

El primer enfoque a los sistemas de manejo de bases de datos que se considerará es el jerárquico. Ya que uno de los DBMS más antiguo y de empleo más amplio se basa en ese enfoque (IMS, Information Management System de IBM).
El árbol o jerarquía es una interconexión de nodos sin que exista un ciclo con las restricciones que siguen: Existe un nodo especial llamado raíz que siempre se coloca en la parte superior del diagrama. Las ramas del árbol crecen hacia abajo a partir de la raíz. Todo nodo (con excepción de la raíz) está conectado hacia arriba con un solo nodo. Existirán nodos padres nodos hijos, nodos hojas, nodos hermanos, nodos intermedios.

La jerarquía se definirá con un ejemplo.

Liga de Béisbol.
Es necesario mantener información de cada equipo de liga, incluyendo los datos de jugadores, de todos los entrenadores, de los distintos tipos de experiencia profesional anterior que tenga cada entrenador, y los bates, (cada bate tiene un número único de serie) que tiene cada equipo.
Cada nodo debe representar un distinto tipo de registro. Cada registro tendrá un cierto número de campos.
En este caso, el registro "equipo" tendrá los campos que describen al equipo como: nombre del equipo, ciudad del equipo y gerente del equipo, etc.
Cada rama de la siguiente figura representa una relación 1:N; por lo tanto, en cada equipo hay muchos jugadores, entrenadores y bates. Y por cada entrenador hay varios elementos de experiencia profesional.

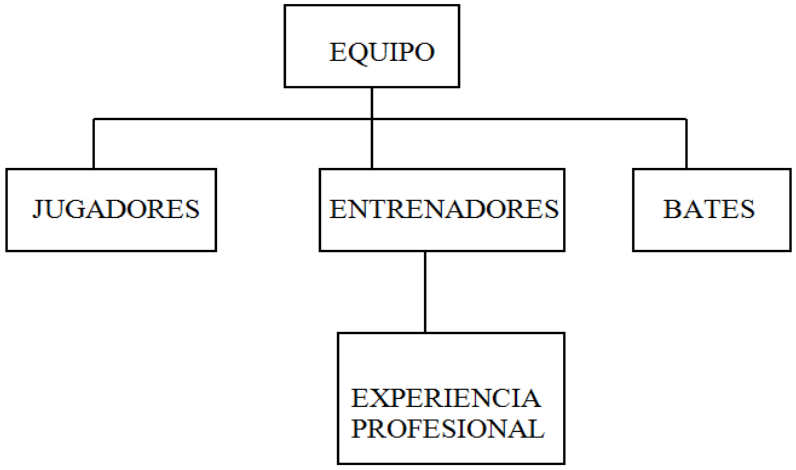


Fig. 4.2 Jerarquía de los equipos de béisbol.

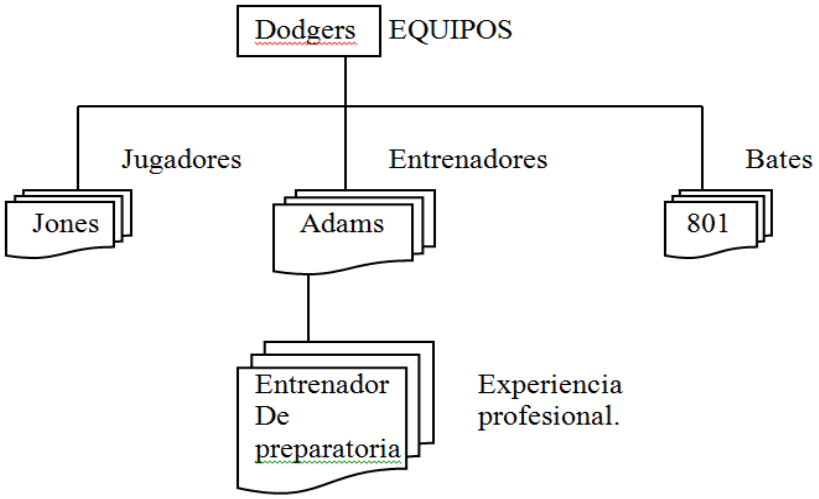


Fig. 4.3 Ocurrencia de los Dodgers en la jerarquía de los equipos de béisb

.

Limitaciones:

- No maneja relaciones N:M
- La forma de ver los registros es desde el nodo raíz.

Uno desea también incluir en los jugadores los datos de estadísticas y posición. Además que un jugador tal vez no esté jugando con un equipo específico en esa temporada. El modelo podría quedar de la siguiente forma:

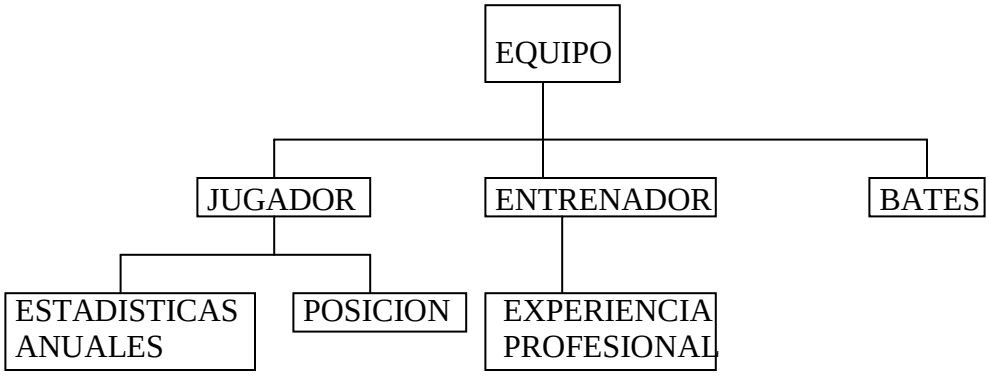


Fig. 4.4 jerarquía combinada de equipos y jugadores.

El diagrama de la figura 4.4 no procede ya que no es posible tener acceso a un jugador que en una temporada específica no esté inscrito en un equipo de béisbol

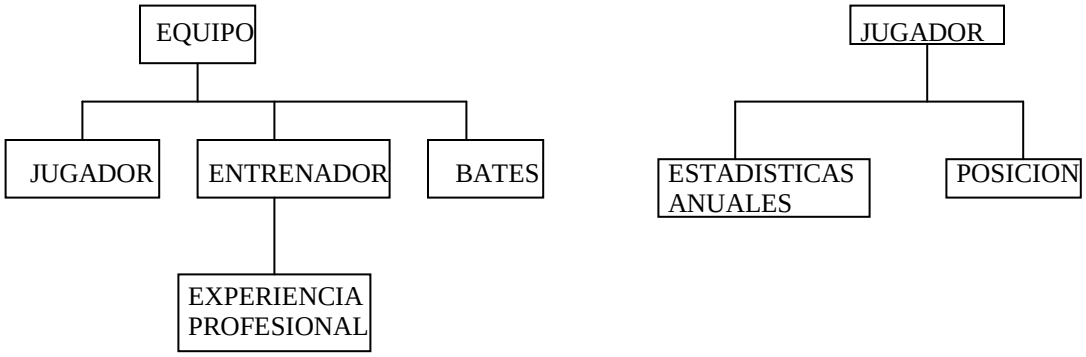


Fig.4.5 Jerarquía de equipos y jugadores.

Una desventaja del modelo presentado en la figura 4.5 es que existe redundancia de información ya que se repite la información del jugador en las dos jerarquías.
Lo conveniente es tener un acceso más flexible y almacenamiento menos restrictivo, además de reducir la redundancia. Por lo que se desarrollaron relaciones lógicas.
Esto se realiza sustituyendo el segmento jugadores con un tipo especial de segmento apuntador conocido como ‘segmento hijo lógico’. De esta forma se respetan las dos jerarquías pero a la vez una jerarquía se integra a la otra.

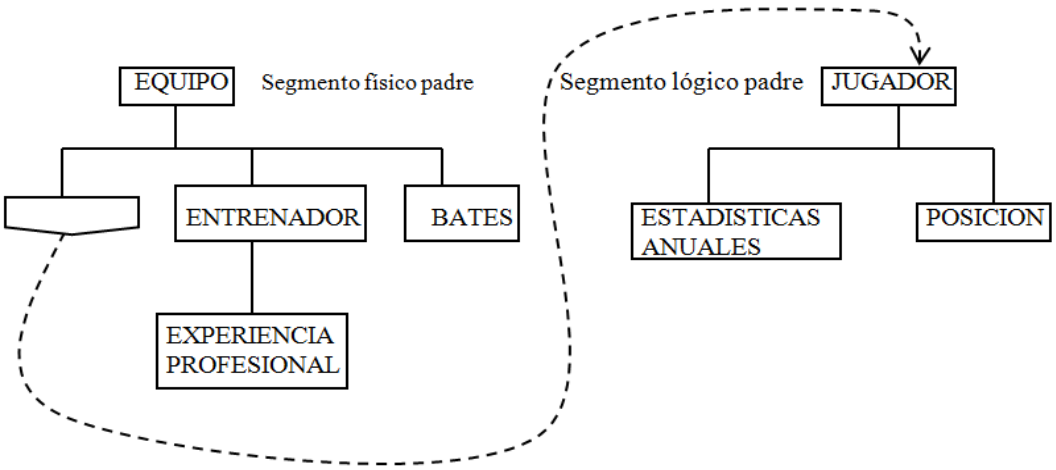


Fig. 4.6 Relación lógica unidireccional.

De esta forma tenemos:

1. Se evita la redundancia de información al tener 2 jerarquías independientes.
2. Se permite un enfoque más amplio en el registro jugadores (jugadores que en este momento no estén jugando en un equipo estarían incluidos).
3. Se tiene la posibilidad de tener acceso a jugadores en forma directa o por medio de equipo. (cuáles jugadores juegan en el equipo X?)

4.1.1 Intersección de datos.

La intersección de datos es la información descrita acerca de la relación entre dos entidades que no pueden ir en ninguno de sus segmentos. Por ejemplo, los años que un jugador ha participado en un equipo en especial. Esa información puede ser parte de equipo o parte de jugador.

Para representar esta información se puede aprovechar el segmento hijo lógico. En el segmento hijo lógico existe una ocurrencia diferente del segmento hijo lógico por cada equipo y jugador relacionados. Por lo tanto, la ocurrencia de hijo lógico que conecta a 2 ocurrencias de entidad es el lugar natural para almacenar tal intersección.

4.1.2 Relación muchos a muchos.

Una jerarquía simple no puede manejar la cardinalidad M:N.

Supóngase que la naturaleza de los datos es histórica, de manera que interesa llevar un control de todo equipo en que ha participado un jugador especifico, y de todos los jugadores que alguna vez han participado en un equipo determinado. Esta es una relación M:N y se puede implantar por medio de la estructura de relación lógica dual o bidireccional.

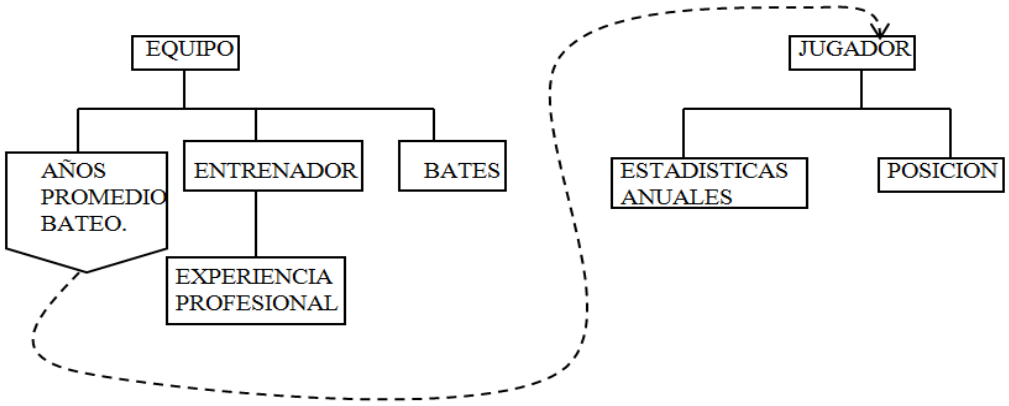


Fig. 4.7 Datos de Intersección.

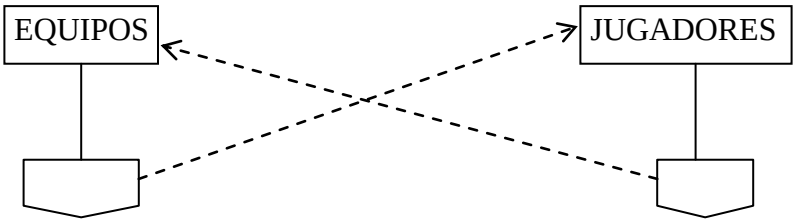


Fig. 4.8 Relación lógica bidimensional para una relación muchos a muchos.

Por ejemplo, los Dodgers han tenido a Doe y Jones en su equipo; los Gigantes a Jones y Smith. Por el contrario, Doe ha jugado para los Dodgers, Jones para los Dodgers y los Gigantes, y Smith con los Gigantes. Los apuntadores hijo, gemelo y padre lógico operan como antes, pero en ambas direcciones.

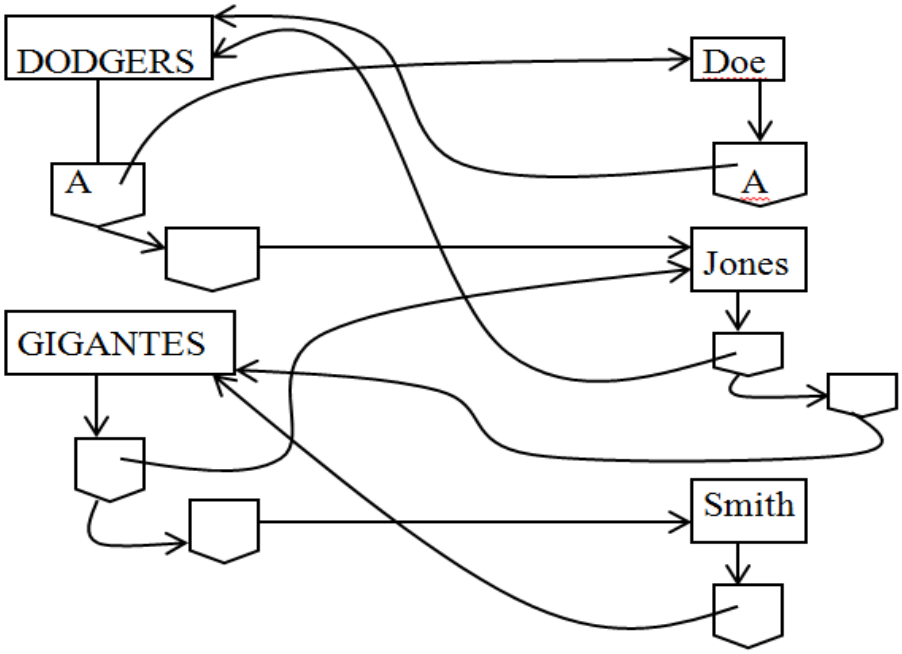


Fig. 4.9 Una ocurrencia de la relación lógica bidireccional.

En este arreglo hay dos ocurrencias de hijo lógico por cada coincidencia entre un jugador y un equipo, específicamente una bajo el jugador incluido y una bajo el equipo implicado. Las dos casillas con la letra 'A' de la figura 4.9 conectan a los Dodgers con Doe. De paso se observará que si existe la intersección de datos, como en cuanto al número de años que un jugador ha participado en un equipo en este ejemplo, esto se almacena en los dos segmentos hijos lógicos adecuados. Si alguna vez hay cambio en un lugar, el sistema automáticamente actualiza el otro.

Por cierto, pareció extraño hace poco cuando se hablaba acerca de los padres 'lógicos' y 'físicos' de un segmento hijo lógico? Se afirmó que el segmento hijo lógico puede tener dos tipos de segmento padre, uno físico y otro lógico. No viola eso el hecho de que en un árbol un nodo puede tener sólo un padre? Sí y no.

Las dos estructuras separadas de la figura 4.5 son ambas claramente jerarquías. Las estructuras combinadas que se muestran en las figuras 4.6 y 4.8 son, técnicamente, redes restringidas. Lo que importa, en términos de IMS, es que aunque técnicamente el empleo de relaciones lógicas crea redes en estructura física subyacente, los programadores siempre escriben sus programas con base en jerarquías.

Todas las estructuras de base de datos IMS que se han considerado hasta aquí se conocen como bases de datos físicas IMS. Los programadores escriben su código de acuerdo con las bases de datos lógicas IMS. En realidad las bases de datos lógicas son una descripción de lo que los programadores deben considerar que es el aspecto de los datos. Los segmentos de las bases de datos lógicas se mapean sobre segmentos de las bases de datos físicas, que contienen los datos propiamente dichos. Una base de datos lógica IMS, en el caso más simple, es un subárbol de una jerarquía física IMS, consta de un nodo y algunos o todos sus descendientes. El nodo inicial, esto es, la raíz de la base de datos lógica debe ser la raíz de la base de datos física o un nodo que apunta un índice secundario.

La figura 4.10 es una base de datos lógica obtenida de las jerarquías físicas de equipo y jugadores y de la relación lógica entre ellas. Compárese las figuras 4.10 y 4.6 y véase la forma en que la relación lógica que conecta a las dos bases de datos físicas se vuelve una rama en la base de datos lógica.

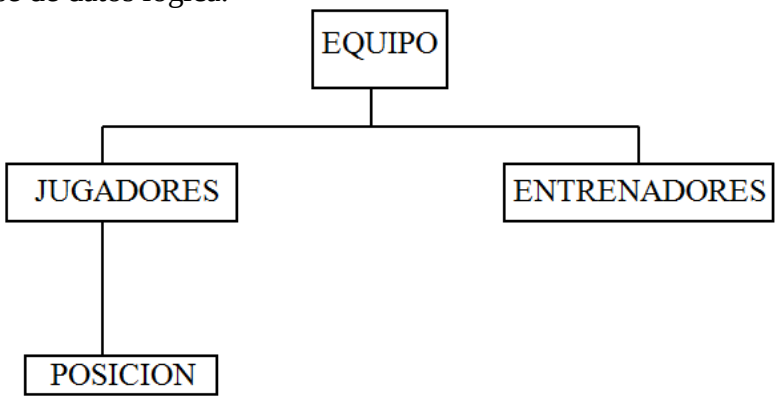


Fig. 4.10 Base de da tos lógica obtenida a partir de dos jerarquías físicas conectadas por una relación lógica.

La fig. 4.11 es una descripción más detallada de la jerarquía equipos de la figura 4.2. En este caso es una base de datos lógica que abarca toda la base de datos física de equipos. No sólo muestra los segmentos contenidos en la jerarquía, sino que además los campos dentro de éstos. Tanto los nombres de segmento como de campo se abrevian en forma adecuada para utilizarlos en proposiciones de programa. El campo izquierdo de cada segmento es un campo llave único. El segmento raíz debe tener un campo de este tipo, pero los segmentos subordinados pueden o no tenerlo. La figura 4.12 es la ocurrencia de la jerarquía equipos para los Dodgers (registro de base de datos de los Dodgers) que primero se mostró en la figura 4.3 pero esta vez incluyendo las ocurrencias de campo.

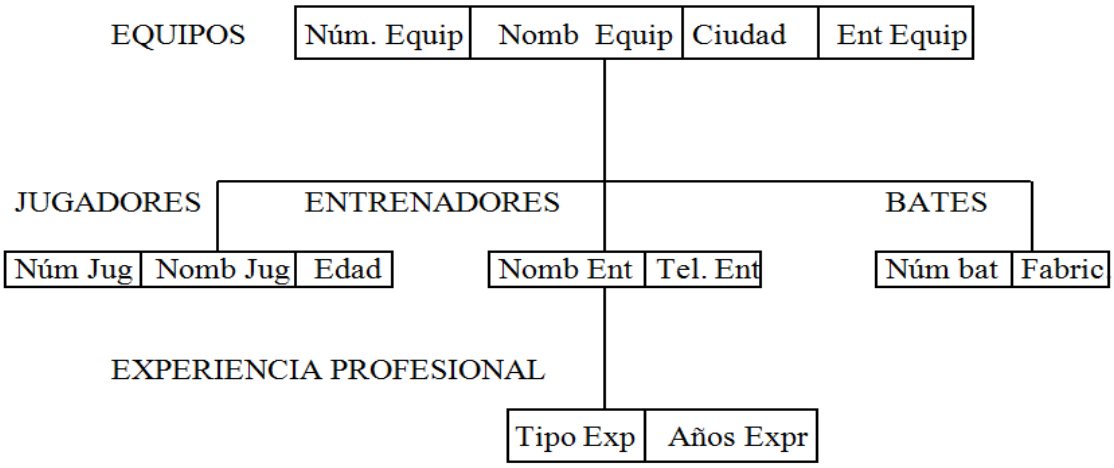


Fig. 4.11 Jerarquía de los equipos en la que se muestran los nombres de segmento y de campo.

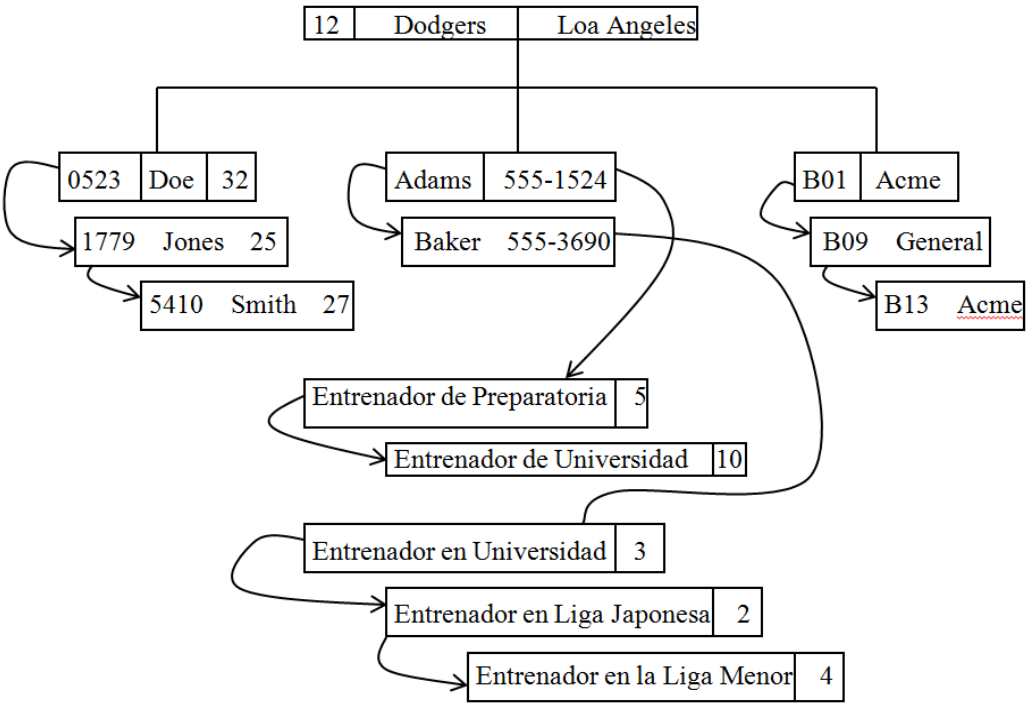


Fig. 4.12 Ocurrencias de campo en el registro de base de datos de los Dodgers.

4.1.3 Restricciones de Integridad.

1. Ningún registro puede existir en el modelo jerárquico sin ser relacionado a un registro padre. Excepto el registro raíz. Esto tiene las siguientes implicaciones:
   a. Un registro hijo no puede ser insertado al menos que sea ligado a uno padre.
   b. Un registro hijo puede ser borrado independientemente de su padre; sin embargo, el borrar un registro padre automáticamente borra a todos sus hijos.
   c. Las reglas a. y b. no aplican a los registros padres virtuales. Cuando existe un hijo virtual apuntando a un padre, no es posible borrar el registro mientras se apunte al padre desde un hijo virtual.
2. Si un hijo tiene 2 o más papás del mismo tipo de registro, el hijo tiene que duplicarse siendo un hijo para cada papá.
3. Un hijo, teniendo dos o más papás de diferentes tipos de registro, puede realizarse siempre y cuando tenga sólo un padre real y los demás se representan como padres virtuales.
4. Si un registro duplicado es modificado, es responsabilidad del programa de asegurarse de que todas las copias sean modificadas de la misma forma.

4.1.4 Algoritmo que traduce del modelo Entidad – Relación al Jerárquico

En el modelo jerárquico sólo existen los tipos de restricción 1:N como una relación padre a hijo. En adición, un tipo de registro puede tener a lo más un tipo de registro padre; de aquí que, los tipos de relación M:N son difíciles de representar. Dos formas de hacerlo es:

1. Se representa el tipo de relación M:N como si fuera un tipo de relación 1:N. En este caso, las instancias de registros en la parte N-aria son duplicados porque cada registro puede ser relacionado con varios padres. Este tipo de representación mantiene todos los tipos de registro en una jerarquía simple, con un costo de repetición de registros. El programa que actualiza la base de datos debe mantener la consistencia de copias duplicadas. Para mostrar esta opción se presenta en la figura 4.13 el modelo jerárquico de los requerimientos de COMPAÑIA vistos en el capítulo III.

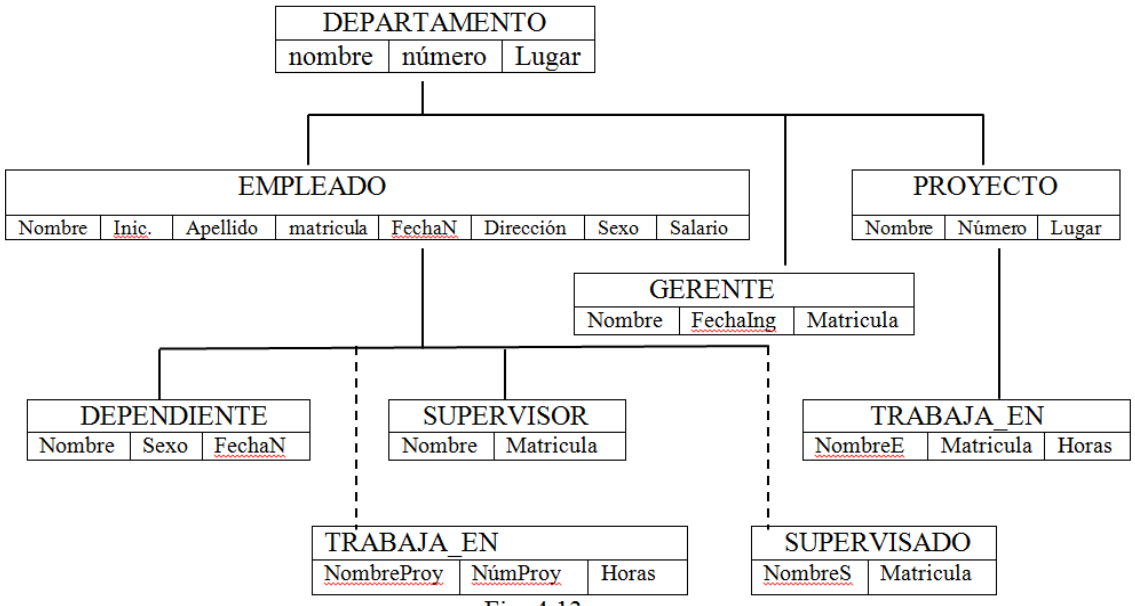


Fig. 4.13

2. Crear más de una jerarquía y tener tipos de vínculos padre-hijo virtuales (VPHV) (apuntadores lógicos) a partir de un tipo de registros que aparece en una jerarquía al tipo de registro raíz de otra jerarquía. Estos apuntadores pueden servir para representar el tipo de vínculo M:N. Aun así, una restricción, adoptada del modelo IMS, impide que un tipo de registros tenga más de un hijo virtual. La figura 4.14 muestra ésta opción utilizando de la misma manera los requerimientos de COMPAÑA.

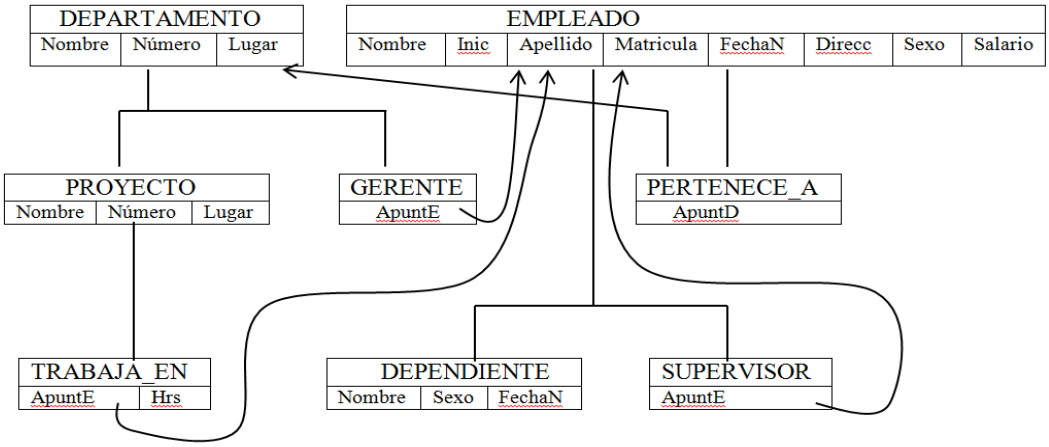


Fig. 4.14

Como se deben considerar múltiples opciones, no existe un método estándar para transformar un esquema ER a un jerárquico.

Por último, consideremos la transformación de los tipos de vínculo o tipos de relación n-arios, con n>2. La figura 4.15 muestra dos opciones para establecer la transformación del tipo de relación ternario SUMINISTRA mostrado al final del capítulo 3.

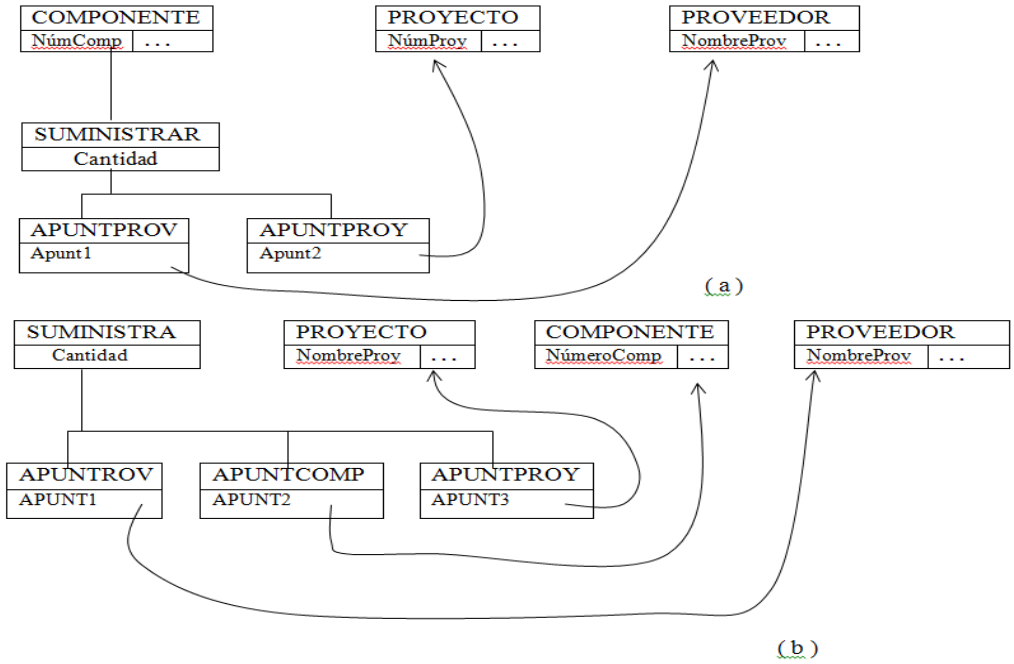


Fig. 4.15 Transformación de un tipo de relación o vinculo n-airo (n=3) SUMINISTRAR del modelo ER al jerárquico. (a) Una opción para representar un vinculo ternario. (b) Representación de un vínculo ternario con tres tipos de VPHV.

Es evidente que el modelo jerárquico ofrece muchas opciones para representar el mismo esquema ER., y podríamos haber empleado muchas otras representaciones además de las que vimos. Las cuestiones de acceso eficiente a los datos y de limitar la redundancia vs facilitar la obtención son importantes al elegir una representación específica. En general, se considera que el modelo jerárquico es inferior, en cuanto a su capacidad de modelado, que el modelo relacional y el modelo de red, por las siguientes razones:

- Los tipos de vínculos M:N sólo pueden representarse añadiendo registros redundantes o usando vínculos padre-hijo virtuales y registros apuntadores.
- Todos los tipos de vínculo 1:N en una jerarquía se deben mantener en la misma dirección.
- Un tipo de registros en la jerarquía puede tener cuando más un tipo de registros propietario (real)
- Un tipo de registro puede tener cuando más dos padres: uno real y uno virtual (Esta limitación es especifica de IMS).

4.2 MODELO DE DATOS DE RED.

Los DBMS orientados a redes fueron desarrollados por la Conference of Data Systems Languages (CODASYL) y su Data Base Task Group (DBTG) y Data Description Language Committee (DDLC). CODASYL es una organización constituida por representantes voluntarios de fabricantes y usuarios de computadoras en la industria y en los gobiernos federales de EEUU y Canadá.
Las especificaciones para la DBMS se iniciaron a finales de los 60's. Algunos de los documentos más notorios se presentaron en 1971, 1973, 1978 y 1982.

Red.- Es un conjunto de puntos interconectados de alguna forma. La conexión entre dos puntos se conoce como arista. Dos puntos son adyacentes si están conectados por una arista. El grado de un nodo es el número de aristas que inciden en él. Una ruta es un conjunto de aristas conectadas en serie. Un ciclo es una ruta que empieza y termina en el mismo lugar.
Un árbol es una red que carece de ciclos.

4.2.1 Estructura física CODASYL.

El bloque básico de construcción de una red CODASYL es un conjunto. Un conjunto está constituido por dos tipos de registro que entre si tienen una relación uno a muchos. En el caso de equipos de béisbol, un conjunto será:

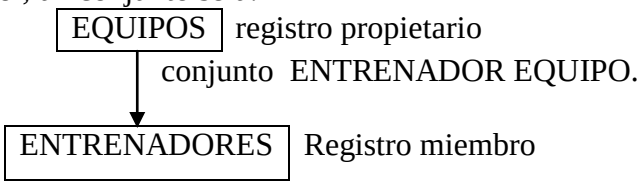


Fig. 4.16 Conjunto CODASYL. La flecha muestra la relación 1 ⟶ N.

Así como antes se habló de ocurrencias de registros y jerarquías, aquí puede hablarse de ocurrencias de conjuntos; las cuales consisten en una ocurrencia del tipo registro

propietario del conjunto y todas las del tipo de registro miembro que estén asociadas. En la figura 4.17 se observa una ocurrencia del conjunto ENTRENADOR EQUIPO; los Dodgers tienen tres entrenadores, Adams, Baker y Carter. La misma figura también muestra la forma en que se acostumbra almacenar las ocurrencias del conjunto CODASYL: con apuntadores. El apuntador principal (y el único necesario) denominado 'siguiente', aparece en forma de líneas sólidas en la figura 4.17. Con los apuntadores siguientes la ocurrencia del registro propietario (Dodgers) apunta a la primera ocurrencia de un registro miembro (Adams). Esa ocurrencia de registro miembro apunta a la siguiente, y así sucesivamente. Obsérvese que, excepto por el último apuntador siguiente que va de la última ocurrencia de registro miembro a la ocurrencia de registro propietario, la cadena de apuntadores siguiente se parece al arreglo de apuntadores hijo-gemelo en IMS. Se puede observar que la cadena de apuntadores siguientes forma un 'ciclo'. Solo recuérdese que en este caso se tiene un ciclo de ocurrencias de registro dentro de una ocurrencia de conjunto, mientras que la esencia del análisis de ciclo que se realizó antes en este escrito estaba encaminado hacia ciclos de tipos de registro.
En la figura 4.17 se ilustran dos tipos opcionales de apuntadores que en la práctica se emplean para algunos de los arreglos y procesamientos más complicados de redes. El apuntador al 'anterior' funciona de la misma forma que el operador siguiente, pero en dirección opuesta. El apuntador 'propietario' apunta de cada ocurrencia de registro miembro a la ocurrencia de registro propietario de ese conjunto.
Los conjuntos pueden combinarse de diferentes maneras para crear redes en una amplia gama de formas y tamaños. En la figura 4.18 se muestran dos conjuntos. Nótese que el registro ENTRENADORES es el tipo de registro miembro del conjunto ENTRENADOR EQUIPOS y al mismo tiempo es el tipo de registro propietario del conjunto ENTRENADOR EXP . En la figura 4.19 se muestran dos conjuntos que tienen el mismo registro propietario.
Véase la figura 4.20 en donde hay tres conjuntos: UNIVERSIDAD, EQUIPO, ORIGEN que muestran a todos los jugadores de la liga que provienen de una universidad específica, que juegan en un equipo y que nacieron en un estado determinado respectivamente. En esa estructura, los tres conjuntos tienen el mismo tipo de registro miembro JUGADORES. De esta forma la estructura no es un árbol sino una red más general.

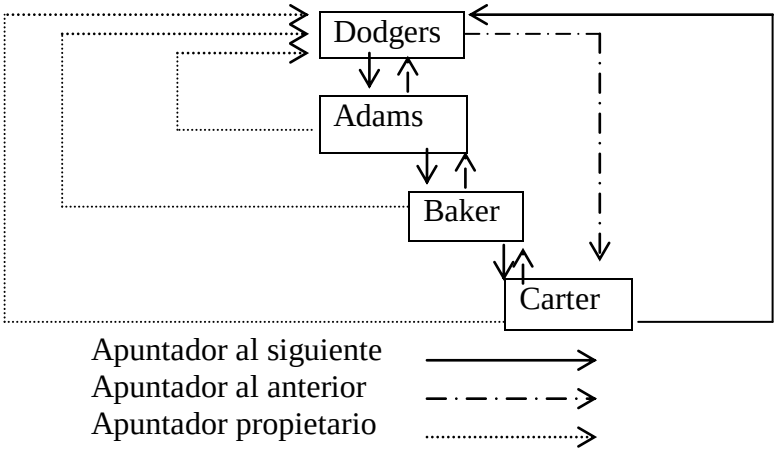


Fig. 4.17 Una ocurrencia del conjunto ENTRENADOR EQUIPO

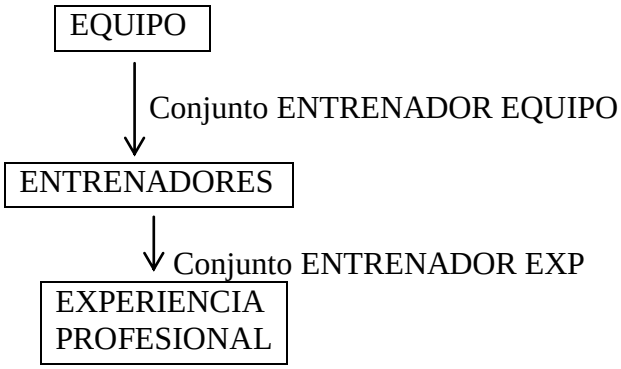


Figura 4.18 Un registro como tipo de registro miembro de un conjunto y de tipo de registro propietario de otro.

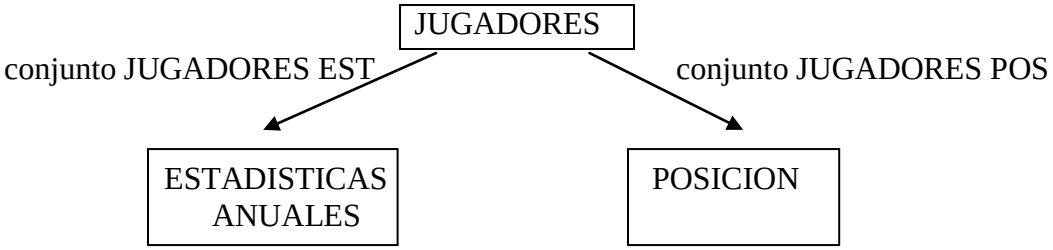


Fig. 4.19 Un registro como tipo de registro propietario de dos conjuntos diferentes

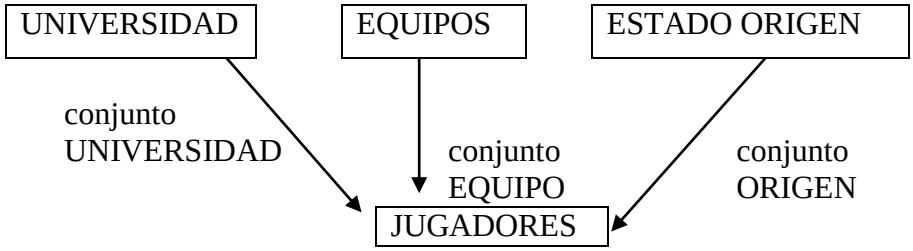


Fig. 4.20 Un registro como tipo registro miembro de tres conjuntos diferentes.

En la figura 4.21 se muestra el arreglo de apuntador para una ocurrencia de cada uno de esos tres conjuntos. Una ocurrencia de registro participa en (y debe tener una casilla apuntador al 'siguiente' para el efecto) una cadena de ocurrencias de registro por cada conjunto al que pertenece. En la estructura de la figura 4.20, el registro JUGADORES es el tipo de registro miembro de cada uno de los tres conjuntos, lo que significa que puede demostrarse que un jugador se graduó de una universidad, participa en un equipo y nació en un estado.

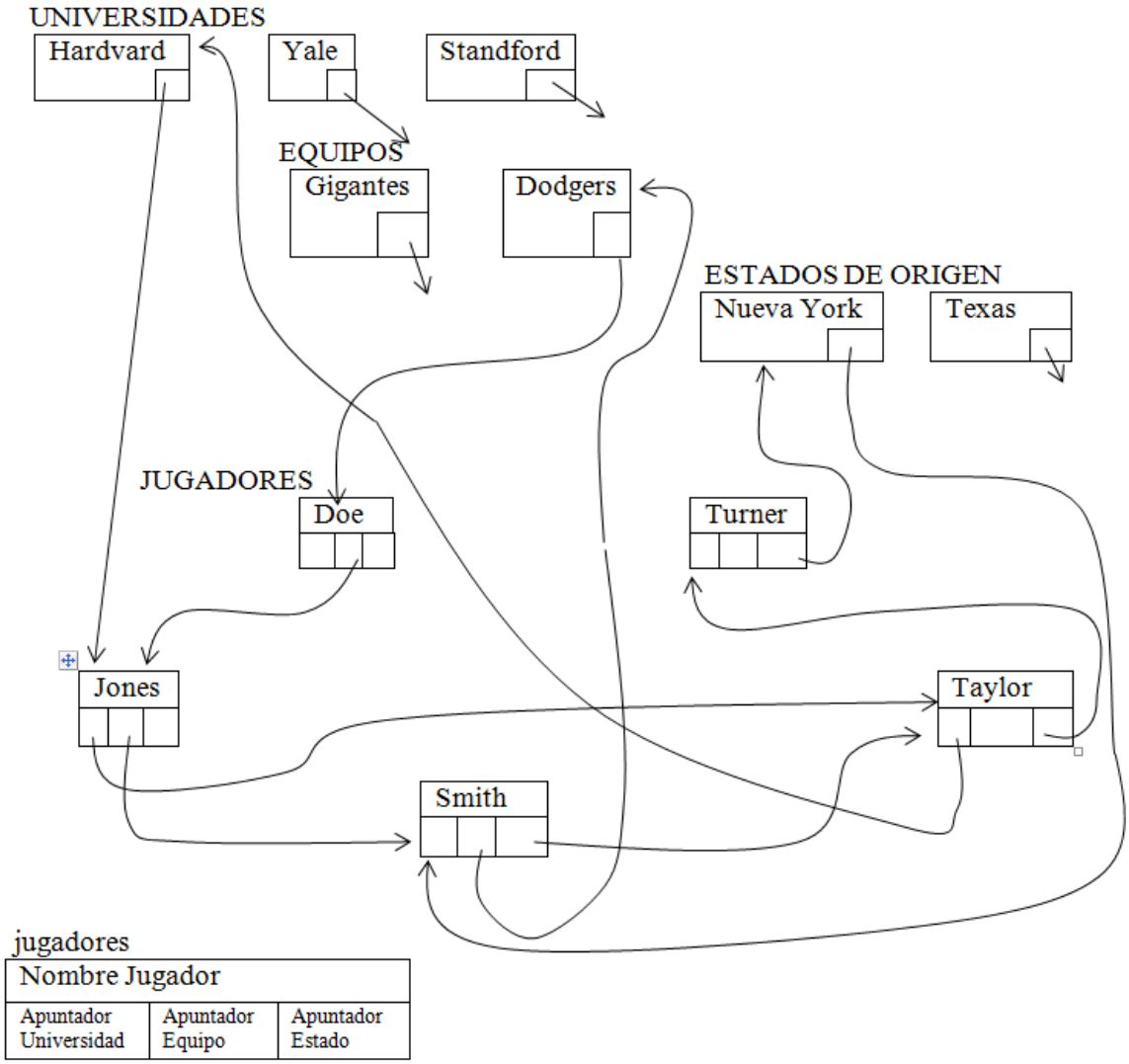


Fig. 4.21 Una ocurrencia de cada uno de los tres conjuntos de la figura 4.20.

El diagrama de la fig. 4.21 muestra que Jones y Taylor se graduaron en Hardvard; Doe, Jones y Smith juegan para los Dodgers y Smith, Taylor y Turner nacieron en Nueva York. Si se muestran todos los datos, se vería en qué cadena de universidad está Doe, en qué cadena de estado de origen está Jones, y así sucesivamente.

En la figura 4.22 se muestra otra estructura de red que no es un árbol. Puede conservarse a los jugadores de un equipo en el conjunto EQUIPO JUGADOR, los entrenadores de un equipo en el conjunto EQUIPO ENTRENADOR y a los entrenadores que corresponden a un jugador específico en el conjunto ENTRENADO POR. Pero si también se desea conservar lo contrario del conjunto ENTRENADO POR, es decir, los jugadores de los que un entrenador es responsable, entonces los dos juntos formarían una relación de muchos a muchos, lo que lleva al siguiente análisis de 'registro de unión'.

