

# **DISEÑO DE BASES DE DATOS**

**Autor:**

**Dolores Cuadra, Elena Castro y Paloma Martínez.**

Coordinación pedagógica:

M<sup>a</sup> Cinta Cascales Angosto.

Edición:

Ana Isabel Arribas Partido.

Diseño de la portada:

Eduardo Sánchez Rubio



**Ministerio de Educación, Cultura y Deporte**

---

Dirección General de Educación y Formación Profesional

---

CNICE (Centro Nacional de Información y Comunicación Educativa)

---

I.S.B.N.: 84-369-3473-3

N.I.P.O.: 176-01-106-5

# ÍNDICE DE CONTENIDOS

<b>PARTE I : INTRODUCCIÓN.....</b>	<b>1</b>
1. UNA INTRODUCCIÓN A LAS BASES DE DATOS .....	3
2. METODOLOGÍAS DE DESARROLLO DE BASES DE DATOS. ....	12
<b>PARTE II : FASES DE DISEÑO .....</b>	<b>19</b>
1. FASE DE ANÁLISIS DE REQUISITOS: MODELO ENTIDAD INTERRELACIÓN (E/R).....	20
2. FASE DE DISEÑO LÓGICO ESTÁNDAR: MODELO RELACIONAL .....	53
3. TRANSFORMACIÓN DEL MODELO E/R AL MODELO RELACIONAL.....	102
4 FASE DE DISEÑO LÓGICO ESPECÍFICO: SGBD COMERCIAL.....	117
<b>GLOSARIO DE TÉRMINOS.....</b>	<b>164</b>

## **PARTE I : INTRODUCCIÓN**

1. Una introducción a las Bases de Datos
  - 1.1 La Base de Datos como un componente de los Sistemas de Información.
  - 1.2 Definición de Base de Datos
  - 1.3 Sistemas de Gestión de Bases de Datos (SGBD).
  - 1.4 Arquitectura una Bases de Batos a tres niveles.
  - 1.5 Lenguajes de los SGBD
  - 1.6 Otras herramientas de los SGBD
  - 1.7 Algunas arquitecturas de Sistemas de Bases de Datos
2. Metodologías de Desarrollo de Bases de Datos.
  - 2.1 Qué es una metodología y para qué sirve.
  - 2.2 Modelos de datos como instrumentos de diseño de Bases de Datos.
  - 2.3 Una Metodología de desarrollo de Bases de Datos

## **PARTE II : FASES DE DISEÑO**

1. Fase de Análisis de Requisitos: Modelo Entidad / Interrelación.
  - 1.1 Introducción
  - 1.2 Elementos básicos del Modelo E/R
  - 1.3 Extensiones del Modelo E/R
  - 1.4 Caso práctico
2. Fase de Diseño Lógico Estándar: Modelo Relacional.
  - 2.1 Introducción.
  - 2.2 Elementos básicos del Modelo Relacional

2.3 Estática del Modelo Relacional

2.4 Dinámica del modelo relacional

2.5 Caso práctico

2.6 Repaso de Consultas SQL

3. Transformación del modelo E/R al modelo relacional.

3.1 Introducción

3.2 Reglas básicas para la transformación del modelo E/R al modelo Relacional

3.3 Caso práctico.

4. Fase de Diseño Lógico Específico: SGBD comercial.

4.1 Introducción

Caso práctico en Access 97

## 1. Una introducción a las Bases de Datos

Con el fin de introducir al lector en la tecnología de las bases de datos aclararemos primeramente algunos conceptos necesarios que ayudarán a comprender la finalidad de las bases de datos y su papel dentro de los sistemas de información.

Las organizaciones o empresas emplean sistemas basados en ordenadores no sólo para el tratamiento administrativo de sus datos operacionales, es decir, los que son necesarios para llevar a cabo las tareas repetitivas (como son la contabilidad, gestión de stock, gestión de nóminas, etc.) sino también para tareas relativas a función de dirección dando soporte a la toma de decisiones (por ejemplo, elaboración de planes de descuento, nuevas líneas de negocio, etc.). Estos sistemas se denominan *Sistemas de Información* (SI) y se definen como un conjunto de personas, procedimientos y equipos diseñado, construido y gestionado para tratar la información de la organización de acuerdo a sus necesidades, De Miguel et. al. (1999).

### 1.1 La Base de Datos como un componente de los Sistemas de Información

Desglosando en detalle los componentes de un SI nos encontramos con cinco grandes componentes:

- El *Contenido*, es decir, los datos con su correspondiente descripción, almacenados en un soporte de ordenador (por ejemplo, en unos grandes almacenes se tendrían los datos de clientes, ventas, productos, etc.).
- *Equipo físico* (hardware) formado por la unidad central de proceso y los equipos periféricos (discos, terminales, impresoras, redes, ...).
- *Equipo lógico* (software) compuesto por los programas, documentación, lenguajes de programación, etc. que debe gestionar los datos (creación, consulta, recuperación y mantenimiento) así como controlar las comunicaciones y dar soporte a tratamientos específicos (por ejemplo, gestión de personal, facturación, etc.).
- El *Administrador*, encargado de asegurar la calidad de los datos almacenados y de permitir su uso correcto y permanente. El administrador o administradores debe controlar la disponibilidad, la confidencialidad y la integridad de los datos. La disponibilidad se refiere a que los datos deben estar accesibles en todo momento, es decir, que ante cualquier tipo de

catástrofe o fallo, se tengan los mecanismos adecuados de recuperación para que el sistema siga funcionando; la confidencialidad se encarga de no desvelar datos a usuarios no autorizados y la integridad asegura que los datos no se falseen, es decir, que sean correctos, válidos y precisos.

- Un conjunto de *Usuarios* formado por las personas que acceden al sistema de información y que pueden ser de dos tipos: informáticos (analistas y programadores encargados de desarrollar las aplicaciones, bases de datos, etc.) y los usuarios finales con pocos conocimientos de informática que requieren consultas y actualizar los datos mediante interfaces adecuados a sus características.

El componente que nos interesa en este libro, la Base de Datos, se encuentra en el software del SI. A continuación, vamos a dar una definición más precisa de este componente que desglosaremos en todos sus aspectos. Posteriormente, estudiaremos la necesidad de las Bases de Datos en los SI actuales en contraposición a los sistemas de ficheros.

## 1.2 Definición de Base de Datos

*"Una Base de Datos (BD) es una colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos y su definición (estructura de la BD), única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación, comunes y bien determinados, facilitarán la seguridad del conjunto de los datos", De Miguel et. al. (1999).*

Veamos en qué consiste cada uno de los aspectos mencionados en esta definición de Base de Datos que no son más que distintas definiciones según distintas perspectivas.

La BD es un conjunto de datos relativos a una determinada parcela del mundo real (por ejemplo, una biblioteca, una empresa petroquímica, una universidad, etc.,) que se almacenan en un soporte informático no volátil (es decir, dispositivos de memoria secundaria como discos, cintas, etc. que hacen que los datos no desaparezcan "cuando no se están usando").

Además, no debe existir redundancia, es decir, no deben existir duplicidades perjudiciales ni innecesarias (a ser posible un determinado tipo de dato, por ejemplo, los datos de un cliente de una empresa, sólo deben aparecer en un sitio en la BD). En ocasiones, es necesaria cierta

redundancia (a nivel de almacenamiento físico<sup>1</sup>) que mejora la eficiencia de la BD, por ejemplo, ante determinados tipos de consultas de datos. Sin embargo, esta redundancia siempre debe ser controlada por el sistema para que no se produzcan inconsistencias; piense el lector qué sucedería si los datos de los clientes de una empresa se repiten en varias partes de la BD y no se controlara: puede ocurrir que si un cliente cambia de dirección postal y sólo se actualiza esta información en uno de los sitios, entonces la BD quedaría en estado inconsistente

(el cliente aparece con datos distintos en distintas partes de la BD). Posteriormente, se volverá al tema de la redundancia cuando se estudien los modelos de datos como herramientas de diseño de Bases de Datos.

Por otro lado, las BD han de atender a múltiples usuarios de la organización (informáticos que desarrollan programas de acceso a la BD, administrativos usuarios de las aplicaciones, etc.) así como a distintas aplicaciones (por ejemplo, aplicaciones de contabilidad, de facturación, etc., todas ellas accediendo a los datos contenidos en la BD de la empresa).

Esta visión unificada de los datos se contrapone con los sistemas tradicionales de ficheros. Aunque no entraremos aquí en detalles de en qué consisten los sistemas de ficheros que se utilizaban con anterioridad a la aparición de las BD si indicaremos algunos aspectos relevantes que los diferencian de las BD; estos aspectos son:

1. Independencia de datos y procesos: es el objetivo fundamental de las BD, mantener separados los datos de los tratamientos que los utilizan. En los sistemas basados en ficheros cada fichero se diseñaba para responder a las necesidades de una aplicación determinada y apenas podían utilizarse por otra aplicación. En las BD, los datos se encuentran en un único almacén y son accedidos por todas las aplicaciones.
2. Descripción de los datos junto con los datos: la definición y descripción<sup>2</sup> del conjunto de datos contenidos en la BD deben ser únicas y estar integradas con los mismos datos. Posteriormente se verá que los datos están interrelacionados y estructurados de acuerdo a un modelo capaz de recoger el máximo contenido semántico. En los sistemas de ficheros, los datos se encuentran en distintos ficheros, diseñados *ad-hoc* para cada tipo de aplicación, y la descripción de los datos se encuentra junto con los programas de la aplicación. Recuérdese,

---

<sup>1</sup> En secciones posteriores estudiaremos que se habla de tres niveles (conceptual, lógico y físico) en la arquitectura de una BD. Por ahora, nos basta saber que el nivel físico concierne a cómo se almacenan los datos en los ficheros de la BD.

<sup>2</sup> En las siguientes secciones se verá que la descripción de los datos es lo que se denomina estructura o *esquema de la BD*.



por ejemplo, un programa en Pascal en el que al principio del código se define de qué tipo son las variables y las estructuras de datos que se van a manejar en el programa que accede a los ficheros.

3. Procesos de actualización y recuperación bien establecidos: la recuperación y actualización de los datos se realiza de acuerdo a procesos bien determinados que se incluyen en el Sistema de Gestión de Bases de Datos<sup>3</sup> (SGBD). El SGBD también proporcionará los instrumentos necesarios para el mantenimiento de la seguridad (confidencialidad, disponibilidad e integridad) del conjunto de datos.

### 1.3 Sistemas de Gestión de Bases de Datos

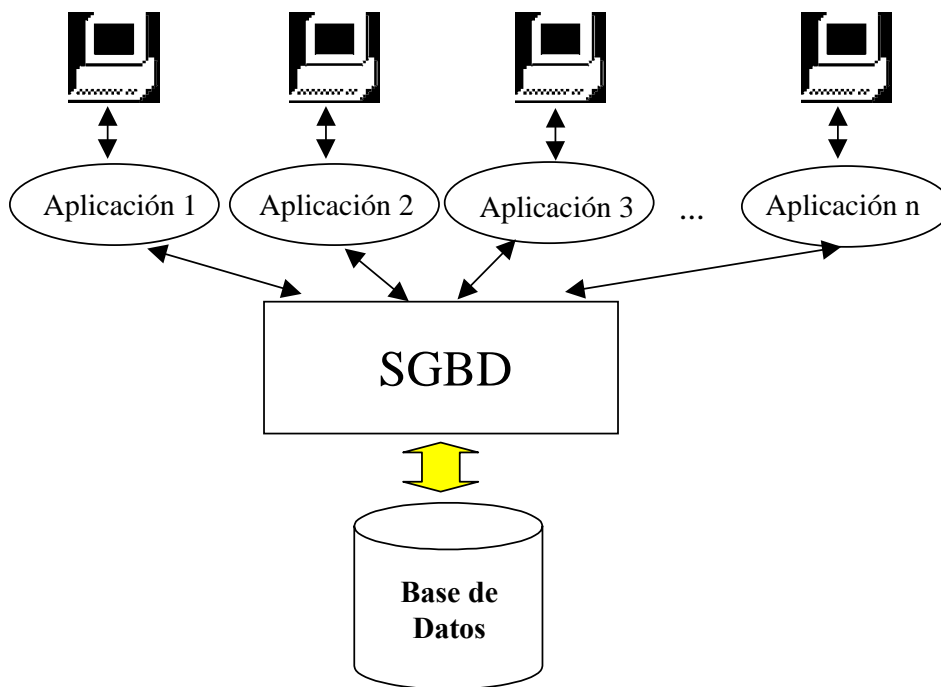
En la sección anterior se ha estudiado qué es una BD y se ha mencionado alguna de las funcionalidades de un SGBD. En esta sección nos centraremos en describir en profundidad qué es un SGBD y cuál es la funcionalidad que debe proporcionar.

"Un Sistema de Gestión de Bases de Datos (SGBD<sup>4</sup>) es un conjunto coordinado de programas, procedimientos, lenguajes, herramientas, etc., que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o administradores de una BD, los medios necesarios para describir y manipular los datos integrados en la BD, manteniendo su integridad, confidencialidad y disponibilidad", De Miguel et al. (1999).

---

<sup>3</sup> En el siguiente apartado se estudiará que es un SGBD. Por ahora, sólo decir que el conjunto de programas y herramientas que nos permite crear, actualizar, manipular y mantener una BD.

<sup>4</sup> En inglés, *Database Management System* (DBMS).



**Figura 1: Sistema de Bases de Datos**

Se denomina Sistema de Bases de Datos a la unión de una BD, un SGBD más las aplicaciones que acceden a la BD. La Figura 1.1 muestra la arquitectura de un sistema de BD; en ella se observa que el SGBD hace de interfaz entre los usuarios que acceden a la BD mediante las aplicaciones y la Base de Datos que contiene toda la información.

Ya hemos mencionado en la sección anterior que existen distintos tipos de usuarios con necesidades diferentes. Para poder dar soporte a estos usuarios el SGBD debe proporcionar una serie de funciones que se describen a continuación:

1. **Función de definición:** permite a los diseñadores de la BD describir los elementos de datos, su estructura y las relaciones que existen entre ellos; como se estudiará en la sección 1.5, el SGBD proporciona un lenguaje para la definición las tablas, los atributos que la componen, las restricciones semánticas así como las características de tipo físico o almacenamiento.
2. **Función de manipulación:** permite a los usuarios de la BD añadir, suprimir o modificar los datos de la misma siempre y cuando se respeten los aspectos de seguridad que haya establecido el administrador de la BD.

3. **Función de control:** esta función aúna los interfaces que requieren los distintos tipos de usuarios para comunicarse con la BD así como las herramientas necesarias para el administrador para establecer los mecanismos de seguridad y mantenimiento de la BD.

Para que el SGBD pueda llevar a cabo estas funciones se necesita un lenguaje que permita especificar lo que cada tipo de usuario necesita en su comunicación con la BD. En las BD relacionales se emplea el SQL (Standard Query Language) sobre el que hablaremos en la sección 1.5.

#### 1.4 Arquitectura de BD a tres niveles

A continuación, vamos a definir cuál es la arquitectura de una BD según la visión que de ella tienen los distintos tipos de usuario. En una BD se identifican tres capas de estructuración según tres niveles de abstracción. Así, se distingue un nivel externo, un nivel lógico y un nivel físico.

- El **nivel externo** se corresponde con la visión de la BD que cada usuario tiene en particular. Esto significa que no todos los usuarios necesitar conocer la BD completa sino que únicamente necesitan una vista parcial de ella (la que le permita llevar a cabo su trabajo); por ejemplo, un administrativo que trabaje elaborando las nóminas de los empleados de una empresa no necesita conocer los datos relativos a las ventas de productos de esa empresa.
- El **nivel lógico** se corresponde con la visión total de la empresa; esta vista global se interpone entre el nivel externo y el nivel físico siendo independiente tanto del equipo como de cada usuario en particular; por ejemplo, el administrador de la BD si necesita tener una vista completa de la BD de la empresa para llevar a cabo su trabajo.
- El **nivel físico** se corresponde con la vista del soporte físico informático en cuanto a que se refiere a la forma en que se organizan los datos en el almacenamiento físico (índices o punteros, longitud de los campos, caminos de acceso a los datos, particionamientos de memoria, etc.).

La gestión de estos tres niveles debe estar soportada en cualquier SGBD.

#### 1.5 Lenguajes de un SGBD

De acuerdo a las funciones a las que debe dar soporte un SGBD estudiadas en la sección 1.3 y a los distintos niveles de estructuración de una BD vistos en la sección 1.4, los SGBD deben

proporcionar un lenguaje para que los distintos tipos de usuario puedan comunicarse con la BD. Así, en los SGBD relacionales se tiene el lenguaje SQL que de acuerdo a su función se descompone en:

- (a) Lenguaje de Definición de Datos (LDD): utilizado para definir la estructura lógica de la BD (nivel lógico), la estructuras externas requeridas para el desarrollo de las diferentes aplicaciones (nivel externo) así como la estructura interna (nivel físico).
- (b) Lenguaje de Manipulación de Datos (LMD): una vez se ha descrito la BD, ésta ya está preparada para cargar los datos en las estructuras definidas y para su utilización. Así, el LMD permite añadir, suprimir, modificar y buscar datos en la BD. Es el SGBD el que se encarga de acceder al correspondiente soporte físico para localizar los datos con los que se harán las operaciones especificadas.
- (c) Lenguaje de Control: el administrador de la BD utiliza este lenguaje para especificar los aspectos de seguridad física (copias de seguridad, re arranque de la BD en caso de caída, etc.) así como de protección frente a accesos no permitidos (autorizaciones y contraseñas, perfiles de usuarios, etc.). El lenguaje de control también se requiere para definir los interfaces que necesitan los distintos usuarios para comunicarse con la BD.

## 1.6 Otras herramientas de los SGBD

Aparte de los lenguajes vistos en la sección 1.6, los SGBD proporcionan otro tipo de herramientas de gran utilidad en el desarrollo de aplicaciones de Bases de Datos. Entre otras, existen:

- herramientas de ayuda al desarrollo (CASE<sup>5</sup>) en las fases de análisis, diseño e implementación de BD que generalmente incluyen diagramadores para esquemas conceptuales y lógicos de bases de datos, generadores de código SQL, etc.
- generadores de informes y pantallas que facilitan la presentación de los datos recuperados de la BD.
- generadores de aplicaciones basados en lenguajes de cuarta generación (4GL) que permiten a los usuarios desarrollar aplicaciones sin tener que programar en lenguajes convencionales

---

<sup>5</sup> Computer Aided Software Engineering

- facilidades de usuario para facilitar la consulta de los datos (menús, interfaces gráficas, etc.).

### 1.7 Algunas arquitecturas de Sistemas de Bases de Datos

En este apartado revisaremos brevemente algunos conceptos relacionados con las distintas arquitecturas de Sistemas de BD. En una arquitectura se reflejan aspectos como la conexión en red, el paralelismo y la distribución:

- Red: permite que algunas tareas se ejecuten en un sistema servidor y que otras se ejecuten en los clientes (son lo que se denominan sistemas de BD cliente-servidor).
- Paralelismo: acelerar la ejecución de tareas (transacciones, etc.) de acuerdo al sistema informático subyacente (sistemas de BD paralelos).
- Distribución: Datos situados donde se han generado o donde son más necesarios pero accesibles desde todos los sitios (sistemas de BD distribuidos).

Según estos aspectos distinguimos sistemas centralizados, sistemas cliente-servidor, sistemas paralelos y sistemas distribuidos.

En los sistemas *centralizados* existe un único sistema informático sin interacción con otros ordenadores. En estos sistemas podemos diferenciar entre:

- (a) Sistema monousuario formado por un ordenador personal o por una estación de trabajo con una única CPU y un sistema operativo monousuario (no permite que varios usuarios puedan acceder simultáneamente a la BD)
- (b) Sistema multiusuario formado por varias CPU y con sistema operativo multiusuario con terminales conectados al sistema servidor; estos terminales no poseen ninguna funcionalidad propia aparte de la de visualizar el resultado de los procesos que se ejecutan en el servidor.

En los sistemas *cliente-servidor* existe un reparto de funcionalidades, es decir, los terminales se sustituyen por ordenadores personales que gestionan el interfaz de usuario SQL, interfaz de formularios, diseñadores de informes e interfaz gráfica. Los sistemas servidores satisfacen las peticiones generadas por los sistemas clientes.

Se distinguen dos tipos de servidores:

- (a) Servidores de transacciones (servidores de consultas) con un interfaz mediante el que los clientes envían peticiones para realizar una acción que el servidor ejecutará y cuyos resultados se devuelven al cliente.
- (b) Servidores de datos (el servidor envía los datos a las máquinas clientes en las que se realiza el procesamiento enviando después los datos de vuelta).

Respecto a los sistemas *paralelos* representan una solución al manejo de BD muy grandes o con un gran volumen de transacciones por segundo. El objetivo es realizar operaciones simultáneamente mediante el uso de varios procesadores y varios discos en paralelo. Los modelos de arquitecturas para máquinas paralelas son:

- (a) Memoria compartida: Todos los procesadores comparten una memoria común.
- (b) Disco compartido: Los procesadores comparten un disco común (cada procesador con su memoria)
- (c) Sin compartimiento: No hay compartición ni de disco ni de memoria
- (d) Jerárquico: modelo híbrido de los anteriores.

Por último, en los sistemas *distribuidos* la BD se almacena en varios ordenadores que no comparten ni memoria ni discos pero que se comunican mediante redes de alta velocidad o líneas telefónicas. Estos ordenadores se encuentran en varios lugares geográficos distintos. En un sistema distribuido se dan dos tipos de transacciones:

1. Transacciones locales: Acceso a datos del ordenador en el que se inició la transacción
2. Transacciones globales: Acceso a datos de un ordenador distinto o acceso a datos de varios ordenadores distintos.

Las ventajas que proporcionan los sistemas distribuidos frente a los sistemas centralizados son la compartición de datos (acceso a datos en distintos sitios), por ejemplo, dos sucursales bancarias pueden compartir datos entre sí; la autonomía en cuanto a que cada administrador controla su BD y, por último, la disponibilidad de los datos pues si un ordenador falla, están los demás para poder seguir trabajando, en particular, si hay duplicación de datos.

## 2. Metodologías de Desarrollo de Bases de Datos.

Una vez estudiados brevemente los principales aspectos relacionados con la tecnología de las Bases de Datos en cuanto a terminología y conceptos básicos (qué es una BD, Sistemas de Gestión de Bases de Datos, niveles de abstracción, arquitecturas, etc.) en este capítulo se expondrán aquellos conceptos relacionados con el diseño de BD que es el tema principal de este libro. Para ello, definiremos en primer lugar qué es una metodología y para qué sirve; la sección 2.2 está dedicada a los modelos de datos como instrumento necesario para diseñar BD y, por último, la sección 2.3 expone la metodología de desarrollo de BD que se seguirá en los siguientes capítulos de este libro.

### 2.1. Qué es una metodología y para qué sirve.

*"Una metodología es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de un producto software"*, Amescua et al. (1995); en el caso que nos ocupa en este libro, el producto software es una Base de Datos<sup>6</sup>. Una metodología nos indica las actividades a seguir en el desarrollo de principio a fin de la Base de Datos, qué es lo que hay que realizar en cada actividad indicando qué se necesita como entrada, qué se produce como salida e incluso quién está involucrado. Por lo tanto, es como un libro de recetas de cocina en el que se va indicando paso a paso todas las actividades a realizar para lograr la Base de Datos deseada.

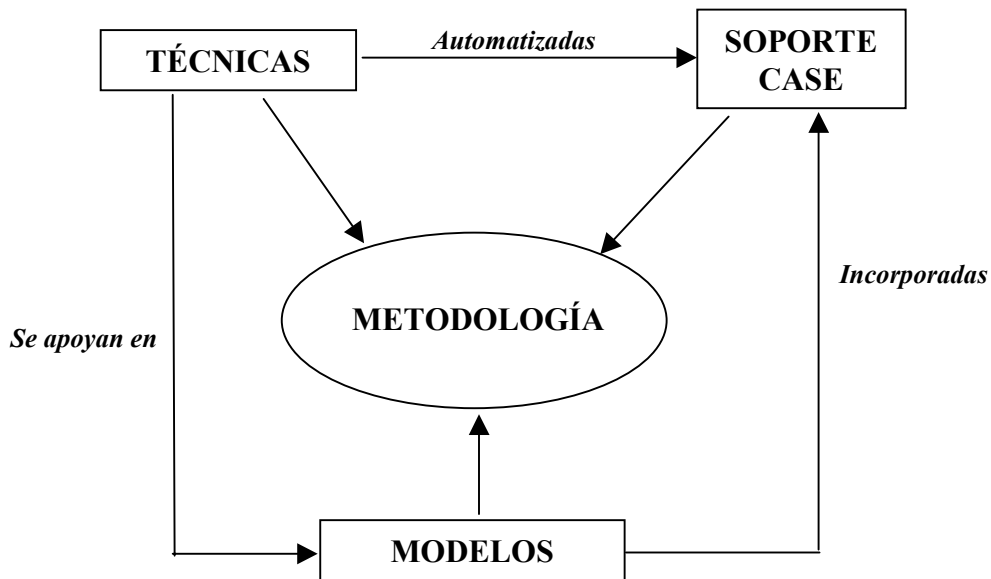
Como muestra la Figura 1.2, una metodología se apoya en los siguientes elementos: técnicas, modelos y soporte CASE. Veamos en qué consiste cada uno de estos elementos.

Las *técnicas* representan cómo llevar a cabo cada una de las actividades o pasos de los que consta la metodología, es decir, proporcionan procedimientos para llevar a cabo cada tarea; en ocasiones estas técnicas son procedimentales (secuencia perfectamente definida de los pasos a realizar en una tarea como en un algoritmo) y en otros casos son heurísticas (reglas, recomendaciones o sugerencias a seguir que en ningún caso establecen el proceso exacto de realización de una tarea; generalmente se utilizan en tareas con un alto componente creativo).

Los *modelos* son los instrumentos que empleamos para representar una determinada realidad (generalmente tienen una notación gráfica que facilita su comprensión y validación); se utilizan en las técnicas para soportar la actividad que llevan a cabo. En el siguiente apartado se estudiarán los modelos de datos involucrados en el diseño de una BD

---

<sup>6</sup> En adelante, nos referiremos siempre a metodologías para Bases de Datos



**Figura 1.2: Relación entre los componentes de una metodología**

Por último, las herramientas CASE permiten dar soporte automatizado a la aplicación de las técnicas de una metodología así como a los modelos que incorporan. Los entornos CASE no solo deben automatizar las técnicas aisladas correspondientes a una metodología sino también dar soporte a toda la metodología de desarrollo mediante la incorporación de un conductor metodológico que ayude al analista, diseñador o programador a desarrollar su labor en cada actividad definida en la metodología.

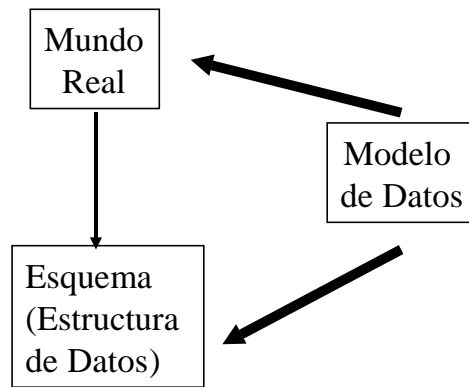
Todos estos conceptos se estudiarán particularizados para una metodología de BD en las siguientes secciones.

## 2.2. Modelos de datos como instrumentos de diseño de Bases de Datos.

El diseño de BD consiste en describir la estructura de la BD de forma que se represente fielmente la parcela del mundo real<sup>7</sup> que se quiere almacenar. Ello se realiza mediante un proceso de abstracción (que se denomina modelado) que se apoya en un *modelo* de datos. Un modelo de datos es el instrumento que se aplica a un UD para obtener una estructura de datos que se denomina *esquema* de la BD (Figura 1.3).

<sup>7</sup> En el ámbito de las Bases de Datos se denomina Universo del Discurso (UD)





**Figura 1.3: Aplicación de un Modelo de Datos**

Un modelo de datos proporciona un conjunto de conceptos, reglas y convenciones que nos permiten especificar y manipular los datos que queremos almacenar en la BD. Todo modelo de datos se compone de una parte estática y una parte dinámica como se explica a continuación.

**A. Estática:** Conjunto de estructuras (también denominados *constructores* del modelo) que permiten definir los datos y sus restricciones asociadas especificados según un Lenguaje de Definición de Datos (apartado 1.5). Esta parte estática consta de elementos permitidos y elementos no permitidos:

1. *Elementos permitidos*: son los objetos, asociaciones entre objetos, propiedades, etc. que proporciona el modelo para representar una determinada realidad (estos elementos varían de un modelo a otro según su riqueza semántica).
2. *Elementos no permitidos*: Son las restricciones que representan las limitaciones impuestas a la estructura del modelo o a los datos que invalidan ciertos ejemplares de la BD. Las restricciones son de dos tipos:
  - Restricciones inherentes: son las limitaciones impuestas a las estructuras del modelo (reglas impuestas para la utilización y combinación de los distintos constructores del modelo).
  - Restricciones semánticas (o de usuario): son las restricciones que se deducen de los supuestos semánticos explícitos o implícitos o derivados de nuestro conocimiento del mundo real que se quiere reflejar en la BD (por ejemplo, "todo empleado debe pertenecer a un departamento", "el sueldo de un determinado empleado siempre será inferior al sueldo de su jefe", etc.).

**B. Dinámica:** Formada por un conjunto de operadores que permiten manipular los datos y que están reflejados en el Lenguaje de Manipulación de Datos (apartado 1.5). Como se estudiará después, esta parte dinámica no tiene sentido para todos los modelos de datos.

Además, cada modelo de datos tiene una representación gráfica que suele ser en forma de grafos o tablas.

A lo largo del desarrollo de una BD se utilizan varios modelos de datos que nos permiten representar la realidad según las distintas fases de una metodología y según distintos niveles de abstracción. Aunque los siguientes capítulos se dedicarán al estudio pormenorizado de cada uno de los modelos, la Tabla 1 muestra la parte estática y dinámica del Modelo Entidad/Interpelación (E/R) para modelado conceptual y el Modelo Relacional para diseño lógico de BD.

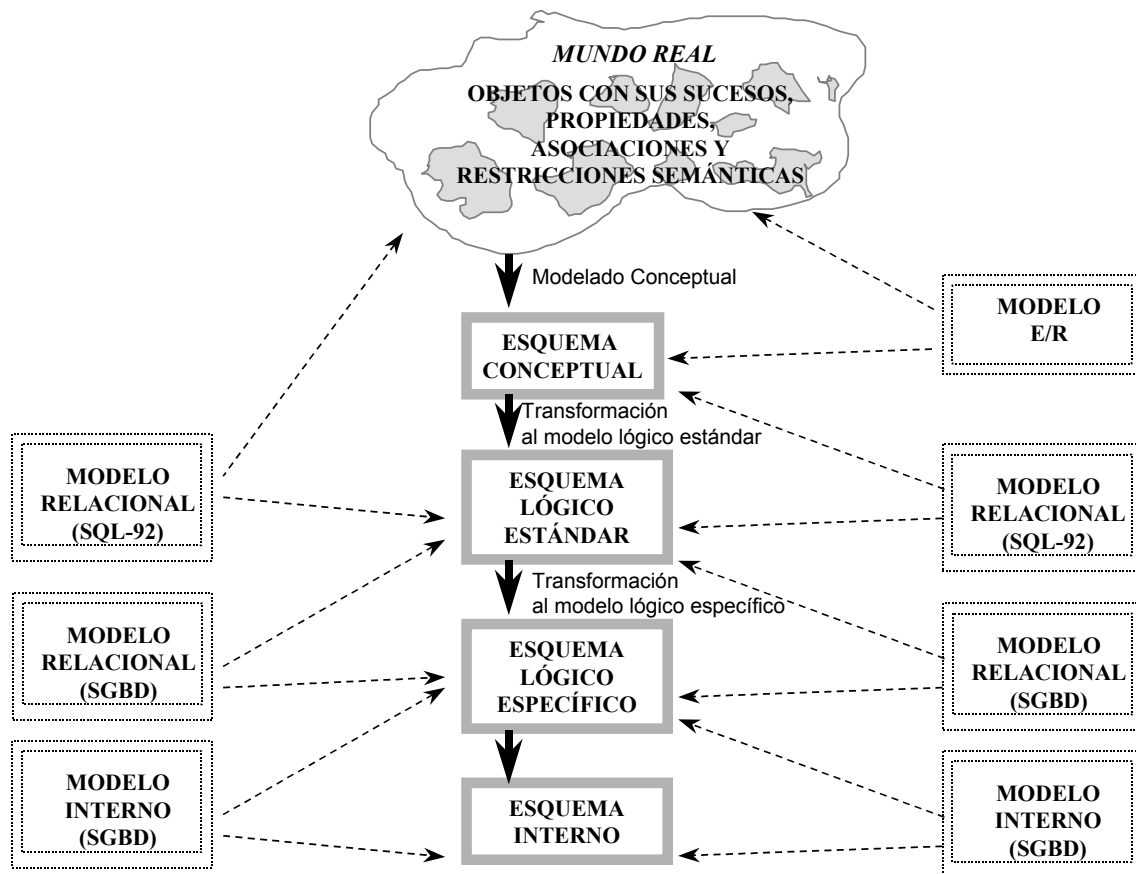
	Modelo E/R	Modelo Relacional
<b>Elementos permitidos</b>	- Entidades, atributos, interrelaciones, jerarquías, dominios	- Relación, atributos, dominios
<b>Restricciones inherentes</b>	- Obligatoriedad de Identificador Principal de Entidad. - No existen interrelaciones entre interrelaciones.	- Obligatoriedad de clave primaria. - Orden de tuplas y atributos no es significativo. - No existen grupos repetitivos. - Regla de integridad de entidad.
<b>Restricciones semánticas</b>	- <i>Restricciones sobre atributos:</i> identificador principal, identificador alternativo, simples/compuestos, univaluados/multivaluados, obligatorios/opcionales y atributos derivados. - <i>Restricciones sobre interrelaciones:</i> restricciones de cardinalidad, tipo de correspondencia, dependencia en existencia, dependencia en identificación, interrelaciones exclusivas - <i>Restricciones sobre jerarquías:</i> exclusividad/solapamiento, totalidad/parcialidad	- Definición de clave primaria. - Restricción de unicidad. - Restricción de obligatoriedad. - Integridad referencial (clave ajena). - Restricciones de verificación.
<b>Dinámica</b>	- No es de interés puesto que es un modelo abstracto no implementable	- Lenguaje SQL-92.
<b>Forma de representación</b>	- Grafos	- Grafos y tablas.

Tabla 1: Resumen de la parte estática y dinámica de los modelos E/R y Relacional.

### 2.3. Una Metodología de desarrollo de Bases de Datos

Aunque existen distintas metodologías para el desarrollo de BD, Elsmari y Navathe (1997), en este libro se seguirá la propuesta en De Miguel et al. (1999) que cubre las fases de diseño conceptual, diseño lógico estándar y diseño lógico específico. La figura 4 muestra las fases de esta metodología con los modelos que se

aplican en cada una de ellas. Aunque el uso de esquemas conceptuales facilita el diseño de BD no siempre la fase de modelado conceptual se lleva a cabo. La parte derecha de la Figura 1.4 muestra la metodología completa de diseño de BD mientras que la parte izquierda muestra el diseño de BD partiendo del diseño relacional de una BD.



**Figura 1.4: Metodología de Diseño de Bases de Datos, De Miguel et. al. (1999)**

Veamos a continuación en qué consiste cada una de estas fases:

**1. Modelado Conceptual**: consiste en la representación del UD (parte del mundo real que se quiere almacenar en la BD) en esquemas conceptuales E/R. Mediante los constructores del modelo E/R se recoge toda la semántica que puede obtenerse mediante la observación del UD o bien a partir de unas especificaciones textuales (esquemas descriptivos) que describan la información que debe contener la BD. En este libro se construirán esquemas conceptuales de BD a partir de esquemas descriptivos.

Esta primera fase de análisis tiene como objetivo poder validar con el usuario (persona o conjunto de personas que nos encargan una BD para cubrir sus necesidades de negocio) la información que contendrá la BD. Por ello, los esquemas E/R son los de mayor nivel de

abstracción (capacidad para ocultar los detalles y fijarse en lo esencial), con constructores muy naturales (estructuras muy cercanas al usuario y fácilmente comprensibles por personas no informáticas). Nótese que los esquemas conceptuales no son directamente implementables en un ordenador; por ello, no tienen ninguna connotación física y pueden traducirse a cualquier modelo lógico<sup>8</sup>. El esquema E/R viene a ser para una BD como los planos de un arquitecto son imprescindibles para una casa: algo necesario a priori en la construcción de una BD. Sin estos *planos* es imposible conocer cuáles son los requisitos que deberán contemplarse en la BD.

La construcción de esquemas E/R es una labor creativa que se realiza en sucesivos pasos de refinamiento; consecuentemente, no todos los analistas obtendrán el mismo esquema E/R cuando modelan una determinada realidad pues dependerá de la labor intelectual que lleve a cabo cada uno en su visión del UD. Sin embargo, si es posible seguir una serie de heurísticas o recomendaciones de gran utilidad cuando se modela una BD. En el capítulo dedicado al modelo E/R se estudiarán estas heurísticas en detalle.

**2. Transformación de esquemas conceptuales E/R a esquemas relacionales:** Una vez se ha validado con el usuario el esquema E/R correspondiente a la BD ya es posible realizar la transformación a un esquema lógico, en nuestro caso, a un esquema relacional. Para este paso si que existe un procedimiento exhaustivo a seguir con el fin de traducir todos los constructores del modelo E/R a constructores del modelo Relacional. En un primer paso, se hace una transformación al modelo relacional estándar (SQL-92). El modelo relacional estándar no es directamente implementable en un SGBD relacional pues cada SGBD implementa de manera libre un subconjunto de este estándar. Es en la fase de transformación a un modelo lógico específico (es decir, el propio de cada SGBD comercial<sup>9</sup>) cuando ya se habla de BD directamente trasladables a un producto comercial.

**3. Diseño físico:** En esta fase se tienen en cuenta aspectos relacionados con la carga de la BD, la optimización de consultas y otros aspectos relacionados con la eficiencia en el almacenamiento y funcionamiento de la BD y que son realizadas por el administrador de la BD a través de las utilidades que proporciona el SGBD que se vaya a utilizar.

---

<sup>8</sup> Aunque en este libro sólo se estudia el modelo relacional como modelo para Diseño Lógico de BD existen otros modelos (jerárquico, en red, etc.).

<sup>9</sup> Por ejemplo, Oracle, Access, SQL-Server, Informix, etc. son SGBD comerciales que no implementan de igual manera el modelo relacional.

Como se observa en la parte izquierda de la Figura 1.4, también es posible realizar el diseño de la BD directamente en el modelo relacional sin llevar a cabo previamente la fase de modelado conceptual. En este caso, el diseñador plasmará directamente en un esquema relacional la semántica del mundo real que debe quedar recogida en la BD.

**Autoevaluación: - PARTE I: INTRODUCCIÓN**

## **PARTE II : FASES DE DISEÑO**

2. Fase de Análisis de Requisitos: Modelo Entidad / Interrelación.
  - 2.1 Introducción
  - 2.2 Elementos básicos del Modelo E/R
  - 2.3 Extensiones del Modelo E/R
  - 2.4 Caso práctico
5. Fase de Diseño Lógico Estándar: Modelo Relacional.
  - 2.7 Introducción.
  - 2.8 Elementos básicos del Modelo Relacional
  - 2.9 Estática del Modelo Relacional
  - 2.10 Dinámica del modelo relacional
  - 2.11 Caso práctico
  - 2.12 Repaso de Consultas SQL
6. Transformación del modelo E/R al modelo relacional.
  - 3.4 Introducción
  - 3.5 Reglas básicas para la transformación del modelo E/R al modelo Relacional
  - 3.6 Caso práctico.
7. Fase de Diseño Lógico Específico: SGBD comercial.
  - 4.2 Introducción
  - 4.3 Caso práctico en Access 97

## 1. Fase de Análisis de Requisitos: Modelo Entidad Interrelación (E/R)

### 1.1.Introducción

Los datos constituyen en la actualidad el arma más poderosa de cualquier organización o empresa. Una buena gestión de los datos puede influir de manera más que notable en los beneficios de cualquier organización. Pongámonos en el caso de una entidad bancaria y pensemos en los miles de clientes con cuyos datos se realizan operaciones diarias; la mala utilización de los mismos puede traer consigo pérdidas enormes para la empresa. En ocasiones esta mala utilización puede ser debida a la falta de formación de los empleados, pero muchas veces es ocasionada por un mal diseño del sistema de información o base de datos que gestiona los datos.

Hoy en día todas las empresas cuentan con herramientas informáticas de creación de bases de datos; entonces, ¿por qué se producen fallos?. La respuesta no está en las herramientas en sí, sino, y reincidiendo en el tema, en cómo se diseña la base de datos. Cada herramienta dispone de sus propios utensilios de diseño, pero todos ellos se basan en los mismos conceptos teóricos, conceptos que si se desconocen no pueden ser aplicados.

Por lo dicho anteriormente parece, si no completamente necesario, sí al menos muy conveniente, la utilización de un modelo de datos que permita diseñar bases de datos a nivel conceptual (y por tanto muy cercana al usuario) y por supuesto la formación de personal cualificado en este campo.

El modelo Entidad/Interrelación (E/R) es un modelo conceptual que ha demostrado ser muy válido para cumplir con este objetivo, pues está a un nivel de abstracción lo suficientemente elevado como para poder diseñar cualquier base de datos con independencia de la máquina en la que se implemente. Además, en la actualidad disponemos en el mercado de una amplia gama de herramientas que automatizan en gran parte las tareas del diseño<sup>10</sup> y que toman como base este modelo de datos.

El modelo E/R fue propuesto por Peter Chen en 1976. Desde entonces muchos autores se han interesado por él, estudiándolo y ampliándolo, consiguiendo así diversas variantes del modelo (distintas formas de representación de los objetos), pero todas ellas parten del mismo concepto:

---

<sup>10</sup> Herramientas CASE (Computer Aided Software Engineering).

el conocimiento del *mundo real* que se desea representar a través de un análisis de los requisitos o especificaciones del problema.

En la realización del esquema o diseño conceptual de cualquier base de datos es fundamental el conocimiento del problema a modelar y es en este conocimiento donde representan un papel primordial los usuarios finales del sistema, pues es en esta primera etapa de modelización en la que el diseñador de la base de datos debe hacer tantas entrevistas como sean necesarias con los usuarios para conseguir clarificar todas las especificaciones del problema. Una vez clarificados los objetivos y las necesidades se deberá pasar al diseño propiamente dicho de la base de datos.

El modelo E/R, como todos los modelos, consiste en un conjunto de conceptos, reglas y notaciones que permiten formalizar la semántica del mundo real que se pretende modelar (también denominada Universo del Discurso) en una representación gráfica o diagrama que denominamos esquema de la Base de Datos.

En este capítulo se explican cuáles son los elementos básicos que componen el modelo E/R y cómo se utilizan a la hora de diseñar una Base de Datos.

Aunque, como ya se comentó en la PARTE I de este documento, todo modelo de datos tiene una parte estática y otra dinámica; en este capítulo únicamente nos referiremos a la estática del modelo E/R, pues la parte dinámica carece de utilidad al no ser soportada por ningún SGBD actual.

## 1.2.Elementos básicos del Modelo E/R

Los elementos u objetos básicos del modelo E/R son cuatro: entidades, interrelaciones, atributos y dominios. A continuación se explican cada uno de ellos.

### Entidades

Las **entidades**, también llamadas tipos de entidad, representan conjuntos de elementos con existencia propia y que se caracterizan por las mismas propiedades. Generalmente son personas, cosas, lugares,..., es decir, conceptos sobre los que necesitamos guardar información y distinguibles de los demás objetos. Su representación gráfica se hace por medio de un rectángulo dentro del cual se escribe el nombre de la entidad en mayúsculas (generalmente un sustantivo).

Por ejemplo, si queremos diseñar una base de datos para gestionar todos los alumnos de los cursos Mentor, entre los tipos de entidad que deberíamos definir estarían ALUMNO y CURSO. El primero representaría el conjunto de todos los alumnos que se inscriben en los diferentes



cursos, el segundo recogería todos los cursos ofertados por el aula Mentor. Su representación gráfica sería (véase el esquema de la figura 2.1).

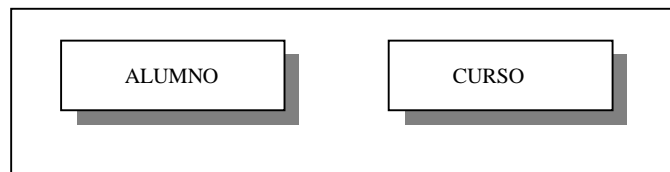


Figura 2.1: Dos ejemplos de entidades

## Atributos

Todo tipo de entidad tiene unas características o cualidades propias que queremos recoger dentro de nuestro diseño. El modelo E/R define estas cualidades como **atributos**, así por ejemplo el *nombre del alumno*, el *teléfono*, etc., describen propiedades de cada uno de los miembros que pertenecen al tipo de entidad ALUMNO. Estas propiedades no tienen existencia propia, es decir, sólo tienen sentido en el esquema de la Base de Datos en tanto en cuanto aparecen formando parte de una entidad o, como veremos más adelante, de otro de los elementos del modelo E/R, de una interrelación.

Supongamos que de cada alumno queremos la información referente a su D.N.I., Nombre, Dirección, Teléfono y Nacionalidad. En la figura 2.2, aparece cómo representamos los atributos en el modelo E/R.

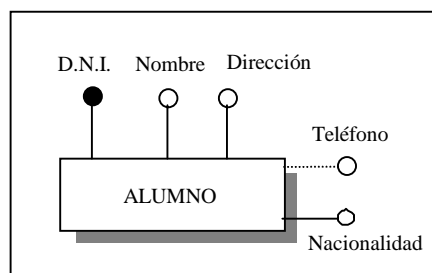


Figura 2.2: Un ejemplo de entidad con sus atributos

Los ejemplares, también denominados ejemplares o elementos, de un tipo de entidad se definen como los valores correspondientes a los atributos que hemos definido para ella.

