BASES DE DATOS

Índice:

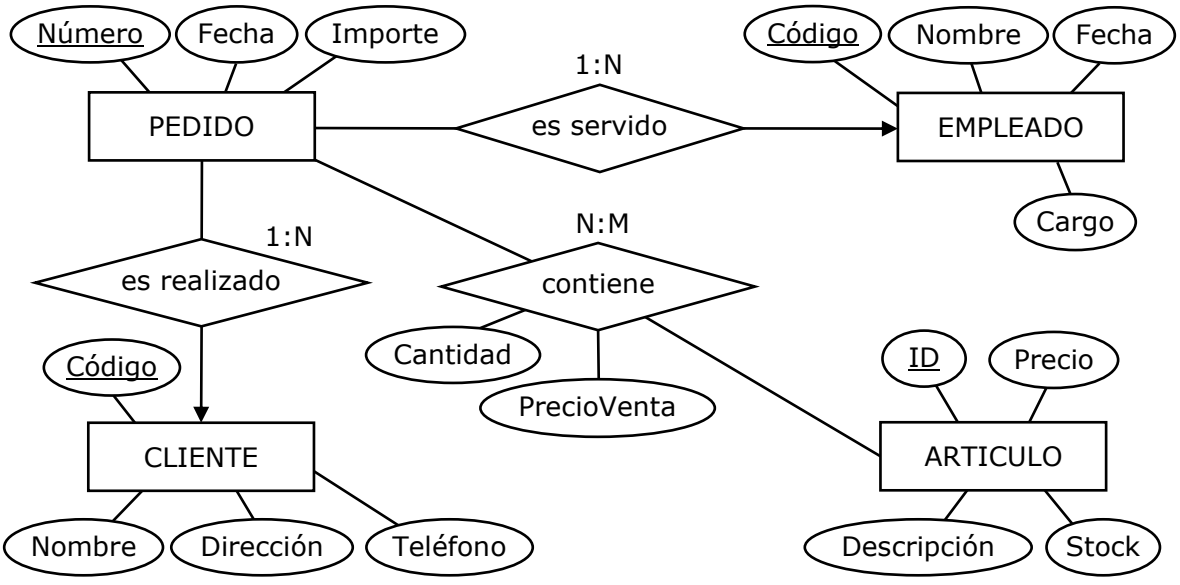
Ejercicio 1 – Modelo Entidad Relación. Tienda

Crear el diagrama entidad-relación para una base de datos en la que se desea almacenar datos sobre los pedidos de una tienda, teniendo en cuenta lo siguiente:

- Cada pedido tiene un número único, una fecha y un importe total.
- Cada pedido es servido por un empleado del que se guardará su código de empleado (único), nombre, fecha de incorporación a la empresa y cargo que ocupa.
- Cada pedido es realizado por un único cliente, del que se guardará un código de cliente (único), nombre, dirección y teléfono.
- Por último, se quieren guardar los artículos incluidos en cada pedido. Para ello, en la base de datos también se guardarán los datos de los artículos que están a la venta. Estos datos son: un identificador único de artículo, descripción, precio actual y stock.
- Para cada producto incluido en cada pedido, anotaremos la cantidad de producto servido en cada pedido y el precio al que ha sido vendido, que no tiene por qué coincidir con el precio actual almacenado para cada producto.

Solución

Existen varias notaciones para representar diagramas E/R (Chen, Pata de gallo, IDEF1X y Rein85). El más extendido es el modelo Chen, pero éste también tiene algunas variaciones, por ejemplo, para representar la cardinalidad de las relaciones se puede usar la expresión (mínimo, máximo) en cada extremo de la relación o se pueden usar flechas, donde un extremo con flecha indica que la cardinalidad es "uno" y sin flecha significa "muchos". Este ejercicio se resuelve usando el modelo "Chen" con flechas para representar la cardinalidad.

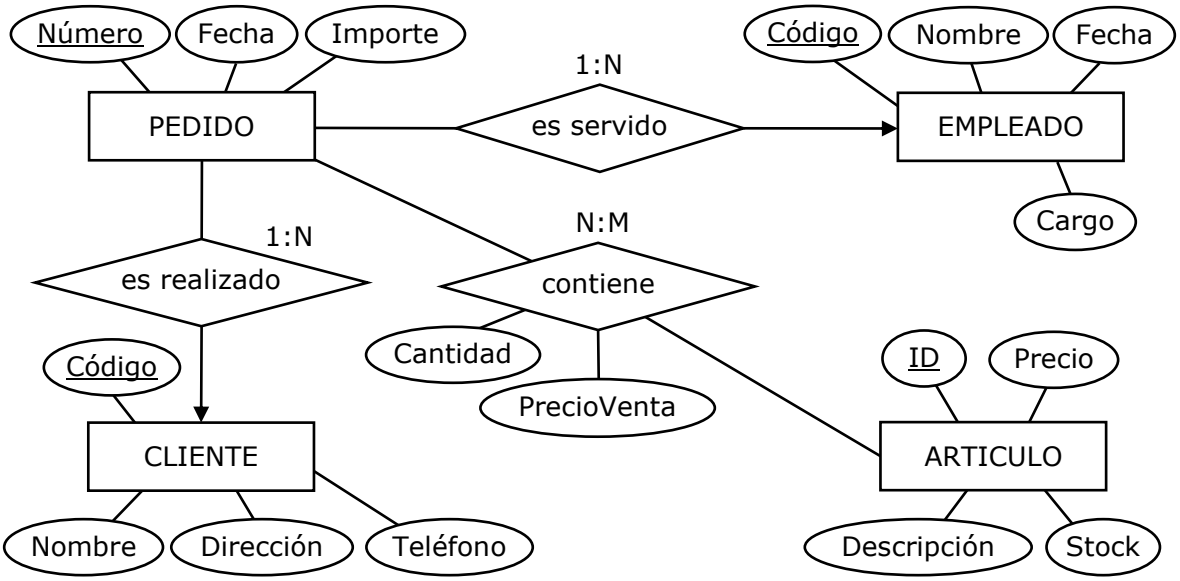


Ejercicio 2 – Modelo Relacional. Tienda

Transformar el modelo E/R del primer ejercicio al Modelo Relacional.

Consideraciones previas

El modelo relacional almacena todos los datos en relaciones (tablas) junto con sus atributos. Partimos del modelo E/R del ejercicio 1 que es el siguiente:



En el esquema se puede observar que hay 3 relaciones con 2 tipos de cardinalidad. La forma de actuar según la cardinalidad es la siguiente:

- **Relación 1:N (uno a muchos):** La clave principal de la entidad con cardinalidad 1 pasa a formar parte de la tabla con cardinalidad N como clave ajena (clave foránea). Los atributos de la relación, si los hubiera, pasarían a ser atributos de la tabla N.
- **Relación N:M (muchos a muchos):** Se crea una nueva relación (tabla) que contendrá las claves de las 2 entidades, además de los atributos de la relación. El conjunto de claves ajenas forma la clave principal de la nueva tabla.
- **Relación 1:1 (uno a uno):** En este caso, existen 3 posibilidades:
  - Cardinalidad (0,1) en ambos extremos: Se crea una tabla nueva donde se incluirán las claves de las 2 entidades como claves ajenas. La clave primaria de la nueva tabla será una (cualquiera) de las 2.
  - Cardinalidad (0,1) en un extremo y (1,1) en el otro: La clave principal de la entidad con cardinalidad (1,1) pasa a la otra entidad como clave ajena.
  - Cardinalidad es (1,1) en ambos extremos: Generalmente se pasa la clave de una entidad cualquiera a la otra como clave ajena. Sin embargo, también se pueden pasar las claves primarias de ambas entidades como claves ajenas en las dos tablas.