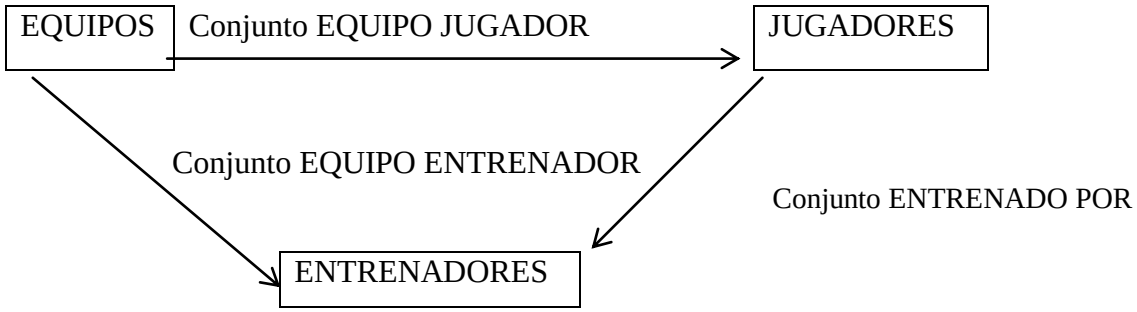


Fig. 4.22 Otra estructura de red.

4.2.2 Registros de unión

Hasta aquí no se ha dicho cosa alguna acerca de almacenar datos en relaciones muchos a muchos en redes CODAYL. En realidad (de manera semejante a la situación en IMS) las estructuras de red CODASYL no pueden representar directamente relaciones muchos a muchos. Podría decirse que dentro de un conjunto una ocurrencia de registro elemento puede asociarse con una sola ocurrencia de registro propietario. En la figura 4.16 un entrenador sólo debe estar asociado con un equipo; no puede haber datos históricos acerca de todos los equipos en que ha trabajado un entrenador durante su carrera. Las relaciones muchos a muchos se desarrollan en las redes CODASYL a través de la introducción de un nuevo tipo de registro, conocido como registro de unión. 'enlace', 'intersección' o 'conexión'. En vez de un enlace único muchos a muchos entre dos tipos de registros que se estén representando en un conjunto, habrá dos conjuntos; uno de los cuales tendrá una de esas dos formas de registro como tipo de registro propietario y uno de unión de nueva creación como tipo de registro miembro. El siguiente conjunto tendrá el otro de los dos tipos de registro como tipo de propietario y ese mismo registro unión como tipo de registro miembro. En realidad, el arreglo es muy semejante al de la estructura de relación lógica en IMS.

En la figura 4.23 se muestra la relación muchos a muchos entre EQUIPO y JUGADORES con la misma base histórica representada en forma de red CODASYL. Los conjuntos EQUIPO CONTROL y JUGADOR CONTROL juntos forman la relación muchos a muchos. Habrá una ocurrencia de registro CONTROL por cada caso en que un jugador específico haya participado con un determinado equipo o al contrario. En la figura 4.24 se presentan ocurrencias de los dos conjuntos descritos en la figura 4.23. Las ocurrencias de registro CONTROL y los apuntadores se analizarán a continuación; las primeras, trazadas a la mitad de la figura 4.24 tienen cada dos casillas de apuntador al siguiente, ya que el registro CONTROL es el tipo de registro elemento de los dos conjuntos involucrados. En cada ocurrencia, el apuntador izquierdo es para el conjunto EQUIPO CONTROL, el derecho es para el conjunto JUGADOR CONTROL. Los Dodgers han tenido a Doe y Jones como jugadores, en una época u otra, mientras que los Gigantes han tenido a Jones y Smith. Por el contrario, Doe sólo ha jugado con los Dodgers, Smith únicamente con los Gigantes y Jones en ambos.

La figura 4.24 también proporciona una buena ilustración de la necesidad del apuntador opcional 'propietario'. Recuérdese que todos los apuntadores que se muestran en la figura 4.24 son del tipo siguiente. Supóngase que se empieza en la ocurrencia Jones y se desea

determinar todos los equipos en que ha participado, y otra información acerca de esos equipos que está almacenada en la ocurrencia de registro EQUIPOS. El apuntador 8 desde Jones lleva al sistema a la ocurrencia B en el control; el sistema en ese punto podría, hablando de manera informal, conservar su lugar en B y seguir el apuntador 2 a los Dodgers, proporcionando uno de los equipos en que ha jugado Jones. Pero esto es sólo porque B corresponde al último de los jugadores en la cadena Dodgers y por tanto, apunta a éste. El caso más general y problemático se ilustra al continuar el apuntador 9 a la siguiente ocurrencia del registro de control en la cadena Jones, que es el C. A fin de llegar al registro de equipos asociado con el registro C (los Gigantes), el sistema debe seguir el apuntador 11 y después el 4 (y en general tal vez muchos más apuntadores en una red mucho más larga). Al hacer esto pasa a través de la ocurrencia CONTROL D que no tiene relación alguna con Jones. Por otra parte, si las ocurrencias CONTROL tuvieran apuntadores propietarios, entonces desde C sólo podría utilizar un apuntador propietario para encontrar en forma directa a los Gigantes que es el registro propietario de esa ocurrencia del conjunto EQUIPO CONTROL.
En la figura 4.25 se muestra toda la red de EQUIPOS y JUGADORES. Recuérdese que los datos de intersección se almacenarán en el registro unión.

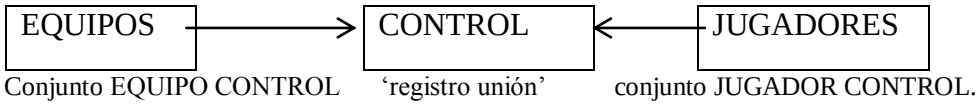


Fig. 4.23 Una relación muchos a muchos en CODASYL.

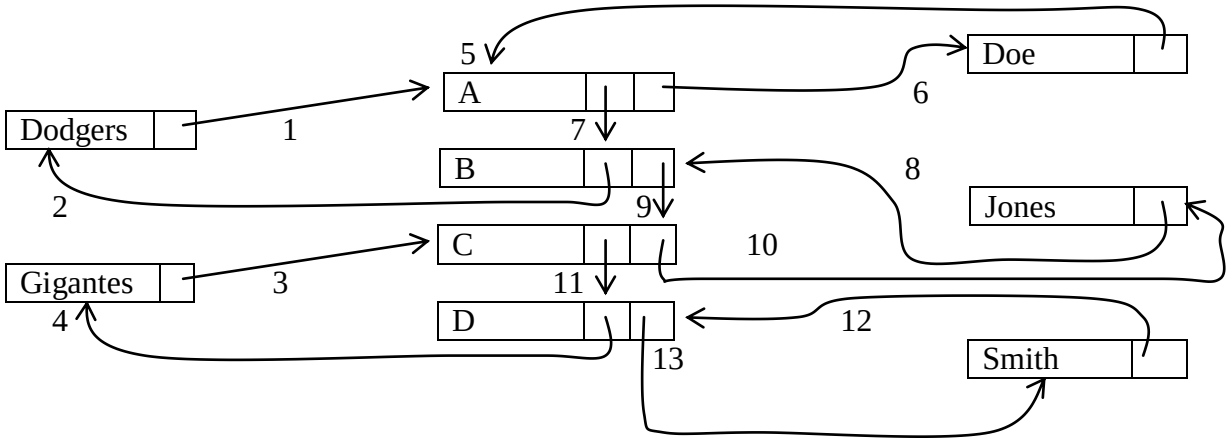


Fig. 4.24 Ocurrencias de los conjuntos mostrados en la figura 4.23.

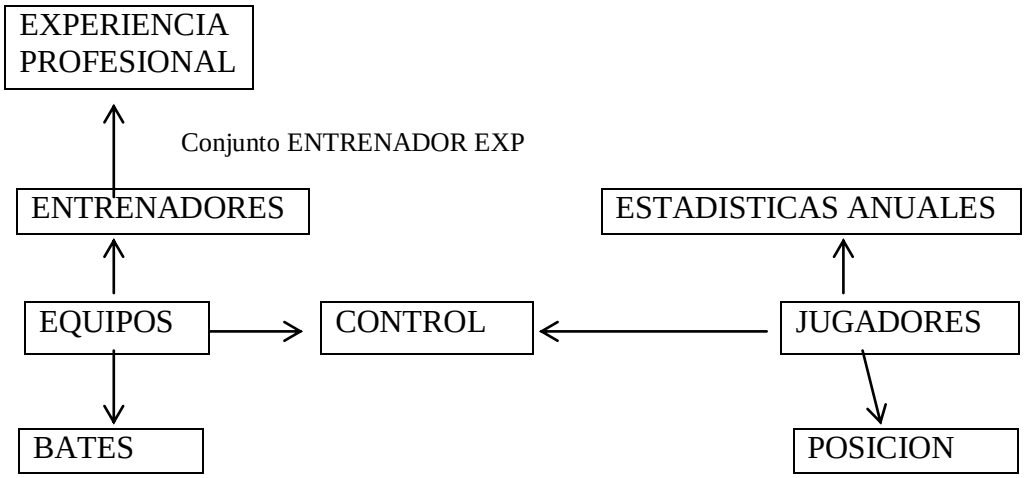
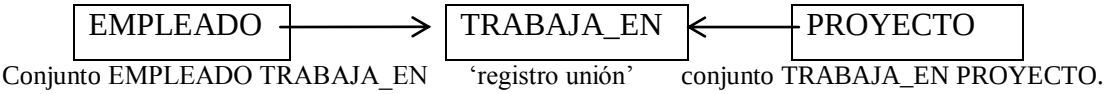
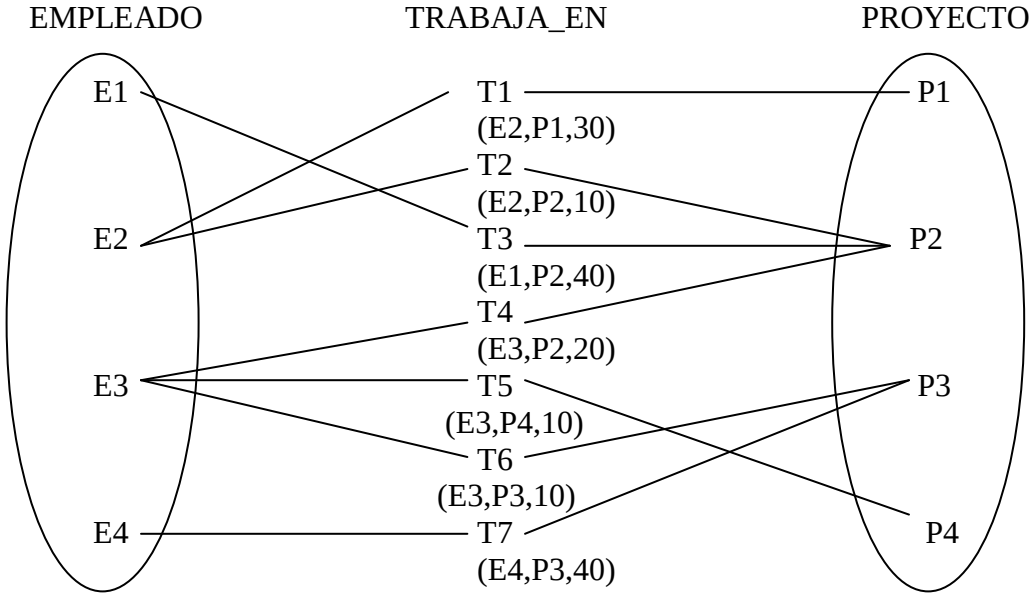


Fig. 4.25 Ejemplo de equipos y jugadores en forma de red.

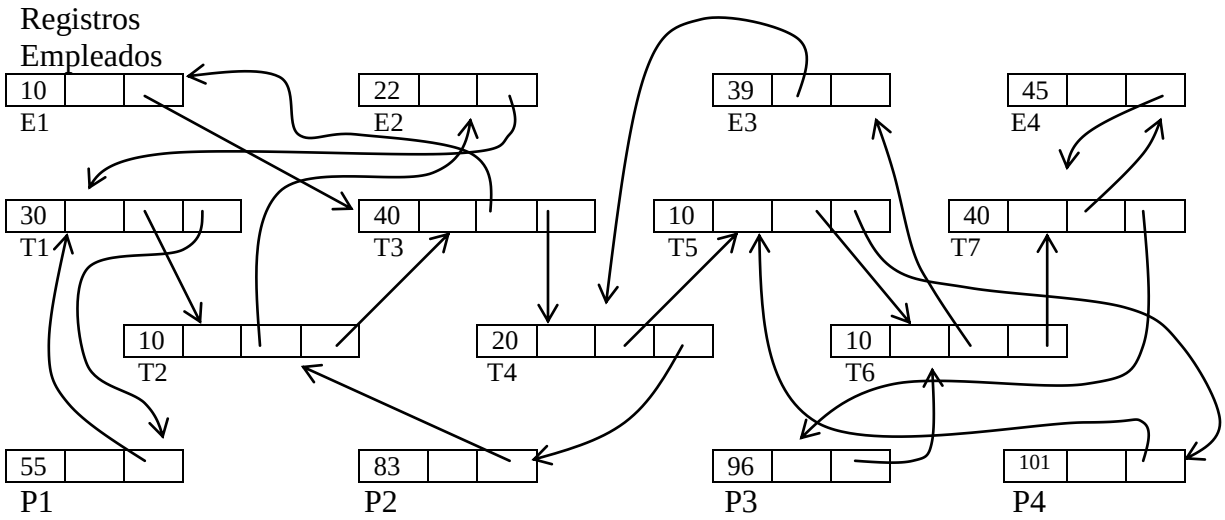
Otro ejemplo de representación N:M se localiza en el requerimiento COMPAÑIA. En este caso, el tipo de relación trabaja_en funciona como registro de control o de unión.



(A)

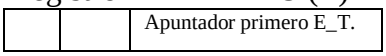


(B)

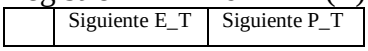


Notación:

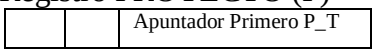
Registro EMPLEADO (E)



Registro TRABAJA-EN (T)



Registro PROYECTO (P)



(C)

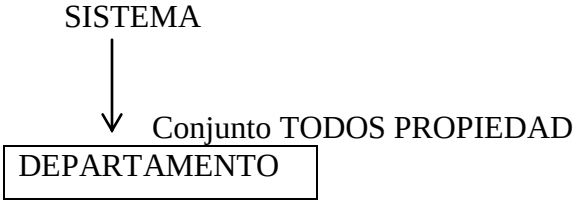
Fig. 4.26 (A) Representación de un vinculo o tipo de relación M:N empleando un tipo de enlace ficticio. (B) Representación de algunas ocurrencias de un vinculo M:N con 'ocurrencias enlazadas'. (C) Algunas ocurrencias de los tipos de conjunto E_T y P_T del tipo de registros de enlace TRABAJA_EN correspondientes a los ejemplares del vinculo M:N que se muestran en la figura 4.26(B).

Cada registro del tipo de registro TRABAJA_EN debe ser propiedad de un registro EMPELADO a través del conjunto E_T y de un registro PROYECTO a través del conjunto P_T, y sirve para relacionar estos dos registros propietarios. Esto se ilustra conceptualmente en la figura 4.26 (B).
La figura 4.26(C) muestra un ejemplo de ocurrencias individuales de registro y de conjunto en una representación de lista enlazada que corresponde al esquema de la figura 4.26(A). Cada registro del tipo de registro TRABAJA_EN tiene dos apuntadores SIGUIENTE: el rotulado SIGUIENTE E_T apunta al siguiente registro en un ejemplar del conjunto P_T, y el rotulado SIGUIENTE P_T apunta al siguiente registro en un ejemplar del conjunto P_T. Cada registro TRABAJA_EN relaciona sus dos registro propietario, y además contiene el número de horas que un empleado trabaja en cada proyecto. En la figura 4.26(B), que ilustra las mismas ocurrencias que la figura 4.26(C), se muestran los registros T individualmente, sin los apuntadores.
Si queremos encontrar todos los proyectos en los que trabaja un cierto empleado, comenzamos en el registro EMPLEADO y recorremos todos los registros

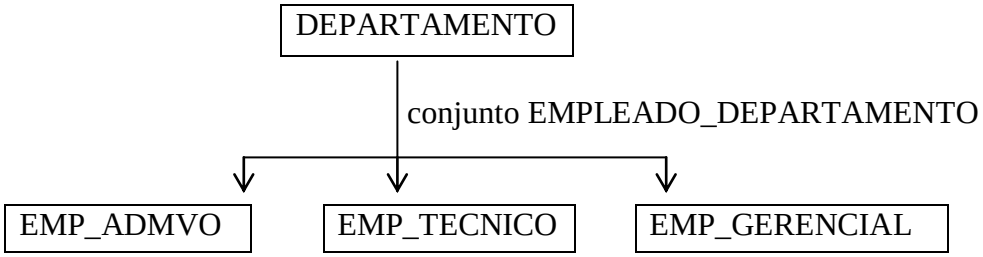
TRABAJA_EN propiedad de ese empleado, empleando los apuntadores PRIMERO E_T y SIGUENTE E_T. En cada registro TRABAJA_EN de la ocurrencia de conjunto encontramos su registro PROYECTO propietario siguiendo los apuntadores SIGUIENTE P_T hasta hallar un registro de tipo PROYECTO. Por ejemplo, en el caso del registro EMPLEADO E2, seguimos el apuntador PRIMERO E_T de E2 que conduce a T1, el apuntador SIGUIENTE E_T de T2 que conduce de vuelta a E2. Así, establecemos que T1 y T2 son los registros miembro de la ocurrencia de conjunto E_T propiedad de E2. Si seguimos el apuntador SIGUIENTE P_T de T1 llegamos a su propietario, P1 y seguimos el apuntador SIGUIENTE P_T de T2 (a través de T3 y T4) llegamos a su propietario P2. En forma similar, podemos encontrar todos los registros EMPLEADO relacionados con un PROYECTO en particular. EL DBMS realiza automáticamente todo este rastreo de apuntadores; el programador dispone de órdenes en DML para buscar directamente el propietario o el siguiente miembro.

Los conjuntos se clasifican en:

1. Sistema dueño (singular). Es un conjunto sin registro dueño; en lugar de eso tiene al sistema como dueño. El sistema es un dueño "virtual" con solo una ocurrencia de registro. Este tipo de conjunto sirve para:
   1.1 Proveer un punto de entrada a la base de datos vía el registro miembro.
   1.2 Puede servir para ordenar los registros de un tipo de registros dado mediante las especificaciones de ordenamiento del conjunto. Si un usuario especifica varios conjuntos propiedad del sistema para el mismo tipo de registro, puede tener acceso a sus registros en diferente orden.

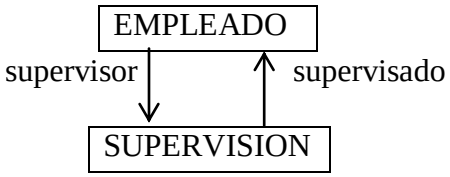


2. Conjuntos multimiembro. Los conjuntos multimiembro se utilizan en los casos en los que los registros miembro de un conjunto pueden pertenecer a más de un tipo de registro (no son soportados por casi ningún DBMS comercial).



3. Conjuntos recursivos. Un conjunto recursivo es un tipo de conjuntos en el que el mismo tipo de registro desempeña el papel tanto de propietario como de miembro. En el modelo CODASYL original estaban prohibidos los conjuntos recursivos debido a

la dificultad de procesarlos con el lenguaje de manipulación de datos (DML) de CODASYL



Retención en conjuntos
Se emplea para especificar si el registro de un tipo de registro puede existir por sí mismo o debe de relacionarse a un registro dueño como miembro. Existen tres opciones:

1. Opcional. Un registro miembro puede existir sin necesidad de ser miembro de alguna ocurrencia de conjunto. Puede ser conectado y desconectado de la ocurrencia de conjunto a voluntad. Ejemplo: un estudiante que no se relaciona a un departamento en particular (opcional).
2. Mandatorio. Un registro siempre estará conectado a un dueño especifico. Se le puede reconectar de una ocurrencia de conjunto a otra.
3. Fijo. Al igual que mandatorio, ningún registro miembro puede existir por sí solo. Por añadidura, una vez insertado en una ocurrencia de conjunto, queda fijo; no se le puede recolectar a otra ocurrencia de conjunto.

Las restricciones de inserción son:

1. Automática. El nuevo registro miembro se conecta automáticamente a una ocurrencia de conjunto apropiada cuando se inserta el registro.
2. Manual. El nuevo registro no se conecta a ninguna ocurrencia de conjunto. Si el programador lo desea, puede conectarse después explícitamente (manualmente) el registro a una ocurrencia de conjunto

		Opción de retención		
		OPCIONAL	MANDATORIO	FIJO
Opción de Inserción	MANUAL	El programa de aplicación se encarga de insertar el registro miembro en la ocurrencia de conjunto	No es muy útil	No es muy útil
	AUTOMATICO	El DBMS inserta automáticamente el nuevo registro. Se puede conectar, desconectar y reconectar.	Se puede reconectar un miembro a un propietario distinto.	No se puede reconectar un miembro a un propietario distinto.

Tabla 4.1 Opciones de inserción y retención de conjuntos.

El orden del conjunto puede realizarse como:

1. Ordenación por un campo especifico (por ejemplo, un ordenamiento alfabético)
2. Por default. Un nuevo registro miembro se inserta en una posición arbitraria determinada por el sistema.
3. Primero o último. Un nuevo registro miembro se convierte en el primero o último miembro de la ocurrencia de conjunto en el momento en que se inserta. Por tanto, esto equivale a tener los registros miembro de un ejemplar de conjunto almacenados en orden cronológico.
4. El siguiente o previo. El nuevo registro miembro se inserta después o antes del registro actual de la ocurrencia de conjunto.

Las opciones deseadas en cuanto a inserción, retención u ordenamiento se especifican cuando se declara el tipo de conjuntos en el lenguaje de definición de datos.

4.2.3 Mapeo del modelo ER al modelo red.

Teniéndose el modelo conceptual E-R, éste se puede transformar al modelo tipo red. En un esquema tipo red podemos representar el tipo de relación como un tipo de conjunto si es 1:N. El problema existe cuando no existe una analogía total entre 1:1 o N:M. Un método simple para representar un tipo de relación 1:1 es emplear un tipo de conjunto pero hacer que cada ejemplar de conjunto tenga como máximo un registro miembro. Esta restricción la deben imponer los programas que actualizan la base de datos, ya que el DBMS no lo hace. En el caso de tipos de relación M:N, la representación estándar es usar dos tipos de conjuntos y un tipo de registro de enlace. El modelo de red permite campos vectoriales y grupos repetitivos, con los que es posible representar directamente atributos compuestos y multivaluados, o incluso tipos de entidades débiles.

1. Entidades normales. Por cada tipo de entidad normal E del esquema ER, crear un tipo de registro R en el esquema de red. Todos los atributos simples (o compuestos) de E se incluyen como campos simples (o compuestos) de R. Todos los atributos multivaluados de E se incluyen como campos vectoriales o grupos repetitivos de R.
2. Entidades débiles. Para cada tipo de entidades débiles ED con el tipo de entidades identificador EI, o bien (a) creamos un tipo de registro D que represente a ED, haciendo a D el tipo de registro miembro de un tipo de conjuntos que relaciona D con el tipo de registros que representa a EI como el propietario, o bien (b) creamos un grupo repetitivo en el tipo de registro que representa a EI para que represente los atributos de ED. Si escogemos la primera alternativa, el campo clave del tipo de registros que representa a EI se puede repetir en D.
3. Vínculos uno a uno y muchos a muchos. Para cada tipo de relación 1:1 o 1:N binario no recursivo, I entre los tipos de entidades E1 y E2, creamos un tipo conjunto que relaciona los tipos de registro R1 y R2 que representan a E1 y E2, respectivamente. En el caso de un tipo de vinculo 1:1, elegimos arbitrariamente R1 o a R2 como propietario y al otro como miembro; sin embargo, es preferible escoger como miembro un tipo de registro que represente una participación total en el tipo de vinculo. Otra opción para transformar un tipo de vinculo binario 1:1 I entre E1 y E2 consiste en crear un solo tipo de registro R que combine E1, E2 e I, e incluya todos

sus atributos; esto resulta útil si ambas participaciones, de E1 y E2, en I son totales y ni E1 ni E2 participan en otros tipos de vínculos.
En el caso de un tipo de vinculo 1:N se escoge como propietario el tipo de registro R1 que representa al tipo de entidad E1 en el lado 1 del tipo de vínculo, y como miembro se elige el tipo de registro R2 que representa al tipo de entidad E2 en el lado N del tipo de vínculo. Cualesquier atributos del tipo de vínculo I se incluye como campos en el tipo de registro miembro R2
En general, podemos duplicar arbitrariamente uno o más atributos de un tipo de registros propietario de un tipo de conjunto –sea que represente un vínculo 1:1 o uno 1:N- en el tipo de registros miembro. Si el atributo duplicado es un atributo llave única del propietario, puede servir para declarar una restricción estructural sobre el tipo de conjuntos o para especificar la selección automática de propietario sobre la pertenencia al conjunto.

4. Vínculos binarios de muchos a muchos. Para cada tipo de relación o tipo de vinculo M:N binario I entre los tipos de entidad E1 y E2, creamos un tipo de registro enlace X y hacemos que sea el tipo de registros miembro de dos tipos de conjuntos, cuyos propietarios son los tipos de registros R1 y R2 que representan a E1 y E2. Cualesquier atributo de I se convierte en campos de X. Si lo desea, el diseñador puede duplicar los campos únicos (llaves) de un tipo de registro propietario como campos de X.
5. Vínculos recursivos. Por cada tipo de vínculos 1:1 o 1:N binario recursivo en el que el tipo de entidades E participe en ambos papeles, creamos un tipo de registros de enlace 'ficticio' V y dos tipos de conjuntos que relacionen V con el tipo de registros X que representa a E. Se hará obligatorio que uno de los tipos de conjuntos, o ambos, tengan ejemplares de conjuntos con un solo registro miembro: uno en el caso de un tipo de vínculo 1:N, y los dos si se trata de un tipo de vínculos 1:1.
6. Vínculos n-arios. Por cada tipo de vínculos n-ario I, con n>2, creamos un tipo de registros de enlace X y hacemos que sea el tipo de registros miembro de n tipos de conjuntos. El propietario de cada tipo de conjuntos es el tipo de registros que representa a uno de los tipos de entidades que participan en el tipo de vínculos I. Cualesquier atributo I se convertirán en campos de X. El diseñador puede, si lo desea, duplicar los campos únicos (llave) de los tipos de registros propietarios como campos de X.
   Por ejemplo, consideremos el tipo de vinculo SUMINISTRA en el modelo ER que se muestra en la figura 4.27(a). Este puede transformarse al tipo de registros SUMINISTRA y a los tres tipos de conjuntos que se muestran en la figura 4.27(b), donde decidimos no duplicar ningún campo de los propietarios.

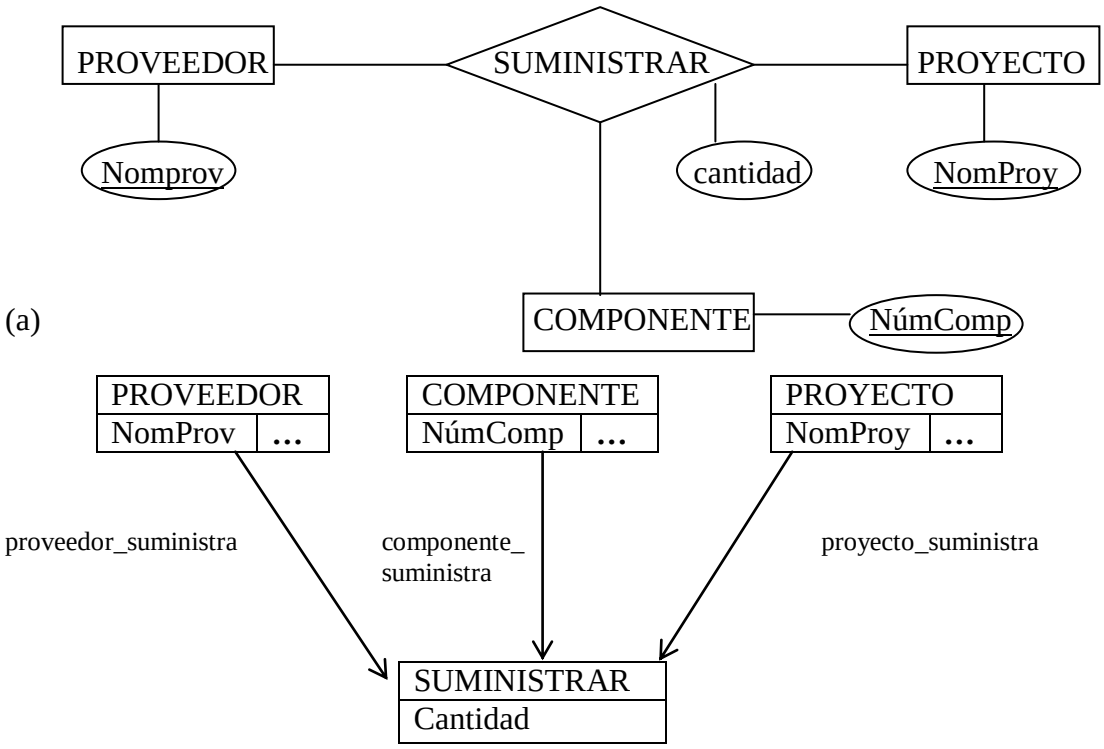


Fig. 4.27 Transformación del tipo de vínculos n-ario (n=3) SUMINISTRAR del modelo ER al modelo red. (a) El modelo ER. (b) El modelo red.

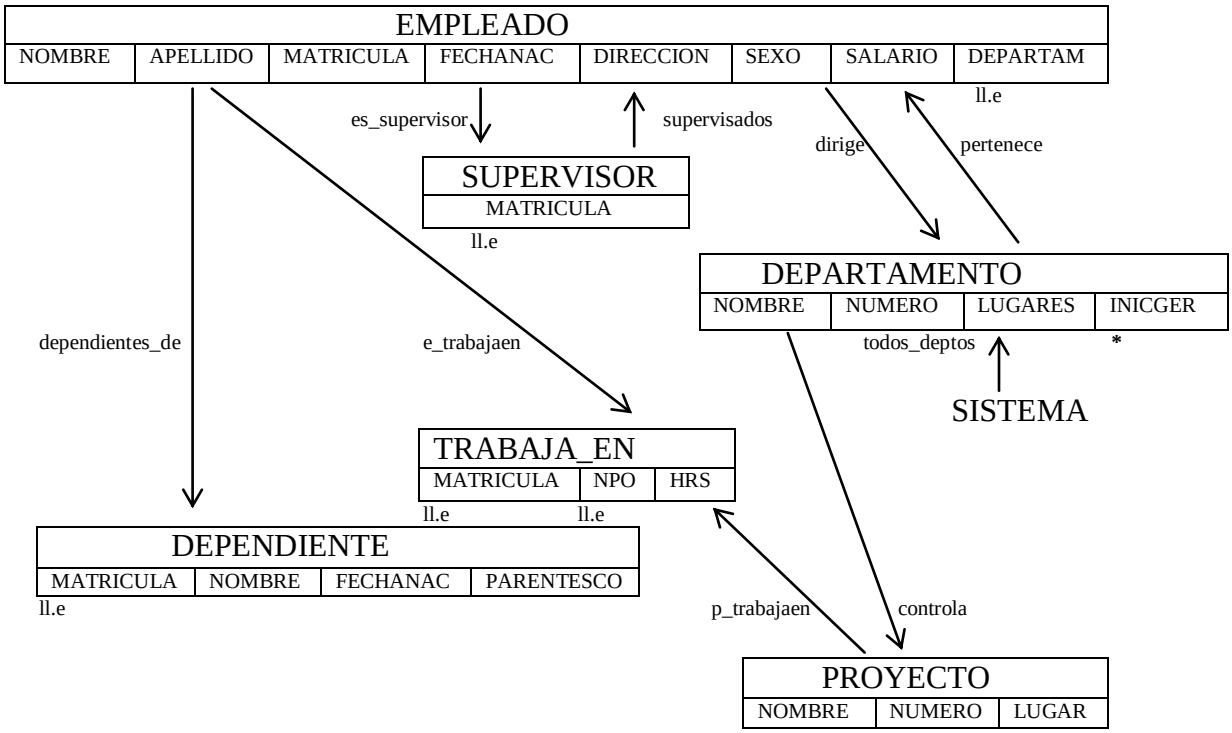


Fig. 4.28 Esquema de red para la base de datos COMPAÑIA.

En el caso del requerimiento COMPAÑIA, el algoritmo de mapeo del modelo ER al modelo RED funciona de la siguiente forma:

1. Creamos los tipos de registro EMPLEADO, DEPARTAMENTO y PROYECTO e incluimos todos los campos, como se muestra en la figura 4.28, con excepción de los que se marcan con ll.e (llave externa) o con * (atributo de vinculo). Observe que el campo LUGARES del tipo de registro DEPARTAMENTO es un campo vectorial porque representa un atributo multivaluado.
2. En la figura 4.28 elegimos la alternativa (a); creamos un tipo de registro DEPENDIENTE y lo hacemos del tipo de registros miembros del tipo de conjunto s DEPENDIENTES_DE, propiedad de EMPLEADO. Duplicamos la llave MATRICULA de empleado.
3. En nuestro ejemplo representamos el tipo de vinculo 1:1 DIRIGE de la sección III.3.3 con el tipo de conjunto DIRIGE, y escogemos DEPARTAMENTO como tipo de registro miembro debido a su participación total. El atributo Fechainic de DIRIGE se convierte en el campo INICGER del tipo de registro miembro DEPARTAMENTO. Los dos tipos de vínculos 1:N no recursivos de la sección III.3.3 PERTENECE_A y CONTROLA, se representan con los dos tipos de conjunto PERTENECE y CONTROLA , de la figura 4.28. En el caso del tipo de conjunto PERTENECE , optamos por repetir un campo de llave única, NOMBRE, del tipo de registros propietario DEPARTAMENTO en el tipo de registro miembro EMPLEADO, y lo llamamos DEPARTAM. Decidimos no repetir ningún campo llave para el tipo de conjuntos CONTROLA. En general, un campo único de un tipo de registros propietario podría repetirse en el tipo de registros miembro.
4. En la figura 4.28 creamos el tipo de registros de enlace TRABAJA_EN para representar el tipo de vinculo M:N TRABAJA_EN, e incluimos HORAS en él como campo. También creamos dos tipos de conjuntos E_TRABAJAEN y P_TRABAJAEN con TRABAJA_EN como tipo de registros miembro. Optamos por duplicar los campos llave única MATRICULA y NUMERO de los tipos de registros propietario EMPLEADO y PROYECTO en TRABAJA_EN, y los llamamos MATRICULA y NPO.
5. En la sección III.3.3 tenemos un tipo 1:N recursivo, SUPERVISION. Creamos el tipo de registros de enlace ficticio SUPERVISOR y los dos tipos de conjuntos ES_SUPERVISOR y SUPERVISADOS. Los programas de actualización de la base de datos obligan al tipo de conjunto ES_SUPERVISOR a tener sólo un tipo de registro miembro en sus ejemplares de conjunto. Podemos considerar que cada registro miembro ficticio SUPERVISOR del tipo de conjunto ES_SUPERVISOR representa a su registro EMPLEADO propietario en el papel de supervisor. El tipo de conjuntos SUPERVISADOS sirve para relacionar el registro SUPERVISOR "ficticio" con todos los registros EMPLEADO que representen a los empleados a los que supervisa directamente.

Comentarios finales sobre los modelos jerárquico y red.
Sobre el modelo jerárquico:

Uno de los sistemas de gestión de bases de datos jerárquico más populares fue el Information Management System (IMS) de IBM, introducido primeramente en 1968. Las ventajas del IMS y su modelo jerárquico son las siguientes:

- Estructura simple. La organización de una base de datos IMS era fácil de entender. La jerarquía de la base de datos se asemejaba al diagrama de organización de una empresa o un árbol familiar.
- Organización padre/hijo. Una base de datos IMS era excelente para representar relaciones padre/hijo, tales como 'A es una pieza de B' o 'A es propiedad de B'.
- Rendimiento. IMS almacenaba las relaciones padre/hijo como punteros físicos de un registro de datos a otro, de modo que el movimiento a través de la base de datos era rápido. Puesto que la estructura era sencilla. IMS podía colocar los registros padre/hijo cercanos unos a otros en el disco, minimizando la entrada/salida de disco.

IMS sigue siendo el DBMS más ampliamente instalado en los maxicomputadores IBM. Se utiliza en más del 25 por 100 de las instalaciones de maxicomputadores IBM.

Sobre el modelo red.
En 1971 la Conferencia sobre Lenguajes de Sistemas de Datos publicó un estándar oficial para bases de datos en red, que se hizo conocido como el modelo CODASYL, IBM nunca desarrolló un DBMS en red por si mismo, eligiendo en su lugar extender el IMS a lo largo de los años setenta, compañías de software independientes se apresuraron a adoptar el modelo en red, creando productos tales como el IDMS de Cullinet, el Total de Cincom y el DBMS Adabas que se hizo muy popular.
Para un programador, acceder a una base de datos en red era muy similar a acceder a una base de datos jerárquicos. Un programa de aplicación podía:

- Hallar un registro padre especifico mediante clave.
- Descender al primer hijo en un conjunto particular.
- Moverse lateralmente de un hijo al siguiente dentro del conjunto.
- Ascender desde un hijo a su padre en otro conjunto.

Una vez más el programador tenía que recorrer la base de datos registro a registro, especificando esta vez qué relación recorre además de indicar la dirección.
Las bases de datos en red tenían varias ventajas:

- Flexibilidad. Las múltiples relaciones padre/hijo permitían a una base de datos en red representar datos que no tuvieran una estructura jerárquica sencilla.
- Normalización. El estándar CODASYL reforzó la popularidad del modelo de red, y los vendedores de minicomputadores tales como Digital Equipment Corporation y Data General implementaron bases de datos en red.
- Rendimiento. A pesar de su superior complejidad, las bases de datos en red reforzaron el rendimiento aproximándolo al de las bases de datos jerárquicos. Los conjuntos se representaron mediante punteros a registros de datos físicos, y en algunos sistemas, el administrador de la base de datos podía especificar la agrupación de datos basadas en una relación de conjuntos.

Las bases de datos tenían sus desventajas también. Igual que las bases de datos jerárquicos, resultaban muy rígidas. Las relaciones de conjunto y la estructura de los

registros tenían que ser especificadas de antemano. Modificar la estructura de la base de datos requería típicamente la reconstrucción de la base de datos completa.
Tanto las bases de datos jerárquicas como las de en red eran herramientas para programadores. Un programador tenía que escribir un programa que recorriera su camino a través de la base de datos. La anotación de las peticiones para informes a la medida duraba con frecuencia semanas o meses, y para el momento en que el programa estaba escrito la información que se entregaba con frecuencia ya no merecía la pena.

4.3 MODELO RELACIONAL.

Las desventajas de los modelos jerárquicos y en red en conjunto a una intensa investigación en bases de datos. En 1970, el doctor Edgar F. Codd de IBM publicó un escrito titulado "A Relational Model of Data for Large Shared Data Banks" (Codd, Junio, 1970). Esta publicación señaló el inicio del campo de la base de datos relacional. Durante los años siguientes el enfoque relacional a la base de datos ha recibido mucha publicidad; sin embargo, sólo ha sido durante los últimos años que han aparecido en el mercado sistemas de manejo de base de datos relacional que son viables desde el punto de vista comercial, y en este momento existen en el mercado muchos paquetes de DBMS relacionales. El modelo relacional se está estableciendo firmemente en el mundo de las aplicaciones de bases de datos.

4.3.1 CONCEPTOS DEL MODELO RELACIONAL.

El modelo relacional representa la base de datos como una colección de relaciones. En términos informales, cada relación semeja una tabla o, hasta cierto punto, un archivo simple. Por ejemplo, se considera que la base de datos de archivos que se muestra en la figura 4.29 se llama ESTUDIANTE porque cada fila representa hechos acerca de una entidad estudiante en particular. Los nombres de la columna - Nombre, NúmEstudiante, Grado, Carrera- en particular- especifican cómo interpretar los valores de datos de cada fila, con base en la columna en la que se encuentra cada valor. Todos los valores de una columna tienen el mismo tipo de datos.

ESTUDIANTE	Nombre	NúmEstudiante	Grado	Carrera
	Suárez	17	1	ICO
	Borja	8	2	ICO

Fig. 4.29 Ejemplo de base de datos.

En la terminología del modelo relacional, una fila se denomina *tupla*, una cabecera de columna es un *atributo* y la tabla es una *relación*. El tipo de datos que describen los tipos de valores que pueden aparecer en cada columna se llama *dominio*. A continuación definiremos estos términos – *dominio, tupla, atributo* y *relación* – con mayor precisión:
Un dominio D es un conjunto de valores atómicos. Por atómico queremos decir que cada valor del dominio es indivisible en lo tocante al modelo relacional. Un método común de especificación de los dominios consiste en especificar un tipo de datos al cual pertenecen los valores que constituyen el dominio. También resulta útil especificar un nombre para el dominio que ayude a interpretar sus valores. Un ejemplo de dominio es:

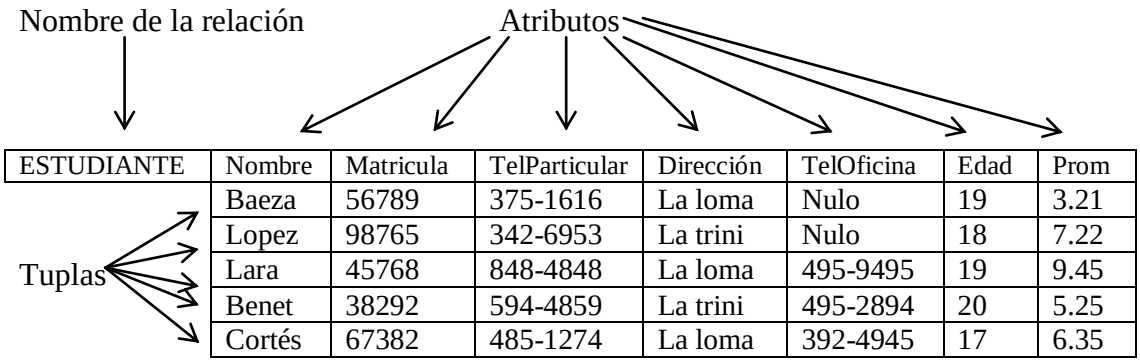
- Números_telefónicos_de_EUA. Significado: El conjunto de números telefónicos de 10 dígitos válidos en los Estados Unidos. Formato o tipo de datos: Cadena de

caracteres de la forma (ddd)ddd-dddd, donde cada d es un dígito numérico (decimal) y los primeros tres dígitos forman un código de área telefónica válido.

Así pues, un dominio debe tener un nombre, un tipo de datos y un formato. También puede incluirse información adicional para interpretar los valores de un dominio; por ejemplo, un dominio numérico como Peso_de_personas deberá especificar las unidades de medición: libras o kilogramos. A continuación definiremos el concepto de esquema de relación, que describe la estructura de una relación.

Un esquema de relación R, denotado por R (A_1, A_2, A_3, . . ., A_n), se compone de un nombre de relación, R, y un conjunto ordenado de atributos A_1, A_2, A_3, . . ., A_n. Cada atributo A_i es el nombre de un papel desempeñado por algún dominio D en el esquema R. Se dice que D es el dominio de A_i y se denota como dom (A_i). Un esquema de relación sirve para describir una relación; R es el nombre de la relación. El grado de la relación es el número de atributos, n, de su esquema de relación.
El siguiente es un esquema de relación para una relación de grado 7, que describe estudiantes universitarios:
ESTUDIANTE (Nombre, Matricula, TelParticular, Dirección, TelOficina, Edad, Prom)

En este esquema de relación, ESTUDIANTE es el nombre de la relación, la cual tiene siete atributos. Podemos especificar los siguientes dominios para algunos de los atributos de la relación ESTUDIANTE: dom (Nombre)= nombres; dom (Matricula)=matricula del estudiante; dom (TelParticular)= Números telefónicos locales; dom(TelOficina)=Número telefónico de la oficina; dom (Prom)=Promedio de notas.
Una relación o ejemplar de relación **r** del esquema de relación R (A_1, A_2, A_3, . . ., A_n) denotado también por r(R), es un conjunto de n- tuplas r= {t_1, t_2, t_3, . . ., t_m}. Cada tupla t es una lista ordenada de n valores t= < v_1, v_2, v_3, . . ., v_n>, donde cada valor v_i, 1 <=i<=n, es un elemento de dom(A_i) o bien un valor nulo especial. También se acostumbra usar los términos intensión de una relación para el esquema R y extensión (o estado) de una relación para un ejemplar de relación r(R).
La figura 4.30 muestra un ejemplo de una relación ESTUDANTE, que corresponde al esquema ESTUDIANTE que acabamos de especificar. Cada tupla de la relación representa una entidad estudiante en particular. Presentamos la relación en forma de tabla, en la que cada tupla aparece como una fila y cada atributo corresponde a una cabecera de columna que indica un papel o interpretación de los valores de esa columna. Los valores nulos representan atributos cuyos valores se desconocen o no existen para algunas tuplas ESTUDIANTE individuales.



ESTUDIANTE	Nombre	Matricula	TelParticular	Dirección	TelOficina	Edad	Prom
	Baeza	56789	375-1616	La loma	Nulo	19	3.21
	Lopez	98765	342-6953	La trini	Nulo	18	7.22
	Lara	45768	848-4848	La loma	495-9495	19	9.45
	Benet	38292	594-4859	La trini	495-2894	20	5.25
	Cortés	67382	485-1274	La loma	392-4945	17	6.35

Fig. 4.30 Relación Estudiante.

La definición anterior de relación puede expresarse también como sigue: una relación r(R) es un subconjunto del producto cartesiano de los dominios que definen a R:

$$r(R) \subseteq (dom(A_i)Xdom(A_2)X...Xdom(A_n))$$

El producto cartesiano especifica todas las combinaciones posibles de valores de los dominios implicados. Así pues, si denominamos el número de valores o cardinalidad de un dominio D con |D|, y suponemos que todos los dominios son finitos, el número total de tuplas del producto cartesiano es

$$|dom(A_1)|* |dom(A_2)|*\ldots *|dom(A_n)|$$

De todas estas posibles combinaciones, un ejemplar de relación en un momento dado –el estado actual de la relación- refleja sólo las tuplas válidas que representan un estado específico del mundo real. En general, a medida que cambia el estado del mundo real, cambia la relación, transformándose en otro estado de la relación. Sin embargo, el esquema R es relativamente estático, y no cambia con frecuencia; lo hace, por ejemplo, cuando se añade un atributo para representar información nueva que no estaba representada originalmente en la relación.
Es posible que varios atributos tengan el mismo dominio. Los atributos indican diferentes papeles, o interpretaciones, del dominio. En la relación ESTUDIANTE, por ejemplo, el mismo dominio Números_Telefónicos_locales desempeña el papel de TeleParticular, refiriéndose al "teléfono particular de un estudiante", y el de TelOficina, refiriéndose al "teléfono de la oficina del estudiante".

