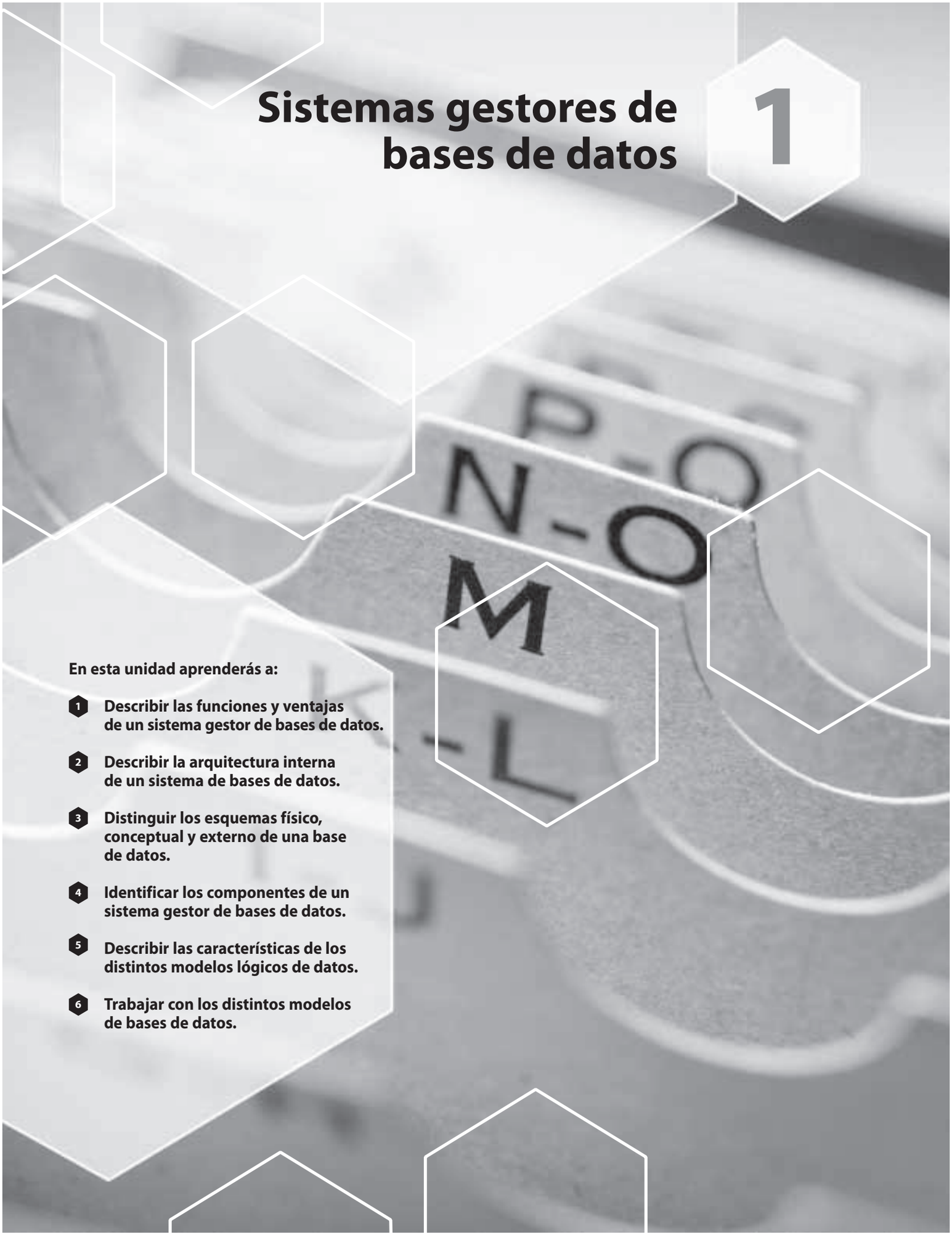
Sistemas gestores de bases de datos

1



En esta unidad aprenderás a:

1. **Describir las funciones y ventajas de un sistema gestor de bases de datos.**
2. **Describir la arquitectura interna de un sistema de bases de datos.**
3. **Distinguir los esquemas físico, conceptual y externo de una base de datos.**
4. **Identificar los componentes de un sistema gestor de bases de datos.**
5. **Describir las características de los distintos modelos lógicos de datos.**
6. **Trabajar con los distintos modelos de bases de datos.**

1.1 Introducción

Definimos un **Sistema Gestor de Bases de Datos** o **SGBD**, también llamado DBMS (*Data Base Management System*) como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina **Base de Datos** o **BD**, (*DB Data Base*).

Antes de aparecer los SGBD (década de los setenta), la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo. Éstos consistían en un conjunto de programas que definían y trabajaban sus propios datos. Los datos se almacenan en archivos y los programas manejan esos archivos para obtener la información. Si la estructura de los datos de los archivos cambia, todos los programas que los manejan se deben modificar; por ejemplo, un programa trabaja con un archivo de datos de alumnos, con una estructura o registro ya definido; si se incorporan elementos o campos a la estructura del archivo, los programas que utilizan ese archivo se tienen que modificar para tratar esos nuevos elementos. En estos sistemas de gestión de archivos, la definición de los datos se encuentra codificada dentro de los programas de aplicación en lugar de almacenarse de forma independiente, y además el control del acceso y la manipulación de los datos viene impuesto por los programas de aplicación.

Esto supone un gran inconveniente a la hora de tratar grandes volúmenes de información. Surge así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan. Se trata de estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan.

Inconvenientes de un sistema de gestión de archivos:

- **Redundancia e inconsistencia de los datos,** se produce porque los archivos son creados por distintos programas y van cambiando a lo largo del tiempo, es decir, pueden tener distintos formatos y los datos pueden estar duplicados en varios sitios. Por ejemplo, el teléfono de un alumno puede aparecer en más de un archivo. La redundancia aumenta los costes de almacenamiento y acceso, y trae consigo la inconsistencia de los datos: las copias de los mismos datos no coinciden por aparecer en varios archivos.

- **Dependencia de los datos física-lógica,** o lo que es lo mismo, la estructura física de los datos (definición de archivos y registros) se encuentra codificada en los programas de aplicación. Cualquier cambio en esa estructura implica al programador identificar, modificar y probar todos los programas que manipulan esos archivos.

- **Dificultad para tener acceso a los datos,** proliferación de programas, es decir, cada vez que se necesite una consulta que no fue prevista en el inicio implica la necesidad de codificar el programa de aplicación necesario. Lo que se trata de probar es que los entornos convencionales de procesamiento de archivos no permiten recuperar los datos necesarios de una forma conveniente y eficiente.

- **Separación y aislamiento de los datos,** es decir, al estar repartidos en varios archivos, y tener diferentes formatos, es difícil escribir nuevos programas que aseguren la manipulación de los datos correctos. Antes se deberían sincronizar todos los archivos para que los datos coincidiesen.

- **Dificultad para el acceso concurrente,** pues en un sistema de gestión de archivos es complicado que los usuarios actualicen los datos simultáneamente. Las actualizaciones concurrentes pueden dar por resultado datos inconsistentes, ya que se puede acceder a los datos por medio de diversos programas de aplicación.

- **Dependencia de la estructura del archivo con el lenguaje de programación,** pues la estructura se define dentro de los programas. Esto implica que los formatos de los archivos sean incompatibles. La incompatibilidad entre archivos generados por distintos lenguajes hace que los datos sean difíciles de procesar.

- **Problemas en la seguridad de los datos.** Resulta difícil implantar restricciones de seguridad pues las aplicaciones se van añadiendo al sistema según se van necesitando.

- **Problemas de integridad de datos,** es decir, los valores almacenados en los archivos deben cumplir con restricciones de consistencia. Por ejemplo, no se puede insertar una nota de un alumno en una asignatura si previamente esa asignatura no está creada. Otro ejemplo, las unidades en almacén de un producto determinado no deben ser inferiores a una cantidad. Esto implica añadir gran número de líneas de código en los programas. El problema se complica cuando existen restricciones que implican varios datos en distintos archivos.

Todos estos inconvenientes hacen posible el fomento y desarrollo de SGBD. El objetivo primordial de un gestor es proporcionar eficiencia y seguridad a la hora de extraer o almacenar información en las BD. Los sistemas gestores de BBDD están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información.

Una BD es un gran almacén de datos que se define una sola vez; los datos pueden ser accedidos de forma simultánea por varios usuarios; están relacionados y existe un número mínimo de duplicidad; además en las BBDD se almacenarán las descripciones de esos datos, lo que se llama *metadatos* en el diccionario de datos, que se verá más adelante.

El SGBD es una aplicación que permite a los usuarios definir, crear y mantener la BD y proporciona un acceso controlado a la misma. Debe prestar los siguientes servicios:

- **Creación y definición de la BD:** especificación de la estructura, el tipo de los datos, las restricciones y relaciones entre ellos mediante lenguajes de definición de datos. Toda esta información se almacena en el diccionario de datos, el SGBD proporcionará mecanismos para la gestión del diccionario de datos.

- **Manipulación de los datos** realizando consultas, inserciones y actualizaciones de los mismos utilizando lenguajes de manipulación de datos.

- **Acceso controlado a los datos de la BD** mediante mecanismos de seguridad de acceso a los usuarios.

- **Mantener la integridad y consistencia** de los datos utilizando mecanismos para evitar que los datos sean perjudicados por cambios no autorizados.
- **Acceso compartido a la BD,** controlando la interacción entre usuarios concurrentes.
- **Mecanismos de respaldo y recuperación** para restablecer la información en caso de fallos en el sistema.

1.2 Arquitectura de los sistemas de bases de datos

En 1975, el comité ANSI-SPARC (*American National Standard Institute - Standards Planning and Requirements Committee*) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la BD física. En esta arquitectura el esquema de una BD se define en tres niveles de abstracción distintos:

- **Nivel interno o físico:** el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la BD mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera.
- **Nivel externo o de visión:** es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la BD que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios.
- **Nivel conceptual:** describe la estructura de toda la BD para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento. Representa la información contenida en la BD. En la Figura 1.1 se representan los niveles de abstracción de la arquitectura ANSI.

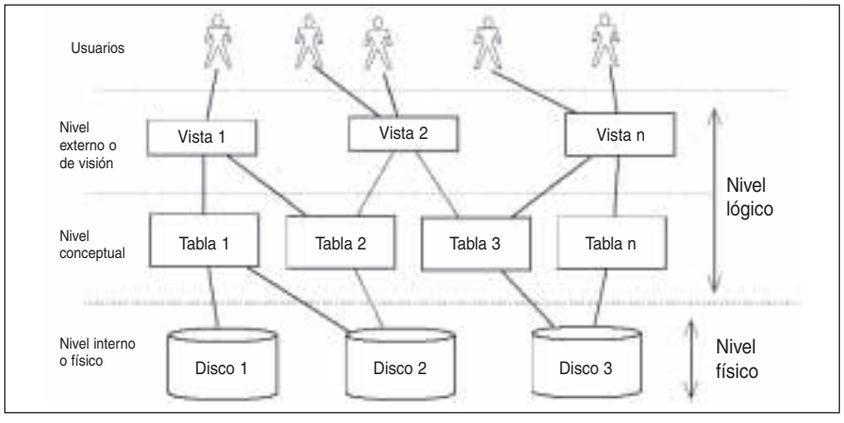


Figura 1.1. *Niveles de abstracción de la arquitectura ANSI.*

Esta arquitectura describe los datos a tres niveles de abstracción. En realidad los únicos datos que existen están a nivel físico almacenados en discos u otros dispositivos. Los SGBD basados en esta arquitectura permiten que cada grupo de usuarios haga referencia a su propio esquema externo. El SGBD debe de transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos de esquema conceptual, para finalmente ser una petición expresada en el esquema interno que se procesará sobre la BD almacenada. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación, el SGBD es capaz de interpretar una solicitud de datos y realiza los siguientes pasos:

- El usuario solicita unos datos y crea una consulta.
- El SGBD verifica y acepta el esquema externo para ese usuario.
- Transforma la solicitud al esquema conceptual.
- Verifica y acepta el esquema conceptual.
- Transforma la solicitud al esquema físico o interno.
- Selecciona la o las tablas implicadas en la consulta y ejecuta la consulta.
- Transforma del esquema interno al conceptual, y del conceptual al externo.
- Finalmente, el usuario ve los datos solicitados.

Para una BD específica sólo hay un esquema interno y uno conceptual, pero puede haber varios esquemas externos definidos para uno o para varios usuarios.

Con la arquitectura a tres niveles se introduce el concepto de independencia de datos, se definen dos tipos de independencia:

- **Independencia lógica:** la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se podrá modificar el esquema conceptual para ampliar la BD o para reducirla, por ejemplo, si se elimina una entidad, los esquemas externos que no se refieran a ella no se verán afectados.

- **Independencia física:** la capacidad de modificar el esquema interno sin tener que alterar ni el esquema conceptual, ni los externos. Por ejemplo, se pueden reorganizar los archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización, o se pueden añadir nuevos archivos de datos porque los que había se han llenado. La independencia física es más fácil de conseguir que la lógica, pues se refiere a la separación entre las aplicaciones y las estructuras físicas de almacenamiento.

En los SGBD basados en arquitecturas de varios niveles se hace necesario ampliar el catálogo o el diccionario de datos para incluir la información sobre cómo establecer las correspondencias entre las peticiones de los usuarios y los datos, entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra

en el diccionario. La independencia de los datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios. Sólo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

Sin embargo, los dos niveles de correspondencia implican un gasto de recursos durante la ejecución de una consulta o de un programa, lo que reduce la eficiencia del SGBD. Por esta razón pocos SGBD han implementado la arquitectura completa.

1.3 Componentes de los SGBD

Los SGBD son paquetes de software muy complejos que deben proporcionar una serie de servicios que van a permitir almacenar y explotar los datos de forma eficiente. Los componentes principales son los siguientes:

A. Lenguajes de los SGBD

Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores de aplicaciones y usuarios finales.
Los lenguajes van a permitir al administrador de la BD especificar los datos que componen la BD, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios. Los lenguajes del SGBD se clasifican en:

- **Lenguaje de definición de datos (LDD o DDL):** se utiliza para especificar el esquema de la BD, las vistas de los usuarios y las estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la BD.