Por ejemplo dos ejemplares del tipo de entidad ALUMNO serían:

- (*DNI*, 7515958), (*Nombre*, Juan), (*Dirección*, C/ Irún, nº 9 Madrid), (*Teléfono*, 91-675-65-65), (*Nacionalidad*, Española)

- (DNI, 7077777), (Nombre, Ana), (Dirección, C/ Bailén, nº 9, Madrid), (Teléfono, 91-678-98-99), (Nacionalidad, Española)

Por lo tanto los valores de los atributos constituyen una parte importante de los datos que almacenaremos posteriormente en la Base de Datos. Es importante destacar que un mismo concepto no tiene por qué representarse siempre de la misma forma (por ejemplo, como una entidad o como un atributo). Así, si estuviéramos modelando una Base de Datos para una tienda de ropa, probablemente tendríamos una entidad denominada PRENDA y uno de sus atributos podría ser *Color* (roja, negra, etc.). Sin embargo, si estuviéramos hablando de una Base de Datos para gestionar la información de un taller de vehículos dedicado a trabajos de chapa y pintura, el concepto de color puede tener tal importancia que pase a ser una entidad COLOR, pues tiene existencia propia y un conjunto de propiedades (*código de color, textura, tipo de mezcla*, etc.).

### Tipos de atributos

Como se puede observar en la figura 2.2 no todos los atributos se representan de la misma forma; ello significa que existen diversas formas de recoger restricciones semánticas sobre los atributos de una entidad o de una interrelación. En el ejemplo aparece el atributo D.N.I. con un círculo negro, este tipo de atributo se denomina **identificador principal (IP)** y lo que indica es que el atributo o propiedad DNI es único para cada ejemplar del tipo entidad ALUMNO.

Para poder distinguir una ejemplar de otra, dentro de un mismo tipo de entidad, el modelo E/R obliga a que cada vez que definimos un tipo de entidad se defina un atributo que identifique cada ejemplar, es decir, un IP. Por lo tanto en todos los tipos de entidad tiene que aparecer de forma obligatoria una característica que identifique de forma única cada uno de los ejemplares.

Esta es la representación que nos proporciona el modelo E/R para distinguir este tipo de atributo del resto de atributos que componen el tipo de entidad. En un tipo de entidad sólo puede aparecer un **identificador principal**, pero pueden existir distintos atributos que también

identifiquen los ejemplares de esta; este tipo de atributos se denominan **Identificadores Alternativos (IA)**.

Veamos un ejemplo, supongamos que queremos añadir para el tipo de entidad ALUMNO, la dirección de correo electrónico que este posee, sabiendo que es única para cada uno de los alumnos. El atributo *e-mail* sería un identificador alternativo y como vemos en la figura 2.3 se

representa con un círculo mitad negro mitad blanco, indicando que su valor es único para cada ejemplar del tipo de entidad ALUMNO.

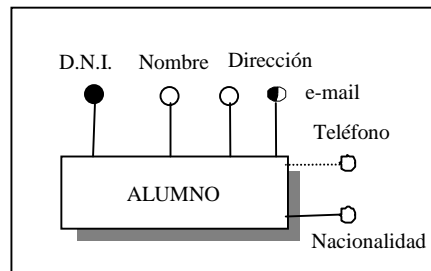


Figura 2.3: La entidad ALUMNO y sus atributos

En el ejemplo de la figura 2.3, el atributo *Teléfono* aparece representado con una línea de puntos lo que significa que estamos ante un **Atributo Opcional** que nos informa de que existen alumnos que puede que no tengan número de teléfono o que al fin y al cabo es un atributo cuyo valor no es demasiado importante y por eso no lo ponemos como obligatorio. Por tanto, cuando los valores de un atributo van a ser desconocidos o por alguna otra causa no van a tener valor se denominan **Atributos Opcionales**.

Supongamos que para el tipo de entidad CURSO es importante recoger las siguientes propiedades: nombre, libro de consulta y dirección Web. De estas tres características de CURSO elegiremos como identificador principal el *nombre*, ya que cada curso tiene un nombre distinto, la *dirección Web* sería un identificador alternativo porque toma valores únicos para cada curso y *libro de consulta* sería un atributo opcional ya que permitimos que haya cursos que no tengan o que desconozcamos su libro de referencia. La entidad CURSO con sus atributos queda representada en la figura 2.4.

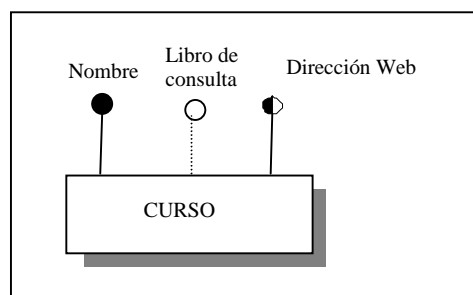


Figura 2.4: Un ejemplo de atributos IP, IA y opcional

Existen otras formas de recoger restricciones semánticas sobre los atributos que se estudiarán en el capítulo siguiente, en donde ampliaremos estos conceptos.

## Dominios

Supongamos que el atributo nacionalidad, véase figura 2.5, sólo puede tomar los valores “española” o “extranjera”. Para los conjuntos de valores sobre los que se definen los atributos utilizaremos un objeto del modelo E/R denominado **Dominio**. Un dominio se define por un nombre y un conjunto de valores. En nuestro ejemplo véase la definición del dominio Nacionalidad en la figura 2.6 resaltado en color azul.

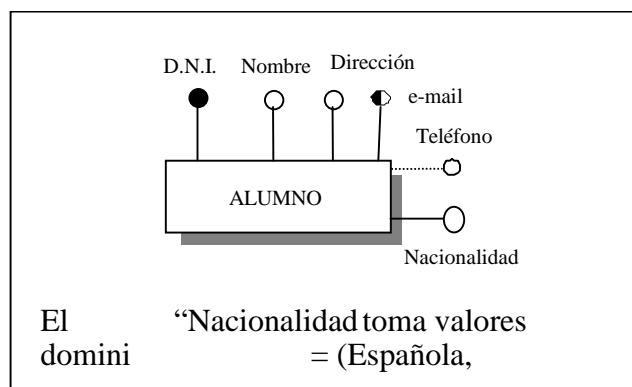


Figura 2.6: Dominio Nacionalidad y su representación textual

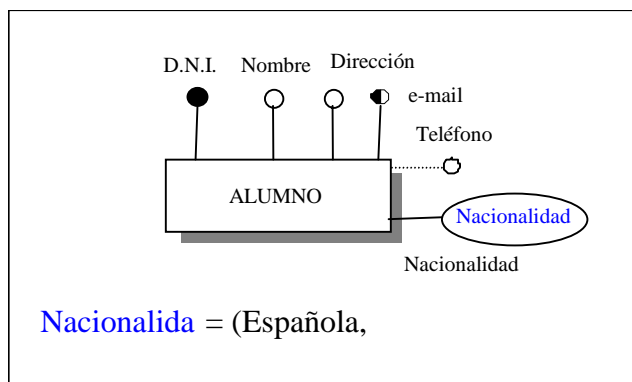


Figura 2.5: Dominio Nacionalidad

En general los dominios no se suelen representar en el modelo por problemas de espacio, pero para tener constancia de los valores que puede tomar un atributo se suele anotar después de la representación gráfica una representación textual.

## Autoevaluación: . - PARTE II.1: PREGUNTA DE REFUERZO.

### Interrelaciones

Las interrelaciones representan asociaciones del mundo real entre una o más entidades. Por ejemplo, en la figura 2.1 presentábamos los alumnos y los cursos del Mentor como entidades sin

ningún tipo de relación, pero para poder expresar que un alumno esta matriculado en distintos cursos y que en un curso se pueden matricular alumnos necesitamos una **Interrelación** que nos muestre la asociación existente entre ellos. Por lo tanto, vemos la necesidad de poder representar este concepto ya que aparece continuamente en el mundo real; algunos ejemplos son: “las sucursales de una entidad bancaria están relacionadas con sus clientes”, “las editoriales se relacionan con los libros que publican”, “los tutores de los cursos Mentor tienen asignados una serie de alumnos”, etc.

Gráficamente las interrelaciones se representan mediante un rombo unido a los tipos de entidad mediante líneas; dentro del rombo se escribe el nombre de la interrelación en minúsculas, que en general, suele coincidir con un verbo en infinitivo.

Volviendo al ejemplo anterior veamos como se representa la relación existente entre los alumnos que realizan cursos. Podríamos definir una interrelación **Realizar** entre ambas entidades, como muestra la figura 2.7.

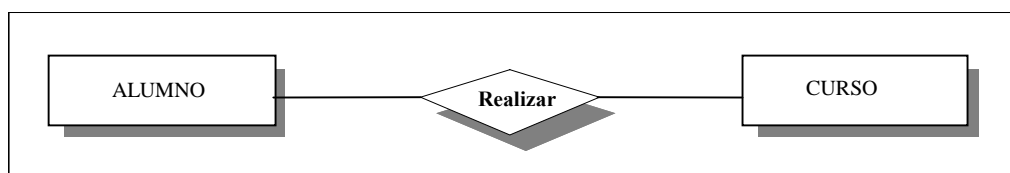


Figura 2.7: Ejemplo de Interrelación

No todas las relaciones o asociaciones son iguales, en general se dividen en relaciones que denominamos de *uno a muchos*, como por ejemplo la que presentamos a continuación: “una sucursal es únicamente de una entidad bancaria (*uno*) pero una entidad bancaria posee varias sucursales (*muchos*)”. También existen las relaciones *muchos a muchos*, como por ejemplo “un curso Mentor tiene asociados tutores (*muchos*) y los tutores pueden tutorar distintos cursos Mentor (*muchos*)”.

Para poder recoger estas características que nos distinguen unas relaciones de otras, que nos permite, además, recoger más información acerca del problema que estamos modelando, vamos a introducir las siguientes propiedades de una interrelación: grado, tipo de correspondencia y cardinalidad.

El grado de una interrelación es el número de entidades que intervienen en ella, debe ser como mínimo dos, es decir, el número de entidades que intervienen en una interrelación debe ser de al menos dos; existe un caso especial en el que sólo participa una entidad en la interrelación aunque

de dos formas distintas (es lo que se denomina interrelación reflexiva, como se verá después). En el ejemplo de la figura 2.7 se representa una interrelación binaria, denominada así por tratarse de una interrelación entre dos tipos de entidad. De la misma forma, cuando el grado es tres se habla de interrelaciones ternarias y, en general, de interrelaciones  $n$ -arias cuando el grado es  $n$ . El tipo de interrelaciones que aparece de forma habitual en el modelado de una Base de Datos es la interrelación binaria y a partir de ahora nos centraremos solo en este tipo de interrelaciones.

El Tipo de correspondencia de una interrelación binaria se define como el número máximo de ejemplares de un tipo de entidad que pueden estar asociados con un ejemplar del otro tipo de entidad. Su representación gráfica se hace por medio de un par  $X:Y$  colocado sobre el rombo de la interrelación, donde  $X$  e  $Y$  representan los ejemplares asociadas de los tipos de entidad en estudio<sup>11</sup>. En nuestro ejemplo, en principio, el número de cursos a los que un alumno puede optar es ilimitado y el de alumnos que realizan un curso también, por tanto la correspondencia sería  $N:M$  o muchos a muchos (Figura 2.8).

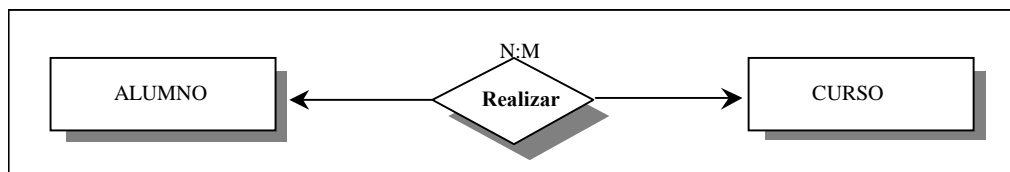


Figura 2.8: Tipo de Correspondencia  $N:M$

Si, por el contrario, en las especificaciones del problema se nos dijera que cada alumno solo puede matricularse de un curso, el tipo de correspondencia entre **ALUMNO** y **CURSO** cambiaría, sería  $1:N$  o uno a muchos, y se representaría de la manera que aparece en la figura 2.9.

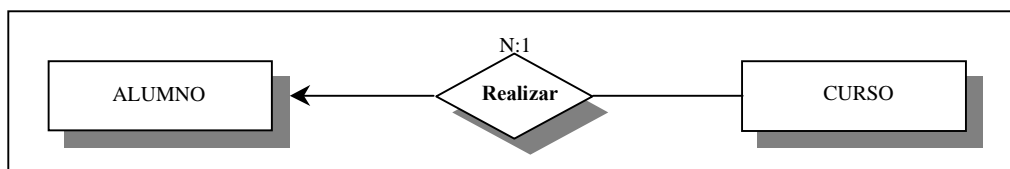


Figura 2.9: Tipo de Correspondencia  $N:1$

<sup>11</sup> Esta representación se puede generalizar en el caso de grado  $n$ , de la forma  $X:Y:Z:.....$

La cardinalidad de un tipo de entidad que interviene en una interrelación binaria se define como el número mínimo y el número máximo de ejemplares de un tipo que pueden relacionarse con un elemento de otro tipo de entidad. Para representar las cardinalidades utilizamos un par  $(x, y)$  situado sobre la línea que une el tipo de entidad con la interrelación, donde  $x$  indica el número mínimo e  $y$  el número máximo. Además, y cuando la cardinalidad máxima es  $n$ , se dibuja una punta de flecha hacia la entidad correspondiente (figura 2.8). En el ejemplo que nos ocupa y suponiendo que no se establece ninguna restricción adicional, el número mínimo de alumnos que pueden matricularse en un curso es uno (no tendría sentido un curso con 0 matriculados), y el número máximo  $n$  (número ilimitado), por tanto la cardinalidad del tipo de entidad ALUMNO es  $(1, n)$  como se muestra en la figura 2.10.

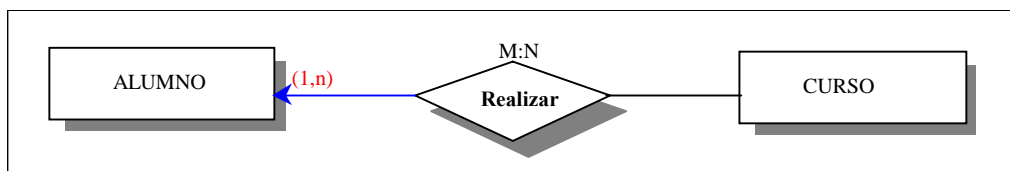


Figura 2.10: Ejemplo de cardinalidades

La interpretación de la interrelación Realizar sería “un curso Mentor es realizado como mínimo por un alumno y como máximo  $n$ ”. Si tuviéramos limitación en la matriculación de los alumnos en un curso, por ejemplo, los cursos Mentor como máximo admiten 40 alumnos, lo representaríamos de la siguiente forma:

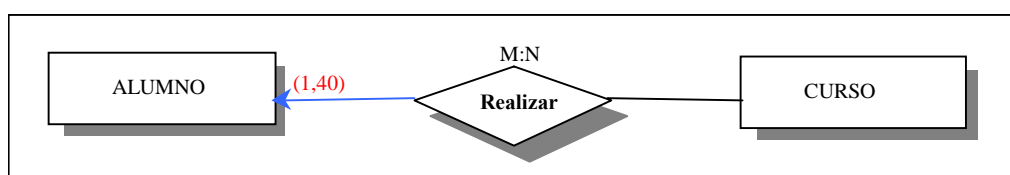


Figura 2.11: Ejemplo de cardinalidades

De la misma manera, el número mínimo de cursos que puede realizar un alumno es uno y el máximo  $n$ , es decir, la cardinalidad de CURSO es  $(1, n)$  y por tanto tendríamos que representar la punta de flecha hacia la entidad CURSO y encima de esta línea la cardinalidad como se muestra en la figura 2.12.

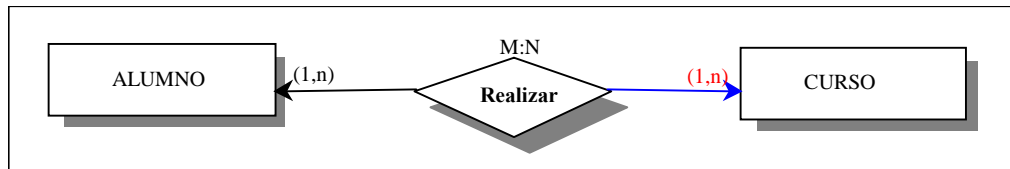


Figura 2.12: Cardinalidades de la interrelación Realizar

Las cardinalidades mínimas y máximas son, como se puede apreciar, una extensión del tipo de correspondencia y nos dan más información referente al tipo de interrelación que estamos representando.

Veamos otro ejemplo con la relación que existe entre los empleados de una empresa y el departamento en el que trabajan. Sabemos que un empleado trabaja en un departamento y que a cada departamento se le asigna al menos un empleado. De cada empleado se desea la siguiente información: un código de empleado (número que le identifica), DNI, nombre completo, dirección, teléfono y número de afiliación de la seguridad social. Para los departamentos necesitamos un nombre, único para cada uno de ellos, una localización y un número de teléfono. ¿Cuál sería su diseño en el modelo E/R?

Podemos detectar de forma clara que necesitamos dos entidades, EMPLEADO y DEPARTAMENTO, objetos que tienen existencia propia con determinadas características. Para la entidad EMPLEADO tenemos como identificador principal el código de empleado, figura 2.13; el DNI, que es único para cada empleado será un atributo alternativo ya que hemos elegido el código como identificador principal por especificaciones del problema (figura 2.14).

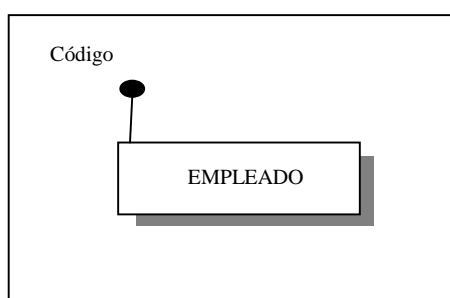


Figura 2.13: Entidad EMPLEADO y su IP

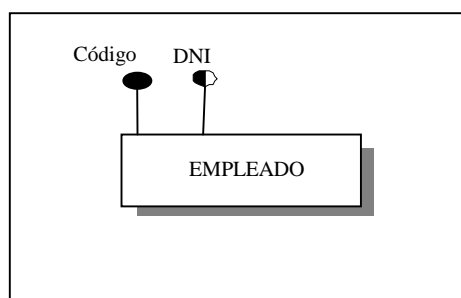


Figura 2.14: Entidad EMPLEADO con IP y IA

Los atributos *nombre* y *dirección* de EMPLEADO son obligatorios ya que dicha información la consideramos importante; por ejemplo, sin ellos no podríamos mandar la nómina o contactar por cualquier causa con los empleados de la empresa. El *teléfono* lo podemos considerar como un atributo opcional y, por último, el número de afiliación de la seguridad social (*NSS*) al tomar



valores únicos para cada empleado lo consideraremos un atributo alternativo. La entidad EMPLEADO con sus propiedades queda representada como se muestra en la figura 2.15.

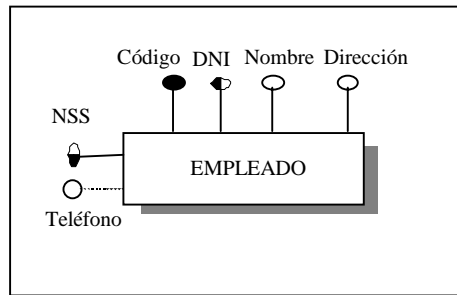


Figura 2.15: Atributos de la entidad EMPLEADO

Un razonamiento similar nos llevará a representar la entidad DEPARTAMENTO con las características que la definen como se muestra en la figura 2.16.

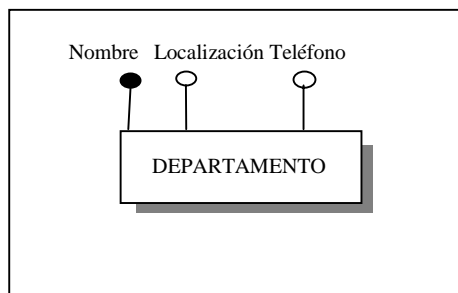


Figura 2.16: Atributos de la entidad DEPARTAMENTO

La interrelación que une las entidades representadas anteriormente, EMPLEADO y DEPARTAMENTO, es binaria ya que relaciona dos entidades; el tipo de correspondencia es 1:N o de uno a muchos, ya que un empleado está asignado a **un** departamento y a un departamento pertenecen **varios** empleados. Por último, se indican las cardinalidades que recogen explícitamente como se relacionan cada una de los ejemplares de las entidades participantes en dicha interrelación (figura 2.17).

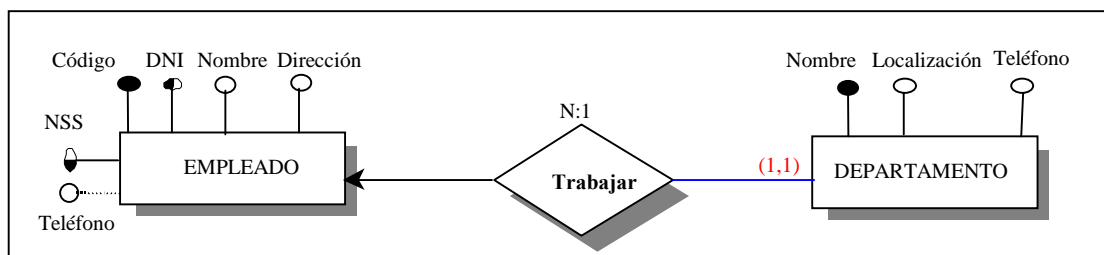


Figura 2.17: Interrelación Trabajar

La interpretación de la interrelación Trabaja sería la siguiente: “Un empleado trabaja como mínimo y como máximo en un solo departamento (1,1)” y tendríamos una línea continua entre el rombo de la interrelación y la entidad DEPARTAMENTO para reflejar este hecho.

Si interpretamos la figura 2.18 desde el tipo de entidad DEPARTAMENTO su lectura sería la siguiente: “En un departamento trabajan como mínimo un empleado y como máximo N”.

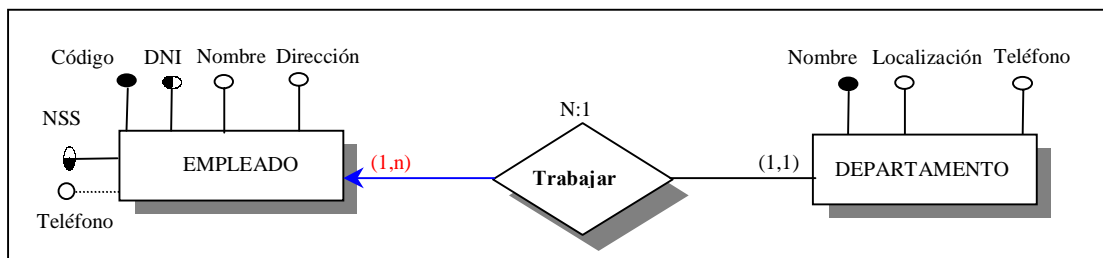


Figura 2.18: Interrelación Trabajar

### Atributos de una interrelación

Como ya se ha mencionado, los atributos no solo están referidos a los tipos de entidad. Las interrelaciones también pueden tener atributos propios, atributos cuyos valores tienen sentido únicamente en el caso de que se establezca la relación entre los tipos de entidad que las une, como pueden ser las fechas de comienzo y de finalización de un curso, que no tienen sentido si dicho curso no es realizado por al menos un alumno. Un ejemplo de estos atributos se muestra en la figura 2.19 en color verde.

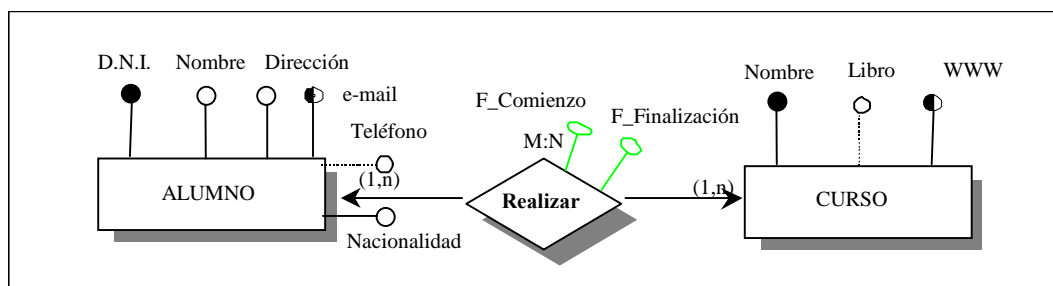


Figura 2.19: Interrelación con atributos

¿Cómo serían los ejemplares de la interrelación “Realizar”? Si pensamos en el mundo real, los valores nos vienen dados de la siguiente forma: Juan ha realizado el curso de “Iniciación a Internet” durante el periodo de 12-02-96 al 05-06-96. Algo parecido ocurre en el modelo E/R, los elementos que se encuentran en la interrelación “Realizar” son de la siguiente forma:

{(DNI, 7515458), (Nombre, Iniciación a Internet), (F\_Comienzo, 12-02-96), (F\_Finalización, 05-06-96)}.

{(DNI, 856593), (Nombre, Access Avanzado), (F\_Comienzo, 02-12-96), (F\_Finalización, 15-03-97)}.







Todos estos ejemplares se corresponden con los valores de los atributos identificadores de los tipos de entidad ALUMNO y CURSO que están relacionados, junto con los atributos propios de la interrelación. La interpretación o lectura que tienen estos elementos es la siguiente: el alumno con DNI 7515458 ha realizado el curso “Iniciación a Internet” durante el periodo del 12-02-96 al 05-06-96; el alumno con DNI 856593 ha realizado el curso “Access Avanzado” durante el periodo del 02-12-96 al 15-03-97.

Hay que distinguir entre una ejemplar de un tipo de entidad y un tipo de interrelación, pues una ejemplar de un tipo de interrelación existe siempre y cuando existan ejemplares de los tipos de entidad que intervienen en la asociación. Los ejemplares no tienen representación gráfica en el modelo E/R pues se corresponden con los datos que realmente se almacenarán en la base de datos y no con el diseño conceptual de ésta.

Hemos visto los elementos básicos del modelo E/R que nos permitirán el diseño de la Base de Datos de forma Conceptual, es decir, tendremos una representación sencilla y natural del caso que queremos modelar que, además, no depende del Sistema Gestor de Bases de Datos que utilizamos para su posterior implementación y que lo que intentará será recoger de la mejor forma posible todas las especificaciones del problema de manera que sea fácilmente comprensible por usuarios no informáticos.

**Autoevaluación: . - PARTE II.1: PREGUNTA DE REFUERZO.**

Veamos un cuadro **resumen de los conceptos del Modelo E/R** tratados hasta el momento:

Elementos del modelo E/R	Representación Gráfica	Descripción
<b>Entidad</b>		Cosa u objeto con identidad propia de la que necesitamos guardar información.
Ejemplar de una entidad		Un ejemplar, también denominado ejemplar, de un tipo de entidad es el conjunto de los valores correspondientes a los atributos definidos para ella.
<b>Atributo</b>		Característica o propiedad de un tipo de entidad.
Identificador principal		Identifica de manera única los ejemplares o ejemplares de una entidad
Identificador alternativo		Distingue de manera única los ejemplares o ejemplares de una entidad
Atributo Obligatorio		Indica que el atributo siempre debe tomar un valor para cada ejemplar de la entidad o interrelación a la que pertenece
Atributo Opcional		Indica que el atributo puede no tomar valor para cada ejemplar de la entidad o interrelación a la que pertenece
<b>Interrelación</b>		Asociación o relación que existe entre entidades.
Grado de una interrelación		Número de entidades que participan en una interrelación.
Tipo de Correspondencia en interrelaciones binarias	1:N N:M 1:1	Número máximo de ejemplares de un tipo de entidad que pueden estar asociados con un ejemplar del otro tipo de entidad.
Cardinalidades Mínima y Máxima en interrelaciones binarias	(x, y)	Número máximo y mínimo de ejemplares de una entidad que puede relacionarse con un único ejemplar de la otra.
Ejemplar de una interrelación binaria		Un ejemplar, también denominado ejemplar, de una interrelación es la asociación de los valores de los atributos identificadores principales de las entidades participantes en la interrelación.

### 1.3. Extensiones del Modelo E/R

Posteriormente al modelo E/R propuesto por Chen se realizaron algunas extensiones para darle más riqueza semántica. Esto significa que se le han añadido nuevos conceptos para que el modelo se adapte mejor a la realidad que queremos modelar, es decir, recoja mayor semántica.

Vamos a introducir algunos de estos nuevos conceptos retomando el ejemplo visto en el apartado anterior sobre una empresa en el que habíamos representado la relación que existía entre los empleados y los departamentos de la empresa.

Supongamos que la empresa es un consorcio de distintas librerías especializadas en libros y revistas informáticas llamada INTERFAZ. Sabemos que los empleados de INTERFAZ están asignados a un departamento y que la relación entre EMPLEADO y DEPARTAMENTO se representa como se indica en la figura 2.20.

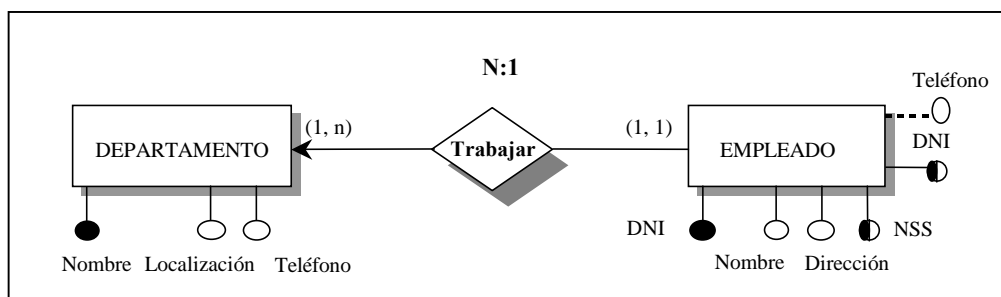


Figura 2.20: Interrelación Trabajar

### Entidades

En el apartado 2 se estudió que las entidades en un esquema E/R son los objetos principales sobre los que debe recogerse información y generalmente denotan personas, lugares, cosas o eventos de interés. En esta sección vamos a estudiar cómo las entidades pueden clasificarse por la fuerza de sus atributos identificadores.

Las entidades fuertes o regulares tienen existencia propia, es decir, poseen identificadores internos que determinan de manera única la existencia de sus ejemplares. Las entidades débiles son dependientes de otras entidades y pueden serlo por dos motivos: bien porque la existencia de sus ejemplares en la base de datos depende de una entidad fuerte bien porque sus ejemplares

requieran para su identificación de los atributos identificadores (algunas veces llamados atributos externos) de otra entidad.

Por ejemplo, los ejemplares correspondientes a los alumnos del MENTOR no dependen de ninguna otra entidad para existir en la base de datos; por ello la entidad ALUMNO es una entidad fuerte. Sin embargo, en el caso de una base de datos de una cadena hotelera podríamos tener el tipo de entidad HABITACIÓN dependiente del tipo de entidad HOTEL ya que para que existan ejemplares de HABITACIÓN es necesario que existan ejemplares de HOTEL. Un ejemplar de HABITACIÓN no tiene existencia por si misma porque siempre estará asociada a una ejemplar de HOTEL. Además, si se elimina un determinado ejemplar de la entidad HOTEL de la base de datos también deberán desaparecer los ejemplares de la entidad HABITACIÓN asociadas a él. La representación de una entidad débil difiere de la de una entidad regular pues el rectángulo de la entidad débil es de doble recuadro como se muestra en la figura 2.21.



Figura 2.21: Notación gráfica de una entidad débil

### Interrelaciones binarias

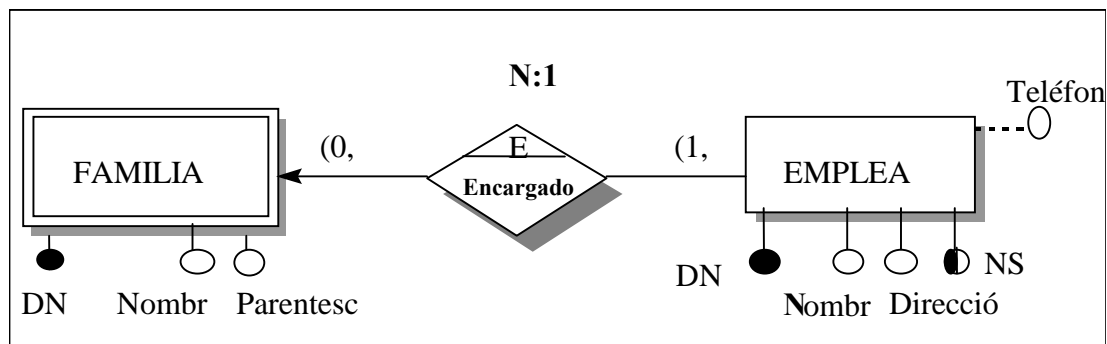
La clasificación anterior entre entidades fuertes y débiles da lugar a dos tipos de interrelaciones según los tipos de entidades que asocian.

Las interrelaciones regulares relacionan tipos de entidades regulares o fuertes. Las interrelaciones débiles relacionan un tipo de entidad regular y un tipo de entidad débil. Además, en las interrelaciones débiles podemos distinguir:

Dependencia en existencia. Este tipo de interrelación refleja que los ejemplares del tipo de entidad débil que se relacionan con un determinado ejemplar del tipo de entidad regular dependen de él y, si éste desaparece, ellos también. Veamos un ejemplo que clarifique esta definición:

Supongamos que la empresa INTERFAZ necesita conocer los datos de los familiares que están a cargo de cada empleado de la empresa (cónyuge, hijos, etc.) para de esta manera apoyar a aquellos cuya carga familiar sea numerosa.

Para saber los familiares que dependen de cada empleado debemos crear un nuevo tipo de entidad, que denominaremos FAMILIAR, cuyos atributos podrían ser el DNI (como IP), el nombre completo y parentesco con el empleado. Como se puede observar, la existencia de un miembro de la familia depende plenamente de que ese miembro tenga a una persona de su familia trabajando en la empresa, o lo que es lo mismo que exista un ejemplar de EMPLEADO que este relacionado con él; es decir, los familiares sólo existen en la base de datos si existe un empleado con el que se relacionen y si un determinado EMPLEADO se va de la empresa, entonces se eliminarán todas los ejemplares de FAMILIAR que dependan de él. Así, tenemos una interrelación de dependencia en existencia entre EMPLEADO y FAMILIAR representada como muestra la figura 2.22.



**Figura Ejemplo una interrelación Dependencia Existen**

Dependencia en Identificación: Este tipo de interrelación complementa a la anterior en que, además de que los ejemplares del tipo de entidad débil dependen de la existencia de un ejemplar de la entidad regular, también necesitan para su identificación el IP de la entidad regular. Así, veíamos anteriormente que la entidad HABITACIÓN era débil respecto al HOTEL al que pertenece. Si construimos la interrelación existente entre ambas entidades debemos pensar si el atributo “Nº de Habitación” de la entidad HABITACIÓN es suficiente para identificar cada ejemplar de esta.

Como se muestra en la figura 2.23 el IP, es decir, el número de habitación se repite para distintos hoteles (la habitación número 1 existe en el hotel “Mar” y en el hotel “Sol”). Para solucionar este problema, existen dos soluciones:

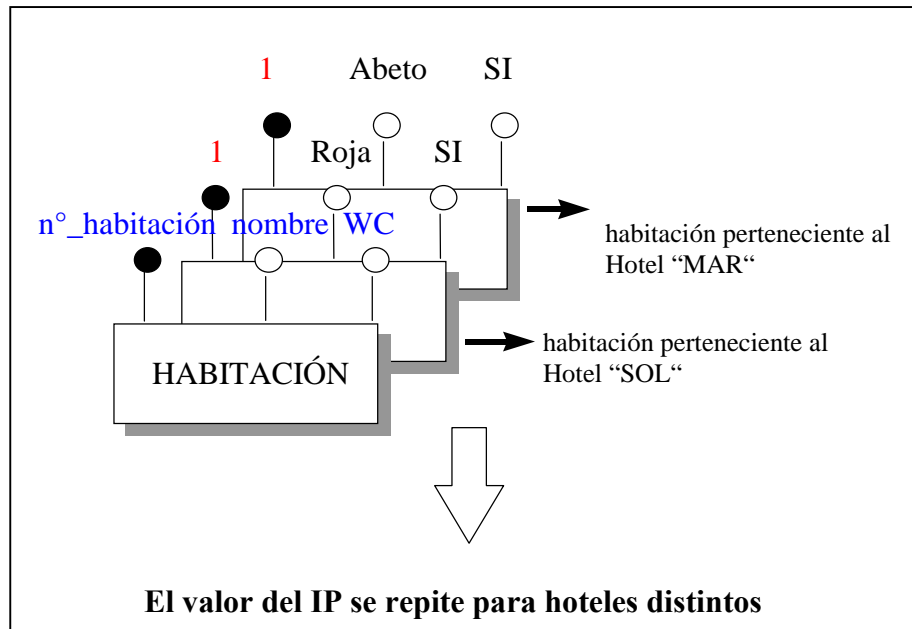


Figura 2.23: Ocurrencias de la entidad

- (a) La primera consiste en cambiar el IP, por ejemplo, poner el nombre de la habitación como IP; esto significa que los nombres de la habitación no pueden repetirse en los distintos hoteles y esto no es posible asegurarlo.
- (b) La segunda, y más razonable, consiste en crear una interrelación débil de dependencia en identificación, es decir, los ejemplares de la entidad débil requieren para su identificación de los atributos identificadores de la entidad fuerte. Así, cada ejemplar de **HABITACIÓN** está identificada por la concatenación de su número y del nombre del hotel en que se encuentra. Por ejemplo, la habitación 1 “Sol”, habitación 1 “Mar”, etc. Su representación es la que se muestra en la figura 2.24.



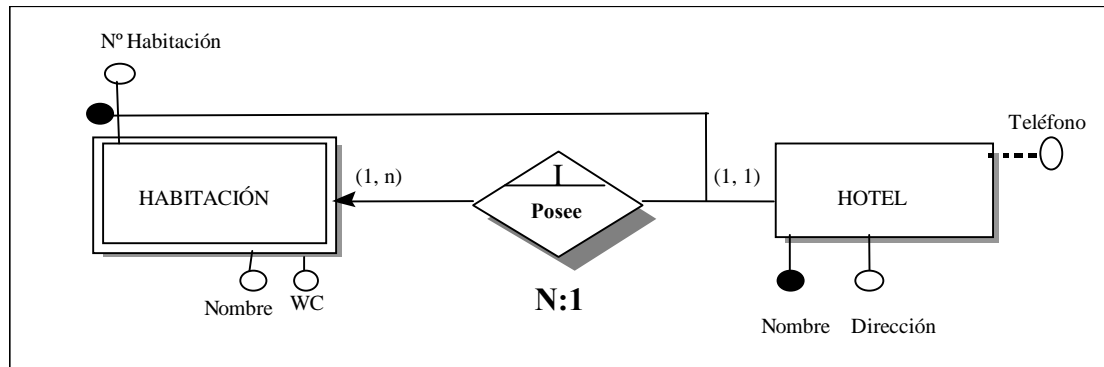
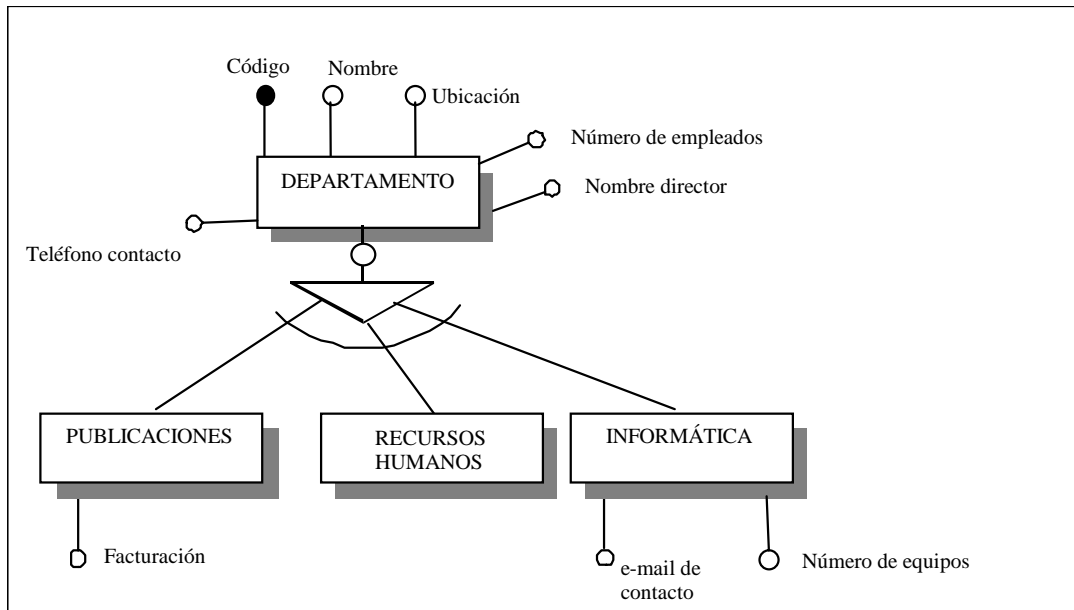


Figura 2.24 : Ejemplo de una interrelación con Dependencia en Identificación

### Autoevaluación: .- PARTE II.1: PREGUNTA DE REFUERZO.

Otro tipo de interrelación es la denominada jerárquica que expresa la clasificación de un determinado tipo de entidad en uno o más tipos de entidad. Por ejemplo, supongamos que la empresa Interfaz tiene tres departamentos INFORMATICA, PUBLICACIONES y RECURSOS HUMANOS. Esta clasificación de los departamentos se representaría como una jerarquía (también denominada generalización). Las generalizaciones nos proporcionan un mecanismo de abstracción que permite descomponer una entidad (que se denominará supertipo) en subtipos. De esta forma vemos un conjunto de ejemplares de una entidad como de otra entidad. Así, por ejemplo, una "Persona" es un "Animal" y un "Reptil" es un "Animal"; en este caso, "Animal" puede considerarse el supertipo y "Persona" y "Reptil" son subtipos de "Animal". Los ejemplares o ejemplares de "Persona" lo son también de "Animal" e igual sucede con las de "Reptil".

La figura 2.25 muestra la jerarquía de departamentos de la empresa INTERFAZ representada por un triángulo invertido que une el supertipo con los subtipos.



**Figura 2.25: Ejemplo de generalización total exclusiva**

### **Autoevaluación: .- PARTE II.1: PREGUNTA DE REFUERZO.**

#### **Atributos**

En este apartado ampliaremos nuestro conocimiento acerca de las restricciones semánticas sobre los atributos de las entidades y de las interrelaciones para de esta forma poder representar más fielmente los requisitos que nos piden para el diseño de una determinada base de datos.

Supongamos que en la entidad Empleado queremos recoger que un empleado puede tener más de un teléfono, tendríamos un atributo Teléfonos que tendría cero o más valores, esto es lo que llamamos atributo multivaluado y se representa como se muestra en la figura 2.26.

Otro tipo de atributo es el atributo compuesto, que representa una agregación de atributos simples. Vamos a modificar el atributo Nombre de la entidad EMPLEADO ya que queremos un atributo, Nombre Completo, compuesto por Nombre y Primer Apellido. Su representación sería la que se muestra en la figura 2.26.

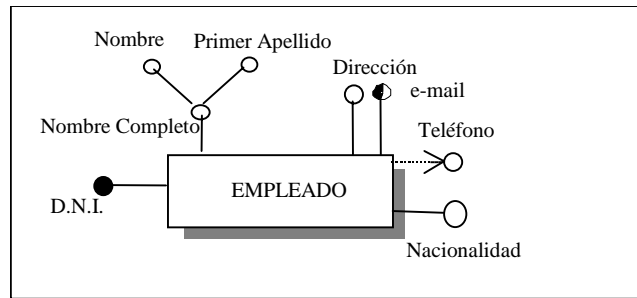


Figura 2.26: Ejemplo de atributo compuesto

**Autoevaluación: .- PARTE II.1: PREGUNTA DE REFUERZO.**

Además, todas las restricciones semánticas definidas para los atributos pueden combinarse entre sí, es decir, (pueden existir en un esquema E/R atributos multivaluados simples opcionales, univaluados compuestos opcionales, multivaluados obligatorios, multivaluados compuestos, etc.).

**Autoevaluación: .- PARTE II.1: MODELO ENTIDAD/INTERRELACIÓN****1.4. Caso Práctico**

Para este caso práctico se ha pensado en una continuación del ejemplo que se ha ido desarrollando a lo largo de este capítulo sobre el diseño de una Base de Datos que recoja información acerca del proyecto llevado a cabo por el Ministerio de Educación denominado MENTOR. Hay que tener en cuenta que los supuestos semánticos de este ejemplo son hipotéticos. A continuación se expondrán los requisitos que se van considerar en este apartado para llevar a cabo el diseño de la base de datos. Dicho proyecto se encarga de ofertar cursos por Internet para alumnos del territorio nacional.

- La información que se desea almacenar en la Base de Datos se refiere a los alumnos matriculados en cada curso, teniendo en cuenta la fecha de inicio y la fecha de finalización de cada alumno en un determinado curso y sabiendo que un alumno se ha podido matricular de uno o varios cursos y que un curso tiene como mínimo un alumno.
- De los alumnos se desea saber el DNI, nombre completo, dirección, teléfono, nacionalidad, pero sólo interesa saber si la nacionalidad es española o no, y la dirección de correo

electrónico. La dirección de correo electrónico es imprescindible para poder realizar los cursos y además es única para cada alumno.

La información referente a los cursos consta del nombre, título del libro de consulta que se utiliza (aunque existen cursos que no lo poseen) y dirección de Internet donde se encuentra todo el material que se puede utilizar durante el curso.