4.3.1 RESTRICCIONES DEL MODELO RELACIONAL

En esta sección analizaremos los diversos tipos de restricciones que se pueden especificar en un esquema de bases de datos relacional. Entre estas restricciones se cuentan las de dominio, de clave, de integridad de entidades y de integridad referencial.

Restricciones de dominio
Las restricciones de dominio especifican que el valor de cada atributo A debe ser un valor atómico del dominio dom(A) para ese atributo.

Restricciones de llave.

Una relación se define como un conjunto de tuplas. Por definición, todos los elementos de un conjunto son distintos; por tanto, todas las tuplas de una relación deben ser distintas. Esto significa que no puede haber dos tuplas que tengan la misma combinación de valores para todos sus atributos. Por lo regular existen otros subconjuntos de atributos de un esquema de relación R con la propiedad de que no debe haber dos tuplas en un ejemplar de relación r (R) con la misma combinación de valores para estos atributos. Suponga que denotamos un subconjunto así de atributos con SC; entonces, para cualesquiera dos tuplas distintas t_1 y t_2 en un ejemplar de relación r(R), tenemos la siguiente restricción:

$$t_1[SC] <> t_2[SC]$$

Todo conjunto de atributos SC de este tipo es una superclave o superllave del esquema de relación R. Toda relación tiene por lo menos una superclave; el conjunto de todos los atributos. Sin embargo, una superllave puede tener atributos redundantes, así que un concepto más útil es el de clave (o el de llave), que carece de redundancia. Una clave K de un esquema de relación R es una superclave de R con la propiedad adicional de que la eliminación de cualquier atributo A de K deja un conjunto de atributos K' que no es una superclave de R. Por tanto, una llave es una superclave mínima; una superclave a la cual no podemos quitarle atributos sin que deje de cumplirse la restricción de unicidad.
El valor de un atributo clave puede servir para identificar de manera única una tupla de relación. Observe que el hecho de que un conjunto de atributos constituya una clave es una propiedad del esquema de la relación; es una restricción que debe cumplirse en todos los ejemplares de relaciones del esquema. La clave se determina a partir del significado de los atributos en el esquema de la relación; por ende, la propiedad no varía con el tiempo; debe seguir siendo válida aunque insertemos tuplas nuevas en la relación.
En general, un esquema de relación puede tener más de una clave. En tal caso, cada una de ellas se denomina 'clave candidato'. Es común designar a una de las claves candidatas como 'llave primaria' de la relación. Esta es la clave candidata cuyos valores sirven para identificar las tuplas de la relación. Cabe señalar que, cuando un esquema tiene varias claves candidatas, la elección de una para fungir como llave primaria es arbitraria; sin embargo, casi siempre es mejor escoger una llave primaria con un solo atributo o un número reducido de atributos.

Esquemas y ejemplares de bases de datos relacionales
Hasta ahora hemos visto relaciones y esquemas de relaciones individuales. Pero, de hecho, una base de datos relacional suele contener muchas relaciones y en éstas las tuplas están relacionadas de diversas maneras. En esta sección se definirá una base de datos relacional y un esquema de bases de datos relacional. Un esquema de bases de datos relacional S, o meta base de datos) es un conjunto de esquemas relacionales $S=\{R_1, R_2, R_3, \ldots, R_m\}$ y un conjunto de restricciones de integridad RI. Un ejemplar de bases de datos BD de S es un conjunto de ejemplares de relaciones $BD= \{r(R_1), r(R_2), r(R_3), \ldots, r(R_m)\}$ tal que $r(R_i)$ es un ejemplar de R_i y tal que las relaciones $r(R_i)$ satisfacen las restricciones de integridad especificadas en RI. La figura 4.31 muestra un esquema de base de datos relacional que llamamos COMPAÑIA y la figura 4.32 muestra un ejemplar de bases de datos relacional que corresponde a COMPAÑIA. Se empleará este esquema para mostrar el funcionamiento de álgebra relacional, cálculo relacional y de SQL en las secciones siguientes.

En algunas de las primeras versiones del modelo relacional se hizo la suposición de que el mismo concepto del mundo real, al representarse como un atributo, tendría nombres idénticos en todas las relaciones. Esto crea problemas cuando se usa el mismo concepto del mundo real con diferentes papeles (significados) en la misma relación. Por ejemplo, el concepto del número de matrícula aparece dos veces en la relación EMPLEADO de la figura 4.31; una vez en el papel de matrícula del empleado, y otra en el papel de número de matrícula de supervisor. A fin de evitar problemas, les dimos nombres distintos.
Por lo que la restricción de dominio indica que se deben tener valores atómicos en cada atributo en cada una de las tuplas de un esquema de relación y la restricción de llave primaria indica que no deben existir valores repetidos en la llave primaria de un ejemplar de relación
Las restricciones de integridad se especifican en el esquema de una base de datos y se deben cumplir en todos los ejemplares de ese esquema. Además de las restricciones de dominio y de clave, hay otros dos tipos de restricciones en el modelo relacional: integridad de entidades e integridad referencial.

Integridad de entidades, integridad referencial y claves externas

- Restricción de integridad de entidades. Establece que ningún valor de clave primaria puede ser nulo. Esto es porque el valor de la clave primaria sirve para identificar las tuplas individuales en una relación; el que las claves primarias tengan valores nulos implica que no podemos identificar algunas tuplas.
- Restricción de integridad referencial. Las restricciones de clave y de integridad de entidades se especifican sobre relaciones individuales. La restricción de integridad referencial se especifica entre dos relaciones y sirve para mantener la consistencia entre tuplas de las dos relaciones. En términos informales, la restricción de integridad referencial establece que una tupla en una relación que haga referencia a otra relación deberá referirse a una tupla existente en esa relación. Por ejemplo, en la figura 4.4 el atributo DNO de EMPLEADO da el número del departamento para el cual trabaja cada empleado; por tanto, su valor en cada tupla de EMPLEADO deberá coincidir con el valor del número de departamento en alguna tupla de la relación DEPARTAMENTO.

  Para dar una definición más formal de integridad referencial primero definimos el concepto de llave externa. Un conjunto de atributos CE en el esquema de relación R_i es una llave externa de R_i si satisface las siguientes reglas:

  - Los atributos de CE tienen el mismo dominio que los atributos de la clave primaria CP de otro esquema de relación R_j; se dice que los atributos CE hacen referencia o se refieren a la relación R_j.
  - Un valor de CE en una tupla t_i de R_i ocurre como valor de CP en alguna tupla t_j de R_j o bien es nulo. En el primer caso tenemos $t_1[CE] = t_2[CP]$, y decimos que la tupla t_i hace referencia o se refiere a la tupla t_j.

  Podemos representar diagramáticamente las restricciones de integridad referencial trazando un arco dirigido de cada clave externa a la relación a la cual hace referencia. Para mayor claridad, la punta de la flecha puede apuntar a la clave primaria de la relación referida. La figura 4.33 muestra el esquema de la figura 4.31 con las restricciones de integridad referencial representadas de esta manera.

La llave foránea tiene la función de guardar la semántica del tipo de relación del modelo ER y apoya a la restricción de dominio.

EMPLEADO

NOMBRE	PATERNO	MATERNO	MATRICULA	FNAC	DIRECCION	SEXO	SALARIO	SUPERV	DNO

DEPARTAMENTO

DNOMBRE	DNUMERO	MATRIGERENTE	FECHAING

LOCAL_DEPTO

DNUMERO	DLOCALIZACIÓN

PROYECTO

NOMBREPROY	PNUMERO	PLOCALIZACIÓN	DNO

TRABAJA_EN

MATRICULAEMP	PNO	HRS

DEPENDIENTE

EMPLEADOMAT	NOMBRE_DEPENDIENTE	SEXO	FECHANAC	RELACION

Fig. 4.31 Esquema de la base de datos relacional COMPAÑIA; las claves primarias están subrayadas.

DEPARTAMENTO

Dnombre	Dnumero	matrigerente	Fechaing
Prod. Alimentos	1	00007	5/19/71
Administración	4	54321	1/1/85
Investigación	5	45555	5/22/78

EMPLEADO

Nombre	Paterno	materno	Matricula	nacimiento	Dirección	sexo	salario	supervisor	dno
James	Bond	bond	00007	10/10/27	sn diego	m	9000		1
Francisco	López	López	45555	8/12/45	sn diego	m	4000	00007	5
Josefa	España	english	53453	7/31/62	sn diego	f	2500	45555	5
Jennifer	Ortega	lozano	54321	6/20/31	Tocuila	f	4300	00007	4
Juan	Perez	arellano	56789	9/1/55	sn diego	m	3000	45555	5
Ramses	Sanchez	mendez	84444	9/15/52	sn miguel	m	3800	45555	5
Alicia	Zelaya	juarez	87777	7/19/58	la trinidad	f	2500	54321	4
Aldama	Hidalgo	veracruz	87987	3/29/59	sn diego	m	2500	54321	4

LOCAL_DEPTO

Dnumero	dlocalizacion
1	sn diego
4	Papalotla
5	Atenco
5	sn diego
5	sn Felipe

TRABAJAEN

Matriculaemp	pno	Hrs
00007	20	0
45555	10	10
45555	2	10
45555	20	10
45555	3	10
53453	1	20
53453	2	20
53453	3	10
54321	20	15
54321	30	20
56789	1	32.5
56789	2	7.5
56789	3	10
84444	3	40
87777	10	10
87777	30	30
87987	10	35
87987	30	5

PROYECTO

Nombreproy	pnumero	plocalizacion	Dno
Producto1	1	Atenco	5
Automatizcion	10	Pagalotla	4
Producto2	2	sn Felipe	5
Organización	20	sn diego	1
Producto3	3	sn diego	5
Publicidad	30	Pagalotla	4

DEPENDIENTE

Empleadomat	nombredependiente	Sexo	Fechanac	Relación
45555	Alicia	F	5/4/76	Hija
45555	Juana	F	3/5/48	Esposa
45555	Teodoro	M	10/25/73	Hijo
54321	Abne	M	2/29/32	Esposo
56789	Alicia	F	12/31/78	Hija
56789	Elizabeth	F	5/5/57	Esposa
56789	Juan	M	1/1/63	Hijo
56789	Miguel	M	1/1/78	Hijo

Fig. 4.32 Ejemplar (estado) de base de datos relacional del esquema COMPANIA.

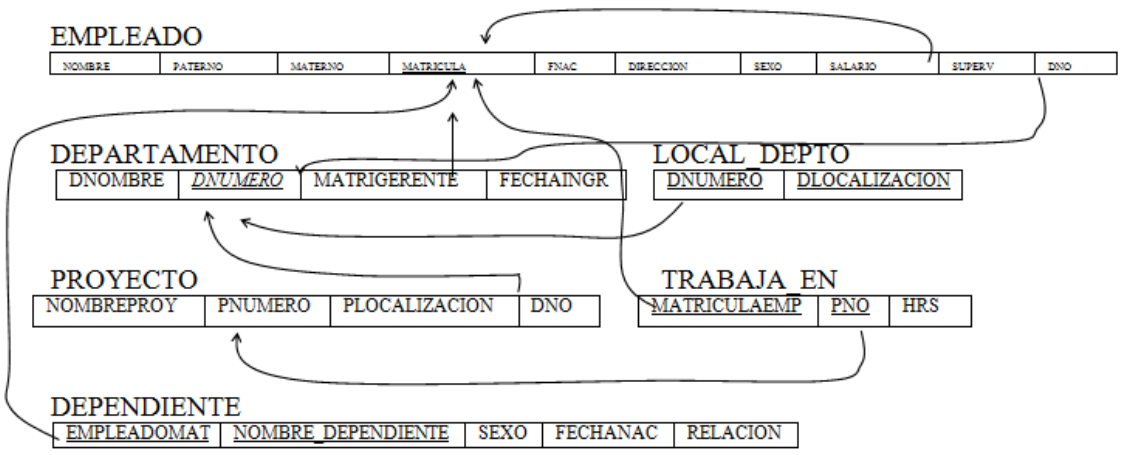


Fig. 4.33 Restricciones de integridad referencial representadas en el esquema de la base de datos relacional COMPAÑIA.

4.3.2 OPERACIONES DE ACTUALIZACION CON RELACIONES.

Son tres las operaciones de actualización básicas que se efectúan con relaciones; insertar, eliminar y modificar. Insertar sirve para insertar una o más tuplas nuevas en una relación; eliminar sirve para eliminar tuplas, y modificar sirve para alterar los valores de algunos atributos. Siempre que se apliquen operaciones de actualización, se debe cuidar de no violar las restricciones de integridad especificadas en el esquema de la base de datos relacional.

La operación insertar.

La operación insertar proporciona una lista de valores de atributos para una nueva tupla t que se ha de insertar en una relación R. La inserción puede violar cualquiera de los cuatro tipos de restricciones que vimos en la sección anterior. Las restricciones de dominio se pueden violar si se proporciona un valor de atributo que no aparezca en el dominio correspondiente. Las restricciones de clave pueden violarse si un valor clave de la nueva tupla t ya existe en otra tupla de la relación r(R). La integridad de entidades puede violarse si la clave primaria de la nueva tupla (t) es nula. Y la integridad referencial puede violarse si el valor de cualquier clave externa de t hace referencia a una tupla que no existe en la relación referida.

Si una inserción viola una o más restricciones, disponemos de dos opciones. La primera es rechazar la inserción, en cuyo caso sería útil que el DBMS explicara al usuario por qué fue rechazada. La segunda es intentar corregir la razón por la que se rechazó la inserción.

La operación eliminar.

La operación eliminar sólo puede violar la integridad referencial, si las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se ha de eliminar (t_1[CE] = t_2[CP]).

Si una operación de eliminación provoca una violación disponemos de tres opciones. La primera es rechazar la eliminación, las otras dos dependen de:

- Si la clave externa no es parte de la llave primaria, se puede colocar el valor nulo en el campo correspondiente a la llave foránea (CE).
- Si la clave externa es parte de la llave primaria, entonces de debe de eliminar la tupla t_1[CE].

La operación modificar.

La operación modificar sirve para cambiar los valores de uno o más atributos en una tupla (o tuplas) de una relación R. Es necesario especificar una condición para los atributos de R a fin de seleccionar la tupla (o tuplas) que se modificarán.

La modificación de un atributo que no es llave primaria ni llave externa; casi nunca causa problemas; basta con que el DBMS constate que el nuevo valor sea del tipo de datos correcto y esté en el dominio. Modificar un valor de llave primaria es similar a eliminar una tupla e insertar otra en su lugar, porque usamos la llave primaria para identificar las tuplas. Por tanto, los problemas que ya vimos al hablar de inserciones y eliminaciones se pueden presentar en este caso también. Si se modifica un atributo de llave externa, el DBMS debe asegurarse de que el nuevo valor haga referencia a una tupla existente en la relación de referencia.

4.3.3 TRANSFORMACION DEL MODELO E-R AL MODELO RELACIONAL.

En esta sección se explicará la forma de transformar a un esquema de base de datos relacional a partir de un esquema conceptual creado empleando el modelo ER. Muchas herramientas CASE (computer-aided software engineering; ingeniería de software asistida por computador) se basan en el modelo ER y en sus variaciones. Los diseñadores de bases de datos utilizan interactivamente estas herramientas computarizadas para crear un esquema ER para su aplicación de bases de datos. Muchas herramientas se valen de diagramas ER o variaciones de ellos para crear el esquema gráficamente, y luego lo convierten automáticamente en un esquema de base de datos relacional expresado en el DDL de un DBMS relacional específico.
(Nota: Si el modelo ER es una abstracción correcta del minimundo, entonces se puede comentar que, utilizando el algoritmo de transformación abajo indicado en forma adecuada, la BD estará normalizada)

Algoritmo.

1. Por cada tipo de entidades normal "E" del modelo ER, se crea una relación "R" que contenga todos los atributos simples de E. Se incluyen sólo los atributos simples y componentes de un atributo compuesto, si lo hay. Se elige uno de los atributos clave de E como clave primaria de R.
2. Por cada tipo de entidad débil D del esquema ER con tipo de entidades propietarias E, se crea una relación R y se incluyen todos los atributos simples (o componentes simples de los atributos compuestos) de D como atributos de R. Además, se incluyen como atributos de clave externa de R los atributos de clave primaria de la relación o relaciones que corresponden al tipo o tipos de entidades propietarias; con esto damos cuenta del tipo de relación o vinculo identificador de D. La clave primaria de R es la combinación de las claves de las propietarias y la clave parcial de D, si existe.
3. Por cada tipo de relación binario 1:1 R del esquema ER, se identifican las relaciones S y T que corresponden a los tipos de entidades que participan en R. Se escoge una de las relaciones –digamos S- y se incluye como llave externa en S la clave primaria de T. Es mejor elegir un tipo de entidades con participación total R en el papel de S. Se incluyen todos los atributos simples (o componentes simples de los atributos compuestos) del tipo de relación 1:1 R como atributos de S.
4. Por cada tipo de vínculo o tipo de relación normal 1: N R, se identifica la relación S que representa el tipo de entidades participantes del lado N del tipo de vínculos. Se incluye como llave externa en S la llave primaria de la relación T que representa al otro tipo de entidades que participan en R; la razón es que cada ejemplar de entidad del lado N está relacionado con un máximo de un ejemplar de entidad del lado 1. Se incluyen todos los atributos simples (o componentes simples de los atributos compuestos) del tipo de vínculo 1: N como atributos de S.
5. Por cada tipo de relación R binario M:N, se crea una nueva relación S para representar a R. Se incluyen como atributos de llave externa en S las llaves primarias de las relaciones que representan los tipos de entidades participantes; su combinación constituirá la llave primaria de S. También se incluyen todos los atributos simples (o componentes simples de los atributos compuestos) del tipo de vínculo M: N como atributos de S. Observe que no podemos representar un tipo de vínculos M:N con un solo atributo de clave externa en una de las relaciones participantes –como hicimos en

el caso de los tipos de vínculos 1:1 y 1:N- debido a la razón de cardinalidad M:N. Cabe destacar que siempre es posible transformar los vínculos 1:1 o 1:N de una manera similar a como se hace con los vínculos M:N. Esta alternativa es útil sobre todo cuando hay pocos ejemplares del vínculo, a fin de evitar valores nulos en las llaves externas. En este caso, la llave primaria de la relación "vinculo" será la llave externa de sólo una de las relaciones "entidad" participantes. En el caso de un vínculo 1: N, ésta será la relación entidad del lado N; en el caso de un vínculo 1:1 se elegirá la relación entidad con participación total (si existe).

6. Por cada atributo multivaluado A se crea una nueva relación R que contiene un atributo correspondiente A más el atributo de llave primaria K (como llave externa en R) de la relación que representa el tipo de entidades o de vínculos que tiene A como atributo. La clave primaria de R es la combinación de A y K. Si el atributo multivaluado es compuesto, se incluyen sus componentes simples.
7. Por cada tipo de vínculos n-ario R, n>2, se crea una nueva relación S que represente a R. Se incluyen como atributos de llave externa en S las llaves primarias de las relaciones que representan los tipos de entidades participantes. También se incluyen los atributos simples (o los componentes simples de los atributos compuestos) del tipo de vínculos n-ario como atributos de S. La llave primaria de S casi siempre es una combinación de todas las llaves externas que hacen referencia a las relaciones que representan los tipos de entidades participantes. No obstante, si la restricción de cardinalidad en cualquiera de los tipos de entidad E que participan en R es uno, entonces la llave primaria de S no debe incluir el atributo de la llave externa que hace referencia a la relación E' correspondiente a E; la razón es que, en este caso, cada una de las entidades e de E participará en cuando más un ejemplar de vínculo de R y, por tanto, podrá identificar de manera única ese ejemplar.

Ejemplo de la transformación del modelo Entidad Relación al modelo Relacional.
Usando el esquema ER de la base de datos COMPAÑÍA de la sección 3.3.3, fig. 3.8 se explicará la traducción al modelo relacional empleando el algoritmo antes señalado:

1. Se crean las relaciones EMPLEADO, DEPARTAMENTO y PROYECTO que corresponden a los tipos de entidades normales EMPLEADO, DEPARTAMENTO y PROYECTO de la sección 3.3.3. A éstas en ocasiones se les denomina relaciones "entidades". No se incluyen todavía los atributos de llave externa y de vínculo; se agregarán durante los pasos subsecuentes. Entre ellos están los atributos SUPERVISOR y DNO de EMPLEADO, MATRIGERENTE y FECHAING de DEPARTAMENTO y DNO de PROYECTO. Escogemos MATRICULA, DNUMERO, Y PNUMERO como llaves primarias de las relaciones EMPLEADO, DEPARTAMENTO y PROYECTO respectivamente.
2. En nuestro ejemplo creamos la relación DEPENDIENTE en este paso, que corresponde al tipo de entidad débil DEPENDIENTE. Incluimos la clave primaria de la relación EMPLEADO –que corresponde al tipo de entidades propietarias- como atributo de llave externa de DEPENDIENTE; cambiamos su nombre a EMPLEADOMAT, aunque no era preciso hacerlo. La llave primaria de la relación DEPENDIENTE es la combinación de {EMPLEADOMAT,NOMBRE_DEPENDIENTE} ya que NOMBRE_DEPENTIENTE es la llave parcial de DEPENDIENTE.

3. En nuestro ejemplo, transformamos el tipo de relación 1:1 DIRIGE de la sección 3.3.3 eligiendo DEPARTAMENTO para desempeñar el papel de S, debido a que su participación de DIRIGE es total (todo departamento tiene un gerente). Incluimos la clave primaria de la relación EMPLEADO como clave externa en la relación DEPARTAMENTO y cambiamos su nombre a MATRIGERENTE. También incluimos el atributo simple FECHAINGR de DIRIGE en la relación DEPARTAMENTO.
4. En nuestro ejemplo, establecemos ahora la transformación de los tipos de vínculos PERTENECE_A, CONTROLA y SUPERVISION de la sección 3.3.3. En el caso de PERTENCE_A incluimos la llave primaria de la relación DEPARTAMENTO como llave externa en la relación EMPLEADO y la llamamos DNO. En el caso de SUPERVISION, incluimos la llave primaria de la relación EMPLEADO como llave externa en la relación empleado y la llamamos SUPERV. El vínculo CONTROLA corresponde al atributo de clave externa DNO de PROYECTO.
5. En nuestro ejemplo, establecemos la transformación del tipo de vínculo M:N TRABAJA_EN de la sección 3.3.3 creando la relación TRABAJA_EN. La relación TRABAJA_EN recibe a veces el nombre de relación "vinculo", ya que corresponde a un tipo de vínculos. Incluimos las llaves primarias de las relaciones PROYECTO y EMPLEADO como claves externas en TRABAJA_EN. También incluimos un atributo HORAS del tipo de vínculos. La clave primaria de la relación TRABAJA_EN es la combinación de los atributos de clave externa {MATRICULAEMP, PNO}.
6. En nuestro ejemplo, creamos una relación LOCAL_DEPTO. El atributo LUGARES representa el atributo multivaluado DLOCALIZACION representa el atributo multivaluado LUGARES de DEPARTAMENTO, en tanto que DNUMERO –como clave externa- representa la llave primaria de la relación DEPARTAMENTO. La clave primaria de LOCAL_DEPTO es la combinación de {DNUMERO, DLOCALIZACION}. Habrá una tupla en LOCAL_DEPTO por cada lugar en que esté ubicado un departamento.

Observe que no analizamos la transformación de los tipos de vínculos n-arios (n>2) porque no hay ninguno en la sección 3.3.3. El modelo relacional se observa en la fig. 4.33.

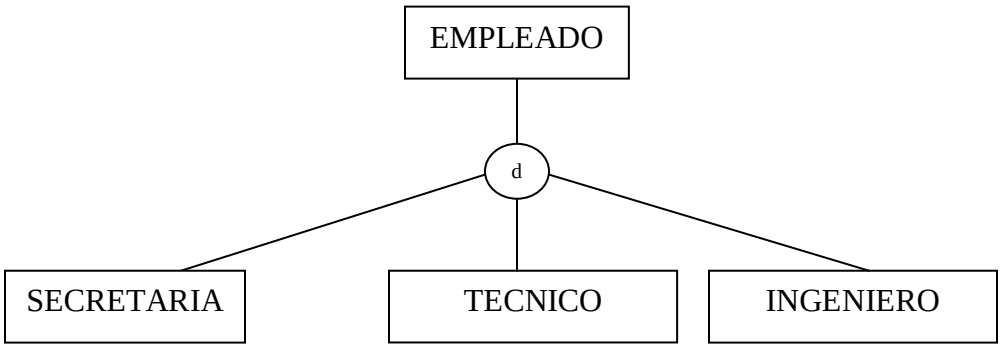
4.3.4 TRANSFORMANDO EL MODELO ERE AL MODELO RELACIONAL.

Existen varias opciones para transformar un número de subclases que forman juntas una especialización (o alternativamente, que son generalizadas dentro de una superclase), tales como las subclases de {SECRETARIA, TECNICO E INGENIERO} de empleado. Lo que se va a realizar es introducir otros pasos más al ya conocido algoritmo de transformación del modelo ER al modelo relacional. El paso 8 da las opciones más comunes; estas opciones no son las únicas, pueden existir otras. Se emplea la notación Atrs(R) para denotar los atributos de la relación R y LP(R) para denotar la llave primaria de R.

Paso 8. Convierta cada especialización con m subclases $\{S_1, S_2, S_3, \ldots, S_m\}$ y superclases C, donde los atributos de C son $\{k, a_1, a_2, \ldots, a_n\}$ y k es la llave primaria, dentro de los siguientes esquemas de relación usando uno de los siguientes criterios:

Opción 8A: Crear una relación L para C con atributos Atrs (L)= $\{k, a_1, a_2, \ldots, a_n\}$ y LP=k. Crear una relación L_i para cada subclase S_i, $1 \leq i \leq m$, con los atributos $Atrs(L_i)=\{k\} \cup \{\text{Atributos de } S_i\}$ y $LP(L_i)=k$.
Opción 8B: Crear una relación L_i para cada subclase S_i, $1 \leq i \leq m$, con los atributos $Atrs(L_i)=\{\text{atributos de } S_i\} \cup \{k, a_1, a_2, \ldots, a_n\}$ y $LP(L_i) = k$.
Opción 8C: Crear una relación simple L con los atributos Atrs(L)= $\{k, a_1, a_2, \ldots, a_n\} \cup \{\text{atributos de } S_1\} \cup \ldots \cup \{\text{atributos de } S_n\} \cup \{t\}$ y LP{L}=k. Esta opción es para una especialización cuyas subclases son disjuntas, y t es un tipo de atributo que indica la subclase a la que pertenece la tupla. Esta opción tiene el potencial de generar una gran cantidad de blancos.
Opción 8D: Crear un esquema simple L con los atributos Atrs(L)= $\{k, a_1, a_2, \ldots, a_n\} \cup \{\text{atributos de } S_1\} \cup \ldots \cup \{\text{atributos de } S_n\} \cup \{t_1, t_2, t_3, \ldots, t_m\}$ y LP{L}=k. Esta opción es para una especialización cuyas subclases se sobreponen y cada t_i $1 \leq i \leq m$, es un atributo booleano indicando si una tupla pertenece a una subclase S_i.

Opción 8A:



EMPLEADO

Matricula	Paterno	Materno	Nombre	F_Nac	Dirección	Tipo_trabajo

SECRETARIA

Matricula	Velocidad_Escr

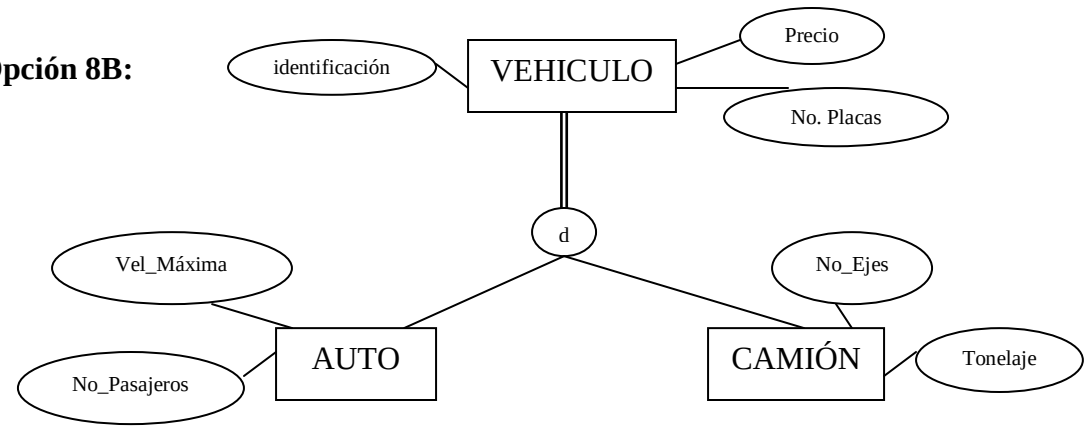
TECNICO

Matricula	Tipo_Técnico

INGENIERO

Matricula	Tipo_Ing

Opción 8B:



AUTO

Identificación	No_Placa	Precio	Vel_Máxima	No_Pasajeros

CAMIÓN.

Identificación	No_Placa	Precio	No_Ejes	Tonelaje

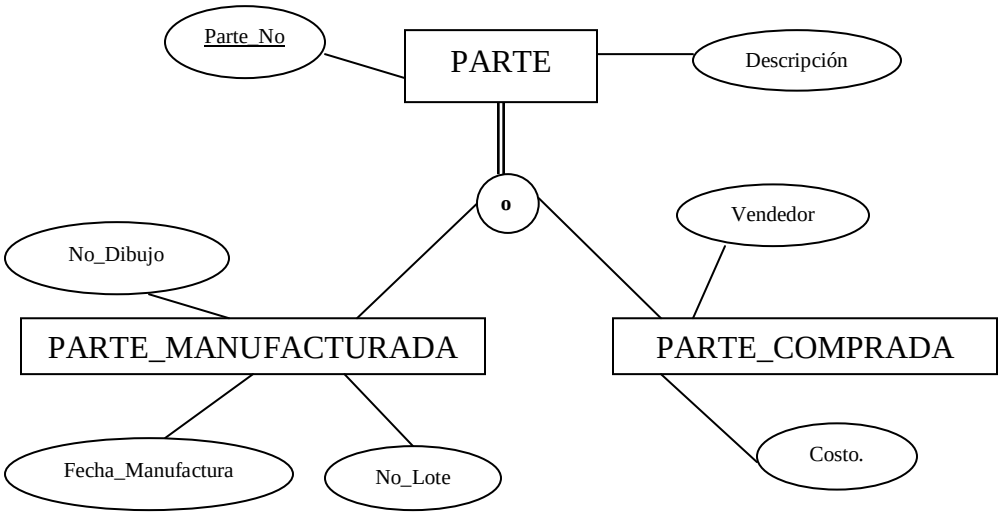
Opción 8C:

Utilizando la superclase {EMPLEADO} anterior, se tiene:

EMPLEADO

Matricula	Paterno	Materno	Nombre	F_Nac	Dirección	Tipo_Trabajo	Vel_Escr	Grado_Tec	Tipo_ing.

Opción 8D:



PARTE

Parte_no.	Descripción	Bandera_Manuf.	Dibujo_No.	F_Manufactura	No_Lote	Band_Comprado	Lista_Precio

Fig. 4.34

LATTICE

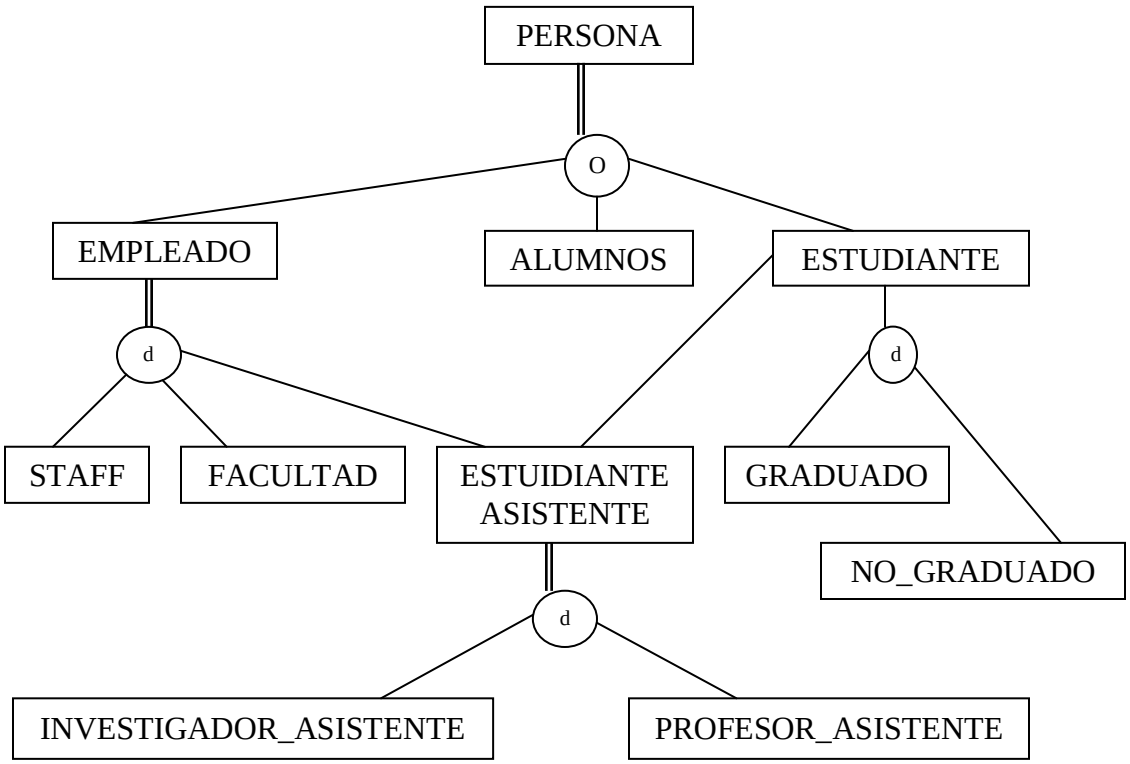


Fig. 4.35

Los atributos del ejemplo son:
PERSONA={Nombre, sexo, dirección, fecha de nacimiento, RFC}
EMPLEADO={Salario} ALUMNO={RFC, GRADOS({GRADO,AÑO,MAYOR})}
ESTUDIANTE={Departamento} STAFF={Posición} FACULTAD{Rango}
GRADUADO={Programa} NO_GRADUADO={Clase}
INVESTIGADOR_ASISTENTE={Proyecto}
PROFESOR_ASISTENTE={Curso}
ESTUDIANTE_ASISTENTE={%Tiempo}

Para mostrar una posible transformación de un lattice al modelo relacional se utilizará la figura 4.35. Se usará la opción 8A para PERSONA/ {EMPLEADO, ALUMNOS, ESTUDIANTE}, opción 8C tanto para EMPLEADO/ {STAFF, FACULTAD, ESTUDIANTE_ASISTENTE} como para ESTUDIANTE/ ESTUDIANTE_ASISTENTE., y la opción 8D para ESTUDIANTE_ASISTENTE/{INVESTIGADOR_ASISTENTE, PROFESOR_ASISTENTE} y ESTUDIANTE/ {GRADUADO, NO_GRADUADO}.

PERSONA

RFC	Nombre	F_Nac	Sexo	Dirección

EMPLEADO

RFC	Salario	Tipo_Empleado	Posición	Rango	%_Tiempo	Bandera_I_A	Bandera_P_A	Proyecto	Curso

ALUMNO

RFC

GRADO DEL ALUMNO

RFC	AÑO	GRADO	MAYOR

ESTUDIANTE

RFC	DEPARTAMENTO	Band_Graduado	Band_No_Graduado	Clase	Prog._Grad.	Band_Estudante_Asistente

Fig. 4.36

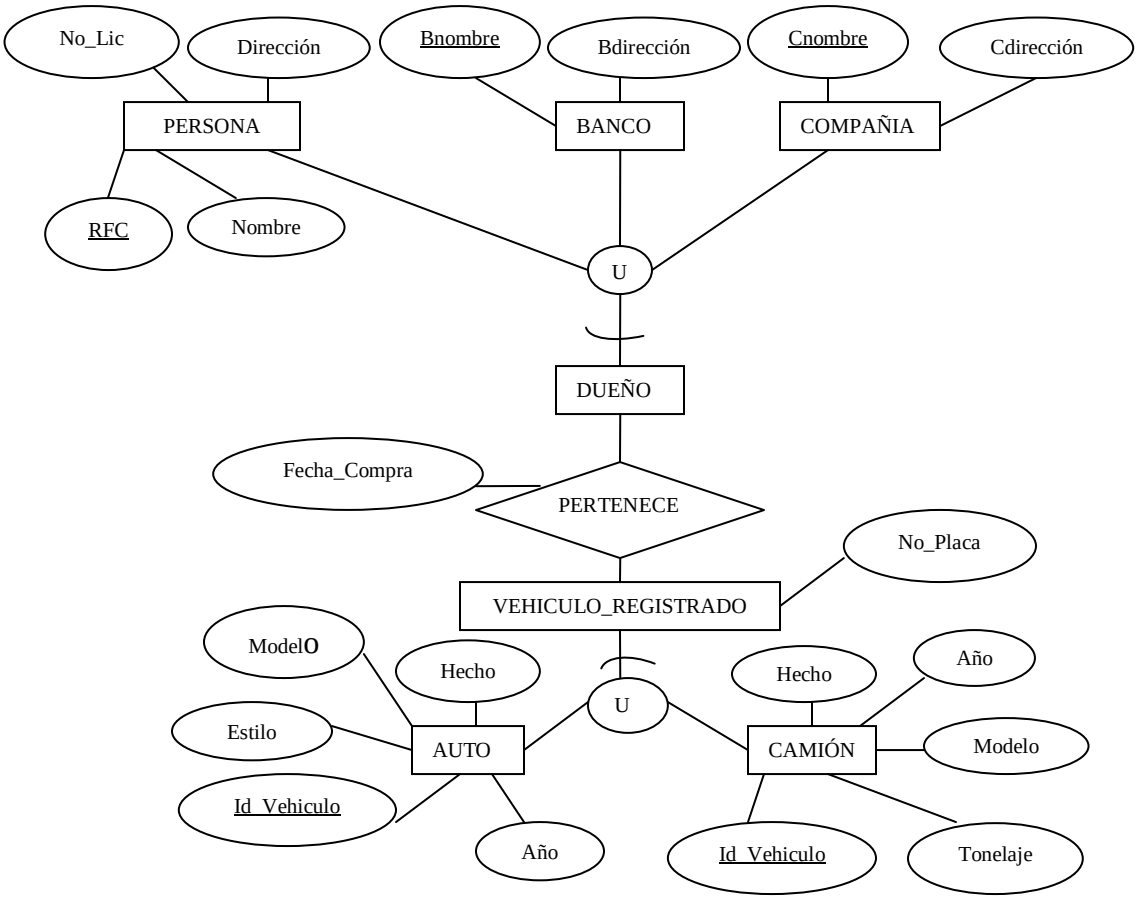
Una subclase compartida es una subclase de varias superclases. Estas clases deben de tener el mismo atributo llave; de otra forma, la subclase debería de ser modelada como categoría. Se pueden aplicar cualquiera de las opciones discutidas en el paso 8, aunque la opción 8A es la más usada. La figura anterior usa la opción 8D para la subclase compartida ESTUDIANTE_ASISTENTE.

TRANSFORMACIÓN DE CATEGORÍAS.

Una categoría es una subclase de la unión de dos o más superclases que pueden tener diferentes llaves porque pueden ser de diferentes tipos de entidades. Un ejemplo es la categoría mostrada en la figura 4.37.

Para transformar una categoría cuyas superclases tienen diferentes llaves, se acostumbra especificar una nueva llave, llamada **llave sustituta**, cuando se crea una relación que corresponda a la categoría. Esto es porque las llaves de las clases que definen a la subclase son diferentes, de esta forma no se puede usar cualquiera de las llaves en forma exclusiva para identificar todos los tipos de entidad en la categoría. Ahora se puede crear una relación que corresponda al esquema DUEÑO que corresponda a la categoría DUEÑO. La llave primaria de DUEÑO es la llave sustituta Id_dueño. También se adiciona la llave sustituta Id_dueño como llave foránea a cada relación correspondiente a la superclase de la categoría.

Para la categoría cuyas superclases tienen la misma llave, tal como VEHICULO no se requiere de llave sustituta.



PERSONA

RFC	No_Lic	Nombre	Dirección	Id_Dueño

VEHICULO_REGISTRADO

Id_Vehiculo	No_Placa

BANCO

Bnombre	Bdirección	Id_Dueño

AUTO

Id_Vehiculo	Estilo	Hecho	Modelo	Año

COMPAÑÍA

Cnombre	Cdirección	Id_Dueño

CAMIÓN

Id_Vehiculo	Hecho	Modelo	Tonelaje	Año

DUEÑO

Id_Dueño

PERTENECE

Id_Dueño	Id_Vehiculo	Fecha_de_Compra

Fig. 4.37

4.3.6 LAS DOCE REGLAS DE CODD.

En su artículo de 1985 en Computerworld, Ted Codd presentó doce reglas que una base de datos debe obedecer para que sea considerada verdaderamente relacional. Las doce reglas de Codd se muestran en la tabla 4.1, y desde entonces se han convertido en una

definición semioficial de una base de datos relacional. Las reglas se derivan del trabajo teórico de Codd sobre el modelo relacional, y representan realmente más un objetivo ideal que una definición de una base de datos relacional.
Ningún DBMS relacional actualmente disponible satisface totalmente las doce reglas de Codd. De hecho, se está convirtiendo en una práctica popular elaborar <tarjetas de tanteo> para productos DBMS comerciales, que muestran bien o mal que éstos satisfacen a una de estas reglas. Desgraciadamente, las reglas son subjetivas de modo que los calificadores están generalmente llenos de notas a pie de página y no revelan demasiado acerca de los productos.
La regla 1 es básicamente la definición informal de una base de datos relacional presentada al comienzo de esta sección. La regla 2 refuerza la importancia de las claves primarias para localizar datos en la base de datos. El nombre de la tabla localiza la tabla correcta, el nombre de columna encuentra la columna correcta y el valor de la clave primaria encuentra la fila que contiene un dato individual de interés. La regla 3 requiere soporte para falta de datos mediante el uso de valores nulos.
La regla 4 requiere que una base de datos relacional sea autodescriptiva. En otras palabras, la base de datos debe contener ciertas tablas de sistema cuyas columnas describan la estructura de la propia base.
La regla 5 ordena la utilización de un lenguaje de bases de datos relacional, tal como SQL. El lenguaje debe ser capaz de soportar todas las funciones básicas de un DBMS – creación de una base de datos, recuperación y entrada de datos, implementación de la seguridad de la base de datos, etc.
La regla 6 trata de las vistas, que son tablas virtuales utilizadas para dar diferentes usuarios de una base de datos diferentes vistas de su estructura. Es una de las reglas más difíciles de implementar en la práctica, y ningún producto comercial la satisface totalmente hoy día.
La regla 7 refuerza la naturaleza orientada a conjuntos de una base de datos relacional. Requiere que las filas sean tratadas como conjuntos en operaciones de inserción, supresión y actualización. La regla está diseñada para prohibir implementaciones que sólo soportan la modificación o recorrido fila a fila de la base de datos.
La regla 8 y la regla 9 aíslan al usuario o al programa de aplicación de la implementación de bajo nivel de la base de datos. Especifican que las técnicas de acceso a almacenamiento específicas utilizadas por el DBMS, e incluso los cambios a la estructura de las tablas en la base de datos, no deberían afectar a la capacidad del usuario de trabajar con datos.
La regla 10 dice que el lenguaje de base de datos debería soportar las restricciones de integridad que restringen los datos que pueden ser introducidos en la base de datos y las modificaciones que puedan ser efectuadas en ésta. Esta es otra de las reglas que no soportan la mayoría de los productos comerciales DBMS.
La regla 11 dice que el lenguaje de bases de datos debe ser capaz de manipular datos distribuidos localizados en otro sistema de información. La regla 12 impide otros caminos en la base de datos que pudieran subvertir su estructura relacional y su integridad.

1. La regla de información. Toda la información de una base de datos relacional está respaldada explícitamente a nivel lógico y exactamente de un modelo –mediante valores en tablas.
2. Regla de acceso garantizado. Todos y cada uno de los datos (valor atómico) de una base de datos relacional se garantiza que sean lógicamente accesibles recurriendo a una combinación de nombre de tabla, valor de clave primaria y nombre de columna.
3. Tratamiento sistemático de valores nulos. Los valores nulos (distintos de la cadena de caracteres vacía o de una cadena de caracteres en blanco y distinta del cero o de cualquier otro número) se soportan en los DBMS completamente relacionales para representar la falta de información y la información inaplicable de un modo sistemático e independiente del tipo de datos.
4. Catálogo en línea dinámica basado en el modelo relacional. La descripción de la base de datos se representa a nivel lógico del mismo modo que los datos ordinarios, de modo que los usuarios autorizados puedan aplicar a su interrogación el mismo lenguaje relacional que aplican a los datos regulares.
5. Regla de sublenguaje completo de datos. Un sistema relacional puede soportar varios lenguajes y varios modos de uso terminal (por ejemplo, el modo de rellenar con blancos). Sin embargo, debe haber al menos un lenguaje cuyas sentencias sean expresables, mediante alguna sintaxis bien definida, como cadenas de caracteres, y que sea completa en cuanto al soporte de todos los puntos siguientes:
   - Definición de datos.
   - Definición de vistas.
   - Manipulación de datos (interactiva y por programa).
   - Restricciones de integridad.
   - Autorización
   - Fronteras de transacciones (comienzo, cumplimentación y vuelta atrás).
6. Regla de actualización de vistas. Todas las vistas que sean teóricas actualizables son también actualizables por el sistema.
7. Inserción, actualización y supresión de alto nivel. La capacidad de manejar una relación de base de datos o una relación derivada como un único operando se aplica no solamente a la recuperación de datos, sino también a la inserción, actualización y supresión de datos.
8. Independencia física de los datos. Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cualesquiera que sean los cambios efectuados ya sea las representaciones de almacenamiento o a los métodos de acceso.
9. Independencia lógica de los datos. Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cuando se efectúen sobre las tablas de base cambios preservadores de la información de cualquier tipo que teóricamente permita alteraciones.
10. Independencia de integridad. Las restricciones de integridad específicas para una base de datos relacional particular deben ser definibles en el sublenguaje de datos relacional y almacenables en el catálogo, no en los programas de aplicación.
11. Independencia de distribución. Un DBMS relacional tiene independencia de distribución.