- **Lenguaje de manipulación de datos (LMD o DML):** se utilizan para leer y actualizar los datos de la BD. Es el utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones. Los hay *procedurales*, en los que el usuario será normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan. Las sentencias de un LMD procedural están embebidas en un lenguaje de alto nivel llamado *anfitrión*. Las BD jerárquicas y en red utilizan estos LMD procedurales.

  No procedurales son los lenguajes declarativos. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde un terminal, también pueden ir embebidas en un lenguaje de programación de alto nivel. Estos lenguajes permiten especificar los datos a obtener en una consulta, o los datos a modificar, mediante sentencias sencillas. Las BD relacionales utilizan lenguajes no procedurales como SQL (*Structured Quero Language*) o QBE (*Query By Example*).

- La mayoría de los SGBD comerciales incluyen **lenguajes de cuarta generación (4GL)** que permiten al usuario desarrollar aplicaciones de forma fácil y rápida, también se les llama *herramientas de desarrollo*. Ejemplos de esto son las herramientas del SGBD

ORACLE: SQL Forms para la generación de formularios de pantalla y para interactuar con los datos; SQL Reports para generar informes de los datos contenidos en la BD; PL/SQL lenguaje para crear procedimientos que interractuen con los datos de la BD.

B. El diccionario de datos

El **diccionario de datos** es el lugar donde se deposita información acerca de todos los datos que forman la BD. Es una guía en la que se describe la BD y los objetos que la forman.

El diccionario contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información.

En una BD relacional, el diccionario de datos proporciona información acerca de:

- La estructura lógica y física de la BD.
- Las definiciones de todos los objetos de la BD: tablas, vistas, índices, disparadores, procedimientos, funciones, etcétera.
- El espacio asignado y utilizado por los objetos.
- Los valores por defecto de las columnas de las tablas.
- Información acerca de las restricciones de integridad.
- Los privilegios y roles otorgados a los usuarios.
- Auditoría de información, como los accesos a los objetos.

Un diccionario de datos debe cumplir las siguientes características:

- Debe soportar las descripciones de los modelos conceptual, lógico, interno y externo de la BD.
- Debe estar integrado dentro del SGBD.
- Debe apoyar la transferencia eficiente de información al SGDB. La conexión entre los modelos interno y externo debe ser realizada en tiempo de ejecución.
- Debe comenzar con la reorganización de versiones de producción de la BD. Además debe reflejar los cambios en la descripción de la BD. Cualquier cambio a la descripción de programas ha de ser reflejado automáticamente en la librería de descripción de programas con la ayuda del diccionario de datos.
- Debe estar almacenado en un medio de almacenamiento con acceso directo para la fácil recuperación de información.

C. Seguridad e integridad de datos

Un SGBD proporciona los siguientes mecanismos para garantizar la seguridad e integridad de los datos:

- Debe garantizar la protección de los datos contra accesos no autorizados, tanto intencionados como accidentales. Debe controlar que sólo los usuarios autorizados accedan a la BD.

- Los SGBD ofrecen mecanismos para implantar restricciones de integridad en la BD. Estas restricciones van a proteger la BD contra daños accidentales. Los valores de los datos que se almacenan deben satisfacer ciertos tipos de restricciones de consistencia y reglas de integridad, que especificará el administrador de la BD. El SGBD puede determinar si se produce una violación de la restricción.

- Proporciona herramientas y mecanismos para la planificación y realización de copias de seguridad y restauración.

- Debe ser capaz de recuperar la BD llevándola a un estado consistente en caso de ocurrir algún suceso que la dañe.

- Debe asegurar el acceso concurrente y ofrecer mecanismos para conservar la consistencia de los datos en el caso de que varios usuarios actualicen la BD de forma concurrente.

D. El administrador de la BD

En los sistemas de gestión de BBDD actuales existen diferentes categorías de usuarios. Estas categorías se caracterizan porque cada una de ellas tiene una serie de privilegios o permisos sobre los objetos que forman la BD.

En los sistemas Oracle las categorías más importantes son:

- **Los usuarios de la categoría DBA** (*Database Administrator*), cuya función es precisamente administrar la base y que tienen, el nivel más alto de privilegios.

- **Los usuarios de la categoría RESOURCE,** que pueden crear sus propios objetos y tienen acceso a los objetos para los que se les ha concedido permiso.

- **Los usuarios del tipo CONNECT,** que solamente pueden utilizar aquellos objetos para los que se les ha concedido permiso de acceso.

El DBA tiene una gran responsabilidad ya que posee el máximo nivel de privilegios. Será el encargado de crear los usuarios que se conectarán a la BD. En la administración de una BD siempre hay que procurar que haya el menor número de administradores, a ser posible una sola persona.

El objetivo principal de un DBA es garantizar que la BD cumple los fines previstos por la organización, lo que incluye una serie de tareas como:

- Instalar SGBD en el sistema informático.
- Crear las BBDD que se vayan a gestionar.
- Crear y mantener el esquema de la BD.
- Crear y mantener las cuentas de usuario de la BD.
- Arrancar y parar SGBD, y cargar las BBDD con las que se ha de trabajar.
- Colaborar con el administrador del S.O. en las tareas de ubicación, dimensionado y control de los archivos y espacios de disco ocupados por el SGBD.
- Colaborar en las tareas de formación de usuarios.
- Establecer estándares de uso, políticas de acceso y protocolos de trabajo diario para los usuarios de la BD.
- Suministrar la información necesaria sobre la BD a los equipos de análisis y programación de aplicaciones.
- Efectuar tareas de explotación como:
  - Vigilar el trabajo diario colaborando en la información y resolución de las dudas de los usuarios de la BD.
  - Controlar en tiempo real los accesos, tasas de uso, cargas en los servidores, anomalías, etcétera.
  - Llegado el caso, reorganizar la BD.
  - Efectuar las copias de seguridad periódicas de la BD.
  - Restaurar la BD después de un incidente material a partir de las copias de seguridad.
  - Estudiar las auditorías del sistema para detectar anomalías, intentos de violación de la seguridad, etcétera.
  - Ajustar y optimizar la BD mediante el ajuste de sus parámetros, y con ayuda de las herramientas de monitorización y de las estadísticas del sistema.

En su gestión diaria, el DBA suele utilizar una serie de herramientas de administración de la BD.

Con el paso del tiempo, estas herramientas han adquirido sofisticadas prestaciones y facilitan en gran medida la realización de trabajos que, hasta no hace demasiado, requerían de arduos esfuerzos por parte de los administradores.

1.4 Modelos de datos

Uno de los objetivos más importantes de un SGBD es proporcionar a los usuarios una visión abstracta de los datos, es decir, el usuario va a utilizar esos datos pero no tendrá idea de cómo están almacenados físicamente.

Los modelos de datos son el instrumento principal para ofrecer esa abstracción. Son utilizados para la representación y el tratamiento de los problemas. Forman el problema a tres niveles de abstracción, relacionados con la arquitectura ANSI-SPARC de tres niveles para los SGBD:

- **Nivel físico:** el nivel más bajo de abstracción; describe cómo se almacenan realmente los datos.

- **Nivel lógico o conceptual:** describe los datos que se almacenan en la BD y sus relaciones, es decir, los objetos del mundo real, sus atributos y sus propiedades, y las relaciones entre ellos.

- **Nivel externo o de vistas:** describe la parte de la BD a la que los usuarios pueden acceder.

Para hacernos una idea de los tres niveles de abstracción, nos imaginamos un archivo de artículos con el siguiente registro:

```
struct ARTICULOS
 {  int Cod;
    char Deno[15];
    int cant_almacen;
    int cant_minima ;
    int uni_vendidas;
    float PVP;
    char reponer;
    struct VENTAS Tventas[12];
 };
```

El **nivel físico** es el conjunto de bytes que se encuentran almacenados en el archivo en un dispositivo magnético, que puede ser un disco, una pista a un sector determinado.

El **nivel lógico** comprende la descripción y la relación con otros registros que se hace del registro dentro de un programa, en un lenguaje de programación.

El último nivel de abstracción, el **externo**, es la visión de estos datos que tiene un usuario cuando ejecuta aplicaciones que operan con ellos, el usuario no sabe el detalle de los datos, unas veces operará con unos y otras con otros, dependiendo de la aplicación.

Si trasladamos el ejemplo a una BD relacional específica habrá, como en el caso anterior, un único nivel interno y un único nivel lógico o conceptual, pero puede haber varios niveles externos, cada uno definido para uno o para varios usuarios. Podría ser el siguiente:

Curso	Nombre	Nombre de asignatura	Nota
1	Ana	Programación en lenguajes estructurados	6
1	Ana	Sistemas informáticos multiusuario y en red	8
2	Rosa	Desa. de aplic. en entornos de 4.ª Generación y H. Case	5
2	Juan	Desa. de aplic. en entornos de 4.ª Generación y H. Case	7
1	Alicia	Programación en lenguajes estructurados	5
1	Alicia	Sistemas informáticos multiusuario y en red	4

Tabla 1.1. *Vista de la BD para un usuario.*

- **Nivel externo:** Visión parcial de las tablas de la BD según el usuario. Por ejemplo, la vista que se muestra en la Tabla 1.1 obtiene el listado de notas de alumnos con los siguientes datos: Curso, Nombre, Nombre de asignatura y Nota.

- **Nivel lógico y conceptual:** Definición de todas las tablas, columnas, restricciones, claves y relaciones. En este ejemplo, disponemos de tres tablas que están relacionadas:

  – *Tabla ALUMNOS*. Columnas: NMatrícula, Nombre, Curso, Dirección, Población. Clave: NMatrícula. Además tiene una relación con NOTAS, pues un alumno puede tener notas en varias asignaturas.

  – *Tabla ASIGNATURAS*. Columnas: Código, Nombre de asignatura. Clave: Código. Está relacionada con NOTAS, pues para una asignatura hay varias notas, tantas como alumnos la cursen.

  – *Tabla NOTAS*. Columnas: NMatrícula, Código, Nota. Está relacionada con ALUMNOS y ASIGNATURAS, pues un alumno tiene notas en varias asignaturas, y de una asignatura existen varias notas, tantas como alumnos.

Podemos representar las relaciones de las tablas en el nivel lógico como se muestra en la Figura 1.2:

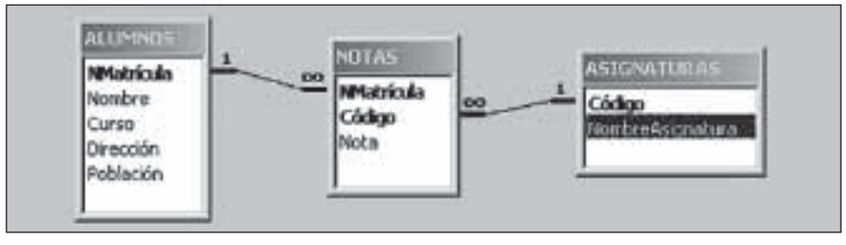


Figura 1.2. *Representación de las relaciones entre tablas en el nivel lógico.*

- **Nivel interno**: En una BD las tablas se almacenan en archivos de datos de la BD. Si hay claves, se crean índices para acceder a los datos, todo esto contenido en el disco duro, en una pista y en un sector, que sólo el SGBD conoce. Ante una petición, sabe a qué pista, a qué sector, a qué archivo de datos y a qué índices acceder.

Para la representación de estos niveles se utilizan los *modelos de datos*. Se definen como el conjunto de conceptos o herramientas conceptuales que sirven para describir la estructura de una BD: los datos, las relaciones y las restricciones que se deben cumplir sobre los datos. Se denomina **esquema de la BD** a la descripción de una BD mediante un modelo de datos. Este esquema se especifica durante el diseño de la misma.

Podemos dividir los modelos en tres grupos: *modelos lógicos basados en objetos*, *modelos lógicos basados en registros* y *modelos físicos de datos*. Cada SGBD soporta un modelo lógico.

Modelos lógicos basados en objetos

Los modelos lógicos basados en objetos se usan para describir datos en el nivel conceptual y el externo. Se caracterizan porque proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos. Los modelos más conocidos son el modelo *entidad-relación* y el *orientado a objetos*.

Actualmente, el más utilizado es el modelo entidad-relación, aunque el modelo orientado a objetos incluye muchos conceptos del anterior, y poco a poco está ganando mercado. La mayoría de las BBDD relacionales añaden extensiones para poder ser relacionales-orientadas a objetos.

Modelos lógicos basados en registros

Los modelos lógicos basados en registros se utilizan para describir los datos en los modelos conceptual y físico. A diferencia de los modelos lógicos basados en objetos, se usan para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de la implementación.

Los modelos basados en registros se llaman así porque la BD está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija. La estructura más rica de estas BBDD a menudo lleva a registros de longitud variable en el nivel físico.

Los modelos basados en registros no incluyen un mecanismo para la representación directa de código de la BD, en cambio, hay lenguajes separados que se asocian con el modelo para expresar consultas y actualizaciones. Los tres modelos de datos más aceptados son los modelos *relacional*, *de red* y *jerárquico*. El modelo relacional ha ganado aceptación por encima de los otros; representa los datos y las relaciones entre los datos mediante una colección de tablas, cuyas columnas tienen nombres únicos, las filas (*tuplas*) representan a los registros y las columnas representan las características (*atributos*) de cada registro. Este modelo se estudiará en la siguiente Unidad.

Modelos físicos de datos

Los modelos físicos de datos se usan para describir cómo se almacenan los datos en el ordenador: formato de registros, estructuras de los archivos, métodos de acceso, etcétera. Hay muy pocos modelos físicos de datos en uso, siendo los más conocidos el modelo *unificador* y *de memoria de elementos*.