Solución

EMPLEADO (Código, Nombre, Fecha, Cargo)

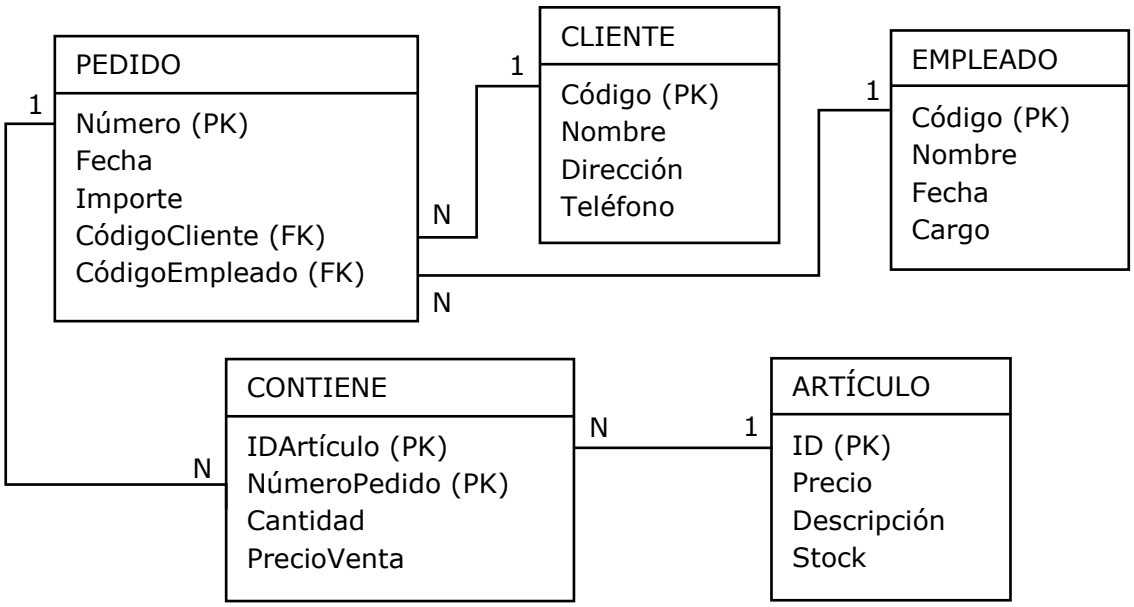
CLIENTE (Código, Nombre, Dirección, Teléfono)

PEDIDO (Número, Fecha, Importe, CódigoCliente, CódigoEmpleado)

CONTENIDO (IDArtículo, NúmeroPedido, Cantidad, PrecioVenta)

ARTÍCULO (ID, Precio, Descripción, Stock)

Otro tipo de modelo relacional es el diagrama de clase:



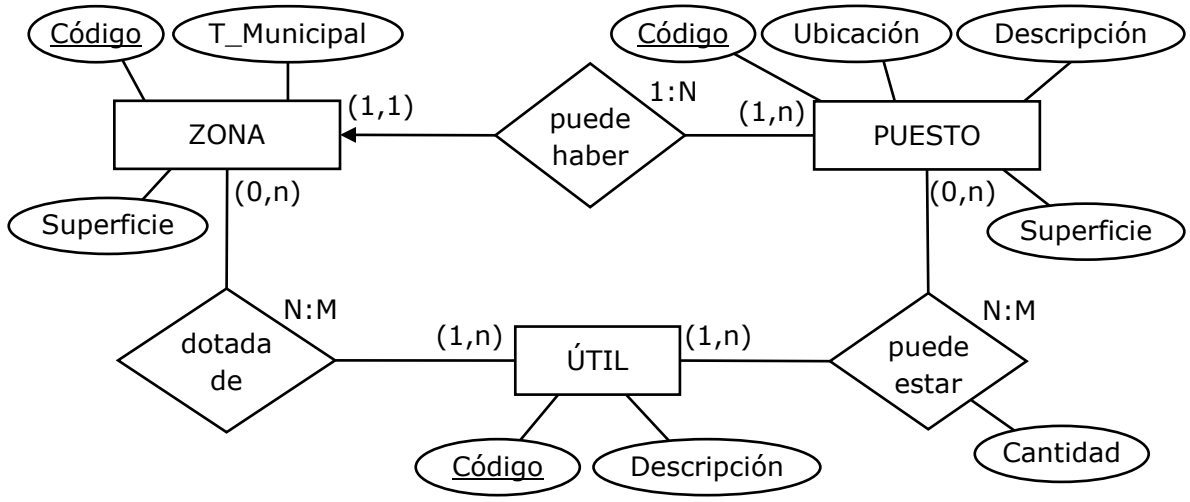
Ejercicio 3 – Modelo E/R y Relacional. Extinción de incendios

Crear el diagrama entidad-relación y relacional para una base de datos en el que se desea almacenar datos sobre una organización de extinción de incendios.

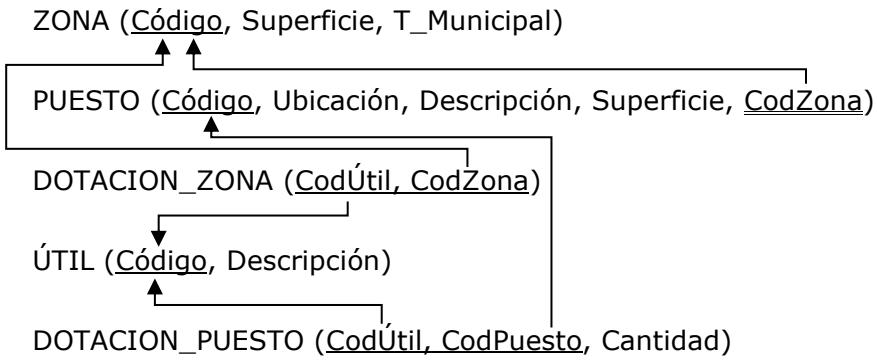
- Esta organización controla distintas zonas geográficas. Para cada zona almacenamos un código único de zona, su superficie y el término municipal al que está asignada.
- En cada zona puede haber distintos puestos, de los que se guardan un código único, una ubicación, una descripción y su superficie.
- Cada zona está dotada de distintos útiles de extinción de incendios. Cada uno de estos útiles tiene un código y descripción.
- Un mismo útil de extinción (por ejemplo, una pala) puede estar en varios puestos. En la base de datos almacenamos la cantidad de un útil que hay en cada puesto.

Solución

En este ejercicio se diseñará el diagrama usando el modelo Chen con flechas y, además, se indicará la cardinalidad de cada extremo usando la expresión (mínimo, máximo).



Y el modelo relacional es el siguiente:



Ejercicio 4 – Modelo E/R y Relacional. Clínica veterinaria

Queremos informatizar una pequeña clínica veterinaria donde los vecinos traen a sus mascotas. Cuando un vecino acude a la consulta por primera vez, se recoge la siguiente información:

- Del propietario de la mascota se registra su DNI (único), apellidos, nombre, dirección y teléfono. Hay que tener en cuenta que algunos vecinos llevan animales que están abandonados y, por tanto, no tienen propietario.
- Del animal se registra su id (único), nombre, fecha de nacimiento, raza, peso, altura, vacunas y otros datos recogidos como descripción (por ejemplo, una mancha en la pata).

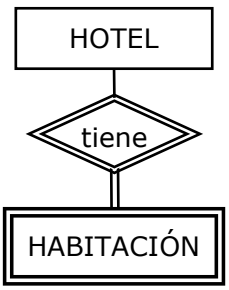
Tanto la primera vez como cuando el propietario lleva de nuevo a su mascota para una consulta, se necesita guardar la fecha de la consulta, motivo de la consulta, diagnóstico y tratamiento.

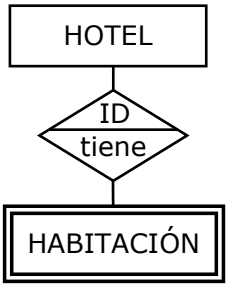
Consideraciones previas

En este ejercicio se puede ver que la entidad "Consulta" es una entidad débil por identificación. Una entidad débil puede ser por identificación cuando necesita la clave principal de la otra entidad para identificarse o por existencia cuando su existencia depende totalmente de la otra entidad.