12. Regla de subversión. Si un sistema relacional tiene un lenguaje de bajo nivel (un solo registro cada vez), ese bajo nivel no puede ser utilizado para subvertir o suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior. (múltiples registros a la vez).

Tabla 4.1 Las 12 reglas de Codd para DBMS relacional.

4.4 EJERCICIOS

1. Defina el modelo relacional.
2. Defina dominio.
3. Defina esquema de relación.
4. Defina grado de relación.
5. Defina ejemplar de relación.
6. Defina intención y extensión (o estado) de una relación.
7. Explique la restricción de dominio.
8. Explique la restricción de llave.
9. Compare súper llave con llave o clave primaria.
10. Defina un esquema de B. D.'s.
11. Defina un ejemplar de B. D.'s.
12. Explique la restricción de integridad de entidades.
13. Explique la función de la clave o llave externa.
14. Comente cómo funciona la restricción de integridad referencial.
15. Comente qué es la operación "insertar".
16. Comente qué es la operación "eliminar".
17. Comente qué es la operación "modificar".
18. Indique cuales pueden ser las violaciones de las siguientes inserciones y el porqué. Use la extensión de base de datos mostrada en las páginas 54 y 55.
    a. Empleado<'Pedro', 'Liu', 'Tang','12345','010950','Texcoco','M',2800,null,'4'>
    b. Empleado<'Laura','Soto','Luna','87777',…..>
    c. Empleado<'Lorena','Aldama','Hidalgo', null,….>
    d. Empleado<'Cecilia','Lao','Tse','789789','050550','Tocuila','F',200,'54321','7'>
19. Indique cuáles pueden ser las violaciones al borran las siguientes tuplas.
    a. En trabaja_en, la tupla con matriculaempl='87777' y PNO = 10.
    b. En empleado, la tupla con matrícula = '45555'.
    c. En departamento, la tupla con Dnumero= 5.
    d. En proyecto, la tupla con pnumero = 10.
1. Indique cuáles pueden ser las violaciones al modificar las siguientes tuplas.
    a. Empleado<'Bob', 'Lao','Liu','75543','010157','Atenco','M',390,'65432','1'>
    b. Proyecto<'ProductoA',4,'Tocuila',2>
    c. Departamento<'Producción',4,'Texcoco','75543','011088'>
    d. Trabaja_en<'53453',null,40.0>
    e. Dependiente<'53453','Juan','M','121260','Hijo'>
2. De cada una de los requerimientos dados como ejercicio de la sección 3.6, transformarlo al modelo relacional.

5 CONSULTAS

(Elmasri & Navate, 2008)
(Reinosa, Maldonado, Nuñoz, Damiano, & Abrustsky, 2012)

5.1 ALGEBRA RELACIONAL.

5.1.1 INTRODUCCION

Hasta aquí sólo hemos examinado los conceptos para definir la estructura y las restricciones de un modelo de base de datos en el modelo relacional para ejecutar las operaciones relacionales de actualización. Ahora dirigiremos nuestra atención al álgebra relacional: una colección de operaciones que sirven para manipular relaciones enteras. Estas operaciones sirven, por ejemplo, para seleccionar tuplas de relaciones individuales y para combinar tuplas relacionadas a partir de varias relaciones con el fin de especificar una consulta –una solicitud de obtención- de la base de datos. El resultado de cada operación es una nueva relación, que podremos manipular en una ocasión futura.

5.1.2 UN PANORAMA GENERAL DEL ALGEBRA

El álgebra relacional consiste en un conjunto de operadores de alto nivel que operan sobre relaciones. Cada uno de estos operadores toma una o dos relaciones como entrada y produce una nueva relación como salida. Desde luego, sería posible en principio definir cualquier cantidad de operadores que se ajustan a esta sencilla definición (ya sea una o dos relaciones como entrada, otra relación como salida). No obstante, Codd definió un conjunto muy específico de ocho operadores de este tipo, en dos grupos de cuatro cada uno:

1. Las operaciones tradicionales de conjuntos. Unión, intersección, diferencia y producto cartesiano. (todas ellas con ligeras modificaciones, debidas al hecho de tener relaciones como operando, y no conjuntos arbitrarios; después de todo, una relación es un tipo especial de conjunto); y
2. Las operaciones relacionales especiales. Selección, proyección, reunión y división.

Por añadidura, Codd tenía un claro objetivo cuando definió precisamente estos ocho operadores. Pero se debe entender que sin duda es posible definir operadores adicionales de naturaleza algebraica, y en efecto muchos de ellos han sido propuestos por distintos autores. En la primera parte de esta sección analizaremos primero los operadores originales de Codd.

Los ocho operadores originales se presentan en forma simbólica en la figura 5.1. A grandes rasgos funcionan como sigue:

SELECCION:	Extrae las tuplas especificadas de una relación dada (o sea, restringe la relación sólo a las tuplas que satisfagan una condición especifica)
PROYECCION:	Extrae los atributos especificados de una relación dada.
PRODUCTO:	A partir de dos relaciones especificadas, constituye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones.
UNION:	Construye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones especificadas.
INTERSECCION:	Construye una relación formada por aquellas tuplas que aparezcan

en las dos relaciones especificadas.

DIFERENCIA: Construye una relación formada por todas las tuplas de la primera relación que no aparezcan en la segunda de las dos relaciones específicas.

REUNION: A partir de dos relaciones especificadas, construye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las dos relaciones, tales que las dos tuplas participantes en una combinación dada satisfagan alguna condición especifica.

DIVISION: Toma dos relaciones, una binaria y una unaria, y construye una relación formada por todos los valores de un atributo de la relación binaria que concuerdan (en el otro atributo) con todos los valores en la relación unaria.

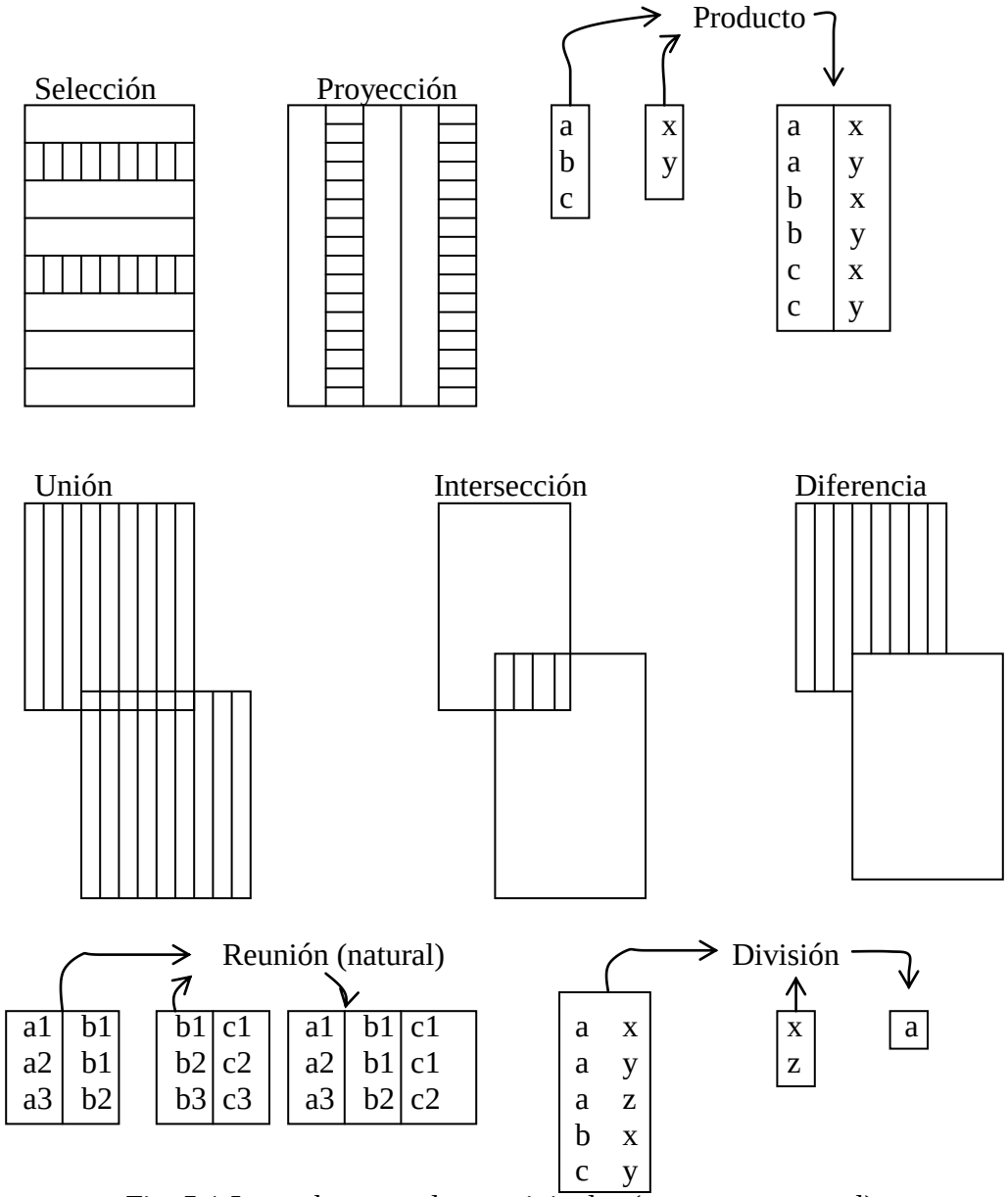


Fig. 5.1 Los ocho operadores originales (panorama general).

Adviértase que el resultado de cada una de las operaciones es (por supuesto) otra relación. **Esta es la importantísima propiedad de cerradura**. En esencia, **dado que el resultado de cualquier operación es un objeto del mismo tipo que los operandos** – todos son relaciones- el resultado de una operación puede convertirse en operando de otra. Así pues, es posible (por ejemplo) sacar la proyección de una unión, o una reunión de dos restricciones, o la diferencia de una unión y una intersección, etcétera, etcétera. Dicho de otro modo, es posible escribir expresiones relacionales anidadas; es decir, expresiones en las cuales los operandos mismos están representados mediante expresiones, y no sólo mediante nombres.

Nota: Existe una analogía obvia entre la capacidad de anidar expresiones algebraicas en álgebra relacional y la capacidad de anidar expresiones aritméticas en aritmética ordinaria. En efecto, el hecho de que las relaciones estén cerradas en el álgebra es importante exactamente por las mismas razones por las que es importante el hecho de que los números estén cerrados en la aritmética ordinaria.

5.1.3 UNA SINTAXIS PARA EL ALGEBRA RELACIONAL.

5.1.3.1 La operación SELECCIONAR (σ)

La operación SELECCIONAR sirve para seleccionar un subconjunto de tuplas de una relación que satisfacen una condición de selección. Por ejemplo, para seleccionar el subconjunto de tuplas EMPLEADO que trabajan en el departamento 4 o cuyo salario rebasa los $3,000.00 podemos especificar las dos condiciones con la operación SELECCIONAR, como sigue:

$$\sigma_{\text{DNO =4 OR SALARIO > \$3,000.00}}(\text{EMPLEADO})$$

En general, denotamos la operación SELECCIONAR con

$$\sigma_{\text{<condición de selección>}}(\text{<nombre de la relación>})$$

donde el símbolo σ (sigma) denota el operador de SELECCIONAR, y la condición de selección es una expresión booleana especificada en términos de los atributos de la relación.

La relación que resulta de la operación SELECCIONAR tiene los mismos atributos que la relación especificada en <nombre de la relación>. La expresión booleana especificada en la <condición de selección> se compone de una o más cláusulas de la forma

<nombre del atributo> <operador de comparación> <valor constante>, o

<nombre del atributo> <operador de comparación> <nombre del atributo>

donde <nombre de atributo> es el nombre de un atributo en <nombre de la relación>, <operador de comparación> es normalmente uno de los operadores {=, <, <=, >=, <>} y <valor constante> es un valor constante del dominio del atributo. Las cláusulas pueden conectarse arbitrariamente con los operadores booleanos Y (AND), O (OR) y NO (NOT) para formar una condición de selección en general. Por ejemplo, si queremos seleccionar las tuplas de todos los empleados que trabajan en el departamento 4 y ganan más de $2,500.00 o que trabajan en el departamento 5 y ganan más de $3,000.00, podemos especificar la siguiente operación SELECCIONAR:

$$\sigma_{(DNO=4\ Y\ SALARIO>2500)\ O\ (DNO=5\ Y\ SALARIO>3000)}(EMPLEADO)$$

El resultado se muestra en la figura 5.2.

nombre	paterno	materno	matricula	nacimiento	direccion	sexo	Salario	supervisor	dno
francisco	lopez	lopez	45555	8/12/45	sn diego	m	4000	00007	5
jennifer	ortega	lozano	54321	6/20/31	tocuila	f	4300	00007	4
ramses	sanchez	mendez	84444	9/15/52	sn miguel	m	3800	45555	5

Figura 5.2

El operador SELECCIONAR es unario; esto es, se aplica a una sola relación. Por añadidura, la operación de selección no puede abarcar más de una tupla. El grado de la relación resultante de una operación SELECCIONAR es el mismo que el de la relación original R a la que se aplicó la operación, porque tiene los mismos atributos de R.

5.1.3.2 La operación PROYECCION (Π)
Si visualizamos una relación como una tabla, la operación SELECCIONAR selecciona algunas filas de la tabla y desecha otras. La operación PROYECCION, en cambio, selecciona ciertas columnas de la tabla y desecha las demás. Por ejemplo, si queremos el apellido, en nombre y el salario de todos los empleados, podemos usar la siguiente operación:

$$\Pi_{PATERNO,\ NOMBRE,\ SALARIO}(EMPLEADO)$$

La relación resultante se muestra en la figura 5.3

paterno	nombre	Salario
bond	james	9000
lopez	francisco	4000
espana	josefa	2500
ortega	jennifer	4300
perez	juan	3000
sanchez	ramses	3800
zelaya	alicia	2500
hidalgo	aldama	2500

Fig. 5.3

La forma general de la operación PROYECTAR es

$$\Pi_{<LISTA\ DE\ ATRIBUTOS>}(<nombre\ de\ la\ relación>)$$

donde Π (pi) es el símbolo de la operación PROYECTAR y <lista de atributos> es una lista de atributos de la relación especificada por <nombre de la relación>. La relación así creada tiene sólo los atributos especificados en <lista de atributos> y en el mismo orden en que aparecen en lista. Por ello, su grado es igual al número de atributos en <lista de atributos>.
Si la lista de atributos sólo contiene atributos no llave de una relación, es probable que aparezcan tuplas repetidas en el resultado. La operación PROYECTAR elimina

implícitamente cualquier tupla repetida, así que el resultado de la operación PROYECTAR es un conjunto de tuplas y por tanto una relación válida. Por ejemplo, consideremos la siguiente operación:

$$\Pi_{\text{SEXO, SALARIO}}(\text{EMPLEADO})$$

El resultado se muestra en la figura 5.4. Siempre que aparezcan dos o más tuplas idénticas al aplicar una operación PROYECTAR, sólo una se conservará en el resultado; esto se conoce como **eliminación de duplicados** y es necesaria para garantizar que el resultado de la operación PROYECTAR sea también una relación: un *conjunto* de tuplas. El número de tuplas en una relación que resulta de una operación PROYECTAR siempre es menor que el número de tuplas de la relación original, o igual a él. Si la lista de proyección incluye una clave de la relación, la relación resultante tendrá el *mismo número* de tuplas que la origina.

sexo	salario
M	9000
M	4000
F	2500
F	4300
M	3000
M	3800
F	2500
M	2500

Fig. 5.4

5.1.3.3 Secuencia de operaciones y cambio de nombre de los atributos.
En general, es posible que deseemos aplicar varias operaciones del álgebra relacional una tras otra. Para ello, podemos escribir las operaciones en una sola expresión del álgebra relacional anidándolas, o bien, podemos aplicar una operación a la vez y crear relaciones de resultados intermedios. En el segundo caso, tendremos que nombrar las relaciones que contienen los resultados intermedios. Por ejemplo, si queremos obtener el nombre de pila, el apellido y el salario de todos los empleados que trabajan en el departamento no. 5, debemos aplicar una operación SELECCIONAR y una operación PROYECTAR. Podemos escribir una sola expresión del álgebra relacional, a saber:

$$\Pi_{\text{PATERNO, NOMBRE, SALARIO}}(\sigma_{\text{DNO=5}}(\text{EMPLEADO})).$$

Como alternativa, podemos mostrar explícitamente la secuencia de operaciones, dando un nombre a cada una de las relaciones intermedias, como sigue:

$$\text{EMPLS\_DEP5} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLEADO})$$

$$\text{RESULTADO} \leftarrow \Pi_{\text{NOMBRE, PATERNO, SALARIO}}(\text{EMPL\_DEP5})$$

A menudo es más sencillo descomponer una secuencia compleja de operaciones especificando relaciones de resultados intermedios que escribiendo una sola expresión del álgebra relacional. También podemos usar esta técnica para cambiar el nombre de los

atributos de las relaciones intermedias y de la resultante. Esto puede ser útil cuando se trata de operaciones más complejas. Si queremos cambiar de nombre los atributos de una relación resultante al aplicar una operación de álgebra relacional, bastará con que incluyamos una lista con los nuevos nombres de atributos entre paréntesis, como el siguiente ejemplo:

$$TEMP \leftarrow \sigma_{DNO=5}(EMPLEADO)$$

$$R(NOMPILA,APPATERNO,SALARIO) \leftarrow \prod_{NOMBRE,\ PATERNO,\ SALARIO}(TEMP)$$

Si no se cambian los nombres, los atributos de la relación resultante de una operación SELECCIONAR serán los mismos que los de la relación original y estarán en el mismo orden. En el caso de una operación PROYECTAR sin cambio de nombres, la relación resultante tendrá los mismos nombres de atributos que aparecen en la lista de proyección, y en el mismo orden.

5.1.3.4 Operaciones de la teoría de conjuntos.

Las operaciones de teoría de conjuntos se aplican al modelo relacional porque las relaciones se definen como conjuntos de tuplas, y pueden servir para procesar las tuplas de dos relaciones como conjuntos. Por ejemplo, si queremos obtener los números de matrícula de todo empleado que trabaje en el departamento 5 o que supervise directamente a un empleado que trabaje en ese mismo departamento, podemos utilizar la operación UNION como sigue:

$$EMPS\_DEP5 \leftarrow \sigma_{DNO=5}(EMPLEADO)$$

$$RESULTADO1 \leftarrow \prod_{MATRICULA}(EMP\_DEP5)$$

$$RESULTADO2(MATRICULA) \leftarrow \prod_{SUPERVISOR}(EMP\_DEP5)$$

$$RESULTADO \leftarrow RESLTADO1 \cup RESULTADO2$$

La relación RESULTADO1 contiene los números de matrícula de todos los empleados que trabajan en el departamento 5, y RESULTADO2 contiene los números de matrícula de todos los empleados que supervisan directamente a empleados que trabajan en el departamento 5.

Se utilizan varias operaciones de la teoría de conjuntos para combinar de diversas maneras los elementos de dos conjuntos, entre ellas, UNION, INTERSECCION y DIFERENCIA. Estas operaciones son binarias; es decir, se aplican a dos conjuntos. Al adaptar estas operaciones a la base de datos relacionales, debemos asegurarnos de que se puedan aplicar a dos relaciones para que el resultado también sea una relación válida. Para ello, las dos relaciones a las que se aplique cualquiera de las tres operaciones anteriores deberán tener el mismo tipo de tuplas; esta condición se denomina *compatibilidad de unión.* Se dice que dos relaciones R (A_1, A_2, A_3,. . ., A_n) y S(B_1, B_2,. . ., B_n) son compatibles con la unión si tienen el mismo grado n y si $dom(A_i) = dom(B_i)$ para $1 \leq i \leq n$. Esto significa que las dos relaciones tienen el mismo número de atributos y que cada par de atributos correspondientes tienen el mismo dominio.

Cabe señalar que tanto UNION como INTERSECCION son operaciones conmutativas; es decir,

$R \cup S = S \cup R$ Y $R \cap S = S \cap R$.

Ambas operaciones pueden aplicarse a cualquier número de relaciones, y las dos son operaciones asociativas; esto es,

$R \cup (S \cup T) = (R \cup S) \cup T$, y

$(R \cap S) \cap T = R \cap (S \cap T)$

La operación DIRERENCIA no es conmutativa, es decir, en general,

$R - S \neq S - R$.

El producto cartesiano, denotado por X. También es una operación binaria de conjuntos, pero las relaciones a las que se aplica no tienen que ser compatibles con la unión. Esta operación, conocida también como PRODUCTO CRUZADO o REUNION CRUZADA sirve para combinar tuplas de dos relaciones para poder identificar las tuplas relacionadas entre sí. En general, el resultado de $R(A_1, A_2, A_3, \ldots, A_n)$ X $S(B_1, B_2, \ldots, B_m)$ es una relación Q con n + m atributos $Q(A_1, A_2, A_3, \ldots, A_n, B_1, B_2, \ldots, B_m)$, en ese orden. La relación resultante Q tiene una tupla por cada combinación de tuplas: una de R y una de S. Por tanto, si R tiene n_R tuplas y S tiene n_s tuplas, R X S tendrá $n_R * n_S$ tuplas. Para ilustrar el empleo del PRODUCTO CARTESIANO, suponga que deseamos obtener una lista de los dependientes de cada uno de los empleados femeninos. Esto se puede lograr como sigue:

$$EMPS\_FEM \leftarrow \prod_{NOMBRE,PATERNO,\ MATRICULA}(\sigma_{SEXO='F'}(EMPLEADO))$$

$$DEPENDIENTES\_EMP \leftarrow EMPS\_FEM \times DEPENDIENTE$$

$$DEPENDIENTES\_REALES \leftarrow \sigma_{MATRICULA=EMPMATRICULA}(DEPENDIENTES\_EMP)$$

$$RESULTADO \leftarrow \prod_{NOMBRE,\ PATERNO,\ NOMBRE\_DEPENDIENTE}(DEPENDIENTES\_REALES).$$

El resultado se muestra en la figura 5.5.

EMPS_FEM

Nombre	paterno	Matricula
Alicia	zelaya	87777
Jennifer	ortega	54321
Josefa	espana	53453

DEPENDIENTES_EMP

nombre	Paterno	matricula	empleadomat	nombre	sexo	fechanac	relacion
Alicia	zelaya	87777	45555	alicia	f	5/4/76	hija
jennifer	ortega	54321	45555	alicia	f	5/4/76	hija
Josefa	espana	53453	45555	alicia	f	5/4/76	hija
Alicia	zelaya	87777	45555	teodoro	m	10/25/73	hijo
jennifer	ortega	54321	45555	teodoro	m	10/25/73	hijo
Josefa	espana	53453	45555	teodoro	m	10/25/73	hijo
Alicia	zelaya	87777	45555	juana	f	3/5/48	esposa
jennifer	ortega	54321	45555	juana	f	3/5/48	esposa
Josefa	espana	53453	45555	juana	f	3/5/48	esposa
Alicia	zelaya	87777	54321	abne	m	2/29/32	esposo
jennifer	ortega	54321	54321	abne	m	2/29/32	esposo
Josefa	espana	53453	54321	abne	m	2/29/32	esposo
Alicia	zelaya	87777	56789	miguel	m	1/1/78	hijo
jennifer	ortega	54321	56789	miguel	m	1/1/78	hijo
Josefa	espana	53453	56789	miguel	m	1/1/78	hijo
Alicia	zelaya	87777	56789	alicia	f	12/31/78	hija
jennifer	ortega	54321	56789	alicia	f	12/31/78	hija
Josefa	espana	53453	56789	alicia	f	12/31/78	hija
Alicia	zelaya	87777	56789	elizabeth	f	5/5/57	esposa
jennifer	ortega	54321	56789	elizabeth	f	5/5/57	esposa
Josefa	espana	53453	56789	elizabeth	f	5/5/57	esposa
Alicia	zelaya	87777	56789	juan	m	1/1/63	hijo
jennifer	ortega	54321	56789	juan	m	1/1/63	hijo
Josefa	espana	53453	56789	juan	m	1/1/63	hijo

DEPENDIENTES_REALES

Nombre	paterno	Matricula	empleadomat	nombre	Sexo	fechanac	relacion
jennifer	ortega	54321	54321	abne	M	2/29/32	esposo

RESULTADO

Nombre	Paterno	Nombrede
jennifer	ortega	abne

Fig. 5.5 Un ejemplo de la operación del PRODUCTO CARTESIANO

Como esta secuencia de PRODUCTO CARTESIANO seguido de SELECCIONAR se utiliza con mucha frecuencia para identificar y seleccionar tuplas relacionadas de dos relaciones, se creó una operación especial llamada REUNION, con el fin de especificar

esta secuencia con una sola operación. El PRODUCTO CARTESIANO nunca se usa como operación significativa por sí sola.

5.1.3.5 La operación REUNION.

La operación REUNION, denotada por $\triangleright\triangleleft$, sirve para combinar tuplas relacionadas de dos relaciones en una sola tupla. Esta operación es muy importante en cualquier base de datos relacional que comprenda más de una relación, porque permite procesar los vínculos entre las relaciones. A fin de ilustrar la reunión, supongamos que deseamos obtener el nombre del gerente de cada uno de los departamentos. Para obtenerlo, necesitamos combinar cada tupla de departamento con la tupla de empleado cuyo valor de matrícula del empleado coincida con la matricula del gerente. Esto se logra aplicando la operación REUNION y proyectando sobre los atributos requeridos.

$$\text{GTE\_DEPTO} \leftarrow \text{DEPARTAMENTO} \triangleright\triangleleft_{\text{MATRICULA=MATRIGERENTE}} \text{EMPLEADO}$$

$$\text{RESULTADO} \leftarrow \prod_{\text{DNOMBRE, APELLIDO, NOMBRE}} (\text{GTE\_DEPTO})$$

La segunda operación se ilustra en la figura 5.6.

GTE_DEPTO

dnombre	dnumero	Matrigerente	…	nombre	paterno	matricula	…
investigacion	5	45555	…	francisco	lopez	45555	…
Administracio	4	54321	…	jennifer	ortega	54321	…
prod.	1	00007	…	james	bond	00007	…

Fig. 5.6 La operación REUNION.

La forma general de una operación REUNION con dos relaciones $R(A_1, A_2, A_3, \ldots, A_n)$ y $S(B_1, B_2, \ldots, B_m)$ es

$$R \triangleright\triangleleft_{\text{<condición de reunión>}} S$$

El resultado de la REUNION es una relación Q con n + m atributos Q ($A_1, A_2, A_3, \ldots, A_n, B_1, B_2, \ldots, B_m$), en ese orden; Q tiene una tupla por una combinación de tuplas –una de R y una de S- *siempre que la combinación satisfaga la condición de reunión*. Cada combinación de tuplas para las cuales la evaluación de la condición con los valores de los atributos produzca el resultado 'verdadero' se incluirá en la relación resultante. Q, como una sola tupla.

Una condición de reunión tiene la forma:

<condición> y <condición> y . . . <condición>

donde cada condición tiene la forma $A_i \theta B_j$, A_i es un atributo de R, B_j

es un atributo de S, A_i y B_j tienen el mismo dominio y θ es uno de los operadores de comparación {=,<=,>=,<,>,<>,=}.

5.1.3.6 La operación división

La operación DIVISION es útil para un tipo especial de consultas que se presenta ocasionalmente en aplicaciones de bases de datos. Un ejemplo es: "obtener los nombres de los empleados que trabajan en todos los proyectos en los que trabaja Juan Pérez". Para expresar esta consulta con la operación DIVISIÓN, procedemos como sigue. Primero,

obtenemos la lista de números de los proyectos en los que trabaja Juan Pérez, colocando el resultado en la relación intermedia T1:

$$T1 \leftarrow \prod_{PNO}(TRABAJA\_EN \rhd\lhd_{MATRICULAEMP=MATRICULA}(\sigma_{NOMBRE='Juan' \text{ y } PATERNO='Pérez'}(EMPLEADO))$$

En seguida, creamos una relación intermedia que incluye <pno, matriculaemp>:

$$T2 \leftarrow \prod_{MATRICULAEMP,\ PNO}(TRABAJA\_EN)$$

Por último, aplicamos la operación DIVISIÓN a las dos relaciones, obteniendo los números de seguro social de los empleados que queremos:

$$T3 \leftarrow T2 \div T1$$

$$RESULTADO \leftarrow \prod_{NOMBRE,\ PATERNO}(T3 \rhd\lhd_{MATRICULAEMP=MATRICULA} EMPLEADO)$$

El operador DIVISIÓN se puede expresar como una secuencia de operaciones Π, X, y – como sigue:

$$T1 \leftarrow \Pi_Y(R)$$
$$T2 \leftarrow \Pi_Y((SXT1)-R)$$
$$T \leftarrow T1-T2$$

5.1.3.7 OTRAS OPERACIONES RELACIONALES.

Algunas solicitudes que se hacen comúnmente a las bases de datos no se pueden atender con las operaciones estándar del álgebra relacional. La mayoría de los lenguajes de consulta comercial para los DBMS relacionales cuentan con mecanismos para atender dichas solicitudes, y en esta sección definiremos operaciones adicionales para expresar las consultas. Estas operaciones amplían el poder expresivo del álgebra relacional.

5.1.3.7.1 Funciones agregadas.

El primer tipo de solicitud que no se puede expresar en el álgebra relacional es la especificación de funciones agregadas matemáticas sobre colecciones de valores de la base de datos. Un ejemplo sería la obtención del salario promedio o total de todos los empleados o el número de tuplas de empleados. Entre las funciones que suelen aplicarse a colecciones de valores numéricos están SUMA, PROMEDIO, MAXIMO y MINIMO. La función CUENTA sirve para contar tuplas. Todas estas funciones pueden aplicarse a una colección de tuplas.

Otro tipo de solicitud que se hace con frecuencia implica agrupar las tuplas de una relación según el valor de algunos de sus atributos y después aplicar una función agregada independientemente a cada grupo. Un ejemplo sería agrupar las tuplas de empleados por DNO de modo que cada grupo incluya las tuplas de los empleados que trabajan en el mismo departamento. Después podríamos preparar una lista de valores de DNO junto con, digamos, el salario promedio de los empleados del departamento. Podemos definir una operación FUNCION empleando el símbolo $\mathcal{F}$ para especificar estos tipos de solicitudes, como sigue:

$$<\text{atributos de agrupación}>\ \mathcal{F}\ <\text{lista de la relación}>(<\text{nombre de la relación}>)$$

donde <atributo de agrupación> es una lista de atributos de la relación especificada en <nombre de la relación>, y <lista de funciones> es una lista de pares

(<función><atributo>). En cada uno de esos pares, <función> es una de las funciones permitidas y <atributo> es un atributo de la relación especificada en <nombre de relación>. La relación resultante tiene los atributos de agrupación más un atributo por cada elemento de la lista de funciones. Por ejemplo, para obtener los números de departamento, el número de empleados de cada departamento y su salario promedio, escribimos

$$R\ (dep, numempl, salarioprom) \leftarrow {}_{DNO}\mathcal{F}_{CUENTA\ MATRICULAS,\ PROMEDIO\ SALARIO}(EMPLEADO)$$

El resultado de la operación se localiza en la figura 5.7.

Dno	Numempl	salarioprom
1	1	9000
4	3	3100
5	4	3325

Fig. 5.7

Si no se especifican atributos de agrupación, las funciones se aplicarán a los valores de los atributos de todas las tuplas de la relación, y la relación resultante tendrá una sola tupla. Por ejemplo, en la figura 6.8 se muestra el resultado de la siguiente operación:

$$\mathcal{F}_{CUENTA\ \ MATRICULA,PROMEDIO\ SALARIO}\ (EMPLEADO)$$

Vale la pena subrayar que el resultado de aplicar una función agregada es una relación, no un número escalar, aunque tenga un solo valor.

Expr1000	Expr1001
8	3950

Figura 5.8

5.1.3.7.2 Operaciones de reunión y unión externa.

Las operaciones de reunión antes descritas seleccionan tuplas que satisfacen la condición de reunión. En este caso, las tuplas que no cumplen con la condición de reunión se eliminan del resultado. Podemos usar un conjunto de operaciones, llamadas REUNIONES EXTERNAS, cuando queremos conservar en el resultado todas las tuplas que están en R o en S, o en ambas, ya que tengan o no tuplas coincidentes en la otra relación.

Por ejemplo, suponga que deseamos una lista de todos los nombres de empleados y también el nombre de departamentos que dirigen, si es el caso que dirijan un departamento; podemos aplicar una REUNIÓN EXTERNA IZQUIERDA denotado por ⟕ para obtener el resultado como sigue:

$$R \leftarrow \prod_{DNOMBRE,FECHAINGR,PATERNO,NOMBRE}(EMPLEADO \text{ ⟕ }_{MATRICULA=MATRIGERENTE} GERENTE)$$

La operación de reunión externa izquierda conserva en R ⟕ S todas las tuplas de la primera relación R (o relación de la izquierda); si no se encuentra una tupla coincidente en S, los atributos en S del resultado se rellenan con valores nulos. El resultado de esta operación se muestra en la figura 5.9.

Dnombre	fechaing	paterno	nombre
		perez	juan
Investigacion	5/22/78	lopez	francisco
		zelaya	alicia
Administracion	1/1/85	ortega	jennifer
		sanchez	ramses
		espana	josefa
		hidalgo	aldama
prod. Alimentos	5/19/71	bond	james

Fig. 5.9

Una operación similar, la REUNION EXTERNA DERECHA, denotada por ⟗, conserva en el resultado de R ⟗ S todas las tuplas de la segunda relación S (la de la derecha). Una tercera operación, la REUNION EXTERNA COMPLETA, denotada por ⟗, conserva todas las tuplas de ambas relaciones, izquierda y derecha, cuando no se encuentran tuplas coincidentes, rellenándolas con valores nulos si es necesario.
Otra capacidad que tienen casi todos los lenguajes comerciales (pero no el álgebra relacional) es la de especificar operaciones con los valores después de extraerlos de la base de datos. Por ejemplo, se pueden aplicar operaciones aritméticas como +, -, * a los valores numéricos.

5.2 CÁLCULO RELACIONAL.

5.2.1 **Introducción**

Mientras que su base matemática del algebra relacional es la teoría de conjuntos, el cálculo relacional se basa en el cálculo de predicados. El cálculo relacional especifica qué se va a recuperar sin indicar cómo hacerlo, mientras que el álgebra relacional indica, paso a paso, cómo recuperar los elementos solicitados. Por lo que el cálculo relacional es considerado declarativo y no procedural (o imperativo). El cálculo relacional es equivalente al álgebra relacional, por lo que cualquier consulta que se pueda expresar en cálculo relacional, se puede expresar en álgebra relacional. (En el cálculo relacional no se pueden realizar consultas que incluyan filtros. Para filtros, ver algebra relacional).
Existen dos formas de hacer cálculo relacional:

- Cálculo relacional por tuplas
- Cálculo relacional por dominio.

Estas notas mostrarán la forma de responder consultas con cálculo relacional por tuplas.

5.2.2Cálculo relacional por tuplas.

Es esquema básico del cálculo relacional por tuplas es:

$$\{t \mid cond(t)\}$$

Donde la parte izquierda muestra el resultado de la consulta dadas (es equivalente a la proyección del algebra relacional) las condiciones de la parte derecha de la expresión.

En forma más específica:

$$\{t_1.A_1, t_2.A_3, t_3.A_3, t_n.A_n \mid cond(t_1, t_1, t_2, t_3, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m})\}$$

Donde t_i son las tuplas y cada A_i son los atributos con respecto a la relación que está conectada con t_i, y la condición es una formula bien formada del cálculo de predicados.
Los operadores que se utilizan en el cálculo relacional son:

Símbolo	Nombre
$\forall$	Operador Universal
$\exists$	Operador Existencial
$\nexists$	Operador no Existencial
$\rightarrow$	Implicación.
Λ	AND
V	OR
$\sim$	NEGACIÓN

La negación del operador universal es el operador existencial. Por ejemplo, se puede decir el siguiente predicado:

Todo ser humano es valiente.

Se puede expresar matemáticamente como:

$$\forall x: VALIENTE(x)$$

Donde la variable x representa a los seres humanos.

Su negación sería: Existe al menos un ser humano que no es valiente, en lógica de predicados quedaría de la siguiente forma:

$$\exists x: \sim VALIENTE\ (x)$$

De la misma forma, la negación de un operador existencial es el operador universal. Por ejemplo:

Algún ser humano es valiente.

Este predicado quedaría el lógica de predicados de la siguiente manera:

$$\exists x: VALIENTE\ (x)$$

Su negación sería: Todos los seres humanos no son valientes.

$$\forall x: \sim\ VALIENTE(x)$$

La implicación tiene la virtud de poder ordenar predicados, por ejemplo:
Todo animal que es gato, invariablemente es felino. Todo animal que es felino, invariablemente es mamífero. Todo mamífero invariablemente es un ser vivo. De esta forma podemos ordenar los predicados de la siguiente forma:

$$\forall x:(GATO(x) \rightarrow FELINO(x) \rightarrow MAMIFERO(x) \rightarrow SER\ VIVO(x))$$

Pero no podemos decir: Todo animal que es felino, invariablemente es gato.

$$\forall x:(FELINO(x) \rightarrow GATO(x))$$

Ya que existen tipos de felinos que no son gatos (panteras, tigres, etc.). Esto indica que todo elemento que cumple con la parte izquierda del implicación, forzosamente cumple con la parte derecha de la implicación.

5.2.2.1 Consultas con el operador AND y el operador existencial:
Las consultas realizadas en estas secciones se basan en la extensión de la figura 4.5 (pág. 45).
El operador AND permite realizar la operación de selección vista en el álgebra relacional. Un ejemplo de ello se ve en la consulta 1.

1. Dar el nombre de todo empleado cuyo salario sea mayor a $5,000.ºº

   {e.nombre, e.paterno, e.materno | EMPLEADO (e) Λ e.salario > 5000.ºº}

El operador AND también permite realizar consultas que equivalen al operador "reunión" del algebra relacional. Un ejemplo de esto se puede observar en la consulta 2 y en la consulta 3.

2. Recupera el nombre y dirección de los empleados que trabajen para el departamento de Investigación.

   {e.paterno, e.nombre, e.dirección | EMPLEADO (e) Λ
   (∃d)DEPARTAMENTO(D) Λ d.dnombre="Investigación" Λ d.dnumero=e.dno}

3. Para todo proyecto localizado en San Diego, liste el número de proyecto, el número de departamento que lo controla y el nombre del gerente.

{p.pnumero, p.dno, e.paterno, e.nombre | PROYECTO (p) Λ (∃e)(EMPLEADO(e) Λ
p.plocalización= "San Diego" Λ (∃d)(DEPARTAMENTO(d) Λ p.dno=d.dnumero
Λd.matrigerente=e.matricula}

El operador OR permite realizar consultas equivalentes al operador "unión" del algebra relacional, un ejemplo de su uso se muestra en la consulta 4.

4. Hacer una lista de proyectos donde trabaje al menos un empleado cuyo apellido paterno sea López, siendo éste trabajador o gerente del departamento que controla el proyecto.

   {p.pnumero|PROYECTO (p) Λ ((∃e) (∃w)(EMPLEADO(e) Λ TRABAJAEN(w)
   Λ w.pno=p.pnúmero Λ e,paterno="López" Λ e,matricula=w.matriculaemp)
   V
   ((∃m) (∃d)(EMPLEADO(m) Λ DEPARTAMENTO(d) Λ p.dnum=d.dnumero Λ
   d.matrigerente=m.matricula Λ m.paterno=López"))}

El operador existencial negado nos permite realizar consulta equivalentes al operador "diferencia" del algebra relacional. Ejemplo de este operador se muestra en las consultas 5 y 6.

5. Dar la lista de los empleados que no son gerentes.

   {e.paterno, e.materno, e.nombre | EMPLEADO (e) Λ (∄)(d)DEPARTAMENTO (d) Λ e.matricula=d.matrigerente}

6. Dar el nombre de los empleados que no son supervisores.

   {e.paterno, e.materno, e.nombre | EMPLEADO (e) Λ (∄)(d)EMPLEADO(d) Λ e.matricula = d.superv}

El operador universal nos permite trabajar a todos los elementos de un conjunto dado que cumplen una condición específica, por ejemplo, ver consulta 7.

7. Dar el nombre de los empleados cuyo salario sea mayor al salario de todos los empleados que están adscritos al departamento No. 5.

   {e.paterno, e.materno, e.nombre | EMPLEADO (e) Λ ((∀x)EMPLEADO(x) Λ x.dno=5 Λ e.salario > x.salario)}

La implicación, en conjunto con el operador universal, es el equivalente a la división en algebra relacional. Como ejemplo del uso de la combinación de estos operadores se observa en la consulta 8.

8. Dar el nombre de todo empleado que trabaja en todo proyecto controlado por el departamento No. 5.

   {e.paterno, e.materno. e.nombre | EMPLEADO (e) Λ ((∀x)PROYECTO(x) Λ x.dno=5)
   → ((∃w)TRABAJAEN (w) Λ w.matriculaemp=e.matricula Λ x.pnumero=w.pno))}

   Sabemos que:

   $P \rightarrow Q \equiv \sim$P V Q

   Por lo tanto, la expresión queda de la siguiente forma:

   {e.paterno, e.materno. e.nombre | EMPLEADO (e) Λ (((∀x) ~ (PROYECTO(x) Λ x.dno=5) V (∃w)TRABAJAEN(w) Λ w.matriculaemp=e.matricula Λ x.pnumero = w.pno))}

   Utilizando las Leyes de De Morgan:

   ~(P Λ Q) $Q \equiv \sim$P V ~Q

   Se obtiene la siguiente expresión:

{e.paterno, e.materno. e.nombre | EMPLEADO (e) Λ (((∀x) ~ PROYECTO(x) V
~x.dno=5) V ((∃w)TRABAJAEN (w) Λ w.matriculaemp=e.matricula Λ x.pnumero = w.pno))}

También puede quedar como:

{e.paterno, e.nombre | EMPLEADO (e) F'}

Donde:

F' = (∀x) (~∀(PROYECTO(x)) V F_1)

F_1 = (~(x.dnum=5) V F_2)

F_2 = (∃w)(TRABAJAEN(w) Λ w.matriculaemp=e.matricula Λ x.pnumero = w.pno)

5.3 Preguntas de repaso.

1. Enumere los operadores del algebra relacional y el objetivo de cada uno de ellos.
2. Indique el objetivo de la unión compatibilidad.
3. Indique los operadores del cálculo relacional e indique semejanzas y diferencias del cálculo y álgebra relacional
4. Comente el significado del operador existencial y el cuantificador universal.
5. Indique el significado del operador de implicación
6. ¿Cuándo se dice que un lenguaje de consulta es relacionalmente completo?

6. SQL

(Elmasri & Navate, 2008)
(Groff & Weinberg, 1992)

6.1 INTRODUCCION.

El concepto de bases de datos relacional fue desarrollado originalmente por el Dr. Ted Codd, un investigador de IBM. En Junio de 1970, el Dr. Codd publicó un artículo titulado "Un modelo relacional de datos para grandes bancos de datos compartidos" (Codd, Junio, 1970) que esquematizaba una teoría matemática de cómo los datos podrían ser almacenados y manipulados utilizando una estructura tabular.

El artículo desencadenó una racha de investigaciones en bases de datos relacionales, incluyendo un importante proyecto dentro de IBM conocido como System/R.

En 1974 y 1975 la primera fase del proyecto produjo un prototipo mínimo de un DBMS relacional, incluyendo un trabajo sobre lenguajes de consulta de bases de datos. Uno de estos lenguajes se denominó SEQUEL, un acrónimo de Structured English Query Languaje. En 1976 y 1977 el prototipo fue reescrito desde el principio. La nueva implementación soportaba consultas multitabla y acceso de varios usuarios.

En 1978 y 1979 fueron distribuidos los primeros prototipos donde el lenguaje fue renombrado a SQL por razones legales.

La compañía Relational Software, Inc., fundada en 1977, lanzó al mercado su producto conocido como ORACLE, y se convirtió en el primer software relacional comercialmente disponible, adelantándose 2 años al primer producto de IBM. Además se ejecutaba en minicomputadoras VAX de Digital que eran más baratas que las Main Frame de IBM. En la actualidad, la empresa se renombró a Oracle Corporation.

Profesores de Berkeley de la U. de California construyeron un prototipo de un DBMS relacional, llamando a su sistema Ingres. Incluyó un lenguaje relacional conocido como QUEL. En 1980 fundaron Relational Technology, Inc. y construyeron la versión comercial en 1981. En 1989 fue renombrada como Ingres Corporation en 1989, cambiando QUEL por SQL.

SQL Data System (SQL/DS) fue anunciado en 1981 por IBM entregándose en 1982. En 1983, IBM anunció una versión una versión de SQL/DS para VM/CMS, en aplicaciones corporativas de centros de información.

En 1983, IBM anuncia DB2 para el S.O. MVS de IBM. Desde entonces DB2 es el principal producto de IBM y detrás de él, el lenguaje SQL.

Una desventaja de los productos relacionales con respecto a las otras arquitecturas fue el rendimiento. Excepto los productos IBM. Esto se debió porque los productos relacionales provenían de pequeños vendedores "novatos". Además de que las B.D's relacionales se ejecutaban en minis en lugar de Maxis IBM.

Una ventaja, su lenguaje de consulta ha permitido realizar consultas ad hoc a la base de datos, y obtener respuestas inmediatas, sin permitir el desarrollo de programación compleja.