- Cada curso tiene asociado un grupo de personas expertas, llamadas tutores, que son las encargadas de resolver las dudas propuestas por los alumnos, la evaluación de los mismos e incluso el hombro para que estos se desahoguen. Dentro de los tutores de un mismo curso existe una figura importante que es la de coordinador que se encarga de realizar labores de unificación y planificación. No hay que olvidar que una persona experta puede ser tutora de varios cursos y que además un coordinador de curso es un tutor. La información que se quiere almacenar en la BD acerca de los tutores es la siguiente: DNI, nombre completo y dirección de correo electrónico.
- El proyecto MENTOR, además, tiene en cuenta que ha de facilitar a los alumnos el acceso a Internet y por lo tanto ha instalado aulas con todos los servicios necesarios para el pleno desarrollo de los cursos.
- Cada alumno pertenece a un aula y el mantenimiento tanto de los ordenadores como de los programas se lleva a cabo por los administradores de aula.
- Cada aula tiene asignado un código único, un nombre y una dirección.
- La información que se necesita de cada administrador es su DNI, nombre completo, dirección de correo electrónico.

### **Diseño propuesto**

Para realizar el diseño conceptual de la BD en el modelo E/R seguiremos una serie de pasos que nos ayudarán a identificar los elementos básicos del modelo. Estos pasos son iterativos, es decir, un esquema E/R se construye según distintas fases de refinamiento. Además, las soluciones no son únicas, cada diseñador puede ver el mundo real de distinta forma, dando lugar a distintos esquema E/R válidos. Sin embargo, si se puede estudiar si un determinado esquema E/R refleja mejor que otro los supuestos semánticos del enunciado. No hay que olvidar que en el esquema E/R de una base de datos hay que recoger la mayor semántica posible y no dejar para las siguientes fases de desarrollo (diseño lógico e implementación) ningún supuesto semántico, siempre que sea posible.



**1<sup>er</sup> paso:** Identificar y enumerar las posibles entidades teniendo en cuenta la siguiente heurística: en general, un tipo de entidad es un *sustantivo* dentro de una oración con una serie de propiedades o características. Por ejemplo, DNI del alumno, nombre del curso, etc...

En el texto presentamos en **negrita** y subrayado los tipos de entidad que hemos detectado.

- La información que se desea almacenar en la Base de datos se refiere a los alumnos matriculados en cada curso. Teniendo en cuenta la fecha de inicio de cada alumno en un determinado curso así como su fecha de finalización y sabiendo que un alumno se ha podido matricular de uno o varios cursos y que un curso tiene como mínimo un alumno.
- De los **alumnos** se desea saber el DNI, nombre completo, dirección, teléfono, nacionalidad, pero sólo interesa saber si la nacionalidad es española o no, y la dirección de correo electrónico. La dirección de correo electrónico es imprescindible para poder realizar los cursos y además única para cada alumno.
- La información referente a los **cursos** consta de nombre de este, título del libro de consulta que utiliza (aunque existen cursos que no lo poseen) y dirección de Internet donde se encuentra todo el material del que consta.
- Cada curso tiene asociado un grupo de personas expertas, llamadas tutores, que son las encargadas de resolver los problemas propuestos por los alumnos, la evaluación de los mismos e incluso el hombro para que estos se desahoguen. Dentro de los tutores de un mismo curso existe una figura importante que es la de coordinador que se encarga de realizar labores de unificación. No hay que olvidar que una persona experta puede ser tutora de varios cursos y que además un coordinador de curso es un tutor. La información que se quiere almacenar en la BD acerca de los **tutores** es la siguiente: DNI, nombre completo y dirección de correo electrónico.
- El proyecto MENTOR, además, tiene en cuenta que ha de facilitar a los alumnos el acceso a Internet y por lo tanto a instalado aulas con todos los servicios necesarios para el pleno desarrollo de los cursos.
- Cada alumno pertenece a un aula y el mantenimiento tanto de los ordenadores como de los programas se lleva a cabo por los administradores de aula.
- Cada **aula** tiene asignado un código único, un nombre y una dirección.
- La información que se necesita de cada **administrador** es su DNI, nombre completo, dirección de correo electrónico.

Los tipos de entidades que hemos localizado son: ALUMNO, CURSO, TUTOR, AULA y ADMINISTRADOR. Del enunciado se podría deducir que COORDINADOR es también un tipo de entidad; dejamos para más adelante la discusión sobre si este concepto puede representarse como una entidad, un atributo o una interrelación.



**2º paso:** Identificar y enumerar las posibles interrelaciones, teniendo en cuenta la siguiente heurística: en general, una interrelación viene reflejada por un *verbo* dentro de una oración que relaciona dos objetos.

En el texto aparece un número correlativo como superíndice en los verbos que indican la posible existencia de una interrelación.

- La información que se desea almacenar en la Base de datos se refiere a los alumnos matriculados<sup>1</sup> en cada curso. Teniendo en cuenta la fecha de inicio de cada alumno en un determinado curso así como su fecha de finalización y sabiendo que un alumno se ha podido matricular de uno o varios cursos y que un curso tiene como mínimo un alumno.
- De los alumnos se desea saber el DNI, nombre completo, dirección, teléfono, nacionalidad, pero sólo interesa saber si la nacionalidad es española o no, y la dirección de correo electrónico. La dirección de correo electrónico es imprescindible para poder realizar los cursos y además única para cada alumno.
- La información referente a los cursos consta de nombre de este, título del libro de consulta que utiliza (aunque existen cursos que no lo poseen) y dirección de Internet donde se encuentra todo el material del que consta.
- Cada curso tiene asociado<sup>2</sup> un grupo de personas expertas, llamadas tutores, que son las encargadas de resolver los problemas propuestos por los alumnos, la evaluación de los mismos e incluso el hombro para que estos se desahoguen. Dentro de los tutores de un mismo curso existe una figura importante que es la de coordinador que se encarga de realizar labores de unificación. No hay que olvidar que una persona experta puede ser tutora de varios cursos y que además un coordinador de curso es un tutor. La información que se quiere almacenar en la BD acerca de los tutores es la siguiente: DNI, nombre completo y dirección de correo electrónico.
- El proyecto MENTOR, además, tiene en cuenta que ha de facilitar a los alumnos el acceso a Internet y por lo tanto a instalado aulas con todos los servicios necesarios para el pleno desarrollo de los cursos.
- Cada alumno pertenece<sup>3</sup> a un aula y el mantenimiento<sup>4</sup> tanto de los ordenadores como de los programas se lleva a cabo por los administradores de aula.
- Cada aula tiene asignado un código único, un nombre y una dirección.
- La información que se necesita de cada administrador es su DNI, nombre completo, dirección de correo electrónico.

Para que nos sea más sencillo saber qué tipos de entidades están relacionadas vamos a construir una matriz donde en la primera fila y la primera columna se enuncian los tipos de entidad anteriormente enumerados y se señalará en el cruce de filas y columnas aquellas interrelaciones que hemos detectado. De esta forma se facilita también la identificación de posibles interrelaciones que no aparecen explícitamente expresadas en los supuestos semánticos del enunciado pero que son, bien de sentido común, bien deducidas del enunciado; estas interrelaciones también tienen que aparecer en el esquema E/R de la base de datos.

Interrelaciones	ALUMNO	CURSO	TUTOR	AULA	ADMINISTRADOR
ALUMNO		Matricular 1		Pertenecer 3	
CURSO			Asociar 2 ¿Coordinar?		
TUTOR					
AULA					Mantener 4
ADMINISTRADOR					

En la tabla se muestra el nombre de las interrelaciones y una numeración que indica el orden en el que han ido apareciendo en el texto; se muestran sombreadas las celdas que representan interrelaciones simétricas a las definidas en el resto de las celdas. Además de las interrelaciones extraídas del texto hay que estudiar si en las celdas vacías deberían aparecer nuevas interrelaciones. Así, podría existir la interrelación Coordinar entre TUTOR y CURSO; dejaremos esta interrelación entre interrogaciones con el fin de estudiar posteriormente si debe reflejarse de esta forma.



**3<sup>er</sup> Paso:** Dibujar las interrelaciones (estudiando el tipo de correspondencia y las cardinalidades) y los tipos de entidad con los atributos correspondientes.

### Interrelación Matricular

Tanto los atributos de CURSO como de ALUMNO se presentaron a lo largo del capítulo, pero recordamos que el IP (identificador principal) de CURSO es *Nombre* y el de ALUMNO es *DNI*. Tanto el atributo *WWW* como el *e-mail* son identificadores alternativos, es decir, los valores que toman para cada elemento del tipo de entidad CURSO o ALUMNO son únicos; Los atributos *Libro* y *Teléfono* son opcionales. La figura 2.27 muestra el esquema E/R correspondiente a la interrelación **Matricular**.

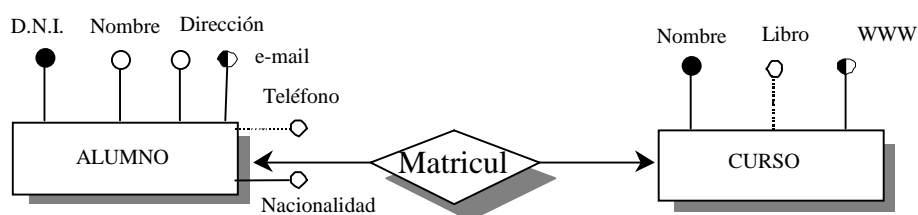


Figura 2.27: Interrelación **Matricula**

El estudio del tipo de correspondencia y las cardinalidades máximas y mínimas también se realizó durante el desarrollo del capítulo pero recordaremos que la correspondencia es *de muchos a muchos* (N:M) ya que un alumno puede matricularse de uno o varios cursos, cardinalidad (1,n), y en un curso se matriculan uno o varios alumnos, cardinalidad (1,n). Además, para saber las fechas en las que un alumno inició y finalizó un curso se introducen dos atributos que pertenecen a la interrelación Matricular (figura 2.28)

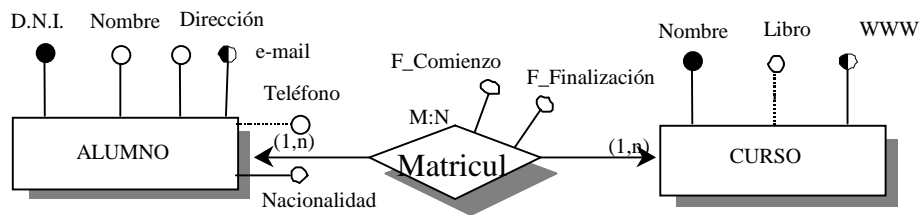


Figura 2.28: Interrelación **Matricula** con cardinalidades y atributos en la interrelación

### Interrelación Asociar

Antes de analizar las propiedades de la interrelación **Asociar** veamos los atributos del tipo de entidad TUTOR.

Según se muestra en el enunciado, la información que se requiere para los tutores es: *DNI*, *nombre completo* y *dirección de correo*. De estos tres atributos tenemos que elegir cual de ellos puede ser el IP. Elegiremos el *DNI* aunque bien podría ser la dirección de correo si esta es única. Por lo que el *e-mail* será un atributo alternativo y el *nombre completo* un atributo obligatorio.

El tipo entidad y sus atributos quedan representados como muestra la figura 2.29.

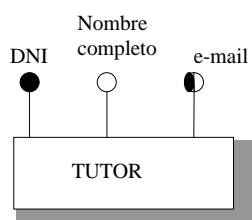


Figura 2.29: Entidad TUTOR

La interrelación **Asociar** relaciona las entidades TUTOR y CURSO (véase la tabla del paso 2 y Figura 2.30); tenemos que estudiar el tipo de correspondencia y las cardinalidades máximas y mínimas para completar las propiedades de la interrelación.

Si leemos detenidamente las especificaciones del texto tenemos que un tutor puede realizar sus labores en varios cursos y que en un curso puede ser tutorado por varias personas expertas (tutores) por lo tanto la correspondencia es N:M o *muchos a muchos*.



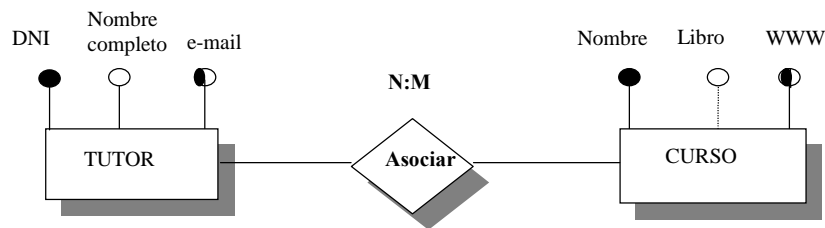


Figura 2.30: Interrelación Asociar

Veamos ahora las cardinalidades máximas y mínimas del tipo de entidad TUTOR en la interrelación **Asociar**; para ello, tenemos que mirar al tipo de entidad TUTOR y preguntarnos:

- ¿A cuántos cursos está asociado como mínimo un TUTOR? La respuesta podría ser 0 si consideramos que podemos tener tutores que en un momento dado no estén tutorando ningún curso, o, por el contrario, podríamos poner un 1 con lo que supondríamos que todos los tutores que tenemos en nuestra base de datos siempre estarán ocupados con algún curso. Nos quedamos con la primera alternativa para poder dejar descanso a los tutores. De esta forma la cardinalidad mínima es 0.
- ¿A cuántos cursos está asociado como máximo un TUTOR? En este caso, como en las especificaciones no se restringe el número de cursos que un tutor puede impartir, la cardinalidad máxima será N.

Gráficamente, las cardinalidades de la entidad TUTOR se representan al lado contrario de esta, es decir, junto al tipo de entidad CURSO (Figura 2.31).

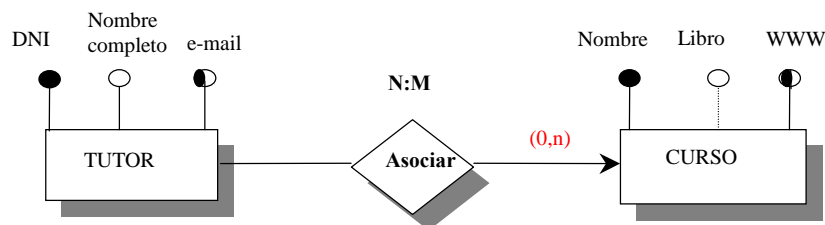
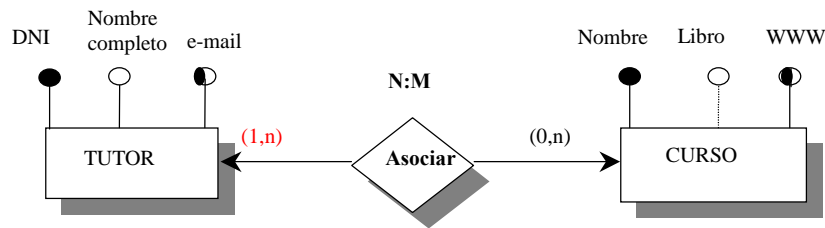


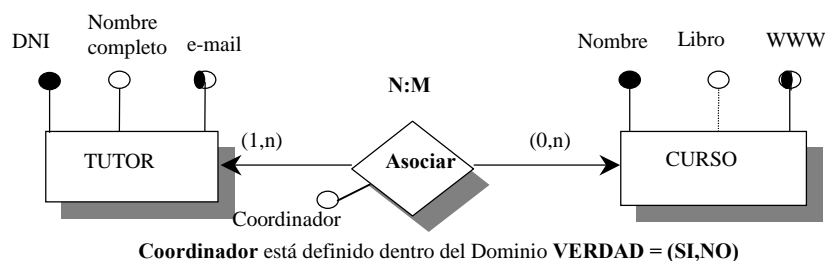
Figura 2.31: Cardinalidad de TUTOR en la interrelación Asociar

De forma análoga se razonaría para el caso de las cardinalidades asociadas a CURSO (figura 2.32). Un curso como mínimo ha de ser tutorado por un TUTOR (cardinalidad mínima 1) y como máximo por N (cardinalidad máxima N).

Figura 2.32: Cardinalidades en la interrelación **Asociar**

En los pasos 1 y 2 dejamos sin estudiar el concepto de coordinador de los cursos. Volviendo a releer el texto nos preguntamos ¿qué pasa con la figura del coordinador?, ¿cuál sería su representación?

Como bien se indica en los supuestos semánticos del enunciado, el coordinador es también un tutor y, además, puede desarrollar una función añadida de planificación en determinados cursos. Esto significa que un tutor en determinados cursos (pero solamente en aquellos donde participa) puede realizar dos labores: tutor y coordinador. Por lo que una solución puede ser considerar, dentro de la interrelación **Asociar** un atributo, **Coordinador**, definido dentro del dominio **Verdad** que toma los valores (**SI**, **NO**), y el cual nos indicará con el valor **SI** que un determinado tutor desarrolla la función de coordinador en un curso con el que se relaciona (Figura 2.33).

Figura 2.33: Interpretación de la figura **Coordinador**

Analicemos como se interpreta el atributo **Coordinador** en la interrelación **Asociar**. El atributo **Coordinador** toma un valor para cada ejemplar de la interrelación **Asociar**; esto significa que como un determinado curso puede tener más de un tutor y el atributo **Coordinador** podría tomar el valor de **SI** en esas ejemplares de la interrelación **Asociar**, entonces estamos permitiendo en el

esquema E/R que un curso tenga más de un coordinador. Esto significa que no respetamos uno de los supuestos semánticos del enunciado.

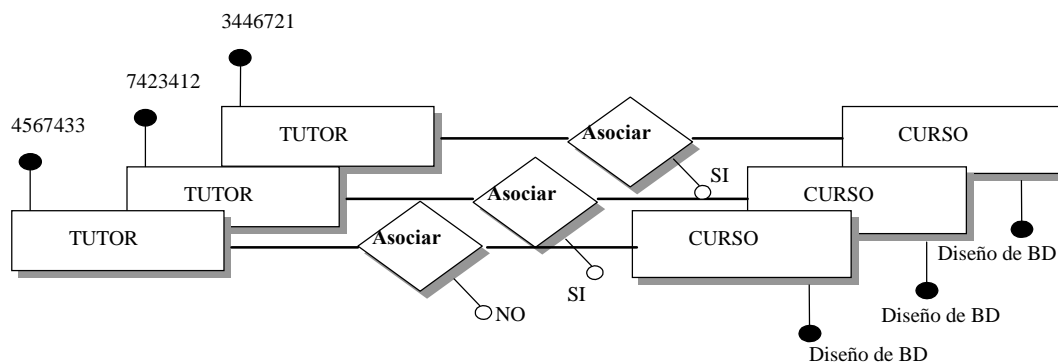


Figura 2.34: Ejemplares de la interrelación **Asociar**

Veámoslo con los ejemplos de ejemplares de la interrelación mostrados en la figura 2.34; el curso “Diseño de BD” tiene tres tutores cuyos DNI son 3446721, 7423412, 4567433. El tutor con DNI 4567433 no es coordinador y los tutores con DNI 3446721 y 7423412 están definidos con coordinadores. Si no queremos violar la restricción semántica de que un curso no tenga más de un coordinador, entonces en el diseño lógico de la BD se debería definir algún mecanismo que cuando se haya definido un coordinador para un curso, entonces no se permita introducir ninguno más. Sin embargo, la solución de la figura 2.34 si contempla la restricción semántica consistente en que los coordinadores de los cursos deben ser tutores de los mismos, es decir, no es posible definir un coordinador de un curso que no sea tutor del mismo.

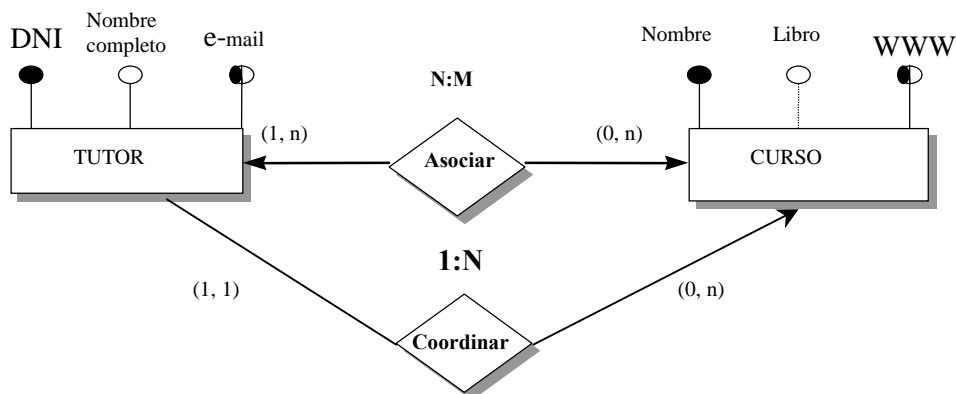


Figura 2.35: Interrelación **Coordinar**.

Otra posible solución para representar la semántica de la figura de coordinador de un curso consiste en utilizar otra interrelación denominada **Coordinar** entre **TUTOR** y **CURSO** con las cardinalidades mostradas en la figura 2.35. La interrelación **Coordinar** representa que un determinado **TUTOR** puede ser coordinador de más de un **CURSO** y que un **CURSO** tiene uno y

solo un TUTOR que lo coordina. Si bien esta propuesta de solución recoge a la perfección la restricción de que un curso sólo tiene un coordinador, sin embargo, no recoge que el coordinador de un curso tenga que ser obligatoriamente un tutor de ese curso (ver figura 2.36). Para controlar esta última restricción habría que incluir un mecanismo en el diseño lógico que obligara a que el coordinador de un curso debe ser un tutor del mismo.

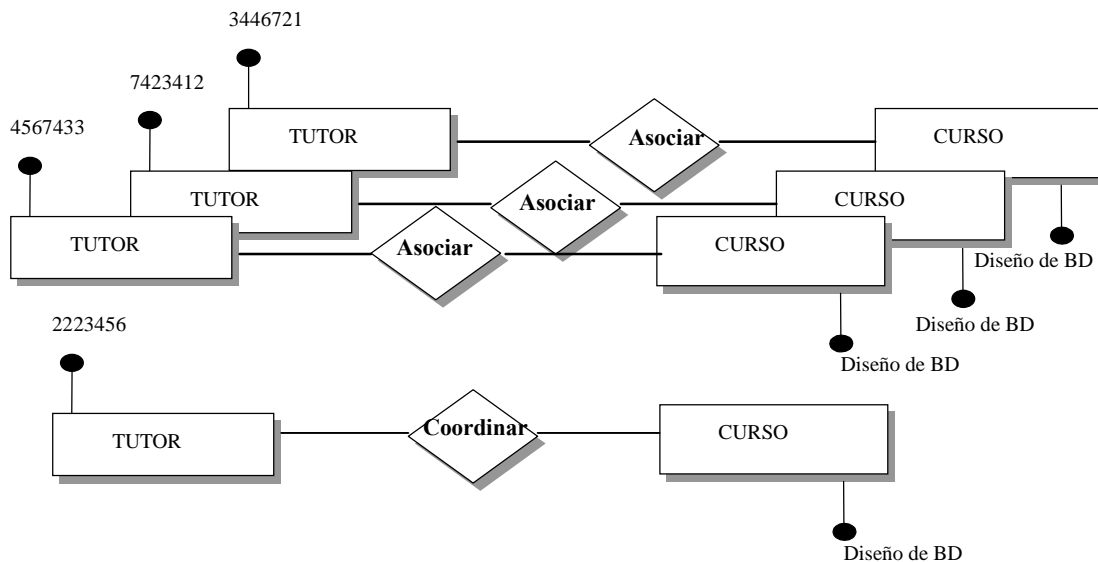


Figura 2.36: Ejemplares de la interrelación **Asociar** y **Coordinar**

Aunque existen otras formas de representar la figura del coordinador de un curso, no vamos a presentarlas aquí con el fin de no complicar el ejercicio. Se han mostrado las dos más significativas. Para este caso práctico se ha seleccionado la primera propuesta de solución (considerar el atributo **Coordinador** en la interrelación **Asocia**). De esta forma, el esquema E/R definido hasta el momento se muestra en la figura 2.37:

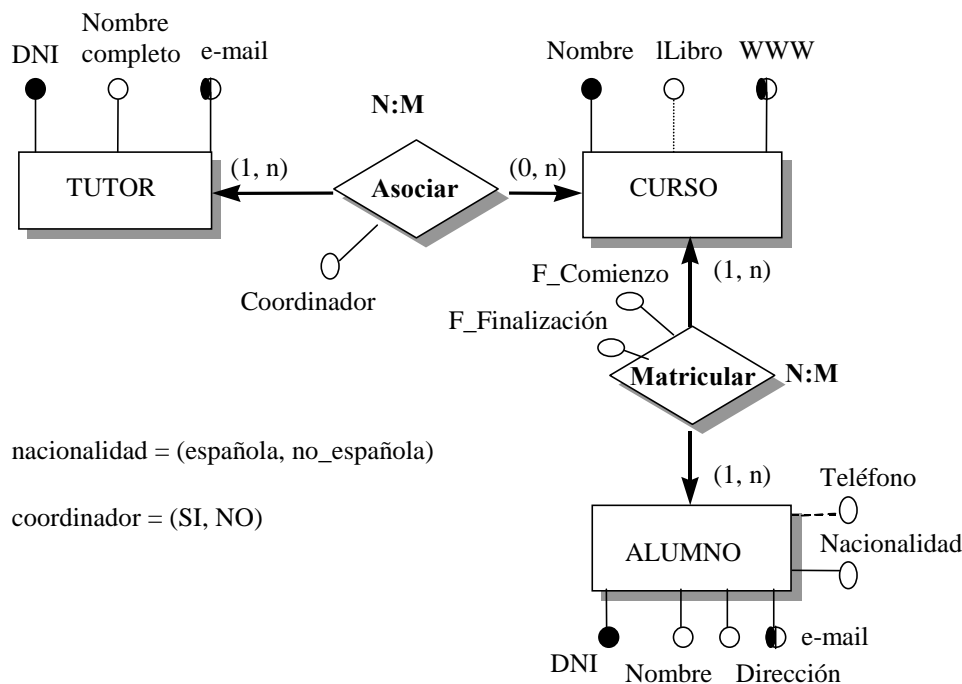


Figura 2.37: Esquema E/R parcial

### Interrelación Pertenecer

Esta interrelación asocia las entidades ALUMNO y AULA. En primer lugar se representarán los atributos del tipo de entidad AULA que (*código\_aula*, *nombre* y *dirección*). El *código\_aula* será el IP, y el resto de atributos serán obligatorios.

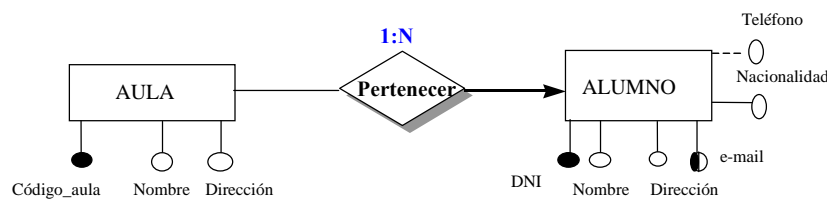
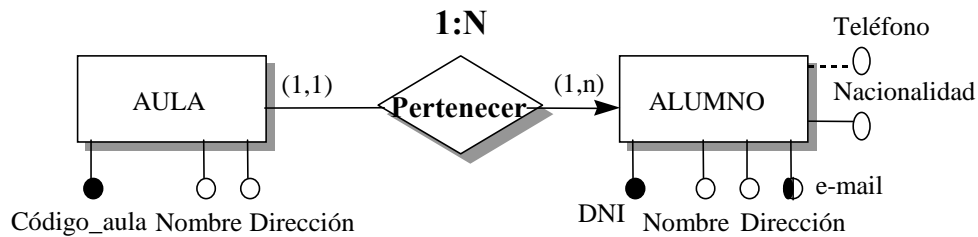


Figura 2.38: Interrelación Pertenecer

Para la interrelación **Pertenecer** tenemos un tipo de correspondencia de *uno a muchos* porque según el enunciado cada ALUMNO pertenece a un AULA (suponemos que un alumno no puede estar asociado a más de un aula); además, un AULA puede tener asociados varios ALUMNOS (figura 2.38). Las cardinalidad mínima y máxima de ALUMNO es (1,1) ya que el alumno está asignado a una y solo un AULA. Las asociadas a AULA serían (1,n) ya que no tendría mucho sentido mantener un aula sin alumnos (figura 2.39).

Figura 2.39: Cardinalidades de la interrelación **Pertenece**

El esquema E/R obtenido hasta el momento es el mostrado en la figura 2.40.

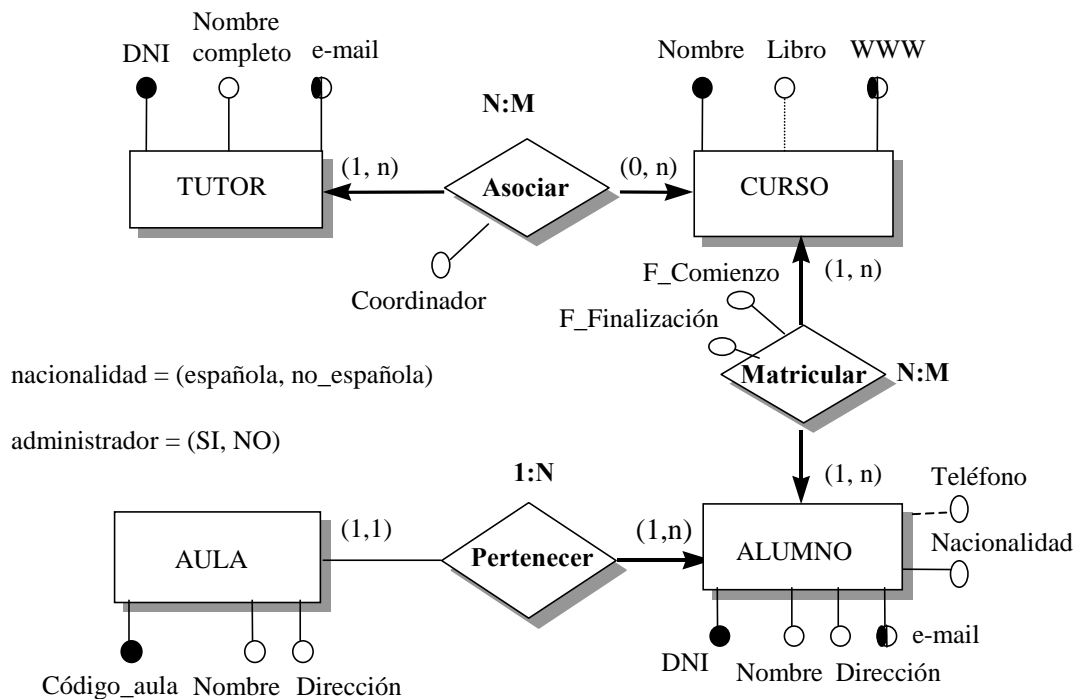


Figura 2.40: Esquema E/R parcial

### Interrelación Mantener

La interrelación **Mantener** se da entre las entidades **ADMINISTRADOR** y **AULA**. Los atributos del tipo de entidad **ADMINISTRADOR** son **DNI**, **nombre completo** y **e-mail** y representan el IP, un atributo obligatorio y otro alternativo, respectivamente.

El tipo de correspondencia es de *uno a muchos* (1:N) ya que un **ADMINISTRADOR** solamente puede estar asignado a un **AULA** y sin embargo un **AULA** puede ser mantenida por más de un **ADMINISTRADOR** lo que nos indica también las cardinalidades: (1,n) para el tipo de entidad **AULA** y (1,1) para el tipo de entidad **ADMINISTRADOR**.

Así, el diseño conceptual de la base de datos referente al proyecto MENTOR se representa en la figura 2.41<sup>12</sup>:

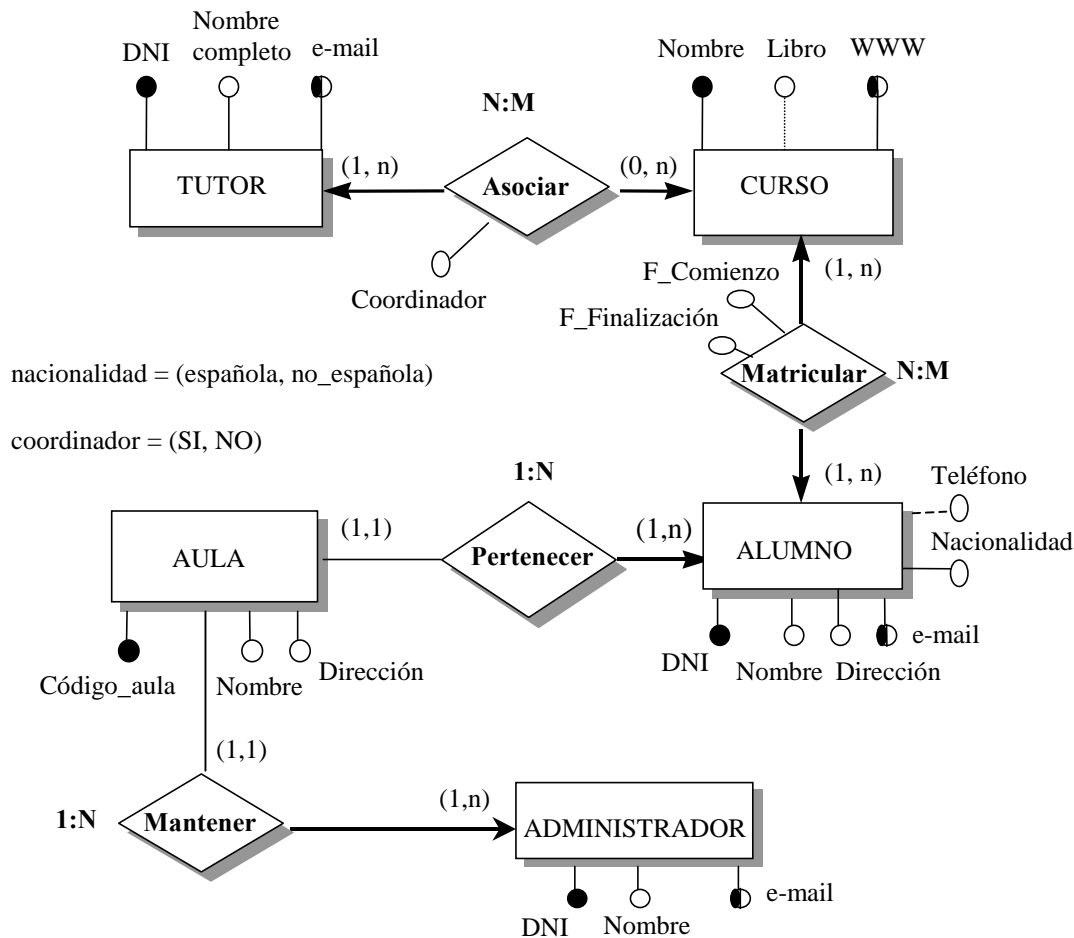


Figura 2.41: Esquema E/R completo

Observando el esquema E/R final, las entidades TUTOR y ADMINISTRADOR tienen atributos comunes. Podremos identificar generalizaciones si encontramos una serie de atributos comunes a un conjunto de entidades; estos atributos comunes describirán al supertipo y los atributos particulares permanecerán en los subtipos. Puede ocurrir que los subtipos no tengan atributos propios, como es el caso que nos ocupa; en ese caso, sólo existirán subtipos si éstos van a participar en interrelaciones (aparte de las interrelaciones en las que participe el supertipo). Así, podemos tener, como muestra la figura 2.42, el supertipo PERSONA con los atributos *DNI*, *nombre* y *e-mail* y los subtipos TUTOR y ADMINISTRADOR que no tienen ningún atributo propio. Los subtipos TUTOR y ADMINISTRADOR siguen participando en las mismas interrelaciones que en la figura 2.41. Sin embargo, el supertipo PERSONA no participaría en

<sup>12</sup> Como se ha mencionado al principio del ejercicio, esta representación conceptual no es única; puede haber diversas interpretaciones. Lo importante es que el usuario o la persona que nos ha encargado el diseño este conforme con este y refleje lo más fielmente posible las características del problema.

ninguna interrelación; por ello, se ha optado por eliminar la generalización de la solución propuesta.

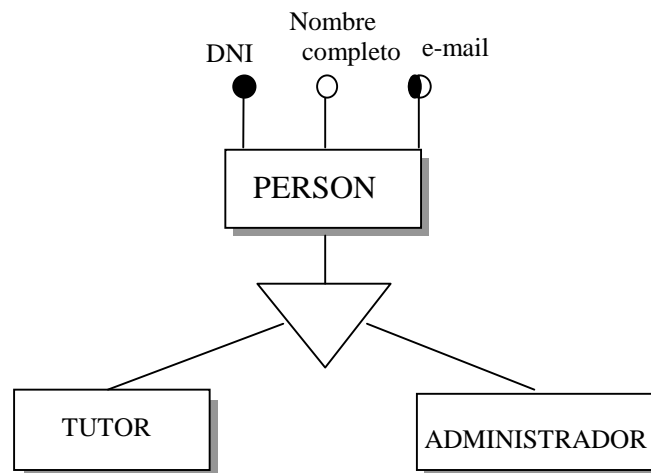


Figura 2.42: Generalización de PERSONA

## Caso Práctico Propuesto: .- PARTE II.1: MODELO ENTIDAD/INTERRELACIÓN

### 2. Fase de Diseño Lógico Estándar: Modelo Relacional

#### 2.1. Introducción

El modelo relacional fue propuesto en 1970 como alternativa a los modelos existentes hasta ese momento por el investigador Edgar Codd, tomando como base la teoría matemática de las relaciones. Este modelo perseguía la sencillez de comprensión y de manipulación de la base de datos por parte del usuario final.

En un principio el modelo era simplemente un modelo teórico objeto de investigación por parte de diversos centros, pero a medida que la tecnología fue evolucionando fueron surgiendo los primeros SGBD que lo soportaban y hoy en día el modelo relacional es el más conocido y difundido por los sistemas comerciales (ORACLE, INFORMIX,... ).



La parte estática del modelo relacional consiste en la definición de los objetos permitidos. En la parte dinámica se definen las operaciones que se pueden realizar con estos objetos y sobre la base de datos.

Para que un SGBD pueda implementar un modelo de datos necesita apoyarse en un lenguaje de programación que gestione dicho modelo. El lenguaje aceptado por todos los sistemas de bases de datos comerciales es el SQL<sup>13</sup>, lenguaje estructurado de consulta que permite crear, manipular y obtener datos de todos los objetos que constituyen la base de datos. Este lenguaje fue normalizado por la Organización Internacional de Estándares (ISO) en 1992<sup>14</sup> y sufre, como todos los estándares, revisiones periódicas que hacen que continuamente se editen anexos de la norma. No obstante y como casi todas las normas, no está soportada por ningún producto comercial, es decir, cada SGBD incorpora su propio SQL que recoge, dependiendo de cada sistema, lo más significativo del estándar.

En este capítulo nos vamos a centrar en definir los elementos básicos del modelo, así como sus partes estática y dinámica. Comentaremos las sentencias básicas del estándar SQL para operar con estos objetos, pero no entraremos con detalle en su sintaxis, pues ello sería objeto de un estudio más amplio que se escapa del objetivo de este curso.

Por último, explicaremos una metodología de diseño de bases de datos relacionales mediante la traducción de los objetos del modelo E/R explicado en capítulo anterior.

## 2.2. Elementos básicos del Modelo Relacional

Los elementos básicos del modelo relacional son las relaciones, cuya representación gráfica se realiza en forma de tabla. En las relaciones se pueden distinguir varios tipos de elementos: su nombre, los atributos que contiene y que representan las columnas de la tabla, y las tuplas o filas de la tabla. Además también podemos incluir los dominios que son aquellos conjuntos de donde los atributos toman los valores.

Supongamos que queremos modelar una base de datos para mantener información a cerca de los museos de la Comunidad de Madrid y las exposiciones que en ellos se pueden visitar, a través

---

<sup>13</sup> Structured Query Language

<sup>14</sup> Se conoce como SQL'92

del modelo de datos relacional . De cada museo se desea almacenar su nombre, la dirección, la zona (Madrid, El Escorial, etc) y el número de salas que contiene. Tendríamos la relación que se muestra en la figura 2.43.

MUSEO			
Nombre	Dirección	Zona	Salas
Arqueológico	Serrano, 13	Madrid	43
Centro de Arte Reina Sofía	Santa Isabel, 52	Madrid	21
Universidad de Alcalá de Henares	Plaza de San diego, s/n	Alcalá deHenares	7

Figura 2.43: Ejemplo de representación de una relación

El nombre de la relación de la figura es MUSEO, los atributos o columnas son “Nombre”, “Dirección”, “Zona” y “Sala”. Las tuplas o filas son (Arqueológico; Serrano, 13; Madrid; 43), (Centro de Arte Reina Sofía; Santa Isabel, 52; Madrid; 21) y (Universidad de Alcalá de Henares; Plaza de San Diego, s/n; Alcalá de Henares; 7 ). El atributo “Zona” pertenece a un dominio Zonas = (Madrid, Alcalá de Henares, Torrejón de Ardoz,...).

Dentro de una relación podemos distinguir además dos nociones: el grado y la cardinalidad. El grado es el número de atributos (columnas) que posee la relación, en el caso de nuestro ejemplo el grado sería cuatro. La cardinalidad se refiere al número de tuplas (filas) de la relación, en nuestro caso tres.

Hasta ahora hemos hablado de los elementos básicos del modelo, hemos dicho que las relaciones son tablas, pero estas tablas tienen sus limitaciones, no son las tablas que normalmente manejamos; por ejemplo, en estas tablas se prohíbe que exista más de un valor en cada celda, este tipo de prohibiciones pertenecen a las restricciones inherentes al modelo y las comentaremos más adelante en la siguiente sección.

A continuación presentamos la estática y la dinámica del modelo relacional definiendo formalmente todos los objetos y comentando las sentencias básicas del lenguaje SQL para tratarlas.

### 2.3. Estática del Modelo Relacional

Una relación, como se dijo en el apartado anterior se compone de un nombre, unos atributos (con sus correspondientes dominios) y un conjunto de tuplas. Es necesario distinguir entre lo que es la relación en sí y su contenido, es decir, una relación se define mediante un nombre y unos atributos de la siguiente manera:

NOMBRE\_RELACIÓN ( Atributo1: dominio1, Atributo2: dominio2,....., AtributoN: dominioN )

**Figura 2.44: Definición de relación**

Los dominios son opcionales, pues los atributos pueden tomar valores en un conjunto acotado definido previamente, o bien y en principio, ser infinitas las posibilidades de un valor determinado.

El cuerpo de una relación, es decir, su contenido, es el conjunto de tuplas de la relación que, por su propia naturaleza varía con el tiempo, pues no tendría mucho sentido disponer de una base de datos que almacenara en todo momento la misma información, que esta información no variara con el tiempo, ya que en este caso con disponer de ficheros aislados para almacenarla podríamos tener suficiente<sup>15</sup>.

En la figura 2.45 se muestra un ejemplo de las partes de las que se compone una relación.

---

<sup>15</sup> Estos son los llamados ficheros históricos, archivos que recopilan información que ya no va a variar, como podrían ser las transacciones efectuadas por los clientes de una sucursal bancaria en el año 1996.

Definición de relación:

**ARTISTA** (Nombre; Nacionalidad: Nacionalidades; Especialidad: Especialidades)

Contenido de la relación:

**ARTISTA**

Nombre	Nacionalidad	Especialidad
Agustín Redondella	Nacional	Pintura
Dolores Dahlahaus	Internacional	Fotografía
Mario Scifano	Internacional	Fotografía

**Figura 2.45: Definición y contenido de una relación**

Dentro de las relaciones existen principalmente dos tipos: relaciones base y vistas. Las relaciones base son relaciones con existencia propia, las vistas son relaciones que provienen de otras relaciones base. Un ejemplo de relación base sería la relación **ARTISTA** de la figura 2.45, una vista sería la relación obtenida de seleccionar en la relación anterior aquellos artistas cuya especialidad fuera la fotografía, a la que podríamos llamar **FOTÓGAFO**<sup>16</sup>.

Las relaciones se crean mediante el lenguaje SQL con las sentencias “CREATE TABLE” o “CREATE VIEW” dependiendo del tipo de relación de que se trate.

Hasta este momento únicamente hemos hablado de los objetos del modelo relacional, pero no hemos hecho mención de cómo interactúan estos objetos en la base de datos, ni de cómo diseñar realmente una base de datos en dicho modelo. Para ello debemos previamente comentar las restricciones, tanto las inherentes al modelo, como las restricciones de usuario, que son aquellas que permiten recoger con la mayor fidelidad posible el mundo real objeto de nuestro diseño.

<sup>16</sup> Nótese que hemos dado un nombre a la vista, pues una vista es un tipo de relación y por tanto debe constar de los elementos de toda relación.

Restricciones inherentes.-

1. Ningún atributo puede tomar más de un valor para cada tupla. Esto es lo mismo que decir que en cada una de las celdas de una tabla que represente una relación no puede haber más de un valor. Así por ejemplo, no sería válido que en la relación ARTISTA el artista Mario Scifano tuviera como especialidades Fotografía y Escultura. Otra cosa distinta sería que en nuestro mundo real un artista pudiera tener varias especialidades, en cuyo caso deberíamos modelar la relación de otra manera, por ejemplo creando otra donde se almacenaran todas las posibles especialidades y relacionándola con la primera, como veremos más adelante en el capítulo.
2. No importa el orden ni de las tuplas ni de los atributos.
3. Todas las tuplas de una relación deben ser distintas.

Pues bien, qué ocurriría si en nuestro mundo real existiesen dos artistas con el mismo nombre, nacionalidad y especialidad. No piense el lector que este supuesto es tan descabellado, por ejemplo podría darse el caso de un artista contemporáneo que coincidiera en nombre con otro clásico (al no almacenarse la fecha de nacimiento no habría forma de distinguirlos). Podríamos actuar de forma negligente y no almacenar uno de los dos, con lo cual la obra de este autor quedaría sin reflejarse. Otra posibilidad sería pensar en añadir un nuevo atributo que identificara a cada artista unívocamente dando valores distintos a cada tupla, como puede ser un “Código de Artista”. De esta forma cada tupla sería distinta de las demás.

Este atributo que identifica unívocamente cada tupla de una relación es lo que se denomina en el modelo relacional **clave primaria** y en SQL se define como “PRIMARY KEY”. Para representar la clave primaria de una relación, se suele poner en negrita el o los atributos implicados en dicha clave.