Para representar las entidades débiles se utiliza doble contorno y para representar la relación que une con la entidad fuerte existen varias formas válidas:

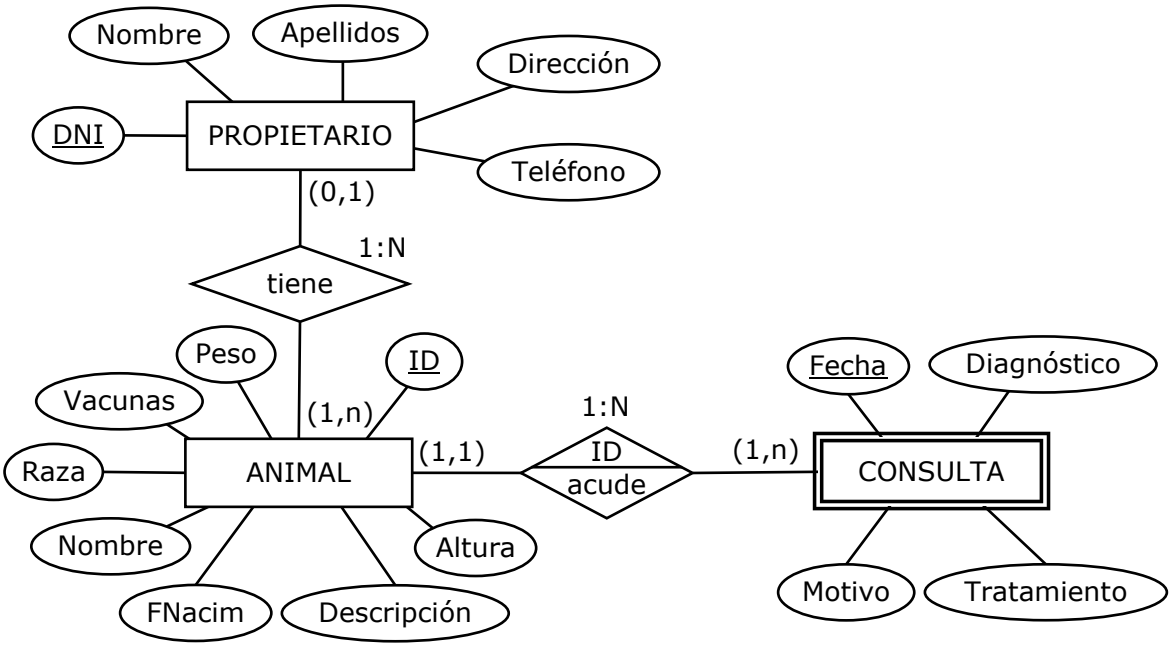
- Doble contorno en la relación y una doble línea que una con la entidad débil.
- Contorno simple, pero poniendo dentro de la relación "ID" si es por identificación o "EX" si es por existencia.





Solución

A partir de ahora, no se usarán las flechas para indicar la cardinalidad y en su lugar se usará solamente la expresión (mínimo, máximo) en cada extremo de las relaciones.



Al pasarlo al modelo relacional, hay que tener en cuenta que "Consulta" es una entidad débil por identificación. Las entidades débiles por identificación adquieren las claves principales de la entidad fuerte y formarán parte de la clave principal (si es débil por existencia no adquiere nada y se trata igual que una entidad normal).

PROPIETARIO (DNI, Nombre, Apellidos, Dirección, Teléfono)

ANIMAL (ID, FNacim, Nombre, Raza, Peso, Altura, Vacunas, Descripción, DNIPropiet)

CONSULTA (FechaConsulta, IDAnimal, Motivo, Diagnóstico, Tratamiento)

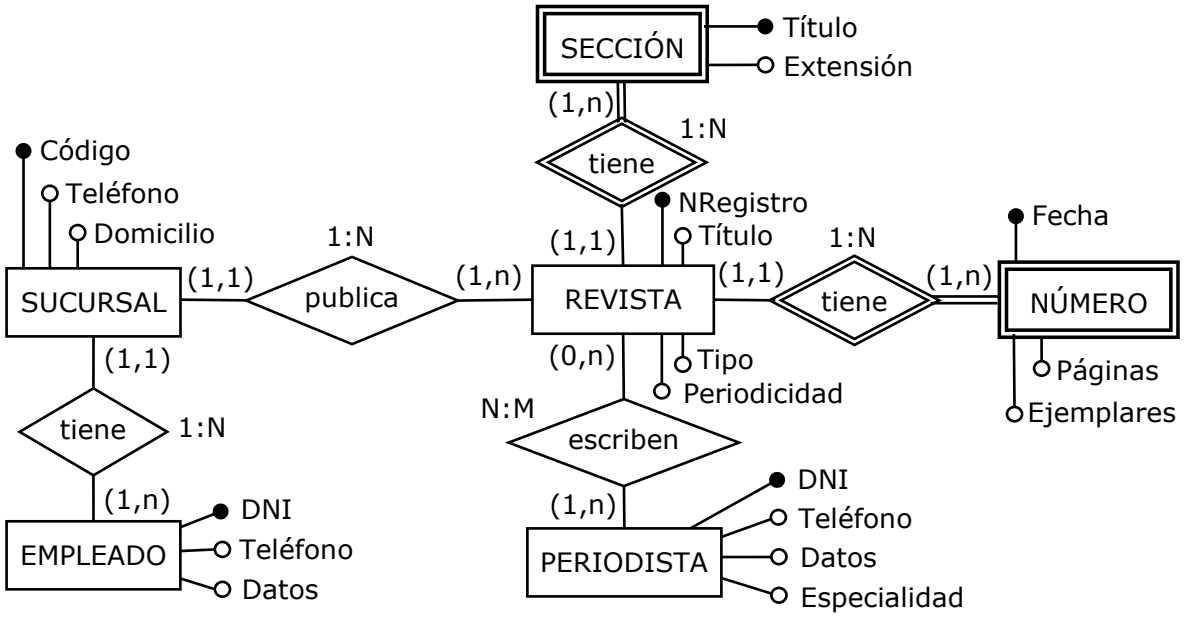
Ejercicio 5 – Modelo E/R y Relacional. Cadena editorial

Tenemos la siguiente información sobre una cadena editorial.

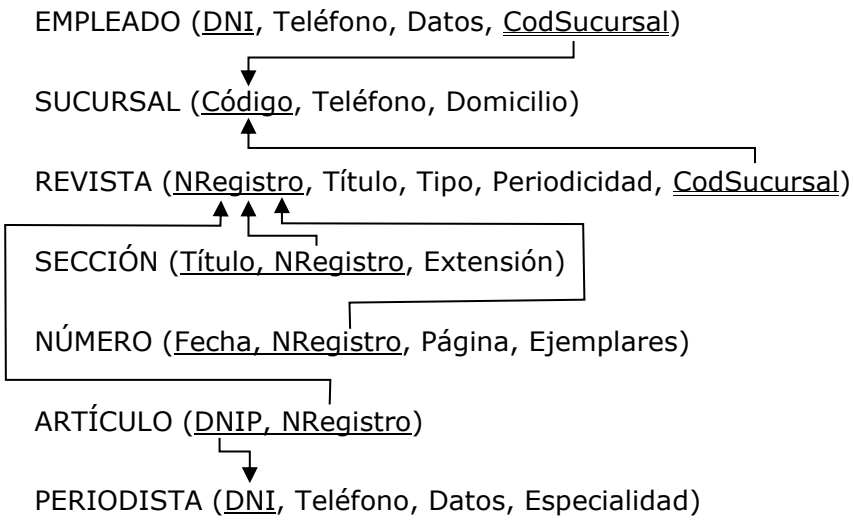
- La editorial tiene varias sucursales, con domicilio, teléfono y código de sucursal.
- Cada sucursal tiene varios empleados, de los cuales tenemos sus datos personales, DNI y teléfono. Un empleado trabaja en una única sucursal.
- En cada sucursal se publican varias revistas, de las que almacenamos su título, número de registro, periodicidad y tipo.
- La editorial tiene periodistas (que no trabajan en las sucursales) que pueden escribir artículos para varias revistas. Almacenamos los mismos datos que para los empleados, añadiendo su especialidad.
- Guardaremos también las secciones fijas de cada revista, que tendrán su título y una extensión.
- Para cada revista almacenamos información de cada número, que incluirá la fecha, número de páginas y el número de ejemplares vendidos.

Solución

Otra forma válida para representar los atributos es poniendo círculo al lado de cada atributo y, si es clave, un círculo coloreado de negro.