Al paso de los años, el rendimiento de una base de datos relacional mejoró sustancialmente. En 1986 aparece el primer estándar ANSI/ISO para SQL. También emergió como estándar para sistemas UNIX. Al hacerse los computadores personales más potentes y al conectarse en redes de área local, comenzaron a necesitar una gestión

de bases de datos más sofisticados. Los vendedores de bases de datos para PC's adoptaron SQL.
Cuando la industria de la informática entró en los 90's, las instalaciones SQL se contaban por cientos de miles. SQL estaba claramente establecido como lenguaje estándar de bases de datos relacionales.
Los estándares existentes para SQL son:

ANSI American National Standars Institute
ISO International Standars Institute
SAA System Aplication Architecture
Es un estándar para hacer lo más compatible posible las cuatro familias de Computadoras incompatibles de IBM.
X/OPEN Estándar realizado por vendedores europeos para SQL bajo UNIX.
FIPS Federal Information Processing Standar

A pesar de la existencia de los estándares, ningún producto de SQL comercial disponible se conforma exactamente a él. Y como cada producto introduce nuevas capacidades, cada uno amplía su dialecto.

6.2 EL MITO DE LA PORTABILIDAD

Las insuficiencias en el estándar SQL ANSI/ISO y las diferencias entre los dialectos SQL son suficientemente significativas para que una aplicación deba ser siempre modificada cuando se pasa de una base de datos SQL a otra. Estas diferencias incluyen:

- Código de error. El estándar no especifica los códigos de error a devolver cuando SQL detecta un error.
- Tipos de datos. El estándar define un conjunto mínimo de tipos de datos, pero omite algunos de los más populares y útiles, como las fechas y horas.
- Tablas de sistema. El estándar no dice nada sobre las tablas de sistema que proporcionan información referente a la estructura de la propia B. D.'s. Cada vendedor tiene su propia estructura, e incluso las cuatro implementaciones de SQL de IBM se diferencian una de la otra.
- SQL interactivo. El estándar especifica el SQL programado utilizado por un programa de aplicación, y no el SQL interactivo.
- Interfaz de programa. El estándar utiliza una técnica abstracta para utilizar SQL desde dentro de un programa de aplicación escrito en COBOL, C, FORTRAN, etc., ningún producto SQL comercial utiliza esta técnica, y hay variaciones considerables en las interfaces reales de programas utilizados.
- SQL dinámico. El estándar no incluye las características para desarrollar herramientas como escritores de informes. Estas características conocidas como SQL dinámico existen en todos los sistemas de B. D.'s. SQL, pero varían significativamente de producto a producto.
- Diferencias semánticas. Es posible ejecutar la misma consulta en dos implementaciones SQL diferentes y producir resultados distintos. Estas diferencias ocurren en el manejo de NULL, las funciones de columna y la eliminación de filas duplicadas.

- Secuencia de cotejo. El estándar no define la secuencia de cotejo (ordenación). Los resultados de una consulta ordenada serán diferentes si la consulta se ejecuta en una computadora personal (código ASCII) y una mainframe (código EBCDIC).
- Estructura de B. D's. El estándar especifica el lenguaje SQL a utilizar una vez que una B. D's particular ha sido abierta. Los detalles de la denominación de la B. D.'s y cómo se establece la conexión inicial con la B. D.'s varía ampliamente y no es portable.

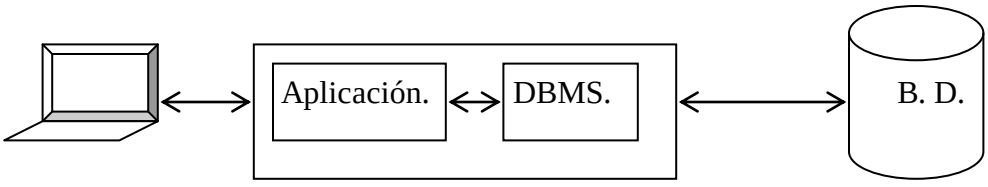
A pesar de estas diferencias, en 1989 comenzaron a emerger herramientas de B. D.'s comerciales que proclaman su portabilidad a través de diferentes productos de B. D.'s SQL. En todo caso las herramientas requieren un adaptador para cada DBMS para manejar conversión de tipos, traducir código de errores, etc. El día de la portabilidad transparente a través de diferentes productos DBMS basados en SQL estándar aún no llega.

6.3 SQL y la conexión por red.

La creciente popularidad de la conexión de computadoras por red durante los últimos años ha tenido un fuerte impacto en la gestión de bases de datos y ha dado a SQL una nueva prominencia. Conforme las redes pasan a ser más comunes, las aplicaciones que han corrido tradicionalmente en una minicomputadora o maxicomputadora tradicional se están transfiriendo a redes de área local con estaciones de trabajo de sobremesa y servidores. En estas redes SQL juega un papel crucial como vínculo entre una aplicación que corre en una estación de trabajo y el DBMS que gestiona los datos compartidos en el servidor.

6.3.1 Arquitectura centralizada.

La arquitectura de base de datos tradicional utilizada por DB2 y las B.D.'s sobre minis como Oracle e Ingres se muestran en la siguiente figura.



Sistema Computador.

Fig. 6.1 Gestión de bases de datos en una arquitectura centralizada.

En esta arquitectura el DBMS y los datos físicos residen ambos en un sistema maxi o minicomputador central, junto con el programa de aplicación esta arquitectura el DBMS y los datos físicos residen ambos en un sistema maxi o minicomputador central, junto con el programa de aplicación. Como el sistema es compartido por muchos usuarios, cada usuario experimenta una degradación del rendimiento cuando el sistema tiene una carga fuerte.

6.3.2 Arquitectura de servidor de archivos.

En esta arquitectura, una aplicación que corre en un computador personal puede acceder de forma transparente a datos localizados en un servidor de archivos, que almacena los archivos compartidos. Cuando una aplicación en PC solicita datos de un archivo compartido, el software de red recupera automáticamente el bloque solicitado del archivo en el servidor. En este caso, cada PC ejecuta su propia copia del software DBMS.

Esta arquitectura puede producir un fuerte tráfico de red y un bajo rendimiento para ciertas consultas.

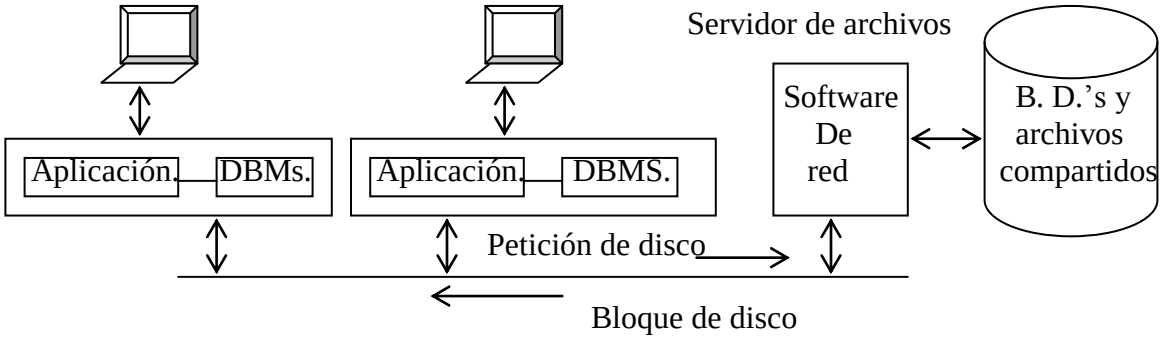


Fig. 6.2 Gestión de bases de datos en una arquitectura servidora de archivos.

6.3.3 Arquitectura cliente/servidor.

Las computadoras personales están combinadas en una red de área local junto con el servidor de base de datos que almacena las B. D.'s compartidas. Las funciones del DBMS están divididas en dos partes. Las "frontales" (front-end) de base de datos, tales como herramientas de consulta interactiva, escritores de informes y programas de aplicación, se ejecutaban en PC. La máquina de soporte (back-end) de la BD que almacena y gestiona los datos se ejecuta en el servidor. SQL se ha convertido en el lenguaje de BD estándar para comunicación entre las herramientas frontales y la máquina de soporte de esta arquitectura.

La arquitectura cliente/servidor reduce el tráfico de red y divide la carga de trabajo de la B. D.'s las funciones intensivas tales como visualización de datos se concentran en la PC. Las funciones intensivas en procesos de datos como el proceso de consultas se realiza en el servidor.

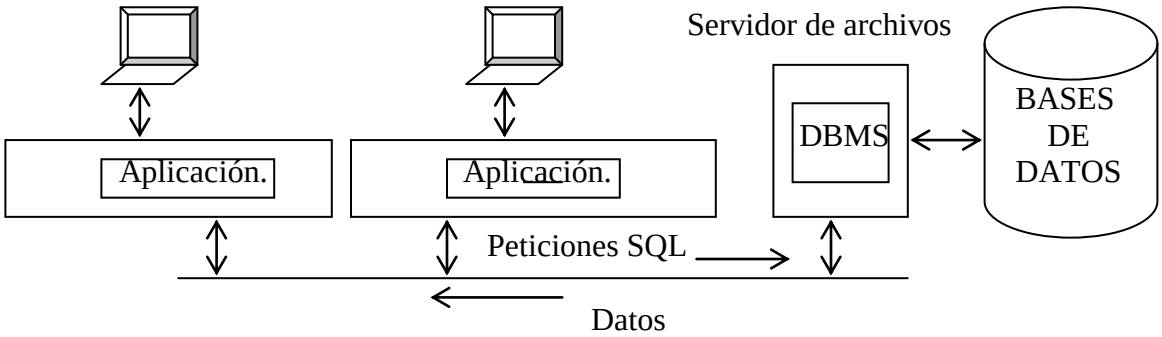


Fig. 6.3 Gestión de bases de datos en una arquitectura cliente/servidor.

6.4 El lenguaje de consultas.

SQL permite la creación de tablas, modificar tuplas y la realización de consultas basadas en álgebra relacional.
En esta sección se explicará la realización de consultas y se observarán sus resultados empleando el DBMS ACCESS 97.
SQL tiene una instrucción básica para obtener información de una base de datos: la instrucción SELECT (seleccionar). Esta instrucción no tiene que ver con la operación SELECCIONAR del álgebra relacional. La instrucción SELECT de SQL tiene muchas opciones y matices, por lo que se presentarán sus características gradualmente.
La forma básica de la instrucción SELECT, en ocasiones denominada transformación (mapping) o bloque SELECT FROM WHERE, consta de las tres cláusulas SELECT, FROM (de) y WHERE (donde) y se construye como:

SELECT <lista de atributos>
FROM <lista de tablas>
WHERE <condición>

Donde

- <lista de atributos> es una lista de nombres de los atributos cuyos valores va a obtener la consulta.
- <lista de tablas> es una lista de los nombres de las relaciones requeridas para procesar la consulta.
- <condición> es una expresión condicional (booleana) de búsqueda para identificar las tuplas que obtendrá la consulta.

Se mostrará paso a paso como realizar consultas en SQL empleando la base de datos COMPAÑIA cuya extensión se observa en la figura 4.5.

CONSULTA 1.
Obtener la fecha de nacimiento y la dirección del empleado cuyo nombre es Juan Perez Arellano

```
SELECT nacimiento, dirección
FROM empleado
WHERE materno = 'arellano' and paterno='perez' and nombre='juan'
```

En esta consulta sólo interviene la relación EMPLEADO en la cláusula FROM. La consulta selecciona las tuplas EMPLEADO que satisfagan la condición de la cláusula WHERE y luego proyecta el resultado sobre los atributos nacimiento y dirección listados en la cláusula SELECT.
Una consulta SQL simple con un solo nombre de relación en la cláusula FROM es similar a un par de operaciones SELECCIONAR-PROYECTAR del álgebra relacional. La cláusula SELECT de SQL especifica los atributos de proyección y la cláusula WHERE especifica la condición de selección. La única diferencia es que en la consulta SQL podemos obtener tuplas repetidas en el resultado, porque no se impone la restricción de que una relación sea un conjunto. La tabla 6.1 muestra el resultado de la consulta 1 con la base de datos COMPAÑIA.

nacimiento	dirección
9/1/55	sn diego

Tabla 6.1 Resultado de la consulta 1.

Consulta 2.
Obtener el nombre y la dirección d todos los empleados que pertenecen al departamento de investigación.

```
SELECT paterno, materno, direccion, dno
FROM empleado, departamento
WHERE dnombre='investigación' and dnumero=dno
```

En esta consulta se incluye la operación REUNION del álgebra relacional en la condición dnumero=dno. El resultado de la consulta 2 se presenta en la tabla 6.2. En general, es posible especificar cualquier cantidad de condiciones de selección y reunión en una sola consulta SQL. En la siguiente consulta existen dos condiciones de reunión.

paterno	materno	dirección	Dno
Perez	Arellano	sn diego	5
Lopez	Lopez	sn diego	5
Sanchez	mendez	sn miguel	5
Espana	English	sn diego	5

Tabla 6.2 Resultado de la consulta 2.

Consulta 3.

Para cada proyecto ubicado en sn diego, listar el número del proyecto, el número del departamento controlador y el nombre, dirección y fecha de nacimiento del gerente de ese departamento.

```
SELECT paterno, materno, nacimiento, nombreproy, dnumero
FROM empleado, departamento, proyecto
WHERE dnumero=proyecto.dno and plocalizacion='sn diego' and
        matrigerente=matricula
```

La condición de reunión dnumero=proyecto.dno relaciona un proyecto con su departamento controlador, en tanto que la condición de reunión matrigerente=matricula relaciona el departamento controlador con el empleado que lo dirige. En SQL se puede usar el mismo nombre para dos o más atributos siempre que éstos permanezcan en diferentes relaciones. Si este es el caso, y una consulta hace referencia a dos o más atributos del mismo nombre, es preciso calificar el nombre del atributo con el nombre de la relación, a fin de evitar la ambigüedad. Esto se hace anteponiendo el nombre de la relación al nombre del atributo y separando los dos con un punto. En la consulta 3, el atributo del número de departamento tiene el mismo nombre en la tabla empleado y en la

tabla proyecto, y ambos se localizan en la cláusula FROM por lo que se debe de especificar cuál de los dos atributos se deben usar. El resultado de la consulta 3 se muestra en la tabla 6.3.

Paterno	materno	nacimiento	Nombrepro	dnumero
Lopez	Lopez	8/12/45	producto3	5
Bond	Bond	10/10/27	organizacion	1

Tabla 6.3 Resultado de la consulta 3.

También puede haber ambigüedad en consultas que hagan referencia dos veces a la misma relación, como en el siguiente ejemplo.

Consulta 4

Para cada empleado, obtener su nombre de pila y apellido y el nombre de pila y apellido de su supervisor inmediato.

```
SELECT e.nombre, e.paterno, s.nombre, s.paterno
FROM empleado e, empleado s
WHERE e.supervisor=s.matricula
```

En este caso, podemos declarar los nombres de relación alternativos E y S, llamados seudónimos, para la relación EMPLEADO. Podemos pensar que E y S son dos copias distintas de la relación empleado; la primera, E, representa empleados en el papel de supervisados, y la segunda, S, representa empleados en el papel de supervisores. Luego, podemos aplicar una REUNION a las dos copias. En realidad sólo hay una relación EMPLEADO, y la condición de reunión reúne la relación consigo misma encontrando las tuplas que satisfacen la condición de reunión e.supervisor=s.matricula. Obsérvese que éste es un ejemplo de consulta recursiva de un nivel. El resultado de la consulta 4 se muestra en la tabla 6.4.

e.nombre	e.paterno	s.nombre	s.paterno
Juan	perez	Francisco	lopez
Francisco	lopez	James	bond
Alicia	zelaya	Jennifer	ortega
Jennifer	ortega	James	bond
Ramses	sanchez	Francisco	lopez
Josefa	espana	Francisco	lopez
Aldama	hidalgo	Jennifer	ortega

Tabla 6.4 Resultado de la consulta 4.

Consulta 5

Seleccionar todas las matriculas de empleado

SELECT matricula
FROM empleado

La omisión de la cláusula WHERE indica una selección de tuplas incondicional; por tanto, todas las tuplas de la relación especificada en la cláusula FROM son aceptadas y seleccionadas para el resultado de la consulta. Esto equivale a la condición WHERE TRUE, que significa todas las filas de la tabla. El resultado de la consulta 5 se muestra en la tabla 6.5.

Matricula
00007
45555
53453
54321
56789
84444
87777
87987

Tabla 6.5 resultado de la consulta 5.

Consulta 6.
Seleccionar las combinaciones de todas las matrículas y nombre de departamento de la base de datos.
SELECT matricula, dnombre
FROM empleado, departamento

Si se especifica más de una relación en la cláusula FROM y no hay cláusula WHERE, se selecciona el PRODUCTO CRUZADO –todas las posibles combinaciones de tuplas- de esas relaciones. El resultado de la consulta 6 se muestra en la tabla 6.6.

matricula	dnombre
56789	investigacion
56789	administracion
56789	prod. Alimentos
45555	investigacion
45555	administracion
45555	prod. Alimentos
87777	investigacion
87777	administracion
87777	prod. Alimentos
54321	investigacion
54321	administracion
54321	prod. Alimentos
84444	investigacion
84444	administracion
84444	prod. Alimentos
53453	investigacion

53453	administracion
53453	prod. Alimentos
87987	investigacion
87987	administracion
87987	prod. Alimentos
00007	investigacion
00007	administracion
00007	prod. Alimentos

Tabla 6.6 Resultado de la consulta 6.

Consulta 7.
Se desean observar todos los atributos de los departamentos y su localización

```
SELECT *
FROM departamento, localizacion
WHERE departamento.dnumero=localizacion.dnumero
```

Si queremos obtener los valores de todos los atributos de las tuplas seleccionadas, no tenemos que listar los nombres de los atributos explícitamente en SQL; basta con especificar un asterisco (*), que significa *todos los atributos*. Por ejemplo, la consulta 7 obtiene los valores de todos los atributos de los departamentos y su localización. El resultado se muestra en la tabla 6.7

dep	dnombre	departa	matr	fechaing	localizaci	localizacio	Dlocalizacio
1	investigacion	5	4555	5/22/78	3	5	atenco
1	investigacion	5	4555	5/22/78	5	5	sn diego
1	investigacion	5	4555	5/22/78	4	5	sn felipe
2	administracio	4	5432	1/1/85	2	4	papalotla
3	prod.	1	0000	5/19/71	1	1	sn diego

Tabla 6.7 Respuesta de la consulta 7.

SQL no trata las relaciones, en general, como conjuntos; puede haber tuplas repetidas en el resultado de una consulta o en la relación. SQL no elimina automáticamente las tuplas repetidas en los resultados de las consultas por las siguientes razones:

- La eliminación de duplicados es una operación costosa. Una forma de implementarla es ordenar las tuplas primero y luego eliminar los duplicados.
- Es posible que el usuario desee ver las tuplas repetidas en el resultado de una consulta.
- Cuando se aplica una función agregada a tuplas, en la mayoría de los casos no queremos eliminar los duplicados.

Consulta 8.
Obtener el salario de todos los empleados.

```
SELECT salario
FROM empleado
```

Como se observará en la tabla 6.8, si existen salarios repetidos, se repetirán.

salario
9000
4000
2500
4300
3000
3800
2500
2500

Tabla 6.8 Resultado de la consulta 8.

Si lo que se desea es eliminar las tuplas repetidas del resultado de una consulta SQL, se puede usar la palabra reservada DISTINCT en la cláusula SELECT, para indicar que sólo deben conservarse tuplas distintas en el resultado.

Consulta 9
Se desea el salario no repetido de todos los empleados.

```
SELECT distinct salario
FROM empleado
```

En este caso, cada salario diferente sólo se observará una vez. Esto se observa en la tabla 6.9.

Salario
2500
3000
3800
4000
4300
9000

Tabla 6.9 Resultado de la consulta 7.9

Algunas de las operaciones de conjuntos del álgebra relacional se han incorporado directamente en SQL. Hay una operación de unión de conjuntos (UNION, y en SQL2 hay también operaciones de diferencia de conjuntos (EXCEPT) y de intersección de conjuntos (INTERSECT). Las relaciones resultantes de estas operaciones son conjuntos de tuplas; es decir, las tuplas repetidas se eliminan del resultado (a menos que la operación vaya seguida de la palabra reservada ALL). Como las operaciones de conjuntos sólo se aplican a relaciones unión compatibles, debemos asegurarnos de que las dos relaciones a las que apliquemos la operación tengan los mismos atributos y de que éstos aparezcan en el mismo orden en ambas relaciones. El siguiente ejemplo ilustra el empleo de UNION.

Consulta 10
Preparar una lista con todos los números de los proyectos en los que participa un empleado de apellido bond, sea como trabajador o como gerente del departamento que controla el proyecto.

```
(SELECT pnumero, matricula
FROM proyecto, departamento, empleado
WHERE proyecto.dno=dnumero and matrigerente=matricula and materno='bond')
UNION
(SELECT pnumero, matricula
FROM proyecto, trabajaen, empleado
WHERE pnumero=pno and matriculaemp=matricula and materno='bond')
```

La primera consulta SELECT obtiene los proyectos en los que participa bond como gerente del departamento que controla el proyecto, y la segunda obtiene los proyectos en los que participa bond como trabajador. El resultado se da en la tabla 6.10

pnumero	matricula
20	00007

Tabla 6.10

En algunas consultas es preciso obtener valores existentes en la base de datos para usarlos en una condición de comparación. Una forma cómoda de formular tales consultas es mediante consultas anidadas, que son consultas SELECT completas dentro de la cláusula WHERE de otra consulta, la cual se denomina consulta exterior. La consulta 10 se puede reformar como consulta anidada como se muestra enseguida.

Consulta 11
Misma consulta que la 10.

```
SELECT  distinct pnumero
FROM proyecto
WHERE pnumero in (SELECT pnumero
                  FROM proyecto,  departamento, empleado
                  WHERE proyecto.dno=dnumero and matrigerente=matricula
                        and paterno='bond')
    or
    pnumero in (SELECT pno
                  FROM  trabajaen, empleado
                  WHERE paterno='bond' and matriculaemp=matricula)
```

La primer consulta anidada selecciona los números de los proyectos en que un bond participa como gerente, en la segunda selecciona los números de proyectos en que un bond participa como trabajador. En la consulta exterior, seleccionamos una tupla PROYECTO si el valor de pnumero de esa tupla está en el resultado de cualquiera de las dos consultas anidadas. El operador de comparación IN compara un valor *v* con un

conjunto (o multiconjunto) de valores V y produce TRUE si *v* es uno de los elementos de V. La tabla 6.11 muestra el resultado de la consulta.

Pnumero
20

Tabla 6.11

Además del operador IN, podemos usar varios otros operadores de comparación para comparar un valor individual *v* (por lo regular un nombre de atributo) con un conjunto V (por lo regular un consulta anidada). El operador =ANY (o =SOME) devuelve TRUE si el valor *v* es igual a algún valor del conjunto V, y por tanto equivale a IN. Las palabras reservadas ANY y SOME tienen el mismo significado. Otros operadores que se pueden combinar con ANY y SOME incluyen >, >=, <, <=, y <>. También es posible combinar la palabra reservada ALL con uno de estos operadores. Por ejemplo, la condición de comparación (*v* > ALL V) devuelve TRUE si el valor *v* es mayor que todos los valores del conjunto V.

Consulta 12.
Mostrar todo empleado cuyo salario es mayor que el de todos los empleados del departamento 5.

```
SELECT paterno, nombre
FROM empleado
WHERE salario > all (SELECT salario
                     FROM empleado
                     WHERE dno= '5')
```

paterno	nombre
ortega	Jennifer
bond	James

Tabla 6.12 Resultado de la consulta 12.

En general, podemos tener varios niveles de consulta anidados. Una vez más nos enfrentamos a posibles ambigüedades en los nombres de los atributos si hay atributos con el mismo nombre: uno en la relación listada en la cláusula FROM de la consulta exterior y el otro en una relación listada en la cláusula FROM de la consulta anidada. La forma de manejar atributos con mismo nombre en diferentes niveles de anidamiento se pueden manejar como las variables globales y locales en un lenguaje como pascal, que permite anidar procedimientos y funciones. Para ilustrar la ambigüedad de los nombres de atributos en las consultas anidadas, consideremos la consulta 13, cuyo resultado se muestra en la tabla 6.13

Consulta 13.
Obtener el nombre de todos los empleados que tienen un dependiente con el mismo nombre de pila que el empleado.

```
SELECT e.paterno, e.nombre
FROM empleado e
WHERE e.matricula in (SELECT empleadomat
                      FROM dependientes
                      WHERE empleadomat=matricula and
                            nombre=nombredependiente and
                            sexo=e.sexo);
```

paterno	nombre
perez	juan

Tabla 6.13. Resultado de la consulta 13.

En la consulta anidada 13, debemos calificar e.sexo porque se refiere al atributo SEXO del EMPLEADO de la consulta exterior, y DEPENDIENTE también tiene un atributo llamado SEXO. Todas las referencias no calificadas a SEXO en la consulta anidada se refieren a SEXO de dependiente.

Siempre que una condición en la cláusula WHERE de una consulta anidada hace referencia a un atributo de una relación declarada en la consulta exterior, se dice que las dos consultas están correlacionadas.

En general, una consulta escrita con bloques SELECT… FROM… WHERE anidados y que emplee los operadores de comparación = o IN siempre puede expresarse como una consulta de un solo bloque. Por ejemplo, la consulta 13 puede escribirse como sigue:

Consulta 14

Responde la consulta 13 sin necesidad de anidamiento.

```
SELECT e.paterno, e.nombre
FROM empleado e, dependientes d
WHERE e.matricula=d.empleadomat and e.sexo=d.sexo
      and e.nombre=d.nombredependiente
```

paterno	nombre
perez	juan

Tabla 6.14. Resultado de la consulta 14

La función de SQL EXISTS sirve para comprobar si el resultado de una consulta anidada correlacionada está vacío. La consulta 15 muestra cómo se responde la consulta 13 empleando la función EXIST.

Consulta 15.

Responde la consulta 13 empleando la función EXISTS.

```
SELECT e.paterno, e.nombre
FROM empleado e
WHERE EXISTS (SELECT *
              FROM dependientes
              WHERE matricula=empleadomat and sexo=e.sexo
                    and nombre=nombredependiente)
```

paterno	Nombre
perez	Juan

Tabla 6.15. Resultado de la consulta 15.

EXISTS y NOT EXISTS casi siempre se usan junto con una consulta anidada correlacionada. En general, EXISTS(Q) devuelve TRUE si hay por lo menos una tupla en el resultado de la consulta Q, y devuelve FALSE en caso contrario. En la consulta 16 se muestra un ejemplo de la función NOT EXISTS.

Consulta 16
Obtener los nombres de los empleados que no tienen dependientes.

```
SELECT nombre, paterno
FROM empleado
WHERE not exists (SELECT *
                  FROM dependientes
                  WHERE empleadomat=matricula)
```

nombre	Paterno
alicia	Zelaya
ramses	Sanchez
josefa	Espana
aldama	Hidalgo
james	Bond

Tabla 6.16. Resultado de la consulta 16.

En la consulta anidada correlacionada 16 se obtienen todas las tuplas DEPENDIENTE relacionadas con una tupla EMPLEADO. Si no existe ninguna, se selecciona la tupla EMPLEADO. En la consulta 17 se muestra otro ejemplo de la función EXISTS combinada con el operador lógico AND.

Consulta 17.
Lista los nombres de los gerentes que tienen por lo menos un dependiente.

```
SELECT nombre, paterno
FROM empleado
WHERE  exists (SELECT *
               FROM dependientes
               WHERE empleadomat=matricula)
       and
       exists (SELECT *
               FROM departamento
               WHERE matricula=matrigerente)
```

nombre	Paterno
francisco	Lopez
jennifer	Ortega

Tabla 6.17. Resultado de la consulta 17.

La división del álgebra relacional se puede implementar por medio de la función CONTAINS que existe en la implementación original de SQL en SYSTEM R. Este operador se eliminó posteriormente del lenguaje debido a la dificultad para implementarlo de forma eficiente. En la actualidad, la mayoría de las implementaciones comerciales de SQL no cuenta con dicho operador, el cual compara dos conjuntos de valores y devuelve TRUE si un conjunto contiene todos los valores del otro. El operador CONTAINS se puede expresar usando EXISTS y NOT EXISTS en los sistemas que no tienen el operador CONTAINS.

Consulta 18
Obtener el nombre de todos los empleados que trabajan en todos los proyectos controlados por el departamento número 5.

```
SELECT nombre, paterno
FROM empleado
WHERE  not exists (SELECT *
                   FROM trabajaen b
                   WHERE (b.pno in (SELECT pnumero
                                    FROM proyecto
                                    WHERE DNO='5'))
                   and
                   not exists
                   (SELECT *
                    FROM trabajaen c
                    WHERE c.matriculaemp=matricula and
                          c.pno=b.pno))
```

nombre	paterno
juan	perez
josefa	espana

Tabla 6.18. Resultado del la consulta 18.

En 18, la consulta anidada exterior selecciona todas las tuplas trabajaen (B) cuyo número de proyecto sea el de un proyecto controlado por el departamento 5, si es que no hay una tupla trabajaen (c) con el mismo número de proyecto y con el mismo número de matrícula que la tupla EMPLEADO considerada en la consulta exterior. Si no existe ninguna tupla así, seleccionamos la tupla EMPLEADO. La consulta 18 selecciona todos los empleados tales que no exista un proyecto controlado por el departamento 5 en le cual no trabaje el empleado.

Cabe señalar que en el álgebra relacional la consulta 18 requiere un tipo de cuantificador que en el cálculo relacional se denomina cuantificador universal. El cuantificador existencial negado NOT EXISTS puede servir para expresar una consulta cuantificada universalmente.

En SQL también es posible usar un conjunto explícito de valores en dichas consultas, en vez de una consulta anidada. Los Conjuntos de este tipo se encierran entre paréntesis

Consulta 19

Obtener el número de seguro social de todos los empleados que trabajan en los proyectos 1, 2 o 3

```
SELECT distinct matriculaemp
FROM trabajaen
WHERE pno in ('1','2','3')
```

matriculae
45555
53453
56789
84444

Tabla 6.19. Resultado de la consulta 19.

El valor NULO se maneja con el operador IS o IS NOT, en vez de usar = o <>. La razón es que SQL considera que cada valor nulo es distinto de los demás valores nulos, por lo que la comparación de igualdad no es apropiada. De esto se sigue que, cuando se especifica una condición de búsqueda de reunión (comparación), las tuplas con valores nulos en el atributo de reunión no se incluirán en el resultado. La consulta 20 ilustra lo anterior. (En ACCESS 97 se permite en forma indistinta el manejo del valor nulo con el operador IS o el operador =).

Consulta 20

Obtener los nombres de todos los empleados que no tienen supervisores.

```
SELECT paterno, nombre
FROM empleado
WHERE supervisor is null
```

paterno	nombre
bond	james

Tabla 6.20. Respuesta de la consulta 20.

Es posible cambiar el nombre de cualquier atributo que aparezca en el resultado de una consulta si se añade el calificador AS seguido del nuevo nombre. La construcción AS servirá para declarar seudónimos tanto de atributos como de relaciones. Se reescribirá la consulta 4 para mostrar el uso de seudónimos.

Consulta 4A. Misma a la consulta 4.

```
SELECT e.paterno as apellido_empleado, s.paterno as apellido_supervisor
FROM empleado as e, empleado as s
WHERE e.supervisor = s.matricula
```

apellido_empleado	apellido_supervisor
perez	lopez
lopez	bond
zelaya	ortega
ortega	bond
sanchez	lopez
espana	lopez
hidalgo	ortega

Tabla 6.4A. Resultado de la consulta 4A.

El concepto de tabla reunida (o relación reunida) se incorporó a SQL2 para que los usuarios pudieran especificar una tabla resultante de una operación de reunión de la cláusula FROM de una consulta. Quizá sea más fácil comprender esta construcción, en vez de mezclar todas las condiciones de selección y de reunión en la cláusula WHERE. Por ejemplo, consideremos la consulta, que obtiene el nombre y la dirección de todos los empleados que trabajan para el departamento de 'investigación'. Para algunos usuarios, podría ser más fácil especificar primero la reunión de la relación EMPLEADO y DEPARTAMENTO, y luego seleccionar las tuplas y atributos deseados. Ver consulta 2A.

Consulta 2A.
Misma que consulta 2 usando el operador INNER JOIN.

```
SELECT paterno, nombre, dnombre
FROM empleado  e  INNER JOIN departamento d ON e.dno=d.dnumero
WHERE dnombre='investigacion'
```

paterno	nombre	dnombre
Perez	juan	investigacio
Lopez	francisco	investigacio
sanchez	ramses	investigacio
espana	josefa	investigacio

Tabla .6.2.A. Resultado de la consulta 2.A

En una operación de reunión por omisión es la reunión interna, en la que sólo se incluye una tupla en el resultado si existe una tupla que coincida con una de la otra relación. Por ejemplo, en la consulta 4A, sólo aparecen en el resultado los empleados que tienen un supervisor; si una tupla EMPLEADO tienen NULL como valor en SUPERVISOR quedará excluida. Si el usuario requiere la inclusión de todos los empleados, deberá usar una reunión externa explícita. En SQL2 esto se maneja especificando explícitamente la reunión externa en una tabla reunida, como se ilustra en la consulta 4B.

Consulta 4B.
Misma que la consulta 4 usando el operador LEFT JOIN.

```
SELECT e.paterno as apellido_empleado, s.paterno as apellido_supervisor
FROM empleado e LEFT JOIN empleado s ON e.supervisor=s.matricula
```

Apellido_e	apellido_su
Perez	lopez
Lopez	bond
Zelaya	ortega
Ortega	bond
Sanchez	lopez
Espana	lopez
Hidalgo	ortega
Bond	

Tabla 6.4B Resultado de la consulta 4B.

En este caso, LEFT JOIN empata la tabla izquierda del operador con la derecha indicada en el operador aún si la parte derecha no empata con la parte izquierda. RIGHT JOIN es equivalente con LEFT JOIN con la diferencia en que el lado izquierdo no empata con el derecho. Observe la consulta

Consulta 21
Dé el nombre de todo empleado que sea en este momento gerente de algún departamento, su número de departamento y fecha en que tomó el puesto.

```
SELECT dnombre, fechaing, paterno, nombre
FROM departamento RIGHT JOIN EMPLEADO ON
empleado.matricula=departamento.matrigerente;
```

dnombre	fechaing	Paterno	nombre
		Perez	Juan
Investigacio	5/22/78	Lopez	francisco
		Zelaya	Alicia
Administraci	1/1/85	Ortega	Jennifer
		Sanchez	Ramses
		Espana	Josefa
		Hidalgo	Aldama
prod.	5/19/71	Bond	James

Tabla 6.21. Resultado de la consulta 21.

También es posible anidar las especificaciones de reunión; es decir, una de las tablas de una reunión puede ser ella misma una tabla de reunión; es decir, una de las tablas de reunión puede ser ella misma una tabla reunida. Esto se ilustra en la consulta 3A

Consulta 3.A
Misma que la consulta 3 empleando el operador JOIN anidado.

```
SELECT pnumero,proyecto.dno,paterno,direccion,nacimiento
FROM (proyecto INNER JOIN departamento on
proyecto.dno=departamento.dnumero)
INNER JOIN empleado on departamento.matrigerente=empleado.matricula
WHERE plocalizacion='san diego'
```

pnumero	dno	paterno	direccion	nacimiento
3	5	lopez	sn diego	8/12/45
20	1	bond	sn diego	10/10/27

Tabla 6.3A. Resultado de la consulta 3A.

SQL también tiene funciones de agrupación y agregación. La función COUNT devuelve el número de tuplas o valores especificados en una consulta. Las funciones SUM, MAX, MIN y AVG se aplican a un conjunto o multiconjunto de valores numéricos y devuelven, respectivamente, la suma, el valor máximo, el valor mínimo y el promedio de esos valores. Estas funciones se pueden usar en la cláusula SELECT o en una cláusula HAVING (que se verá enseguida). Se mostrarán estas funciones en las siguientes consultas.

Consulta 22.

Obtener la suma de los salarios de todos los empleados, el salario máximo, el salario mínimo y el salario promedio.

```
SELECT SUM(salario) as suma_salario, MAX (salario) as salario_máx,
MIN(salario) as salario_min, AVG(salario) as salario_promedio
FROM empleado
```

suma_salar	salario_má	salario_min	salario_promedi
31600	9000	2500	3950

Tabla 6.22 Respuesta a la consulta 22.

Consulta 23.
Hallar la suma de los salarios de todos los empleados del departamento 'investigacion', así como el salario máximo y el salario promedio de dicho departamento

```
SELECT SUM(salario) as suma_salario, MAX (salario) as salario_máx,
MIN(salario) as salario_min, AVG(salario) as salario_promedio
FROM empleado, departamento
WHERE dnumero=dno and dnombre='investigacion'
```

suma_salar	salario_má	salario_min	salario_promedi
13300	4000	2500	3325

Tabla 6.23 Resultado de la consulta 23.

Consulta 24
Obtener el número de empleados del departamento de 'investigacion'.

```
SELECT count(*) as empleados_departamento_investigacion
FROM empleado, departamento
WHERE dnumero=dno and dnombre='investigacion'
```

empleados_departamento_investigacion
4

Tabla 6.24 Resultado de la consulta 24.

Aquí el asterisco (*) se refiere a las tuplas, así que COUNT(*) devuelve el número de filas en el resultado de la consulta.

En algunos casos puede ser que necesitemos funciones para seleccionar tuplas específicas. En tales casos, especificaremos una consulta anidada correlacionada con la función deseada, y usaremos esa consulta en la cláusula WHERE de una consulta exterior.

Consulta 25
Se desea obtener el nombre de todos los empleados que tienen dos o más dependientes.

```
SELECT paterno, nombre
FROM empleado
WHERE (SELECT count(*)
        FROM dependientes
        WHERE matricula=empleadomat)>=2
```

paterno	nombre
perez	juan
lopez	francisco

Tabla 6.25 Resultado de la consulta 25.

En algunos casos nos interesa aplicar las funciones agregadas a subgrupos de tuplas de una relación, con base en los valores de algunos atributos. A estos atributos los llamamos atributos de agrupación, y se aplica la función de agregación en forma independiente cada uno de los grupos formados. SQL cuenta con la cláusula GROUP BY para este fin. La cláusula GROUP BY especifica los atributos de agrupación, que también deberán aparecer en la cláusula SELECT, para que el valor que resulte de aplicar cada función a un grupo de tuplas aparezca junto con el valor de los atributos de agrupación.

Consulta 26.
Para cada departamento, obtener el número de departamento, el número de empleados del departamento y su salario promedio.

```
SELECT dno, count(*), avg(salario)
FROM empleado
GROUP BY dno
```

Dno	Expr1001	Expr1002
1	1	9000
4	3	3100
5	4	3325

Tabla 6.26 Resultado de la consulta 26.

Consulta 27
Para cada proyecto, obtener el número y el nombre de proyecto, así como el número de empleados que trabajan en él.

```
SELECT pnumero, nombreproy, count(*)
FROM proyecto, trabajaen
WHERE pno=pnumero
GROUP BY pnumero, nombreproy
```

pnumero	nombrepro	Expr1002
1	producto1	2
10	automatizcio	3
2	producto2	3
20	organizacion	3
3	producto3	4
30	publicidad	3

Tabla 6.27 Resultado de la consulta 27.

La cláusula HAVING especifica una condición en términos del grupo de tuplas asociado a cada valor de los atributos de agrupación; sólo los grupos que satisfagan la condición estarán en el resultado de la consulta.

CONSULTA 28
Para cada proyecto en el que trabajan más de dos empleados, obtener el número y nombre del proyecto, así como el número de empleados que trabajan en él.

```
SELECT pnumero, nombreproy, count(*) as numero_de_empleados
FROM proyecto, trabajaen
WHERE pno=pnumero
GROUP BY pnumero, nombreproy
HAVING count(*)>2
```

Pnumero	nombrepro	numero_de_empleados
10	automatizcio	3
2	producto2	3
20	organizacion	3
3	producto3	4
30	publicidad	3

Tabla 6.28 Resultado de la consulta 28

Consulta 29

Para cada proyecto, obtener el número y el nombre de proyecto, así como el número de empleados del departamento 5 que trabajan en el proyecto.

```
SELECT pnumero, nombreproy, count(*) as numero_de_empleados
FROM proyecto, trabajaen, empleado
WHERE pno=pnumero and matriculaemp=matricula and empleado.dno='5'
GROUP BY pnumero, nombreproy
```

pnumero	nombrepro	numero_de_empleados
1	producto1	2
10	automatizcio	1
2	producto2	3
20	organizacion	1
3	producto3	4

Tabla 6.29 Resultado de la consulta 29.

Debemos tener especial cuidado cuando se aplican dos condiciones diferentes (una a la función de la cláusula SELECT y otra a la función de la cláusula HAVING). Por ejemplo, suponga que se desea contar el número total de empleados cuyos salarios rebasan los $3,000.00 en cada departamento, pero sólo en el caso de departamentos en los que trabajan más de cinco empleados. Aquí la condición (Salario>3000) se aplica sólo a la función COUNT de la cláusula SELECT. Si realizamos la siguiente consulta en SQL en forma incorrecta:

Consulta 30

Para cada departamento que tenga cuatro empleados o más, obtener el nombre del departamento y el número de empleados que ganan más de $3,000.00.

```
SELECT dnombre, count(*)
FROM departamento, empleado
WHERE dnumero=dno and salario>3000
GROUP BY dnombre
HAVING count(*)>=4
```

Esta consulta da como resultado una tabla en blanco.

La consulta en SQL no es correcta porque seleccionará sólo los departamentos que tengan cuatro empleados o más que ganen más de $3,000.00. La regla es que la cláusula WHERE se ejecuta primero, para seleccionar tuplas individuales; la cláusula HAVING se aplica después, para seleccionar grupos individuales de tuplas. Por tanto, las tuplas ya están restringidas a los empleados que ganan más de $ 3,000.00 antes de aplicarse la función de la cláusula HAVING. Una forma de escribir la consulta correctamente es con una consulta anidada, como en la consulta 30A.

Consulta 30A.
Misma que la consulta 30.

```
SELECT dnombre, count(*)
FROM departamento, empleado
WHERE dnumero=dno and salario>3000 and
        dno in (SELECT dno
                FROM empleado
                GROUP BY dno
                HAVING count(*)>=4)
GROUP BY dnombre;
```

dnombre	Expr1001
Investigacio	2

Tabla 6.30A. Respuesta de la consulta 30.

SQL tiene cierta facilidad para la comparación de una cadena de caracteres. Un operador que lo permite es el operador LIKE. Las cadenas parciales se especifican mediante dos caracteres reservados: '*' sustituye a un número arbitrario de caracteres y '?' sustituye a un solo carácter arbitrario.

Consulta 31.

Obtener todos los empleados cuya dirección este en un lugar cuya parte del nombre es diego.

```
SELECT nombre, paterno
FROM empleado
WHERE direccion LIKE '*diego'
```

nombre	paterno
juan	perez
francisco	lopez
josefa	espana
aldama	hidalgo
james	bond

Tabla 6.31. Resultado de la consulta 31.

Consulta 32.
Encontrar todos los empleados que nacieron en la década de los 50's.

```
SELECT nombre, paterno, nacimiento
FROM empleado
WHERE nacimiento LIKE '?????5*'
```

nombre	paterno	nacimiento
juan	perez	9/1/55
alicia	zelaya	7/19/58
ramses	sanchez	9/15/52
aldama	hidalgo	3/29/59

Tabla 6.32 Resultado de la consulta 32.

Otra característica es la posibilidad de usar aritmética en las consultas. Los operadores aritméticos estándar +, -, *, /, se pueden aplicar a valores numéricos en una consulta

Consulta 33.
Mostrar los salarios resultantes si cada empleado que trabaja en el proyecto Producto3 recibe un aumento del 10%.

```
SELECT paterno, nombre, 1.1*salario as aumento_salario_10_porciento
FROM empleado, trabajaen, proyecto
WHERE matriculaemp=matricula and pnumero=pno and
nombreproy='producto3'
```

paterno	nombre	aumento_salario_10_porcie
sanchez	ramses	4180
lopez	francisco	4400
perez	Juan	3300
espana	josefa	2750

Tabla 6.33 Resultado de la consulta 33.

SQL permite al usuario ordenar las tuplas del resultado de una consulta según sus valores de uno o más atributos, empleando la cláusula ORDER BY.

Consulta 34.

Obtener una lista de empleados y de proyectos en los que trabajan, ordenados por departamento que controla cada proyecto (el departamento será ordenado en forma descendente) y, en cada departamento los empleados serán ordenados en forma ascendente, por cada empleado los números de proyecto se pide en forma descendente.

```
SELECT distinct empleado.dno, paterno, pno
FROM proyecto, departamento, trabajaen, empleado
WHERE pnumero=pno and matriculaemp=matricula
        and empleado.dno=proyecto.dno
ORDER BY empleado.dno desc, paterno asc, pno desc
```

dno	paterno	pno
5	espana	3
5	espana	2
5	espana	1
5	lopez	3
5	lopez	2
5	perez	3
5	perez	2
5	perez	1
5	sanchez	3
4	hidalgo	30
4	hidalgo	10
4	ortega	30
4	zelaya	30
4	zelaya	10
1	bond	20

Tabla 6.34 Resultado de la consulta 34.

6.5. Análisis

Una consulta en SQL puede constar de un máximo de seis cláusulas, pero sólo son obligatorias las dos primeras, SELECT y FROM. Las cláusulas s especifican en el siguiente orden (las que están entre corchetes son opcionales).

```
SELECT <lista de atributos>
FROM <lista de tablas>
[WHERE <condición>]
[GROUP BY <atributo(s) de agrupación>]
[HAVING <condición de agrupación>]
[ORDER BY <lista de atributos>]
```

Las consultas se evalúan aplicando primero la cláusula FROM, seguida de la cláusula WHERE, y luego GROUP BY y HAVING. Si no se especifica ninguna de las últimas tres cláusulas (GROUP BY, HAVING, ORDER BY), podemos considerar que la

consulta se ejecuta de la siguiente manera: para cada combinación de tuplas – una de cada una de las relaciones especificadas en la cláusula FROM –evaluar la cláusula WHERE; si el resultado es TRUE, colocar en el resultado de la consulta los valores de los atributos de esta combinación de tuplas que están especificados en la cláusula SELECT. Desde luego, ésta no es una forma eficiente de implementar la consulta, y cada DBMS tiene rutinas especiales de optimización de consultas para elegir un plan de ejecución.
En general, hay muchas maneras de especificar la misma consulta en SQL. Esta flexibilidad ofrece ventajas y desventajas. La principal ventaja es que el usuario puede escoger la técnica que le resulte más cómoda para especificar una consulta. En general es preferible escribir las consultas con el mínimo de anidamiento y de ordenamiento implícito que sea posible.

La desventaja de contar con muchas formas de especificar la misma consulta es que ello puede confundir al usuario, quien tal vez no sabrá cuál técnica usar para especificar ciertos tipos de consultas.