Algún lector avisado podría discrepar de la opción escogida para identificar a los artistas de nuestra relación, pues para qué se elige introducir un atributo cuyo contenido es tan pobre semánticamente hablando si, habiendo escogido la “Fecha de nacimiento”, éste, junto con el atributo “Nombre” identificarían por completo todas las tuplas. La justificación de nuestra elección radica en motivos de eficiencia, es mucho más eficiente a la hora de almacenar nuestra base de datos, una clave primaria corta en el sentido de que va a ser numérica o alfanumérica de pocos caracteres, mientras que el nombre y la fecha ocuparían más.

Hemos justificado la razón de nuestra elección de clave primaria, no obstante “Nombre” y “Fecha de nacimiento” sería también un identificador válido para la relación. Este identificador se denomina en el modelo relacional **clave alternativa**<sup>17</sup>. Es decir, una clave alternativa es aquella que aun pudiendo haber sido escogida como clave primaria de una relación, no lo ha sido por otros motivos, como pueden ser motivos de eficiencia, etc.

Es importante aclarar que se ha hablado de clave primaria y clave alternativa y en el ejemplo hemos mencionado a dos atributos, esto es porque la clave puede estar formada por varios atributos pero siempre será una sola clave. Una clave primaria es aquella que identifica cada tupla de una relación de manera mínima, es decir, podríamos encontrarnos por ejemplo, cuatro atributos de una relación que identifiquen perfectamente a cada tupla, pero si con solo tres de ellos también las podemos identificar, la clave estará formada por esos tres.

#### 4. Regla de integridad de la entidad.

Esta regla hace referencia a que ningún atributo que forme parte de la clave primaria puede tomar un valor nulo.

#### Restricciones de usuario.-

##### 1. Valores de uno o varios atributos que no pueden repetirse.

Supongamos que, aun con todas las aclaraciones hechas en el apartado de restricciones inherentes, el usuario decide que no puede haber dos artistas con el mismo nombre, esto haría que el atributo “Nombre” fuera único para cada tupla.

De aquí se deduce que toda clave primaria es única también, pero el lenguaje SQL mediante su definición de clave primaria ya lo contempla.

##### 2. Atributos que deben tener siempre valores para todas las tuplas de la relación.

Por ejemplo, y en el caso que nos ocupa, supongamos que no está permitido que un artista no tenga especialidad definida, esto significaría obligar a que el atributo especialidad tomara siempre valores, y el SQL lo reflejaría a través de “NOT NULL”.

El hecho de encontrarnos con atributos que no tengan valores es muy frecuente. Supongamos que en la relación ARTISTA queremos almacenar también la extensión de la

---

<sup>17</sup> En SQL se define UNIQUE

obra de cada autor. Es claro que en muchos casos este dato no se conocerá, por tanto tendríamos entradas en la tabla cuya celda correspondiente a la obra estaría vacía.

La obligatoriedad de valores en un atributo es también una propiedad de la clave primaria, por su propia definición.

### 3. Integridad Referencial.

Se podría decir que la integridad referencial trata la forma en la que los datos de dos o más tablas se deben relacionar para no atentar contra la integridad de la base de datos, es decir, para evitar que haya información no necesaria.

La integridad referencial se controla mediante lo que se denomina **clave ajena**. Una clave ajena de una tabla es un atributo o un conjunto de atributos, que es clave candidata<sup>18</sup> de otra tabla con la cual está relacionada la primera, en SQL se define como “FOREIGN KEY”. Para aclarar conceptos supongamos que se desean almacenar en la base de datos de nuestro ejemplo los premios que les han sido concedidos a los artistas por la Real Academia y, teniendo en cuenta que un premio es otorgado a un único artista. Esta relación PREMIO aparece en la figura 2.4. Se puede observar que posee un atributo “Artista” que almacena el artista al que se le ha concedido cada premio, y que coincide con la clave primaria de la relación ARTISTA (“Cod\_artista”); “Artista” es clave ajena de la relación PREMIO. Se representa mediante una flecha que sale de la clave ajena y apunta hacia la clave primaria de la tabla con la que se relaciona.

Existen situaciones en las que el control de la integridad referencial no es tan sencillo como en el caso anterior. Supongamos que se desean almacenar las Escuelas a las que pertenece cada artista y la relación que existe entre cada escuela y los artistas de nuestra base de datos, sabiendo que a una misma escuela pueden pertenecer varios artistas y que un artista puede haber pertenecido a escuelas diferentes (en distintos periodos de tiempo); en primer lugar debemos crear una relación ESCUELA cuyos atributos serán “Cod\_escuela” (que será la clave primaria de la relación) y “Nombre”. Para relacionar los artistas con las escuelas, en principio podemos pensar en varias soluciones:

Podríamos introducir el atributo Cod\_artista en la relación ESCUELA como clave ajena que llamara a la relación ARTISTA, pero pensemos en las filas de la tabla resultante. Por cada escuela solo tendríamos un artista que perteneciera a ella, puesto que el modelo relacional no

---

<sup>18</sup> Una clave candidata es aquel atributo o conjunto de atributos que pueden ser clave de una relación.

admite más de un valor en cada celda, lo que no es correcto. Podríamos pensar en el caso simétrico e introducir el atributo `Cod_escuela` en la relación `ARTISTA`. Esto significaría que cada artista solo puede pertenecer a una escuela, supuesto que va en contra de lo que se pide en el enunciado.

¿Qué hacer entonces?

El problema se resuelve introduciendo una nueva relación `PERTENECE` cuyos atributos sean “`Cod_artista`” y “`Cod_escuela`”. Además estos atributos formarán en conjunto la clave primaria y de esta forma tendremos a cada artista con las distintas escuelas a las que pertenece y cada escuela con todos sus artistas, como se puede ver en la figura 2.5. Podemos incluir dos nuevos atributos “`Fecha inicio`” y “`Fecha fin`”<sup>19</sup>, para recoger el periodo de tiempo en que cada artista pertenece a una escuela determinada. Por último, tanto “`Cod_artista`” como “`Cod_escuela`” por separado, son claves ajenas de las relaciones `ARTISTA` y `ESCUELA` respectivamente.

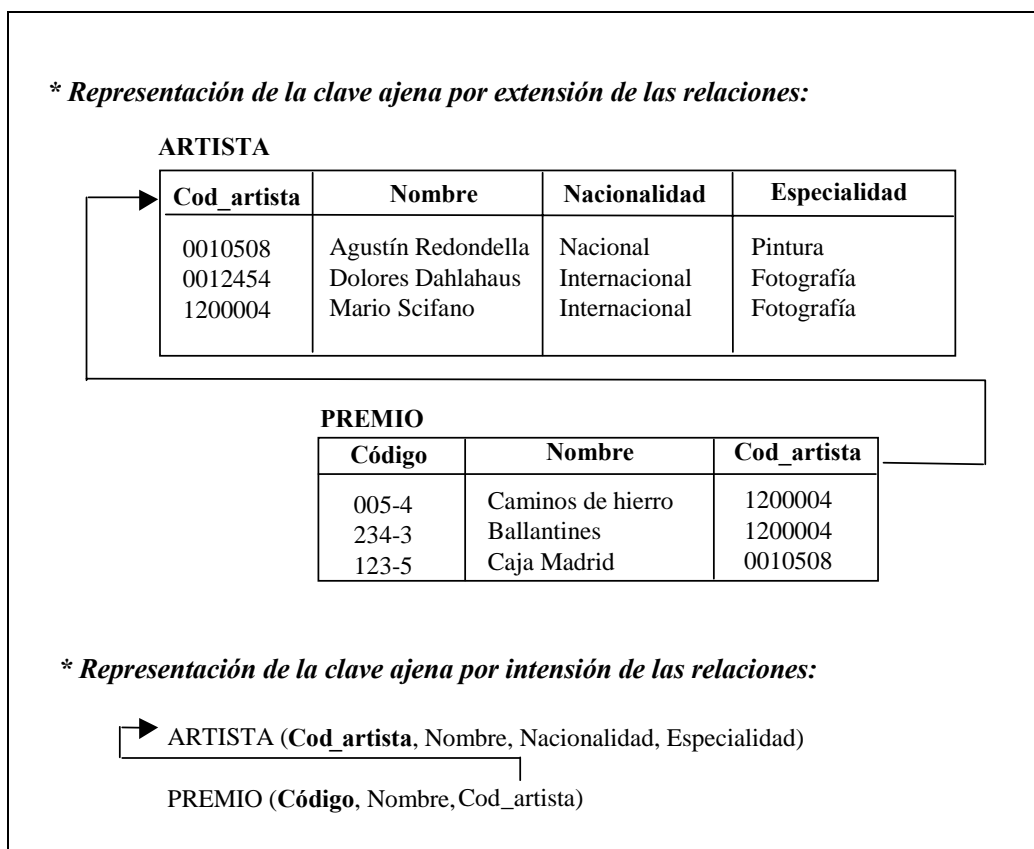
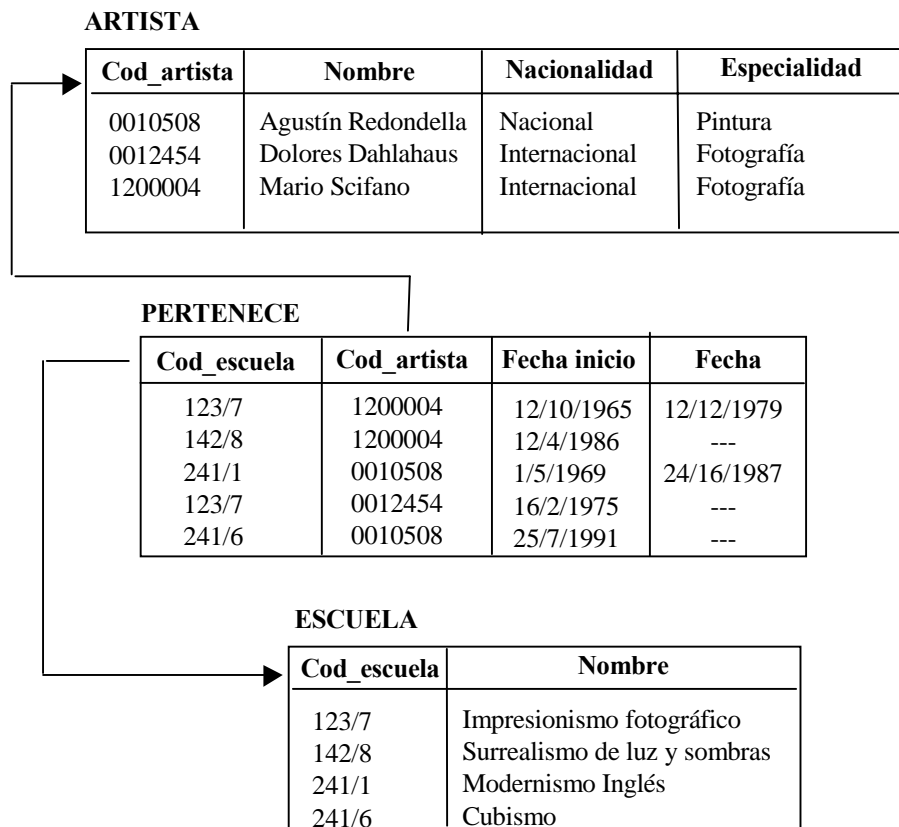


Figura 2.46: Ejemplo de utilización de clave ajena

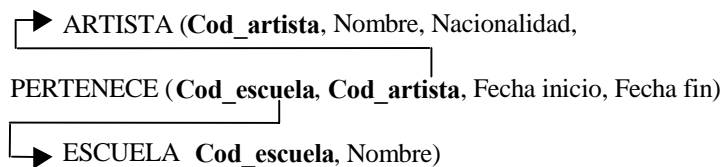
<sup>19</sup> Este atributo puede tomar valores nulos pues los artistas de una determinada escuela pueden seguir perteneciendo a ella en la actualidad.



**\* Representación de la clave ajena por extensión de las relaciones:**



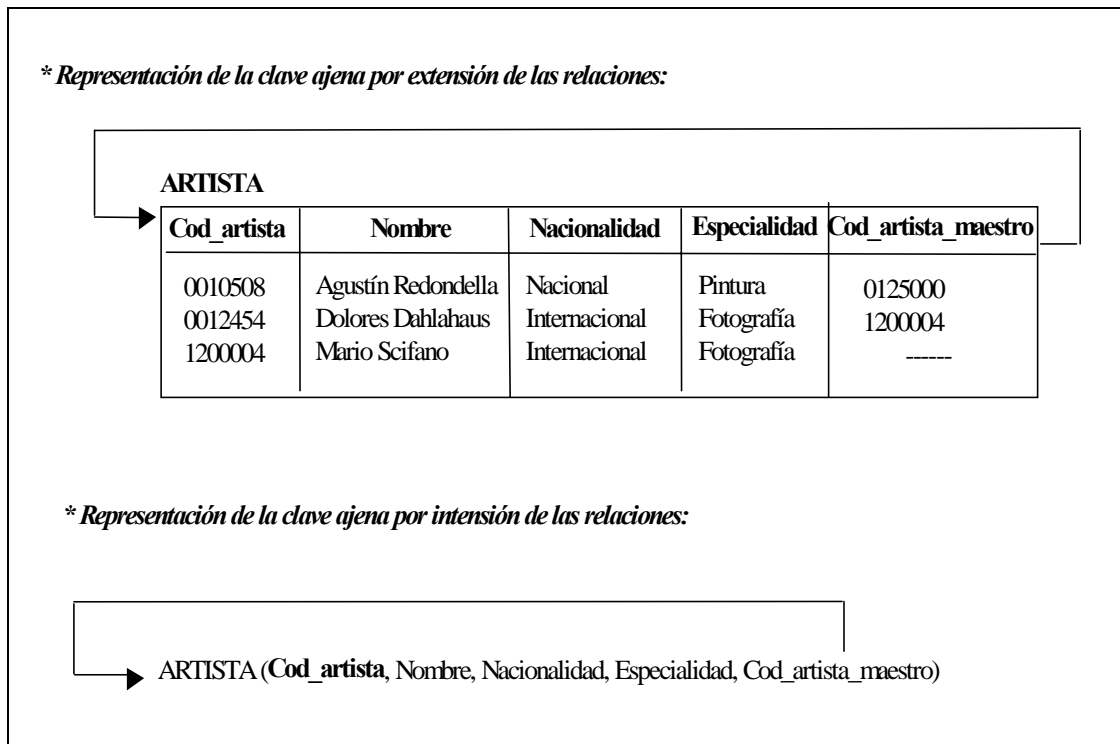
**\* Representación de la clave ajena por intensión de las relaciones:**



**Figura 2.47: Ejemplo de utilización de clave ajena**

Hasta el momento hemos visto como se pueden interrelacionar dos relaciones distintas mediante la definición de clave ajena, pero qué sucede en el caso de que una relación deba interrelacionarse con ella misma, esta situación, aunque en principio pueda parecer atípica, ocurre muchas veces en el mundo real. Supongamos por ejemplo, que en el caso de nuestros artistas, se desea recoger la relación que existe entre un artista y su maestro, en el caso de que exista; un maestro a su vez es un artista y por tanto tiene todas las cualidades (atributos) de ellos.

Podríamos introducir un atributo “Cod\_artista\_maestro” en la relación ARTISTA de manera que fuera clave ajena de ARTISTA que referencia a la misma relación ARTISTA, como se muestra en la figura 2.48.



**Figura 2.48: Ejemplo de utilización de clave ajena**

Como consecuencia de la introducción de la noción de clave ajena en las relaciones, se pueden definir ciertas operaciones de borrado y modificación que permiten mantener la integridad de los datos de nuestras bases de datos. Estas operaciones son las siguientes:

A) Operación restringida (“**NO ACTION**”<sup>20</sup>).

El borrado (DR) o la modificación (UR) de las filas de la relación que contiene la clave ajena no se permite mientras existan tuplas en la relación a la que se referencia. Esta situación ocurre por ejemplo en la relación ARTISTA con la clave ajena

“Cod\_artista\_maestro”. Solamente eliminamos un artista cuando no tenga ningún maestro, en caso contrario se rechaza la operación.

<sup>20</sup> También conocida como RESTRICT

También se podría dar este tipo de operación en el caso por ejemplo de que en nuestra base de datos de artistas se desee mantener siempre el histórico de todos los premios, en este caso no debemos permitir borrar ningún artista de la relación ARTISTA al que se le haya concedido algún premio, solo podremos eliminar a los artistas que no tienen premios.

B) Operación en cascada (“**CASCADE**”).

En este tipo de operación, cuando se elimina (DC) o modifica (UC) una tupla de la relación que es referenciada, los cambios se transmiten en cascada a las tuplas de la relación que contiene la clave ajena cuyos valores se han modificado. Así por ejemplo en el caso del apartado anterior, utilizando este tipo de operación, si borramos un artista se borran también todos los datos de su maestro, por lo que no es la mejor opción. En cambio, si utilizamos la operación “**CASCADE**” en la relación PERTENECE, cuando eliminemos un artista, se eliminará la relación que existe entre este y la escuela a la que pertenecía, pero no los datos propios de la escuela.

Estos dos tipos de operaciones sobre la clave ajena son los más habituales y los que implementan la mayoría de los SGBD comerciales. Existen otros dos tipos de operaciones (puesta a nulos “**SET NULL**” y valor por defecto “**SET DEFAULT**”) que por su complejidad y falta de uso en los SGBD no se van a definir aquí, no obstante hay una amplia bibliografía sobre el tema, la cual aparece al final del capítulo y que desde aquí animamos a que se consulte.

## **Autoevaluación: .- PARTE II.2: MODELO RELACIONAL. ESTÁTICA**

### **2.4. Dinámica del Modelo Relacional**

La parte dinámica del Modelo Relacional, al igual que la dinámica de cualquier modelo de datos, define las operaciones que se pueden hacer con la base de datos. Estas operaciones pueden ser de varios tipos:

A) Operaciones de **recuperación** de datos de la base de datos (consultas).

B) Operaciones de **actualización** de datos de la base de datos. Estas operaciones son:

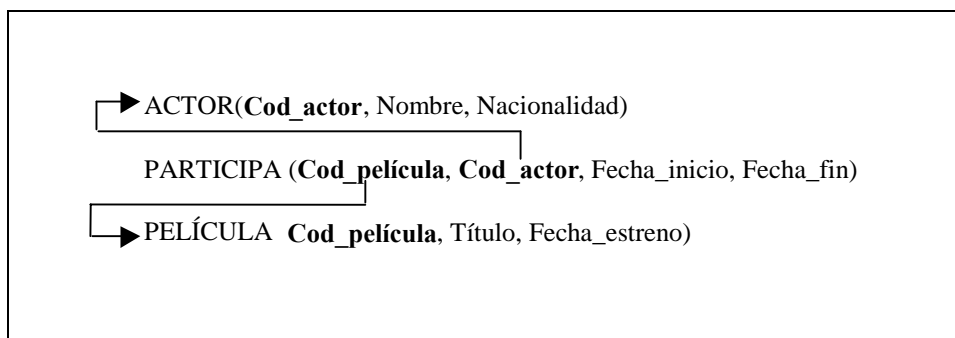
- Inserción de tuplas en la base de datos.
- Modificación de tuplas (de algunos de sus valores)

- Borrado de tuplas.

Estas operaciones se expresan mediante lenguajes de manipulación relacionales. Los lenguajes de manipulación pueden dividirse en lenguajes navegacionales, en los que se debe indicar el camino para llegar al dato, y lenguajes de especificación, en los cuales se recupera la información mediante la imposición de condiciones y no indicando el camino. El lenguaje de manipulación de datos del SQL (LMD) es un lenguaje de especificación, que consiste en un conjunto de sentencias que nos permiten manipular la base de datos.

En este apartado vamos, a través de un ejemplo, a introducir las sentencias básicas del SQL para la manipulación de una base de datos.

Supongamos una base de datos de un videoclub, en la que entre otros objetos se almacenan las películas y los actores que intervienen en ellas (Figura 2.49.).



**Figura 2.49: Subesquema de la base de datos de un videoclub**

Las operaciones que se pueden hacer sobre la base de datos son las siguientes:

- Altas INSERT INTO (Figura 2.50.)
- Bajas DELETE FROM (Figura 2.51.)
- Modificaciones UPDATE (Figura 2.52.)
- Consultas SELECT

```
INSERT INTO PELÍCULA
VALUES (1, 'Entrevista con un Vampiro', 1/5/1997)
```

**PELÍCULA**

Cod_película	Título	Fecha_estreno
1	Entrevista con un Vampiro	1/5/1997

**Figura 2.50: Ejemplo de inserción de tuplas**

Supongamos que tenemos la tabla PELÍCULA con las siguientes filas:

**PELÍCULA**

Cod_película	Título	Fecha_estreno
1	Entrevista con un Vampiro	1/5/1997
2	Abre los Ojos	7/3/1998

```
DELETE FROM PELÍCULA
VALUES (1, 'Entrevista con un Vampiro', 1/5/1997)
```

**PELÍCULA**

Cod_película	Título	Fecha_estreno
2	Abre los Ojos	7/3/1998

**Figura 2.51 : Ejemplo de borrado de tuplas**

Supongamos que queremos modificar la fecha de la película 'Abre los Ojos':

```
UPDATE PELÍCULA  
SET Fecha_estreno=25/3/1998  
WHERE Título ='Abre los Ojos'
```

**PELÍCULA**

Cod_película	Título	Fecha_estreno
1	Entrevista con un Vampiro	1/5/1997
2	Abre los Ojos	25/3/1998

**Figura 2.52.: Ejemplo de modificación de tuplas**

Mención especial debe hacerse a las consultas. El formato general de las consultas es:

SELECT-FROM-WHERE, donde la cláusula WHERE es opcional, dependerá de si imponemos alguna condición sobre las tuplas a seleccionar o bien de si necesitamos unir varias tablas para obtener un resultado. Las consultas las podemos dividir en consultas sobre una tabla y consultas sobre varias tablas.

Supongamos que tenemos la ejemplar del esquema relacional del videoclub que aparece en la Figura 2.53:

ACTOR			PARTICIPA			
Cod_actor	Nombre	Nacionalidad	Cod_actor	Cod_pelicula	Fecha_inicio	Fecha_fin
1	TomGuise	Norteamericana	1	1	1/1/1996	1/3/1996
2	Brad Pitt	Norteamericana	2	1	4/3/1996	30/12/1996
3	Penélope Cruz	Española	3	2	1/5/1997	31/10/1997
4	Javier Bardem	Española	4	2	1/5/1997	31/12/1997

PELÍCULA		
Cod_pelicula	Título	Fecha_estreno
1	Entrevista con un Vampiro	1/5/1997
2	Abr los Ojos	7/3/1998

Figura 2.53: Ejemplar del videoclub

## A) Consultas sobre una sola tabla:

En este caso la condición **WHERE** únicamente se utiliza en el caso de que queramos restringir el conjunto de filas que se recupere.

Como se puede apreciar en la Figura 2.54, la recuperación de las puede hacerse por el total de las columnas (atributos) de la tabla implicada en la consulta, o bien seleccionando aquellos atributos que queramos obtener en la consulta.

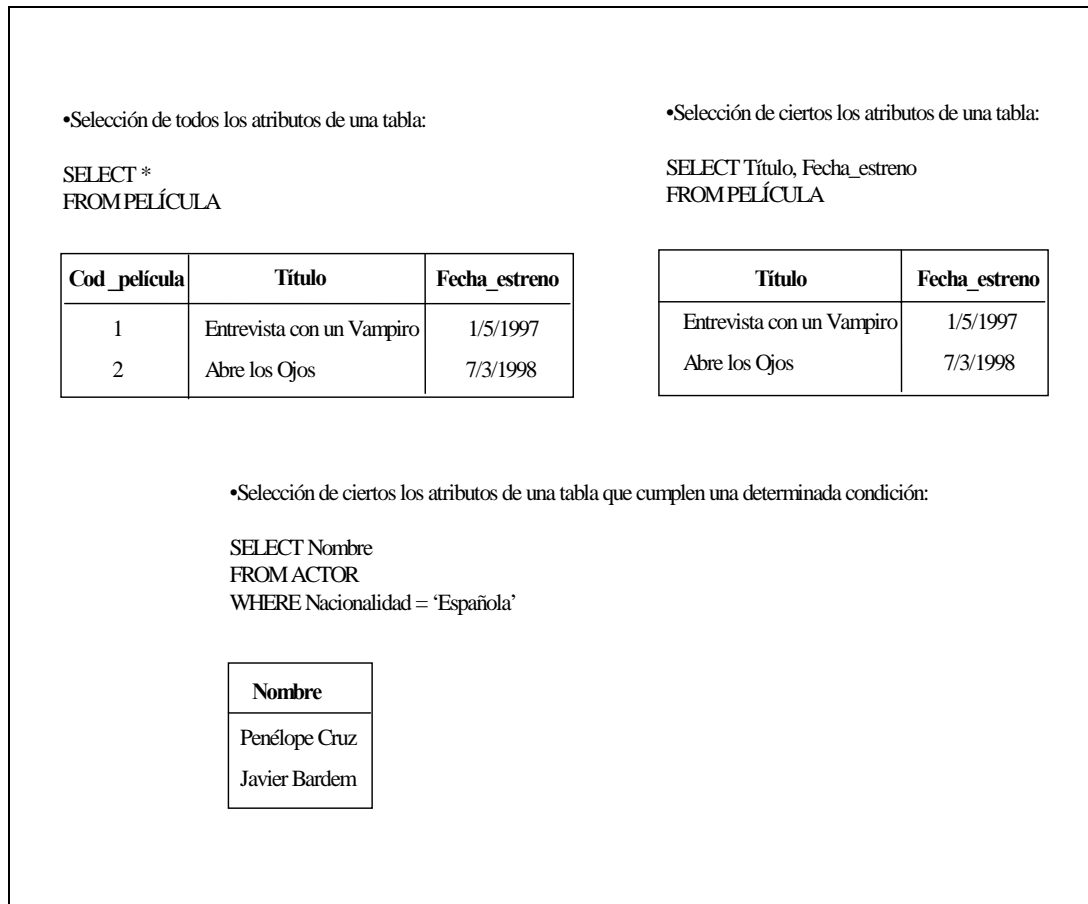


Figura 2.54: Consultas sobre una única tabla

## B) Consultas sobre varias tablas:

En este caso la condición **WHERE** no es opcional. Se debe utilizar para unir las tablas<sup>21</sup> que intervienen en la consulta. Esta unión se realiza por la clave ajena de una tabla y la clave primaria de la tabla a la que referencia, como se muestra en la figura 2.55.

<sup>21</sup> Esta unión es lo que se denomina **JOIN**.



- Selección de los actores y las películas en las que intervienen:

```
SELECT Nombre,
FROM PELÍCULA, ACTOR, PARTICIPA
WHERE PELÍCULA.Cod_película=PARTICIPACod_película AND ACTOR.Cod_actor=PARTICIPACod_actor
```

Nombre	Título
Tom Guise	Entrevista con un Vampiro
Brad Pitt	Entrevista con un Vampiro
Penélope Cruz	Abre los Ojos
Javier Bardem	Abre los Ojos

- Selección de los actores que intervienen en la película 'Entrevista con un Vampiro':

```
SELECT Nombre
FROM PELÍCULA, ACTOR, PARTICIPA
WHERE PELÍCULA.Cod_película=PARTICIPACod_película AND ACTOR.Cod_actor=PARTICIPACod_actor AND
Título='Entrevista con un Vampiro'
```

Nombre
Tom Guise
Brad Pitt
Penélope Cruz
Javier Bardem

**Figura 2.55.: Ejemplo de consultas en las que intervienen varias tablas**

Con lo explicado hasta ahora de la parte dinámica del Modelo Relacional hemos pretendido dar una visión general de las operaciones que se pueden realizar sobre una base de datos, no obstante existen otras muchas funciones que se pueden aplicar sobre los atributos de una o varias tablas para recuperar información y que pueden ser consultadas en la bibliografía que aparece al final del capítulo.

**Autoevaluación: .- PARTE II.2: MODELO RELACIONAL. ESTÁTICA**

## 2.5. Caso práctico

Para fijar los conocimientos del modelo relacional asimilados hasta el momento, vamos a diseñar una base de datos completa y a hacer ciertas consultas sobre ella.

“ Antonio es uno de tantos trabajadores que llega a casa cansado y estresado. Desde hace algún tiempo alivia su estrés elaborando recetas de cocina que luego prueban sus familiares y amigos. Para ello se apoya en los múltiples manuales de cocina que ha ido coleccionando e incluyendo en su biblioteca desde que se inició en el arte culinario, y luego confecciona sus propias recetas. Tanto ha sido su interés que en la actualidad posee un gran número de recetas nuevas a modo de apuntes, imposible de organizar de una manera efectiva.

Los amigos de Antonio, en agradecimiento a tantas veladas de buena comida, vino y compañía han decidido agradecerle con una base de datos que le gestione su maravilloso gran hobby. Para ello deben controlar todas las recetas que posee, teniendo en cuenta que:

Cada receta proviene de una idea original de un libro de cocina de la biblioteca de Antonio, y se desea almacenar su origen.

Cada receta tiene un tipo (Sopas, Verduras, Carnes,...) e incorpora unos ingredientes de los que se desea saber su nombre y cantidad. Además cada receta contiene una breve explicación de cómo mezclar los ingredientes y obtener el producto final y el título de la receta original de la que proviene. Es bien sabido por los amigos de Antonio que de cada receta que él prueba, incorporando ciertos cambios, consigue nuevas recetas, por lo que sería interesante almacenar si cada receta es idea surgida de una receta original, o si por el contrario, proviene de una receta elaborada alguna vez ya por Antonio.

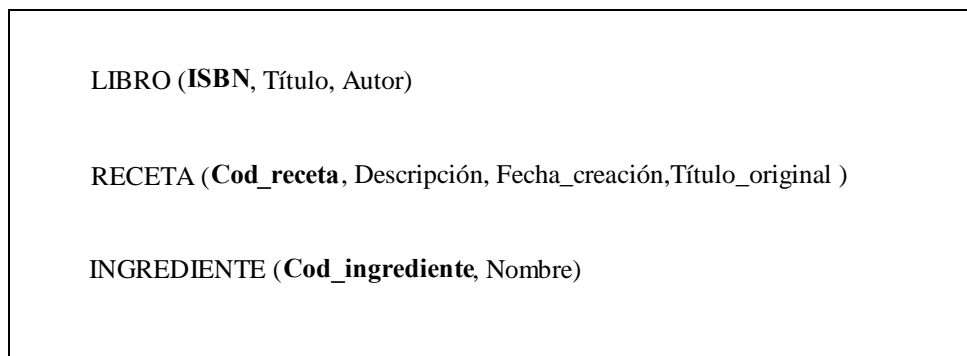
Lo que los amigos de Antonio quieren es que cuando él quiera pueda consultar las recetas por tipo y por ingrediente además de poder localizar el libro que le dio la idea de cada una de sus recetas. También sería interesante saber qué receta proviene de alguna otra y cual no.

### **Solución.**

En primer lugar debemos detectar cuáles son las relaciones base que aparecen en el enunciado y qué atributos contienen (Figura 2. 56.). Estos son:

LIBRO. Esta relación debe contener al menos el Título de la obra y el autor. Como clave primaria podemos suponer el ISBN, que como todos sabemos es único por cada libro editado.

- a) RECETA. De cada receta deberemos incluir el tipo , la descripción, el título original (título de la receta en la que se basa) y la fecha de creación. Además supondremos el atributo “Cod\_receta” como clave primaria.
- b) INGREDIENTE. En principio, de los ingredientes se desea almacenar el nombre y la cantidad que de él se ha empleado en cada receta, pero como un mismo ingrediente puede ser usado en distintas cantidades para varias recetas, de momento solo introduciremos el nombre como atributo de la relación INGREDIENTE y veremos posteriormente como reflejar la cantidad. Al igual que ocurría en el caso de las recetas, consideraremos el atributo “Cod\_ingrediente” como clave primaria.

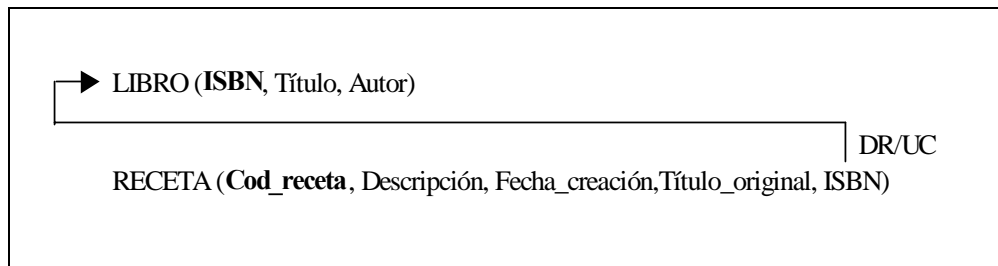


**Figura 2. 56: Relaciones elementales de las base de datos de Antonio**

Una vez encontradas las relaciones base hay que analizar como se intrrelacionan en función de los supuestos realizados en el enunciado:

- De cada receta se desea saber de donde (de qué libro se ha sacado). Por lo tanto deberemos introducir en la relación RECETA el ISBN del libro. Parece claro que no pueda ser de otra manera pues si pensásemos en introducir el código de receta en LIBRO, lo que significaría es que por cada libro Antonio sólo sacó una receta (recuérdese que en el modelo relacional no puede haber más de un valor en cada celda de una tabla). El ISBN introducido en la relación RECETA es clave ajena de ésta y referencia a la relación LIBRO.

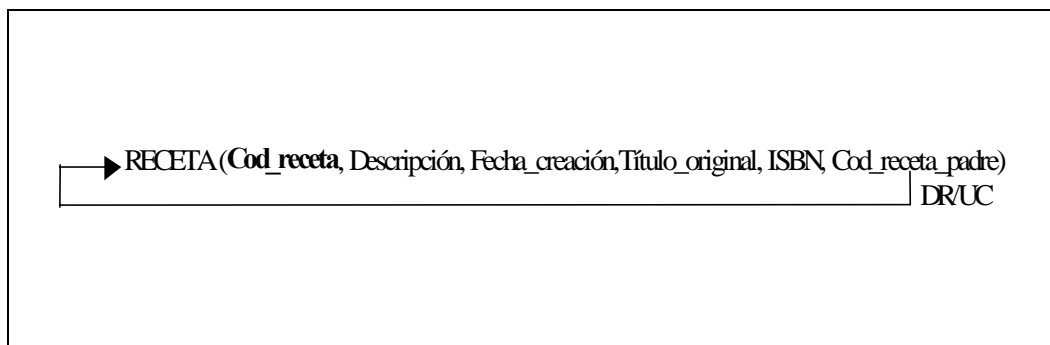
En cuanto a las opciones de borrado y modificación, el borrado debe ser restringido (DR), pues si eliminamos un libro no se debe permitir que se eliminen todas las recetas que Antonio sacó de él; la modificación puede ser en cascada (UC) para que los cambios se actualicen automáticamente (Figura 2. 57).



**Figura 2.57: Asociación entre LIBRO y RECETA**

- Cada receta puede ser original, es decir sacada directamente de un libro de cocina, o bien ser una variación de una ya creada y este aspecto se desea recoger en la base de datos. Independientemente de que la receta sea original o variación, es una receta y por tanto contendrá los mismos atributos que cualquier receta; si duplicásemos la relación RECETA y mantuviésemos una para las recetas originales y otra para las variaciones, provocaríamos redundancia, pues la definición de la relación RECETA se haría dos veces. Lo mejor en estos casos es añadir un atributo “Cod\_receta\_padre” en RECETA, que, en el caso de contener valor, nos dirá de qué otra receta proviene (si no contiene ningún valor es porque la receta correspondiente a la fila en la que este atributo no tiene valor es una receta original).

“Cod\_receta\_padre” es una clave ajena de la relación RECETA que se referencia a sí misma. Si eliminamos una receta que es una variación de otra, no debemos permitir que se borre la receta “padre”, por lo que el borrado debemos considerarlo restringido (DR), en cambio la modificación de una receta “variación”, puede hacerse en cascada pues no va a afectar a la receta padre (Figura 2. 58).



**Figura 2. 58: Asociación entre RECETA original y no original**

- Cada receta está compuesta por ciertos ingredientes (uno o varios) y además un mismo ingrediente, sin tener en cuenta las cantidades del mismo, puede utilizarse en serlo de varias recetas, por tanto debemos introducir una nueva relación COMPUESTA cuyos atributos

serán “Cod\_receta”, “Cod\_ingredient” y “Cantidad”. La clave primaria estará formada por los dos primeros atributos y de esta forma tendremos qué cantidad de un mismo ingrediente se ha utilizado en cada receta y todas las cantidades de cada uno de los ingredientes que intervienen en una misma receta, como puede verse en la Figura 2. 59.

“Cod\_receta” y “Cod\_ingredient” son, respectivamente, claves ajenas de las relaciones RECETA e INGREDIENTE. Los borrados y modificaciones de ambas claves ajenas pueden hacerse en cascada, pues a lo único que afectaría sería a la relación existente entre la receta y sus ingredientes, pero no a la receta o al ingrediente en sí.

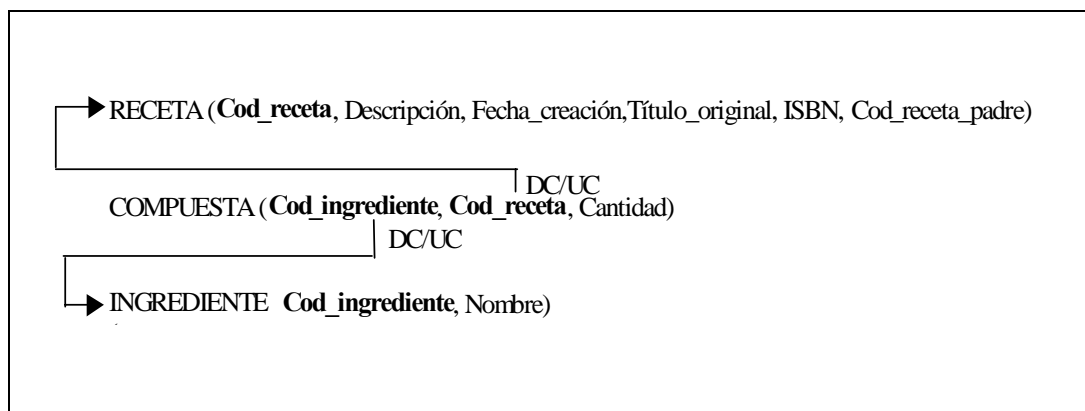


Figura 2. 59: Asociación entre RECETA e INGREDIENTE

Tras haber analizado todos los supuestos del enunciado el esquema relacional correspondiente a la base de datos de recetas de cocina, queda como se refleja en la Figura 2. 60.

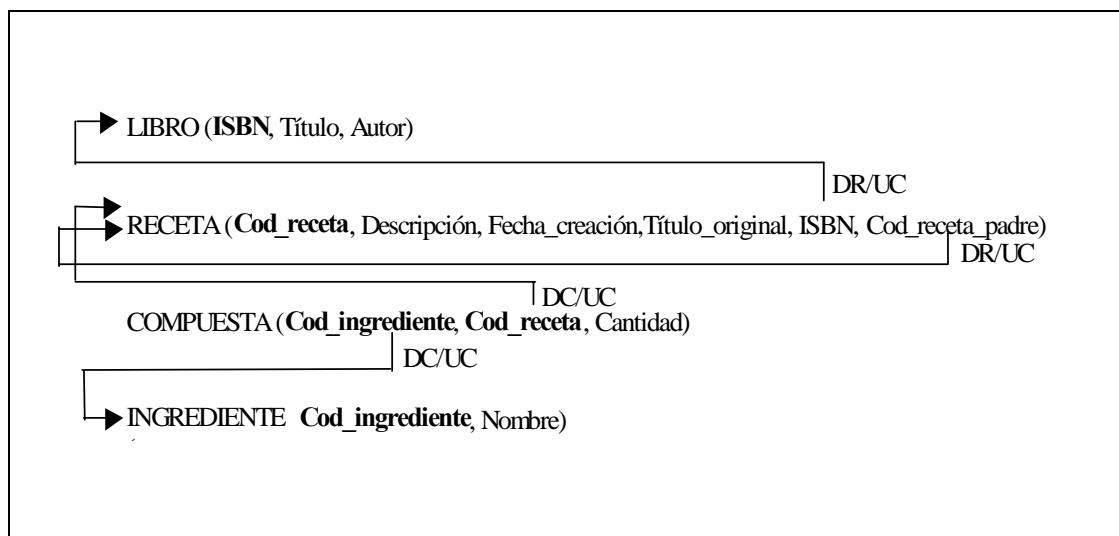


Figura 2.50: Esquema relacional de las Recetas de Cocina

Una vez realizado el diseño de la base de datos hay que gestionarla, es decir, insertar datos, modificar datos, y realizar consultas a la base de datos. Supongamos la siguiente ejemplar del esquema:

## LIBRO

ISBN	Título	Autor
84-404-9666-4	Guía Práctica de Cocina Oriental	K.C.Chin
84-324-8776-3	La Cocina Para Estar en Forma	Karlos Arguiñano
84-403-6999-5	La Cocina Práctica	Karlos Arguiñano

## RECETA

Cod_receta	Tipo	Descripción	Fecha creación	Título original	ISBN	Cod_receta_padre
7	Pasta	La pasta se cuece y se saltea con un poco de aceite. Mezclar la nata con las yemas de huevo batidas e incorporar a la pasta (a fuego lento). Fuera del fuego se le añade queso rallado y se gratina (2 min.)	8/3/1999	Cintas a la Crema	84-324-8776-3	3
8	Pasta	Rehogar mantequilla, anchoas y aceitunas. Añadir la pasta cocida y el brecol. Lonchas de queso por encima y gratinar (1,5 min.).	13/6/1999	Cintas con Brecol	84-324-8776-3	-----
9	Pasta	Rehogar la verdura con el ajo. Incorporar la pasta cocida con las anchoas y las aceitunas. Colocar las lonchas de queso y gratinar (3 min.)	26/6/1999	Cintas con Brecol	84-403-6999-5	8
10	Verduras	Freír los aros de pimiento y colocar encima los puerros cocidos. Cubrir con salsa bechamel. Espolvorear con queso rallado y gratinar (5 min.)	30/6/1999	Puerros con bechamel	84-324-8776-3	-----

## COMPUESTA

## INGREDIENTE

Cod_ingredient	Nombre
4	Cintas
5	Yemas de huevo
6	Queso rallado
7	Nata líquida
8	Brecol
9	Anchoas
10	Aceitunas negras
11	Lonchas de queso
12	Coliflor
13	Espinacas
14	Puerros
15	Zanahorias
16	Judías verdes
17	Ajos
18	Pimientos
19	Bechamel

Cod_receta	Cod_ingredient	Cantidad
7	4	300 grs.
7	5	4
7	6	---
7	7	1 vasito
8	4	200 grs.
8	8	250 grs.
8	9	8
8	10	1 puñado
8	11	4
9	4	200 grs.
9	8	250 grs.
9	9	8
9	10	1 puñado
9	11	4
9	12	200 grs.
9	13	200 grs.
9	14	100 grs.
9	15	100 grs.
9	16	100 grs.
9	17	1 diente
10	14	6
10	18	6 medianos
10	19	1/2 litro
10	6	50 grs.

Supongamos que queremos realizar las siguientes consultas;

1) Consultas de selección

- Libros de cocina de la biblioteca de Antonio.
- Nombre y descripción de las recetas del tipo pasta realizadas en junio de 1999.

*Libros de cocina de la biblioteca de Antonio*

**SELECT \***

**FROM LIBRO**

*Resultado:*

LIBRO

ISBN	Título	Autor
84-404-9666-4	Guía Práctica de Cocina Oriental	K.C.Chin
84-324-8776-3	La Cocina Para Estar en Forma	Karlos Arguiñano
84-403-6999-5	La Cocina Práctica	Karlos Arguiñano

*Nombre y descripción de las recetas del tipo pasta realizadas en junio de 1999*

**SELECT** Descripción, Título\_original

**FROM** RECETA

**WHERE** Tipo = 'Pasta' AND Fecha\_creación >= 1/6/1999

*Resultado:*

RECETA

Descripción	Título_original
Rehogar mantequilla, anchoas y aceitunas. Añadir la pasta cocida y ebrecol. Lonchas de queso por encima y gratinar (1,5min).	Cintas con Brecol
Rehogar la verdura con el ajo. Incorporar la pasta cocida con las anchoas y las aceitunas. Colocar las lonchas de queso y gratinar (3min)	Cintas con Brecol

Consultas de unión:

- Recetas en las que intervienen las ‘Lonchas de queso’
- Título de los libros , descripción y título original de las recetas de tipo ‘Verduras’
- Ingredientes y cantidad de los mismos utilizados en las recetas anteriores a junio de 1999.