Y el modelo relacional es el siguiente:



Ejercicio 6 – Modelo E/R y Relacional. Pequeña empresa

Se quiere crear una base de datos para una pequeña empresa que contenga la siguiente información acerca de los clientes, artículos y pedidos:

- De cada cliente se quiere registrar el número de cliente (único), direcciones de envío (pueden ser varias por cliente), saldo, límite de crédito (depende del cliente, pero en ningún caso debe superar los 18.000€) y descuento.
- De cada artículo se quiere registrar el número de artículo (único) y descripción.
- De cada pedido se quiere registrar la fecha del pedido y la dirección de envío. También se almacenará la cantidad de cada artículo incluido en cada pedido.

Además, se ha determinado que se debe de almacenar la siguiente información de las fábricas: número de fábrica (único), teléfono de contacto y número de artículos que ha provisto la fábrica en total. Como cada fábrica puede fabricar distintos artículos y cada artículo puede ser fabricado por distintas fábricas, se quiere almacenar la cantidad de artículos (existencias) que ha provisto cada una.

También, por información estratégica, se podría incluir información de fábricas alternativas respecto de las que ya fabrican artículos para esta empresa, es decir, incluir información de fábricas que no crean artículos, pero que podrían hacerlo en el futuro.

Nota: Una dirección se entenderá como Nº, calle, comuna y ciudad.

Consideraciones previas

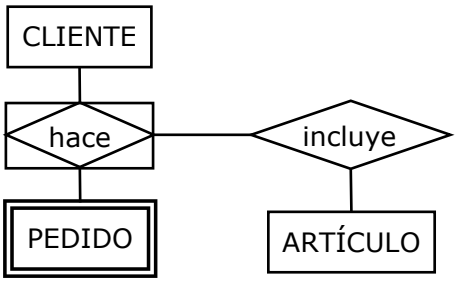
En este ejercicio se puede observar que hay 3 tipos nuevos de atributos:

- "Dirección de envío" es un **atributo compuesto** y se tiene en cuenta solo conceptualmente. Se podría omitir y poner directamente "Nº", "Calle", "Comuna" y "Ciudad" como atributos normales.

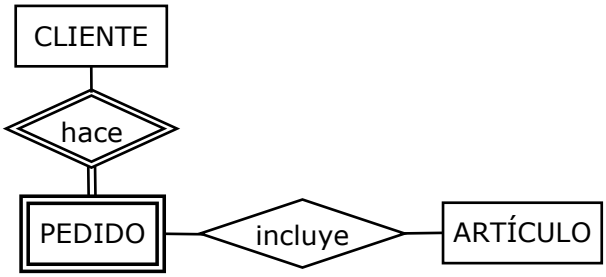
- "Direcciones de envío" es un **atributo compuesto multivaluado**. Los atributos multivaluados son aquellos que pueden tener varios valores.
- "Nº artículos provistos" es un **atributo derivado** ya que su valor puede derivarse u obtenerse de los valores de otros atributos o entidades. En este caso, se puede obtener sumando todas las existencias provistas de cada artículo por cada fábrica. Otro ejemplo de atributo derivado que no tiene que ver con el ejercicio es "Edad" que puede derivarse de "Fecha de Nacimiento".

También hay que tener en cuenta que la entidad débil "Pedido" está relacionada con 2 entidades (Cliente y Artículo) y, cuando esto ocurre, hay varias formas de representarlo:

- Algunos autores consideran que una entidad débil no puede estar relacionada con más entidades excepto con la entidad fuerte de la que depende y, en el caso de que existan más relaciones, éstas no la tendrán con la entidad débil, sino con la propia relación que une la entidad débil con la fuerte.
- Otros autores dicen que una entidad débil sí puede estar relacionada con más entidades, pero tiene que quedar claro quién es la entidad fuerte de la que depende poniéndole a la relación doble contorno o el identificador "ID" o "EX".

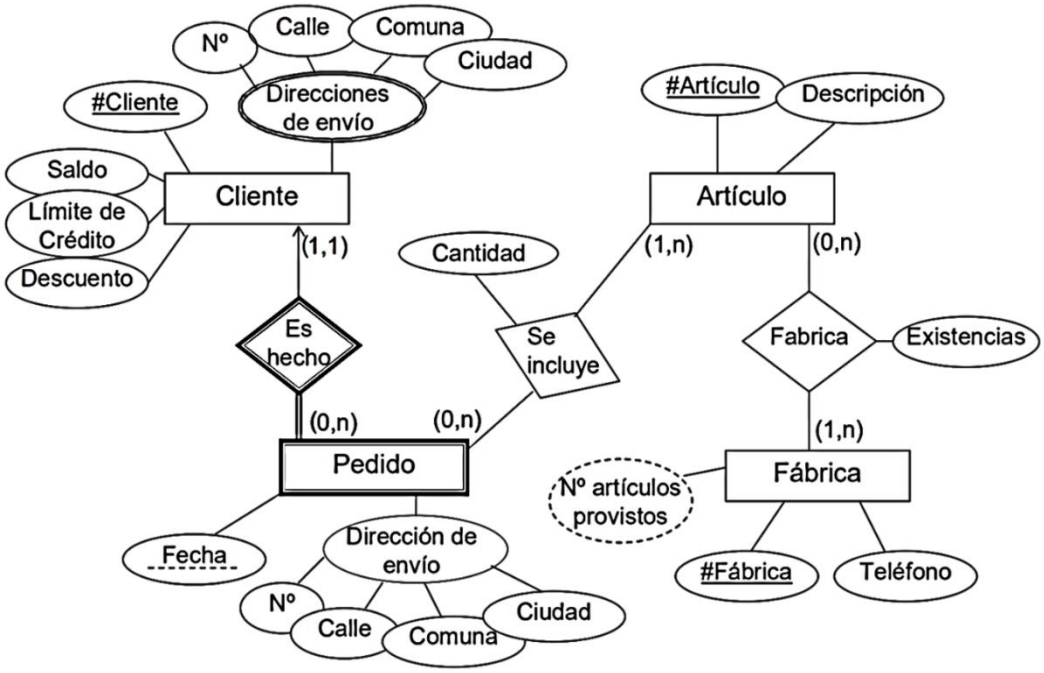


Se relacionan con la entidad fuerte



Se relacionan con la entidad débil

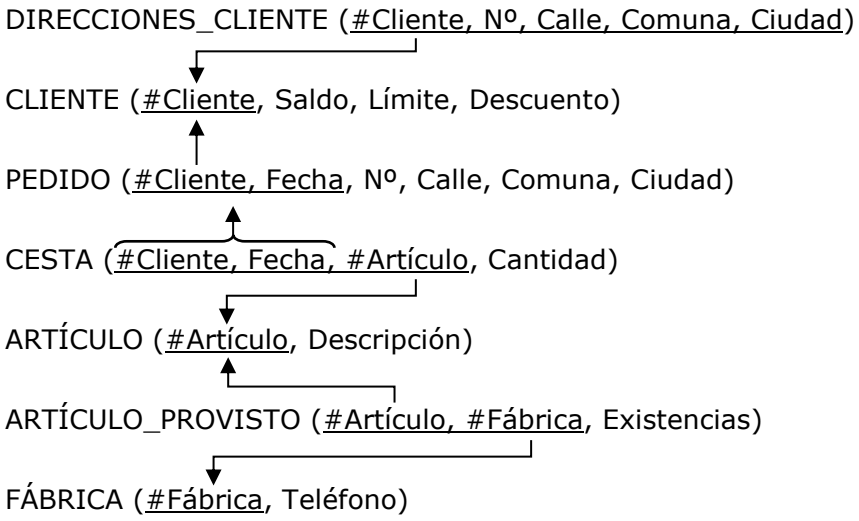
Solución



Antes de pasarlo al modelo relacional hay que tener en cuenta lo siguiente:

- Los atributos compuestos, como "Dirección de envío", no se reflejan en las tablas, solo se ponen los atributos simples que contiene (Nº, Calle, etc.).
- Para los atributos multivaluados, como "Direcciones de envío", se crea una nueva relación formada con la clave principal de la entidad y el/los atributo/s multivaluado/s, siendo todos claves principales de la nueva relación. Un atributo multivaluados equivale a poner una entidad débil con relación 1:N.
- Los atributos derivados, como "Nº artículos provistos", no formarán parte del modelo relacional resultante, quedando eliminados en la parte del diseño.