1.5 El modelo entidad-interrelación

El modelo de datos entidad-interrelación (E-R), también llamado entidad-relación, fue propuesto por Peter Chen en 1976 para la representación conceptual de los problemas del mundo real. En 1988, el ANSI lo seleccionó como modelo estándar para los sistemas de diccionarios de recursos de información. Es un modelo muy extendido y potente para la representación de los datos. Se simboliza haciendo uso de grafos y de tablas. Propone el uso de tablas bidimensionales para la representación de los datos y sus relaciones.

Conceptos básicos

Entidad. Es un objeto del mundo real, que tiene interés para la empresa. Por ejemplo, los ALUMNOS de un centro escolar o los CLIENTES de un banco. Se representa utilizando rectángulos.

Conjunto de entidades. Es un grupo de entidades del mismo tipo, por ejemplo, el conjunto de entidades cliente. Los conjuntos de entidades no necesitan ser disjuntos, se puede definir los conjuntos de entidades de empleados y clientes de un banco, pudiendo existir una persona en ambas o ninguna de las dos cosas.

Entidad fuerte. Es aquella que no depende de otra entidad para su existencia. Por ejemplo, la entidad ALUMNO es fuerte pues no depende de otra para existir, en cambio, la entidad NOTAS es una entidad débil pues necesita a la entidad ALUMNO para existir. Las entidades débiles se relacionan con la entidad fuerte con una relación uno a varios. Se representan con un rectángulo con un borde doble.

Atributos o campos. Son las unidades de información que describen propiedades de las entidades. Por ejemplo, la entidad ALUMNO posee los atributos: número de matrícula, nombre, dirección, población y teléfono. Los atributos toman valores, por ejemplo, el atributo población puede ser ALCALÁ, GUADALAJARA, etcétera. Se representan mediante una elipse con el nombre en su interior.

Dominio. Es el conjunto de valores permitido para cada atributo. Por ejemplo el dominio del atributo nombre puede ser el conjunto de cadenas de texto de una longitud determinada.

Identificador o superclave. Es el conjunto de atributos que identifican de forma única a cada entidad. Por ejemplo, la entidad EMPLEADO, con los atributos Número de la Seguridad Social, DNI, Nombre, Dirección, Fecha nacimiento y Tlf, podrían ser identificadores o superclaves los conjuntos Nombre, Dirección, Fecha nacimiento y Tlf, o también DNI, Nombre y Dirección, o también Num Seg Social, Nombre, Dirección y Tlf, o solos el DNI y el Número de la Seguridad Social.

Clave candidata. Es cada una de las superclaves formadas por el mínimo número de campos posibles. En el ejemplo anterior, son el DNI y el Número de la Seguridad Social.

Clave primaria o principal (*primary key*)**:** Es la clave candidata seleccionada por el diseñador de la BD. Una clave candidata no puede contener valores nulos, ha de ser sencilla de crear y no ha de variar con el tiempo. El atributo o los atributos que forman esta clave se representan subrayados.

Clave ajena o foránea (*foreign key*)**:** Es el atributo o conjunto de atributos de una entidad que forman la clave primaria en otra entidad. Las claves ajenas van a representar las relaciones entre tablas. Por ejemplo, si tenemos por un lado, las entidades ARTÍCULOS, con los atributos código de artículo (clave primaria), denominación, stock. Y, por otro lado, VENTAS, con los atributos Código de venta (clave primaria), fecha de venta, código de artículo, unidades vendidas, el código de artículo es clave ajena pues está como clave primaria en la entidad ARTÍCULOS.

A. Relaciones y conjuntos de relaciones

Definimos una **relación** como la asociación entre diferentes entidades. Tienen nombre de verbo, que la identifica de las otras relaciones y se representa mediante un rombo. Normalmente las relaciones no tienen atributos. Cuando surge una relación con atributos significa que debajo hay una entidad que aún no se ha definido. A esa entidad se la llama *entidad asociada*. Esta entidad dará origen a una tabla que contendrá esos atributos. Esto se hace en el *modelo relacional* a la hora de representar los datos. Lo veremos más adelante.

Un **conjunto de relaciones** es un conjunto de relaciones del mismo tipo, por ejemplo entre ARTÍCULOS y VENTAS todas las asociaciones existentes entre los artículos y las ventas que tengan estos, forman un conjunto de relaciones.

La mayoría de los conjuntos de relaciones en un sistema de BD son binarias (dos entidades) aunque puede haber conjuntos de relaciones que implican más de dos conjuntos de entidades, por ejemplo, una relación como la relación entre cliente, cuenta y sucursal. Siempre es posible sustituir un conjunto de relaciones no binario por varios conjuntos de relaciones binarias distintos. Así, conceptualmente, podemos restringir el modelo E-R para incluir sólo conjuntos binarios de relaciones, aunque no siempre es posible.

La función que desempeña una entidad en una relación se llama *papel*, y normalmente es implícito y no se suele especificar. Sin embargo, son útiles cuando el significado de una relación necesita ser clarificado.

Una relación también puede tener atributos descriptivos, por ejemplo, la FECHA_OPERACIÓN en el conjunto de relaciones CLIENTE_CUENTA, que especifica la última fecha en la que el cliente tuvo acceso a su cuenta (ver Figura 1.3).

Figura 1.3. *Relación con atributos descriptivos.*

Diagramas de estructuras de datos en el modelo E-R

Los diagramas Entidad-Relación representan la estructura lógica de una BD de manera gráfica. Los símbolos utilizados son los siguientes:

– Rectángulos para representar a las entidades.

- Elipses para los atributos. El atributo que forma parte de la clave primaria va subrayado.
- Rombos para representar las relaciones.
- Las líneas, que unen atributos a entidades y a relaciones, y entidades a relaciones. Si la flecha tiene punta, en ese sentido está el uno, y si no la tiene, en ese sitio está el muchos. La orientación señala cardinalidad.
- Si la relación tiene atributos asociados, se le unen a la relación.
- Cada componente se etiqueta con el nombre de lo que representa.

En la Figura 1.4 se muestra un diagrama E-R correspondiente a PROVEEDORES-ARTÍCULOS. Un PROVEEDOR SUMINISTRA muchos ARTÍCULOS.

Figura 1.4. *Diagrama E-R, un proveedor suministra muchos artículos.*

Grado y cardinalidad de las relaciones

Se define **grado de una relación** como el número de conjuntos de entidades que participan en el conjunto de relaciones, o lo que es lo mismo, el número de entidades que participan en una relación. Las relaciones en las que participan dos entidades son *binarias* o *de grado dos*. Si participan tres serán *ternarias* o *de grado 3*. Los conjuntos de relaciones pueden tener cualquier grado, lo ideal es tener relaciones binarias.

Las relaciones en las que sólo participa una entidad se llaman *anillo* o *de grado uno*; relaciona una entidad consigo misma, se las llama *relaciones reflexivas*. Por ejemplo, la entidad EMPLEADO puede tener una relación JEFE DE consigo misma: un empleado es JEFE DE muchos empleados y, a la vez, el jefe es un empleado.

Otro ejemplo puede ser la relación DELEGADO DE los alumnos de un curso: el delegado es alumno también del curso. Ver Figura 1.5.

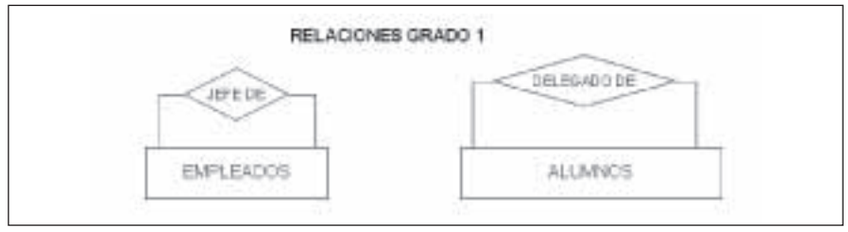


Figura 1.5. *Relaciones de grado 1.*

En la Figura 1.6 se muestra una relación de grado dos, que representa un proveedor que suministra artículos, y otra de grado tres, que representa un cliente de un banco que tiene varias cuentas, y cada una en una sucursal:

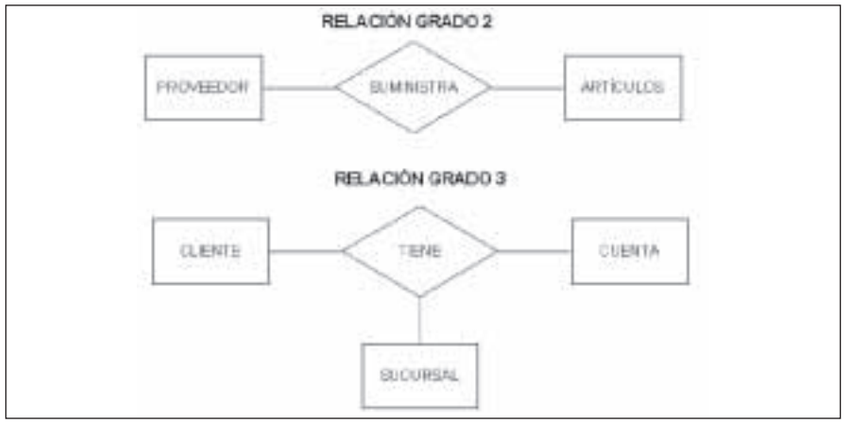


Figura 1.6. *Relaciones de grados 2 y 3.*

En el modelo E-R se representan ciertas restricciones a las que deben ajustarse los datos contenidos en una BD. Éstas son las restricciones de las cardinalidades de asignación, que expresan el número de entidades a las que puede asociarse otra entidad mediante un conjunto de relación.

Las cardinalidades de asignación se describen para conjuntos binarios de relaciones. Son las siguientes:

- **1:1, uno a uno.** A cada elemento de la primera entidad le corresponde sólo uno de la segunda entidad, y a la inversa. Por ejemplo, un cliente de un hotel ocupa una habitación, o un curso de alumnos pertenece a un aula, y a esa aula sólo asiste ese grupo de alumnos. Ver Figura 1.7:

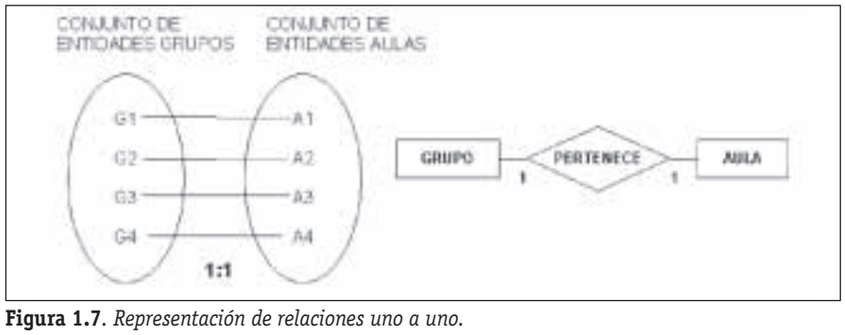


Figura 1.7. *Representación de relaciones uno a uno.*

- **1:N, uno a muchos.** A cada elemento de la primera entidad le corresponde uno o más elementos de la segunda entidad, y a cada elemento de la segunda entidad le corresponde uno sólo de la primera entidad. Por ejemplo, un proveedor suministra muchos artículos (ver Figura 1.8).

Figura 1.8. *Representación de relaciones uno a muchos.*

- **N:1, muchos a uno.** Es el mismo caso que el anterior pero al revés; a cada elemento de la primera entidad le corresponde un elemento de la segunda, y a cada elemento de la segunda entidad, le corresponden varios de la primera.

- **M:N, muchos a muchos.** A cada elemento de la primera entidad le corresponde uno o más elementos de la segunda entidad, y a cada elemento de la segunda entidad le corresponde uno o más elementos de la primera entidad. Por ejemplo, un vendedor vende muchos artículos, y un artículo es vendido por muchos vendedores (ver Figura 1.9).

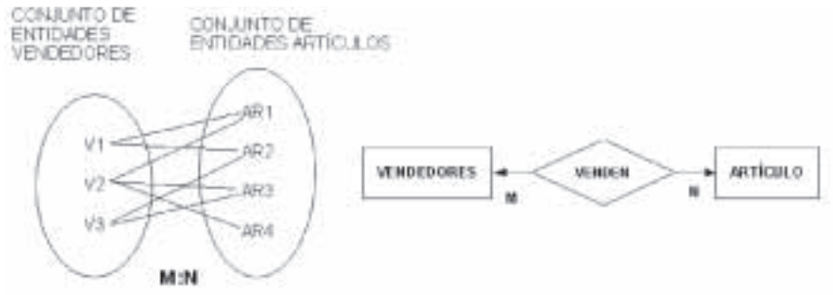


Figura 1.9. *Representación de relaciones muchos a muchos.*

La cardinalidad de una entidad sirve para conocer su grado de participación en la relación, es decir, el número de correspondencias en las que cada elemento de la entidad interviene. Mide la obligatoriedad de correspondencia entre dos entidades.

La representamos entre paréntesis indicando los valores máximo y mínimo: (máximo, mínimo). Los valores para la cardinalidad son: (0,1), (1,1), (0,N), (1,N) y (M,N). El valor 0 se pone cuando la participación de la entidad es opcional.

En la Figura 1.10, que se muestra a continuación, se representa el diagrama E-R en el que contamos con las siguientes entidades:

- EMPLEADO está formada por los atributos Nº. Emple, Apellido, Salario y Comisión, siendo el atributo Nº. Emple la clave principal (representado por el subrayado).

- DEPARTAMENTO está formada por los atributos Nº. Depart, Nombre y Localidad, siendo el atributo Nº. Depart la clave principal.

- Se han definido dos relaciones:

  – La relación «PERTENECE» entre las entidades EMPLEADOS y DEPARTAMENTO, cuyo tipo de correspondencia es 1:N, es decir, a un departamento le pertenecen cero o más empleados (0,N). Un empleado pertenece a un departamento y sólo a uno (1,1).