Recetas en las que intervienen las ‘Lonchas de queso’

SELECT Tipo, Descripción, Fecha\_creación, Título\_original, Cantidad

FROM RECETA, COMPUESTA, INGREDIENTE

WHERE Nombre = ‘Lonchas de queso’ AND INGREDIENTE.Cod\_ingrediente = COMPUESTA.Cod\_ingrediente AND COMPUESTA.Cod\_receta = RECETA.Cod\_receta

Resultado:

Tipo	Descripción	Fecha creación	Título original	Cantida
Pasta	Rehogar mantequilla, anchoas y aceitunas. Añadir pasta cocida y el brecol Lonchas de queso encima y gratinar (1,5min).	13/6/1999	Cintas con Brecol	4
Pasta	Rehogar la verdura con el ajo. Incorporar la pasta cocida con las anchoas y las aceitunas. Colocar lonchas de queso y gratinar (3 min.)	26/6/1999	Cintas con Brecol	4

Título de los libros , descripción y título original de las recetas de tipo ‘Verduras’

SELECT Título, Descripción, Título\_original

FROM RECETA, LIBRO

WHERE Tipo = ‘Verduras’ AND RECETA.ISBN = LIBRO.ISBN

Resultado:

Título	Descripción	Título original
La Cocina Para Estar en Forma	Freir los aros de pimiento y colocar encima los puerros cocidos. Cubrir con salsa bechamel. Espolvorear con queso rallado y gratinar (5 min.)	Puerros con bechamel



*Ingredientes y cantidad de los mismos utilizados en las recetas anteriores a junio de 1999*

**SELECT** Ingrediente, Cantidad

**FROM** RECETA, COMPUESTA, INGREDIENTE

**WHERE** Fecha\_creación < '1/6/1999' AND IGREDIENTE.Cod\_ingrediente = COMPUESTA.Cod\_ingrediente AND COMPUESTA.Cod\_receta = RECETA.Cod\_receta

*Resultado:*

Nombre	Cantidad
Cintas	300 grs .
Yemas de huevo	4
Queso rallado	----
Nata líquida	1 vasito

## Caso Práctico Propuesto: .- PARTE II.2: MODELO RELACIONAL

### 2.6 Repaso de Consultas en SQL

La sintaxis de las consultas en SQL es:

```
SELECT [(*)|ALL|DISTINCT| <columna [, columna]...>)| <expresión de
función>]
FROM <nombre_tabla> [[, nombre_tabla]...]
WHERE <condición de búsqueda> |
[GROUP BY <condición de búsqueda>]
HAVING <condición de búsqueda> |
ORDER BY <lista de atributos> [ASC|DESC] |
<columna> <operador> <sentencia de consulta>
```

donde:

ALL: selecciona todas las columnas

DISTINCT: suprime filas duplicadas

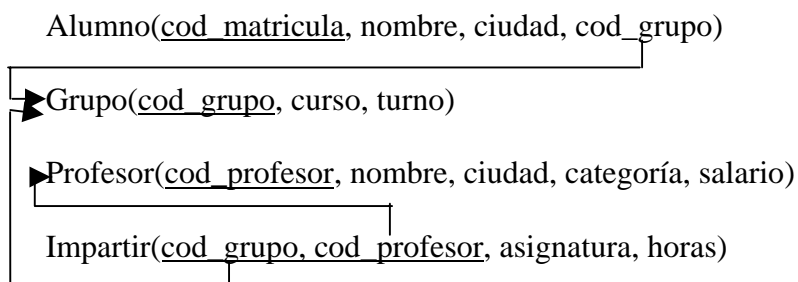
Operadores para la condición de búsqueda: <, >, =, >=, <=, between, like, in,...  
y también los operadores lógicos: NOT, AND, OR.

Presentaremos en este apartado varios ejemplos de bases de datos con las tablas y sus valores. En estos ejemplos iremos aplicando distintas consultas de menor a mayor dificultad con el fin de que mediante estos ejemplos se aprenda las distintas posibilidades que ofrece SQL para realizar consultas a una base de datos.

## BD SOBRE LAS QUE SE MUESTRAN LAS CONSULTAS

### UNIVERSIDAD

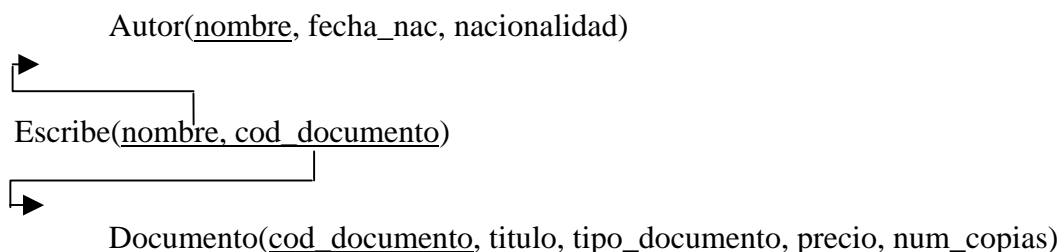
Tablas:



Base de datos en la que almacenamos los alumnos, los grupos donde se han matriculado estos, los profesores y los grupos donde imparten clase indicando la asignatura que imparten.

### BIBLIOTECA

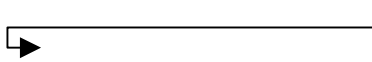
Tablas:



Almacenaremos la información relativa a documentos con los autores que los han escrito.

**EMPRESA**

Tablas:

Empleado(nombre, numero\_dept, salario, fecha\_nac, ext\_telefónica)Departamento(numero\_dept, nombre)

En esta base de datos contemplamos la información de una empresa con datos de los empleados y los departamentos a los que pertenecen dichos empleados.

**VALORES PARA LAS TABLAS DE LAS BD QUE NOS SERVIRÁN DE EJEMPLO.****1. UNIVERSIDAD**

Alumno

cod_matricula	Nombre	Ciudad	cod_grupo
101	Juan Montero	Alcorcón	11
102	Alicia Cristobal	Leganés	11
202	Ana Vallejo	Leganés	21
300	Ignacio López	Móstoles	31
103	Leticia Martínez	Alcorcón	--

Grupo

Cod_grupo	Curso	Turno
11	1	M
12	1	T
21	2	M
22	2	T
31	3	T

## Profesor

<b>Cod_profesor</b>	<b>Nombre</b>	<b>Ciudad</b>	<b>Categoría</b>	<b>Salario</b>
1p	D. Cuadra	Madrid	T1	200.000
2p	E. Nieto	Las Rozas	T2	250.000
3p	P. Martínez	Alcorcón	T1	225.000
4p	C. Nieto	Madrid	T2	150.000
5p	A. Sierra	Madrid	T3	120.000
6p	C. García	Madrid	T1	135.000
7p	J. Montero	Madrid	T3	125.000

## Impartir

<b>Cod_grupo</b>	<b>Cod_profesor</b>	<b>Asignatura</b>	<b>Horas</b>
11	1p	Intr. Informática	20
11	2p	SGBD	15
21	1p	Ficheros y BD	12
31	2p	Diseño de BD	20

## 2. BIBLIOTECA

## Autor

<b>Nombre</b>	<b>Fecha_nac</b>	<b>Nacionalidad</b>
Miguel de Cervantes	9-10-1547	Española
Emily Bronte	2-9-1818	Inglesa
Isaac Asimov	23-4-1930	Americana
Christian Jacq	30-6-1947	Francesa
Ken Follet	5-8-1949	Inglesa
Paloma Martínez	2-9-68	Española
P. Isasi	3-4- 66	Española
D. Borrajo	23-8-65	Española

## Documento

<b>Cod_documento</b>	<b>Título</b>	<b>Tipo_documento</b>	<b>Precio</b>	<b>Num_copias</b>
11	El Quijote	Novela	5000	22
12	Cumbres Borrascosas	Novela	4000	12
13	La Pirámide Asesinada	Novela	2000	15
14	La Ley del Desierto	Novela	2000	10
15	Introducción a la ciencia	Divulgativo	4500	13
16	Los Pilares de la Tierra	Novela	6000	7
17	Gramáticas, Lenguajes y Autómatas	Técnico	3500	6

## Escribe

<b>Nombre</b>	<b>Cod_documento</b>
Miguel de Cervantes	11
Emily Bronte	12
Isaac Asimov	15
Christian Jacq	13
Christian Jacq	14
Ken Follet	16
Paloma Martínez	17
P. Isasi	17
D. Borrajo	17

**3. EMPRESA**

Empleado

Nombre	Numero_dept	Salario	Fecha_nac	Ext_telefónica
Pablo Montero	14	220.000	10-11-67	6543
Beatriz Cristobal	13	300.000	20-9-68	6577
J.Luis Martín	11	150.000	25-6-77	6433
Almudena López	13	350.000	4-5-60	6422
Angel Vallejo	14	400.000	15-4-72	6321
Pedro García	11	200.000	12-3-70	6323

Departamento

Numero_dept	Nombre
11	Contabilidad
13	Marketing
14	Informática

**CONSULTAS SENCILLAS (SOBRE UNA SOLA TABLA)****A. Selección de columnas**

```
SELECT (<columna [, columna]...>|*)
FROM <nombre_tabla>
```

En la expresión del SELECT también pueden aparecer expresiones aritméticas (que dan lugar a columnas derivadas) con suma, resta, multiplicación y división con las prioridades habituales de la aritmética (izquierda a derecha, paréntesis, etc.)

**Ejemplo 1:** calcular la subida en un 15% en el precio de los documentos.

```
SELECT titulo, tipo_documento, precio* 1,15
FROM documento
```

Resultado:

Título	Tipo_documento	Precio *1,15
El Quijote	Novela	5750
Cumbres Borrascosas	Novela	4600
La Pirámide Asesinada	Novela	2300
La Ley del Desierto	Novela	2300
Introducción a la ciencia	Divulgativo	5175
Los Pilares de la Tierra	Novela	6900
Gramáticas, Lenguajes y Autómatas	Técnico	4025

También pueden utilizarse expresiones alfanuméricas: lower (para poner los valores en minúsculas), upper (para ponerlos en mayúsculas), substr (extraer una subcadena de una cadena de caracteres), + (operador de concatenación), length (calcula la longitud de una cadena de caracteres), etc.

#### B. Ordenando filas (ORDER BY)

Las filas se almacenan en el mismo orden en que han sido insertadas. Para obtenerlas ordenadas se aplica ORDER BY. Se puede especificar ASC o DESC (por omisión es ASC).

**Ejemplo 2:** seleccionar el nombre, la nacionalidad y la fecha de nacimiento de todos los autores ordenada por nacionalidad.

```
SELECT nombre, nacionalidad, fecha_nac
FROM autor
ORDER BY nacionalidad
```

Resultado:

Nombre	Nacionalidad	Fecha_nac
Isaac Asimov	Americana	23-4-1930
Miguel de Cervantes	Española	9-10-1547
Paloma Martínez	Española	2-9-68
P. Isasi	Española	3-4- 66
D. Borrajo	Española	23-8-65
Christian Jacq	Francesa	30-6-1947
Emily Bronte	Inglesa	2-9-1818
Ken Follet	Inglesa	5-8-1949

### C. Con restricción (seleccionando filas, WHERE)

Hasta ahora se recuperaban todas las filas. Si se quiere restringir las filas seleccionadas se utiliza WHERE con las condiciones que deben cumplir las filas para ser visualizadas.

Consulta simple: La condición está compuesta por un operador <, >, =, >=, <=, <>, que unen:

Dos columnas de la tabla

Una columna y una constante (numérica, carácter, fecha)

Las columnas que se referencian en la condición no tienen porque ser seleccionadas en el SELECT.

**Ejemplo 3:** seleccionar los autores cuya nacionalidad sea española.

```
SELECT nombre, nacionalidad, fecha_nac
FROM autor
WHERE nacionalidad = 'española'
```

Resultado:

Nombre	Nacionalidad	Fecha_nac
Miguel de Cervantes	Española	9-10-1547
Paloma Martínez	Española	2-9-68
P. Isasi	Española	3-4- 66
D. Borrajo	Española	23-8-65



Consulta compleja: con operadores lógicos: NOT, AND, OR, IN, BETWEEN AND, LIKE, NULL, NOT NULL. ¡¡Cuidado con las prioridades!!

**Ejemplo 4:** seleccionar aquellos autores que no sean alemanes nacidos después de la segunda guerra mundial.

```
SELECT nombre
FROM autor
WHERE nacionalidad <> 'alemana' AND fecha_nac > '02/09/1945'
```

Resultado:

Nombre
Christian Jacq
Ken Follet
Paloma Martínez
P. Isasi
D. Borrajo

**Ejemplo 5:** alumnos que viven en Getafe o Leganés

```
SELECT *
FROM alumno
WHERE ciudad IN ('Getafe', 'Leganés')
```

Resultado:

cod_matricula	nombre	ciudad	cod_grupo
102	Alicia Cristobal	Leganés	11
202	Ana Vallejo	Leganés	21

**Ejemplo 6:** alumnos cuyos números de matricula estén entre 200 y 400 (ambos inclusive).

```
SELECT *
FROM alumno
WHERE cod_matricula BETWEEN 200 AND 400
```

Resultado:

<b>cod_matricula</b>	<b>nombre</b>	<b>ciudad</b>	<b>cod_grupo</b>
202	Ana Vallejo	Leganés	21
300	Ignacio López	Móstoles	31

**Ejemplo 7:** alumnos que no están asignados a un grupo

```
SELECT *
FROM alumno
WHERE cod_grupo IS NULL
```

Resultado:

<b>cod_matricula</b>	<b>nombre</b>	<b>ciudad</b>	<b>cod_grupo</b>
103	Leticia Martínez	Alcorcón	--

**Ejemplo 8:** autores que tienen nacionalidad

```
SELECT nombre
FROM autor
WHERE nacionalidad IS NOT NULL
```

Resultado:

<b>Nombre</b>
Miguel de Cervantes
Emily Bronte
Isaac Asimov
Christian Jacq
Ken Follet
Paloma Martínez
P. Isasi
D. Borrajo

Cuando no conocemos el conjunto exacto de caracteres que forman la constante para indicar la restricción se utiliza LIKE con unos caracteres especiales \_ (un carácter) y % (cualquier número de caracteres)

**Ejemplo 9:** autores cuyo nombre empiece por I

```

SELECT *
FROM autor
WHERE nombre LIKE 'I%'

```

Resultado:

Nombre	Fecha_nac	Nacionalidad
Isaac Asimov	23-4-1930	Americana

Se utilizará DISTINCT si no se quiere que aparezcan filas recuperadas iguales.

**Ejemplo 10:** seleccionar las distintas nacionalidades que existen en la tabla autores.

```

SELECT DISTINCT nacionalidad
FROM autor

```

Resultado:

Nacionalidad
Española
Inglesa
Americana
Francesa

**D. Agrupando filas (GROUP BY y HAVING)**

Hasta ahora todas las consultas vistas devolvían un conjunto de filas que cumplían una serie de condiciones. Es posible agrupar las filas por determinados grupos de valores y aplicar funciones a cada grupo.

A las funciones que en vez de actuar sobre una sola fila actúan sobre un grupo de filas se las denomina funciones de grupo y devuelven un solo valor por cada grupo. El argumento de estas funciones son los valores pertenecientes a una o más columnas. Se pueden aplicar a uno o más grupos de filas de tabla (dependiendo de si se utiliza el GROUP BY o no).

Todas las funciones excluyen los valores nulos excepto COUNT(\*)

- AVG: calcula la media de los valores de la colección
- MAX: calcula el máximo.
- MIN: calcula el mínimo
- SUM: calcula la suma de los valores de la colección
- COUNT(\*): halla cuántos valores hay en la colección incluyendo las nulas
- COUNT(columna): cuenta el número de filas que tienen valor distinto de nulo en la columna de cada grupo
- COUNT(DISTINCT columna): cuenta el número de filas distintas que tienen valor distinto de nulo en la columna de cada grupo.

**Ejemplo 11:** hallar el número de empleados de la empresa.

```
SELECT COUNT (*) "Número de Empleados"  
FROM empleado
```

Resultado:

Número de Empleados

6

**Ejemplo 12:** hallar el salario medio, el mínimo, el máximo de todos los empleados

```
SELECT AVG(salario) "Media", MIN(salario) "Mínimo", MAX(salario) "Máximo"  
FROM empleado
```

Resultado:

MediaMínimoMáximo

270.000150.000400.000

**Ejemplo 13:** hallar el número de empleados y de extensiones telefónicas del departamento 14.

```
SELECT COUNT(*), COUNT(DISTINCT ext_telefonica)
FROM empleado
WHERE numero_dept=14.
```

Resultado:

2

2

**Ejemplo 14:** ¿cuántos empleados que han nacido antes de 1970?

```
SELECT COUNT(*) "Número de Empleados"
FROM empleado
WHERE fecha_nac < '01/01/1970'
```

Resultado:

Número de Empleados

3

El grupo por defecto es la tabla entera. Si se quiere dividir la tabla en grupos se utiliza GROUP BY con la que se especifican la(s) columna(s) por las que se quiere agrupar.

**Ejemplo 15:** calcular la suma de los precios y el precio medio de los documentos agrupados por tipo de documento (novela, divulgación, ...)

```
SELECT SUM(precio), AVG(precio)
FROM documento
GROUP BY tipo_documento
```

Resultado:

Tipo_documento	SUM(precio)	AVG(precio)
Novela	19.000	3800
Divulgativo	4500	4500
Técnico	3500	3500

Es posible indicar en el SELECT nombres de columnas con funciones de agregación siempre y cuando se utilicen esos nombres de columnas en el GROUP BY.

**Ejemplo 16:** calcular la suma de los precios y el precio medio de los documentos agrupados por tipo de documento (novela, técnico, ...)

```
SELECT SUM(precio), AVG(precio), tipo_documento
```

*FROM documento*  
*GROUP BY tipo\_documento*

Resultado:

Tipo_documento	SUM(precio)	AVG(precio)
Novela	19.000	3800
Divulgativo	4500	4500
Técnico	3500	3500

Si se quiere restringir los grupos de salida se utiliza la cláusula HAVING seguida de las condiciones que debe cumplir. El proceso que se sigue es el siguiente: la cláusula WHERE restringe las filas de la tabla, después se agrupan y sólo se seleccionan aquellas que cumplan las condiciones de la cláusula HAVING.

**Ejemplo 17:** número de grupos que existen en cada curso

*SELECT curso, COUNT(\*)*  
*FROM grupo*  
*GROUP BY curso*

Resultado:

Curso	COUNT(*)
1	2
2	2
3	1

**Ejemplo 18:** cursos que tienen un único grupo y que además es de tarde

*SELECT curso*  
*FROM grupo*  
*GROUP BY curso*  
*HAVING COUNT(\*)=1 AND turno='T'*

Resultado:

Curso
3

**Ejemplo 19:** media del número de copias de los documentos agrupados por tipo de documento y que la media sea mayor que 10.

```
SELECT AVG(num_copias), tipo_documento
FROM documento
GROUP BY tipo_documento
HAVING AVG(num_copias)>10
```

Resultado:

Tipo_documento	AVG(num_copias)
Novela	13,2
Divulgativo	13

El orden de ejecución es: FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY.

**Ejemplo 20:** Hallar para cada categoría de profesorado de Madrid el sueldo máximo y mínimo ordenado por categoría, pero sólo de aquellas categorías que tengan asociados más de dos profesores

```
SELECT categoría, MAX(salario), MIN(salario)
FROM profesor
WHERE ciudad='Madrid'
GROUP BY categoría
HAVING COUNT(categoría) > =2
ORDER BY categoría
```

Resultado:

Categoría	MAX(salario)	MIN(salario)
T1	200.000	135.000
T3	125.000	120.000

## CONSULTAS BASADAS EN MÁS DE UNA TABLA

### A. Combinando tablas: Join

Hasta ahora sólo hemos seleccionado datos de una única tabla. Por medio de la combinación de tablas se pueden seleccionar datos de tablas diferentes.

La unión de dos tablas es otra tabla que tiene por columnas la unión de las columnas de las dos y por filas el producto cartesiano (cada fila de la primera tabla se concatena con todas y cada una de las filas de la otra). Tendrá que haber una condición de combinación para restringir las filas del producto cartesiano que realmente nos interesan.

#### Tipos de combinación:

- Producto cartesiano: sin restricciones
- Combinación común: la condición de combinación tiene un operado de igualdad.
- Combinación no común: la condición tiene un operador que no es el de igualdad.
- Autocombinación: se combina una tabla consigo misma.
- Combinación exterior: selecciona las filas de una tabla que no tienen correspondencia con alguna de la otra.

No influye el orden en que se especifican las columnas en el SELECT y las tablas en el FROM.

**Ejemplo 21:** Considerando las tablas AUTOR y ESCRIBE (un autor escribe varios documentos y un documento puede estar escrito por varios autores). Seleccionar el nombre de autor, nacionalidad y el código de los documentos escritos por él.

```
SELECT nombre, nacionalidad, cod_documento  
FROM autor, escribe  
WHERE autor.nombre=escribe.nombre
```



Resultado:

Nombre	Nacionalidad	Cod_documento
Miguel de Cervantes	Española	11
Emily Bronte	Inglesa	12
Isaac Asimov	Americana	15
Christian Jacq	Francesa	13
Christian Jacq	Francesa	14
Ken Follet	Inglesa	16
Paloma Martínez	Española	17
P. Isasi	Española	17
D. Borrajo	Española	17

Puede darse otra condición aparte de la de combinación, por ejemplo:

```
SELECT nombre, nacionalidad, cod_documento
FROM autor, escribe
WHERE autor.nombre=escribe.nombre AND autor.nacionalidad='española'
```

Resultado:

Nombre	Nacionalidad	Cod_documento
Miguel de Cervantes	Española	11
Paloma Martínez	Española	17
P. Isasi	Española	17
D. Borrajo	Española	17

Para las combinaciones no comunes se puede utilizar >, <, >=, <=, BETWEEN AND. Se pueden unir tantas tablas como se quiera pero siempre deberá haber n-1 condiciones de combinación para que la información sea coherente (n= número de tablas).

**Ejemplo 22:** Considerando las tablas AUTOR, ESCRIBE, DOCUMENTO (un autor escribe varios documentos y un documento puede estar escrito por varios autores). Seleccionar el nombre del autor, su fecha de nacimiento y nacionalidad, el código y el tipo de documento escritos por el autor así como el título de los mismos.

```
SELECT nombre, fecha_nac, nacionalidad, cod_documento, titulo, tipo_documento
FROM autor a, escribe b, documento c
WHERE a.nombre=b.nombre AND b.cod_documento=c.cod_documento
```

Resultado:

Nombre	Fecha_nac	Nacionalidad	Cod_documento	Título	Tipo_documento
Miguel de Cervantes	9-10-1547	Española	11	El Quijote	Novela
Emily Bronte	2-9-1818	Inglesa	12	Cumbres Borrascosas	Novela
Isaac Asimov	23-4-1930	Americana	15	Introducción a la ciencia	Divulgativo
Christian Jacq	30-6-1947	Francesa	13	La Pirámide Asesinada	Novela
Christian Jacq	30-6-1947	Francesa	14	La Ley del Desierto	Novela
Ken Follet	5-8-1949	Inglesa	16	Los Pilares de la Tierra	Novela
Paloma Martínez	2-9-68	Española	17	Gramáticas, Lenguajes y Autómatas	Técnico
P. Isasi	3-4-66	Española	17	Gramáticas, Lenguajes y Autómatas	Técnico
D. Borrajo	23-8-65	Española	17	Gramáticas, Lenguajes y Autómatas	Técnico

La autocombinación se realiza por medio de columnas que contienen la misma información. El resultado es otra tabla que tendrá por columnas dos veces las columnas de la tabla original y por filas el producto cartesiano de la tabla por sí misma.

**Ejemplo 23:** seleccionar para cada empleado de la empresa su nombre, salario y el nombre y la extensión telefónica de su jefe. Supongamos la siguiente relación empleado extendida con la información de los jefes:

Empleado

Nombre	Numero_dept	Salario	Fecha_nac	Ext_telefónica	Jefe
Pablo Montero	14	220.000	10-11-67	6543	Beatriz Cristobal
Beatriz Cristobal	13	300.000	20-9-68	6577	--
J.Luis Martín	11	150.000	25-6-77	6433	Beatriz Cristobal
Almudena López	13	350.000	4-5-60	6422	Angel Vallejo
Angel Vallejo	14	400.000	15-4-72	6321	--
Pedro García	11	200.000	12-3-70	6323	Angel Vallejo

*SELECT emp.nombre, emp.salario, superior.nombre, superior.ext\_telefónica*

*FROM empleado emp, empleado superior*

*WHERE emp.jefe=superior.nombre*

Resultado:

Nombre	Salario	Ext_telefónica	Jefe
Pablo Montero	220.000	6577	Beatriz Cristobal
Beatriz Cristobal	300.000	--	--
J.Luis Martín	150.000	6577	Beatriz Cristobal
Almudena López	350.000	6321	Angel Vallejo
Angel Vallejo	400.000	--	--
Pedro García	200.000	6321	Angel Vallejo

La combinación exterior selecciona aquellas tuplas que no tienen correspondencia (en inglés se denomina OUTER JOIN), que puede ser LEFT OUTER JOIN, RIGHT OUTER JOIN Y FULL OUTER JOIN.

**Ejemplo 24:** seleccionar los autores con los documentos que han escrito. Se deben incluir también los autores que no hayan escrito documento alguno

*SELECT nombre, cod\_documento*

*FROM autor, escribe*

*WHERE autor.nombre=escribe.nombre(+)*

Resultado:

No existe ningún autor que no haya escrito documento, luego el resultado es el mismo que sin (+)

## B .Operadores de conjunto (UNION, INTERSECT, EXCEPT)

Son los mismos que los de la Teoría de Conjuntos y combinan dos o más consultas en un mismo resultado. Los SELECT debe tener el mismo número de columnas y deben corresponderse en tipo. En estos operadores está implícita la cláusula DISTINCT y son incompatibles con ORDER BY.

- UNION: recupera todas las filas que han sido seleccionadas en los dos SELECT
- INTERSECT: devuelve las filas comunes que han sido seleccionadas por los mandatos SELECT
- EXCEPT (MINUS): devuelve las filas que han sido seleccionadas por el primer mandato SELECT y no lo han sido por el segundo

**Ejemplo 25:** Nombre de los autores de nacionalidad española e inglesa (obsérvese que si la UNION se realiza sobre la misma tabla, bastaría una condición OR).

```
SELECT nombre, nacionalidad
FROM autor
WHERE nacionalidad='española'
UNION
SELECT nombre, nacionalidad
FROM autor
WHERE nacionalidad='inglesa'
```

Resultado:

Nombre	Nacionalidad
Miguel de Cervantes	Española
Paloma Martínez	Española
P. Isasi	Española
D. Borrajo	Española
Emily Bronte	Inglesa
Ken Follet	Inglesa

**Ejemplo 26:** Supongamos que en el enunciado de la Universidad tenemos otra tabla denominada Coordinador de la siguiente forma:

Coordinador

Cod_coord	Nombre	Ciudad	Categoría	Salario
1p	D. Cuadra	Madrid	T1	200.000
2p	E. Nieto	Las Rozas	T2	250.000
3p	P. Martínez	Alcorcón	T1	225.000

Seleccionar los nombres de las personas que sean profesores y coordinadores.

```
SELECT nombre
FROM profesor
INTERSECT
SELECT nombre
FROM coordinador
```

Resultado:

Nombre
D. Cuadra
E. Nieto
P. Martínez

Seleccionar los nombres de los profesores que no son coordinadores

```
SELECT nombre
FROM profesor
MINUS
SELECT nombre
FROM coordinador
```

Resultado:

Nombre
C. Nieto
A. Sierra
C. García
J. Montero

### Consultas Anidadas

En este caso existe una consulta dentro de otra, es decir, los resultados de una consulta anidada se utilizan como valores de comparación en la cláusula WHERE.

Cuando la cláusula WHERE de la primera consulta lleva el operador =, la consulta anidada sólo podrá recuperar una única fila. Si la consulta anidada devuelve más de un valor, se producirá un error en la ejecución de la primera consulta. Lo mismo ocurre para el resto de los operadores relacionales (<, >, >=, <=).

Veamos dos ejemplos cuando la subconsulta devuelve solo un valor.

**Ejemplo 27:** autores que tengan la misma nacionalidad que Paloma Martínez.

```
SELECT nombre, fecha_nac
FROM autor
WHERE nacionalidad =
```

```
(SELECT nacionalidad
FROM autor
WHERE nombre='Paloma Martínez')
```

Resultado:

Nombre	Fecha_nac
Miguel de Cervantes	9-10-1547
Paloma Martínez	2-9-68
P. Isasi	3-4- 66
D. Borrajo	23-8-65

**Ejemplo 28:** autor y fecha de nacimiento del autor más viejo

```
SELECT nombre, fecha_nac
FROM autor
WHERE fecha_nac=
(SELECT MIN(fecha_nac)
FROM autor)
```

Resultado:

Nombre	Fecha_nac
Miguel de Cervantes	9-10-1547

Ahora cuando la subconsulta devuelve varios valores debemos utilizar el predicado IN que permite comparar un valor con una lista de valores.

**Ejemplo 29:** nombres de los empleados ordenados alfabéticamente que trabajan en el mismo departamento que Pablo Montero o Beatriz Cristobal.

```
SELECT nombre
FROM empleado
WHERE numero_dept IN (SELECT numero_dept
FROM empleado)
```

*WHERE nombre IN ('Pablo Montero', 'Beatriz Cristobal')*

Resultado: empleados que trabajan en el departamento 13 y 14.

Nombre
Pablo Montero
Beatriz Cristobal
Almudena López
Angel Vallejo

También se pueden utilizar los operadores AND y OR en las consultas anidadas. Además, en la subconsulta se puede recuperar más de una columna debiéndose cumplir las siguientes condiciones:

El número de columnas que forma parte de la condición y el de columnas seleccionadas por la consulta anidada debe ser el mismo.

Para que se cumpla la condición tienen que ser iguales todos los valores de las columnas recuperadas por la consulta anidada y los de las columnas que forman parte de la condición en la superior (no basta con que sea igual el valor de una de las columnas)

Ejemplo 30: nombre y nacionalidad de los autores que sean los más jóvenes de cada nacionalidad.

```
SELECT nombre, nacionalidad
FROM autor
WHERE (fecha_nac, nacionalidad) IN
(SELECT MIN(fecha_nac), nacionalidad
FROM autor
GROUP BY nacionalidad)
```

Resultado:

Nombre	Nacionalidad
Isaac Asimov	Americana
Christian Jacq	Francesa
Ken Follet	Inglesa
Paloma Martínez	Española

Podemos utilizar otros dos operadores: ANY y ALL.

**Ejemplo 31:** nombre y nacionalidad de los autores que sean más viejos que cualquiera de los españoles

```
SELECT nombre, nacionalidad
FROM autor
WHERE fecha_nac < ALL
(SELECT fecha_nac
FROM autor
WHERE nacionalidad='española')
```

Resultado: No hay ninguno

También podemos utilizar el predicado EXISTS (es cierto si la cardinalidad de la consulta que tiene asociada es mayor que cero y falso en caso contrario). Se utiliza NOT EXISTS como equivalente a la división en álgebra relacional o un cuantificador universal en el cálculo relacional.

**Ejemplo 32:** Profesores que imparten clases en todos los grupos (profesores tales que no hay un grupo en el cual no impartan clase).

```
SELECT cod_profesor
FROM profesor
WHERE NOT EXISTS ( SELECT cod_grupo
FROM grupo
WHERE NOT EXISTS (SELECT *
FROM impartir
WHERE impartir.cod_grupo=
grupo.cod_grupo AND
impartir.cod_profesor=
profesor.cod_profesor))
```

Resultado: No hay ningún profesor que imparta clase en todos los grupos.



### **3. Transformación del modelo E/R al modelo relacional.**

#### **3.1. Introducción**

Como vimos en el capítulo 2.3 de la primera parte de este libro, “Metodología de desarrollo de una base de datos”, la primera fase en el diseño de una base de datos es la de recoger las características que deseamos plasmar en la base de datos en un modelo conceptual, es decir, en un modelo sencillo, cercano a las personas no informáticas para que de esta forma puedan entender nuestro diseño y realizar la validación de este. Este modelo es el Entidad/Interrelación, que utiliza como elementos básicos las entidades (representadas por rectángulos), las interrelaciones (representadas por rombos) y los atributos (representadas con un círculo unido a una entidad o a una interrelación mediante una línea).

En la segunda fase del diseño tenemos que transformar el esquema realizado en el modelo Entidad/Interrelación a un esquema lógico, ya que no existe ningún Sistema Gestor de Bases de Datos que soporte el modelo E/R. Esta fase es muy importante ya que para pasar de un esquema a otro debemos preservar todas las características recogidas en la primera de fase. De esta forma nos aseguraremos que la Base de Datos que implementemos se corresponda con lo que en un principio queríamos recoger y almacenar.

En este apartado explicaremos las reglas básicas para transformar un esquema E/R en un esquema relacional. De esta forma iremos cubriendo todas las fases del diseño de una base de datos.

#### **3.2. Reglas básicas para la transformación del modelo E/R al modelo relacional.**

La primera de las reglas de transformación nos dice que toda entidad se transforma en una relación o tabla, y los atributos o características asociadas a ella pasan a ser atributos de la relación.

Supongamos que queremos transformar la entidad alumno, Figura 2.51:

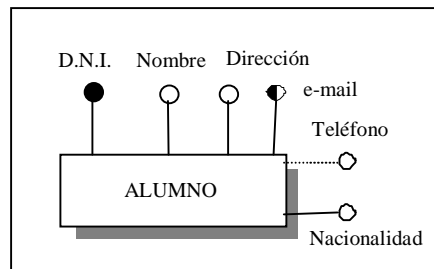


Figura 2.51: La entidad ALUMNO y sus atributos

Según la regla de transformación esta entidad pasa a ser una relación cuyo nombre, en general, es el plural del nombre de la entidad. En nuestro ejemplo, por tanto, se llamará ALUMNOS. Esta relación tendrá como clave primaria (PRIMARY KEY) el identificador principal de la entidad, *DNI*,

ALUMNOS ( Dni )

como clave alternativa (UNIQUE) el E-mail ya que proviene de un identificador alternativo,

ALUMNOS ( Dni, E-mail )

como atributos que no admiten valores (NOT NULL) tenemos Nombre, Dirección y Nacionalidad ya que en el esquema E/R son atributos obligatorios,

ALUMNOS ( Dni, E-mail, Nombre, Dirección, Nacionalidad)

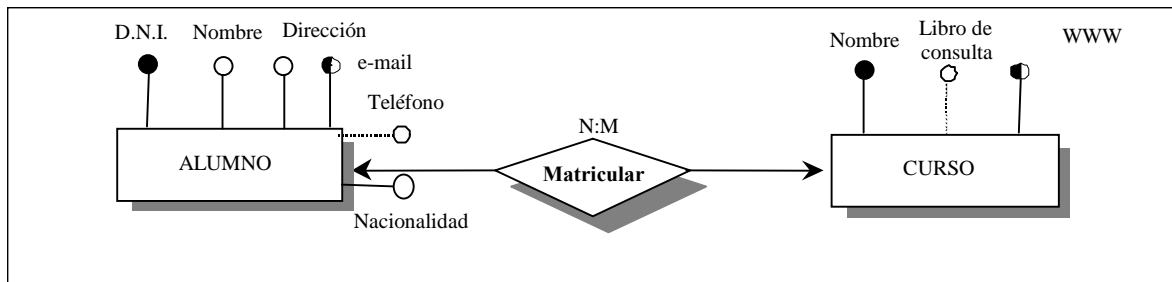
Y por último, tenemos un atributo que puede admitir valores nulos, Teléfono que proviene de un atributo opcional en el esquema E/R.

ALUMNOS ( Dni, E-mail, Nombre, Dirección, Nacionalidad, Teléfono\*)

Como podemos observar la primera regla de transformación se centra en uno de los elementos básicos del modelo E/R, las reglas que nos faltan nos guiarán para realizar la transformación del otro elemento importante de este modelo: la interrelación.

La segunda regla de transformación nos indica que las interrelaciones cuyo tipo de correspondencia es N:M se transforman en una nueva relación cuyo nombre se corresponde con el nombre de la interrelación y donde la clave primaria se compone de los atributos identificadores de las dos entidades que relaciona.

Veamos con el ejemplo de la relación que existe entre ALUMNO y CURSO cómo se transforma la interrelación N:M (Figura 2.52).



**Figura Interrelación con correspondencia N:M**

El primer paso a realizar es convertir las entidades y sus atributos en las relaciones correspondientes:

ALUMNOS ( DNI, Nombre, Dirección, Nacionalidad, E-mail, Teléfono\*)

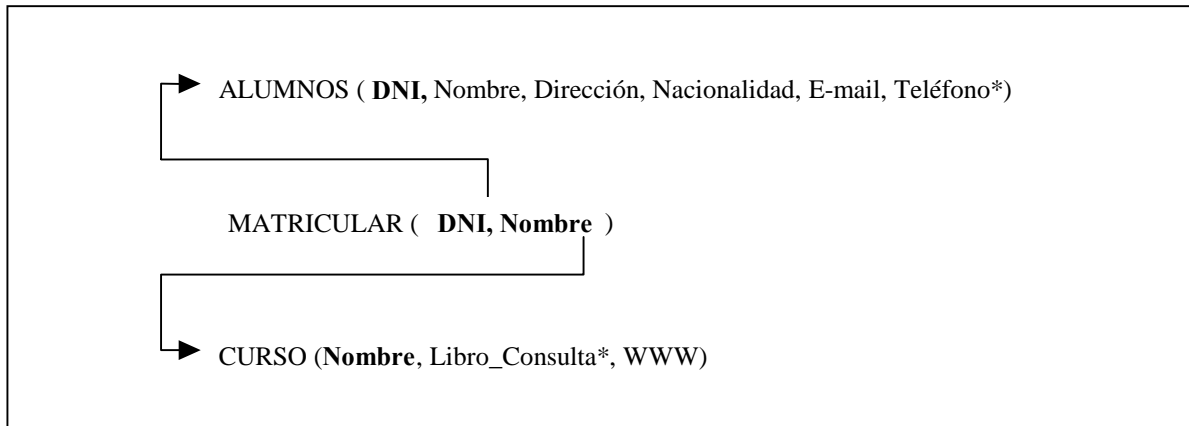
CURSO (Nombre, Libro\_Consulta\*, WWW)

Donde en la relación CURSO tenemos como clave primaria el Nombre, como alternativa WWW y como atributo que admite valores nulos Libro\_Consulta.

Como la interrelación Matricular tiene correspondencia N:M crearemos una nueva relación compuesta de dos atributos, el DNI del Alumno y el Nombre del curso donde ambos formarán la clave primaria:

MATRICULAR (DNI, Nombre)

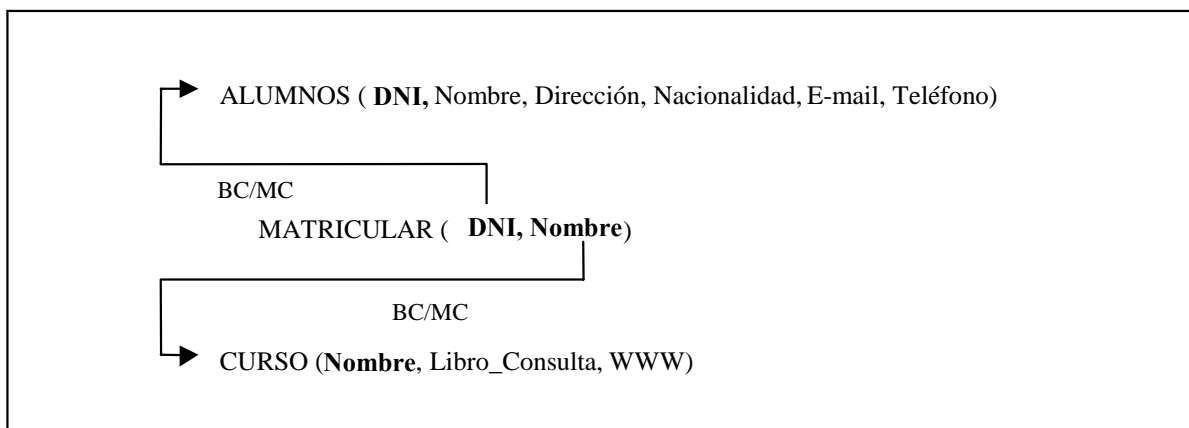
Cada uno de los atributos de la relación MATRICULAR serán además, claves ajenas que referencian a las relaciones ALUMNOS y CURSO respectivamente, de esta forma nos aseguramos que los valores de estos atributos estén almacenados previamente en las relaciones CURSO y ALUMNOS. Esto significa que un alumno no puede realizar un curso o lo que es lo mismo, que exista una tupla en la relación MATRICULAR, sin que el alumno o el curso se hallan registrado como tal.



**Figura 2.53: Transformación de una interrelación N:M**

En el esquema relacional de la Figura 2.53 nos faltaría representar las opciones de borrado y modificación que en este caso sólo pueden ser o restringidas o en cascada ya que, tanto el DNI como el Nombre al pertenecer a la clave primaria de la relación MATRICULAR no pueden tomar valores nulos ni valores por defecto.

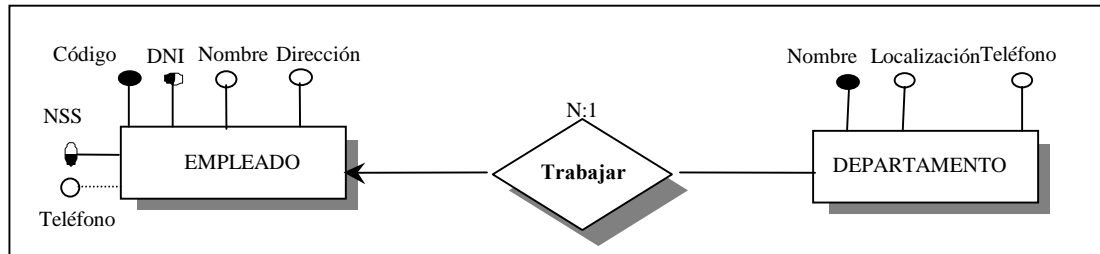
Si borramos un alumno ¿borraríamos todos los cursos que ha realizado?. Para recoger esta característica deberíamos tomar la opción de borrado en cascada al igual que si quisiéramos que las modificaciones en la relación ALUMNO se propagaran a la relación MATRICULAR. Por el contrario, si no queremos borrar a los alumnos que tengan cursos asociados en MATRICULAR escogeríamos la opción de borrado restringida. Tomamos la primera alternativa explicada y razonamos de la misma forma para la clave ajena Nombre.



**Figura 2.54: Transformación de una interrelación N:M con las opciones de borrado y modificación**

La tercera y última regla nos indica que la transformación de interrelaciones cuyo tipo de correspondencia es 1:N se traduce en una propagación de clave o en una nueva relación si la

interrelación posee atributos. Veamos en que consiste el fenómeno de propagación de clave mediante el siguiente ejemplo.



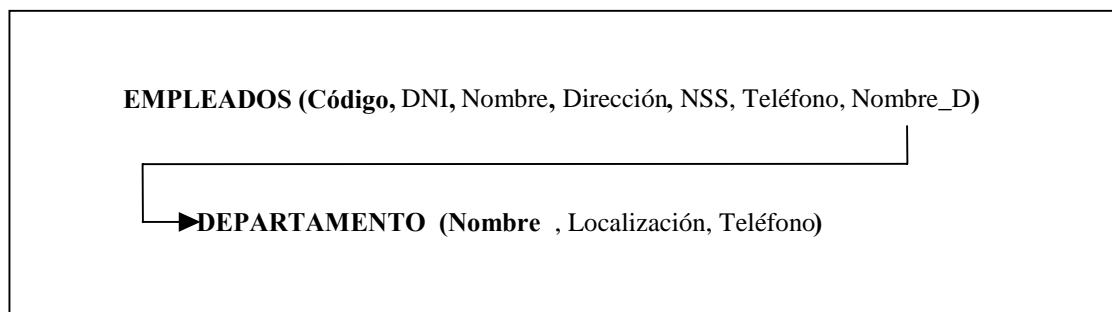
**Figura 2.55: Interrelación Trabajar con correspondencia 1:N**

Transformamos las entidades en relaciones:

EMPLEADOS (Código, DNI, Nombre, Dirección, NSS, Teléfono)

DEPARTAMENTO (Nombre, Localización, Teléfono)

Y para representar el tipo de interrelación 1:N añadimos un nuevo atributo en la relación EMPLEADO, Nombre\_D, que será clave ajena que referencia a la relación DEPARTAMENTO.



**Figura 2.56: Transformación de la Interrelación Trabajar**

De esta manera recogemos que un empleado trabaja en un solo departamento y que un departamento puede tener asociados distintos empleados, o lo que es lo mismo, representamos la semántica de una interrelación del tipo 1:N aunque perdemos el nombre de la interrelación.

### 3.3. Caso práctico

Para afianzar los conocimientos sobre la aplicación de las reglas de transformación vamos a retomar el ejemplo que diseñamos en el caso práctico expuesto en la sección 1 de esta segunda parte, acerca de una base de datos para el proyecto MENTOR.

El esquema E/R correspondiente al diseño de la base de datos se muestra a continuación:

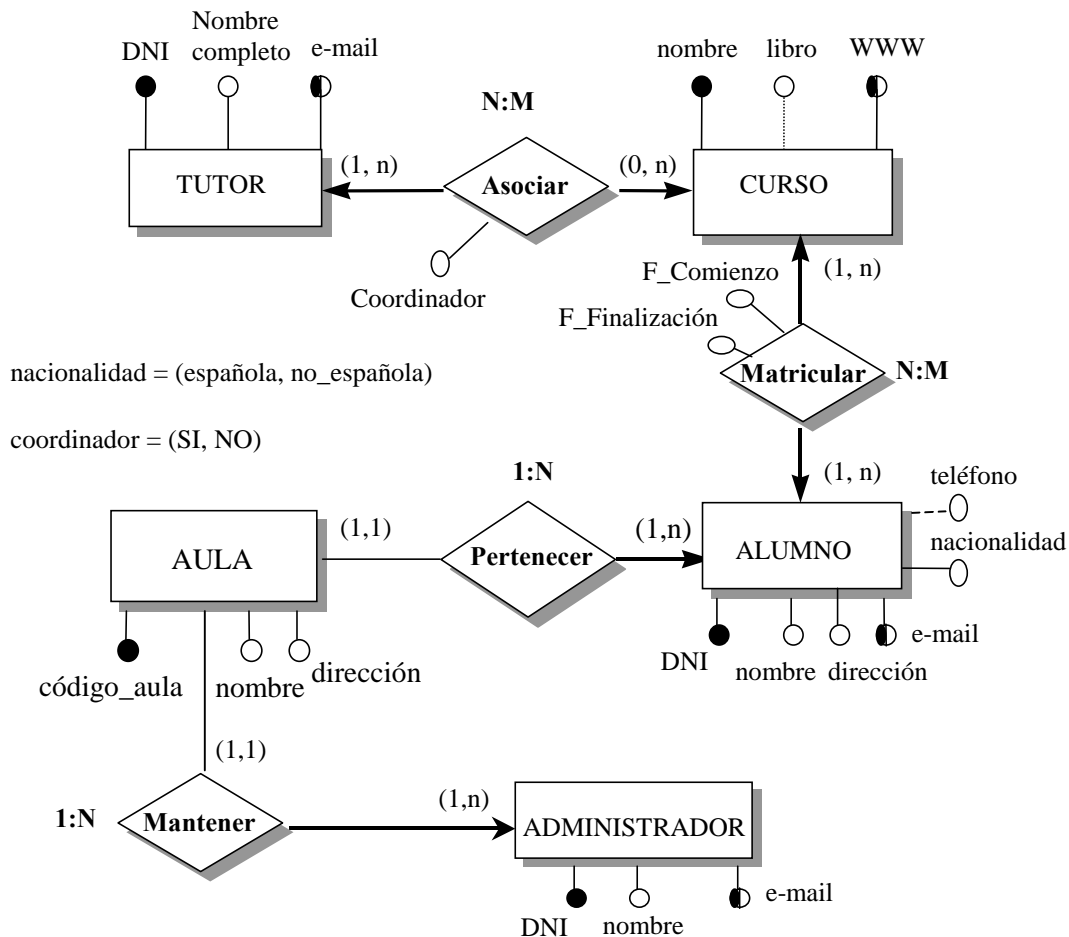
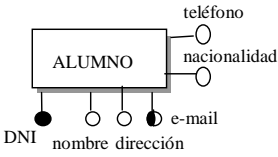
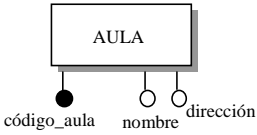


Figura 2.57: Esquema E/R completo

La transformación la haremos interrelación a interrelación, por lo que primero transformaremos las entidades asociadas a estas. Empezaremos por la interrelación PERTENECER y las entidades ALUMNO y AULA que asocia.