Con esto, obtenemos el modelo relacional:



Ejercicio 7 – Modelo E/R y Relacional. Cadena de supermercados

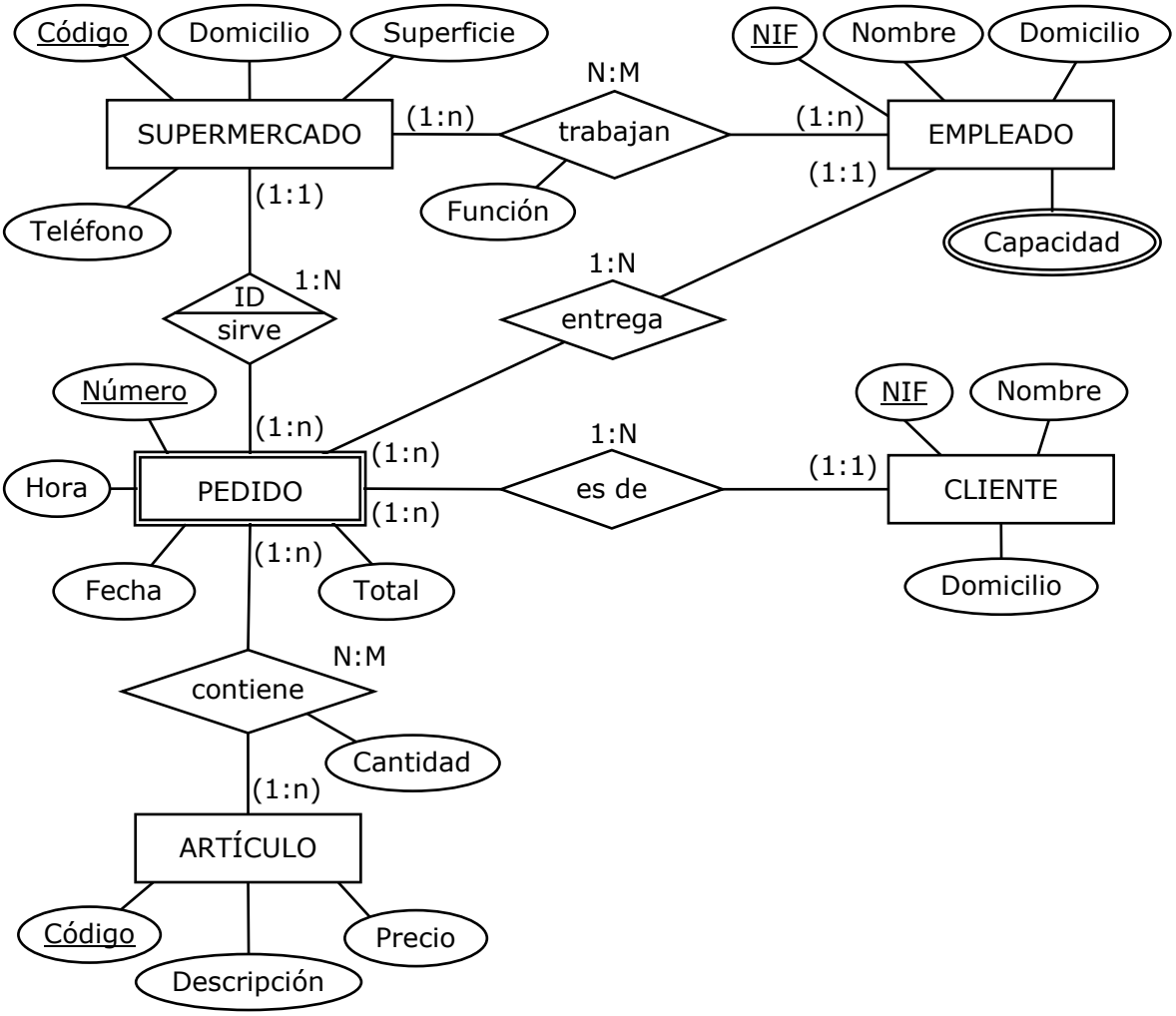
Una cadena de supermercados desea crear una base de datos para su gestión. La cadena tiene varios supermercados, cada uno tiene un código único, un domicilio, una superficie y un teléfono. La cadena tiene empleados, que pueden trabajar en varios supermercados. Para cada empleado, tenemos su NIF, nombre, domicilio y capacidades. Cada empleado desempeña una función en cada supermercado en el que trabaja.

Los supermercados sirven pedidos a domicilio. Para cada pedido, se anota un número de pedido (que puede repetirse en distintos supermercados), la fecha, la hora y el total. El pedido incluye el código, descripción, precio y cantidad de artículos suministrados.

Cada pedido se sirve a un cliente, del que tenemos su NIF, nombre y domicilio, y es entregado por un empleado.

Solución

En este ejercicio la entidad "Pedido" es débil porque depende del supermercado para identificarse ya que su clave se puede repetir en distintos supermercados esta entidad débil tiene relación con varias entidades.



Con esto, obtenemos el modelo relacional:

CAPACIDADES (NIFEmpleado, Capacidad)

EMPLEADO (NIF, Nombre, Domicilio)

PUESTO (NIFEmpleado, CodigoSuperM, Función)

SUPERMERCADO (Código, Domicilio, Superficie, Teléfono)

PEDIDO (CodSuperM, Número, Fecha, Hora, Total, NIFEmpleado, NIFCliente)

COMPOSICIÓN (CodSuperM, NumPedido, CodArtículo, Cantidad)

ARTÍCULO (Código, Precio, Descripción)

CLIENTE (NIF, Nombre, Domicilio)

Ejercicio 8 – Modelo E/R y Relacional. Fábrica de pelotas

Solicitan nuestros servicios para resolver el almacenamiento de datos de un sistema de gestión de la producción de una fábrica de pelotas. La fábrica se compone de una serie de plantas, cada una identificada por un color. De las plantas conocemos la superficie en metros cuadrados y la lista de procesos que se llevan a cabo dentro de ellas; de estos procesos sólo conocemos su nombre y un grado de complejidad asociado.

Dentro de cada planta se encuentran las máquinas. Cada máquina es de una marca y un modelo, y se identifica por un número (único a lo largo de todas las plantas).

Cada máquina es operada por técnicos, debemos conocer en qué rango de fechas estuvieron los técnicos asignados a esa máquina, y además en qué turno (mañana, tarde o noche).

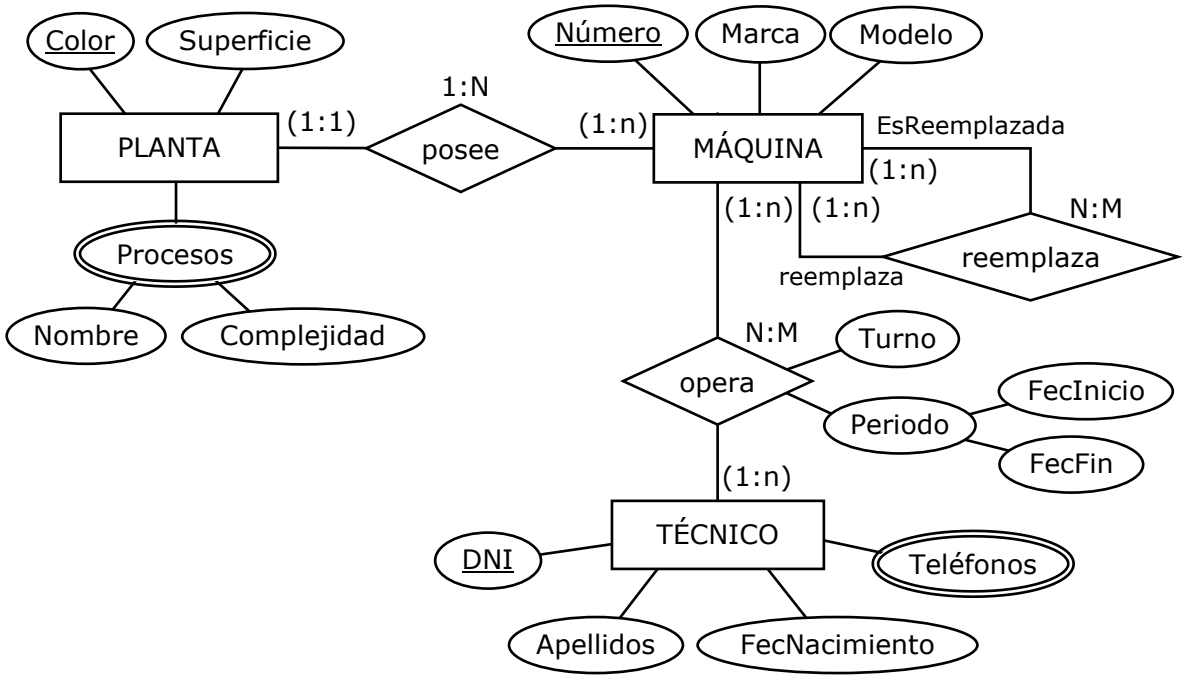
De los técnicos conocemos su DNI, nombre, apellidos y fecha de nacimiento, además de una serie de números telefónicos de contacto.