  – La relación «JEFE», que asocia la entidad EMPLEADO consigo misma. Su tipo de correspondencia es 1:N, es decir, un empleado es jefe de cero o más empleados (0,N). Un empleado tiene un jefe y sólo uno (1,1). Ver Figura 1.10:

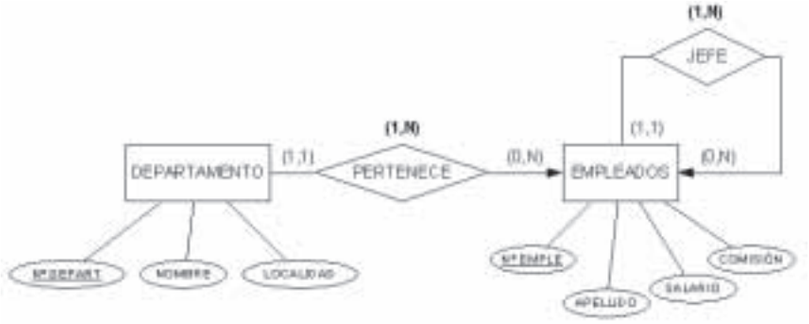


Figura 1.10. *Diagrama E-R de las relaciones entre departamentos y empleados.*

Caso práctico

1 **Vamos a realizar el diagrama de estructuras de datos en el modelo E-R. Supongamos que en un centro escolar se imparten muchos cursos. Cada curso está formado por un grupo de alumnos, de los cuales uno de ellos es el delegado del grupo. Los alumnos cursan asignaturas, y una asignatura puede o no ser cursada por los alumnos.**

Para su resolución, primero identificaremos las entidades, luego las relaciones y las cardinalidades y, por último, los atributos de las entidades y de las interrelaciones, si las hubiera.

1. Identificación de entidades: una entidad es un objeto del mundo real, algo que tiene interés para la empresa. Se hace un análisis del enunciado, de donde sacaremos los candidatos a entidades: CENTROS, CURSOS, ALUMNOS, ASIGNATURAS, DELEGADOS. Si analizamos esta última veremos que los delegados son alumnos, por lo tanto, los tenemos recogidos en ALUMNOS. Esta posible entidad la eliminaremos. También eliminaremos la posible entidad CENTROS pues se trata de un único centro, si se tratara de una gestión de centros tendría más sentido incluirla.

2. Identificar las relaciones: construimos una matriz de entidades en la que las filas y las columnas son los nombres de entidades y cada celda puede contener o no la relación, las relaciones aparecen explícitamente en el enunciado. En este ejemplo, las relaciones no tienen atributos. Del enunciado sacamos lo siguiente:

   – Un curso está formado por muchos alumnos. La relación entre estas dos entidades la llamamos PERTENECE, pues a un curso pertenecen muchos alumnos, relación 1:M. Consideramos que es obligatorio que existan alumnos en un curso. Para calcular los máximos y mínimos hacemos la pregunta: a un CURSO, ¿cuántos ALUMNOS pertenecen, como mínimo y como máximo? Y se ponen los valores en la entidad ALUMNOS, en este caso (1,M). Para el sentido contrario, hacemos lo mismo: un ALUMNO, ¿a cuántos CURSOS va a pertenecer? Como mínimo a 1, y como máximo a 1, en este caso pondremos (1,1) en la entidad CURSOS.

(Continúa)

(Continuación)

– De los alumnos que pertenecen a un grupo, uno de ellos es DELEGADO. Hay una relación de grado 1 entre la entidad ALUMNO que la podemos llamar ES DELEGADO. La relación es 1:M, un alumno es delegado de muchos alumnos. Para calcular los valores máximos y mínimos preguntamos: ¿un ALUMNO de cuántos alumnos ES DELEGADO? Como mínimo es 0, pues puede que no sea delegado, y como máximo es M, pues si es delegado lo será de muchos; pondremos en el extremo (0,M). Y en el otro extremo pondremos (1,1), pues obligatoriamente el delegado es un alumno.

– Entre ALUMNOS y ASIGNATURAS surge una relación N:M, pues un alumno cursa muchas asignaturas y una asignatura es cursada por muchos alumnos. La relación se llamará CURSA. Consideramos que puede haber asignaturas sin alumnos. Las cardinalidades serán (1:M) entre ALUMNO-ASIGNATURA, pues un alumno, como mínimo, cursa una asignatura, y, como máximo, muchas. La cardinalidad entre ASIGNATURA-ALUMNO será (0,N), pues una ASIGNATURA puede ser cursada por 0 alumnos o por muchos.

En la Tabla 1.2 se muestra la matriz de entidades y relaciones entre ellas:

	CURSOS	**ALUMNOS**	**ASIGNATURAS**
CURSOS	----------	PERTENECE (1:M)	----------
ALUMNOS	x	ES DELEGADO(1:M)	CURSA(N:M)
ASIGNATURAS	----------	x	----------

Tabla 1.2. *Matriz de entidades y relaciones entre ellas.*

Las celdas que aparecen con una x indican que las relaciones están ya identificadas. Las que aparecen con guiones indican que no existe relación. En la siguiente figura se muestra el diagrama de las relaciones y las cardinalidades.

3. Identificar los atributos, como el enunciado no explicita ningún tipo de característica de las entidades nos imaginamos los atributos, que pueden ser los siguientes:

CURSOS - COD_CURSO (clave primaria), DESCRIPCIÓN, NIVEL, TURNO y ETAPA

ALUMNOS - NUM-MATRÍCULA (clave primaria), NOMBRE, DIRECCIÓN, POBLACIÓN, TLF y NUM_HERMANOS

ASIGNATURAS - COD-ASIGNATURA (clave primaria), DENOMINACIÓN y TIPO

En la Figura 1.11 se representa el diagrama de estructuras del ejercicio:

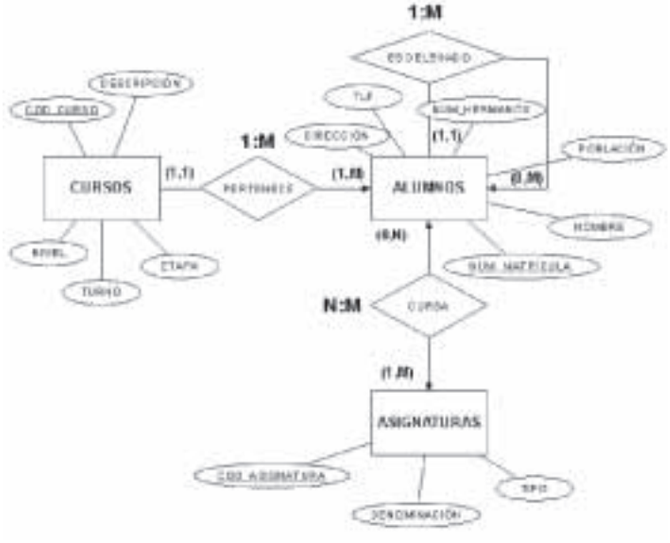


Figura 1.11. *Diagrama de estructuras en el modelo E-R.*

Actividades propuestas

1 Se desea realizar el diagrama de estructuras de datos en el modelo E-R, correspondiente al siguiente enunciado:

Supongamos el bibliobús que proporciona un servicio de préstamos de libros a los socios de un pueblo. Los libros están clasificados por temas. Un tema puede contener varios libros. Un libro es prestado a muchos socios, y un socio puede coger varios libros. En el préstamo de libros es importante saber la Fecha de préstamo y la Fecha de devolución. De los libros nos interesa saber el título, el autor y el número de ejemplares.

Generalización y jerarquías de generalización

Las generalizaciones proporcionan un mecanismo de abstracción que permite especializar una entidad (que se denominará *supertipo*) en subtipos, o lo que es lo mismo generalizar los subtipos en el supertipo.

Una generalización se identifica si encontramos una serie de atributos comunes a un conjunto de entidades, y unos atributos específicos que identificarán unas características.

Los atributos comunes describirán el supertipo y los particulares los subtipos. Una de las características más importantes de las jerarquías es la herencia, por la que los atributos de un supertipo son heredados por sus subtipos. Si el supertipo participa en una relación los subtipos también participarán.

Por ejemplo, en una empresa de construcción podremos identificar las siguientes entidades:

- EMPLEADO, con los atributos N_EMPLE (clave primaria,) NOMBRE, DIRECCIÓN, FECHA_NAC, SALARIO y PUESTO.
- ARQUITECTO, con los atributos de empleado más los atributos específicos: COMISIONES, y NUM_PROYECTOS.
- ADMINISTRATIVO, con los atributos de empleado más los atributos específicos: PULSACIONES y NIVEL.
- INGENIERO, con los atributos de empleado más los atributos específicos: ESPECIALIDAD y AÑOS_EXPERIENCIA.

En la Figura 1.12 se representa este ejemplo de generalización.

La generalización es total si no hay ocurrencias en el supertipo que no pertenezcan a ninguno de los subtipos, es decir, que los empleados o son arquitectos, o son administrativos, o son aparejadores, no pueden ser varias cosas a la vez. En este caso, la generalización sería también exclusiva. Si un empleado puede ser varias cosas a la vez la generalización es *solapada* o *superpuesta*.

La generalización es *parcial* si existen empleados que no son ni ingenieros, ni administrativos, ni arquitectos. También puede ser *exclusiva* o *solapada*. Las cardinalidades en estas relaciones son siempre (1,1) en el supertipo y (0,1) en los subtipos, para las exclusivas. (0,1) o (1,1) en los subtipos para las solapadas o superpuestas.

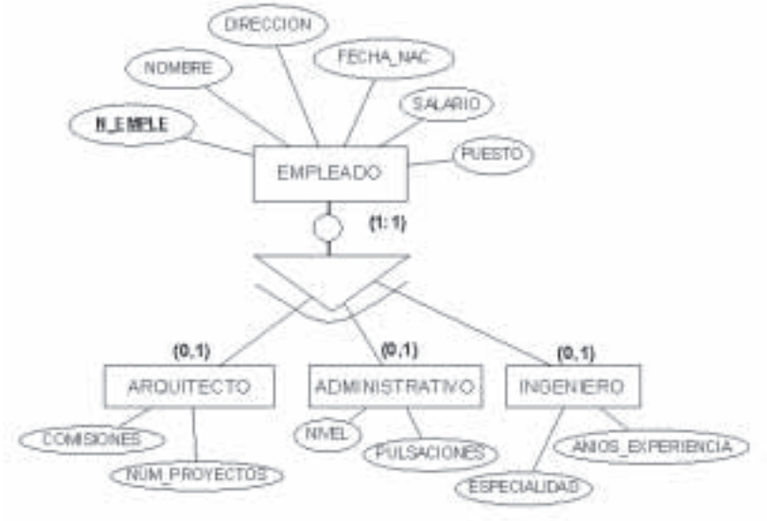


Figura 1.12. *Representación de una generalización.*

Así pues, habrá jerarquía solapada y parcial (que es la que no tiene ninguna restricción) solapada y total, exclusiva y parcial, y exclusiva y total. En la Figura 1.13 se muestran cómo se representan.

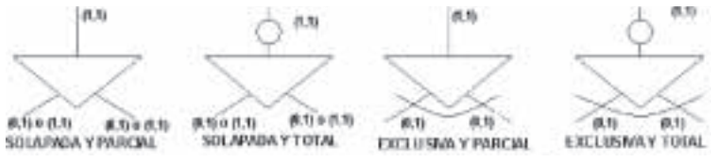


Figura 1.13. *Tipos y representación de jerarquías.*

Actividades propuestas

2 **Representa las siguientes jerarquías e indica el tipo de generalización:**

Un concesionario de coches vende coches nuevos y usados. Los atributos específicos de los nuevos son las unidades y el descuento; de los usados son los kilómetros y el año de fabricación.

Consideramos el conjunto de personas de una ciudad, distinguimos a los trabajadores, estudiantes y parados. De los trabajadores nos interesa el número de la Seguridad Social, la empresa de trabajo y el salario. De los estudiantes, el número de matrícula y el centro educativo, y de los parados la fecha del paro.

En un campo de fútbol los puestos de los futbolistas pueden ser: portero, defensa, medio y delantero.

Agregación

Una limitación del modelo E-R es que no es posible expresar relaciones entre relaciones. En estos casos se realiza una agregación, que es una abstracción a través de la cual las relaciones se tratan como entidades de nivel más alto. Por ejemplo, consideramos una relación entre EMPLEADOS y PROYECTOS, un empleado trabaja en varios proyectos durante unas horas determinadas y en ese trabajo utiliza unas herramientas determinadas. La representación del diagrama de estructuras se muestra en la Figura 1.14 siguiente:

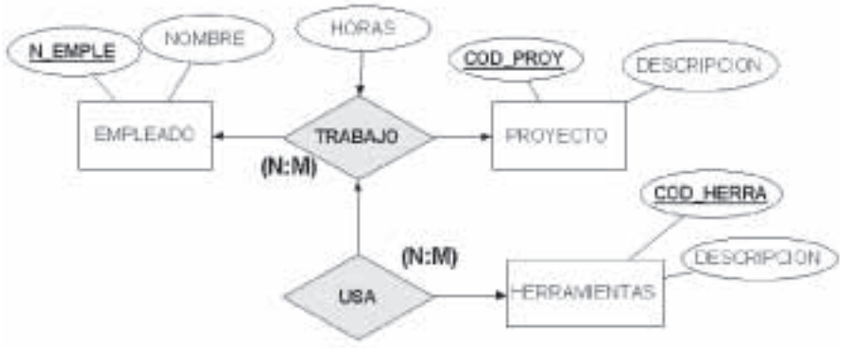


Figura 1.14. *Diagrama E-R de una relación entre otra relación.*

Si consideramos la agregación, tenemos que la relación TRABAJO con las entidades EMPLEADO y PROYECTO se pueden representar como un conjunto de entidades llamadas TRABAJO, que se relacionan con la entidad HERRAMIENTAS mediante la relación USA. Ver Figura 1.15:

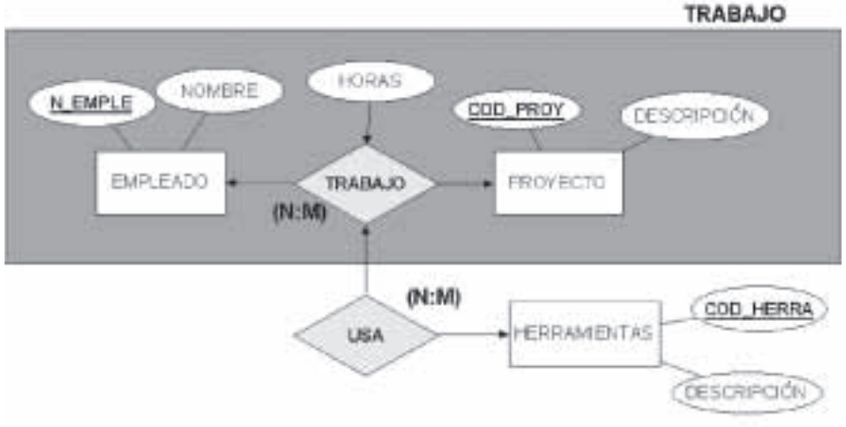


Figura 1.15. *Conjunto de entidades y relaciones para representar una relación entre una relación.*

Actividades propuestas

3 **Se desea realizar el diagrama de estructuras de datos en el modelo E-R correspondiente al siguiente enunciado:**

Una compañía de distribución de productos para el hogar dispone de proveedores que le suministran artículos. Un artículo sólo puede proveerlo un proveedor.