Otro problema es que puede ser más eficiente ejecutar una consulta especificada de una manera que ejecutar la misma consulta especificada de otro modo.

6.5 EJERCICIO.

Se tiene el siguiente requerimiento, su modelo conceptual, su modelo de implementación, su extensión y 68 consultas. Contestar las consultas en algebra relacional, en cálculo relacional y en SQL

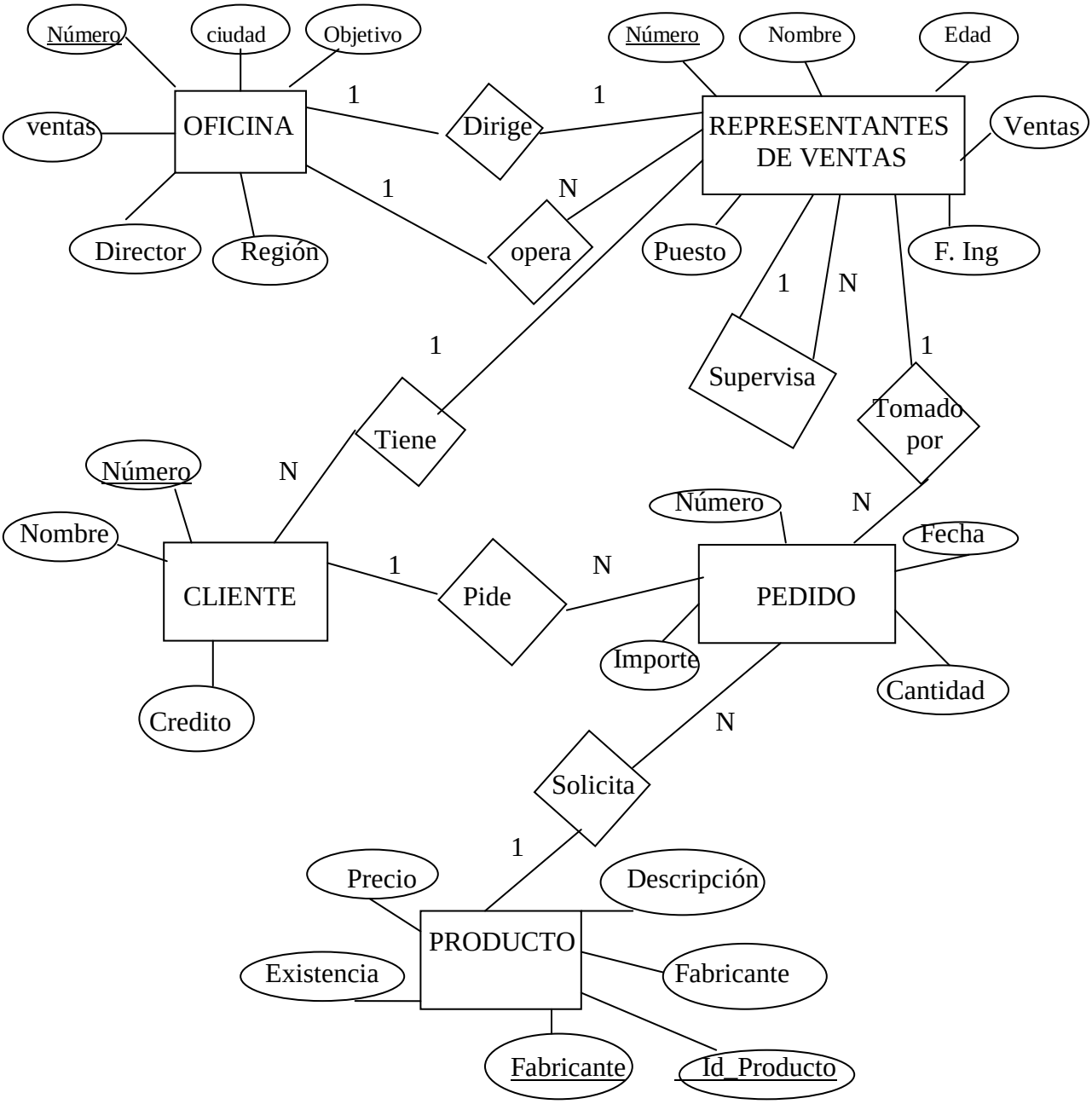
REQUERIMIENTOS DE UNA COMPAÑÍA DE VENTAS

Una pequeña compañía sobre venta de productos varios solicita el desarrollo de un sistema en el cual se permita llevar un control adecuado sobre los pedidos realizados, una relación de clientes, y personal que labora en tal compañía. Lo que se solicita es:

1. Se desea saber en qué ciudad y región se localizan cada una de las oficinas de promoción y ventas. Sus ventas objetivo y sus ventas acumuladas, que representante de ventas las dirige y quienes laboran en ellas como representante de ventas. Cada oficina tendrá un número que las identifique.
2. Se desea guardar la información de cada representante de ventas. Los representantes tendrán un número único que los identifique, nombre, edad, cargo, fecha de contratación, quien los supervisa, cuota mínima que deben de cubrir al año, ventas acumuladas al momento, clientes que atienden (son políticas de la compañía que los representantes de ventas no puedan tener clientes que tengan otros representantes de ventas), y se quiere llevar un control de todos los pedidos que levanta cada represéntate de ventas.
3. Por cada cliente, se solicita que se tenga un número de cliente que lo identifique, el nombre, límite de crédito y los pedidos que ha solicitado.
4. Se desea llevar un control de los productos en venta y se quiere que se almacenen los siguientes datos: el nombre del fabricante, identificación del producto (cada

fabricante pone su identificación del producto en forma totalmente independiente con los demás fabricantes, por lo que puede haber más de un tipo de producto con el mismo nombre), descripción del producto, precio y existencia en bodega.

5. Para llevar un mejor control de los pedidos, la compañía solicita que se facture por producto vendido, su fecha y cantidad solicitada.



MODELO ENTIDAD-RELACION DE UNA COMPAÑIA DE VENTAS DE PRODUCTOS VARIOS
INTENCIÓN DE LA BASE DE DATOS

OFICINA

Número	Ciudad	Región	Ventas objetivo	Ventas acumuladas	Director

REPRESENTANTE

Número	Nombre	Edad	Oficina	Cargo	Contrato	Supervisor	Cuota mínima	Ventas

CLIENTE

Número	Nombre	Crédito Máximo	Representante

PRODUCTO

Fabricante	Identificación del producto	Descripción	Precio	Existencia

PEDIDO

No. factura	**Fecha**	Cantidad	Cliente	Representante	Fabricante	Identificación del producto	importe

EXTENSIÓN DE LA BASE DE DATOS.

clientes			
numero	**nombre**	**representante**	**credito**
2101	JONES MFG.	106	$65,000.00
2102	FIRST CORP.	101	$65,000.00
2103	ACME MFG.	105	$50,000.00
2105	AAA INVESTMENTS	101	$45,000.00
2106	FRED LEWIS CORP.	102	$65,000.00
2107	ACE INTERNATIONAL	110	$35,000.00
2108	HOLMS AND LANDIS	109	$55,000.00
2109	CHEN ASSOCIATES	103	$25,000.00
2111	JCP INC.	103	$50,000.00
2112	ZETACORP	108	$50,000.00
2113	IAN AND SCHMIDT	104	$20,000.00
2114	ORION CORP.	102	$20,000.00
2115	SMITHSON CORP.	101	$20,000.00
2117	J. P. SINCLAIR	106	$35,000.00
2118	MIDWEST SYSTEMS	108	$60,000.00
2119	SOLOMON INC.	109	$25,000.00
2120	RICO ENTERPRISES	102	$50,000.00
2121	QMA ASSOC.	103	$45,000.00
2122	THREE-WAY LINES	105	$30,000.00
2123	CARTER AND SONS.	102	$40,000.00
2124	PETER BROTHERS	107	$40,000.00

OFICINAS					
OFICINA	CIUDAD	REGIÓN	DIRECTOR	OBJETIVO	VENTAS
11	NEW YORK	ESTE	106	$575,000.00	$692,637.00
12	CHICAGO	ESTE	104	$800,000.00	$735,042.00
13	ATLANTA	ESTE	105	$350,000.00	$367,911.00
21	LOS ANGELES	OESTE	108	$725,000.00	$835,915.00
22	DENVER	OESTE	108	$300,000.00	$186,042.00

PEDIDOS							
NUMERO	FECHA	CLIENTE	REPRESENTANTE	FABRICANTE	PRODUCTO	CANTIDAD	IMPORTE
110036	30-Ene-00	2107	110	ACI	4100Z	9	$22,500.00
112961	17-Dic-89	2117	106	REI	2A44L	7	$31,500.00
112963	17-Dic-89	2103	105	ACI	41004	28	$3,276.00
112968	12-Oct-89	2102	101	ACI	41004	34	$3,978.00
112975	12-Oct-89	2111	103	REI	2A44G	6	$2,100.00
112979	12-Oct-89	2114	102	ACI	4100Z	6	$15,000.00
112983	27-Dic-89	2103	105	ACI	41004	6	$702.00
112987	31-Dic-89	2103	105	ACI	4100Y	11	$27,500.00
112989	03-Ene-90	2101	106	FEA	114	6	$1,458.00
112992	04-Nov-89	2118	108	ACI	41002	10	$760.00
112993	04-Ene-89	2106	102	REI	2A45C	24	$1,896.00
112997	08-Ene-90	2124	107	BIC	41003	1	$652.00
113003	25-Ene-90	2108	109	IMM	779C	3	$5,625.00
113007	08-Ene-90	2112	108	IMM	773C	3	$2,925.00
113012	11-Ene-90	2111	105	ACI	41003	35	$3,745.00
113013	14-Ene-90	2118	108	BIC	41003	1	$652.00
113024	20-Ene-90	2114	108	QSA	XK47	20	$7,100.00
113027	22-Ene-90	2103	105	ACI	41002	54	$4,104.00
113034	29-Ene-90	2107	110	REI	2A45C	8	$632.00
113042	02-Feb-90	2113	101	REI	2A44R	5	$22,500.00
113045	02-Feb-90	2112	108	REI	2A44R	10	$45,000.00
113048	10-Feb-90	2120	102	IMM	779C	2	$3,750.00
113049	10-Feb-90	2118	108	QSA	XK47	2	$776.00
113051	10-Feb-90	2118	108	QSA	K47	4	$1,420.00
113055	15-Feb-90	2108	101	ACI	4100X	6	$150.00
113057	18-Feb-90	2111	103	ACI	4100X	24	$600.00

PEDIDOS							
NUMERO	FECHA	CLIENTE	REPRESENTANTE	FABRICANTE	PRODUCTO	CANTIDAD	IMPORTE
113058	23-Feb-90	2108	109	FEA	112	10	$1,480.00
113062	24-Feb-90	2124	107	FEA	114	10	$2,430.00
113065	27-Feb-90	2106	102	QSA	XK47	6	$2,130.00
113069	02-Mar-90	2109	107	IMM	775C	22	$31,350.00

PRODUCTOS				
FABRICANTE	IDPRODUCTO	DESCRIPCIÓN	PRECIO	EXISTENCIA
ACI	41001	ARTICULO TIPO 1	$55.00	277
ACI	41002	ARTICULO TIPO 2	$76.00	167
ACI	41003	ARTICULO TIPO 3	$107.00	207
ACI	41004	ARTICULO TIPO 4	$117.00	139
ACI	4100X	AJUSTADOR	$25.00	37
ACI	4100Y	EXTRACTOR	$2,750.00	25
ACI	4100Z	MONTADOR	$2,500.00	28
BIC	41003	MANIVELA	$652.00	3
BIC	41089	RETEN	$225.00	78
BIC	41672	PLATE	$180.00	0
FEA	112	CUBIERTA	$148.00	115
FEA	114	BANCADA MOTOR	$243.00	15
IMM	773C	RIOSTRA 1/2-Tm	$975.00	28
IMM	775C	RIOSTRA 1-Tm	$1,425.00	5
IMM	779C	RIOSTRA 2-Tm	$1,875.00	9
IMM	887H	SOPORTE RIOSTRA	$54.00	223
IMM	887P	PERNO RIOSTRA	$250.00	24
IMM	887X	RETENEDOR RIOSTRA	$475.00	32
QSA	KX48	REDUCTOR	$134.00	203
QSA	XK47	REDUCTOR	$355.00	38
QSA	XK48A	REDUCTOR	$117.00	37
REI	2A44G	PASADOR BISAGRA	$350.00	14
REI	2A44L	BISAGRA IZQDA.	$4,500.00	12
REI	2A44R	BISAGRA DCHA.	$4,500.00	12
REI	2A45C	VASTAGO TRINQUETE	$79.00	210

REPVENTAS								
NUMERO	NOMBRE	EDAD	OFICINA	PUESTO	CONTRATO	SUPERVISOR	CUOTA	VENTAS
101	DAN ROBERTS	45	12	REP VENTAS	20-Oct-86	104	$300,000.00	$305,673.00
102	SUE SMITH	48	21	REP VENTAS	10-Dic-86	108	$350,000.00	$474,050.00
103	PAUL CRUZ	29	12	REP VENTAS	01-Mar-87	104	$275,000.00	$286,775.00
104	BOB SMITH	33	12	DIR VENTAS	19-May-87	106	$200,000.00	$142,594.00
105	BILL ADAMS	37	13	REP VENTAS	12-Feb-88	104	$350,000.00	$367,911.00
106	SAM CLARK	52	11	VP VENTAS	14-Jun-88		$275,000.00	$299,912.00
107	NANCY ANGELLI	49	22	REP VENTAS	14-Nov-88	108	$300,000.00	$186,042.00

REPVENTAS								
NUMERO	NOMBRE	EDAD	OFICINA	PUESTO	CONTRATO	SUPERVISOR	CUOTA	VENTAS
108	LARRY FITCH	62	21	DIR VENTAS	12-Oct-89	106	$350,000.00	$361,865.00
109	MARY JONES	31	11	REP VENTAS	12-Oct-99	106	$300,000.00	$392,725.00
110	TOM SNYDER	41		REP VENTAS	13-Jun-90	101		$75,985.00

CONSULTAS

Responder las consultas en algebra relacional, cálculo relacional y en SQL.

1. Lista de las oficinas de ventas con sus objetivos y ventas reales
2. Lista de las oficinas de ventas en la región Este con sus objetivos y ventas
3. Lista de las oficinas de ventas de la región Este cuyas ventas exceden a sus objetivos, ordenadas en orden alfabético por ciudad
4. Cúales son los objetivos y ventas promedio para las oficinas de la región Este?
5. Lista los nombres, oficinas y fechas de contrato de todos los vendedores.
6. Cuál es el nombre, cuota y ventas del empleado 107?
7. Cuáles son las ventas promedio de los vendedores?
8. Lista el nombre y la fecha de contrato de cualquier vendedor cuyas ventas sean superiores a $500,000.00
9. Lista de los vendedores, sus cuotas y sus direcciones
10. Lista la ciudad, la región y el importe por encima o por debajo del objetivo para cada oficina.
11. Muestra el valor del inventario para cada producto.
12. Muestra qué sucederá si se eleva la cuota de cada vendedor un 3% de sus ventas anuales hasta la fecha correspondiente.
13. Lista el nombre, mes, y año de contrato para cada vendedor
14. Muestra las oficinas en donde las ventas exceden al objetivo
15. Halla los vendedores contratados antes de 1988
16. Lista las oficinas cuyas ventas están por debajo del 80% del objetivo
17. Lista las oficinas no dirigidas por el empleado 108.
18. Recupera el nombre y el límite de crédito del cliente número 2107
19. Lista vendedores que superan sus cuotas
20. Lista los vendedores que están por debajo de sus cuotas
21. Hallar los pedidos que caen en los rangos: de 20000 a 29999.99. Use el comando BETWEEN.
22. Lista los vendedores cuyas ventas no están entre el 80% y el 120% de sus ventas. Use el comando NOT BETWEEN.
23. Hallar todos los pedidos obtenidos por los vendedores 107,109, 101 y 103. Use el comando IN.
24. Hallar el vendedor que aún no tiene asignada una oficina
25. Lista los vendedores a los que se les ha asignado una oficina.
26. Hallar los vendedores que están por debajo de la cuota o con ventas inferiores a los $300,000.00
27. Hallar los vendedores que están por debajo de la cuota y tienen ventas inferiores a $300,000.00
28. Hallar todo vendedor que está por debajo de la cuota, pero cuyas ventas no son inferiores a $150,000.00
29. Hallar todos los vendedores que:a) trabajan en Denver, Nueva York o Chicago; b) no tienen director y fueron contratados a partir de junio de 1988, o c) están por encima de la cuota, pero tienen ventas de $600,000.00 o menos
30. Muestra las ventas de cada oficina, ordenadas en orden alfabético por región, y dentro de cada región por ciudad. Usar ORDER BY

31. Lista las oficinas, clasificadas en orden descendente de rendimiento de ventas, de modo que las oficinas con mayor rendimiento aparezcan primero
32. Lista todos los productos para los cuales más de $30,000.00 del producto hayan sido ordenados en un sólo pedido.
33. Lista todos los productos en donde el precio del producto excede de $2,000.00 o en donde más de $30,000.00 del producto haya sido ordenado en un solo pedido. Usar el comando UNION
34. Lista todos los producto en donde el precio del producto supera a $2,000.00 o en donde más de $30,000.00 del producto han sido ordenados en un solo pedido, clasificados por fabricante y número de producto.
35. Lista todas las oficinas con un objetivo mayor a $600,000.00
36. Lista los pedidos superiores a $25,000.00, incluyendo el nombre del vendedor que tomó el pedido y el nombre del cliente que lo solicitó
37. Lista todas las combinaciones de vendedores y oficinas en donde la cuota del vendedor es superior al objetivo de la oficina.
38. Lista los nombres de los vendedores y sus supervisores.
39. Muestra todas las combinaciones posibles de vendedores y ciudades.
40. Muestra todos los vendedores y ciudades en donde trabajan.
41. Lista el nombre de las empresas y todos los pedidos para el cliente 2103
42. Cuáles son las cuotas y ventas totales para todos los vendedores?
43. Cuál es el total de los pedidos aceptados por Bill Adams?
44. Calcula el precio medio de los productos del fabricante ACI
45. Cuáles son las cuotas asignadas mínima y máxima?
46. Cuál es la fecha de pedido más antigua en la base de datos?
47. Cuál es el mejor rendimiento de ventas de todos los vendedores?
48. Cuántos vendedores superan su cuota?
49. Cuántos pedidos de más de $25,000.00 hay en los registros?
50. Cuántas oficinas de ventas tienen vendedores que superan sus cuotas?
51. Cuál es el tamaño medio de pedido?
52. Cuál es el tamaño medio de pedido para cada vendedor?
53. Cuál es el rango de cuotas asignadas en cada oficina?
54. Cuántos vendedores están asignados a cada oficina?
55. Por cada oficina con dos o más personas, calcular la cuota total y las ventas totales para todos los vendedores que trabajan en la oficina.
56. Muestra el precio, la existencia y la cantidad total de los pedidos de cada producto para los cuales la cantidad total pedida es superior al 75% de existencias. Usar HAVING.
57. Lista las oficinas en donde el objetivo de ventas de la oficina excede a la suma de las cuotas de los vendedores individuales.
58. Lista los vendedores cuyas cuotas son iguales o superiores al objetivo de la oficina de ventas en Atlanta.
59. Lista los vendedores que trabajan en oficinas que superan su objetivo.
60. Lista los productos para los cuales se ha recibido un pedido de $25,000.00 o más.
61. Lista los clientes asignados a SUE SMITH que no han remitido un pedido superior a $3,000.00. Usar NOT EXISTS
62. Lista las oficinas en donde haya un vendedor cuya cuota represente más del 55% del objetivo en la oficina.
63. Lista los vendedores que han aceptado un pedido que represente más del 10% de su cuota.
64. Lista los nombres y edades de los vendedores que tienen cuotas por encima del promedio.
65. Lista las oficinas de ventas cuyas ventas están por debajo del objetivo promedio.
66. Lista los vendedores que tienen más de 40 años y que dirigen a un vendedor por encima de la cuota.

67. Lista los vendedores cuyo tamaño de pedido medio para productos fabricados por ACI es superior al tamaño de pedido medio global
68. Lista los vendedores cuyo tamaño de pedido medio por importe para productos fabricados por ACI es al menos tan grande como el tamaño de pedido medio por importe global de ese vendedor.

7 DEPENDENCIAS Y NORMALIZACIÓN.

(Elmasri & Navate, 2008)
(de Miguel Castaño, Piattini Velthuis, & Marcos Martinez, 2000)
(Reinosa, Maldonado, Nuñoz, Damiano, & Abrustsky, 2012)

7.1 Introducción

Hasta este momento, hemos supuesto que los atributos se agrupan para formar un esquema de relación empleando el sentido común del diseño de Bases de Datos. Pero **no** hemos contado con una medida formal del porqué una agrupación de atributos para formar un esquema de relación puede ser mejor que otra.

Se tratará de estudiar parte de la teoría que se ha desarrollado en un intento por elegir buenos esquemas de relación: esto es, por medir **formalmente** las razones por las que una agrupación de atributos en esquemas de relación es mejor que otra.

En el inicio se estudiarán 4 medidas informales de la calidad para el diseño de esquemas de relación:

- Semántica de los atributos.
- Reducción de los valores redundantes en las tuplas.
- Reducción de los valores nulos en las tuplas.
- Prohibición de tuplas espurias.

Estas medidas no siempre son independientes entre sí, como veremos.

SEMÁNTICA DE LOS ATRIBUTOS DE UNA RELACIÓN.

Siempre que agrupamos atributos para formar un esquema de relación, supondremos que hay un cierto significado asociado a los atributos. Cuando más fácil sea explicar la semántica de la relación, mejor será el diseño del esquema correspondiente.
Se puede enunciar la siguiente pauta para el diseño de un esquema relacional:

PAUTA 1. Diseñe un esquema de relación de modo que sea fácil explicar su significado. No combine atributos de varios tipos de entidades y tipos de vínculos en una sola relación. Si un esquema de relación corresponde a un tipo de entidad o a un tipo de vínculo, el significado tiende a ser claro. En caso contrario, tiende a ser una mezcla de múltiples entidades y vínculos y, por tanto, será semánticamente confuso.

Ejemplo.
EMPLADO_DEPARTAMENTO
Nombre_empleado, matricula, f_nac, dirección, Núm_dep, Nombre_dep, Matrigerente

EMPLEADO_PROYECTO
Matricula, PNumero, Hrs, Nombre_empleado, Nombre_proy, Lugar_proy.

Aunque no hay nada incorrecto en estas relaciones desde el punto de vista lógico, se les considera diseño deficiente porque violan la pauta número uno al mezclar atributos de entidades distintas del mundo real.

INFORMACIÓN REDUNDANTE EN TUPLAS Y ANOMALIAS DE ACTUALIZACIÓN.

Uno de los objetivos en el diseño de esquemas es minimizar el espacio de almacenamiento que ocupan las relaciones. La agrupación de atributos en esquemas de relación tiene un efecto significativo sobre el espacio de almacenamiento.
Ejemplo, vaciar los valores correspondientes a los esquemas anteriores y compararlos con el esquema primario.
Esto produce un problema grave conocido como las anomalías de actualización y se clasifican en:
ANOMALIAS DE INSERCIÓN.

- Si queremos insertar la tupla de un nuevo empleado en EMPLEADO_DEPARTAMENTO, debemos incluir los valores de los atributos del departamento al que pertenecen o bien nulos.
- Es difícil insertar un la relación EMPLEADO_DEPARTAMENTO un nuevo departamento que aún no tenga empleados. La única forma de hacer esto es colocar valores nulos en los atributos del empleado. Esto originará problemas ya que la matrícula es la llave primaria de EMPLEADO_DEPARTAMENTO, y se supone que cada tupla representará una entidad empleado y no una entidad departamento.

ANOMALIAS DE ELIMINACIÓN. Si se elimina el último empleado que pertenecía a un departamento, éste también desaparecerá

ANOMALIAS DE MODIFICACIONES. Si alteramos el valor de uno de los atributos de un departamento dado, debemos actualizar las tuplas de todos los empleados que pertenezcan a ese departamento; de lo contrario, la Base de Datos se volverá inconsistente.

En base a las tres anomalías anteriores se enuncia la siguiente pauta:

PAUTA 2: Diseñe los esquemas de las relaciones de modo que no haya anomalías de inserción, eliminación o modificación en las relaciones.

La segunda pauta es congruente con la primera y en cierta forma, es otro modo de explicarla. Es importante señalar que hay ocasiones en que es preciso violar esta pautas para mejorar el rendimiento de ciertas consultas.

VALORES NULOS EN LAS TUPLAS

Si muchos de los atributos no se aplican a todas la tuplas de la relación, acabaremos con un gran número de nulos en esas tuplas. Esto puede originar un gran desperdicio de espacio en el nivel de almacenamiento y posiblemente dificultar el entendimiento del significado de los atributos y la especificación de operaciones de REUNIÓN. Otro

problema con los nulos es el manejo de estos cuando se aplican funciones agregadas como COUNT o SUM. Además, los nulos pueden tener múltiples interpretaciones, como:

- El atributo no se aplica a al tupla
- Se desconoce el valor del atributo para la tupla.
- El valor se conoce pero aún no ha sido registrado.

La otra pauta es:

PAUTA 3: Hasta donde sea posible, evite incluir en una relación atributos cuyos valores puedan ser nulos. Si no es posible evitarlos, asegúrese de que se apliquen sólo en casos excepcionales y no a la mayoría de las tuplas de una relación.

Ejemplo: Si sólo el 10% de empleados tienen oficinas individuales, no se justificará incluir un atributo No_oficina en la relación EMPLEADO; más bien, se puede crear una relación OFICINAS_EMPLEADOS que contengan tuplas exclusivamente con el No de oficina y matrícula de cada empleado.

TUPLAS ESPURIAS.

Suponga que se tienen dos esquemas del modelo compañía con los siguientes datos:

LUGARES_EMPLEADO

NOMBRE	PATERNO	LUGAR_PROY
Juan	Perez	Atenco
Juan	Perez	Sn Felipe
Ramses	Sánchez	Sn Diego
Josefa	España	Atenco

EMPLEADO_PROYECTO

MATRICULA	No_Proy	Hrs	NombreProy	LUGAR_PROY
56789	1	32.5	Producto1	Atenco
56789	2	7.5	Producto2	Sn. Felipe
84444	3	20	Producto3	Sn Diego
53453	1	20	Producto1	Atenco

Si se realiza la siguiente consulta:

T < - EMPLEADO_PROYECTO * LUGARES_EMPLEADO

matricula	no_proyecto	horas	nombreproyecto	Lugarproyecto	nombre	paterno
53453	1	20	PRODUCTO1	ATENCO	JUAN	PEREZ
56789	1	32.5	PRODUCTO1	ATENCO	JUAN	PEREZ
56789	2	7.5	PRODUCTO2	SN FELIPE	JUAN	PEREZ
54444	3	20	PRODUCTO3	SN DIEGO	RAMSES	SÁNCHEZ
53453	1	20	PRODUCTO1	ATENCO	JOSEFA	ESPAÑA
56789	1	32.5	PRODUCTO1	ATENCO	JOSEFA	ESPAÑA

Como se observará, la tupla uno y la tupla dos tienen diferente matricula y el mismo nombre, lo mismo sucede con la penúltima y última tupla. Por lo que dos de las cuatro tuplas son espurias. Esto se debe a que la reunión se realizó con un atributo que no es ni llave primaria ni llave foránea. Quedando la siguiente pauta:

PAUTA 4. Diseñe los esquemas de relación de modo que puedan reunirse mediante condiciones de igualdad sobre atributos que sean llaves primarias o llaves foráneas, a fin de garantizar que no se formarán tuplas espurias.

Estas cuatro pautas indican informalmente situaciones que dan origen a esquemas de relación problemáticos.

El siguiente paso es dar formalidad a estas pautas empleando los conceptos de DEPENDENCIAS y NORMALIZACION.

7.2 DEPENDENCIAS.

Una dependencia funcional es una restricción entre dos conjuntos de atributos de la base de datos. Suponiendo que el esquema de la Base de Datos relacional tiene n atributos A_1, A_2, A_3,…, A_n, y toda la base se describe en un solo esquema de relación universal R= { A_1, A_2, A_3,…, A_n }. Se usará este concepto (relación universal) únicamente para desarrollar la teoría de dependencia en los datos.
Una dependencia funcional, denotada por $X \rightarrow Y$, entre dos conjuntos de atributos X y Y, que son subconjuntos de R especifica una restricción sobre las posibles tuplas que podrían formar un ejemplar de relación r (R). La restricción dice que, para cualquier dos tuplas t_1 y t_2 de r(R) tales que $t_1[X] = t_2[X]$, debemos tener también $t_1[Y] = t_2[Y]$. Esto significa que los valores del componente Y de una tupla de r(R) dependen de los valores del componente X, o están determinados por ellos; o bien, que los valores del componente X de una tupla determinan de manera única (o funcionalmente) los valores del componente Y. También decimos que Y depende funcionalmente de X. (D.F).

Así, X funcionalmente determina Y en un esquema de relación R si y sólo si, cada vez que dos tuplas de r(R) coinciden en su valor X, ellos necesariamente coinciden en su valor Y, note que:

- Si una restricción en R indica que no puede haber más de una tupla con un valor dado X en alguna instancia de relación r(R) – esto es, X es una llave candidata de R – esto implica que $X \rightarrow Y$ para algún subconjunto de atributos Y de R.
- Si $X \rightarrow Y$ en R, esto no dice si o no $Y \rightarrow X$ en R.

Las dependencias funcionales son propiedades del significado o la semántica de los atributos, esto es, qué relación hay entre ellos.
Ejemplo, considere la instancia de relación:

EMPLEADO_PROYECTO={Matricula, Pnúmero, Hrs, Paterno, Materno, Pnombre, Plugar}

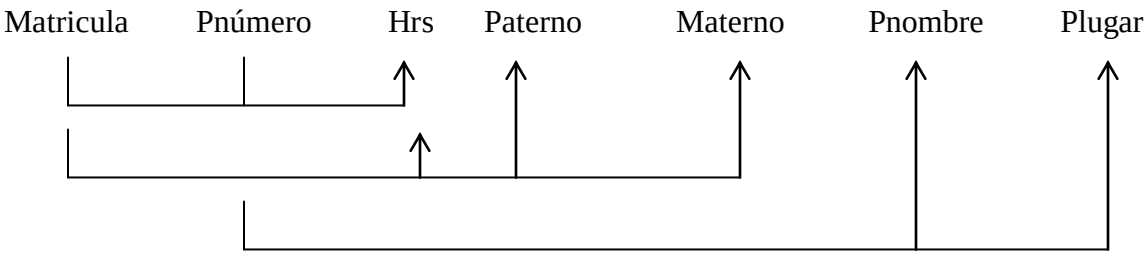
A partir de la semántica de los atributos se deben cumplir las siguientes dependencias funcionales:

- Matricula $\rightarrow$ {Paterno, Materno}
- Pnúmero $\rightarrow$ {Pnombre, Plugar}
- {Matricula, Pnumero} $\rightarrow$ {Hrs}

Esto indica que:

- ➔ El número de matrícula determina en forma única el nombre del empleado.
- ➔ Pnúmero determina de manera única su nombre del proyecto y su localización
- ➔ La combinación de Matricula y Pnúmero determina de manera única las hrs.

Hablando en forma de diagrama queda:



Una dependencia funcional debe **definirla explícitamente alguien que conozca la semántica de los atributos de R.**

Ejemplo:

IMPARTIR

PROFESOR	CURSO	TEXTO
Poncho	Geometría Analítica	Leithold
Polo	Cálculo	Swokowski
Poncho	Compiladores	Hoffman

A primer vista podemos decir que TEXTO $\rightarrow$ CURSO, no se puede confirmar hasta que se cheque que se cumple para todos los posibles estados de la relación de IMPARTE. Por otro lado, bastó presentar un solo contraejemplo para demostrar que no existe una dependencia funcional. Por ejemplo, el hecho de que Poncho imparte tanto Geometría Analítica como Compiladores podemos concluir que PPROFESOR no determina funcionalmente a CURSO y también podemos decir que CURSO no determina funcionalmente a TEXTO.

7.2.1 REGLAS DE INFERENCIA PARA LAS DEPENDENCIAS FUNCIONALES.

Por lo regular, el diseñador del esquema especifica las D. F. que son semánticamente obvias, pero es común que muchas otras D. F. se cumplan en todos los ejemplares de relación permitidos que satisfagan las dependencias de F. El conjunto de todas estas dependencias funcionales se denomina cerradura de F, y se denota con F^+. Por ejemplo, suponga que especificamos el siguiente conjunto F de dependencias obvias sobre la siguiente instancia de relación:

EMP_DEPTO={Nombreemp, Matricula, Fechanac, Dirección, NúmDepto, Dnombre, MatGerente}

F={Matricula $\rightarrow$ {Nombreemp, Fechanac, Dirección, NúmDepto}, NúmDepto $\rightarrow$ {Dnombre, MatGerente}}

Se puede inferir:
Matricula $\rightarrow$ {Dnombre, MatGerente}

La cerradura F^+ de F es el conjunto de todas las dependencias funcionales que pueden inferirse de F. Para determinar una forma sistemática de inferir dependencias, se emplean un conjunto de reglas de inferencia:

1. Regla reflexiva. Si $Y \subseteq X$, entonces $X \rightarrow Y$
2. Regla de aumento. $X \rightarrow Y \mapsto XZ \rightarrow YZ$ **o** $XZ \rightarrow Y$
3. Regla transitiva. $X \rightarrow Y$, $Y \rightarrow Z \mapsto .\ X \rightarrow Z$
4. Regla de descomposición (o proyectiva) . $X \rightarrow YZ \quad \mapsto \quad . X \rightarrow Y$
5. Regla de unión (o aditiva). $X \rightarrow Y$, $X \rightarrow Z \mapsto X \rightarrow YZ$
6. Regla pseudotransitiva. $X \rightarrow Y$, $WY \rightarrow Z \mapsto W X \rightarrow Z$

La regla uno dice que un conjunto de atributos siempre se determina a sí mismo. La regla dos dice que añadir el mismo conjunto de atributos a los dos miembros, izquierdo y derecho, de una dependencia produce otra dependencia válida. La regla tres indica que la dependencia funcional es transitiva. La regla cuatro dice que podemos quitar atributos del miembro derecho de una dependencia; la aplicación repetida de esta regla puede descomponer la D. F. $X \rightarrow \{A_1, A_2, A_3, \ldots, A_n\}$ en el conjunto de dependencias $\{ X \rightarrow A_1, X \rightarrow A_2, X \rightarrow A_3, \ldots, X \rightarrow A_n \}$; la regla cinco permite hacer lo opuesto: combinar un conjunto de dependencias $\{ X \rightarrow A_1, X \rightarrow A_2, X \rightarrow A_3, \ldots, X \rightarrow A_n \}$ en una sola dependencia funcional $X \rightarrow \{A_1, A_2, A_3, \ldots, A_n\}$.

Armstrong (1974) demostró que las reglas de inferencia de la uno a la tres son correctas y completas.
Correcta indica que dado un conjunto de dependencias funcionales F, especificado sobre un esquema de relación R, cualquier dependencia que podemos inferir de F con las 3 reglas se cumplirá en todos los estados de relación r(R) que satisfagan las dependencias de F.
Con **completas** se quiere decir que el empleo repetido de las tres reglas para inferir dependencias hasta que no sea posible inferir más dependencias producirá el conjunto de todas las dependencias posibles que se pueden inferir de F.

Por lo general, los diseñadores de B. D.´s especifican primero el conjunto de dependencias funcionales F que se pueden determinar sin dificultad a partir de la semántica de los atributos en R. Posteriormente pueden usar las reglas de inferencia de Armstrong para inferir dependencias funcionales adicionales que también se cumplirán en R. Una forma semántica de determinar estas dependencias funcionales adicionales consiste en determinar primero todos y cada uno de los conjuntos de atributos X que aparezcan como miembro izquierdo de alguna dependencia funcional en F y después usar las reglas de inferencia de Armstrong para determinar el conjunto de todos los atributos que dependan de X. Así, para cada conjunto de atributos X, determinamos el conjunto X^+ de atributos determinados funcionalmente por X; X^+ se denomina cerradura de X bajo F. Un algoritmo para calcular X^+ es:

$X^+ = X$;
Repetir
    Viejo $X^+ = X^+$;
    Para cada D. F. $Y \rightarrow Z$ en F, hacer:
        Si $Y \subseteq X^+$ entonces
            $X^+ = X^+ \cup Z$;
Hasta que (viejo $X^+ = X$);

Ejemplo:
Empleando la relación
EMP_PROY={Matricula, Pnumero, Hrs, Enombre, Pnombre, Plocalización}
Se tienen las siguientes dependencias funcionales obvias:
F={Matricula $\rightarrow$ {Enombre}, Pnumero $\rightarrow$ {Pnombre, Plugar},{Matricula, Pnumero $\rightarrow$ {Hrs}}

Usando el algoritmo anterior, se puede calcular la siguiente cláusula con respecto a F:

$\{Matricula\}^+ = \{Matricula, Nombre\}$
$\{Pno\}^+ = \{Pno, Pnombre, Plugar\}$
$\{Matricula, Pno\}^+ = \{Matricula, Pno, Nombre, Pnombre, Plugar, Hrs\}$//Por la regla de aumento.

7.3 NORMALIZACIÓN.

En el proceso de normalización se somete un esquema de relación a una serie de pruebas para “certificar” si pertenece o no a una cierta forma normal. Existen tres formas normales que se basan en dependencias funcionales entre los atributos de una relación. Existen la cuarta forma normal y la quinta forma normal que se basan en las dependencias multivaluadas y las dependencias de reunión.

La normalización de los datos puede considerarse como un proceso durante el cual los esquemas de relación insatisfechos se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que poseen propiedades deseables. Un objetivo del proceso de normalización original es garantizar que no ocurran las anomalías de actualización. Las formas normales proveen a los diseñadores de B. D´s lo siguiente:

- Un marco formal para analizar los esquemas de relación con base en sus llaves y en las D. F´s entre sus atributos.
- Una serie de pruebas que pueden efectuarse sobre los esquemas de relación individuales de modo que la base de datos relacional pueda normalizarse hasta el grado deseado. Cuando una prueba falla, la relación que provoca el fallo debe descomponerse en relaciones que individualmente satisfagan las pruebas de normalización.

Las formas normales no garantizan un buen diseño de la base de datos. El proceso de normalización debe confirmar también la existencia de propiedades adicionales que los esquemas relacionales deben poseer. Dos de estas propiedades son:

- La propiedad de reunión sin pérdida o reunión no aditiva, garantiza que no se presentará el problema de las tuplas espurias.
- La propiedad de conservación de las dependencias, asegura que todas las D. F´s estén representadas en algunas de las relaciones individuales resultantes.

La utilidad práctica de las formas normales queda en entredicho cuando las restricciones en las que se basan son difíciles de entender o de detectar por parte de los diseñadores de B. D´s y usuarios que deben descubrir estar restricciones.
Los diseñadores de B. D.´s **no deben que normalizar hasta la F.N. más alta posible**.
Las relaciones pueden dejarse en formas normales inferiores por razones de rendimiento.

Unos conceptos para recordar:

Una súper llave de un esquema de relación R={A_1, A_2, A_3,… A_n} es un conjunto de atributos $S \subseteq R$ con la propiedad de que no habrá un par de tuplas t_1, t_2 en ningún estado de relación permitido r(R) tal que $t_1[S]= t_2[S]$.

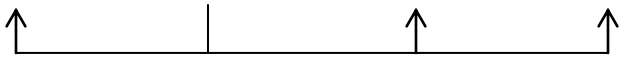
Una llave es una súper llave con la propiedad adicional de que la eliminación de cualquier atributo K hará de que K deje de ser una súper llave. La diferencia entre una llave y una súper llave es que la primera tiene que ser mínima; esto es, si tenemos una llave K = {A_1, A_2, A_3,… A_K}, entonces K- A_i deja de ser llave.
Si un esquema de relación tiene más de una llave mínima, todas son llaves candidatas. Una de ellas de denotará arbitrariamente como llave primaria.
Un atributo del esquema de relación R se denomina atributo primo de R si es miembro de cualquier llave de R. Un atributo es no primo si no es miembro de ninguna llave candidata.
Ejemplo. De la relación TRABAJA_EN, de la B. D.´s Compañía, son atributos primos Matricula y Pno, Hrs no es atributo primo.

7.3.1 PRIMERA FORMA NORMAL.

Los dominios de los atributos deben incluir sólo valores atómicos (simples e indivisibles). La primera forma normal prohíbe las "relaciones dentro de las relaciones" o las "relaciones como atributos de tuplas".
Ejemplo:

DEPARTAMENTO

Nombre Numero Matrigerente Lugares



Podría quedar como:

NOMBRE	NÚMERO	MATRIGERENTE	LUGARES
Investigación	5	45555	(Atenco,Sn Felipe,Sn Diego)
Administración	4	54321	Papalotla
Amaranto	1	00007	Sn Diego

O quedar

NOMBRE	NÚMERO	MATRIGERENTE	LUGARES
Investigación	5	45555	Atenco
Investigación	5	45555	Sn Felipe
Investigación	5	45555	Sn Diego
Administración	4	54321	Papalotla
Amaranto	1	00007	Sn Diego.

La primera forma en que se realizó se observa que su dominio no tiene valores atómicos, por lo que se recomienda separar ese atributo en conjunto con la llave primaria y formar otra relación.
En la segunda forma, la llave primaria es la combinación de {Número, Lugares}. Se observará que tiene el gran problema de redundancias.

La primera forma normal también prohíbe los atributos compuestos que por sí son multivaluados. Estos se denominan relaciones anidadas porque cada tupla puede tener una relación dentro de sí.
Ejemplo:

EMPLEADO_PROYECTO {Matricula, Nombre, Proyecto (Pno, Hrs)}
En este caso, la llave primaria es Matricula y la llave parcial en la relación anidada es Pno. Para pasar esta relación a primera forma normal pasamos los atributos de la relación anidada a una nueva relación y propagamos la llave primaria en ella; la llave primaria de la nueva relación combina la llave parcial con la llave primaria de la relación original

7.3.2 SEGUNDA FORMA NORMAL.

Una D. F. $X \rightarrow Y$ es total si la eliminación de cualquier atributo A de X hace que la dependencia deje de ser válida. Una D. F. $X \rightarrow Y$ es parcial si es posible eliminar un atributo $A \in X$ y la dependencia sigue siendo válida.

Ejemplo:

{Matricula, Pno} $\rightarrow$ {Hrs}

Es una D. F. total ya que no se puede realizar lo siguiente:

{Matricula} $\rightarrow$ {Hrs} o {Pno} $\rightarrow$ {Hrs}

Pero {Matricula, Pno} $\rightarrow$ {Nombreempleado}
Es parcial porque:

{Matricula} $\rightarrow$ {Nombreempleado}

Un esquema de relación R está en segunda forma normal si todo atributo no primo A en R depende funcionalmente de manera total de la clave primaria.

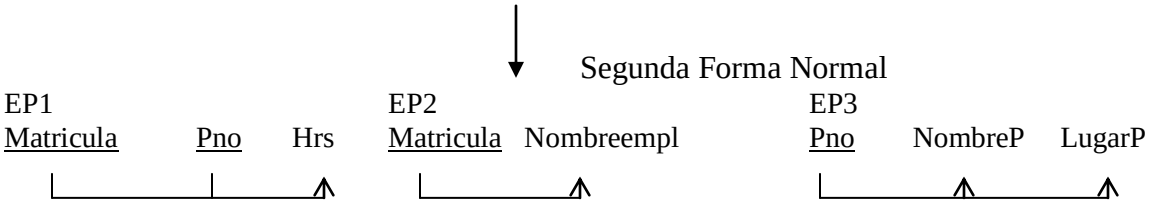
Ejemplo:
EMPLEADO_PROYECTO {Matrícula, Pno, Hrs, Nombreempleado, NombreP, LugarP}

En este caso.

{Matricula, Pno} $\rightarrow$ {Hrs}	Dependencia total
{Matricula} $\rightarrow$ {Nombreempl}	Dependencia parcial
{Pno} $\rightarrow$ {NombreP, LugarP}	Dependencia parcial

No está en segunda forma normal.

Si un esquema no está en segunda forma normal, se puede normalizar a varias relaciones en segunda forma normal en las que los atributos no primos estén asociados sólo a la parte de la llave primaria de la que dependen funcionalmente de manera total. Por lo que queda de la siguiente forma:
EMPLEADO_PROYECTO {Matrícula, Pno, Hrs, Nombreempl, NombreP, LugarP}

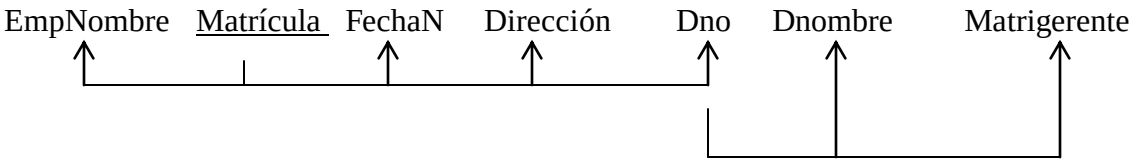


7.3.3 TERCER FORMA NORMAL.

Se basa en el concepto de dependencia transitiva. Una D. F. $X \rightarrow Y$ en un esquema de relación R es una dependencia transitiva si existe un conjunto de atributos Z que no sea un subconjunto de cualquier llave de R, y se cumplen tanto $X \rightarrow Z$ como $Z \rightarrow Y$.