	Objetos en el modelo E/R	Transformación al modelo relacional
	<b>Entidad</b> ALUMNO  <b>Identificador principal</b> DNI  <b>Atributos Obligatorios</b> Nombre Dirección Nacionalidad	<b>Relación</b> ALUMNOS  <b>Clave primaria</b> DNI  <b>Atributos NOT NULL</b> Nombre Dirección Nacionalidad

	Objetos en el modelo E/R	Transformación al modelo relacional
	<b>Entidad</b> AULA  <b>Identificador principal</b> Código_Aula  <b>Atributos Obligatorios</b> Nombre Dirección	<b>Relación</b> AULAS  <b>Clave primaria</b> Código_Aula  <b>Atributos NOT NULL</b> Nombre Dirección

Una vez transformadas las entidades, veamos que tipo de interrelación las asocia. Pertenecer tienen un tipo de correspondencia 1:N y además no contiene ningún atributo, por lo que aplicando la tercera regla de transformación propagamos la clave de la relación AULAS a la relación ALUMNOS, quedando de esta manera.

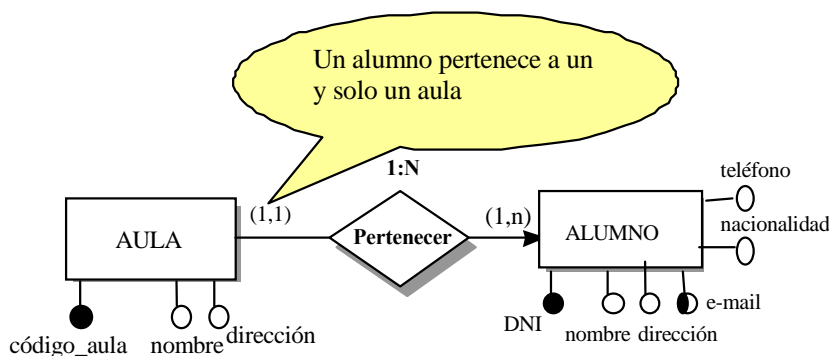
ALUMNOS (DNI, Nombre, Dirección, Teléfono\*, Nacionalidad, *e-mail*, aula)

↓  
AULAS (Código\_Aula, Nombre, Dirección)

Observar que el nuevo atributo de la relación ALUMNOS, Aula, no se llama igual que la clave primaria de la relación a la que referencia ya que no es necesario que tengan el mismo nombre.

Al añadir un nuevo atributo debemos pensar en las propiedades del mismo, es decir, si el atributo admite nulos o no, y las opciones de borrado y modificación de esta, ya que es clave ajena.

Si obligamos a que el atributo Aula tenga un valor (NOT NULL), estaremos recogiendo que un alumno siempre ha de tener un aula asignada. En caso contrario, admitiríamos que existan alumnos en nuestra Base de Datos que no pertenezcan a ningún aula. Según se muestra en la siguiente figura, debemos adoptar la primera propuesta, para no perder información recogida en el esquema conceptual.



Ahora vamos a estudiar las opciones de borrado y modificación asociadas a la clave ajena, Aula.

DNI	Nombre	Dirección	Teléfono	Nacionalidad	e-mail	Aula
07545658	Juan López	Pez,15	4674039	Española	<a href="mailto:JL@mec.es">JL@mec.es</a>	01
56321411	Andrea Solís	Bronce,25	4801325	Española	<a href="mailto:as@mec.es">as@mec.es</a>	03
36952144	Olga Calle	Príncipe,18	7771482	Española	<a href="mailto:oc@mec.es">oc@mec.es</a>	01
85669900	José Pérez	La Cruz,1	4671482	No_española	<a href="mailto:JP@mec.es">JP@mec.es</a>	02

Código_Aula	Nombre	Dirección
01	Los Carrascales	Avd. de España 3, Cercedilla
02	El Hayedo	Gran Vía 10, Coslada
03	La Nogalera	Irún 49,

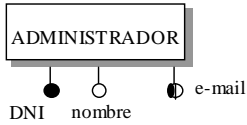
Si quisiéramos borrar la fila o tupla correspondiente al Código\_Aula = '01', ¿qué haríamos con los alumnos, Juan López y Olga Calle, asignados a esta aula?



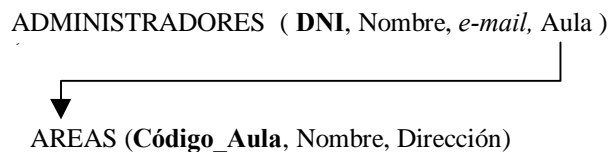
Si eligiéramos el borrado en cascada se eliminarían de forma automática las dos filas en la relación ALUMNOS correspondientes a estos alumnos. Si no queremos perder esta información, tendremos que elegir la opción de borrado restringido.

La opción de modificación será en cascada.

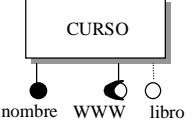
La interrelación MANTENER asocia la entidad AULA con la entidad ADMINISTRADOR. Transformaremos esta última a una nueva relación para comentar la transformación de la interrelación.

	Objetos en el modelo E/R	Transformación al modelo relacional
	<p><b>Entidad</b> ADMINISTRADOR</p> <p><b>Identificador Principal</b> DNI</p> <p><b>Identificador Alternativo</b> e-mail</p> <p><b>Atributos Obligatorios</b> Nombre e-mail</p>	<p><b>Relación</b> ADMINISTRADORES</p> <p><b>Clave Primaria</b> DNI</p> <p><b>Clave Alternativa</b> e-mail</p> <p><b>Atributos NOT NULL</b> Nombre e-mail</p>

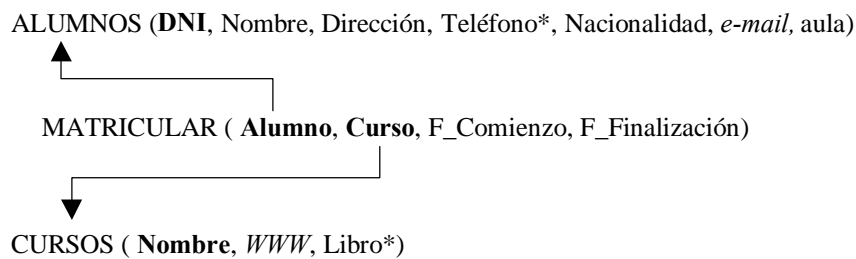
Como la interrelación MANTENER es del tipo 1:N tendremos que aumentar con un nuevo atributo la relación ADMINISTRADORES, que nos dará información acerca del código del aula a la que esta asignado un determinado administrador.



La interrelación MATRICULAR es del tipo N:M y asocia las entidades CURSO y ALUMNO. En la tabla se muestra la transformación de la entidad Curso.

	Objetos en el modelo E/R	Transformación al modelo relacional
	<b>Entidad</b> CURSO	<b>Relación</b> CURSOS
	<b>Identificador Principal</b> Nombre	<b>Clave primaria</b> nombre
	<b>Identificador Alternativo</b> WWW	<b>Clave alternativa</b> WWW
	<b>Atributos Obligatorios</b> WWW	<b>Atributos NOT NULL</b> WWW
	<b>Atributos Opcionales</b> Libro	<b>Atributos NULL</b> libro

Aplicando directamente la regla de transformación para interrelaciones N:M, crearíamos una nueva relación donde incluiríamos los atributos asociados a la interrelación MATRICULAR, F\_Comienzo y F\_Finalización.



La clave primaria de la relación MATRICULAR se compone de dos atributos, Alumno, que contiene valores de DNIs correspondientes a tuplas que aparecen en la relación ALUMNOS y Curso que almacena valores de los nombres de los cursos que se imparten en el Mentor. ¿Esta clave primaria identifica cada una de las tuplas pertenecientes a la relación MATRICULAR?

Supongamos que el alumno Juan López cuyo DNI es 07545658 se matriculó en dos cursos: “Access 97” y “Access 2000“, el 23-05-2000, y finalizó el 19-12-2000 y el 30-1-2001, respectivamente.

**ALUMNO**

DNI	Nombre	Dirección	Teléfono	Nacionalidad	e-mail	Aula
07545658	Juan López	Pez,15	4674039	Española	<a href="mailto:Jl@mec.es">Jl@mec.es</a>	01
56321411	Andrea Solís	Bronce,25	4801325	Española	<a href="mailto:as@mec.es">as@mec.es</a>	03
36952144	Olga Calle	Principe,18	7771482	Española	<a href="mailto:oc@mec.es">oc@mec.es</a>	01
85669900	José Pérez	La Cruz,1	4671482	No_española	<a href="mailto:JP@mec.es">JP@mec.es</a>	02

**MATRICULAR**

Alumno	Curso	F_Comienzo	F_Finalización
07545658	Access 97	23-05-2000	19-12-2000
07545658	Access 2000	23-05-2000	30-01-2001

**CURSO**

Curso	WWW	Libro
Access 97	<a href="http://www.mentor.es/access97.html">www.mentor.es/access97.html</a>	
Access 2000	<a href="http://www.mentor.es/accss2000.html">www.mentor.es/accss2000.html</a>	

En este caso la clave primaria de MATRICULAR identificaría cada una de las tuplas, además de esta forma controlaríamos que el mismo alumno no este matriculado en el mismo curso dos veces.

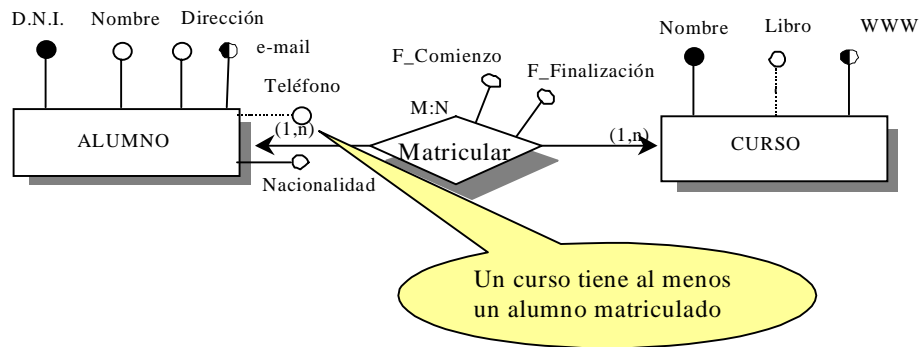
Discutamos ahora las opciones de borrado y modificación de las claves ajenas de esta relación.

Si borramos una fila de la relación alumno y este está matriculado en algún curso. ¿Borraríamos las tuplas en las que aparece su DNI en la relación MATRICULAR?

Pues parece normal que así sea, ya que si el alumno se da de baja es mejor que no aparezca como matriculado en ningún curso del Mentor.

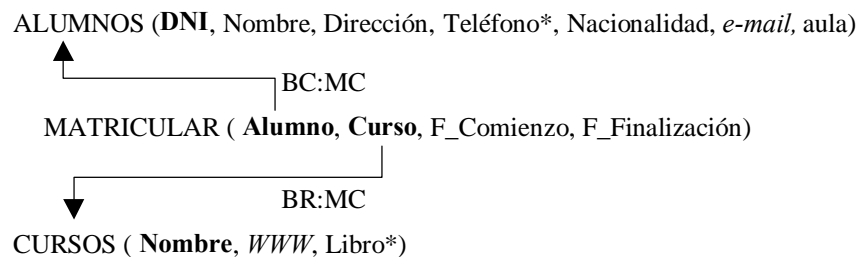
¿Qué pasaría si diéramos de baja un curso? Si borráramos una fila de la tabla CURSO y en la relación MATRICULAR aparecen tuplas con el código de este curso, la opción de borrar en Cascada haría desaparecer estas filas lo que conllevaría que los alumnos que estaban matriculados en este curso no aparezcan matriculados en ningún curso.

Eso haría que la base de datos no cumpliera los requisitos especificados en el modelo Conceptual.



La mejor opción de borrado en ese caso sería la restringida, de esta forma primero avisaríamos a los alumnos matriculados en el curso que se pretende anular para indicarles si quieren matricularse en otro de los cursos del mentor y de esta forma asignarles al nuevo curso que elijan, para luego proceder a la eliminación.

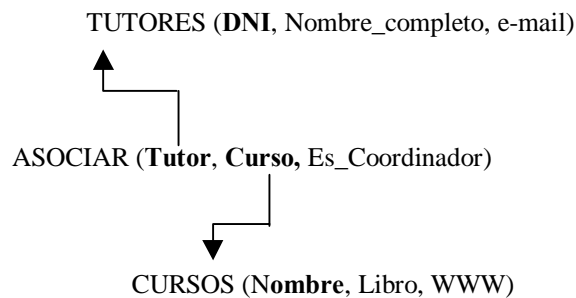
Para las opciones de modificación en ambas claves ajenas serán en cascada.



La última interrelación que nos falta por transformar es ASOCIAR, pero primeramente transformaremos la entidad TUTOR, la cual participa en dicha interrelación.

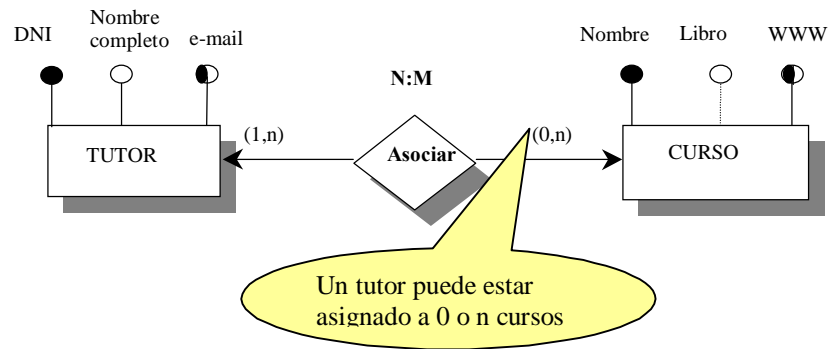
	Objetos en el modelo E/R	Transformación al modelo relacional
	<b>Entidad</b> TUTOR  <b>Identificador Principal</b> DNI  <b>Identificador Alternativo</b> e-mail  <b>Atributos Obligatorios</b> Nombre completo	<b>Relación</b> TUTORES  <b>Clave primaria</b> DNI  <b>Clave alternativa</b> e-mail  <b>Atributos NOT NULL</b> Nombre completo

La interrelación ASOCIAR es del tipo N:M por lo que tendremos que crear una nueva relación.

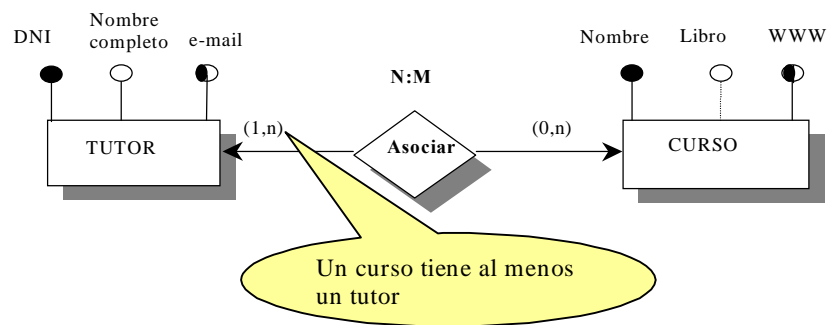


La clave primaria estará formada por dos atributos, Tutor y Curso, que a su vez serán claves ajenas que referenciarán a las relaciones TUTORES y CURSOS, respectivamente.

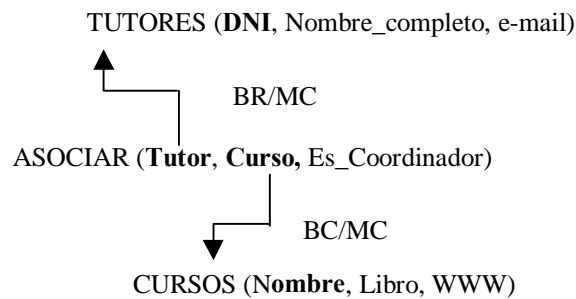
Si borramos un curso de la tabla CURSOS borraremos todas aquellas filas en la tabla ASOCIAR que se relacionen con él y no nos importará que existan tutores que no se relacionen con cursos porque cumplimos las especificaciones del modelo conceptual por lo tanto el borrado es en cascada.



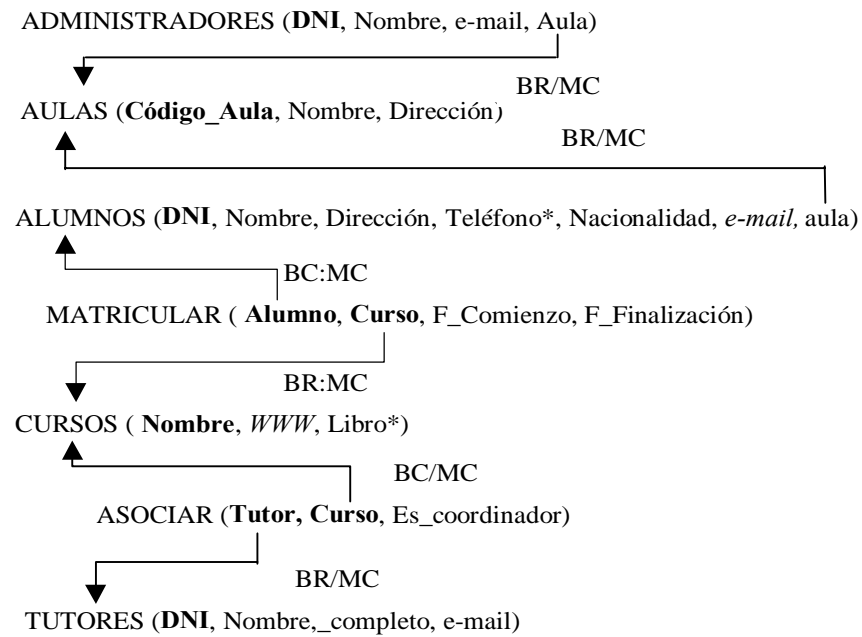
Sin embargo, la opción de borrado para la clave ajena tutor será restringida, de esta forma nos aseguramos que al dar de baja a un tutor en la relación TUTORES dejemos a algún curso sin tutor, esto produciría una inconsistencia con los requisitos especificados.



La opciones de modificación en ambos casos sería en cascada.



El esquema relacional definitivo sería el siguiente:



### Caso Práctico Propuesto: . -PARTE II.3: TRANSFORMACIÓN AL MODELO RELACIONAL

## 4 Fase de Diseño Lógico Específico: SGBD comercial

### 4.1 Introducción

Como ya se explicó en la introducción de este curso, las bases de datos surgen como respuesta a las limitaciones de los sistemas de ficheros tradicionales. Recordemos que estas limitaciones se centraban en cuatro puntos fundamentales:

- El número y tamaño de los ficheros tiende a incrementarse con el tiempo y las aplicaciones que los utilizan.
- El contenido de los ficheros provoca ambigüedad y redundancia en la información.
- El procesamiento de estos ficheros provoca un coste adicional en aplicaciones que están al margen de la información original.
- El control de los datos y de sus interrelaciones se realiza mediante los programas, es decir, no reside dentro de los datos, lo que produce mayor dificultad a la hora de la elaboración de aplicaciones.

La tecnología de bases de datos lo que pretende es lograr los objetivos de eficiencia y eficacia que los sistemas de ficheros no consiguen.

Un sistema Gestor de Bases de Datos (SGBD) consiste en un conjunto de herramientas informáticas o componentes que ponen al servicio del usuario la base de datos para que éstos la manejen y puedan elaborar su trabajo de forma eficiente. No obstante, estos sistemas no son la panacea, ni resultan totalmente eficaces ni son de fácil implantación en la empresas, pues ninguna sistema informático es completamente óptimo y además estos sistemas requieren de personal altamente cualificado el cual es difícil de encontrar y costoso de formar.

El núcleo de todo sistema de información de una empresa son los datos. Estos datos están almacenados en una base de datos. Los usuarios de la base de datos se pueden dividir en usuarios informáticos y usuarios finales. Los primeros son los encargados de la creación y el mantenimiento de la base de datos, así como a la construcción de procedimientos y programas que serán usados por los usuarios finales; estos últimos necesitan tener acceso a los datos para poder llevar a cabo su actividad.

Dentro de los usuarios informáticos la figura fundamental es la del administrador de la base de datos (ABD), que es el encargado de la vigilancia y mantenimiento de toda la base de datos. También debe existir la figura del diseñador de la base de datos, aunque en la práctica las labores



de diseño y de administración suelen recaer sobre una misma persona. Por último existen analistas y programadores, que son los que construyen las aplicaciones que van a utilizar los usuarios finales.

El SGBD debe realizar distintas funciones, realizadas sobre el conjunto de la base de datos y sobre registros específicos (consultas selectivas y actualización). Estas funciones se pueden clasificar en cuatro grandes bloques: Funciones de Definición o de Descripción, Funciones de Manipulación, Funciones de Control o de Utilización y otras funciones de carácter general.

#### **A) Funciones de Definición o de Descripción.-**

Es este tipo de funciones se engloban aquellas que permiten al diseñador de la base de datos especificar los elementos que la integran, su estructura y las relaciones entre ellos, así como las características de tipo físico y las vistas lógicas de los usuarios.

Para realizar estas funciones el SGBD se apoya en un lenguaje de definición de datos (LDD).

#### **B) Funciones de Manipulación.-**

Entre ellas se encuentran aquellas que permiten abrir, cerrar y consultar los datos de la base de datos.

Para realizar estas funciones, el SGBD se apoya en un lenguaje de manipulación de datos (LMD).

#### **C) Funciones de Control o de Utilización.-**

Estas funciones deben responder a las exigencias de utilización de la base de datos que tienen los usuarios y procesos que acceden a ella. De esta forma el SGBD suministra una serie de procedimientos que facilitan la tarea del ABD.

#### **D) Otras funciones.-**

Además de las funciones comentadas anteriormente, el SGBD debe suministrar programas para la carga de ficheros, obtención de copias de seguridad,....

Uno de los objetivos fundamentales de un SGBD debe ser el de proporcionar al usuario una visión lo más abstracta posible de la información, en el sentido de ocultar ciertos detalles referentes a la forma en que los datos se almacenan y se mantienen, pero siempre consiguiendo

una recuperación eficaz de los mismos. Para lograr esto los SGBD se basan en la arquitectura<sup>22</sup> a tres niveles de ANSI<sup>23</sup>. Estos niveles son los siguientes:

5. Nivel interno:

En este nivel se definen todos aquellos aspectos relativos al almacenamiento físico de los datos, los archivos que los contienen, los registros de estos archivos, la forma de acceder a ellos, etc.

5. Nivel conceptual:

A este nivel se define la base de datos sin necesidad de ocuparse de las características físicas de los datos, es decir, se definen las entidades, los atributos, las asociaciones,...

5. Nivel externo:

Este es el nivel con mayor grado de abstracción (el más cercano al usuario). En él se define la percepción individual de cada usuario, es decir, las partes de datos de interés para un usuario o grupo de usuarios.

#### 4. **Caso práctico en ACCESS 97**

A continuación pasaremos a explicar un caso práctico completo desarrollado en un SGBD concreto como es ACCESS. Explicaremos las distintas etapas de creación de la base de datos y las analogías y diferencias con los modelos de datos tratados en los capítulos anteriores.

##### **1.- Supuesto semántico para el estudio en Access 97.**

Para llevar a cabo el diseño de la base de datos nos basaremos en el siguiente supuesto, en el que vamos a representar los cursos, profesores y alumnos de una academia.

El supuesto es el siguiente:

- Los profesores de la academia imparten los cursos que oferta la misma. Al menos cada profesor impartirá un curso, pero también se podría dar que un mismo profesor impartiera varios cursos.
- En los cursos se matriculan alumnos. En un curso se matriculan varios alumnos, pero para que el curso se realice al menos se debe de haber matriculado un alumno.

---

<sup>22</sup> Una arquitectura de un SGBD es una forma de definir u organizar la construcción de las bases de datos.

<sup>23</sup> ANSI es el organismo americano para la estandarización de todos los temas relativos al mundo de la informática.

- A su vez un alumno puede matricularse en varios de los cursos que se ofertan en la academia.
- De los profesores se desea almacenar la siguiente información: Código del profesor, nombre del profesor, apellidos del profesor, sueldo y si trabaja con contrato fijo en la academia.
- De los cursos se desea almacenar la siguiente información: Código del curso, nombre del curso, fecha de inicio del curso, duración del mismo, nº de alumnos que tendrá como máximo y el precio del mismo.
- De los alumnos se desea recoger la siguiente información: Código del alumno, nombre del alumno, apellidos del alumno, dirección, ciudad, teléfono y edad.

## 2.- Modelo Entidad/Relación.

Antes de construir el *esquema entidad-relación* debemos analizar el problema. Una vez analizado, debemos decidir que *entidades* existen y qué *interrelaciones* hay entre ellas.

Del enunciado del problema, deducimos que existen tres entidades que son: **Profesor**, **Alumno** y **Curso**, ya que tienen existencia por sí mismas. Además de éstas tres entidades, apreciamos que existen dos interrelaciones que relacionan las ocurrencias de dichas entidades. Estas interrelaciones son **Imparte** (son impartidos) entre Profesor y Curso; y **Se-Matricula** entre Curso y Alumno.

Una vez identificadas las relaciones y las interrelaciones, hay que determinar:

- **Grado**: Es el número de entidades que participan en la interrelación. En nuestro caso tanto la interrelación *Imparten* como *Se-Matriculan* tienen grado **2**
- **Tipo de Correspondencia**: Es el número máximo de ejemplares de una entidad que pueden estar asociados con un ejemplar de la otra entidad. De esta forma, el tipo de correspondencia de la interrelación *Imparten* es **1:N** (un curso es impartido por 1 profesor como máximo y un profesor puede impartir N cursos como máximo), mientras que el tipo de correspondencia de la interrelación *Se-Matriculan* es **N:M**
- **Cardinalidad** de las entidades que intervienen en las interrelaciones: Es el número mínimo y el número máximo de ocurrencias de una entidad que pueden relacionarse con un elemento de la otra entidad. La representación se realiza mediante un par **(x,y)** situado sobre la línea que une la entidad con la interrelación. Además si la cardinalidad máxima es *n* se dibuja una punta de

flecha hacia la entidad correspondiente. En nuestro caso, las cardinalidades serían: para la entidad *Profesor* (1,1), para la entidad *Alumno* (1, n), para la entidad *Curso* con respecto a la interrelación *Imparten* (1, n) y con respecto a la interrelación *Se-Matricula* (1, n).

Una vez recogidos estos aspectos, debemos identificar que atributos o propiedades nos interesan de las entidades identificadas.

Los atributos de una entidad pueden ser de varios tipos:

- **Atributo Identificador Principal (AIP):** Identifica de manera única las ocurrencias o ejemplares de una entidad. Se representan con un círculo negro unido mediante una línea continua a la entidad. Al lado del círculo se pondremos el nombre del atributo.
- **Atributo Identificador Alternativo (AIA):** Distingue de manera única las ocurrencias o ejemplares de una entidad. Se representan igual que los anteriores salvo que sólo se pinta de negro la mitad del círculo.
- **Atributo Obligatorio (AOB):** Indica que el atributo siempre debe tomar un valor para cada ejemplar de la entidad a la que pertenece. Se representan con un círculo hueco unido a la entidad mediante una línea continua a la entidad. Al lado del círculo pondremos el nombre del atributo.
- **Atributo Opcional (AOp):** Indica que el atributo puede no tomar valor para cada ejemplar de la entidad. Se representan igual que los anteriores salvo que la línea es punteada.

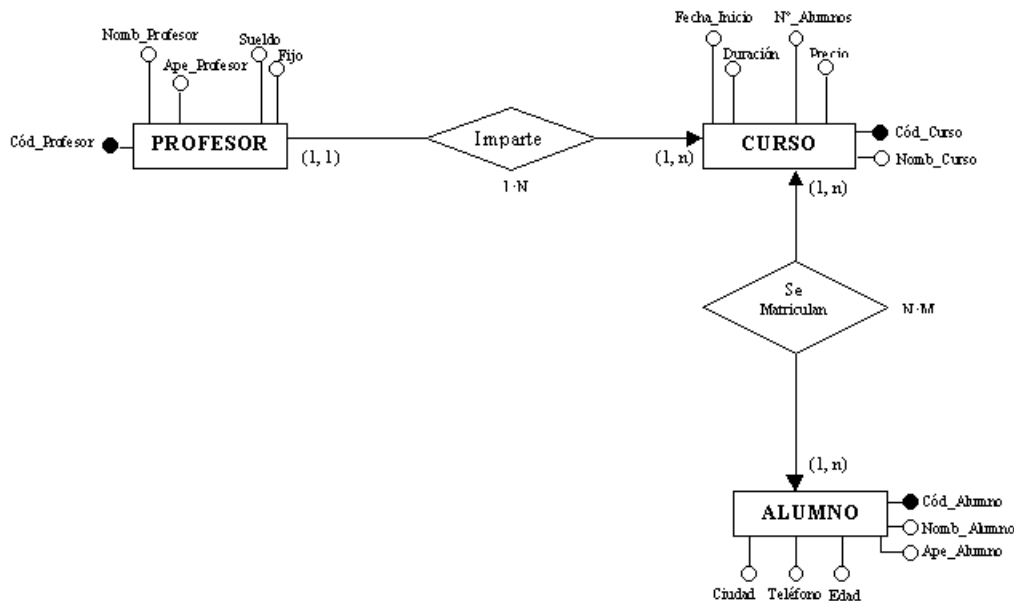
Las interrelaciones también pueden tener atributos propios. Estos atributos pueden ser obligatorios u opcionales.

En nuestro supuesto las entidades van a tener los siguientes atributos:

- **Entidad *Profesor* :** Cód\_Profesor (AIP), Nomb\_Profesor (AOB), Ape\_Profesor (AOB), Sueldo (AOB), Fijo (AOB).
- **Entidad *Curso*:** Cód\_Curso (AIP), Nomb\_Curso (AOB), Fecha\_Inicio (AOB), Duración (AOB), Número\_Alumnos (AOB), Precio (AOB).

- **Entidad *Alumno*:** Cód\_Alumno (AIP), Nomb\_Alumno (AOB), Ape\_Alumno (AOB), Dirección (AOB), Ciudad (AOB), Teléfono (AOB), Edad (AOB).

A continuación, se muestra el modelo E/R que correspondería a nuestro supuesto.



### 3.- Modelo Relacional.

En este apartado vamos a ver la transformación del modelo entidad/relación diseñado en el apartado anterior, para adaptarlo al modelo relacional, con el que posteriormente trabajaremos, ya que Access es un SGBD Relacional, y por tanto, se basa en este modelo.

El elemento básico del modelo relacional es la **relación**. Esta se puede representar como una tabla. En ella podemos distinguir un conjunto de columnas, denominadas **atributos**, que representan propiedades de la misma y que están caracterizadas por un nombre, y un conjunto de filas llamadas **tuplas**, que son las ocurrencias de la relación. El número de filas de una relación se denomina **cardinalidad**, mientras que el número de columnas es el **grado**. Existen también dominios donde los atributos toman sus valores.

Las relaciones presentan las siguientes características:

- No puede haber filas duplicadas, es decir, todas las tuplas tienen que ser distintas.
- El orden de las filas es irrelevante.

- La tabla es plana, es decir, el cruce de una fila y una columna sólo puede haber un valor (no se admiten atributos multivaluados).

Para llevar a cabo la transformación del modelo entidad/relación al modelo relacional, podemos seguir las siguientes reglas:

1. **Transformación de Entidades:** Cada tipo de entidad se transforma en una relación.
2. **Transformación de Atributos de Entidades:** Todo atributo de una entidad se transforma en un atributo (columna) de la relación en la cual se ha transformado la entidad (si el atributo estaba definido sobre un dominio, en el modelo relacional queda también definido sobre el mismo dominio). Los atributos identificadores principales (AIP) se transforman en la clave primaria de la relación. Los atributos identificadores alternativos se transforman en claves alternativas en el modelo relacional. Para los atributos multivaluados se crea una nueva relación formada con la clave primaria de la entidad y el atributo multivaluado, siendo ambos clave primaria de la nueva relación. Los atributos opcionales se transforman en una columna de la relación en la cual se ha transformado la entidad, admitiendo la posibilidad de valores nulos
3. **Transformación de Interrelaciones:**

### 3.1. *Interrelaciones N:M:*

Un tipo de interrelación N:M se transforma en una relación que tendrá como clave primaria la concatenación de los AIP de los tipos de entidad que asocia. Los atributos que forman la clave primaria de esta relación son clave ajena respecto a cada una de las tablas donde este atributo es clave primaria.

### 3.2. *Interrelaciones 1:N:*

Para este tipo de interrelaciones hay dos posibles transformaciones:

**A)** Propagar el AIP del tipo de entidad que tiene cardinalidad máxima 1 a la que tiene N.

**B)** Transformarlo en una relación, como si se tratara de una interrelación N:M.

### 3.3. *Interrelaciones 1:1:*

Para este tipo de interrelaciones hay dos posibles transformaciones:

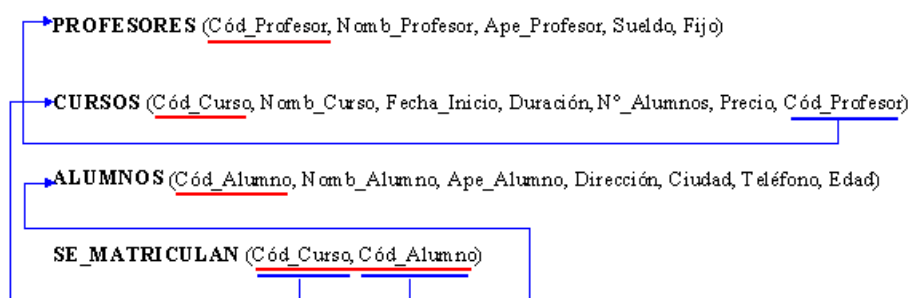
**A)** Crear una nueva relación, en la que la clave primaria estará formada por los AIP de los tipos de entidad que la interrelación asocia.

**B)** Crearemos dos relaciones, una para cada una de las entidades involucradas en la interrelación, y propagaremos una de las claves primarias de una relación a la otra. Si una de las entidades que participa en la interrelación posee cardinalidad (1,1) mientras que en la otra son (0,1) conviene propagar la clave de la entidad con cardinalidad (1,1) a la tabla resultante de la entidad con cardinalidad (0,1).

**C)** Propagar ambas claves. Esto es, propagar la clave primaria de una primera relación a la segunda y de igual forma propagar la clave primaria de la segunda relación a la primera. Se desaconseja esta opción por esta opción, ya que cuando sea implementado en un producto comercial puede dar lugar a diversos problemas. (Ej: problema en la definición porque definimos una clave ajena de una tabla que no se ha creado todavía).

**D)** Crear una sola relación para ambas entidades, donde el AIP de una de la entidades sería la clave primaria, y el AIP de la otra entidad sería clave alternativa.

De esta forma, siguiendo estas reglas de transformación del modelo entidad/relación en el modelo relacional, construiríamos el grafo relacional que se muestra a continuación:



Como se puede observar en el grafo relacional, las *entidades* **Profesor**, **Curso** y **Alumno** del modelo E/R se han transformado en *relaciones o tablas* en el modelo relacional, conservando los mismos atributos de tal forma que aquellos que eran Atributos Identificadores Principales, pasan a ser las claves primarias de las tablas del modelo relacional.

La *interrelación* **Imparte** que era del tipo **1:N**, se ha transformado al modelo relacional introduciendo en la tabla **Curso** el **código del profesor** como *clave ajena* a la *tabla* **Profesor**.

Por otra parte, la *interrelación* **Se\_Matricula** que era del tipo **N:M**, se ha transformado al modelo relacional introduciendo una nueva tabla con este mismo nombre con dos atributos: **Cód\_Curso** y **Cód\_Alumno**. Ambos forman la *clave primaria* de esta nueva tabla y a su vez son *claves ajenas* de las tablas procedentes de las entidades asociadas en la interrelación que ha originado esta nueva tabla.

#### 4.- Diseño de la Base de Datos en Access 97.

En este apartado diseñaremos la base de datos en un SGBD comercial como es Access.

De esta forma, y siguiendo con el supuesto planteado, diseñaremos una base de datos llamada **CLASES**. Esta base de datos deberá tener las siguientes tablas:

- Tabla **CURSOS**: Esta tabla contendrá información relativa a los cursos que se imparten en un centro de estudios. Los campos que contendrá dicha tabla son: Cod\_Curso, Nomb\_Curso, Fecha\_Inicio, Duración, Numero\_Alumnos y Precio. El campo **Cod\_Curso** será la clave principal.
- Tabla **ALUMNOS**: Esta tabla incluirá información de los alumnos del centro. Los campos que contendrá dicha tabla son: Cod\_Alumno, Nomb\_Alumno, Ape\_Alumno, Dirección, Ciudad, Telefono y Edad. El campo **Cod\_Alumno** será la clave principal.
- Tabla **PROFESORES**: En esta tabla se incluirá la información de los profesores del centro de estudios. Los campos que contendrá dicha tabla son: Cod\_Profesor, Nomb\_Profesor, Ape\_Profesor, Sueldo y Fijo. El campo **Cod\_Profesor** será la clave principal.
- Tabla **SE MATRICULAN**: En esta tabla se incluirá la información correspondiente a la matriculación de los alumnos en los distintos cursos. Los campos que contendrá dicha tabla son: Cod\_Curso y Cod\_Alumno. Los campos **Cod\_Curso** y **Cod\_Alumno** serán la clave principal.



**Creación de la Base de Datos.**

Los pasos a seguir para la creación de la base de datos son los siguientes:

1. Elegir el comando **Nueva base de datos** del menú **Archivo**.
2. En la ficha **General** del cuadro de diálogo **Nueva**, seleccionar el icono **Base de datos en blanco**. Hacer clic en el botón **Aceptar**. (Figura 4.1)

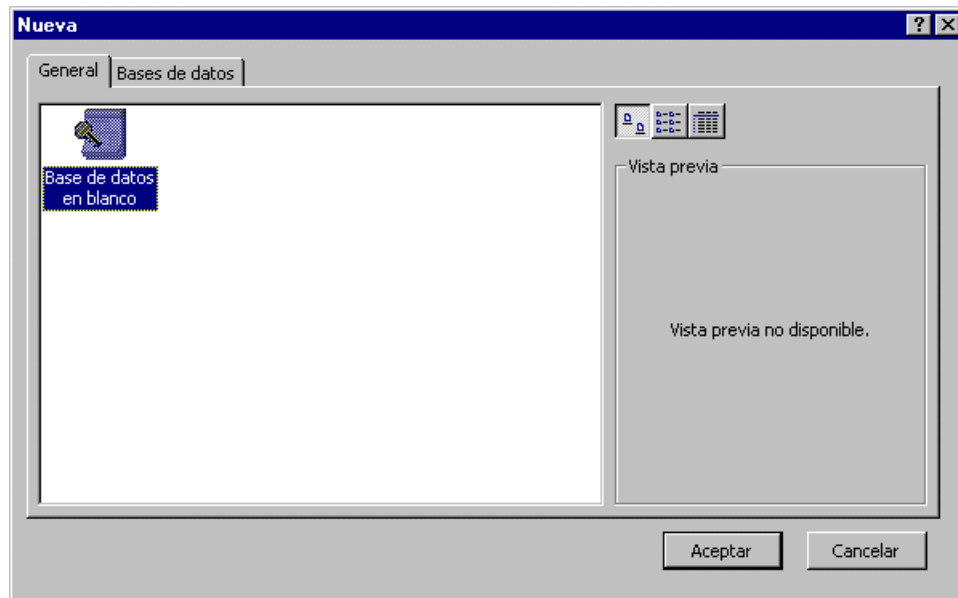


Figura 4.1.

3. En el cuadro de diálogo **Archivo nueva base de datos**, seleccionar la unidad y carpeta donde se desea guardar la nueva base de datos. Escribir el nombre (**CLASES**) para la base de datos. Hacer clic en el botón **Crear**. (Figura 4.2)

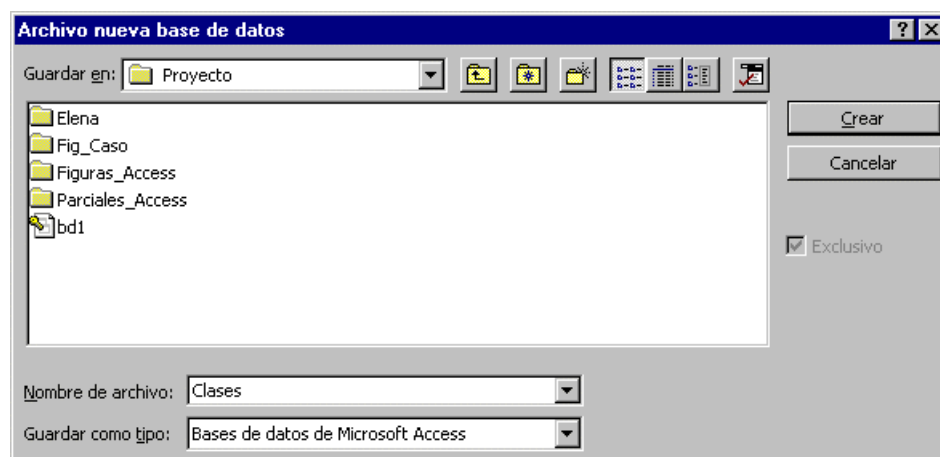


Figura 4.2.

Una vez hecho esto nos aparecerá la ventana de la Base de Datos, en la cuál aparecen una serie de pestañas correspondientes a los componentes de la misma. Ver Figura 4.3.

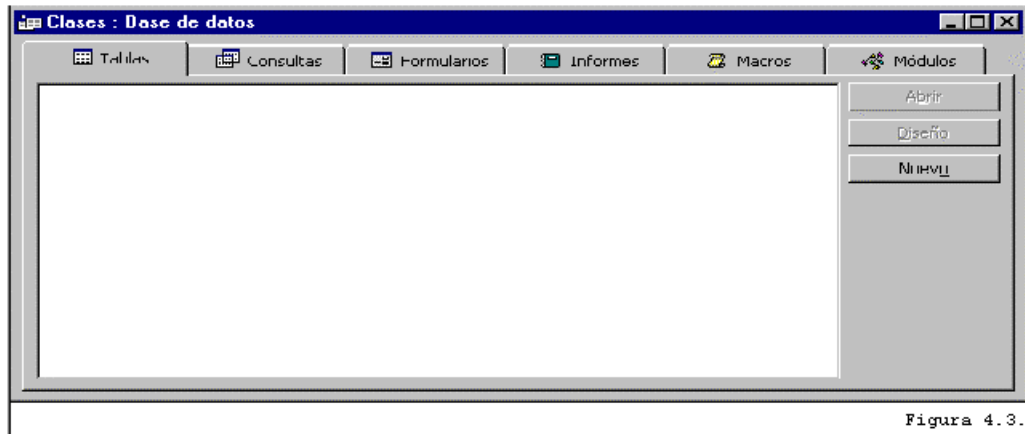


Figura 4.3.

Los distintos elementos que componen una Base de Datos Access son:

- ***Tablas***: Las tablas son las estructuras fundamentales en una base de datos Access ya que almacenan los datos que manejaremos. En una tabla, los datos están organizados en campos (columnas) y registros (filas).
- ***Consultas***: Una consulta es una herramienta para hacer preguntas sobre los datos de nuestras tablas y para realizar acciones sobre los datos. Podemos utilizar las consultas para combinar o unir datos de muchas tablas separadas pero relacionadas. Las consultas también pueden ayudarnos a cambiar, eliminar o añadir grandes cantidades de datos de un solo golpe. Finalmente, podemos utilizar las consultas como las bases para nuestros formularios e informes.
- ***Formularios***: Los formularios nos permiten mostrar e introducir datos en un formato conveniente que se asemeja a los formularios del tipo rellenar espacios en blanco. Nuestros formularios pueden ser fáciles y simples o bastante elaborados, con gráficos, líneas, y opciones de búsqueda automática que hacen que los datos sean introducidos de forma fácil y rápida. Los formularios pueden incluir incluso otros formularios (llamados *subformularios* que nos permiten introducir datos en varias tablas a un tiempo).
- ***Informes***: Los informes nos permiten imprimir o ver previamente datos en un formato eficaz. Como los formularios, los informes pueden ser simples o complicados. Algunos ejemplos son etiquetas postales, listados, sobres, cartas modelo, y facturas. Los informes también pueden presentar resultados de consulta en un formato fácil de comprender.
- ***Macros***: Una macro es un conjunto de instrucciones que automatizan una tarea que necesitamos realizar frecuentemente. Cuando ejecutamos una macro, Access lleva a cabo las acciones de la macro en el orden en el cual estas acciones están escritas. Sin escribir

una sola línea en código de programación, podemos definir macros para abrir automáticamente formularios de bases de datos, imprimir etiquetas postales, etc. Las macros nos permiten reunir un conjunto de tablas, consultas, formularios, e informes en aplicaciones de ejemplo que cualquiera puede utilizar, incluso si sabe poco o nada sobre el propio Access

- **Módulos**: Como las macros, los módulos nos permiten automatizar y acomodar Access a nuestro gusto. Sin embargo, se diferencian de las macros en que los módulos nos dan un control más preciso sobre las acciones que se toman, y requieren que el usuario tenga ciertos conocimientos de programación, así como también de Visual Basic.