La empresa tiene tres tipos de empleados: oficinistas, transportistas y vendedores. Estos últimos venden los artículos. Un artículo es vendido por varios vendedores, y un vendedor puede vender varios artículos en distintas zonas de venta. De las ventas nos interesa saber la fecha de venta y las unidades vendidas.

1.6 Modelo de red

Este modelo utiliza estructuras de datos en red, también conocidas como *estructuras plex*. Las entidades se representan como registros o nodos, y las relaciones como enlaces o punteros. En una estructura red cualquier componente puede vincularse con cualquier otro. Es posible describirla en términos de padres e hijos, pero, a diferencia del modelo jerárquico, un hijo puede tener varios padres.

Las representaciones lógicas basadas en árboles o en estructuras plex, a menudo, limitan el cambio que el crecimiento de la BD exige, hasta tal punto que las representaciones lógicas de los datos pueden variar afectando a los programas de aplicación que usan esos datos. Los conceptos básicos de este modelo son los siguientes:

- **Elemento:** es un campo de datos. Ejemplo: DNI.
- **Agregados de datos:** conjunto de datos con nombre. Ejemplo: Fecha (día, mes, año).
- **Tipos de registro:** representa un nodo, un conjunto de campos. Cada campo contiene elementos. Es la unidad básica de acceso y manipulación. Se asemeja a los registros en archivos o las entidades en el modelo E-R.
- **Conjunto:** colección de dos o más tipos de registro que establece una vinculación entre ellos. Uno de ellos se llama propietario y el otro, miembro. Tienen una relación muchos a muchos (M:M), que para representarla se necesita un registro conector. Los conjuntos están formados por un solo registro propietario y uno o más registros miembros. Un registro propietario no puede ser a la vez miembro de sí mismo. Una ocurrencia del conjunto está formada por un registro propietario y el resto son registros miembros. Una ocurrencia de registro no puede pertenecer a varias ocurrencias del mismo conjunto.
- **Ciclo:** se forma cuando un registro miembro tiene como descendientes a uno de sus antepasados.
- **Bucle, lazo o loop:** es un ciclo en el que los registros propietarios y miembros son el mismo.

Diagramas de estructura de datos en un modelo en red

El **diagrama** es el esquema que representa el diseño de una BD de red. Se basa en representaciones entre registros por medio de enlaces. Existen relaciones en las que participan solo dos entidades (binarias) y relaciones en las que participan más de dos entidades (generales) ya sea con o sin atributo descriptivo en la relación. Se utilizan cuadros o celdas para representar los registros y líneas para representar los enlaces.

Una ocurrencia del esquema son los valores que toman los elementos del esquema en un determinado momento. El modelo es muy flexible pues no hay restricciones. Esto implica la gran dificultad a la hora de implementarlo físicamente a la larga es poco eficiente. Para representar físicamente este modelo se pueden usar punteros y listas. Este modelo es sólo teórico, a la hora de llevarlo a la práctica se introducen las restricciones necesarias.

El diagrama de estructura de datos de red especifica la estructura lógica global de la BD; su representación gráfica se basa en la colocación de los campos de un registro en un conjunto de celdas que se ligan con otro(s) registro(s).

Vamos a suponer que tenemos una BD de red con datos de alumnos y asignaturas.

Registro ALUMNO:

```
struct ALUMNO
 {  int NMatricula;
    char Nombre[35];
    int Curso;
    char Direccion[30];
 };
```

Registro ASIGNATURA:

```
struct ASIGNATURA
 {  int Codigo;
    char NomAsig[35];
 };
```

Entre ALUMNOS y ASIGNATURA existe la relación de ALUMNO-CURSA-ASIGNATURA, que en el modelo E-R se representa como muestra la Figura 1.16:

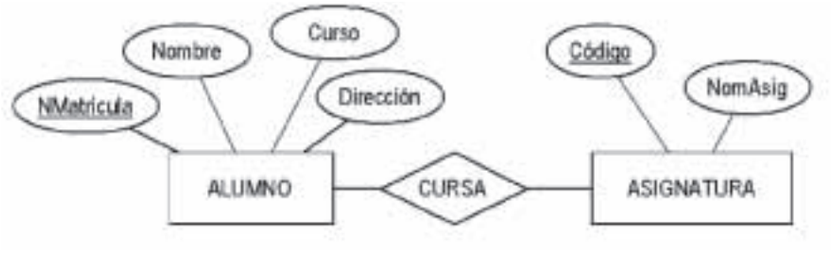


Figura 1.16 *Modelo E-R de ALUMNO-CURSA-ASIGNATURA.*

Las estructuras de datos según la cardinalidad se representan en los siguientes casos:

- **Representación del diagrama de estructura cuando el enlace o la relación no tiene atributos descriptivos**

  – *Caso 1*. Cardinalidad uno a uno (ver Figura 1.17).

Figura 1.17. *Relación uno a uno sin atributos, en el modelo en red.*

  – *Caso 2*. Cardinalidad uno a muchos, un alumno cursa muchas asignaturas (ver Figura 1.18).

Figura 1.18. *Relación uno a muchos, en el modelo en red.*

  – *Caso 3*. Cardinalidad muchos a muchos, muchos alumnos cursan muchas asignaturas (ver Figura 1.19).

Figura 1.19. *Relación muchos a muchos en el modelo en red.*

En la Figura 1.20 se muestra un ejemplo de ocurrencia M:N:

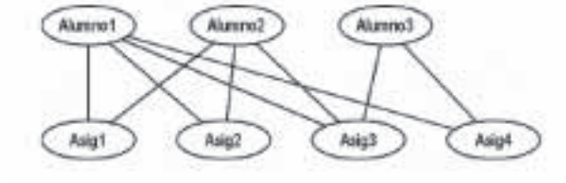


Figura 1.20. *Ejemplo de ocurrencias M:N en el modelo en red.*

- **Cuando el enlace tiene atributos descriptivos**

  Suponemos ahora que en la relación entre ALUMNOS y ASIGNATURAS necesitamos saber la Nota de un alumno en una asignatura, es decir, la relación lleva un atributo descriptivo. El diagrama del modelo E-R se muestra en la Figura 1.21:

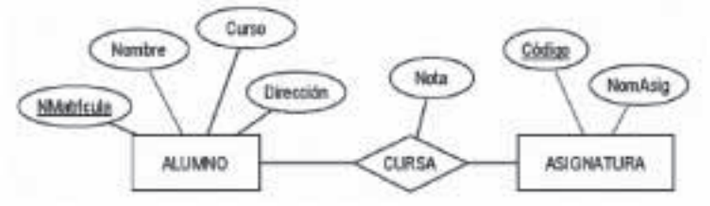


Figura 1.21. *Modelo E-R de ALUMNO-CURSA-ASIGNATURA con el atributo NOTA.*

Para convertirlo al diagrama de estructura de datos realizaremos lo siguiente:

1. Se representan los campos del registro como antes.

2. Se crea un nuevo registro, que llamaremos NOTAS, con el campo Nota.

3. Se crean los enlaces indicando la cardinalidad. A los enlaces les daremos los nombres: ALUMNOTA para la relación entre ALUMNO y NOTAS, y ASIGNOTAS, para la relación entre ASIGNATURAS y NOTAS.

- **Dependiendo de la cardinalidad los diagramas de estructuras de datos se transforman como sigue**

  – *Caso 1*. Cardinalidad uno a uno. Un alumno cursa una asignatura con una sola nota, en la Figura 1.22 se representa el diagrama.

Figura 1.22. *Representación de la cardinalidad 1:1 en el modelo en red.*

  – *Caso 2*. Cardinalidad uno a muchos. Un alumno cursa muchas asignaturas que tienen una nota (ver Figura 1.23).

Figura 1.23. *Representación de la cardinalidad 1:M en el modelo en red.*

– *Caso 3*. Cardinalidad muchos a muchos. Muchos alumnos cursan muchas asignaturas con muchas notas (ver Figura 1.24).

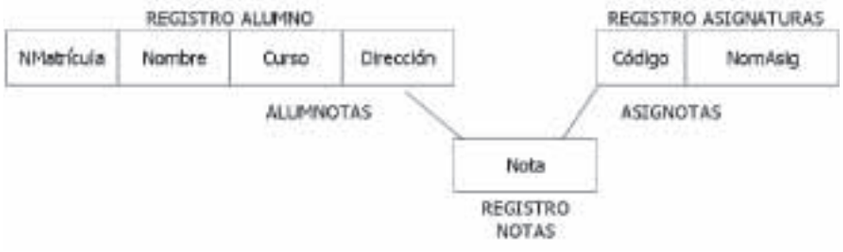


Figura 1.24. *Representación de la cardinalidad N:M en el modelo en red.*

- **Consideramos ahora que en la relación intervienen más de dos entidades y no hay atributos descriptivos en la relación.**

  Suponemos que nos interesa saber los profesores que imparten las asignaturas y agregamos a la relación ALUMNO-CURSA-ASIGNATURA la entidad PROFESOR. El diagrama E-R queda como se muestra en la Figura 1.25:

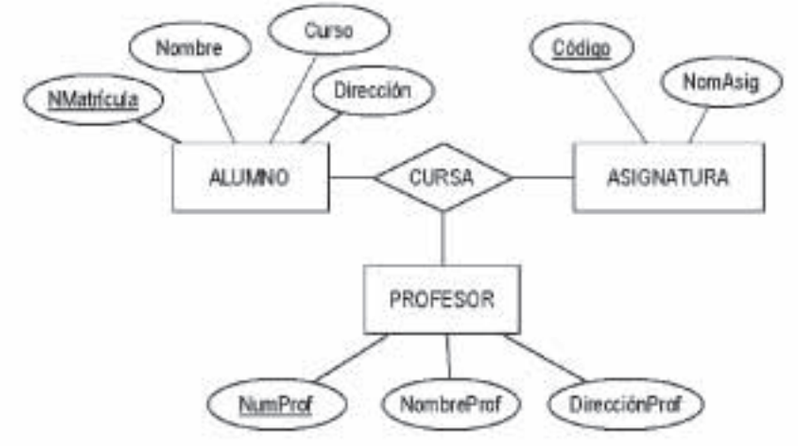


Figura 1.25. *Representación en el modelo E-R de la relación entre ALUMNO-ASIGNATURA-PROFESOR.*

Para realizar la transformación a diagramas de estructura de datos seguimos los siguientes pasos:

1. Se crean los registros para cada una de las entidades que intervienen.

2. Se crea un nuevo tipo de registro que llamaremos ENLACE, que puede no tener campos o tener sólo uno que contenga un identificador único, el identificador lo proporcionará el sistema y no lo utiliza directamente el programa de aplicación. A este registro se le denomina también como *registro de enlace* o *conector*.

– Considerando una relación con cardinalidad uno a uno, el diagrama de estructuras se muestra en la Figura 1.26:

Figura 1.26. *Diagrama de estructuras de la relación uno a uno entre 3 entidades.*

Si el registro enlace tuviese atributos, estos se añadirán y se enlazarán indicando el tipo de cardinalidad de que se trate. Por ejemplo, añadimos el atributo Nota a la relación CURSA (ver Figura 1.27).

Figura 1.27. *Diagrama de estructuras de la relación uno a uno con registro de enlace y atributos.*

Si consideramos que un profesor imparte una sola asignatura y los alumnos tienen nota en una sola asignatura y esa asignatura sólo es cursada por un alumno, una instancia de este modelo sería la que se muestra en la Figura 1.28:

Figura 1.28. *Representación de una ocurrencia de la relación PROFESOR-ALUMNO-ASIGNATURA.*

El registro enlace contendrá los punteros para acceder a cada uno de los registros.

– Si ahora consideramos una cardinalidad muchos a muchos, es decir, un alumno cursa muchas asignaturas y el profesor imparte muchas asignaturas: una asignatura es impartida por muchos profesores y una asignatura es cursada por muchos alumnos, el diagrama de estructuras resultante se muestra en la Figura 1.29.

Figura 1.29. *Diagrama de estructuras para una relación M:M.*

Modelo de datos de CODASYL

Los SGDB que utilizan este modelo de red deben añadir restricciones a la hora de implementar físicamente la BD y obtener un mayor rendimiento del sistema. En 1971, un grupo conocido como CODASYL (*Conference on Data System Languages*) desarrolla el modelo DBTG (*Data Base Task Group*). Este grupo es el que desarrolló los estándares para COBOL. El modelo CODASYL ha evolucionado durante los últimos años y existen diversos productos SGBD orientados a transacciones, sin embargo actualmente estos productos se utilizan muy poco. El modelo es bastante complejo de implementar y los diseñadores y programadores deben de tener mucho cuidado al elaborar BBDD y aplicaciones DBTG. Además este modelo está enfocado al COBOL. Gran parte de las deficiencias detectadas son atribuye a que este modelo fue desarrollado antes de que se establecieran los conceptos esenciales de la tecnología de bases de datos. Ejemplos de BBDD en red son el DMS 1100 de UNIVAC, el EDMS de Xerox, el PHOLAS de Philips y el DBOMP de IBM.