Existen ciertas situaciones (avería, mantenimiento, etc.) en las que una máquina sale del servicio y debe ser reemplazada por otra máquina. En este caso, nos interesa conocer cuál otra máquina ha tomado su lugar. Hay que tener en cuenta que, a lo largo de la vida útil de una máquina, ésta puede reemplazar y puede ser reemplazada por varias máquinas.

Solución

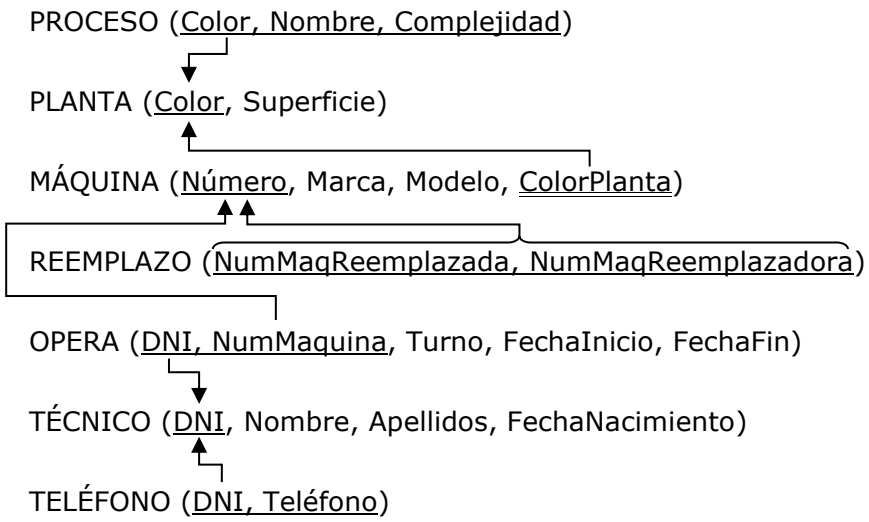
En este ejercicio, la entidad "Máquina" tiene una relación recursiva, también llamada relación reflexiva, ya que una máquina reemplaza a otra máquina.

El atributo "Procesos" se ha puesto como un atributo multivaluado y compuesto. Otra forma equivalente sería poner "Procesos" como una entidad débil de la entidad "Planta".



Para pasarlo al modelo relacional, hay que tener en cuenta las relaciones recursivas. Se pueden dar 2 situaciones:

- Cuando la relación recursiva es **1:N**, la clave principal de la entidad figurará 2 veces en la tabla, una como clave principal y otra como clave ajena que hace referencia a esa clave principal. Por ejemplo: Un empleado supervisa a otro, la tabla tendrá como clave principal el DNI del empleado supervisado y como clave ajena el DNI del empleado supervisor.
- Cuando la relación es **N:M**, se crea una nueva tabla cuya clave principal es el conjunto de claves foráneas. En este caso, la clave principal de "Reemplaza" lo forman la clave de la máquina que es reemplazada y la clave de la máquina que la reemplaza.



Ejercicio 9 – Modelo E/R y Relacional. Sistema de ventas

Se pide crear una base de datos que permita apoyar la gestión de un sistema de ventas. La empresa necesita llevar un control de proveedores, clientes, productos y ventas.

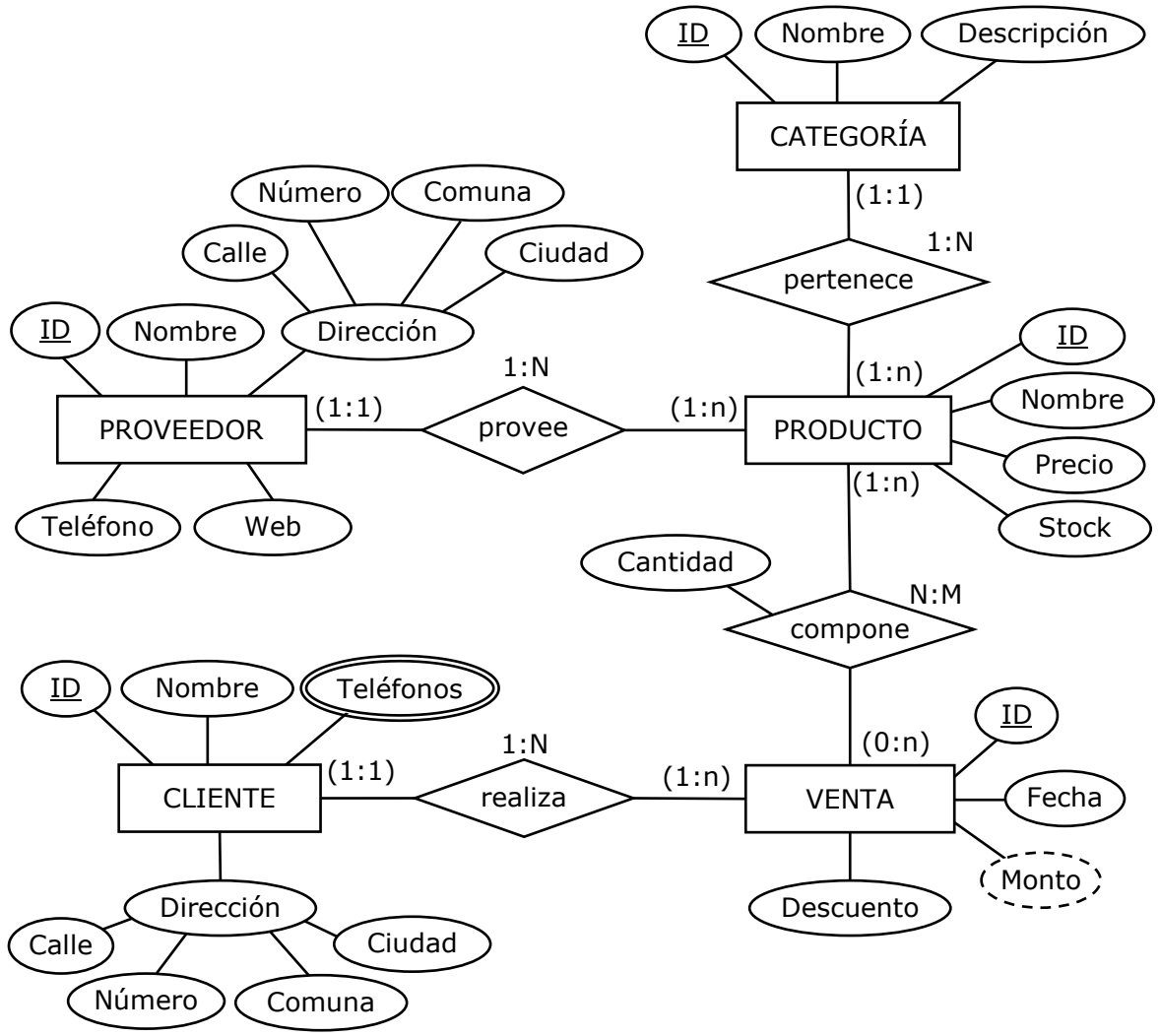
Un proveedor puede proveer de varios productos, pero un producto solamente puede ser provisto de un único proveedor. Del proveedor se quiere registrar un id (único), nombre, dirección, teléfono y página web. Del producto se registra un id (único), nombre, precio y stock. Cada producto pertenece a una única categoría. Una categoría tiene id (único), nombre y descripción.

De las ventas se registra un id (único), fecha, descuento y monto final. Una venta está compuesta de al menos un producto, del que se guardará la cantidad.