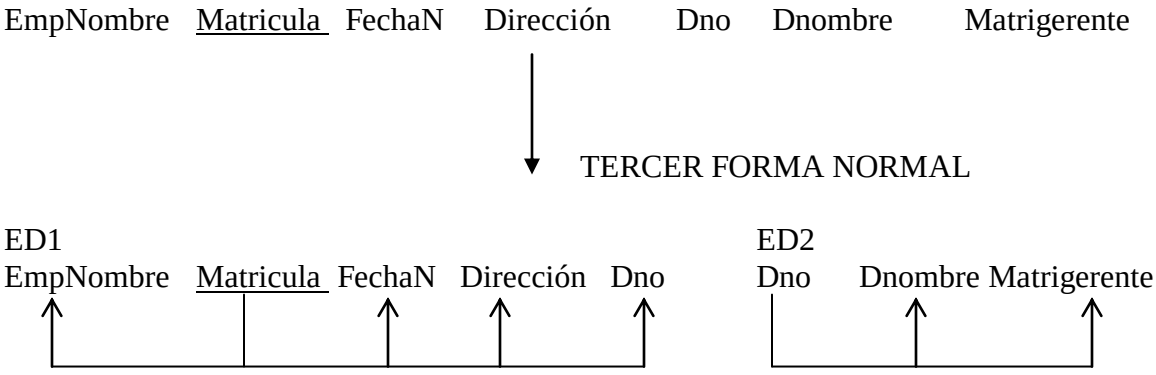
Ejemplo:

EMPLEADO_DEPARTAMENTO



Un esquema de relación R está en tercera forma normal si está en segunda forma normal y ningún atributo no primo en R depende transitivamente de la llave primaria.
El esquema anterior está en segunda forma normal pues no existen dependencias parciales de la llave primaria. Pero no está en tercer forma normal ya que Matrigerente y Dnombre dependen transitivamente de Matricula a través de Dno.
Se puede normalizar EMPLEADO_DEPARTAMENTO descomponiéndolo en dos esquemas de relación en tercer forma normal ED1 y ED2. Se observa que una operación de reunión recuperará la relación original EMPLEADO_DEPARTAMENTO sin generar tuplas espurias.

EMPLEADO_DEPARTAMENTO



DEFINICIÓN GENERAL

2° F. N. Un esquema de relación R está en 2° F. N. si ningún atributo no primo A de R depende parcialmente de cualquier llave (primaria o candidata) de R.

3° F. N. Un esquema de relación R está en 3° F. N. si, siempre que una dependencia funcional $X \rightarrow A$ se cumple en R, o bien:

a. X es una súper llave de R, o
b. A es un atributo primo de R.

Otra forma de decirlo es:
3° F. N. Un esquema de relación R está en 3° F. N. si todo atributo no primo de R es:

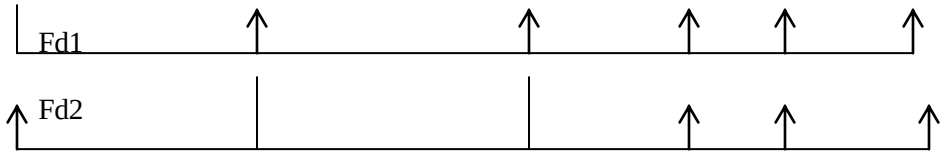
a. Dependiente funcionalmente de manera total de toda clave R, y
b. Dependiente de manera no transitiva de toda llave de R.

Ejemplo:
Se desea hacer una B. D´s para una empresa que se dedica a vender terrenos. Estos terrenos se ubican en diferentes municipios de un estado.
Tendrá un No de lote que son únicos por municipio. También tendrá los identificadores de propiedad que son únicos para todo el estado, sin importar el municipio. La tasa fiscal es fija para un municipio dado (no varía de un lote a otro dentro del mismo municipio).
El precio del lote lo determina su área sin importar en que municipio se encuentre.

La relación universal sería:
LOTES
IDpropiedad NomMunicipio NumLote Área Precio Tasa_Fiscal

Existen dos llaves candidato
{IDpropiedad} y {NomMunicipio, NumLote}

Existen las siguientes D. F´s de estas llaves:
IDpropiedad NomMunicipio NumLote Área Precio Tasa_Fiscal

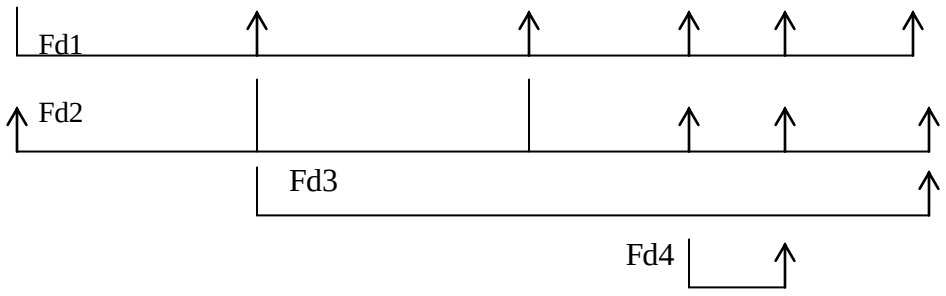


Se escoge a IDpropiedad como llave primaria.
Por los requerimientos también se sabe que:

Fd3. {NomMunicipio} $\rightarrow$ {Tasa_Fiscal}
Fd4.{Área} $\rightarrow$ {Precio}

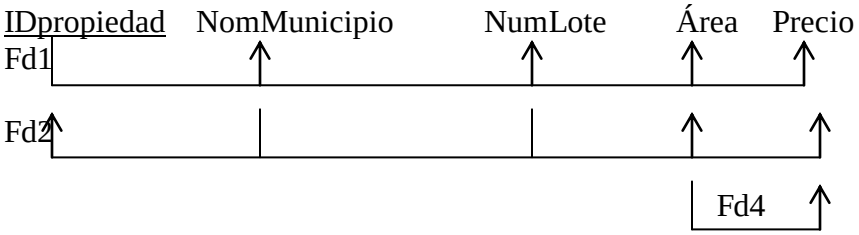
La dependencia Fd3 indica que la Tasa_Fiscal es fija para un municipio dado, mientras Fd4 dice que el precio de un lote es determinado por su área a pesar del municipio donde se encuentre.

<u>IDpropiedad</u> NomMunicipio NumLote Área Precio Tasa_Fiscal

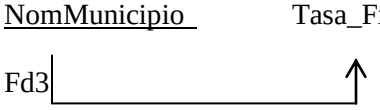


El esquema viola la segunda forma normal ya que Tasa_Fiscal depende en forma parcial de la llave candidata {NomMunicipio, NumLote}. Por lo que se descompone en dos relaciones:

LOTE1

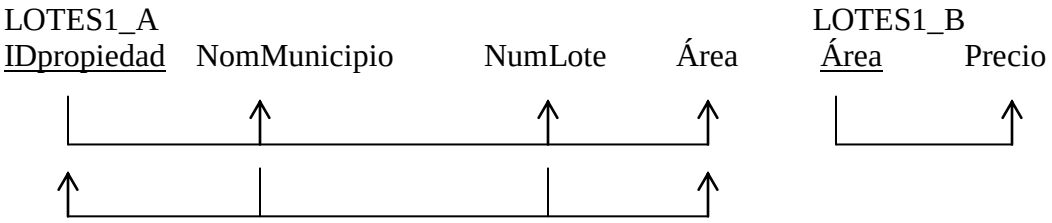


LOTE2

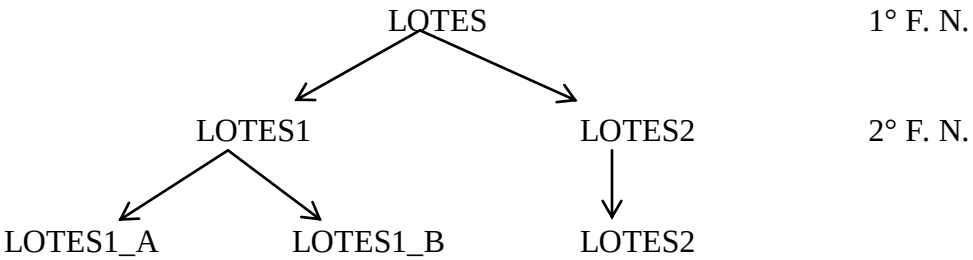


Observe que la dependencia funcional $\{\text{Área}\} \rightarrow \{\text{Precio}\}$ no viola la 2° F. N.

Par a la tercera forma normal, en LOTES1, existe una dependencia transitiva entre Área y Precio. Además, podemos decir que Área no es parte de la súper llave (recordaremos que cuando $X \rightarrow A$ se tiene que cumplir que X sea una súper llave de R, o A sea un atributo primo de R para que exista una 3° F. N.) de LOTES1 y Precio no es un atributo primo. Para normalizar LOTES1 a 3° F. N., se descompone en:



Quedando como:



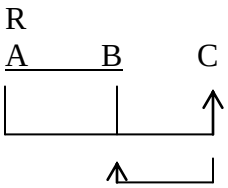
7.3.4 FORMA NORMAL BOYCE-CODD

La forma normal Boyce-Codd es más estricta que la 3° F. N., indicando que toda relación en forma normal B-C está también en 3° F. N; sin embargo, una relación en 3° F. N. no necesariamente está en B-C. Para ejemplificar retornemos a la relación LOTES con sus cuatro dependencias funcionales. Suponga que se tienen cientos de lotes en la relación pero los lotes son de dos municipios solamente Texcoco y Tocuila. Suponga también que el tamaño de los lotes en Tocuila son solo de 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 Hectáreas. Mientras que los tamaños de los lotes en Texcoco son solo de 1.1, 1.2, 1.3, 1.4, 1.5, … , 2.0 Hectáreas. En tal situación se tiene también una dependencia funcional Fd5: Área $\rightarrow$ NomMunicipio. Si se adiciona esta dependencia en la relación LOTES1_A aún está en 3° F. N. porque NomMunicipio es un atributo primo.

La relación que se formaría con los atributos Área y NomMunicipio representado por fd5 puede ser representado por 16 tuplas ya que existen sólo 16 posibles valores de Área. Esta representación reduce la redundancia de repetir la misma información en las miles de tuplas de LOTES1_A y sugiere la necesidad de descomponerla.

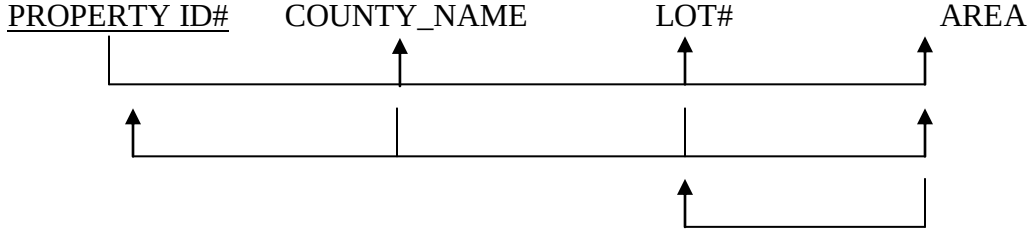
La definición de Boyce-Codd difiere ligeramente de la 3° F. N. Un esquema de relación R está en B-C F.N. si cualquier dependencia funcional X $\rightarrow$ A existente en R, entonces X es una súper llave de R. La única diferencia entre B-C F. N. y 3° F. N. es la condición (b) de la 3° F. N., el cual permite A ser primo si X no es una súper llave, está ausente en B-C F. N.

En la práctica, la mayoría de los esquemas que están en 3° F. N. también están en B-C F. N. Sólo si una dependencia X $\rightarrow$ A existe en un esquema R no siendo X una súper llave y A un atributo primo, entonces R estará en 3° F. N. pero no en B-C F. N. En la siguiente figura se muestra mejor el caso general de tal relación. Lo mejor es tener esquemas de relación en B-C F. N. Si esto no es posible, estará bien que el esquema esté en 3° F. N. Sin embargo, 1° F. N. y 2° F. N. **no se consideran buenos diseños de esquema relaciónales.**



En nuestro ejemplo anterior, la DF viola BCNF en LOTS1A porque AREA no es una súper llave de LOTS1A. Sin embargo, satisface la 3NF porque COUNTY_NAME es un atributo primo. Se puede descomponer LOTS1A en dos relaciones en BCNF LOTS1AX y LOTS1AY, mostrándose el resultado enseguida:

LOTS1A



LOTS1AX LOTS1AY

PROPERTY_ID# AREA LOT# AREA COUNTY_NAME

7.4 DESCOMPOSICIÓN DE RELACIONES E INSUFICIENCIA DE LAS FORMAS NORMALES.

El algoritmo del diseño de B. D's presentado hasta este momento inicia desde un esquema universal de una relación R = {A_1, A_2, A_3,. . . , A_n } donde se incluyen todos los atributos de la B. D`s. El conjunto F de D. F. se especifica en el diseño de la B. D's produciendo una descomposición del esquema universal en esquemas de relaciones D = { R_1, R_2, R_3,. . ., R_m} que formarán el esquema de la B. D's; D será la **descomposición** R.

Se debe tener la seguridad de que cada atributo en R aparezca al menos en un esquema de relación R_i dentro de la descomposición; formalmente:

$$\bigcup_{i=1}^{m} R_i = R$$

Esta condición se conoce como la condición de preservación de atributos.

Una meta es tener cada relación individual R_i de la descomposición D en 3NF o BCNF. Sin embargo, esta condición no es suficiente para que tener un buen diseño de la B. D's por sí mismo ya que en algunos casos aún pueden existir reuniones que produzcan tuplas espurias.

Ejemplo:

emp_locs	
ename	**plocation**
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Narayan, Ramesh K.	Houston
Smith, Johns B.	Sugarland
Smith,Johns B.	Bellaire
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Wong, Franklin T.	Sugarland

project			
pname	**pnumber**	**plocation**	**dnum**
productX	1	Bellaire	5
productY	2	Sugarland	5
productZ	3	Houston	5
computerization	10	Stafford	4
reorganization	20	Houston	1
newbenefits	30	Stafford	4

Tuplas_espurias < - emp_locs * project

tuplas espurias				
ename	Plocation	pname	pnumber	dnum
English, Joyce A.	Bellaire	productX	1	5
Smith,Johns B.	Bellaire	productX	1	5
Wong, Franklin T.	Sugarland	productY	2	5
English, Joyce A.	Sugarland	productY	2	5
Smith, Johns B.	Sugarland	productY	2	5
Wong, Franklin T.	Houston	productZ	3	5
Narayan, Ramesh K.	Houston	productZ	3	5
Wong, Franklin T.	Stafford	computerization	10	4
Wong, Franklin T.	Houston	reorganization	20	1
Narayan, Ramesh K.	Houston	reorganization	20	1
Wong, Franklin T.	Stafford	newbenefits	30	4

En este caso ¿en cuál proyecto trabaja Narayan? o ¿en cuál proyecto trabaja Wong?

En este caso, EMP_LOCS y PROJECT están en BCNF y sin embargo produjeron tuplas espurias.
Por lo que se requieren otros criterios que, junto con la 3NF o BCNF, prevengan un mal diseño.

7.4.1 DESCOMPOSICIÓN Y REUNIONES NO ADITIVAS O SIN PERDIDA DE INFROMACIÓN.

Una propiedad importante que la descomposición D debería poseer es la de reunión no aditiva, el cual asegura que no se generarán tuplas espurias cuando se realice una reunión natural. Si esta propiedad se garantiza en una descomposición, se garantiza que no exista información sin sentido después de una reunión natural.
La reunión natural del EMP_LOCS con PROJECT no tiene la propiedad de reunión no aditiva. En general, se desea que se tenga la capacidad de probar si una descomposición dada D tiene la propiedad no aditiva con respecto a un conjunto de dependencias funcionales F, para tal efecto se tiene el siguiente algoritmo para mostrar si un conjunto de relaciones D tiene la propiedad aditiva.
Este algoritmo muestra que, si alguna hilera en S finaliza con todos los símbolos iguales a "a", entonces la descomposición tiene la propiedad no aditiva con respecto a F. Si no es así, puede ser que se satisfagan todas las dependencias en F, pero no se garantiza la no aditividad.

ALGORITMO:

1. Crear la matriz S con una hilera "i" por cada relación R_i en la descomposición D, y en cada columna cada atributo de la relación R.

2. Se darán valores al conjunto S(i,j)=$b_{i,j}$ donde no incida un atributo de la columna R_i
3. Por cada hilera que represente el esquema de la relación R_i
   Por cada columna j que represente un atributo de A_j
   Si R_i incluye a un atributo A_j
   Entonces S (i, j) = $a_{i,j}$;
4. Repetir hasta que la ejecución del ciclo no modifique a S.
   Para cada dependencia funcional X- > Y en F
   Para cada hilera en S que tenga los mismos símbolos en la
   Columna correspondiente a los atributos en X
   Haga que los símbolos en cada columna que correspondan a un atributo en Y sea el mismo en las hileras como se indica a continuación:
   Si una de las hileras tiene una "a" para la columna j, coloque en la otra hilera la misma "a" en la columna indicada
5. Si una hilera tiene sólo símbolos "a", entonces la descomposición tiene la propiedad no aditiva - de otra manera, no se tiene. Observe los dos ejemplos indicados abajo.

R= {SSN, ENAME, PNUMBER, PNAME, PLOCATION, HRS} D = {R1, R2}
R1 = EMP_LOCS = {ENAME, PLOCATION}
R2 = EMP_PROJ1 = {SSN, PNUMBER, HRS, PNAME, PLOCATION}
F = {SSN -> {ENAME}; PNUMBER -> {PNAME, PLOCATION};
{SSN, PNUMBER} - > HRS}

EJEMPLO 1.

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HRS
R1	b11	a2	b13	b14	a5	b16
R2	a1	b22	a3	a4	a5	a6

Después de aplicar el algoritmo, se observó que no existió cambio alguno.

EJEMPLO 2.

R = {SSN, ENAME, PNUMBER, PNAME, PLOCATION, HRS} D = {R1, R2, R3}
R1 = EMP = {SSN, ENAME}
R2 = PROJ = {PNUMBER, PNAME, PLOCATION}
R3 = WORKS_ON = {SSN, PNUMBER, HRS}
F = {SSN->{ENAME}; PNUMBER->{PNAME, PLOCATION};
{SSN, PNUMBER}-> HRS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HRS
R1	a1	a2	b13	b14	b15	b16
R2	b21	b22	a3	a4	a5	b26

R3	a1	b32	a3	b34	b35	a6

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HRS
R1	a1	a2	b13	b14	b15	b16
R2	b21	b22	a3	a4	a5	b26

		a2		a4	a5	
R3	a1	b32	a3	b34	b35	a6

Se muestra la matriz original y la matriz con los cambios adecuados posteriormente después de ser aplicado el algoritmo. Se observa que la última matriz tiene la propiedad de no aditividad de la reunión.

Ejercicio: Muestre si las siguientes relaciones tienen la propiedad no aditiva de la reunión natural:

UNO:

R = {ENAME, SSN, BDATE, ADDRESS, DNUMBER, DNAME, DMGRSSN}
D = {R1, R2}
R1 = ED1 = {ENAME, SSN, BDATE, ADDRESS, DNUMBER}
R2 = ED2 = {DNUMBER, DNAME, DMGRSSN}
F = {SSN-> {ENAME, BDATE, ADDRESS, DNUMBER};
DNUMBER -> {DNAME, DMGRSSN}}

DOS:

R = {PROPERTY_ID, COUNTY_NAME, #LOT, AREA, TAX_RATE, PRICE}
D= {R1, R2, R3}
R1 = LOTS1A={PROPERTY_ID, COUNTY_NAME, #LOT, AREA}
R2 = LOT1B={AREA, PRICE}
R3= LOTS2={COUNTY_NAME, TAX_RATE}
F={PROPERTY_ID->{COUNTY_NAME, #LOT, AREA}; AREA -> PRICE;
COUNTY_NAME->TAX_RATE}

7.4.2 PROBLEMAS CON VALORES NULOS. (dangling tuples)

Pueden existir problemas cuando existen tuplas con valores nulos en atributos que pueden ser usados para realizar una reunión. Para ilustrar esto, considere la base de datos mostrada enseguida:

EMPLOYEE			
ENAME	**SSN**	**BDATE**	**DNUM**
SMITTH	123456789	09/01/1955	5
WONG	333445555	08/12/1945	5
BENITEZ	886644444	09/01/1953	
BORG	888665555	10/11/2026	1
WALLACE	987654321	20/06/2021	4
BENGER	999775555	26/04/1955	
ZELAYA	999887777	19/07/1958	4

DEPARTMENT		
DNAME	**DNUM**	**DMGRSSN**
HEADQUARTERS	1	888665555
ADMINISTRATION	4	987654321
RESEARCH	5	333445555

SELECT ENAME, SSN, BDATE, DEPARTMENT.DNUM AS DNUM, DNAME, DMGRSSN
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DNUM=DEPARTMENT.DNUM;

PERDIDA POR VALORES NULOS					
ENAME	**SSN**	**BDATE**	**DNUM**	**DNAME**	**DMGRSSN**
SMITTH	123456789	09/01/1955	5	RESEARCH	333445555
WONG	333445555	08/12/1945	5	RESEARCH	333445555
ZELAYA	999887777	19/07/1958	4	ADMINISTRATION	987654321
WALLACE	987654321	20/06/2021	4	ADMINISTRATION	987654321
BORG	888665555	10/11/2026	1	HEADQUARTERS	888665555

Se observa que se han perdido en la consulta las tuplas de BENITEZ y BENGER, ya que tenían el valor de nulo en DNUM. (Observe que esto no viola ninguna restricción de integridad). Suponga que se quiere recuperar la lista de valores (ENAME, DNAME) de todos los empleados, como se observará, dos tuplas no aparecerán en la relación resultante. La operación OUTER JOIN puede resolver tal problema, si se realiza una operación LEFT OUTER JOIN de EMPLOYEE con DEPARTMENT, las tuplas con nulos de EMPLOYEE aparecerán como se muestra en la siguiente tabla:

SELECT ENAME, SSN, BDATE, DEPARTMENT.DNUM AS DNUM, DNAME, DMGRSSN
FROM EMPLOYEE LEFT JOIN DEPARTMENT ON EMPLOYEE.DNUM=DEPARTMENT.DNUM;

RESULTADO DEL LEFT JOIN					
ENAME	**SSN**	**BDATE**	**DNUM**	**DNAME**	**DMGRSSN**
SMITTH	123456789	09/01/1955	5	RESEARCH	333445555
WONG	333445555	08/12/1945	5	RESEARCH	333445555
ZELAYA	999887777	19/07/1958	4	ADMINISTRATION	987654321
WALLACE	987654321	20/06/2021	4	ADMINISTRATION	987654321
BENGER	999775555	26/04/1955			
BENITEZ	886644444	09/01/1953			
BORG	888665555	10/11/2026	1	HEADQUARTERS	888665555

En general, se debe tener un especial cuidado cuando un esquema de B. D's es diseñada teniendo dos o más relaciones y se interrelacionan vía una llave foránea. Esto puede causar una inexplicable pérdida de información en consultas en las que se envuelven reuniones. Además, si ocurren valores nulos en atributos tales como en SALARIO, sus efectos al construir funciones tales como SUM y AVG se deben evaluar con sumo cuidado.

7.5 DEPENDENCIAS MULTIVALUADAS

Las dependencias multivaluadas son una consecuencia de la 1NF donde prohíbe que un atributo de una tupla tenga un conjunto de valores. Si tenemos dos o más atributos multivaluados independientes en el mismo esquema de relación, nos enfrentaremos al problema de tener que repetir todos los valores de uno de los atributos con cada valor de otros atributos para que los ejemplares de la relación sigan siendo consistentes. Esta restricción se especifica como una dependencia multivaluada.
Ejemplo:

EMP		
nombre	**PNOMBRE**	**DEPENDIENTE**
SILVA	X	ANA
SILVA	X	JUAN
SILVA	Y	ANA
SILVA	Y	JUAN

Observe que PNOMBRE y DEPENDIENTE no están relacionados directamente entre sí. Para que las tuplas de la relación sean consistentes, debemos tener una tupla por cada una de las posibles combinaciones de DEPENDIENTE y de PNOMBRE de un empleado. Esta restricción se especifica como una dependencia multivaluada sobre la relación EMP. En términos informales, siempre que se mezclan dos vínculos con cardinalidad 1: N independientes A:B y A:C pueden producir una DMV.

DEFINICIÓN FORMAL:
Una DMV X->->Y sobre un esquema de relación R, donde X y Y son subconjuntos de R, específica la siguiente restricción sobre cualquier ejemplar r(R):

Si existen 2 tuplas T1 y T2 tales que T1[X]=T2[X], entonces deberán existir tuplas T3 y T4 en r(R) con las siguientes propiedades:

- T3[X]=T4[X]=T1[X]=T2[X]
- T3[Y]=T1[Y] y T4[Y]=T2[Y]
- T3[R-(XY)]=T2[R-(XY)] y T4[R-(XY)]=T1[R-(XY)]

En cualquier punto donde X->->Y se cumpla, se dice que X multidetermina a Y, siempre que X->->Y se cumple en R, también se cumple X->->(R-(XY)].
Una DMV X->->Y se denomina trivial si:

a. Y es un subconjunto de X
b. X **U** Y = R

EJEMPLO:

PROY_EMP	
NOMBRE	**NOMBREPROY**
SILVA	X
SILVA	Y

Tiene DMV trivial NOMBRE ->->NOMBREPROY. Una DMV es no trivial si no satisface a (a) o a (b). Una DMV trivial se cumple en cualquier ejemplar de relación r(R); es llamada trivial porque no especifica ninguna restricción sobre R.
Si se tiene una DMV trivial en una relación, tal vez se tengan que repetir valores en forma redundante en las tuplas. En la relación EMP, los valores X y Y de PNOMBRE se repiten con cada valor de DEPENDIENTE (y viceversa). Esta redundancia es indeseable. Sin embargo, el esquema está en 3NF. Por lo tanto, necesitamos definir una 4NF que no permita DMV.

DEFINICIÓN DE SUPERCLAVE:
Existen subconjuntos de atributos de un esquema de relación R con la propiedad de que no debe haber dos tuplas en un ejemplar de relación r(R) con la misma combinación de valores para estos atributos. Supongamos que denotamos un subconjunto de atributos con tales características SC; entonces, para cualquier 2 tuplas distintas T1 y T2 en un ejemplar de relación r(R), se tiene la siguiente restricción:

T1[SC]=T2[SC]

REGLAS DE INFERENCIA PARA D.F. y DMV.

Supongamos que todos los atributos están incluidos en un esquema de relación universal $R=\{A_1, A_2, \ldots A_n\}$ y que X, Y, Z, y W son subconjuntos de R, las siguientes reglas R1 a R8 forman un conjunto completo de dependencias funcionales y DMV:

R1 (Regla reflexiva para D.F.): Si X Y, entonces $X \rightarrow Y$
R2 (Regla de aumento para D. F) $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
R3 (Regla transitiva para D.F) $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
R4 (Regla de complemento para DMV) $\{X \rightarrow\rightarrow Y\} \models \{X \rightarrow\rightarrow (R-(XUY))\}$

R5 (Regla de aumento para DMV) Si X→→Y y W Z entonces WX→→YZ
R6 (Regla transitiva para DMV) {X→→Y, Y→→Z} ⊨X→→Y
R7 (De réplica (DF A DMV) {X→Y} ⊨X→→Y
R8 (Regla de combinación para DF y DMV) Si X→→Y y existen un W con las propiedades (a) W∩Y da como resultado el vacío, (b) W→Z y (c) Y Z, entonces X→Z.

Las reglas R1 a R3 son las reglas de Armstron. Las reglas R4 a R6 son reglas de inferencia de DMV y las reglas R7 y R8 son reglas relacionadas con DF y DMV. En particular, la regla 7 indica que una DF es un caso particular de DMV con la restricción adicional de que la lo máximo un valor de Y se asocia a cada valor de X. Dado un conjunto F de DF y DMV especificado sobre $R=\{A_1, A_2, \ldots A_n\}$, se pueden usar las reglas de inferencia de la R1 a la R8 para inferir completamente el conjunto de dependencias (DF y DMV) F^+ que cumplirá en todas las instancias de r en R que satisfacen a F.

7.5.1 4 N. F.

Un esquema de relación R está en 4. N. F. respecto a un conjunto de dependencias F si, para cada dependencia multivaluada no trivial X→→Y en F^+, X es una superllave de R. Ejemplo, la relación EMP:

EMP		
nombre	**PNOMBRE**	**DEPENDIENTE**
SILVA	X	ANA
SILVA	X	JUAN
SILVA	Y	ANA
SILVA	Y	JUAN

No se encuentra en 4 N. F. porque en las DMV no triviales NOMBRE→→PNOMBRE, y NOMBRE→→DEPENDIENTE, NOMBRE no es una superclave de EMP. Descomponemos EMP en EMP_PROY y EMP_DEP; estas dos nuevas relaciones están en 4 N. F. porque NOMBRE→→PNOMBRE es una DMV trivial en EMP_PROY y NOMBRE→→DEPENDIENTE es una DMV trivial.

Para ilustrar la importancia de mantener las relaciones en 4 N. F., se mostrará la tabla EMP con otros elementos más:

EMP		
nombre	**PNOMBRE**	**DEPENDIENTE**
BROWN	W	BOB
BROWN	W	JIM
BROWN	W	JOAN
BROWN	X	BOB
BROWN	X	JIM
BROWN	X	JOAN

EMP		
nombre	PNOMBRE	DEPENDIENTE
BROWN	Y	BOB
BROWN	Y	JIM
BROWN	Y	JOAN
BROWN	Z	BOB
BROWN	Z	JIM
BROWN	Z	JOAN
SMITH	X	ANNA
SMITH	X	JHON
SMITH	Y	ANNA
SMITH	Y	JOHN

Existen 16 tuplas en EMP. Si se descompone EMP en EMP_PROJECTS y EMP_DEPENDENTS,

EMP_PROJECTS	
NOMBRE	NOMBREPROY
BROWN	W
BROWN	X
BROWN	Z
SMITH	X
SMITH	Y

EMP_DEPENDENTS	
ENAME	DNAME
BROWN	BOB
BROWN	JIM
BROWN	JOAN
SMITH	ANNA
SMITH	JOHN

Se requiere guardar solo un total de 11 tuplas en ambas relaciones. Además, estas tuplas son más pequeñas que las tuplas en EMP. En adición, las anomalías por modificaciones asociadas con DMV se evitan. Por ejemplo, si BROWN inicia a trabajar en otro proyecto, se tienen que insertar otras tres tuplas en EMP (una por cada dependiente). Si olvidamos insertar alguna de las tuplas, la relaciones se vuelve inconsistente. Sin embargo, solo se requiere insertar una tupla en la relación EMP_PROJECTS. Ocurre un problema similar al tratar de eliminar o modificar una tupla si no se garantiza que la relación se encuentre en 4. N. F.

La relación EMP no está en 4. N. F. porqué representa dos tipos de vínculo independientes 1: N., una entre empleados y proyectos y la otra entre empleados y dependientes. A veces se tienen tres tipos de vínculo entre tipos de entidad que dependen de tres tipos de entidades participantes, tal como se muestra en la relación SUMINISTRA:

SUMINISTRA		
SNAME	**PARTNAME**	**PROJNAME**
ADAMSKY	BOLT	PROJY
ADAMSKY	NAIL	PROJX
SMITH	BOLT	PROJX
SMITH	NUT	PROJY
WALTON	NUT	PROJZ

En este caso una tupla representa una específica parte a suministrar en un proyecto, por lo que no existe DMV no trivial. La relación SUMINISTRA está en 4. N. F. por lo que no debe ser descompuesta. Observe que las relaciones conteniendo DMV no triviales tienden a ser todos los atributos la llave de la relación.

DESCOMPOSICIÓN A 4. N. F. SIN PEDIDA EN LA RELACIÓN.

Siempre que se descompone un esquema de relación R en R_1= (XUY) y R_2 = (R - Y) basado en una DMV X→→Y que cumple en R, se tiene la propiedad de descomposición sin perdida en la reunión. Se puede demostrar que esta es una condición necesaria y suficiente para descomponer un esquema en dos que tengan la propiedad de reunión sin pérdida marcando lo siguiente:

> Los esquemas de relación R_1 y R_2 forman una descomposición sin pérdida en la reunión R si y sólo si $(R_1 \cap R_2) \rightarrow\rightarrow (R_1 - R_2)$ (o por simetría, si y Sólo si $(R_2 \cap R_1) \rightarrow\rightarrow (R_2 - R_1)$).

ALGORITMO PARA DESCOMPONER UNA RELACIÓN EN 4 . N. F.

D = { R }
Mientras exista un esquema Q en D que no es en 4 N. F. realice:
    Escoger un esquema de relación Q en D que no esté en 4. N. F;
    Encuentre una DMV no trivial X→→Y en Q que viole la 4. N. F.;
    Reemplace Q en D por dos esquemas (Q – Y) y (X U Y)
Fin.

Ejemplo:

R = {Matricula, Dnombre, Pnombre}
    {Matricula}→→ {Pnombre} (X→→Y)
R_1 = {Matricula, Pnombre} (X U Y)
R_2 = {Matricula, Dnombre} (R – Y)

$R_1 \cap R_2$ = {Matricula} y $R_1 - R_2$ = {Pnumero}

De esta forma:

$(R_1 \cap R_2) \rightarrow\rightarrow (R_1 - R_2)$

{Matricula}→→ {Pnombre}

Por lo que R_1 y R_2 cumplen la propiedad de descomposición sin pérdida en la reunión.

7.5.2 5. N. F.

En algunos casos, puede existir la descomposición sin perdida en la reunión en dos esquemas de relación sino que puede existir la propiedad en más de dos esquemas de relación. Este caso es manejado en la 5. N. F. Es importante indicar que este caso ocurre raramente y en la práctica es difícil de detectar.

Una dependencia de reunión (DR), denotada por DR(R_1, R_2, R_3,…, R_N}, especificada sobre el esquema de relación R, indica la restricción sobre los ejemplares r(R). La restricción establece que toda instancia legal r(R) debe tener la reunión sin perdida cuando se realiza la descomposición R_1, R_2, R_3,…, R_N ; esto es:

$$*(\Pi_{<R1>}, \Pi_{<R1>}, \Pi_{<R1>},\ldots, \Pi_{<Rn>}(r))=r$$

Observe que DMV es un caso especial de DR cuando n=2. Una dependencia en reunión DR(R_1, R_2, R_3,…, R_N} especificada sobre la relación R es DR **trivial** si uno de los esquemas R_i en DR(R_1, R_2, R_3,…, R_N} es igual a R. Tal dependencia se indica como trivial porque posee la propiedad de reunión sin pérdida para cualquier ejemplar de la relación r de R, por lo tanto, no especifica ninguna restricción sobre R.

Un esquema R está en 5. N. F. con respecto a u n conjunto F de dependencias funcionales, multivaluadas y de reunión si, para toda dependencia de reunión no trivial DR(R_1, R_2, R_3,…, R_N} en F^+ toda R_i es una súper llave de R.

Por ejemplo, considere la relación suministra y se incluyen las siguientes restricciones: Donde quiera que un abastecedor A abastece la parte P, y un proyecto J usa la parte P, y el abastecedor A abastece al menos una parte al proyecto J, entonces el abastecedor A también abastecerá la parte P al proyecto J. Esta restricción puede ser reestructurada especificando DR(R_1, R_2, R_3) sobre las tres proyecciones R1(Abastecedor, Parte), R2(Abastecedor, proJecto), y R3(Parte, proJecto) de Abastecer. Si estas restricciones se cumplen, las tuplas debajo de la línea punteada deben existir en cualquier instancia legal de ABASTECER. R1, R2, y R3 están en 5. N. F. Note que la reunión natural de dos de estas relaciones producen tuplas espurias, pero si se aplica la reunión natural a las tres no sucede tal fenómeno. Esto se debe porque existe DR, pero no están especificadas las DMV.

Descubrir DR en la práctica con cientos de atributos es muy difícil, de aquí que en la práctica, los diseñadores de bases de datos le dan una escasa atención a ellos.

SUMINISTRA		
SNAME	**PARTNAME**	**PROJNAME**
ADAMSKY	BOLT	PROJY
ADAMSKY	NAIL	PROJX
SMITH	BOLT	PROJX
SMITH	NUT	PROJY
WALTON	NUT	PROJZ
ADAMSKY	BOLT	PROJX
SMITH	BOLT	PROJY

R1	
SNAME	**PARTNAME**
ADAMSKY	BOLT
ADAMSKY	NAIL
SMITH	BOLT
SMITH	NUT
WALTON	NUT

R2	
SNAME	**PROJNAME**
ADAMSKY	PROJX
ADAMSKY	PROJY
SMITH	PROJX
SMITH	PROJY
WALTON	PROJZ

R3	
PARTNAME	**PROJNAME**
BOLT	PROJX
BOLT	PROJY
NAIL	PROJX
NUT	PROJY
NUT	PROJZ

```
SELECT R1.SNAME, PARTNAME, PROJNAME
FROM R1, R2
WHERE R1.SNAME=R2.SNAME;
```

R1*R2		
SNAME	**PARTNAME**	**PROJNAME**
SMITH	BOLT	PROJX

R1*R2		
SNAME	**PARTNAME**	**PROJNAME**
SMITH	NUT	PROJX
SMITH	BOLT	PROJY
SMITH	NUT	PROJY
ADAMSKY	BOLT	PROJY
ADAMSKY	NAIL	PROJY
WALTON	NUT	PROJZ
ADAMSKY	BOLT	PROJX
ADAMSKY	NAIL	PROJX

```
SELECT SNAME, R1.PARTNAME, PROJNAME
FROM R1, R3
WHERE R1.PARTNAME=R3.PARTNAME;
```

R1*R3		
SNAME	**PARTNAME**	**PROJNAME**
SMITH	BOLT	PROJX
SMITH	BOLT	PROJY
SMITH	NUT	PROJY
SMITH	NUT	PROJZ
ADAMSKY	BOLT	PROJX
ADAMSKY	BOLT	PROJY
WALTON	NUT	PROJY
WALTON	NUT	PROJZ
ADAMSKY	NAIL	PROJX

```
SELECT SNAME, R2.PROJNAME, PARTNAME
FROM R2, R3
WHERE R2.PROJNAME=R3.PROJNAME;
```

R2*R3		
SNAME	**PROJNAME**	**PARTNAME**
ADAMSKY	PROJX	BOLT
SMITH	PROJX	BOLT
ADAMSKY	PROJY	NUT
SMITH	PROJY	NUT
ADAMSKY	PROJY	BOLT
SMITH	PROJY	BOLT
WALTON	PROJZ	NUT
ADAMSKY	PROJX	NAIL
SMITH	PROJX	NAIL

```
SELECT R1.SNAME, R1.PARTNAME, R2.PROJNAME
FROM R1, R2, R3
WHERE R1.SNAME=R2.SNAME AND R2.PROJNAME=R3.PROJNAME
AND R1.PARTNAME=R3.PARTNAME;
```

R1*R2*R3		
SNAME	**PARTNAME**	**PROJNAME**
SMITH	BOLT	PROJX
SMITH	BOLT	PROJY
SMITH	NUT	PROJY
ADAMSKY	BOLT	PROJX
ADAMSKY	BOLT	PROJY
WALTON	NUT	PROJZ
ADAMSKY	NAIL	PROJX

RESUMIENDO

PRIMER FORMA NORMAL:
Se dice que una relación se encuentra en 1FN cuando cada atributo sólo toma un valor del dominio simple subyacente. Es decir, que no existen grupos repetitivos.

Un ejemplo es el que aparece en las tablas siguientes, donde en la primera tabla no se encuentra en 1FN, sin embargo en la segunda tabla sí se cumple ésta restricción.

Nombre	***Titulaciones***
Juan	Informática
	Contador
Antonia	Música

Nombre	***Titulaciones***
Juan	Informática
Juan	Contador
Antonia	Música

SEGUNDA FORMA NORMAL:
Se dice que una relación se encuentra en 2FN si.

- Se encuentra en 1FN
- Cada atributo no primo tiene dependencia funcional completa respecto de alguna de las claves.

Por ejemplo, teniendo una R(ATR, DEP), donde ATR son los atributos de la relación y DEP son las dependencias que tiene la relación, se tiene el siguiente esquema de relación R({A,B,C,D}{A,B->C; A->D}), donde la clave candidata es el conjunto {A,

B}, y por lo tanto los atributos primos son A y B y el atributo no primo es D, depende de A, pero no de una llave. Por lo tanto, el esquema no se encuentra en 2FN.
Sin embargo los esquemas R1 ({A, B, C), {A, B->C}) y R2 ({A,D},{A->D}) si se encuentran en 2FN.

TERCERA FORMA NORMAL.
Se dice que una relación se encuentra en 3FN si:

- Se encuentra en 2FN.
- No existe ningún atributo no primo que dependa transitivamente de alguna llave de R.

Por ejemplo, en el esquema de relación R {(A, B, C},{A->B, B->C}), la clave candidata de la relación es el atributo A, y los atributos no primos B y C. Como el atributo C depende transitivamente de la clave A, la relación no se encuentra en 3FN, aunque sí en 2FN., ya que C depende transitivamente de la clave.
Sin embargo, las relaciones R1 ({A, B}, {A->B}) y R2({B, C}, {B->C}) se encuentran en 3FN.

FORMA NORMAL Boyce Codd (FNBC)

Se dice que una relación se encuentra en FNBC si y sólo si todo determinante es clave candidata (Si X->Y, al conjunto X se le denomina implicante o determinante y al conjunto Y se denomina implicado).
Por ejemplo, la relación R ({A,B},{A<- ->B;A, C ->D}) no se encuentra en FNBC, ya que los determinantes A y B no son claves candidatas, sino que lo son los conjuntos {A,C} y {B, C}
Sin embargo, la relación R1 ({A, B},{A<- ->B}) Y R2({A,C,D},{A,C->D}) si se encuentran en FNBC.

CUARTA FORMA NORMAL (4FN)

Se dice que una relación está en 4FN si y sólo si las únicas dependencias multivaluadas no triviales son aquellas en las que una clave multideternima un atributo
Por ejemplo, la relación R({A,B,C},{A->->B, A->->C}) no se encuentran en 4FN ya que su clave candidata es el conjunto {A,B,C} y en la primer dependencia el atributo C no participa y en la segunda es el atributo B el que no participa.
Sin embargo, las relaciones R1 A->->B}) y R2({A, C},{A->->c}) si se encuentran en 4FN.

QUINTA FORMA NORMAL (5FN)

Se dice que una relación se encuentra en 5FN si y sólo si:

- Se encuentra en 4FN
- Toda dependencia de combinación está implicada por una clave candidata.

También se puede dar otra definición para la 5FN: "una relación se encuentra en 5FN si y sólo si toda dependencia funcional, multivaluada o de combinación no trivial es consecuencia de las claves candidatas".

7.6 ALGORITMO PARA EL CÁLCULO DEL RECUBRIMIENTO MNIMAL.

Un conjunto de dependencias es mínimo y se denota como DEP^m, cuando se cumplen las siguientes condiciones:

a) Todas sus dependencias son elementales, es decir, que todas sus dependencias sean plenas, no triviales y tengan un único atributo implicado.
b) No existe en ninguna de las dependencias atributos extraños, Un atributo A perteneciente a X es extraño en la dependencia X->Y si la dependencia (X-A)->Y se deduce del resto de las dependencias de la relación mediante los axiomas de Armstrong.
   Por ejemplo, sea el esquema de relación R ({A, B, C} {A->C, AB->C})
   Como el atributo A implica funcionalmente a C, el atributo B es redundante en la segunda dependencia.
c) No existe en la relación ninguna dependencia redundante. Una dependencia *dp* se dice que es redundante si sus implicados se deducen a partir del resto de las dependencias de la relación.
   Por ejemplo, sea R ({A, B, C} {A->B, B->C, A->C}). Como el atributo A determina transitivamente el atributo C, la última dependencia (A->C) es una dependencia redundante.

Puede ser que existan varios conjuntos de dependencias mínimas válidas, ya que todo depende de la elección de las dependencias redundantes y de los atributos extraños que se eliminan en el proceso del cálculo del conjunto minimal.

7.7 METODOS DE ANÁLISIS Y SÍNTESIS PARA LA NORMALIZACIÓN.

Los métodos de análisis (o descomposición) y síntesis tratan de transformar, por medio de proyecciones sucesivas, un esquema de relación en un conjunto de n esquemas resultantes. Estos esquemas siempre han de cumplir la conservación de la información (atributos y tuplas), de las dependencias y que existan mínimas redundancias en los datos.

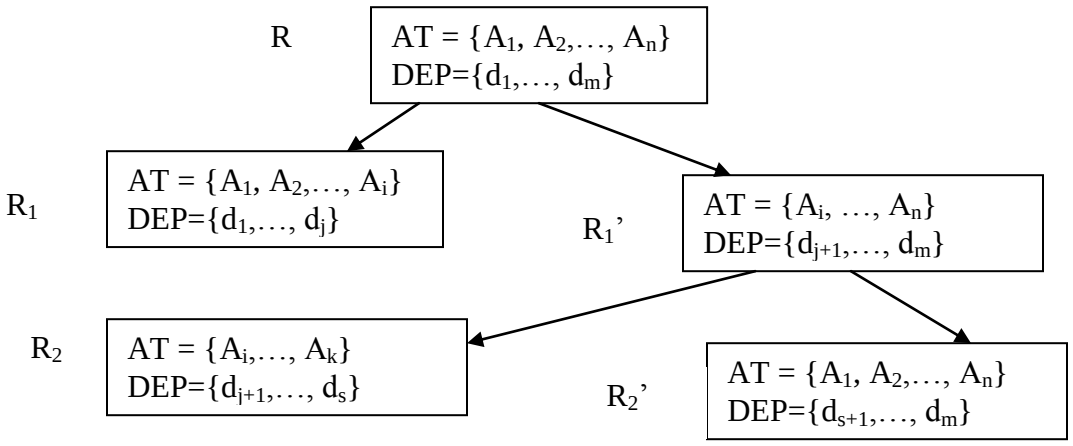
De esta forma siempre los n esquemas resultantes serán equivalentes a la relación inicial y tendrán menor redundancia en los datos.

Se aplican estos métodos para aumentar el nivel de normalización de las relaciones, eliminando redundancias y anomalías en las modificaciones, inserciones y borrados de los datos de la misma.

7.7.1 ANÁLISIS.

El método de análisis fue propuesto por Codd y posteriormente fue ampliado por Russanen, Fagir y Zaniolo para incluir las dependencias multivaluadas y en combinación. Trata la descomposición en sucesivas proyecciones de la relación universal (con todos los atributos del universo del discurso y las dependencias existentes entre ellos) paso a paso hasta alcanzar la 5FN.

Los esquemas resultantes son cada vez de menor grado y se encuentran en un nivel de normalización mayor.
El proceso se lleva a cabo mediante el *árbol de descomposición*, que se muestra en la siguiente figura, donde $A_1, A_2, \ldots, A_n$ son los atributos y $d_1, \ldots, d_n$, las dependencias que existen entre los mismos.



El proceso de descomposición termina cuando las relaciones se encuentran en forma normal objetivo o cuando la continuación del proceso supone pérdida no deseada de dependencias.