En el modelo DBTG sólo pueden emplearse enlaces uno a uno y uno a muchos. En este modelo, existen dos elementos principales que son el propietario y el miembro, donde sólo puede existir un propietario y varios miembros, y cada miembro depende sólo de un propietario.

- **Conjuntos DBTG**

En este modelo sólo se utilizan enlaces uno a uno y uno a muchos. Podemos representar este modelo como se muestra en la Figura 1.30:

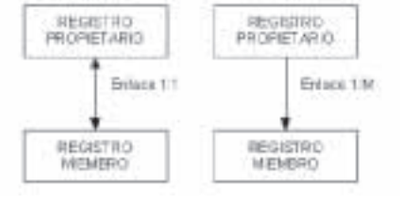


Figura 1.30. *Enlaces 1:1 y 1:M en el modelo DBTG. Conjunto DBTG.*

En el modelo DBTG esta estructura se denomina **conjunto DBTG** o **SET**. El nombre que se le asigna al conjunto suele ser el mismo que el de la relación que une a las entidades. Cada conjunto DBTG puede tener cualquier número de ocurrencias. Puesto que no se permiten enlaces del tipo muchos a muchos, cada ocurrencia del conjunto tiene exclusiva-

mente un propietario y cero o más registros miembros. Además, ningún registro puede participar en más de una ocurrencia del conjunto en ningún momento. Sin embargo, un registro miembro puede participar simultáneamente en varias ocurrencias de diferentes conjuntos. Añadir que un mismo registro no puede ser miembro y propietario a la vez, no está admitida la reflexividad.

En la Figura 1.31 se representan varias ocurrencias de un conjunto DBTG. La tercera ocurrencia no es válida pues todas pertenecen al mismo conjunto, y el miembro 3 no puede tener dos propietarios (ver Figura 1.31).

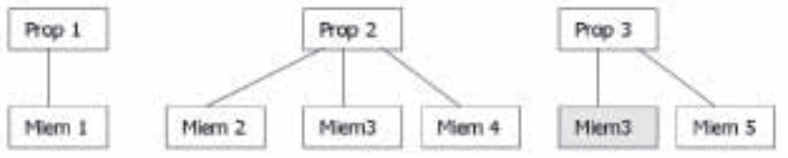


Figura 1.31. *Ocurrencias de un conjunto DBTG.*

Para declarar los conjuntos de un diagrama de estructuras como el que se muestra en la Figura 1.32 realizamos lo siguiente:

Figura 1.32. *Ejemplo diagrama de estructuras. Relación 1:M.*

En el diagrama existen dos conjuntos DBTG:

ALUMNOTAS, cuyo propietario es ALUMNO y cuyo miembro es RNOTAS.
ASIGNOTAS, cuyo PROPIETARIO es ASIGNATURAS y miembro RNOTAS.

La declaración de los conjuntos será:

```
Set name is ALUMNOTAS
   Owner is ALUMNO
  member is RNOTAS
```

```
Set name is ASIGNOTAS
   Owner is ASIGNATURAS
  member is RNOTAS
```

Al igual que otros modelos de datos, el DBTG proporciona un lenguaje de comandos para la manipulación de datos, así como para la actualización y el procesamiento de conjuntos DBTG.

1.7 Modelo jerárquico

El modelo jerárquico es similar al modelo de red. Los datos y las relaciones se representan mediante registros y enlaces. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles. El modelo jerárquico se sirve de árboles para la representación lógica de los datos, su implementación se lleva a cabo mediante árboles y punteros.

No se ha llegado a una formalización matemática del modelo y de sus lenguajes, como ha ocurrido en el caso del modelo relacional; tampoco se ha intentado su estandarización, como en el caso del DBTG. Los primeros sistemas de BBDD jerárquicas aparecieron en 1968: el IMS/360 (*Information Management System*) con su lenguaje de datos el DL/I de IBM, y el SYSTEM 2000 de Intel. Los productos basados en este tipo de modelos han perdido las altas cuotas de mercado de las que disfrutaban hace una década y se consideran sistemas muy superados por la tecnología relacional, aunque aún persisten muchas aplicaciones basadas en este modelo.

Características del modelo jerárquico

La representación gráfica del modelo jerárquico se realiza mediante un árbol invertido, en el que el nivel superior está formado por una única entidad o segmento bajo el cual cuelgan el resto de entidades o segmentos en niveles que se van ramificando. Los diferentes niveles quedan unidos por medio de las relaciones. El nivel más alto de la jerarquía tiene un nodo que se llama *raíz*. Cada nodo representa un tipo de registro llamado segmento con sus correspondientes campos. Los *segmentos* se organizan de manera que en un mismo nivel están todos aquellos que dependen de un segmento de nivel inmediatamente superior.

Los segmentos, en función de su situación en el árbol, se denominan:

- **Segmento padre:** es el que tiene descendientes, todos ellos localizados en el mismo nivel.
- **Segmento hijo:** es el que depende de un segmento de nivel superior. Los hijos de un mismo padre están en el mismo nivel.
- **Segmento raíz:** es el padre que no tiene padre. Ocupa el nivel superior del árbol. El segmento raíz es único.

Por ejemplo, tenemos la BD de un centro escolar llamada CENTROESCOLAR, que cuenta con 5 segmentos:

– El segmento raíz almacena los datos de los CURSOS que se imparten en el centro por ejemplo: Código, Descripción, Tipo enseñanza, Turno y Nivel.

– En el segundo nivel del árbol hay tres segmentos dependientes del segmento raíz. El primero de ellos contiene los datos de los PROFESORES que imparten clase en el curso: Código profesor, Nombre profesor, Dirección, Tlf y Especialidad. El segundo contiene los datos de los ALUMNOS que están matriculados en el curso, por ejemplo:

Número de matrícula, Nombre alumno, Dirección, Tlf, Fecha nacimiento. Y el tercero contiene los datos de las ASIGNATURAS que se imparten en el curso: Código, Descripción y Tipo asignatura.

– El tercer nivel del árbol está ocupado por un solo segmento que depende del segmento ASIGNATURAS y que contiene los datos de las AULAS donde se imparten las asignaturas: Código de aula, Localización y Tipo de aula. En la Figura 1.33. se muestra el esquema jerárquico de estos 5 segmentos a 3 niveles.

Figura 1.33. *Esquema jerárquico de 5 segmentos y 3 niveles.*

Definiciones del modelo jerárquico:

- Una OCURRENCIA de un segmento de una BD jerárquica es el conjunto de valores que toman todos los campos que lo componen en un momento determinado.

- Un REGISTRO es el conjunto formado por una ocurrencia del segmento raíz y todas las ocurrencias del resto de los segmentos de la BD que dependen jerárquicamente de dicha ocurrencia raíz.

- La relación PADRE/HIJO en la que se apoyan las BBDD jerárquicas, determina que el camino de acceso a los datos sea ÚNICO; este camino, denominado CAMINO SECUENCIA JERÁRQUICA, comienza siempre en una ocurrencia del segmento raíz y recorre la BD de arriba abajo, de izquierda a derecha y, por último, de adelante atrás.

Una estructura jerárquica, tiene las siguientes características:

– El árbol se organiza en un conjunto de niveles.

– El nodo raíz, el más alto de la jerarquía, se corresponde con el nivel 0.

– Las líneas que unen los nodos se llaman camino, no tienen nombre, ya que entre dos conjuntos de datos sólo puede haber una interrelación.

– Al nodo de nivel inferior sólo le puede corresponder un único nodo de nivel superior, es decir, un padre puede tener varios hijos y un hijo sólo tiene un padre.

- Todo nodo, a excepción del nodo raíz, ha de tener obligatoriamente un padre.
- Se llaman **hojas** a los nodos que no tienen hijos.
- Se llama **altura** al número de niveles de la estructura jerárquica.
- Se denomina **momento** al número de nodos.
- El número de hojas del árbol se llama **peso**.
- Sólo están permitidas las interrelaciones 1:1 ó 1:N
- Cada nodo no terminal y sus descendientes forman un subárbol, de forma que un árbol es una estructura recursiva.
- La suma total de un nodo padre y sus hijos se llama **familia**.

Actividades propuestas

4 **Dado el árbol que se muestra en la Figura 1.34, escribe sus características: los niveles, el número de hojas, altura, peso, recorrido y momento.**

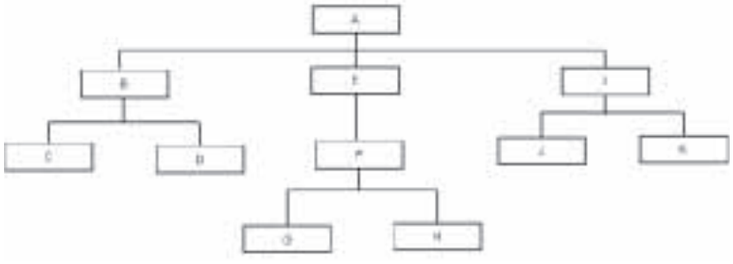


Figura 1.34. *Árbol jerárquico para la Actividad 4.*

Para acceder a la información de las BBDD jerárquicas se hace un recorrido ordenado del árbol. Se suele recorrer en preorden; es decir, raíz, subárbol izquierdo y subárbol derecho.

Las restricciones que presenta este modelo respecto a otros son las siguientes:

- Cada árbol debe tener un único segmento raíz.
- No puede definirse más de una relación entre dos segmentos dentro de un árbol.

- No se permiten las relaciones reflexivas de un segmento consigo mismo.
- No se permiten las relaciones N:M, esto implica la repetición de registros o la redundancias de datos.
- No se permite que exista un hijo con más de un padre.
- Para cualquier acceso a la información almacenada, es obligatorio el acceso por la raíz del árbol, excepto en el caso de utilizar un índice secundario.
- El árbol debe recorrer, siempre de acuerdo a un orden prefijado, el camino jerárquico.
- La estructura del árbol, una vez creada, no se puede modificar. La actualización de la estructura de la BD es bastante complicada, no se podrá insertar un nodo hijo si aún no tiene asignado un padre. La baja de un registro implica que desaparezca todo el subárbol que tiene dicho registro como nodo raíz.

Transformación de un esquema E-R en un esquema jerárquico

Interrelaciones 1:N con cardinalidad mínima 1 en la entidad padre: en este caso, no hay problemas. La representación es igual que en el modelo E-R. En la Figura 1.35 se representa una relación PROFESOR-ALUMNO. Un profesor puede tener muchos alumnos, y un alumno sólo pertenece a un profesor.

Interrelaciones 1:N con cardinalidad mínima 0 en el registro padre, en este caso pueden existir hijos sin padre, por lo que se crea un padre ficticio o se crean dos estructuras arborescentes. La primera estructura arborescente tendrá como nodo padre el registro PROFESOR, y como nodo hijo los identificadores del registro ALUMNO (la clave). De esta forma, no se introducen redundancias estando los atributos de ALUMNO en la segunda arborescencia en la cual sólo existe un nodo raíz ALUMNO sin descendientes. Ver Figura 1.35:

Figura 1.35. *Representación de relaciones 1:M en el modelo jerárquico.*

Si representamos una ocurrencia del segmento padre PROFESOR, observamos que cada profesor tendrá un número de ocurrencias de segmentos hijos, es decir, su propio árbol. Podemos afirmar que una BD jerárquica está formada por un conjunto de árboles disjuntos. Existirá un árbol para cada profesor. Ver Figura 1.36.

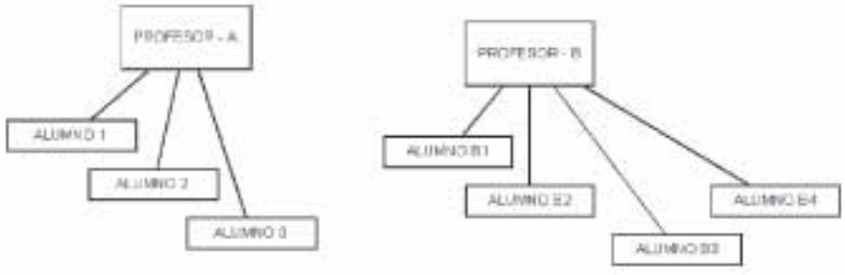


Figura 1.36. *Representación de ocurrencias del segmento PROFESOR.*

- **Interrelaciones N:M.** La solución es parecida a la anterior. Se crean dos arborescencias, en una de ellas se representa el nodo padre PROFESOR con los identificadores del nodo ALUMNO, y en el otro árbol se crean el nodo ALUMNO, sus atributos, relacionados con los identificadores del nodo padre PROFESOR. Ver Figura 1.37:

Figura 1.37. *Representación de relaciones M:N en el modelo jerárquico.*

- **Interrelaciones reflexivas.** En la Figura 1.38 se muestra la interrelación de PIEZA-FORMA PARTE DE-PIEZA, que representa a una PIEZA de un sistema de fabricación de productos de automóviles, por ejemplo, que forma parte de otra PIEZA. Una pieza es componente de otras piezas y está compuesta a su vez de otras piezas. Es una relación reflexiva M:N. Se utiliza la jerarquía de la izquierda si se desea obtener la explosión, es decir, obtener las piezas que forman parte de una pieza, y de la derecha para la implosión, es decir, obtener a partir de una pieza aquellas de las que forma parte.

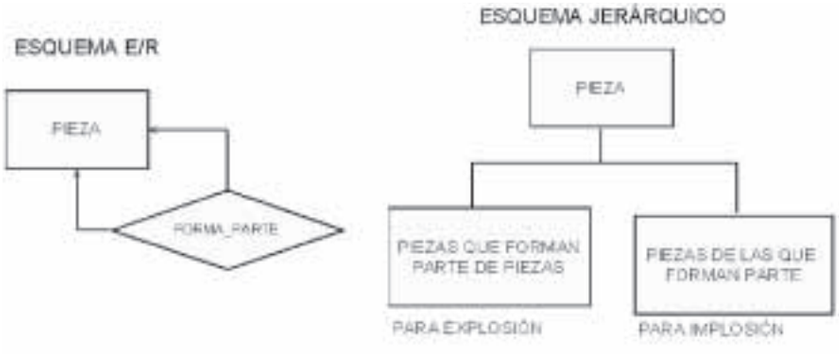


Figura 1.38. *Representación de relaciones reflexivas en el modelo jerárquico.*

La aplicación de todas estas normas de diseño evita la introducción de redundancias y la pérdida de simetría del árbol, sin embargo, complica el esquema jerárquico resultante pues estará formado por muchos árboles.