### ***Creación de las Tablas de la Base de datos.***

Antes de pasar a crear las tablas, debemos decidir el tipo y el tamaño de los campos que componen cada una de las tablas. El tipo y el tamaño de un campo, debe elegirse de acuerdo a la información que se guardará en él. De esta manera, el tipo y el tamaño de los campos por los que se han optado para cada tabla aparecen a continuación en las siguientes tablas.

- Tabla ***CURSOS***:

Nombre del campo	Tipo	Tamaño
Cod_Curso	Autonumérico	Entero Largo
Nomb_Curso	Texto	20
Fecha_Inicio	Fecha	-
Duracion	Numérico	Byte
Numero_Alumnos	Numérico	Byte
Cod_Profesor	Autonumérico	Entero Largo

- Tabla ***ALUMNOS***:

Nombre del campo	Tipo	Tamaño
Cod_Alumno	Autonumérico	Entero Largo
Nomb_Alumno	Texto	15
Ape_Alumno	Texto	25
Direccion	Texto	25
Ciudad	Texto	20
Telefono	Texto	10
Edad	Numérico	Byte

- Tabla **PROFESORES**:

Nombre del campo	Tipo	Tamaño
Cod_Profesor	Autonumérico	Entero Largo
Nomb_Profesor	Texto	15
Ape_Profesor	Texto	25
Sueldo	Numérico	Simple
Fijo	Sí/No	-
Precio	Numérico	Simple

- Tabla **SE\_MATRICULAN**:

Nombre del campo	Tipo	Tamaño
Cod_Curso	Numérico	Entero Largo
Cod_Alumno	Numérico	Entero Largo

Una vez que sabemos de qué tipo y qué tamaño tendrán cada uno de los campos de las tablas, pasamos a continuación a describir los pasos a seguir para crear dichas tablas. Los pasos se describen a continuación:

1. Hacer clic en el botón **Nuevo**. (Figura 4.3)

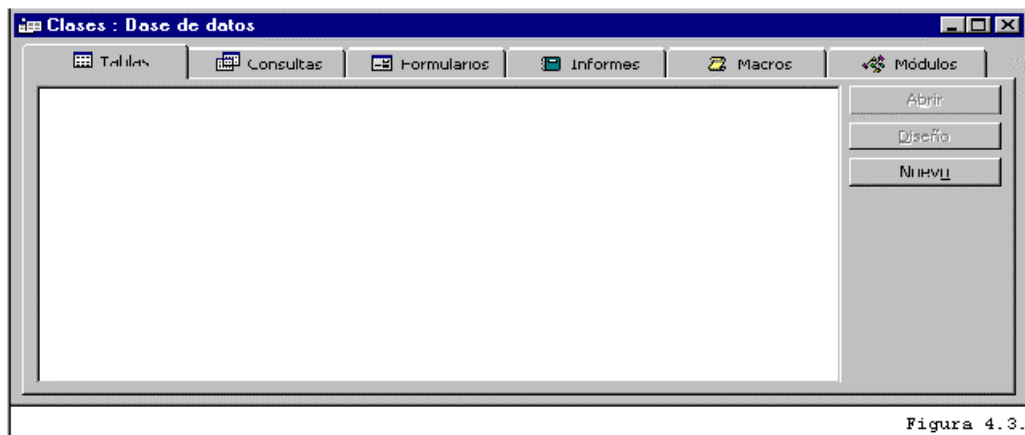


Figura 4.3.

2. En el cuadro de diálogo **Nueva tabla**, seleccionar la opción **Vista Diseño**. Hacer clic en el botón **Aceptar**. (Figura 4.4.)

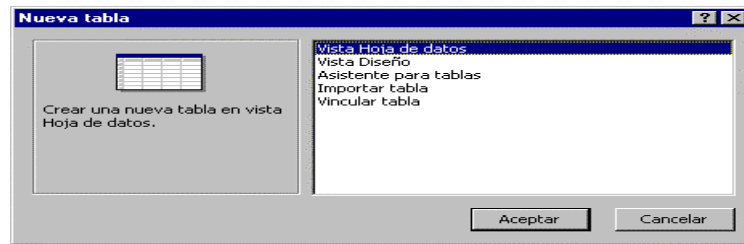


Figura 4.4.

En la Vista Diseño podemos crear una tabla completa a partir de cero, o agregar, eliminar o personalizar campos de una tabla existente. En la Vista Hoja de Datos podemos agregar, modificar o ver los datos de una tabla. También podemos revisar la ortografía de los datos de la tabla, imprimirlos, ordenar o filtrar registros, cambiar la apariencia de la hoja de datos o cambiar la estructura de la tabla agregando o eliminando columnas. Escribir el nombre **Cod-curso**. Pulsar la tecla **INTRO**.

3. Hacer clic en el botón con flecha y elegir el tipo **Autonumérico**. Pulsar la tecla **INTRO** (Figura 4.5).

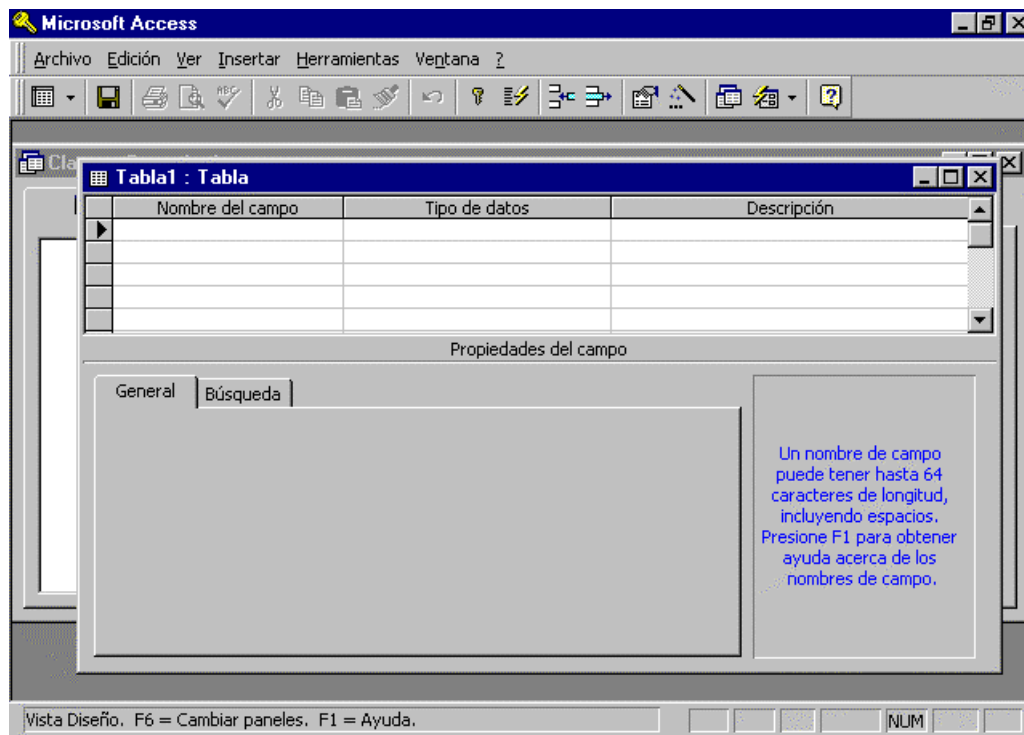


Figura 4.5.

4. Hacer clic en la propiedad **Tamaño del campo**, elegir el valor entero largo.
5. Hacer clic en la fila correspondiente a un nuevo campo.
6. Escribir el nombre **Nom-Curso**. Pulsar la tecla **INTRO**.
7. Hacer clic en el botón con flecha y elegir tipo **Texto**. Pulsar la tecla **INTRO**.

8. Hacer clic en la propiedad **Tamaño del campo**, borrar el valor existente y escribir 20.
9. Hacer clic en la fila correspondiente a un nuevo campo.
10. Escribir el nombre **Fecha inicio**. Pulsar la tecla **INTRO**.
11. Hacer clic en el botón con flecha y elegir tipo **Fecha/Hora**. Pulsar la tecla **INTRO**.
12. Escribir el nombre **Duración**. Pulsar la tecla **INTRO**.
13. Hacer clic en el botón con flecha y elegir el tipo **Numérico**. Pulsar la tecla **INTRO**.
14. Hacer clic en la propiedad **Tamaño del campo**. Hacer clic en el botón con flecha y elegir el valor **Byte**.
15. Hacer clic en la fila correspondiente a un nuevo campo.
16. Escribir el nombre **Número alumnos**. Pulsar la tecla **INTRO**.
17. Hacer clic en el botón con flecha y elegir tipo **Numérico**. Pulsar la tecla **INTRO**.
18. Hacer clic en la propiedad **Tamaño del campo**. Hacer clic en el botón con flecha y elegir el valor **Byte**.
19. Hacer clic en la fila correspondiente a un nuevo campo.
20. Escribir el nombre **Cod-profesor**. Pulsar la tecla **INTRO**.
21. Hacer clic en el botón con flecha y elegir tipo **Numérico**. Pulsar la tecla **INTRO**.
22. Hacer clic en la propiedad **Tamaño del campo**, y elegir el valor *Entero Largo*.
23. Hacer clic en la propiedad **Nuevos valores** y elegir el valor *Incremental*.
24. Hacer clic en la fila correspondiente a un nuevo campo.
25. Escribir el nombre **Precio**. Pulsar la tecla **INTRO**.
26. Hacer clic en el botón con flecha y elegir el tipo **Numérico**. Pulsar la tecla **INTRO**.
27. Seleccionar el campo **Cod-curso** y hacer clic en el botón **Establecer clave principal**. (Icono con una llave).

Todos los pasos anteriores se ven reflejados en la *Figura 4.6*.

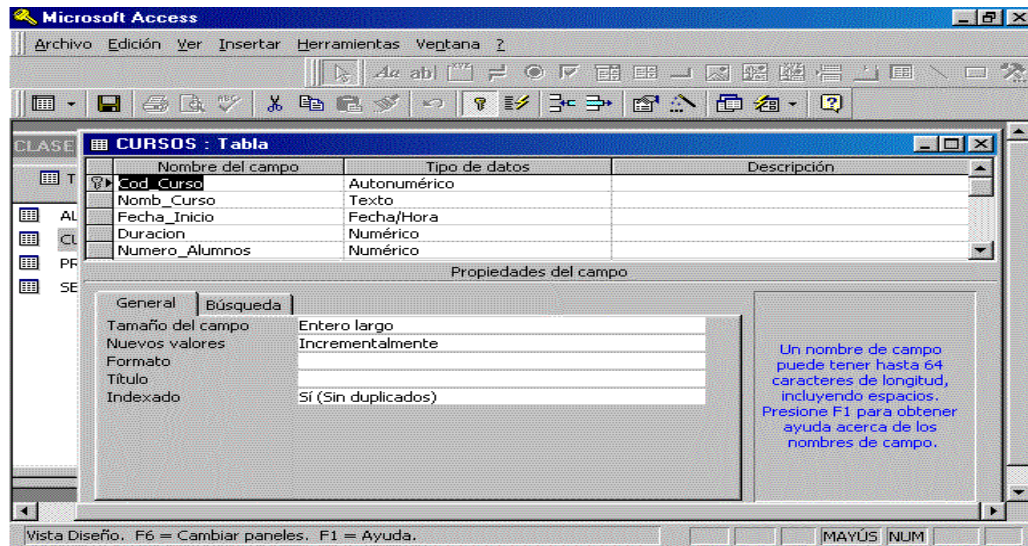


Figura 4.6.

28. Hacer clic en el botón **Guardar** y escribir el nombre de la tabla (**CURSOS**). (Ver Figura 4.7)

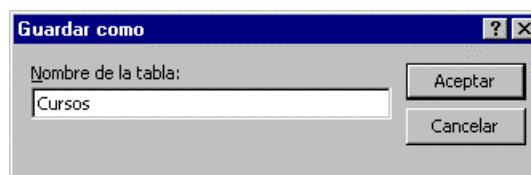


Figura 4.7.

29. Repetir el proceso para crear las tablas **ALUMNOS**, **PROFESORES** y **SE\_MATRICULAN**.

Debemos tener en cuenta, que dentro de una misma tabla, Access no permite que dos campos sean del tipo Autonumérico. Access sólo permitirá un campo cuyo tipo sea Autonumérico por tabla.

Access crea automáticamente un índice con el campo clave principal de una tabla, y lo utiliza para buscar registros y crear combinaciones entre tablas. El índice clave principal requiere que haya una entrada en el campo clave principal, y no admite valores duplicados. El orden de los campos en una clave principal de múltiples campos determina el orden predeterminado de la tabla.

Si no hay clave principal al guardar el diseño de la tabla, Access presentará un cuadro de diálogo preguntando si desea crear una. Si elegimos **Sí**, se agregará un campo de tipo **Autonumérico** a la tabla (con el valor Incrementalmente en la propiedad **Nuevos valores**), y se establecerá dicho campo como clave principal. Si elige **No**, no se creará ninguna clave principal.

Las tablas sin clave principal no pueden utilizarse para una relación, y en ellas las búsquedas u ordenaciones pueden ser más lentas.

### ***Crear Relaciones entre tablas.***

Una vez creadas las distintas tablas, necesitaremos una forma de indicarle a Access cómo debe volver a combinar esa información. El primer paso de este proceso es definir **relaciones** entre las tablas. Una vez realizada esta operación, puede crear consultas, formularios e informes para mostrar información de varias tablas a la vez.

Una relación hace coincidir los datos de los campos clave (normalmente un campo con el mismo nombre en ambas tablas). En la mayoría de los casos, estos campos coincidentes son la *Clave Principal* de una tabla, que proporciona un identificador único para cada registro, y una *Clave Externa* de la otra tabla.

Una vez relacionadas dos tablas, se llamará *Tabla Principal* a aquella que contiene el campo *Clave Principal*, y *Tabla Relacionada* a aquella que contiene el campo coincidente (*Clave Ajena o Externa*).

Existen tres tipos de relaciones en Access:

- **Relación uno a varios:** La relación uno a varios es el tipo de relación más común. En este tipo de relación, un registro de la Tabla A puede tener muchos registros coincidentes en la Tabla B, pero un registro de la Tabla B sólo tiene un registro coincidente en la Tabla A.
- **Relación varios a varios:** En una relación varios a varios, un registro de la Tabla A puede tener muchos registros coincidentes en la Tabla B y viceversa. Este tipo de relación solo es posible si se define una tercera tabla (denominada tabla de la unión) cuya clave principal consta de al menos dos campos: las claves externas de las Tablas A y B.
- **Relación uno a uno:** En una relación uno a uno, cada registro de la Tabla A sólo puede tener un registro coincidente en la Tabla B y viceversa. Este tipo de relación no es habitual, debido a que la mayoría de la información relacionada de esta forma estaría en una sola tabla. Podemos utilizar la relación uno a uno para dividir una tabla con muchos campos, para aislar parte de una tabla por razones de seguridad o para almacenar información que sólo se aplica a un subconjunto de la tabla principal.

La **Integridad Referencial** es un sistema de reglas que utiliza Access para garantizar que las relaciones entre los registros de tablas relacionadas son válidas y que no se eliminan ni



modifican accidentalmente datos relacionados. Podemos establecer la integridad referencial cuando se cumplen todas las condiciones siguientes:

- El campo coincidente de la tabla principal es una *clave principal* o tiene un *índice único* (campo indexado sin duplicados).
- Los campos relacionados tiene el mismo tipo de datos. Existen dos excepciones: un campo **Autonumérico** puede estar relacionado con un campo **Numérico** con la propiedad **Tamaño del campo** establecida a **Entero largo**, y un campo **Autonumérico** con la propiedad **Tamaño del campo** establecida a **Id. de réplica** puede estar relacionado con un campo **Numérico** con la propiedad **Tamaño del campo** establecida a **Id. de réplica**.
- Ambas tablas pertenecen a la misma base de datos Access.

Cuando se exige la integridad referencial, deben observarse las reglas siguientes:

- No podemos introducir un valor en el campo de *clave externa* de la tabla relacionada que no exista en la clave principal de la tabla principal. No obstante, puede introducir un valor Nulo en la clave externa, especificando que los registros no están relacionados.
- No podemos eliminar un registro de una tabla principal, si existen registros coincidentes en una tabla relacionada.
- No podemos cambiar un valor de clave principal en la tabla principal si ese registros tiene registros relacionados.

Si exigimos la integridad referencial e infringimos una de las reglas con las tablas relacionadas, Access muestra un mensaje y no permite el cambio.

Para las relaciones en las que se exige la integridad referencial, podemos especificar si deseamos que Access actualice en cascada y elimine en cascada automáticamente registros relacionados. Si establecemos estas opciones, las operaciones de eliminación y actualización que normalmente impediría la integridad referencial se permiten ahora. Al eliminar registros o al cambiar los valores de clave principal de una tabla principal, Access realiza los cambios necesarios en las tablas relacionadas con el fin de conservar la integridad referencial.

Si activamos la casilla de verificación **Actualizar en cascada los campos relacionados** al definir una relación siempre que cambiemos la clave principal de un registro de la tabla principal, Access actualizará automáticamente la clave principal con el nuevo valor en todos los registros relacionados. Debemos tener en cuenta que si la clave principal de la tabla principal es

un campo *Autonumérico*, la activación de la casilla de verificación *Actualizar en cascada los campos relacionados* no tendrá ningún efecto, porque no podemos cambiar el valor de un campo Autonumérico.

Si seleccionamos la casilla de verificación **Eliminar en cascada los registros relacionados** al definir una relación, siempre que eliminemos registros de la tabla principal, Access eliminará automáticamente los registros relacionados de la tabla relacionada. Al eliminar registros de un formulario y hoja de datos con la casilla de verificación *Eliminar en cascada los registros relacionados* activada, Access nos avisará que es posible que también se eliminen registros relacionados. No obstante, al eliminar registros mediante una consulta de eliminación, Access elimina automáticamente los registros de las tablas relacionadas sin mostrar aviso.

Siguiendo con nuestro ejemplo, en la base de datos **CLASES** vamos a crear las siguientes relaciones entre las tablas creadas anteriormente:

- **PROFESORES - CURSOS.** Estas dos tablas se deben relacionar a través del campo **Cod\_Profesor**. La relación será del tipo **Uno a varios**.
- **CURSOS - SE\_MATRICULAN.** Relacionar estas dos tablas utilizando el campo **Cod\_Curso**. El tipo de relación que debe establecerse será de **Uno a varios**.
- **ALUMNOS - SE\_MATRICULAN.** Estas dos tablas se relacionarán a través del campo **Cod-Alumno**. La relación será del tipo **Uno a varios**.

En este punto hay que mencionar, que realmente, hay una relación entre alumnos y cursos del tipo varios a varios, que Access no implementa directamente, sino que se apoya en una tabla intermedia para representarla. Como en el módulo I de este caso práctico, habíamos realizado el diseño lógico de la base de datos, esta tabla la creamos desde el comienzo.

Para crear estas relaciones, seguiremos los siguientes pasos:

1. Hacer clic en el botón **Abrir base de datos**.
2. Seleccionar la base de datos que se desea abrir. Hacer clic en el botón **Abrir**. En este caso abriremos la base de datos **CLASES**.
3. Hacer clic en el botón **Relaciones** (ver *Figura 4.8*) o también en el menú **Herramientas**, opción **Relaciones**.



Figura 4.8.

4. Seleccionar la tabla CURSOS y hacer clic en el botón **Agregar**.
5. Seleccionar la tabla PROFESORES y hacer clic en el botón **Agregar**.
6. Seleccionar la tabla ALUMNOS y hacer clic en el botón **Agregar**. (Ver *Figura 4.9*)

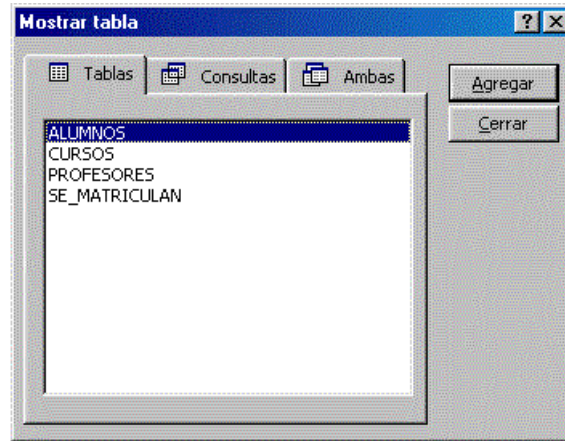


Figura 4.9.

7. Hacer clic en el botón **Cerrar**. Después de esto aparecerá la ventana **Relaciones** según aparece en la *Figura 4.10*.

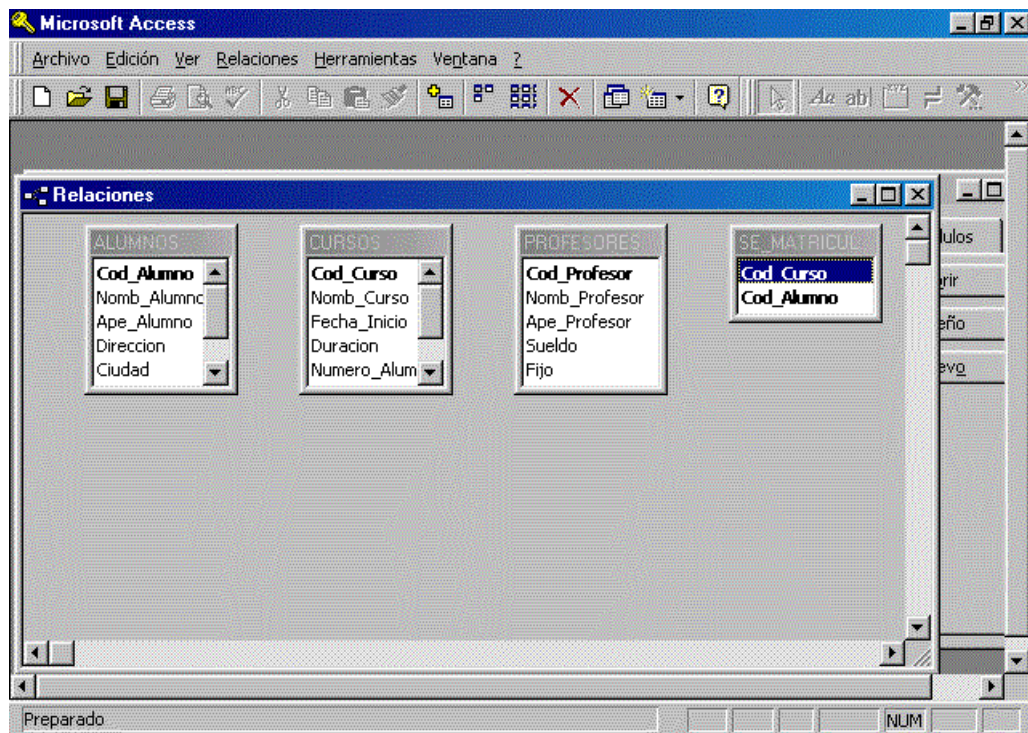


Figura 4.10.

8. Arrastrar el campo **Cod-profesor** desde la tabla PROFESORES hasta el campo **Cod-profesor** de la tabla CURSOS. Una vez que realizamos este paso nos aparece la ventana que se muestra en la *Figura 4.11*.

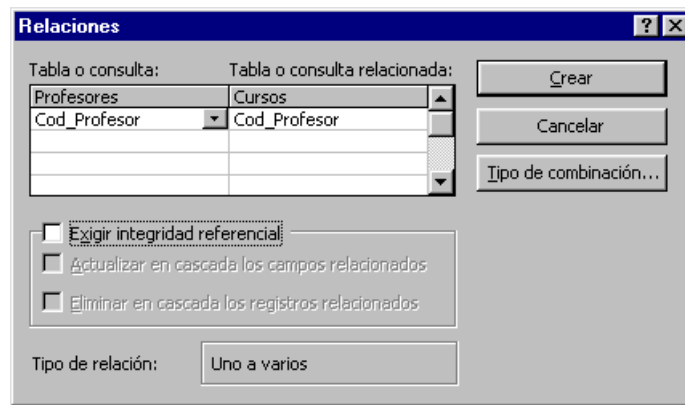


Figura 4.11.

9. Hacer clic en la casilla de selección **Exigir integridad referencial**.
10. Hacer clic en la casilla de selección **Actualizar en cascada los campos relacionados**.
11. Hacer clic en la casilla de selección **Eliminar en cascada los campos relacionados**.
12. Hacer clic en el botón **Crear**.
13. Repetir los pasos 8 a 12 para el resto de relaciones (utilizando en cada caso el campo de relación adecuado). Después de realizar las demás relaciones, en la ventana **Relaciones** se presenta el contenido que se muestra en la *Figura 4.12*.

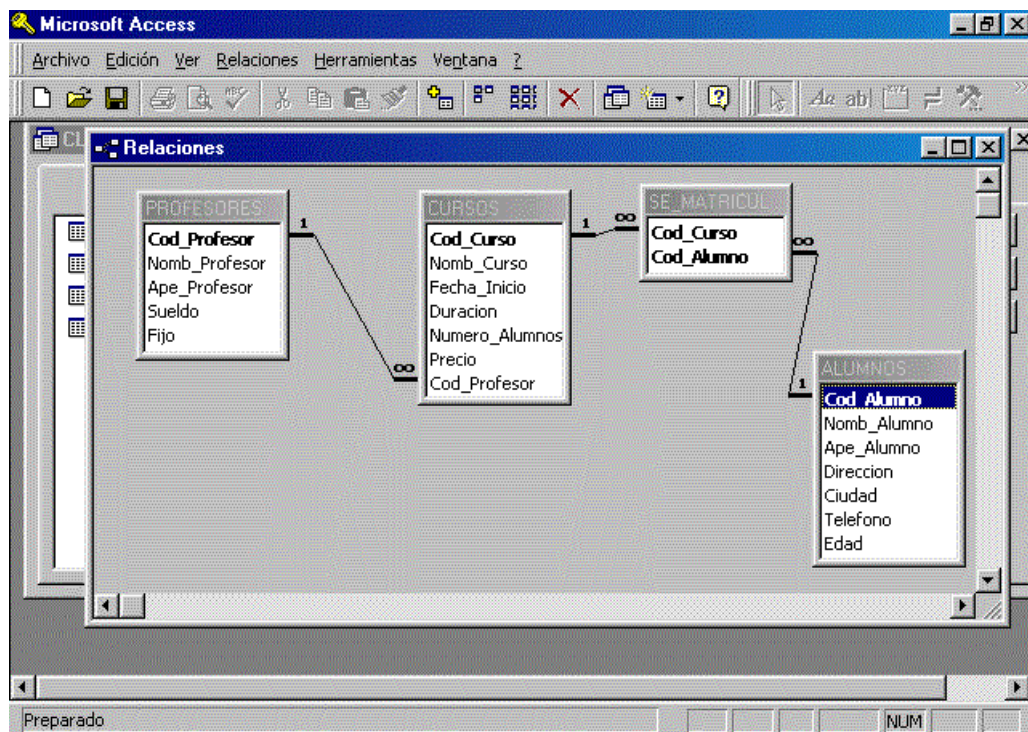


Figura 4.12.

14. Hacer clic en el botón **Guardar**.
15. Elegir el comando **Cerrar** del menú **Archivo**.

***Introducción de información en la Base de Datos.***

Una vez creada la base de datos como tal, y de haber creado cada una de las tablas que la componen, así como haber definido los atributos de cada tabla y los valores posibles de estos, además de haber creado las relaciones entre las tablas, el siguiente paso será introducir información en la base de datos.

La información se introducirá en forma de registros, y los pasos para llevar a cabo dicha operación se detallan a continuación.

En nuestro caso, y adaptándonos al supuesto que hemos planteado al principio, introduciremos los siguientes registros en cada tabla:

- Tabla **CURSOS**:

<b>Cod-Curso</b>	<b>Nomb-Curso</b>	<b>Fecha Inicio</b>	<b>Duración</b>	<b>Número Alumnos</b>	<b>Precio</b>
1	EXCEL-1	01/10/96	20	20	25.000
2	TURBOPASCAL	02/07/96	50	25	90.000
3	WORDPERFECT	05/07/96	30	20	40.000
4	EXCEL-2	16/10/96	20	20	35.000
5	C++	02/10/96	50	30	150.000
6	COBOL-1	05/07/96	50	25	55.000

- Tabla **ALUMNOS**:

<b>Cod- Alumno</b>	<b>Nomb- Alumno</b>	<b>Ape- Alumno</b>	<b>Dirección</b>	<b>Ciudad</b>	<b>Teléfono</b>	<b>Edad</b>
1	Juan	López	Pez, 15	Madrid	4674039	23
2	Andrea	Solís	Bronce, 25	Madrid	4801325	26
3	Olga	Calle	Principe, 18	Alcalá	7771482	18
4	José	Pérez	La Cruz, 1	Alcalá	4671482	25
5	Beatriz	Solbes	Ancha, 16	Madrid	7852012	26
6	Cristina	González	Pinzón, 45	Mérida	5689734	18
7	Gemma	Carranco	Mendoza, 35	Madrid	673338	25
8	Román	Mangas	Pino, 8	Tordesillas	667845	19

- Tabla **PROFESORES**:

<b>Cod-profesor</b>	<b>Nom-profesor</b>	<b>Ape-profesor</b>	<b>Sueldo</b>	<b>Fijo</b>
1	Antonia	Vázquez	185.000	Sí
2	Fernando	Calvo	165.000	No
3	María	Ruiz	210.000	No
4	Gaspar	Montalbán	150.000	Sí

- Tabla **SE\_MATRICULAN**:

<b>Cod_Curso</b>	<b>Cod_Alumno</b>
1	1
1	5
1	6
2	1
2	2
2	4
3	1
3	4
4	3
4	7

Para introducir todos estos registros en la base de datos, hay que seguir los siguientes pasos:

1. Hacer clic en el botón **Abrir base de datos**, en el caso de que no esté abierta. Si ya hemos abierto la base de datos con anterioridad, ir al paso 4.
2. Seleccionar la unidad y la carpeta donde se encuentra almacenada la base de datos **CLASES**.
3. Seleccionar el nombre de la base de datos y hacer clic en el botón **Abrir**.
4. Seleccionar la tabla **CURSOS**. (Ver *Figura 4.13*)



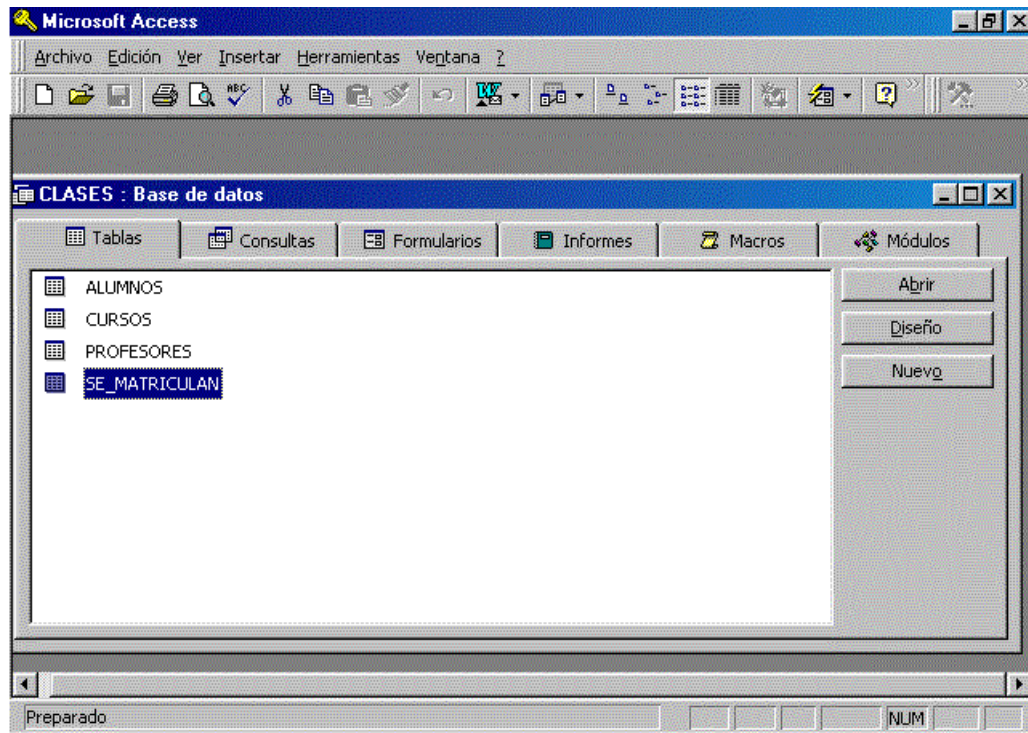


Figura 4.13.

5. Hacer clic en el botón **Abrir**.
6. Pulsar la tecla TAB, puesto que el código es un campo autonumérico que se introduce automáticamente.
7. Escribir EXCEL-1 y pulsar la tecla TAB.
8. Escribir 01/10/96 y pulsar la tecla TAB.
9. Escribir 20 y pulsar la tecla TAB.
10. Escribir 20 y pulsar la tecla TAB.
11. Escribir 25000 y pulsar la tecla TAB.
12. Repetir los pasos 6 a 11 para el resto de los registros.

En la *Figura 4.14*, se muestran los registros de la tabla CURSOS introducidos según los pasos anteriores.



Microsoft Access

Archivo Edición Ver Insertar Formato Registros Herramientas Ventana ?

CLASES - Base de datos

**CURSOS : Tabla**

Cod_Curso	Nomb_Curso	Fecha_Inicio	Duracion	Numero_Alum	Precio
1	EXCEL-1	1/10/96	20	20	250
2	TURBOPASCA	2/07/96	50	25	900
3	WORDPERFE	5/07/96	30	20	400
4	EXCEL-2	16/10/96	20	20	350
5	C++	2/10/96	50	30	1500
6	COBOL-1	5/07/96	50	25	550
*	(Autonumérico)		0	0	

Vista Hoja de datos

Figura 4.14.

13. Elegir el comando **Cerrar** del menú **Archivo**.
14. Realizar la misma operación con el resto de las tablas (ALUMNOS, PROFESORES y SE\_MATRICULAN).

### ***Operaciones de edición, reemplazamiento, ordenación y filtrado de registros.***

Una vez que hemos introducido todos los registros en la base de datos, podemos realizar ciertas operaciones sobre ellos concernientes a edición. Así, por ejemplo, podemos realizar operaciones de reemplazar, ordenación o incluso filtrar aquella información que nos es relevante según cierto criterio.

Para ver como se realizan estas operaciones de edición vamos a realizar los siguientes ejemplos:

**Ejemplo 1:** Reemplazar el apellido Montalbán de la tabla PROFESORES por González.

Para realizar dicha operación realizaremos los siguientes pasos:

1. Abrir la tabla PROFESORES.
2. Situar el punto de inserción en el campo **Ape-profesor**.
3. Elegir el comando **Reemplazar** del menú **Edición**.
4. En el cuadro **Buscar** escribir Montalbán.

5. Escribir en el recuadro **Reemplazar por** González.
6. Hacer clic en el botón **Reemplazar todos**. (Figura 4.15)

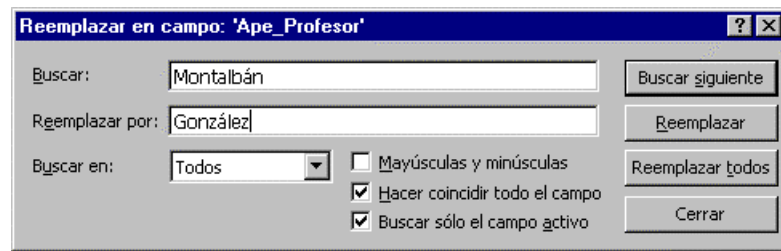


Figura 4.15.

7. Hacer clic en el botón **Sí** para aceptar el cambio.
8. Hacer clic en el botón **Cerrar**.

**Ejemplo 2:** Ordenar la tabla ALUMNOS ascendentemente según el campo **Cod-curso**.

Para realizar esta operación realizaremos los siguientes pasos:

1. Abrir la tabla ALUMNOS.
2. Situar el punto de inserción en el campo **Cod-curso**.
3. Hacer clic en el botón **Ordenar ascendente** de la barra de herramientas **Hoja de datos de la tabla** o también en el menú **Registros, Ordenar** y la opción **Ordenar ascendente** (Figura 4.16)



Figura 4.16.

4. Hacer clic en el botón **Cerrar** de la ventana Hoja de datos.

**Ejemplo 3:** Filtrar los registros de los ALUMNOS que tengan más de 20 años.

Para realizar esta operación realizaremos los siguientes pasos:

1. Abrir la tabla ALUMNOS.
2. Hacer clic en el botón **Filtro por formulario** de la barra de herramientas **Hoja de datos de la tabla** o también en el menú **Registros, Filtro** y la opción **Filtro por formulario**. (Figura 4.17).



Figura 4.17.

3. Situar el punto de inserción en el campo EDAD.
4. Escribir la expresión >20. (Ver *Figura 4.18*).

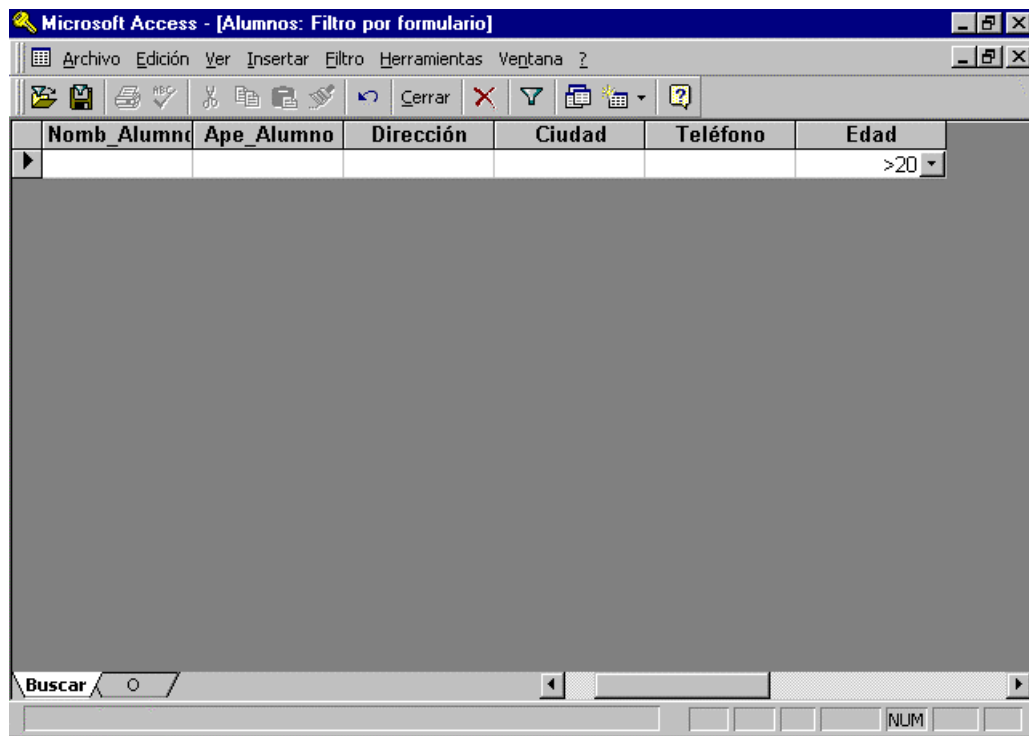


Figura 4.18.

Hacer clic en el botón **Aplicar filtro** de la barra de herramientas **Hoja de datos de la tabla**. (Ver *Figura 4.19*).



Figura 4.19.

5. Para volver a ver todos los registros, hacer clic en el botón **Quitar filtro** de la barra de herramientas **Hoja de datos de la tabla**. (Es el mismo botón que Aplicar filtro, salvo que ahora aparece hundido o presionado)

## 5. Consultas en Access sobre la Base de Datos

Las consultas son objetos que permiten formular preguntas a Access sobre el contenido de una o de varias tablas.

La información obtenida de una consulta debe cumplir los criterios o condiciones impuestas en las consultas, además de aparecer en el orden establecido. La consulta puede realizarse sobre varias tablas que se encuentren relacionadas.

Al resultado de una consulta se le denomina **Hoja de respuestas dinámica** y presenta aspecto de tabla; no obstante, las consultas no suelen crear nuevas tablas, solamente muestran parte de la tabla o de las tablas sobre las que se realiza la consulta.

Las consultas se clasifican según la acción que realizan, y pueden ser consultas de: selección, tablas de referencias cruzadas, creación de tablas, actualización de datos, datos agregados y eliminación.

Para ver cómo se realizan los distintos tipos de consultas en Access, crearemos las siguientes consultas:

### ***Consultas de Selección.***

- **Consulta 1: Profesores-cursos.** Esta consulta deberá mostrar información sobre los profesores y los cursos que imparten ordenados alfabéticamente según el apellido del profesor y la fecha de inicio del curso. La información que se desea visualizar, se obtendrá de los campos Nom-profesor, Ape-profesor, Nom-curso, Fecha-inicio, Número-alumnos y Precio.

Para realizar esta consulta seguiremos los siguientes pasos:

1. Abrir la base de datos CLASES.
2. Hacer clic en la pestaña **Consultas**.(Ver Figura 5.1)

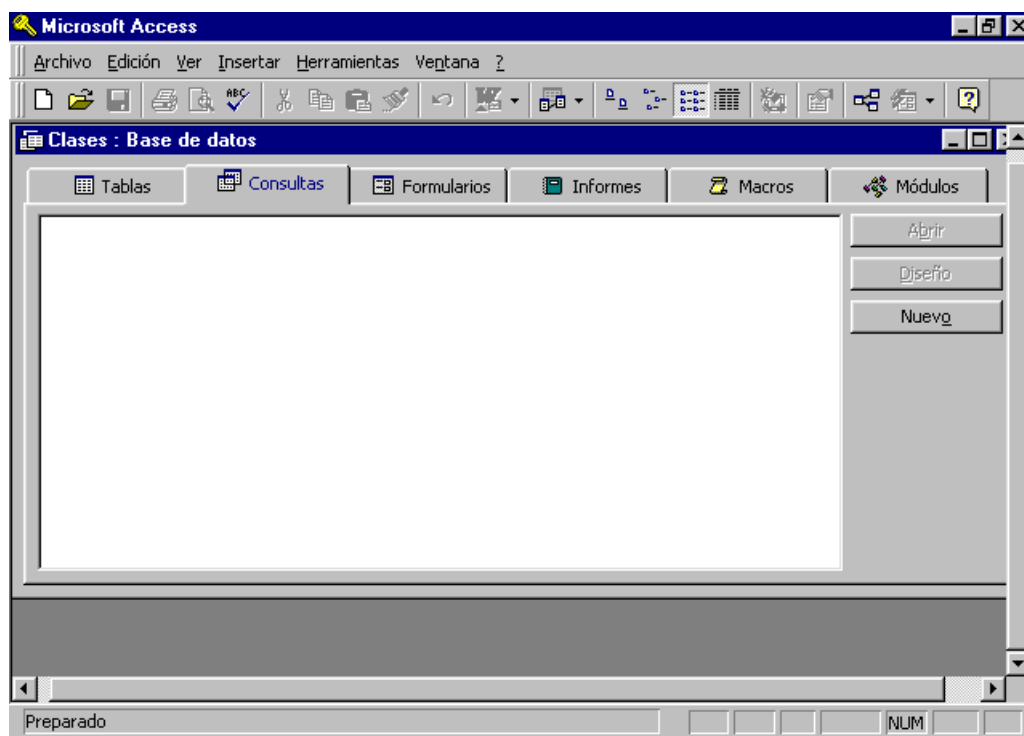


Figura 5.1.

3. Hacer clic en el botón **Nuevo**.
4. En el cuadro de diálogo **Nueva consulta** seleccionar la opción **Vista Diseño**. Hacer clic en el botón **Aceptar**. (Ver *Figura 5.2*)

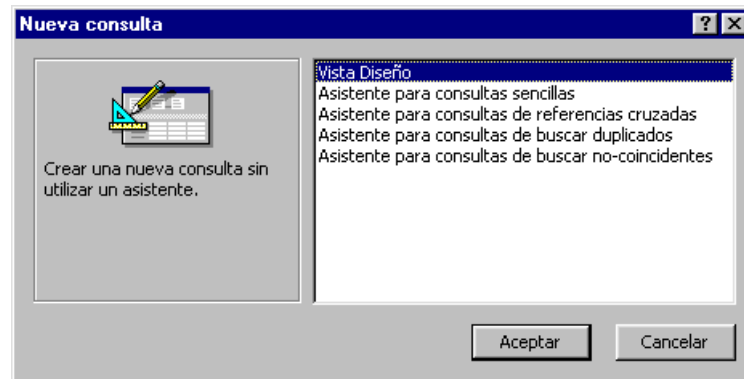


Figura 5.2.

5. Seleccionar la tabla PROFESORES. Hacer clic en el botón **Agregar**. (Ver *Figura 5.3*).

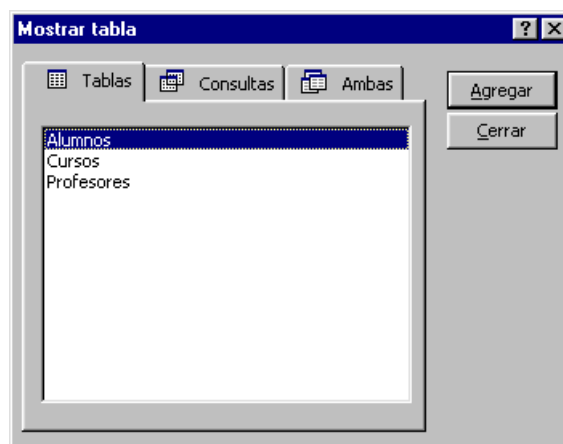


Figura 5.3.