Un cliente puede realizar varias ventas. Del cliente almacenamos un id (único), nombre, dirección y puede tener varios teléfonos de contacto.

La dirección en todos los casos se entiende por calle, número, comuna y ciudad.

Solución



Y obtenemos el modelo relacional:

PROVEEDOR (<u>ID</u>, Nombre, Teléfono, Web, Calle, Número, Comuna, Ciudad)

CATEGORÍA (<u>ID</u>, Nombre, Descripción)

PRODUCTO (<u>ID</u>, Nombre, Precio, Stock, <u>IDCategoría</u>, ID<u>Proveedor</u>)

DETALLEVENTA (<u>IDVenta, IDProducto</u>, Cantidad)

VENTA (<u>ID</u>, Fecha, Descuento, <u>IDCliente</u>)

CLIENTE (<u>ID</u>, Nombre, Calle, Número, Ciudad, Comuna)

TELEFONOSCLIENTE (<u>IDCliente, Teléfono</u>)

Ejercicio 10 – Modelo E/R y Relacional. Biblioteca

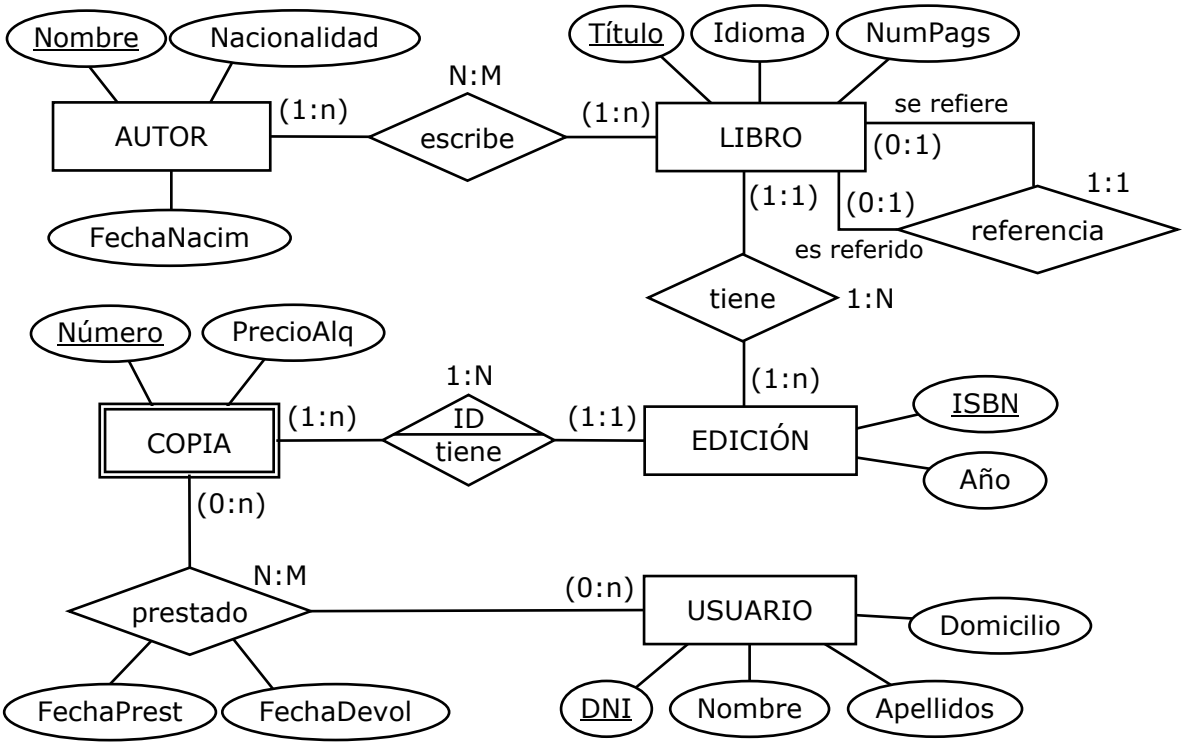
Nos piden modelar el sistema de una biblioteca, brindándonos la siguiente información. Los libros son escritos por autores de los cuales conocemos su nombre, nacionalidad y fecha de nacimiento. Los nombres de los autores no pueden repetirse. Además, sabemos que los libros cuentan con un título único, idioma y número de páginas. Adicionalmente, sabemos que cada libro tiene ediciones, de las cuales sabemos el año y el ISBN (único).

La biblioteca realiza préstamos de distintas ediciones a usuarios. De cada préstamo, sabemos el número de la copia del libro prestado y el precio del alquiler, mientras que de los usuarios sabemos su DNI, su nombre y apellidos y domicilio. También queremos registrar la fecha del préstamo y la fecha de devolución de las transacciones realizadas.

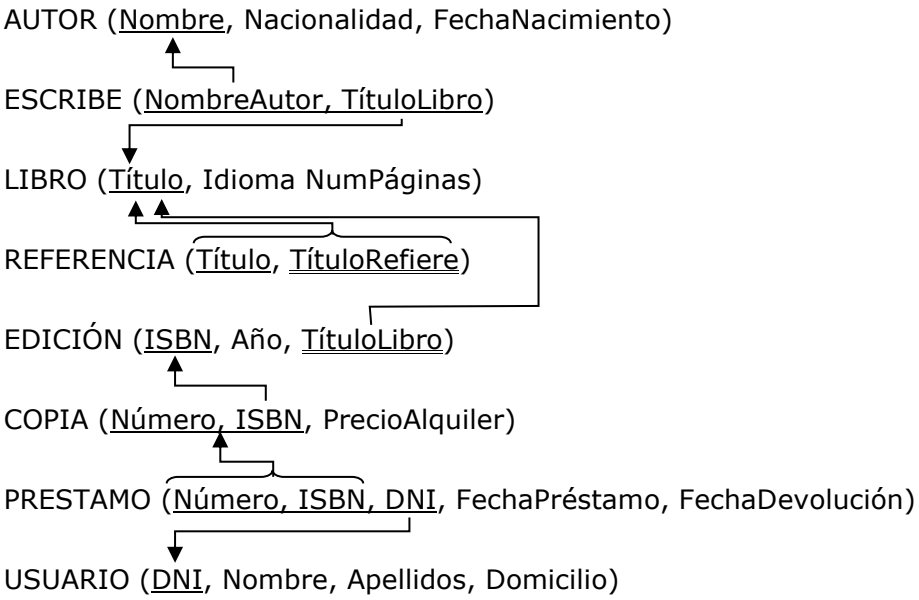
Tenga en cuenta la siguiente información adicional:

- Un autor escribe muchos libros y un libro puede ser escrito por muchos autores.
- Un libro puede tener muchas ediciones.
- Una edición tiene muchas copias, pero cada copia pertenece a una edición.
- Una copia pudo haber sido prestada a muchos usuarios y muchos usuarios pueden haber pedido la misma copia en momentos distintos.
- En algunos casos un libro puede hacer referencia a otro libro, pero solo a uno, lo mismo en el caso inverso.
- Las copias tienen un número único dentro de cada edición, pero el mismo puede repetirse dentro de otras ediciones.

Solución



Y obtenemos el modelo relacional:



Ejercicio 11 – Modelo E/R y Relacional. Farmacias

Debemos diseñar un sistema para registrar las farmacias en diferentes ciudades de nuestro país.

Sabemos que cada farmacia tiene un nombre (único en todo el sistema) y un domicilio. Cada farmacia se ubica en una sola ciudad, pero en una ciudad hay varias farmacias. De cada ciudad, sabemos el nombre, la provincia en la que se encuentra, la cantidad de habitantes y la superficie. Cada ciudad se identifica con el nombre y la provincia.

Conocemos también que cada farmacia puede tener un propietario, y que cada propietario tiene solamente una farmacia. Tenga en cuenta que puede haber farmacias sin propietario. De los propietarios, conocemos el DNI (único), su nombre y su domicilio, compuesto por calle, número, código postal y ciudad.