Actividades propuestas

5 **Representa el modelo jerárquico del diagrama E-R que se muestra en la Figura 1.39. Representa también una ocurrencia.**

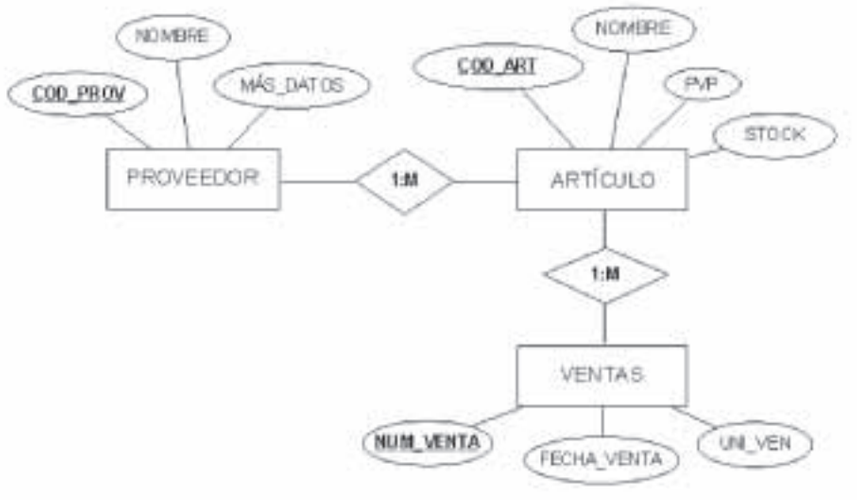


Figura 1.39. *Diagrama E-R para la Actividad.*

1.8 Modelo orientado a objetos

El modelo de datos orientado a objetos surge por las limitaciones del modelo relacional, sobre todo a la hora de abordar tipos de datos más complejos, y por la falta de capacidad semántica del modelo relacional para desarrollar aplicaciones en áreas como el diseño asistido por ordenador, la ingeniería del software, los sistemas basados en el conocimiento y el tratamiento de documentos, multimedia y gestión de redes, que requieren modelar objetos e interrelaciones más complejas.

Este modelo está basado en el paradigma de la programación orientada a objetos (POO). Los SGBO (Sistemas de Gestión de Bases de Objetos) o SDBDOO (Sistemas de Gestión de Bases de Datos Orientados a Objetos) gestionan objetos en los cuales están encapsulados los datos y las operaciones que interactúan con ellos. Además proporcionan un modelo único de datos. No hay diferencia entre el modelo conceptual (el modelo E-R) y el modelo lógico (el relacional), y las aplicaciones pueden acceder directamente al modelo.

Las SGBDOO adoptan una arquitectura que consta de un sistema de gestión que soporta un lenguaje de BBDD orientado a objetos, con una sintaxis similar a un lenguaje de programación también orientado a objetos, como puede ser C++ o Java. El lenguaje de BBDD es especificado mediante un lenguaje de definición de datos (ODL), un lenguaje de manipulación de datos (OML), y un lenguaje de consulta (OQL), siendo todos ellos portables a otros sistemas.

Estructura de los objetos

A nivel conceptual un objeto va a ser lo que una entidad en el modelo E-R, a la hora de implementarlo el objeto es un encapsulamiento de un conjunto de operaciones: código y datos en una única unidad, cuyo contenido no es visible desde el exterior. Esta ocultación de la información permite modificar aspectos del objeto sin que afecte a los demás que interactúan con él.

Las interacciones entre un objeto y el resto del sistema se realizan mediante un conjunto de mensajes.

Un objeto está formado por:

- Un conjunto de propiedades o atributos que contienen los datos del objeto, serían los atributos del modelo E-R.
- Un conjunto de mensajes a los que el objeto responde.
- Un conjunto de métodos, que es código para implementar los mensajes. Un método devuelve un valor como respuesta al mensaje.

En la Figura 1.40 podemos ver la representación de un objeto ALUMNO.

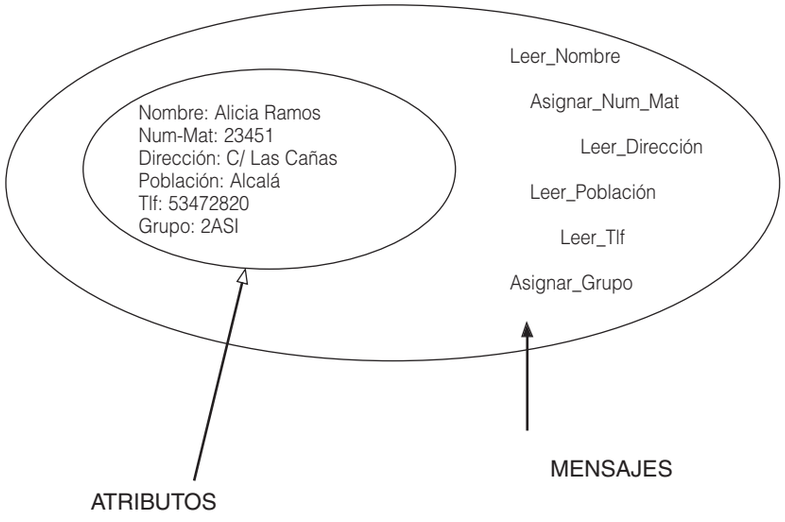


Figura 1.40. *Representación del objeto ALUMNO.*

Jerarquía de clases

Una **clase** es un conjunto de objetos similares. A cada uno de estos objetos se le llama *instancia de su clase*. Todos los objetos de la clase comparten una definición común y se comportan de la misma forma, aunque difieran en los valores asignados a los atributos. Podemos hacer una equivalencia entre entidad del modelo E-R y clase del modelo OO.

A continuación, se muestra la clase EMPLEADO en la que vemos los atributos y los mensajes:

```
class empleado
{
/* atributos */
string nombre;
string dirección;
date fecha_alta;
int salario;
/* mensajes */
string leer_nombre();
string leer_dirección();
date leer_fecha_alta();
int leer_salario();
};
```

Los métodos para el manejo de los mensajes no se han incluido en la clase.

Herencia

Las clases, en un sistema orientado por objetos, se representan en forma jerárquica como en las jerarquías de generalización en el modelo E-R. Hay superclases y subclases. La subclase hereda todos los atributos, mensajes y métodos de la superclase. Suponemos ahora que dentro de la clase EMPLEADO se consideran subclases oficinistas y transportistas. Cada una de ellas heredará los atributos y mensajes de la superclase, y además tendrá otros atributos y otros mensajes. Para identificarlas llevan la palabra clave `isa`. Ver el ejemplo que se muestra:

```
class oficinista isa empleado
{
/* atributos */
string especialidad;
int pulsaciones;
/* mensajes */
string leer_especialidad();
int calculo_pulsaciones(int pulsa);
};
class transportista isa empleado
{
/* atributos */
string tipo_carnet;
int horas_acumuladas;
/* mensajes */
string leer_tipo_carnet();
int sumar_horas(int horas);
};
```

Hay dos enfoques para la creación de BDOO: extender los SGBDR relacionales para que sean capaces de soportar los conceptos de la programación orientada a objetos, añadiendo a los lenguajes de BBDD existentes, como es el caso del SQL3, tipos complejos de datos y la programación orientada a objetos (los SGBD que proporcionan extensiones orientadas a objetos a los sistemas relacionales se denominan Bases de datos relacionales orientadas a objetos). Otra opción es que soporten un modelo de objetos puro y no estar basados en extensiones.

Éstas cogen un lenguaje de POO existente y se amplía para que trabaje con las bases de datos.

Polimorfismo

Otra característica importante del paradigma de la orientación a objetos es el polimorfismo, es decir, la capacidad de que un mensaje sea interpretado de formas distintas según el objeto que lo recibe. El polimorfismo es conocido como la sobrecarga de operadores. Este concepto permite enlazar el mismo nombre o símbolo de operador a dos o más implementaciones diferentes del operador, dependiendo del tipo de objetos a los que éste se aplique.

El modelo de datos orientado a objetos

Los SGBDOO soportan un modelo de objetos puro, en la medida en que no están basados en extensiones de otros modelos como el relacional. Están influenciados por los lenguajes de POO. Ejemplos de SGBDOO son Poet, Jasmine, ObjectStore y GemStone. En las BDOO, la organización ODMG (*Object Data Management Group*), que representa el 100% de las BDOO industriales, ha establecido un estándar que intenta definir un SGBDOO que integre las capacidades de las BBDD con las capacidades de los lenguajes de programación orientados a objetos, de forma que los objetos de la BD aparezcan como objetos del lenguaje de programación. De esta manera, intentan eliminar la falta de correspondencia existente entre los sistemas de tipos de ambos lenguajes. El ODMG define:

- Un modelo de objetos estándar para el diseño de estas BBDD.
- Lenguaje de definición de objetos (ODL, *Object Definition Language*)
- Lenguaje de consultas (OQL, Object Query Language), estilo al SQL. Vinculación con los lenguajes C++, Java y Smalltalk. Definen un lenguaje de manipulación de objetos (OML-*Object Manipulation Language*) que extiende el lenguaje de programación para soportar objetos persistentes (cualidad de algunos objetos de mantener su identidad y relaciones con otros objetos con independencia del sistema o proceso que los creó). El objeto persistente es el que tiene vida aunque el programa que trabaja con él haya terminado, es decir, los datos permanecen después de la sesión del usuario y la ejecución del programa de la aplicación. Para que esto sea así el objeto será guardado en una BD. A los lenguajes que incorporan estas estructuras de datos se les llama *lenguajes persistentes*.

Podemos considerar los SGBDOO como gestores de 3.ª generación. También están incluidos las BD objeto-relacionales. La primera generación serían los SGBD jerárquicos y en red, que utilizaban estructuras de datos tipo listas y árboles para su implementación, y la 2.ª los SGBDR relacionales, que utilizan estructuras tipo tabla (ver Figura 1.41).

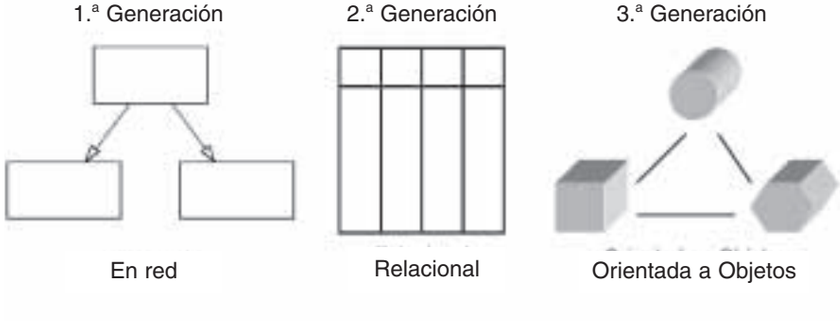


Figura 1.41. *Generaciones de modelos de BBDD.*

En la Tabla 1.3 se muestran los estándares y las BBDD más características de tercera generación:

BD OBJETO-RELACIONAL	BD OBJETOS PUROS
Estándar	Estándar
SQL: 1999 (SQL3), Melton 1999 SQL: 2003 , Melton 2003	ODMG-93, Cattell (1994), Cattell (1995) ODMG V.2.0, Cattell (1997) ODMG V.3.0, Cattell (2000)
Productos	Productos
Oracle (a partir de la V8) POSTGRES Universal Server de INFORMIX.	ObjectStore de Object Design. Persistencia de objetos en C++ y Java. POET. Persistencia de objetos en C++ y Java. Gemstone de Servi Logia, Meier y Stone (1987). Persistencia de objetos Smalltalk, soporta también C++ y Java.

Tabla 1.3. *Sistemas gestores de BBDD de 3.ª generación.*

Para representar el modelado de una BDOO se utilizan el diagrama de clases (que muestra la estructura de clases del sistema incluyendo las relaciones que puedan existir entre ellas) y el diagrama de objetos (que muestra un conjunto de objetos y sus relaciones en un momento determinado, equivale a lo que sería una instancia del diagrama de clases). Para realizar los modelados se utiliza el, **Lenguaje de Modelado Unificado** (*Unified Modeling Language-UML*). Es un lenguaje de modelado orientado a objetos que proporciona las técnicas para desarrollo de sistemas orientados a objetos. Es un modelado visual y utiliza diagramas para representar los esquemas del sistema.

Una clase se representa con un rectángulo dividido en tres secciones, mostrando en la primera el nombre de la clase (y, opcionalmente, otra información que afecte a la clase), en la segunda los atributos y en la última las operaciones. A continuación, se muestra la clase Empleado en la Figura 1.42.