Los pasos del algoritmo de análisis son los siguientes:

1. Hallar el recubrimiento minimal DEP^m.
2. Hallar las claves, los atributos principales y los no principales.
3. Identificar la FN en la que se encuentra la relación.
4. Agrupar las dependencias funcionales (DF) que tiene el mismo implicante o equivalentes y obtener proyecciones independientes con estos grupos de dependencias.
5. Tratar las dependencias multivaluadas (descomponerlas o eliminarlas)
6. Tratar las dependencias en combinación. Si existe alguna que no esté implicada por la clave, habrá que descomponerla.

En el paso cuarto del algoritmo es importante el orden en el que se eligen las dependencias funcionales de una relación para su descomposición en otras relaciones.

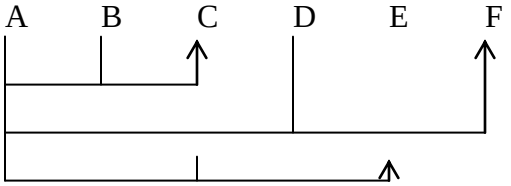
Esto se debe a que los atributos no principales de la nueva relación creada con la/s dependencia/s elegida/s no se mantienen en la relación temporal creada para continuar la descomposición.
Debido a esto es posible que se pierdan dependencias funcionales en la descomposición de la relación, cosa que no siempre es deseable.
Por otro lado, cuando existen varias dependencias en un momento dado que no inducen a pérdidas, la elección de una u otra es independiente del resultado. Es decir, pueden existir distintos árboles de descomposición, y distintos resultados válidos para una misma relación inicial.
En los árboles de descomposición de los ejercicios resueltos la notación utilizada es la siguiente:

- R: relación universal.
- AT: conjunto de atributos de una relación
- DEP: conjunto de dependencias de una relación
- R_1, R_2,…, R_n: Relaciones temporales necesarias para el proceso de descomposición de una relación R.
- R_1', R_2',…, R_n', R_{1aux}, R_{2aux},…, R_{naux}: Relaciones temporales necesarias para el proceso de descomposición de una relación.

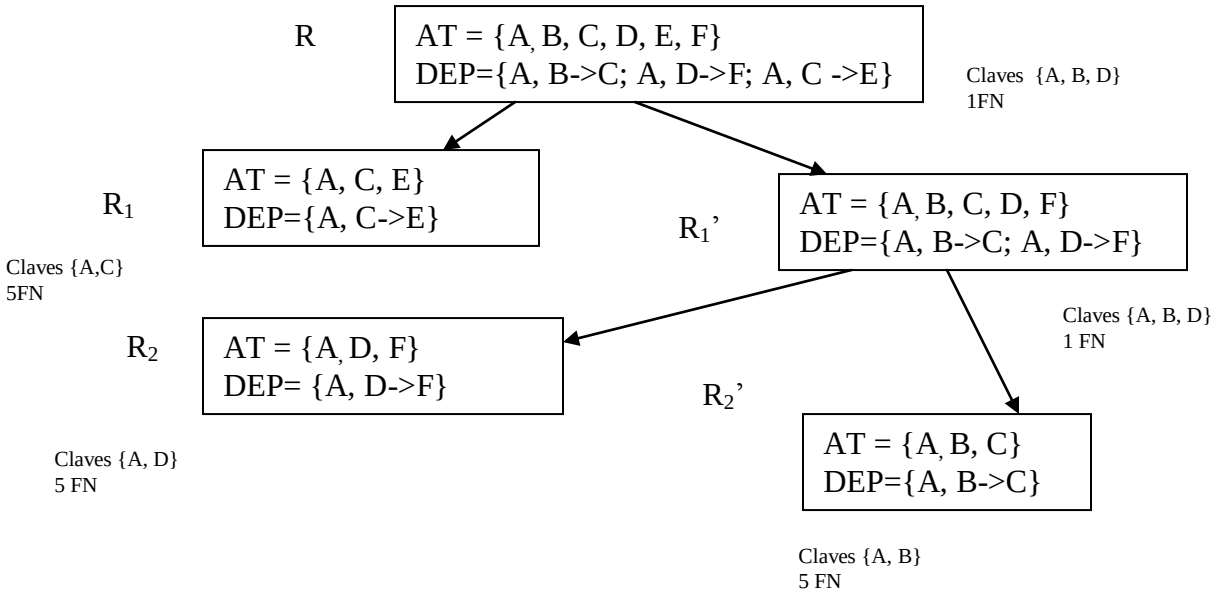
Ejemplo:
R (AT, DEP), donde.

AT={A,B,C,D,E,F}
DEP^m = {A, B->C, A,D->F, A,C->E}

En la relación R, la clave candidata es el conjunto {A, B, D}, y por lo tanto, AP={A, B, D} y ANP={C, E, F}. Se encuentran en 1FN, ya que, por ejemplo, el atributo C en la primera dependencia sólo depende de los atributos {A, B}, pero no de toda la clave.



En AP no se coloca como parte de la AP a C porque C depende en forma directa de {A, B}.
El árbol de descomposición de esta relación se muestra en la siguiente figura:



7.7.2 SINTESIS.

El proceso de síntesis recorre el camino inverso al proceso de análisis. Trata de agrupar las distintas dependencias que existen entre los atributos del universo del discurso, sintetizando relaciones.
Es decir, ambos procesos tienen el mismo fin, apoyándose ambos en el mismo concepto de dependencias y de recubrimiento irredundante.
Sin embargo, el proceso de síntesis no trata las dependencias multivaluadas ni las dependencias en combinación, por lo que solamente asegura que las relaciones resultantes puedan estar en FNBC. Frente al proceso de análisis, que tiene en cuenta estas dependencias y por lo tanto permite llegar hasta la 5FN. Por supuesto, todo esto teniendo en cuenta que no siempre se pueda llegar sin pérdida de dependencias ni de información.
Los pasos del algoritmo de síntesis se resumen a continuación:

1. Hallar el recubrimiento minimal DEP^m.
2. Agrupar las dependencias funcionales (DF) en particiones que tienen el mismo implicante, uniendo los atributos equivalentes.
3. Crear un esquema de relación (R_i) para cada partición, que tenga como atributos todos los que participen en las dependencias y como grupo de dependencias las del grupo.
4. Si existen atributos que no son implicantes ni implicados en DEP^m, formar un esquema de relación con éstos sin dependencias o alternativamente crear un esquema con la clave de la relación y sin dependencias.

Se realiza el proceso de síntesis para el mismo caso que se resolvió mediante análisis, es decir, para la relación universal R{AT, DEP}, donde:
AT={A,B,C,D,E,F}
DEP^m= {A, B->C;A, D->F; A,C->E}
Se crea un Nuevo esquema de relación por cada conjunto de dependencias que poseen el mismo implicante. En este caso los grupos tienen cardinalidad uno, ya que no existen dos dependencias con el mismo determinante. Por lo tanto, el esquema de relación final es:

R1 ({A, B, C}, {A,B->C})
R2 ({A, D, F},{A, D->F})
R3 ({A, C, E}, {A, C->E})

En este caso las relaciones obtenidas son las mismas que con el proceso de análisis y todas ellas se encuentran en 5FN.

7.7.3 EJEMPLO:

Enunciado.
Describir de forma detallada (preferentemente mediante un diagrama), todos los pasos que permitan determinar el nivel de normalización de un esquema de relación y avanzar lo máximo posible en su normalización mediante el proceso de análisis.
Aplique el proceso descrito anteriormente para determinar el nivel de normalización del siguiente supuesto, así como para obtener un esquema relacional en la forma normal que considere más conveniente.
Supuesto.
Se desea diseñar una base de datos en el modelo relacional para una universidad, teniendo los siguientes supuestos semánticos:
Un profesor se identifica por un código de profesor (CP) y todos los profesores tienen un nombre (NP) distintos. Un profesor puede tener varios títulos (T) e intervenir en distintos proyectos (P), no exigiéndole ningún título determinado para intervenir en un cierto proyecto.
Cada asignatura (A) tiene un único profesor como responsable, si bien un mismo profesor puede ser responsable de más de una asignatura. Las asignaturas se dividen en uno o más grupos (G). Todos los alumnos (AL), en cada asignatura, pertenece a un único grupo.
Cada profesor depende siempre y únicamente de un departamento (D). Así mismo, toda asignatura está ligada a un único departamento, el del profesor responsable de la misma.

Discusión del enunciado.

Nivel de Normalización.
Al enfrentarse a un problema en el que se pide hallar el nivel de normalización de relaciones, el primer paso, antes de calcular las claves primarias de la relación, es comprobar que el conjunto de atributos y dependencias está en, al menos, primera forma normal, es decir, que se trata de una relación. Esto se debe a que una de las restricciones inherentes al modelo relacional es que el conjunto de atributos y dependencias se encuentre en 1FN.
Una vez que se tiene la relación en 1FN, se comprueba si se encuentra en los siguientes niveles de normalización. Para ello se necesita saber primero cuáles son sus atributos principales y no principales. Por lo tanto, se han de calcular las dependencias funcionales, multivaluadas y de combinación existentes entre sus atributos, el recubrimiento minimal (o irredundante) de la relación, las claves y los atributos principales y no principales de la misma.
La relación se encuentra en 2FN si se encuentra en 1FN y cumple que cada atributo no principal tiene dependencia funcional completa respecto de cada una de las claves.

Se encuentra en 3FN cuando está en 2FN y no existe ningún atributo no principal que dependa transitivamente de alguna clave de la relación.
Se encuentra en FNBC cuando todo determinante es clave candidata de la relación.
La relación se encuentra en 4FN cuando está en 3FN y cumple que las únicas dependencias multivaluadas no triviales viene determinadas por claves candidatas.
Por último, la relación está en 5FN cuando toda dependencia funcional multivaluada o de combinación no trivial es consecuencia de las claves candidatas.
Cuando una relación no se encuentra en la forma normal deseada, se puede descomponer (mediante las técnicas de análisis o síntesis) en varias con un nivel de normalización más alto (aunque en ocasiones esto puede suponer pérdida de dependencias entre los atributos de la relación).

Normalización del supuesto.
Se procede a determinar el nivel de normalización del supuesto y se normaliza mediante análisis (o descomposición) hasta la forma normal que se cree más conveniente.

Grupos repetitivos.
En el supuesto no existen grupos repetitivos, luego, la relación se encuentra al menos en 1FN.

Dependencias entre atributos.
En primer lugar, se parte de una base de datos de la universidad, donde, en principio, solamente existe una relación con todos los atributos y dependencias entre los mismos (relación universal). Para ello se analizan los atributos de la relación.
Se analiza el caso, determinando los atributos y las dependencias funcionales, multivaluadas y de combinación que existen entre los atributos de la relación. Para realizar un análisis detallado, se introducen referencias al enunciado.

> *Un profesor se identifica por un código de profesor (CP) y todos los profesores tienen nombres (NP) distintos.*

Esta frase indica que la entidad *profesor* se desean guardar los atributos CP y NP. Donde CP identifica al profesor, tratándose de su clave primaria, y por tanto determina funcionalmente al nombre:

$$CP \rightarrow NP$$

Pero también se indica que no existen dos profesores con el mismo nombre. Por lo tanto, el nombre también determina funcionalmente al código (actuaría como una clave alternativa en el modelo relacional). De esta forma, también se da la dependencia funcional:

$$NP \rightarrow CP$$

> *Un profesor puede tener varios títulos (T) e intervenir en distintos proyectos (P) no exigiéndose ningún título determinado para intervenir en un cierto proyecto.*

En este caso, indica que los profesores pueden poseer varios títulos (conjunto bien definido de valores), luego, en este caso no existe ninguna dependencia funcional entre CP y T, sino que se trata de una dependencia multivaluada. También multivaluada a los distintos proyectos en los que interviene el profesor. Como no tiene ninguna relación el título y el proyecto para un determinado profesor, se tiene la dependencia multivaluada

jerárquica CP →→ T | P, donde entre el proyecto y el titulo no existe ninguna dependencia funcional (T→ P y P → T):

$$CP \twoheadrightarrow T \mid P$$

Cada asignatura (A) tiene un único profesor como responsable, si bien un mismo profesor puede ser responsable de más de una asignatura.

Por otro lado, para cada asignatura existe un único profesor que se responsabiliza de la misma, por lo tanto, como para cada valor de la asignatura existe un único valor para profesor responsable, existe una dependencia funcional A → CP. También se da que el profesor puede ser responsable de más de una asignatura, de forma que, para un único valor de P no existe un único valor de A, por lo tanto, no existe la dependencia funcional entre los proyectos y las asignaturas. (CP $\not\rightarrow$ A):

$$A \rightarrow CP$$

Las asignaturas se dividen en uno o más grupos (G). Todo alumno (AL), en cada asignatura, pertenece a un único grupo.

Las asignaturas se dividen en grupos, un grupo sólo puede pertenecer a una asignatura, de forma que, para cada valor de grupo existe un único valor de asignatura, luego existe otra dependencia funcional entre asignatura y grupo:

$$G \rightarrow A$$

Además, todo alumno, en una asignatura pertenece a un único grupo, luego, para cada valor del par (AL, A), se tiene un único valor de G, de forma que existe otra dependencia funcional entre estos tres atributos:

$$AL, A \rightarrow G$$

Cada profesor depende siempre y únicamente de un departamento (D). Así mismo, toda asignatura está ligada a un único departamento, al del profesor responsable de la misma.

Por último, se sabe que un profesor pertenece a un único departamento, luego (CP → D) y toda asignatura pertenece aun departamento, de forma que (A → D):

$$CP \rightarrow D$$
$$A \rightarrow D$$

Resumiendo, la relación universal resultante del análisis del caso es R (AT, DEP), donde:

AT = {CP, NP, T, P, A, G, AL, D}

DEP= {

1. CP → NP
2. NP → CP
3. A → CP
4. G → A
5. AL, A → G
6. CP → D
7. A → D
8. CP →→ T | P

}

Recubrimiento minimal.
En este apartado, se comprueba que el conjunto de las dependencias halladas en el apartado anterior (DEP) es irredundante. Para ello se ha de comprobar que todas las dependencias son elementales, no existe ningún atributo extraño en ninguna de ellas y no existen dependencias redundantes.

Dependencias elementales.
Lo primero es comprobar que todas las dependencias son elementales.

- En todas las dependencias se cumple que tienen un único atributo implicado.
- Todas las dependencias son plenas, es decir, que ningún subconjunto del determinante puede implicar al implicado.
- Ninguna dependencia es trivial, es decir, ningún atributo de determinante aparece como implicado.

Por lo tanto, todas las dependencias son elementales.

Atributos extraños.
A continuación se comprueba que no existen atributos extraños en las dependencias. Un atributo A es extraño si dada la dependencia $X \rightarrow Y$ de DEP, la dependencia $(X - A) \rightarrow Y$ se encuentra en DEP'.
Para comprobar que la dependencia $(X - A) \rightarrow Y$ pertenece a DEP' se ha de cumplir que Y pertenece al cierre de X-A ($(X - A)^*_{DEP}$).
En el conjunto de dependencias, DEP, en la única dependencia donde podrían existir atributos extraños es en (AL, $A \rightarrow G$), ya que tiene más de un atributo como determinante.
Para ver si A es un atributo extraño, se comprueba que $AL \rightarrow G$ no se encuentra en DEP'.
Para ello se calcula el cierre de AL:

- $AL^+ = \{AL\}$, ya que el atributo por sí sólo no determina a ningún otro.

Como el atributo G no se encuentra en el cierre de AL, se puede concluir que A no es un atributo extraño. Para comprobar si AL es atributo extraño se halla el cierre de A:

- $A^+ = \{A, CP, D\}$, pero no determina G.

Por lo tanto, AL tampoco es un atributo extraño.

Dependencias redundantes.
Por último se comprueba s existen dependencias redundantes en el conjunto DEP. Se dice que una dependencia *dp* es redundante si puede derivarse del conjunto DEP' = {DEP–*dp*}
Por lo tanto, por cada dependencia de DEP, se determina si se puede derivar del resto (DEP'). No se resuelve completamente el ejercicio, debido a que el cálculo es repetitivo, sólo se indicarán las dependencias redundantes y porqué lo son:
La dependencia $A \rightarrow D$ es una dependencia redundante, ya que se puede deducir a partir del conjunto de dependencias $\{DEP - \{A \rightarrow D\}\}$:

- $(A)^+_{DEP} = \{A, CP, NP, D\}$
  - (3): $A \rightarrow CP$, luego $(A)^+_{DEP} = \{A, CP\}$
  - (1): $CP \rightarrow NP$, luego $(A)^+_{DEP} = \{A, CP, NP\}$
  - (6): $CP \rightarrow D$, luego $(A)^+_{DEP} = \{A, CP, NP, D\}$
- Como D se determina a partir de DEP', la dependencia (7) es redundante.

A partir de todos estos cálculos, se puede concluir que uno de los conjuntos mínimos de dependencias posibles es el siguiente:

DEP^m = {

1. CP → NP,
2. NP → CP ,
3. A → CP,
4. G → A,
5. AL, A → G,
6. CP → D,
7. CP →→ T | P

}

Claves candidatas de la relación.

El algoritmo de cálculo de si un descriptor (subconjunto de atributos AT) es clave candidata se basa en el cálculo de su cierre transitivo.

En primer lugar se observa que existen dos atributos multivaluados, T y P, por lo que forman parte de cualquier clave candidata al no ser implicados funcionalmente por ningún otro atributo.

A continuación, se calcula el cierre transitivo de todos los atributos de relación:

- $(CP)^+_{DEP}$ = {CP, NP, D}
- $(NP)^+_{DEP}$ = {A, CP, NP}, CP es equivalente a NP.
- $(A)^+_{DEP}$ = {A, CP, NP, D}
- $(G)^+_{DEP}$ = {G, A, CP, NP, D}
- $(AL)^+_{DEP}$ = {AL}
- $(D)^+_{DEP}$ = { D}
- $(T)^+_{DEP}$ = {T}
- $(P)^+_{DEP}$ = {P}

Hasta este momento, ningún atributo simple implica transitivamente a todos los atributos de la relación (AT), por lo que se realizan grupos de atributos que puedan determinar a todo el conjunto AT, eligiendo aquellos atributos con mayor cardinalidad en su cierre transitivo:

- $(T, P, G)^+$ = {T, P, G, A, CP, NP, D}, donde todavía no se recoge el atributo AL.

Por lo tanto, una clave candidata es el conjunto formado por {T, P, G, AL}, ya que no existe ningún subconjunto de la misma que a su vez implique a todo AT. Po otro lado, como la dependencia (5) de DEP (AL, A → G), donde los atributos AL y A implican a G, el conjunto {T, P, A, AL} también es clave candidata.

Resumiendo los cálculos anteriores:

- Claves candidatas: {T, P, G, AL} y {T, P, A, AL}
- Atributos principales (forman parte de alguna clave candidata): {T, P, G, A, AL}
- Atributos no principales (AT - atributos principales): {CP, NP, D}

Forma Normal de la relación

- **1FN:** La relación se encuentra en 1FN al no poseer grupos repetidos.
- **2FN:** La relación no se encuentra en 2FN, ya que existen atributos no principales que no dependen plenamente de la clave. Por ejemplo: en la dependencia (CP→NP), el atributo NP no depende de ninguna clave principal (ni siquiera transitivamente).

Normalización por análisis.

Se va a normalizar la relación R mediante el proceso de análisis, construyendo al mismo tiempo el árbol de descomposición. Se detalla paso a paso el algoritmo:

*Paso 1: Recubrimiento minimal (*calculado anteriormente)

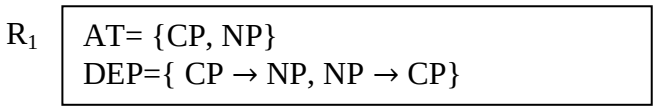
DEP^m = {

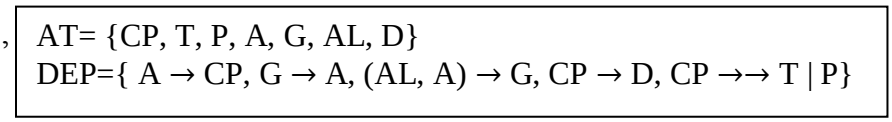
1. CP → NP,
2. NP → CP ,
3. A → CP,
4. G → A,
5. AL, A → G,
6. CP → D,
7. CP →→ T | P

}

Paso 2: Descomposición de R

El primer paso es identificar los atributos equivalentes. Los atributos CP y NP son equivalentes,, que CP → NP, NP → CP. Por lo tanto, se agrupan estas dependencias funcionales y se crea una relación R_1 cuyos atributos son los que participan en ambas dependencias. Al mismo tiempo, se crea una relación R_1' y las dependencias son las de {R - R_1). Gráficamente se muestran las descomposiciones en al siguiente figura.





A continuación se realiza el cálculo de la forma normal en la que se encuentra la relación R_1, para ello, primero se hallan las claves candidatas:

Las claves candidatas de la relación R_1 son {CP} y {NP}

El cálculo de la forma normal en la que se encuentra la relación se detalla a continuación:

- 1FN: si, al tratarse de una relación.
- 2FN y 3FN: Si, ya que no existen atributos no principales.
- FNBC: Si, ya que todo determinante (o implicante) es clave candidata.

- 4FN y 5FN: Si, ya que no existen dependencias multivaluadas ni en combinación en R_1

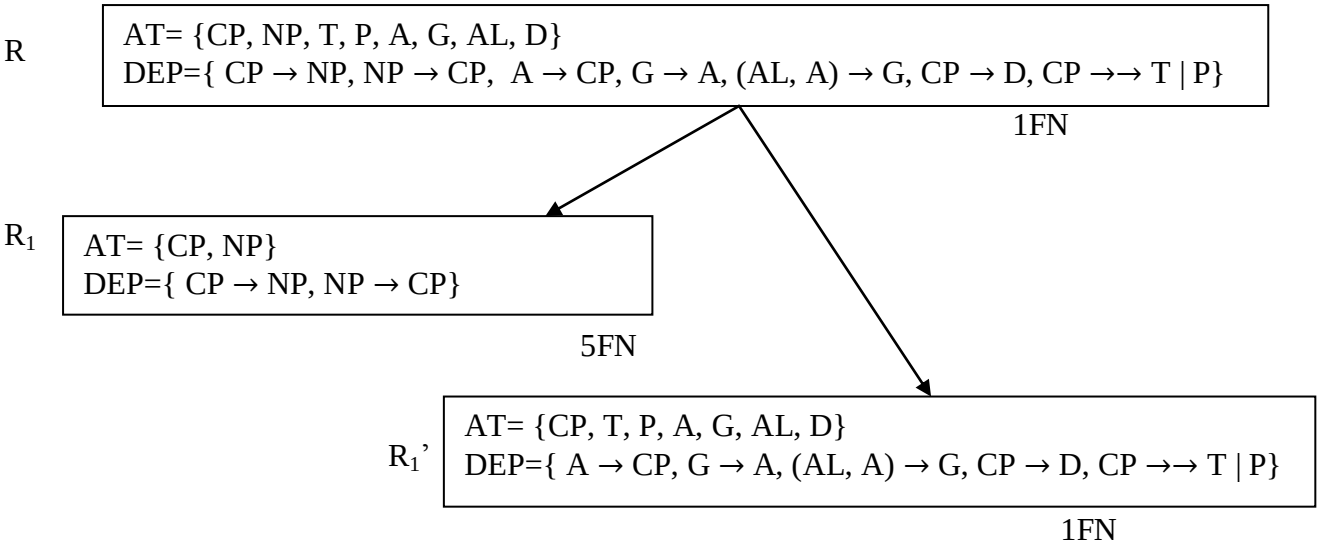
A continuación se calculan las claves candidatas y forma normal en la que se encuentra la relación R_1'.
La clave de R_1' se calcula de forma similar a la clave de R. Como T y P no se encuentran implicados por ningún otro atributo, forman parte de toda clave candidata. El cierre transitivo de G, cuya cardinalidad sigue siendo mayor, es $(G)^+_{DEP}$ = {G, A, CP, D}, por lo que se repite la situación del cálculo en R, y por lo tanto sus claves candidatas son las mismas:

- Claves candidatas: {T, P, G, AL} y {T, P, A, AL}
- Atributos principales:{T, P, G, A, AL}
- Atributos no principales: {CP, D}

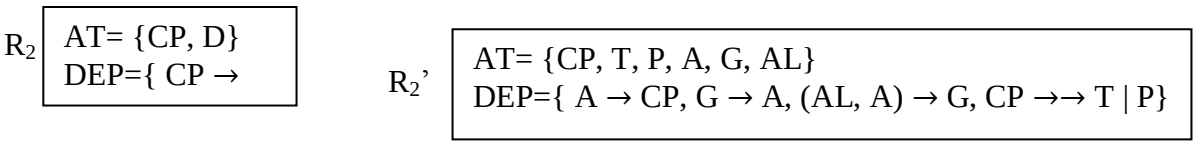
El cálculo de la forma normal en la que se encuentra la relación se detalla a continuación:

- 1FN: Sí, al tratarse de una relación.
- 2FN: No, ya que aún existen atributos no principales que dependen parcialmente de la calve. Por ejemplo en la dependencia A → CP.

De esta forma el primer nivel del árbol de descomposición se representa enseguida:



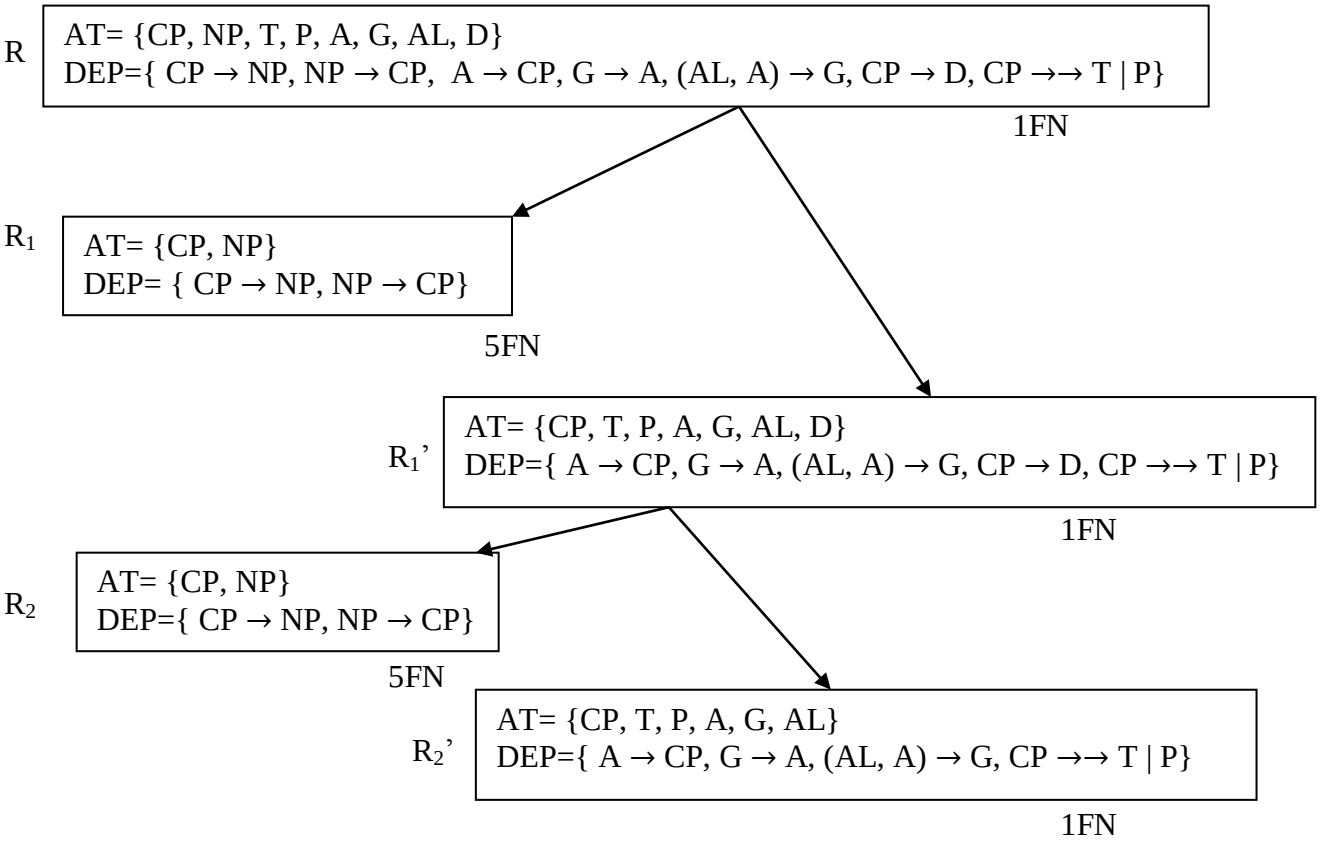
Paso 3: descomposición de R_1'
Se elige otro grupo de dependencias funcionales, en R_1' no se realizan más grupos de dependencias, ya que no existen dos dependencias con el mismo implicante, por lo que se elige, entonces, otra dependencia funcional cualquiera, por ejemplo CP → D, ya que D no participa en ninguna otra dependencia de R_1' se descompone entonces en dos relaciones nuevas R_2 y R_2', con los atributos y dependencias que aparecen a continuación:



La clave de R_2 es CP, ya que determina a D, además la relación se encuentra en 5FN, ya que su atributo no principal (D) se encuentra determinado por toda la clave, su único determinante (CP) es clave y no existe nada más que dependencias funcionales.
La clave de R_2', al igual que ocurría con R_1', está formada por T y P. Además $(AL,G)^+$={G, A, CP, AL} y $(AL,A)^+$={G, A, CP, AL}, luego las claves son las mismas que para R y R_1'.

- Clave(R_2') = Clave(R_1') = Clave(R) = { T, P, AL, G} y {T, P, AL, A}
- R_2', al igual que R_1' también se encuentran en 1FN, ya que CP depende funcionalmente de A, que no es clave candidata, aunque forma parte de una de ellas.

El árbol de descomposición hallado hasta el momento se muestra enseguida:



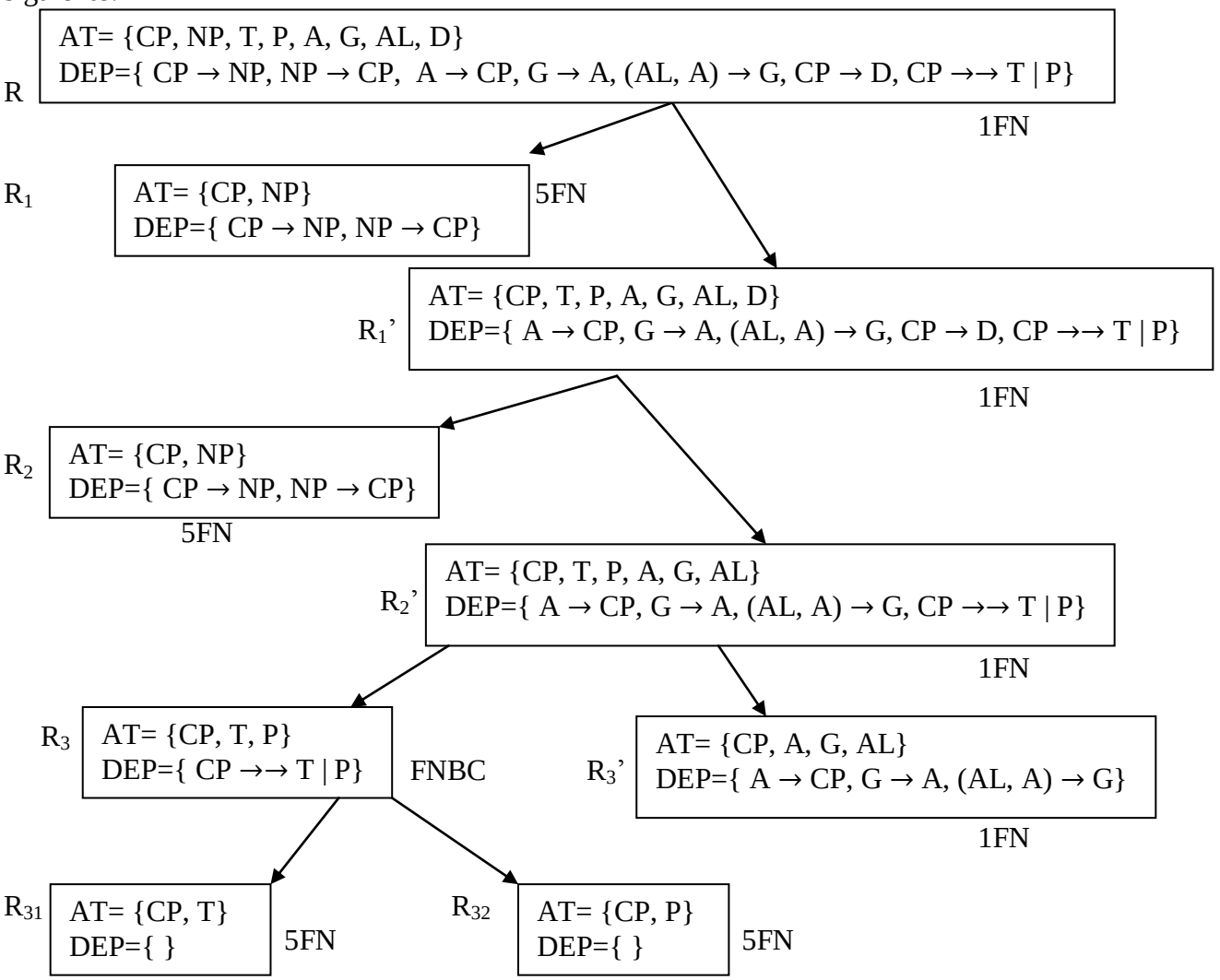
Paso 4: Descomposición de R_2':
A continuación se realiza el tratamiento de las dependencias multivaluadas, ya que si se eligiera cualquiera de las otras dependencias, donde el implicado se encuentra involucrado en otras, se podría perder alguna.
En principio, se crea una relación con la dependencia multivaluada jerárquica:

- R_3 ({CP, T, P]), { CP →→ T | P}).

Donde la clave está compuesta por todos los atributos de la relación. Pero esa relación no se encuentra en 4FN, ya no se encuentra determinada por toda la clave de la relación ({CP, T, P)]. Se ha demostrado que toda dependencia multivaluada jerárquica se puede descomponer en nuevas relaciones sin pérdida de información. La descomposición que se realiza en este caso es la siguiente:

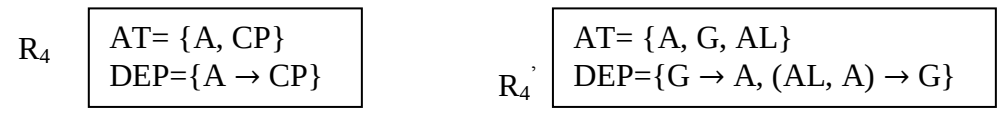
- R_{31}({CP, T}, {})
- R_{32}({CP, P},{})

Esta vez las relaciones R_{31} y R_{32} sí se encuentran en 4FN (todos los atributos forman parte de la clave) y también en 5FN, ya que no existen dependencias en combinación.
Por otro lado, se crea la relación temporal R_3' para poder continuar con la descomposición. El árbol de descomposición parcial hallada hasta el momento es el siguiente:



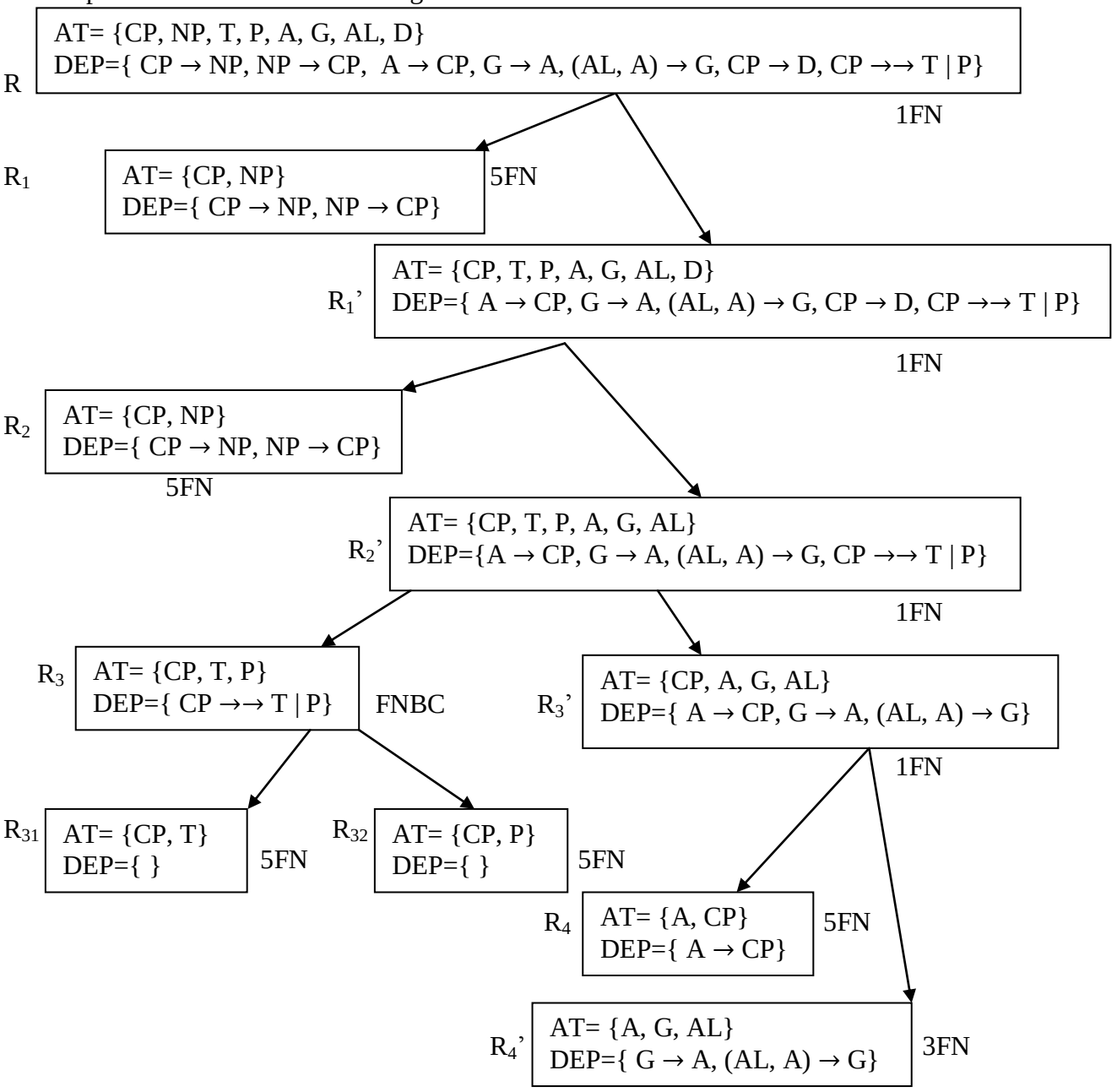
Paso 5: Descomposición de R_2'.

De nuevo se continúa descomponiendo la relación temporal, en este caso se vuelve a la descomposición de las dependencias funcionales, ya que ya se han resuelto todas las multivaluadas que existen. En estos instantes se puede elegir la dependencia A → CP, ya que CP no participa en ninguna otra dependencia en R_3.
Las relaciones que se crean son las que se muestran a continuación:



La relación R_4 se encuentra en 5FN, y su clave es el atributo A, mientras que la relación R_4' tiene como claves {AL, G} y {AL, A}, que se encuentran en 3FN, pero no en FNBC, ya que no todo determinante es clave candidata (en la dependencia G → A).

Paso 6. Descomposición de R_4'.
Si se interesa descomponer R_4' en dos relaciones se puede perder dependencias funcionales, ya que, si se elige la primera dependencia (G → A), el implicado, A participa en la segunda dependencia, y en el caso de que se elija la segunda, G participa en la primera, luego, la relación no se puede descomponer sin pérdida de dependencias.
Se supone (aunque no se detalla en el enunciado) que es requisito indispensable el mantenimiento de todas las dependencias de la relación. Por lo tanto, el árbol de descomposición final se muestra enseguida:



7.8 EJERCICIOS.

1. Se desea diseñar una base de datos para una empresa de ventas que tiene representantes (R) en las distintas áreas (A) donde vende sus productos (P). Se supone que:
   - Los representantes tienen un código (CR) y un conjunto de atributos (AR). Análogamente las áreas tienen CA y AA y los productos CP y AP.
   - En cada A hay varios R y cada R trabaja en varias A.
   - En todas las A se venden todos los P.
   - Un R puede vender varios P y cada P se vende por varios R.
   - Nunca dos R venden el mismo P en la misma A.
   - Todo R vende el mismo conjunto de P en cada A donde trabaja.

   Se pide:
   1) Las dependencias que se deducen de cada uno de los supuestos del enunciado. Si de algún supuesto se deduce la no existencia de alguna dependencia indíquelo también.
   2) Un conjunto mínimo de dependencias válido (recubrimiento minimal o irredundante).
   3) Estructura relacional que considere más adecuada, analizando el nivel de normalización de cada una de las relaciones.
2. Un departamento universitario desea diseñar una base de datos para la gestión de los cursos que imparte durante un cuatrimestre. En la base de datos quiere almacenar los profesores (P), los estudiantes (E), la nota (N) con la que se califica a un alumno en cada asignatura (AS), así como los días de la semana /hora (H) en la que se imparte una asignatura y el aula (AU) (se supone que ni el día/hora ni el aula en los que se imparte una asignatura varían de una semana a otra). Se desea almacenar también el teléfono (TL) y el despacho (D) de cada profesor (se supone que no existen teléfonos compartidos por dos profesores y que en cada despacho sólo hay un profesor y un teléfono). Se sabe asimismo que un profesor imparte clases a varios grupos (G) y en todos ellos utilizan los mismos textos (T).
   Además de los anteriores se dan los siguientes supuestos:
   a) En un momento dado tanto un estudiante como un profesor sólo pueden estar en un aula.
   b) En un momento dado en un aula sólo se puede impartir una asignatura.
   c) En cada despacho hay un solo teléfono.
   d) Un estudiante no puede asistir a las clases de dos asignaturas en una misma hora.
   e) Todas las asignaturas están divididas en los mismos grupos, utilizándose en todos los grupos de la misma asignatura los mismos textos.

   Se pide:
   1. Determinar las dependencias que existen entre atributos.
   2. Recubrimiento minimal, determinando si existe alguna dependencia redundante o algún atributo ajeno.
   3. Forma normal en la que se encuentra la relación.
   4. Esquema relacional normalizado con las observaciones que estime pertinentes, sin tener en cuenta las dependencias multivaluadas o en combinación.

3. En un centro de experimentación farmacéutica sobre animales se desea comprobar la eficiencia de determinados medicamentos (M) así como de los tratamientos (T) que se aplican a animales enfermos. En su fase inicial el centro decide imponer normas muy restrictivas en su funcionamiento. Se desea diseñar una base de datos aplicando la teoría de la normalización, para lo que nos dan las siguientes especificaciones:
    a) Los animales se identifican indistintamente, según las aplicaciones, por un código alfabético © o por un número de orden (N).
    b) Todos los veterinarios que trabajan en el centro tienen apellidos (A) distintos y, además, se les identifica también por su DNI (D).
    c) Un tratamiento sólo se le puede aplicar a un animal (se comprueba mediante el número de animal), y nunca un tratamiento lo puede aplicar dos veterinarios con distintos DNI.
    d) Un veterinario (sus apellidos) no esta nunca asociado a más de un medicamento por cada animal.
    e) Un veterinario sólo puede aplicar un tratamiento por cada animal.
    f) En un tratamiento, a cada animal sólo se le puede aplicar un medicamento.

    Se pide:
    I. Determinar las dependencias funcionales.
    II. Hallar un recubrimiento minimal válido.
    III. Determinar la forma normal en la que se encuentra la relación, explicando el proceso que le ha permitido llegar a la correspondiente conclusión.
    IV. Aplicar el proceso de análisis (árbol de descomposición) para obtener un esquema relacional FNBC realizando las observaciones que considere pertinentes.
4. Una empresa de productos de automóviles desea crear una base de datos relacional aplicando la teoría de la normalización. Las especificaciones que tiene son las siguientes:

    a) Un suministrador se identifica con su código (Cs) y su nombre (N).
    b) Toda pieza tiene un número de pieza (P) y un nombre (Np) únicos.
    c) Para cada modelo de vehículo (M), un suministrador suministra una determinada cantidad © de cada pieza.
    d) Cada pieza de un suministrador está situada en una única dirección (D) de una determinada ciudad (Ci).
    e) A una dirección nunca le corresponde más de un distrito postal (Dp) en cada ciudad.
    f) Una pieza de un suministrador que se suministra para un modelo no puede situarse en dos distritos postales distintos.
    g) Cada distrito postal está asignado a una única ciudad, aunque una misma ciudad le puede corresponder varios distritos postales
    h) El precio (Pr) de la pieza es independiente del suministrador que las provee.
    i) Interesa almacenar el número de serie (Ns) de cada vehículo (que es único) con el color (Co) del vehículo. A efectos de producción, los vehículos se identifican por un número consecutivo (Un) dentro del modelo.

Se pide:

1) Determinar las dependencias funcionales.
2) Hallar un recubrimiento minimal valido.
3) Determinar la forma normal en la que se encuentra la relación, explicando el proceso que ha permitido llegar a la correspondiente conclusión.

Bibliografía

Codd, E. F. (Junio, 1970). A Relational Model of Data for Large Shared Data Banks. *Comunications of ACM* , 377-387.

Date, C. J. (1993). *introducción a los Sistemas de Bases de Datos, Volumen 1.* México: Addison Wesley Longman.

de Miguel Castaño, A., Piattini Velthuis, M., & Marcos Martinez, E. (2000). *Diseño de Bases de Datos Relacionales.* México: Alfaomega Ra-Ma.

Elmasri, R., & Navate, S. (2008). *Fundamentos de Sistemas de Bases de Datos* (Septima ed.). (M. M. ROmo, Ed.) Madrid, España: Addison Wesley, Pearson.

Groff, J., & Weinberg, P. (1992). *Aplique SQL.* España: Mc Graw Hill.

Luis, J. A. (2013). *Big Data, Análisis de grandes volúmenes de datos en organizaciones.* México: Alfaomega.

Pagares Martìnez, G., & Santos Peñas, M. (2006). *inteligencia Artificial e Ingeniería del Conocimiento.* México: Alfaomega Ra-Ma.

Real Académia Española. (s.f.). Recuperado el 24 de 07 de 2014, de http://lema.rea.es/drae

Reinosa, E. J., Maldonado, C. A., Nuñoz, R., Damiano, L. E., & Abrustsky, M. A. (2012). *Bases de Datos.* México: Alfaomega.