6. Seleccionar la tabla CURSOS. Hacer clic en el botón **Agregar**. (Ver *Figura 5.3*).
7. Hacer clic en el botón **Cerrar**. Aparecerá la ventana que se muestra en la *Figura 5.4*.

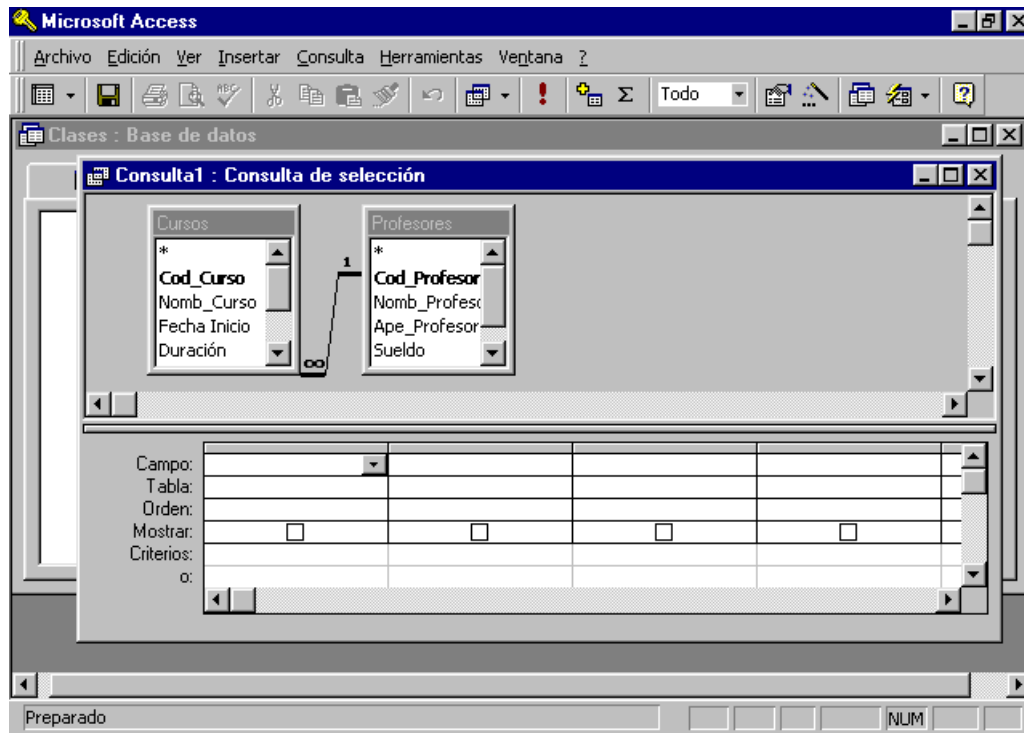


Figura 5.4.

8. Arrastrar el campo **Ape-profesor** a la fila **Campo** de la cuadrícula QBE. Hacer clic en su fila **Orden**, hacer clic en el botón con flecha y seleccionar la opción **Ascendente**.
9. Arrastrar a las siguientes columnas los campos solicitados. (Ver *Figura 5.5*)

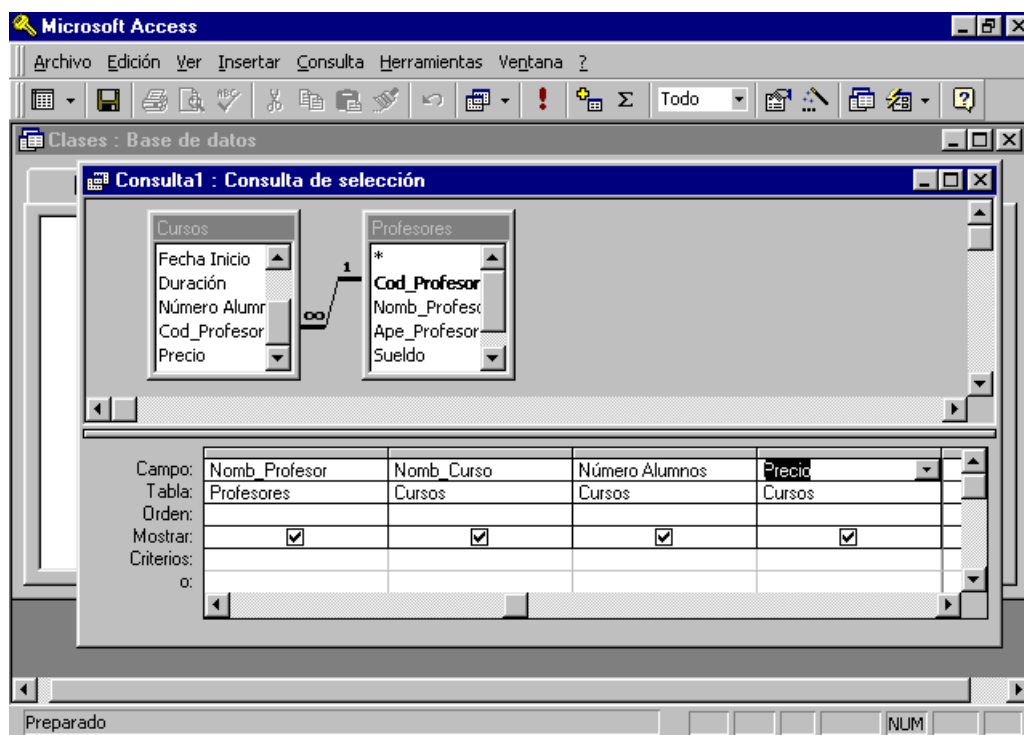


Figura 5.5.

10. Hacer clic en el botón **Vista Hoja de datos**.

11. Hacer clic en el botón **Guardar** y escribir **Profesores-cursos** en el recuadro **Nombre de la Consulta**. (Ver *Figura 5.6*)

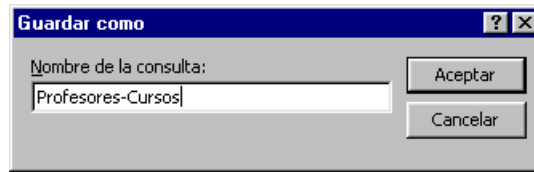


Figura 5.6.

12. Hacer clic en el botón **Aceptar**.

### ***Consultas Paramétricas.***

- **Consulta 2: Profesor-curso según alumno.** Esta consulta debe ser paramétrica en la cual se solicitará el código del alumno y se obtendrá información sobre el curso en el que está matriculado y datos sobre el profesor que imparte dicho curso. La información que se desea obtener será tomada de los campos Cod-alumno, Nom-curso, Fecha-inicio, Nom-profesor, Ape-profesor.

Para realizar esta consulta seguiremos los siguientes pasos:

1. Abrir la base de datos CLASES.
2. Hacer clic en la pestaña **Consultas**.
3. Hacer clic en el botón **Nuevo**.
4. En el cuadro de diálogo **Nueva consulta** seleccionar la opción **Modo de ver Diseño**. Hacer clic en el botón **Aceptar**.
5. Seleccionar la tabla CURSOS. Hacer clic en el botón **Agregar**.
6. Seleccionar la tabla PROFESORES. Hacer clic en el botón **Agregar**.
7. Seleccionar la tabla ALUMNOS. Hacer clic en el botón **Agregar**.
8. Hacer clic en el botón **Cerrar**.
9. Arrastrar los campos especificados en el enunciado de la práctica. (Ver *Figura 5.7*)

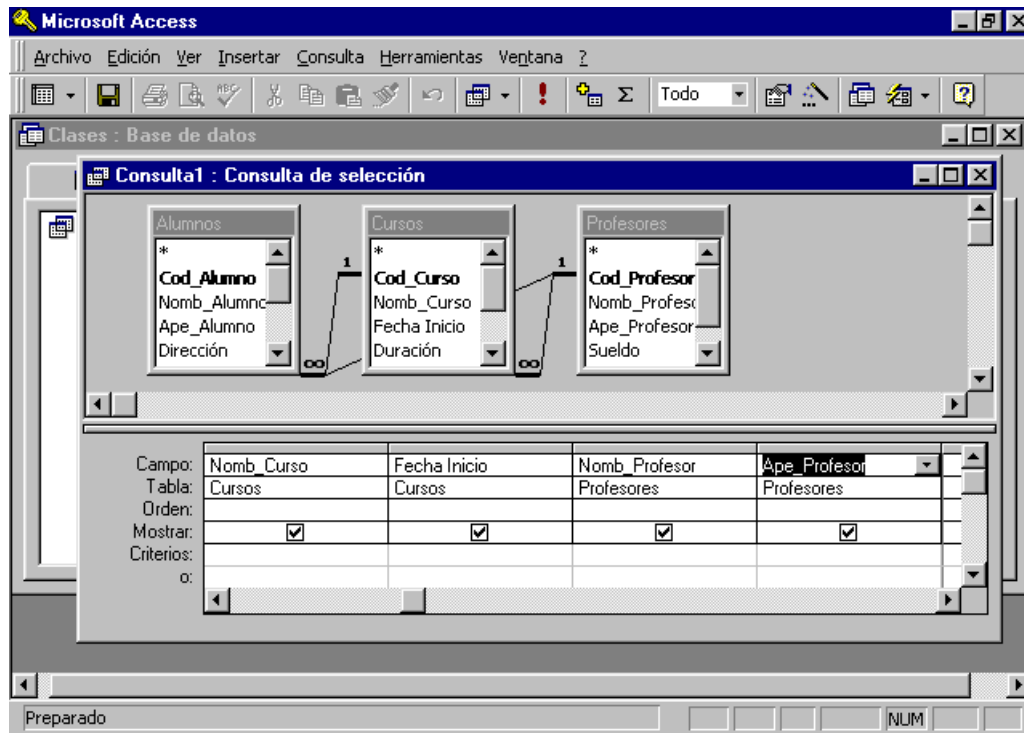


Figura 5.7.

10. En la fila **Criterios** de la columna **Cod-alumno** escribir [**Introduzca código del alumno:**].
11. Hacer clic en el botón **Vista Hoja de datos**.
12. Escribir un código de alumno y hacer clic en el botón **Aceptar**. (Ver *Figura 5.8*)

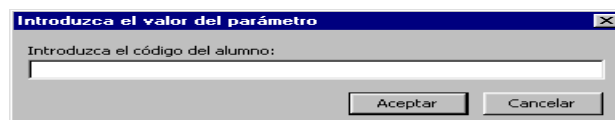


Figura 5.8.

13. Pasar a **Vista Diseño**.
14. Hacer clic en el botón **Guardar** y escribir **Profesores-cursos según alumno** en el recuadro Nombre de la Consulta.
15. Hacer clic en el botón **Aceptar**.

### ***Consulta de Eliminación.***

- **Consulta 3: Baja de alumnos.** Esta consulta será de eliminación y deberá borrar a los alumnos cuyos códigos serán introducidos a través del cuadro de diálogo de una consulta paramétrica.

Para realizar esta consulta debemos seguir los siguientes pasos:



1. Abrir la base de datos CLASES.
2. Hacer clic en la pestaña **Consultas**.
3. Hacer clic en el botón **Nuevo**.
4. En el cuadro de diálogo Nueva consulta, seleccionar la opción **Vista Diseño**. Hacer clic en el botón **Aceptar**.
5. Seleccionar la tabla ALUMNOS. Hacer clic en el botón agregar.
6. Hacer clic en el botón **Cerrar**.
7. Hacer doble clic sobre el campo asterisco que aparece en la tabla. Arrastrar el campo **Cod-alumno** a la fila campo de la cuadrícula QEB.
8. Elegir el comando **Consulta de Eliminación** del menú **Consulta**.
9. En la fila **Criterios** de la columna **Cod-alumno** escribir **[Introduzca código del alumno:]**.
10. Hacer clic en el botón **Ejecutar** (ver *Figura 5.9*), o también en el menú **Consulta** y el comando **Ejecutar**.



Figura 5.9.

11. Escribir un código de alumno y hacer clic en el botón **Aceptar**. (Ver *Figura 5.10*)

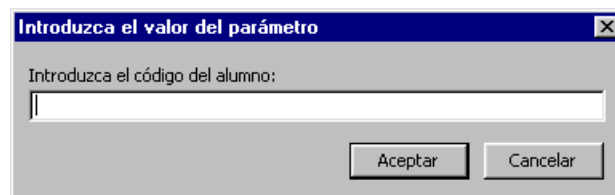


Figura 5.10.

12. Hacer clic en el botón **Sí** para confirmar.
13. Hacer clic en el botón **Guardar** y escribir **Baja de alumnos** en el recuadro **Nombre de la Consulta**.
14. Hacer clic en el botón **Aceptar**.

### ***Consulta de Actualización.***

- **Consulta 4: Aumento de sueldo.** Consulta de actualización que incrementará el sueldo de los profesores en un 10 %.

Para realizar esta consulta debemos seguir los siguientes pasos:

1. Abrir la base de datos CLASES.
2. Hacer clic en la pestaña **Consultas**.
3. Hacer clic en el botón **Nuevo**.
4. En el cuadro de diálogo **Nueva consulta** seleccionar la opción **Vista Diseño**. Hacer clic en el botón **Aceptar**.
5. Seleccionar la tabla PROFESORES. Hacer clic en el botón **Agregar**.
6. Hacer clic en el botón **Cerrar**.
7. Arrastrar el campo **Sueldo** a la fila campo de la cuadrícula QBE.
8. Elegir el comando **Consulta de Actualización** del menú **Consulta**. (Ver Figura 5.11)

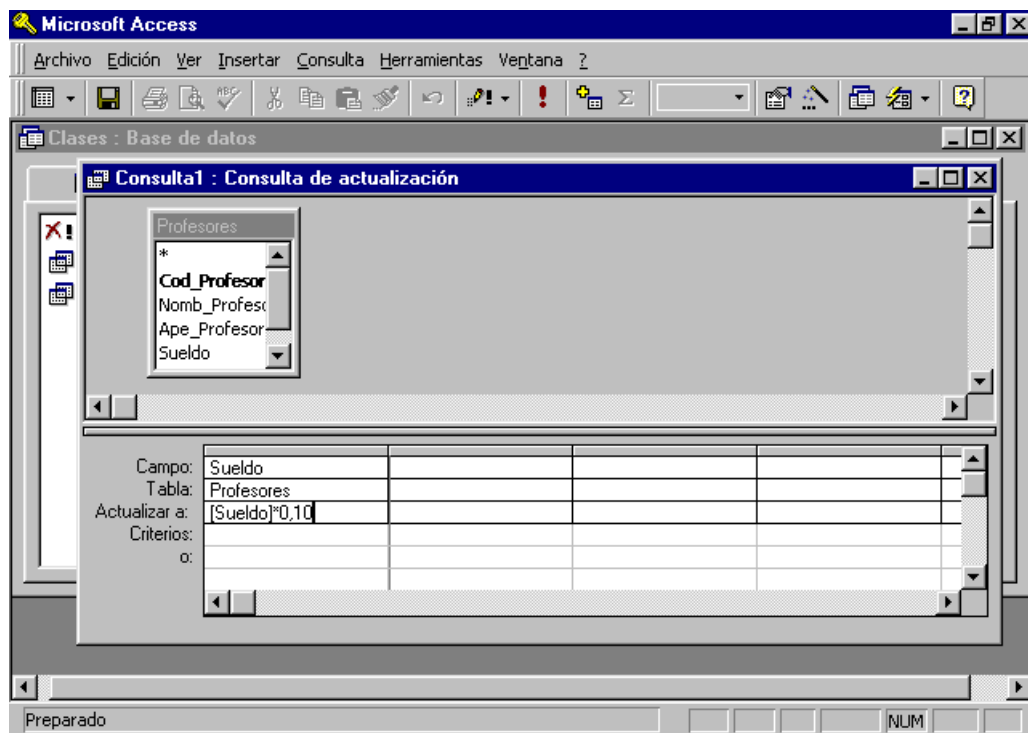


Figura 5.11.

9. En la fila **Actualizar a:** de la columna **Sueldo** escribir **[Sueldo]\*0,10**.
10. Hacer clic en el botón **Ejecutar**.
11. Hacer clic en el botón **Sí** para confirmar.

12. Hacer clic en el botón **Guardar** y escribir **Aumento de sueldo** en el recuadro **Nombre de la Consulta**.
13. Hacer clic en el botón **Aceptar**.

## 6. Formularios en Access.

Los formularios son objetos que Access proporciona para ver, introducir e imprimir datos de una o varias tablas.

El Formulario, por tanto, es una herramienta de Access que nos ofrece las mismas prestaciones que la Hoja de datos o las Consultas añadiendo una mejor forma de hacerlo.

La información incluida en un formulario está contenida en Controles. Los controles se pueden definir como objetos gráficos de un formulario que muestran datos, realizan acciones o mejoran su aspecto.

Para ver cómo se crea y se trabaja con un formulario, vamos a realizar uno, siguiendo con el supuesto propuesto como caso práctico.

De esta manera, vamos a crear un formulario que muestre los valores de la tabla CURSOS utilizando el Asistente para formularios. Incluir los campos Cod\_Curso, Nomb\_Curso, Fecha Inicio y Precio.

Para realizar el formulario requerido debemos seguir los siguientes pasos:

1. Abrir la base de datos CLASES.
2. Hacer clic en la pestaña **Formularios**.
3. Hacer clic en el botón **Nuevo**. Aparecerá la *Figura 6.1*.

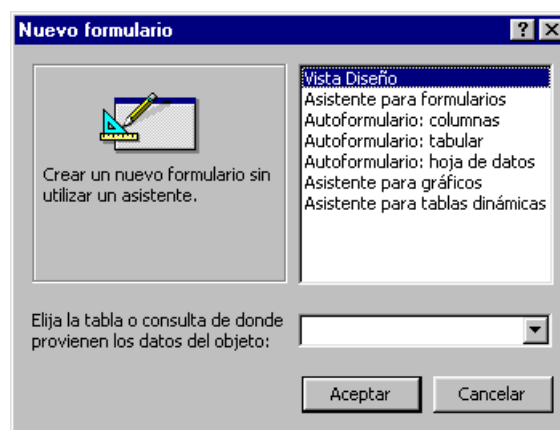


Figura 6.1.

4. Elegir la opción **Asistente para formularios**. (Ver *Figura 6.2*)

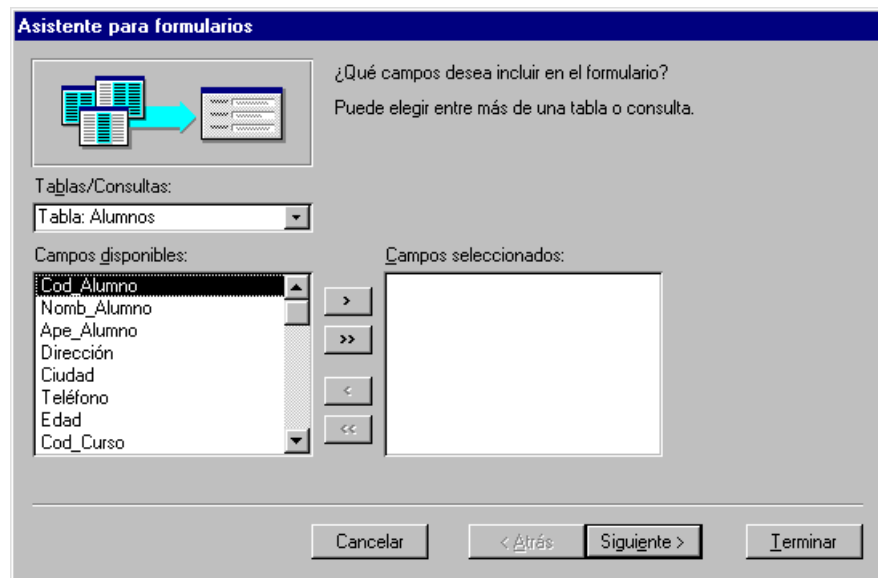


Figura 6.2.

5. Seleccionar la tabla **CURSOS**.
6. Seleccionar el campo **Cod-curso** y hacer clic en el botón **>**.
7. Repetir el paso 7 para los campos **Nom-curso**, **Fecha inicio** y **Precio**.
8. Seleccionar la distribución y estilo que desee. (Ver *Figura 6.3*).

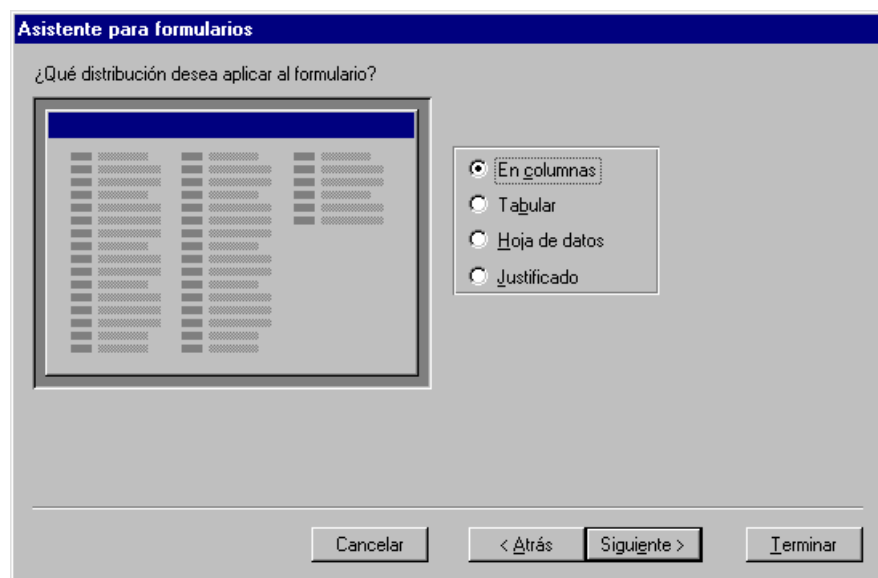


Figura 6.3.

9. Elegir el estilo que desea aplicar al formulario (ver *Figura 6.4*).

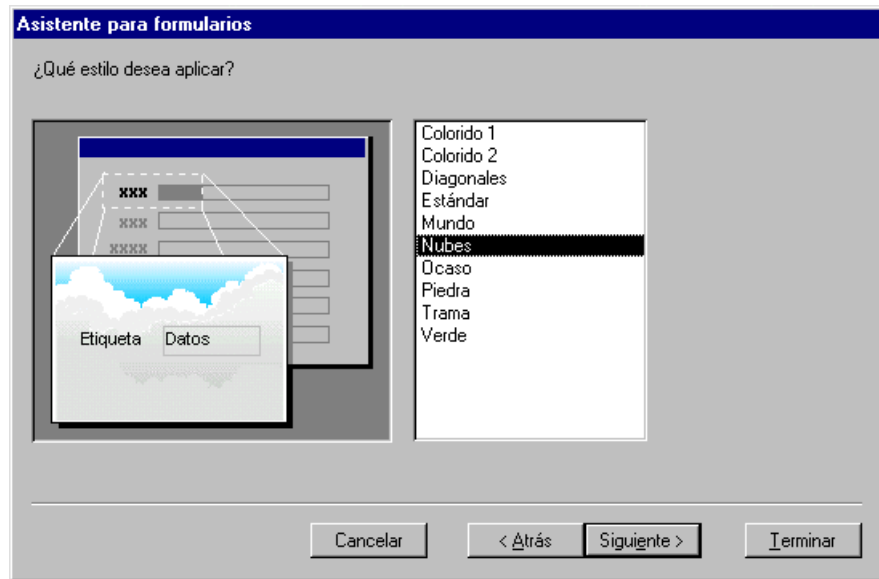


Figura 6.4.

10. Hacer clic en el botón **Siguiente** y aparecerá la ventana de la *Figura 6.5* en la que escribirá el nombre **Relación de cursos**.

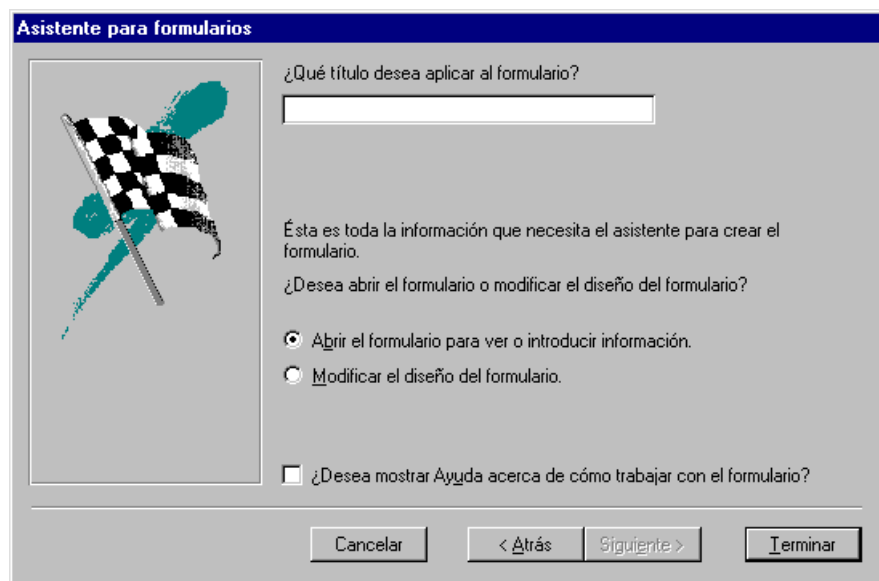


Figura 6.5.

11. Hacer clic en el botón **Terminar**.

Una vez terminado el formulario, se presentará en pantalla de la forma que se muestra en la *Figura 6.6*, según las opciones que se hayan elegido durante el diseño del formulario (en este caso, presentación: justificado y estilo: nubes).

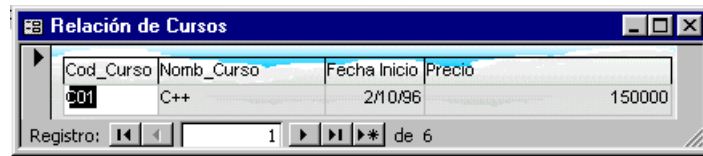


Figura 6.6.

## 7. Subformularios en Access.

Un subformulario es un formulario dentro de un formulario. El formulario primario se denomina *Formulario Principal* y el formulario incrustado en él recibe el nombre de *Subformulario*. Al conjunto se le denomina *formulario/subformulario*.

El utilizar un subformulario nos permite ver la relación existente entre los registros de dos o más tablas al estar vinculado con el formulario principal (son especialmente útiles para mostrar la relación *uno a varios*).

Cuando crea un subformulario basado en tablas que tienen una relación uno a varios, el formulario principal muestra la parte **uno** de la relación y el subformulario muestra la parte **varios** de la relación. El formulario principal está sincronizado con el subformulario para que el subformulario muestre sólo los registros relacionados con el registro del formulario principal.

Para que el formulario/subformulario pueda mostrar datos relacionados:

- Las tablas o consultas deben tener una relación *uno a varios*. Utilizar la tabla correspondiente al extremo *uno* como base para el formulario principal, y la del extremo *varios* como base para el subformulario.
- Contener un campo en común por el que vincularlas. Microsoft Access utiliza este campo para limitar los registros mostrados en el subformulario.
- Indexar los campos que vinculan ambas tablas o consultas. Si ha establecido una relación *uno a varios* entre la clave principal y ajena de ambas tablas, Access habrá creado automáticamente índices para estos campos. Con esto se consigue que el formulario se abra y el desplazamiento por los registros sea más rápido.

Si se utiliza un asistente para crear un subformulario o si arrastra un formulario o una hoja de datos de una ventana de base de datos a otro formulario para crear un subformulario, Microsoft Access sincroniza automáticamente el formulario principal con el subformulario, si se dan las siguientes condiciones:

- Que las tablas seleccionadas hayan sido relacionadas en la ventana **Relaciones**. Normalmente una relación uno a varios.

- Que el formulario principal esté basado en una tabla con una clave principal y el subformulario esté basado en una tabla que contenga un campo con el mismo nombre que la clave principal y con el mismo tipo o un tipo compatible de datos.

Una vez que hemos explicado estos aspectos, vamos a pasar a realizar un ejemplo, creándonos un formulario y un subformulario, acorde con el supuesto planteado anteriormente.

Vamos a crear un formulario principal y un subformulario con el control Subformulario/Subinforme. El formulario principal debe mostrar la tabla PROFESORES y el subformulario la relación de alumnos para cada profesor.

Para crear el formulario principal y el subformulario mencionado anteriormente, debemos realizar los siguientes pasos, según la técnica que utilizemos:

#### A. Crear el subformulario con el Autoformulario: hoja de datos

1. Abrir la base de datos CLASES.
2. Hacer clic en la pestaña Formularios.
3. Hacer clic en el botón Nuevo.
4. Elegir la opción Autoformulario: Hoja de datos. (Ver *Figura 7.1*)

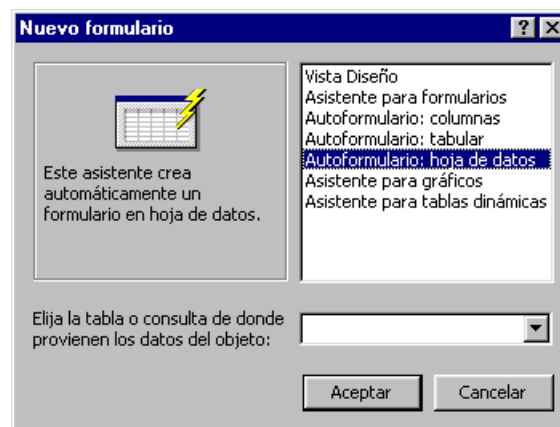


Figura 7.1.

5. Seleccionar la tabla ALUMNOS.
6. Hacer clic en el botón Aceptar.
7. Hacer clic en el botón Guardar y darle el nombre Alumnos Subformulario. (Ver *Figura 7.2*). Pulsar el botón Aceptar.



Figura 7.2.

8. Hacer clic en el botón Cerrar de la ventana del formulario.

## B. Crear el formulario principal con el Autoformulario: en columnas

1. Hacer clic en el botón **Nuevo**.

2. Elegir la opción **Autoformulario: columnas**.

3. Seleccionar la tabla PROFESORES. (Ver Figura 7.3)

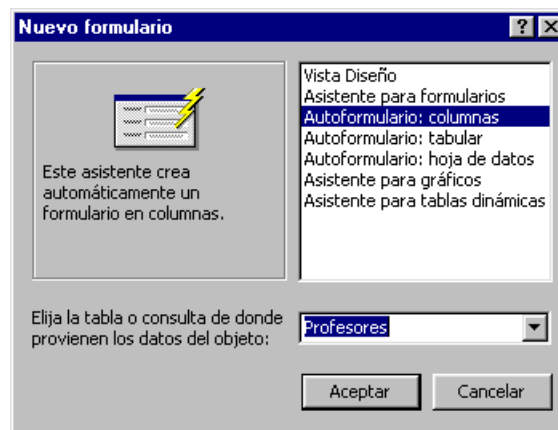


Figura 7.3.

4. Hacer clic en el botón **Aceptar**.

5. Hacer clic en el botón **Guardar** y darle el nombre **Profesores Principal**. (Ver Figura 7.4)

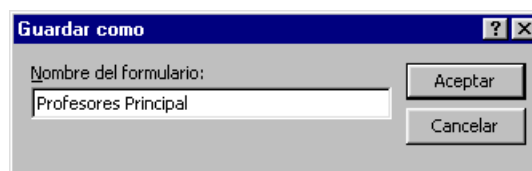


Figura 7.4.

6. Hacer clic en el botón **Vista Formulario** y elegir la opción **Vista Diseño**.

## C. Insertar el Subformulario en el formulario Principal

1. Hacer clic en el control **Asistente para controles** para activarlo si es necesario.



2.Hacer clic en el control **Subformularios/subinformes**. (Ver *Figura 7.5*).



Figura 7.5.

3.Hacer clic en la sección **Pie del formulario**.

4.Hacer clic en la opción **Formulario** del cuadro de diálogo del **Asistente** y elegir el formulario **Alumnos Subformulario**. (Ver *Figura 7.6*).

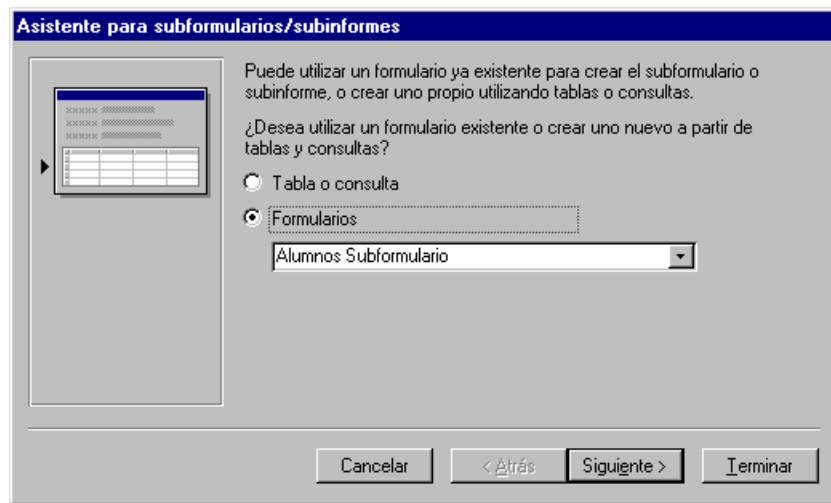


Figura 7.6.

5.Hacer clic en el botón **Siguiete**.

6.Verificar los vínculos establecidos por el **Asistente** y hacer clic en el botón **Siguiete**.

7.Hacer clic en el botón **Finalizar**.

Una vez realizados estos pasos, obtenemos el formulario con el subformulario que se muestra en la *Figura 7.7*.

**Profesores**

Cod\_Profesor: 0F10

Nomb\_Profesor: Antonia

Ape\_Profesor: Vázquez

Sueldo: 18500

Fijo: Sí

**Alumnos Subformulario**

Cod	Nomb_Alum	Ape_Alumno	Dirección	
047	Beatriz	Solbes	Ancha, 16	M
001	Cristina	González	Pinzón, 45	M

Registro: 1 de 2

Registro: 1 de 4

Figura 7.7.

## 8. Informes en Access.

Los informes son los objetos de Microsoft Access que permiten presentar en papel los datos que están almacenados en las tablas o consultas. También es posible imprimir los formularios y las hojas de datos, pero los informes permiten un mayor control sobre la apariencia final de la presentación de los datos y la posibilidad de presentar subtotales y resúmenes de totales por categorías de información, así como de calcular el porcentaje que presenta cada categoría sobre el total.

Un informe es la forma de recuperar y presentar la información almacenada en una base de datos de forma clara y atractiva. Se puede distribuir, pero no se pueden utilizar para modificar la información de la tabla o consulta en la que están basados.

De los distintos métodos que Access nos ofrece para presentar la información usaremos el más adecuado dependiendo de la tarea que se desee realizar:

- Para uso personal, las consultas y formularios serán suficientes para satisfacer cualquier necesidad de presentación de datos.
- Para presentar los datos a otras personas, el formato permitido por los informes es más claro y formal que una consulta o una hoja de datos.
- Para ver la información organizada en grupos podrá crear informes.
- Calcular subtotales, totales y total general junto con el porcentaje de totales para crear informes.

Al crear un informe hay que establecer la conexión entre los objetos gráficos denominados *controles* del informe y los datos de las tablas o consultas que representan.

Para familiarizarnos con los informes, y cómo se deben crear estos, vamos a realizar dos ejemplos basados en el supuesto planteado al principio de este capítulo.

**A. Primero crearemos un informe de la tabla PROFESORES utilizando el Asistente Autoinforme.**

Para realizar este informe hay que seguir los siguientes pasos:

1. Hacer clic en el botón **Abrir base de datos**.
2. Seleccionar la unidad y la carpeta donde se encuentra almacenada la base de datos CLASES.
3. Seleccionar el nombre de la base de datos y hacer clic en el botón **Abrir**.
4. Hacer clic en la ficha **Informes**.
5. Hacer clic en el botón **Nuevo objeto** y elegir la opción **Autoinforme**.
6. Seleccionar la tabla **PROFESORES**. (Ver *Figura 8.1*). Pulsar **Aceptar**.

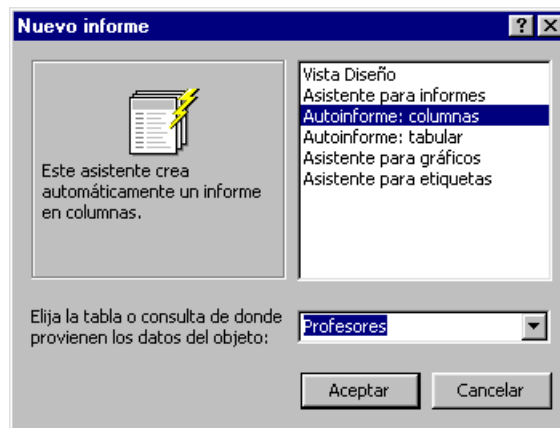


Figura 8.1.

7. Hacer clic en el botón **Guardar** para guardar el informe creado. Después de lo cuál aparecerá la ventana que se muestra en la *Figura 8.2*.

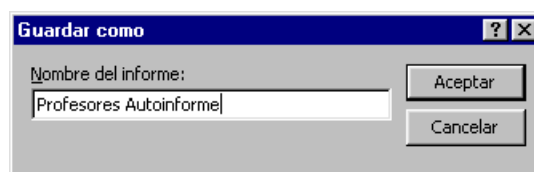


Figura 8.2.

8. Escribir el nombre **Profesores Autoinforme.**

9.Hacer clic en el botón **Aceptar**.

10.Hacer clic en el botón **Cerrar**.

El resultado de realizar estos pasos, es decir, el informe resultante de profesores tendrá una apariencia similar a la que se muestra en la *Figura 8.3*.

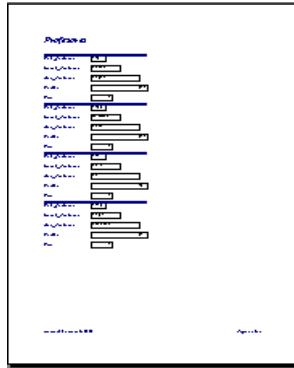


Figura 8.3.

**B. Ahora crearemos un informe agrupado por el campo Cod\_Curso y que muestre los campos Nomb\_Alumno, Ape\_Alumno, Teléfono y ordenado según el campo Ape\_Alumno.**

Para realizar el informe requerido realizaremos los siguientes pasos:

1.Hacer clic en el botón **Abrir base de datos**.

2. Seleccionar la unidad y la carpeta donde se encuentra almacenada la base de datos CLASES.

3. Seleccionar el nombre de la base de datos y hacer clic en el botón **Abrir**.

4.Hacer clic en la ficha **Informe**.

5.Hacer clic en el botón **Nuevo**.

6.Elegir la opción **Asistente para informes**.

7.En el recuadro **Tablas/Consultas** elegir la tabla ALUMNOS.

8.Hacer clic en el botón **Aceptar**. (Ver *Figura 8.4*).

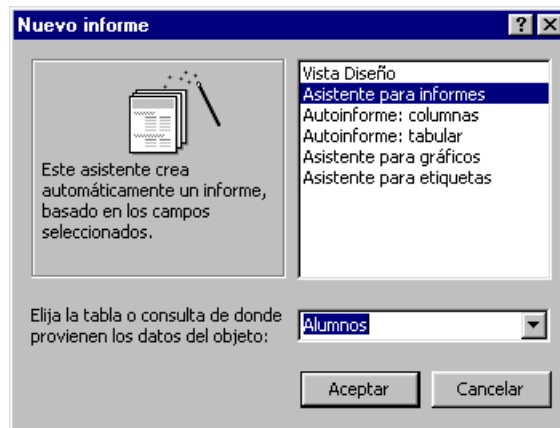


Figura 8.4.

9. Seleccionar el campo **Cod-curso** y hacer clic en el botón >.

10. Repetir el paso 9 para los campos **Nom-alumno**, **Ape-alumno** y **Teléfono**.

11. Hacer clic en el botón **Siguiente**. (Ver Figura 8.5).

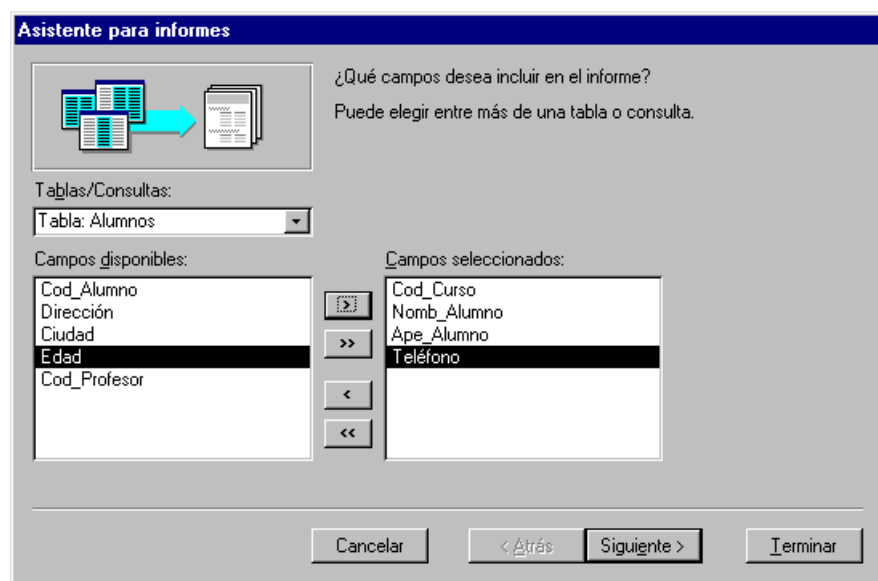


Figura 8.5.

12. Seleccionar el campo **Cod-Curso** y hacer clic en el botón > para agrupar.

13. Hacer clic en el botón **Siguiente**.

14. Seleccionar el campo **Ape-alumno**. (Ver Figura 8.6).

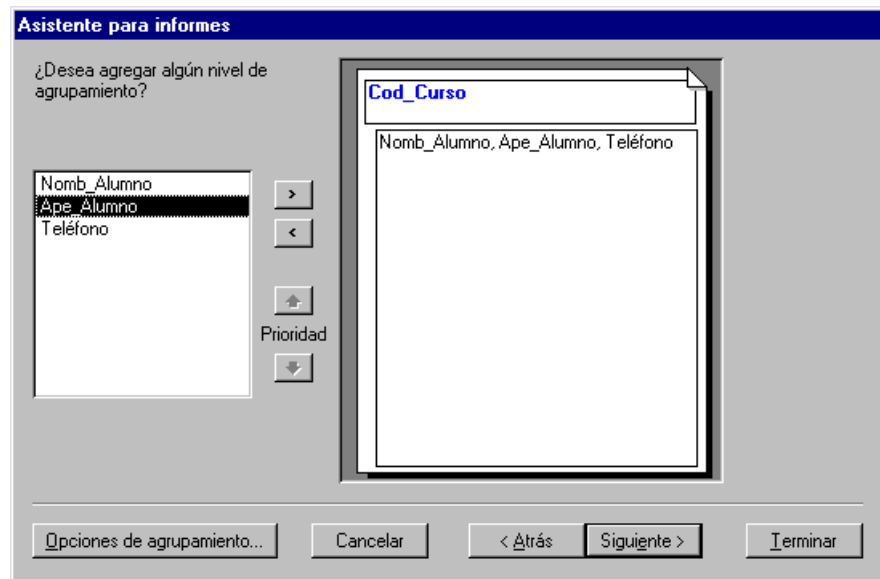


Figura 8.6.

15. Hacer clic en el botón **Siguiente**.
16. Elegir la distribución que desee. Hacer clic en el botón **Siguiente**.
17. Elegir el estilo que desee. Hacer clic en el botón **Siguiente**.
18. Hacer clic en el botón **Finalizar**.
19. Hacer clic en el botón **Guardar**.
20. Escribir el nombre LISTADO DE ALUMNOS POR CURSOS como nombre del informe.
21. Pulsar el botón **Terminar**.

## • BIBLIOGRAFÍA

Amescua, A., García, L., Martínez, P., Díaz, P., (1995). Ingeniería del software de gestión : análisis y diseño de aplicaciones. Ed. Paraninfo.

De Miguel, A., Piattini, M., Marcos, E., (1999). Diseño de base de datos Relacionales. Ed. RAMA.

De Miguel, A., Martínez, P., Castro, E., Caverro, J.M., Cuadra, D., Iglesias, A., Nieto, C., (2001). Diseño de Bases de Datos: Problemas resueltos. Ed. RAMA.

Elsuari, R., Navathe, S., (1997). Sistemas de bases de datos: conceptos fundamentales. Ed. Addison-Wesley Iberoamericana, 2ª Edición.

## GLOSARIO DE TÉRMINOS

Administrador de Bases de Datos: encargado de asegurar la disponibilidad, confidencialidad y la integridad de los datos.

Base de Datos: colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada.

Hardware: equipo físico de un ordenador formado por la unidad central de proceso (CPU) y los periféricos.

Lenguaje de manipulación navegacional: es un lenguaje para consultar y actualizar datos en la base de datos al cual se le debe indicar el camino para llegar al dato con el que se desea operar.

Lenguaje de manipulación de especificación: es un lenguaje para consultar y actualizar datos en la base de datos al cual se le debe indicar una condición para llegar al dato con el que se desea operar.

Metodología: es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de un producto software.

Modelo de Datos (MD): instrumento en el que nos apoyamos para obtener una estructura de datos de acuerdo con en UD que queremos almacenar.

**Modelo Datos Entidad/Interrelación:** modelo de datos conceptual compuesto de entidades, interrelaciones, atributos y dominios.

**Modelo Datos Relacional:** modelo de datos lógico compuesto de relaciones, atributos y dominios.

**Sistema de Gestión de Base de Datos (SGBD):** conjunto coordinado de programas, procedimientos, lenguajes, herramientas, etc., que suministra los medios necesarios para describir, actualizar y administrar los datos de una BD.

**Sistema de Información (SI):** un conjunto de personas, procedimientos y equipos diseño, construido y gestionado para tratar la información de acuerdo a las necesidades de la organización.

**Software:** equipo lógico de un ordenador, compuesto por programas, lenguajes de programación, documentación, etc.

**SQL:** Lenguaje de consulta estructurado (Structured Query Language), que nos sirve para definir y manipular los datos de una base de datos relacional.

**Universo del Discurso (UD):** parcela del mundo real que queremos almacenar en una base de datos.