Figura 1.42. *Clase empleado.*

Las relaciones en UML son conexiones semánticas entre clases. Existen cuatro tipos principales de relaciones en UML:

- **Generalización:** es una relación de especialización (Padre-hijo o superclase-subclase). La generalización se utiliza para modelar la herencia en los lenguajes orientados a objetos. Una de las características de la herencia es que permite simplificar la construcción de clases relacionadas, ya que gracias a ella es posible agrupar las características comunes de un conjunto de clases en una clase padre (superclase) y hacer que todas ellas hereden de la superclase. En el ejemplo se muestra la generalización de la clase Empleado, ver Figura 1.43:

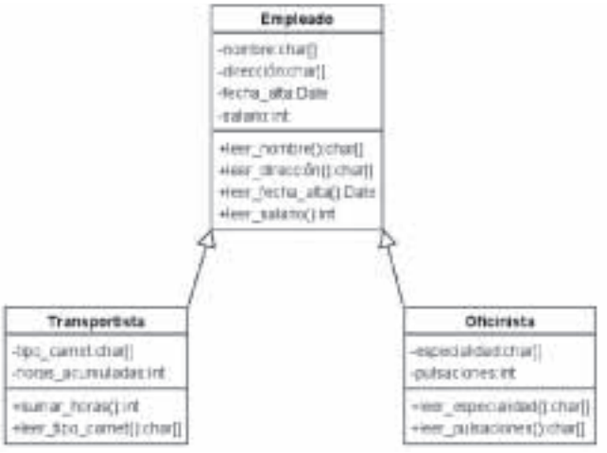


Figura 1.43. *Generalización de la clase Empleado.*

- **Asociación:** representan las relaciones entre instancias de clase. Se representa gráficamente como una línea que conecta las clases relacionadas. Si la clase Documento (A) se relaciona con la clase Persona (B), lo puede hacer de dos formas: en la primera, la clase A puede acceder a los atributos y operaciones públicas de la clase B y viceversa. Se representa con una línea, pero si añadimos una flecha que indique el sentido de dicha asociación restringimos la navegabilidad de la asociación. En el ejemplo, A puede acceder a los atributos y operaciones públicas de la clase B, pero B no puede acceder a A. Ver Figura 1.44:

Figura 1.44. *Representación de asociaciones.*

También una clase puede relacionarse consigo misma, es decir, pueden modelarse asociaciones reflexivas.

En las asociaciones se puede añadir información para aumentar su expresividad y significado:

- **El nombre de la asociación,** un nombre descriptivo que indica la naturaleza de la asociación. Se añade un pequeño triángulo que indique el sentido en que se debe interpretar.

- **Roles,** para indicar el rol que desempeña cada una de las clases en la asociación. Se identifica por un nombre a los finales de la línea, describe la semántica de la relación en el sentido indicado. El rol puede estar representado en el nombre de la clase.

– **Multiplicidad,** describe la cardinalidad de la relación, es decir, cuántos objetos de una clase participan en la relación. Se representa por: 1 (relaciones 1-1), * (0-muchos), 0..1 (0-uno) , 1..* (1-muchos) y m..n (para especificar un número, por ejemplo 7...10).

  En la figura que se muestra se representa una asociación entre las clases Documento y Persona, en la que indicamos el nombre de la asociación, los roles y la multiplicidad (ver Figura 1.45).

Figura 1.45. *Información de las asociaciones.*

- **Realización:** es una relación pactada y establecida entre clases, en la cual una clase especifica un contrato (por ejemplo, un interfaz) que la otra garantiza que cumplirá. Generalmente se emplea la realización para especificar una relación entre una interfaz y una clase o componente que implementa las operaciones especificadas en dicha interfaz.

  La interfaz especifica una serie de operaciones (cuya implementación estará en la clase) pero no proporciona ninguna implementación ni tiene atributos. Es equivalente a una clase abstracta que sólo tiene operaciones abstractas.

  Gráficamente, una interfaz se muestra como una clase con el estereotipo «interfaz» y sus operaciones, o en forma resumida como un círculo y debajo el nombre del interfaz.

  Para indicar que una clase realiza una interfaz, se muestra una línea discontinua acabada en flecha vacía que va de la clase que implementa la interfaz a la interfaz. Ver Figura 1.46:

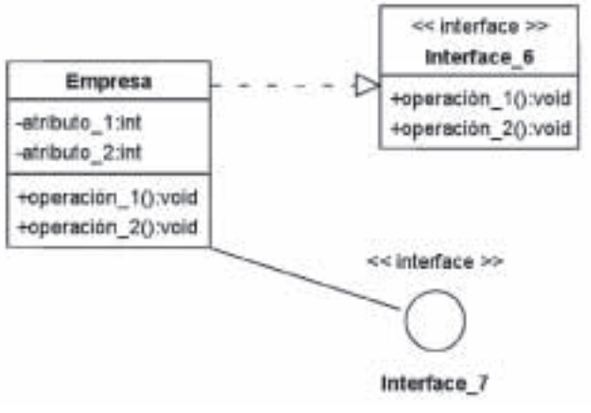


Figura 1.46. *Representación de una clase que realiza una interfaz.*

Cuando una clase realiza una interfaz, debe implementar todas las operaciones indicadas en la misma, además de las exclusivas de la clase. Las interfaces pueden evitar, en ciertos casos, el uso de herencia múltiple. Son muy útiles para independizar la implementación de sistemas o utilidades de los servicios u operaciones que proporcionan, fomentando la reutilización. Otra de las posibilidades de las interfaces consiste en limitar el acceso en las asociaciones entre clases. En el ejemplo que se muestra en la Figura 1.47 indicamos una asociación entre la clase Empresa y Persona a través de la interfaz Trabajador, que contiene los métodos que más interesan a la empresa. Esto evita que la empresa actúe con los datos médicos.

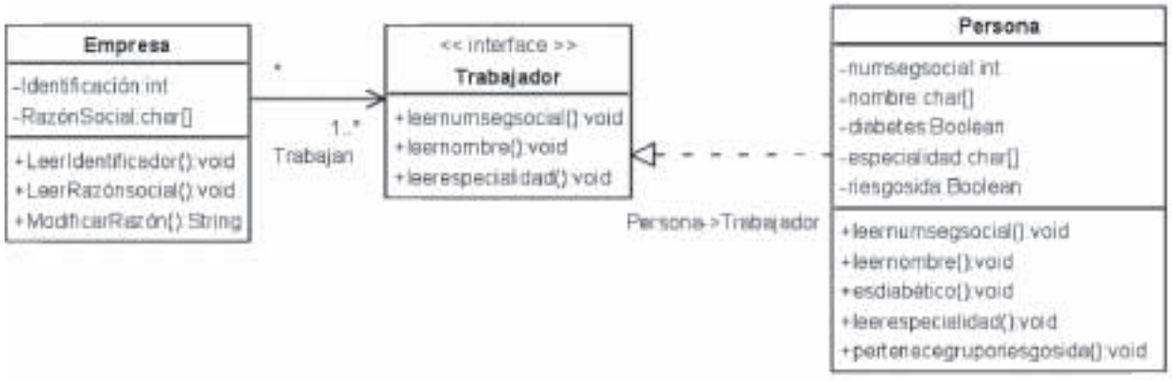


Figura 1.47. *Asociación a través de una interfaz.*

- **Dependencia:** es una relación entre entidades independientes y dependientes. El elemento dependiente (el cliente) requiere conocer al elemento independiente (el que proporciona el servicio) y que esté presente. Las dependencias se usan para modelar relaciones en las cuales un cambio en el elemento independiente (el suministrador) puede requerir cambios en el elemento dependiente (el cliente). Es un tipo de relación unidireccional, ya que el elemento dependiente debe conocer al independiente, pero el independiente desconoce la existencia del elemento dependiente. Gráficamente se muestra como una línea discontinua acabada en flecha que va del elemento dependiente al independiente (esto es, el elemento dependiente señala al elemento del que depende).

  Por ejemplo: la clase Cliente (dependiente) dependerá de servicios proporcionados por la clase Servidor (independiente), y un cambio en la clase Servidor puede ocasionar que la clase Cliente necesite ser adaptada. Un cambio en la clase Cliente no tiene ningún efecto sobre la clase Servidor, ya que esta última es independiente de la primera (no la necesita).

  Estas relaciones se utilizan cuando una clase usa los servicios de otra a través de una interfaz, es decir, depende de la interfaz. Un cambio en la interfaz requeriría un cambio en la clase dependiente. También se utilizan para mostrar la dependencia entre paquetes ya que las clases se suelen agrupar en paquetes.

Conceptos básicos

SGBD o DBMS (Sistema Gestor de Bases de Datos): colección de datos estructurados, organizados y relacionados entre sí, y el conjunto de programas que acceden y gestionan esos datos.

ANSI-SPARC (*American National Standard Institute - Standards Planning and Requirements Committee*): comité que propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal es el de separar los programas de aplicación de la BD física.

Diccionario de datos: es el lugar dónde se deposita información acerca de todos los datos que forman la BD. Contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica también los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información.

Modelos de datos: el instrumento principal para ofrecer la abstracción de los datos, se utilizan para la representación y el tratamiento de los problemas y los representan a tres niveles de abstracción. En la Figura 1.48 podemos ver la relación entre los modelos lógicos y conceptuales de datos.

DBTG (*Data Base Task Group*): modelo de datos lógico de red creado por CODASYL.

ODMG (*Object Data Management Group*): organización que propone los estándares para definir SGBDOO.

Objetos persistentes: cualidad de algunos objetos para mantener su identidad y relaciones con otros objetos, aunque el programa que trabaja con él haya terminado.

Lenguajes persistentes: Son los lenguajes que incorporan objetos persistentes en sus estructuras de datos y permiten que el objeto sea almacenado en la BD.

Lenguaje de Modelado Unificado (*Unified Modeling Language*): lenguaje de modelado orientado a objetos que proporciona las técnicas para desarrollo de sistemas orientados a objetos.

Objetos persistentes: Cualidad de algunos objetos de mantener su identidad y relaciones con otros objetos, aunque el programa que trabaja con él haya terminado.

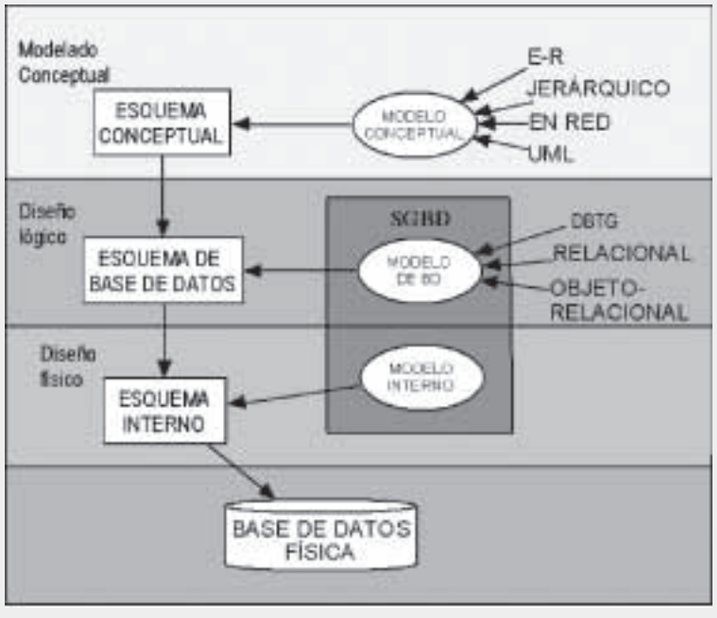


Figura 1.48. *Relación entre los modelos lógicos y conceptuales de datos.*

Actividades complementarias

1 **Realiza el diagrama de datos en el modelo E-R del enunciado:**
En una estación de autobuses contamos con unos autobuses que recorren una serie de lugares y que son conducidos por varios conductores. Se quiere representar los lugares que son recorridos por cada autobús, conducidos por cada conductor y la fecha en la que se visita el lugar. Define las entidades, los posibles atributos; identifica los atributos clave y los datos importantes para la relación entre las entidades. Indica también la cardinalidad.

2 **Realiza el diagrama de datos del ejercicio anterior en el modelo en red.**

3 **Realiza el diagrama de datos en el modelo E-R, que represente el siguiente enunciado:**
La Consejería de Educación gestiona varios tipos de centros: públicos, privados y concertados. Los privados tienen un atributo específico que es la cuota y los concertados la agrupación y la comisión. También asigna plazas a los profesores de la comunidad para impartir clase en esos centros. Un profesor puede impartir clase en varios centros.
Define las entidades, los posibles atributos; identifica los atributos clave, y los datos importantes para la relación entre las entidades. Indica también la cardinalidad.

4 **Representa el ejercicio anterior en el modelo O.O. Indica las posibles operaciones. Considera las asociaciones navegables en ambos sentidos. Indica el nombre de la asociación, los roles, la multiplicidad y las generalizaciones.**

5 **Realiza el diagrama de datos en el modelo E-R, que represente este problema:**
A un taller de automóviles llegan clientes a comprar coches. De los coches nos interesa saber la marca, el modelo, el color y el número de bastidor.
Los coches pueden ser nuevos y de segunda mano. De los nuevos nos interesa saber las unidades que hay en el taller. De los viejos el año de fabricación, el número de averías y la matrícula.
Los mecánicos se encargan de poner a punto los coches usados del taller. Un mecánico pone a punto a varios coches usados.
Un cliente puede comprar varios coches; un coche puede ser comprado por varios clientes. De la compra nos interesa la fecha y el precio.
Define las entidades, los atributos, las relaciones, sus atributos si los hubiera y las cardinalidades.

6 **Representa el ejercicio anterior en el modelo O.O. Considera las asociaciones navegables en ambos sentidos. Indica el nombre de la asociación, los roles y la multiplicidad.**

7 **Realiza el diagrama de datos en el modelo E-R que represente el siguiente problema:**
Una agencia de viajes está formada por varias oficinas que se ocupan de atender a los posibles viajeros. Cada oficina oferta un gran número de viajes. Los viajes trabajan con una serie de destinos y una serie de procedencias. Cada viaje tiene un único destino y una única procedencia. Sin embargo, un destino puede ser objetivo de varios viajes y una procedencia ser punto de partida de varios viajes. Cada viaje tiene muchos viajeros.

8 **Realiza el diagrama de datos en el modelo E-R de este enunciado:**
Una entidad bancaria está formada por varias sucursales y cada sucursal tiene un gran número de cuentas que son propiedad de los clientes. Los datos saldo, debe y haber deben aparecer en cada una de las cuentas. Las cuentas son de dos tipos: cuenta de ahorro con el atributo específico tipo de interés. Y cuenta corriente, con el atributo específico cantidad de descubierto. Las cuentas o son corrientes o son de ahorro.
Un cliente puede tener varias cuentas. Una cuenta es sólo propiedad de un cliente.
Las cuentas realizan una serie de movimientos, en los que además de otros datos deben aparecer la cantidad implicada y la fecha. Existe una serie de tipos de movimientos reconocidos por el banco. Un movimiento pertenece a un tipo. Sin embargo, de un tipo de movimiento puede haber varios movimientos.