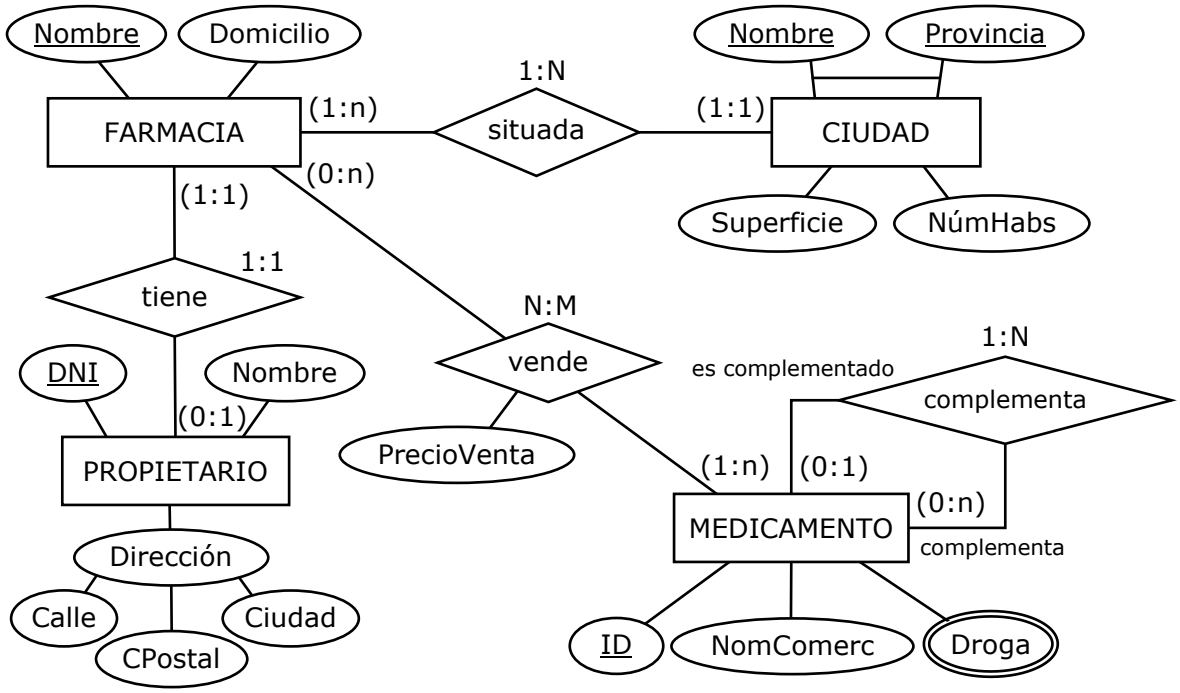
Cada farmacia vende varios medicamentos y un medicamento se vende en varias farmacias. De cada medicamento conocemos su id único, su nombre comercial y las drogas de las cuales se compone. Cada farmacia vende un medicamento a un precio determinado, que no necesariamente es el mismo en diferentes farmacias.

Como último requerimiento, un medicamento puede complementar a otros medicamentos, pero sabemos que cada medicamento puede ser complementado por un solo medicamento.

Consideraciones previas

En este ejercicio la entidad "Ciudad" tiene una clave compuesta por 2 atributos. En este caso, además de subrayar los 2 atributos, se recomienda unir las 2 líneas que van de los atributos a la relación con otra línea.

Solución



Obtenemos el modelo relacional:

CIUDAD (NombreCiudad, Provincia, NúmHabitantes, Superficie)

FARMACIA (NombreFarmacia, Domicilio, NombreCiudad, Provincia)

PROPIETARIO (DNI, NombreFarmacia, Nombre, Calle, CPostal, Ciudad)

VENTAMEDICAMENTO (IDMedicamento, NombreFarmacia, PrecioVenta)

MEDICAMENTO (IDMedicamento, NombreComercial, IDMedComplementario)

DROGASMEDICAMENTO (IDMedicamento, Monodroga)

Ejercicio 12 – Modelo E/R y Relacional. Página de deportes

Una web de estadísticas deportivas desea crear una BBDD para manejar la información de los partidos, equipos e hinchas que visitaron Brasil durante el mundial en el 2014.

Sabemos que los partidos tienen un ID que los identifica en el sistema, así como también la instancia del torneo que se jugó en dicho partido (fase de grupos, octavos, etc.), la duración y la fecha en la que se jugó, compuesta por el día y la hora, el árbitro que dirigió ese partido y los equipos que jugaron el mismo.

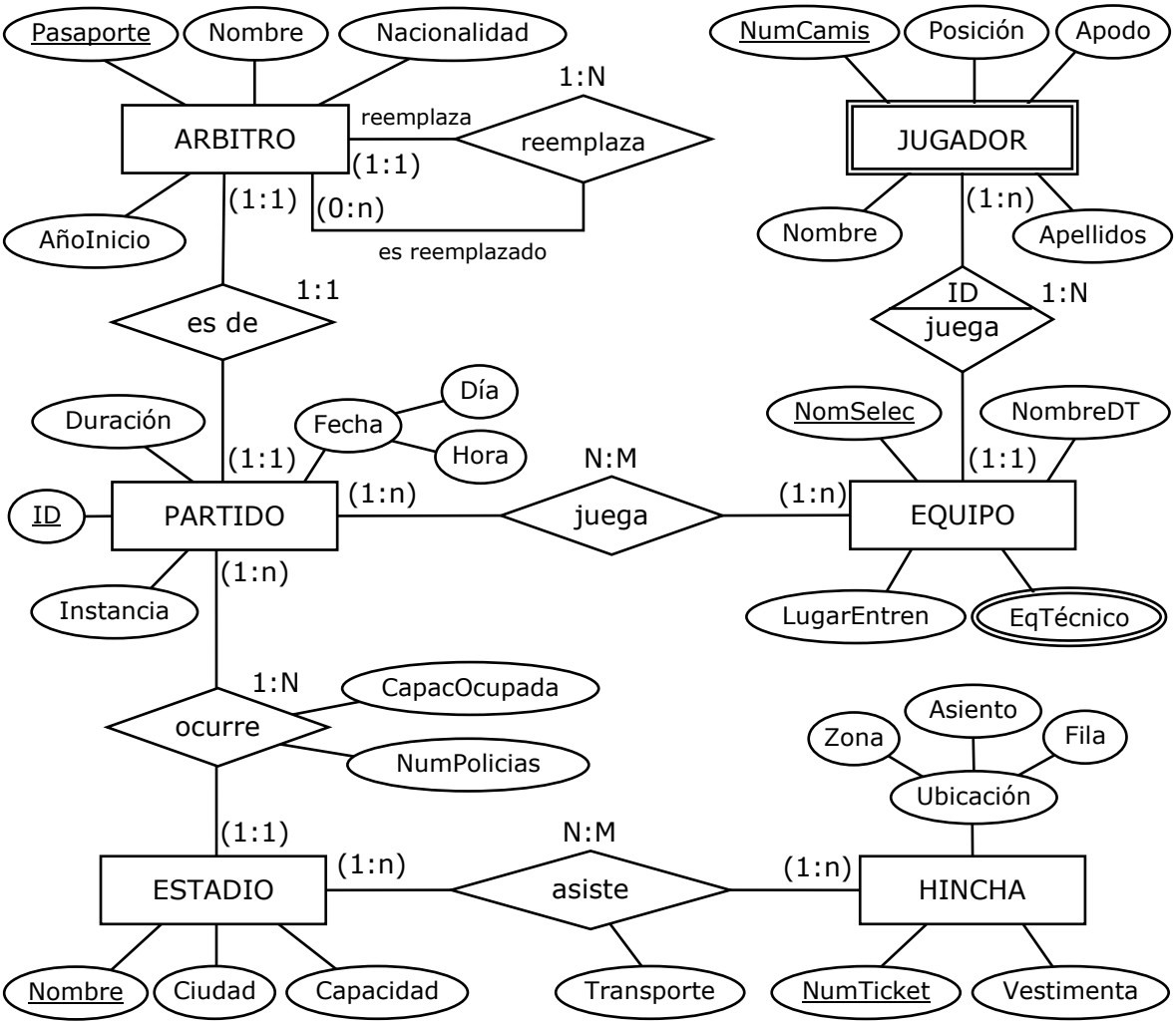
De los árbitros conocemos su pasaporte, nacionalidad, el año en el que inició la actividad y su nombre, también qué árbitro reemplaza a cuál en caso de lesión o enfermedad donde un árbitro puede ser reemplazado por varios y un árbitro reemplaza solo a uno.

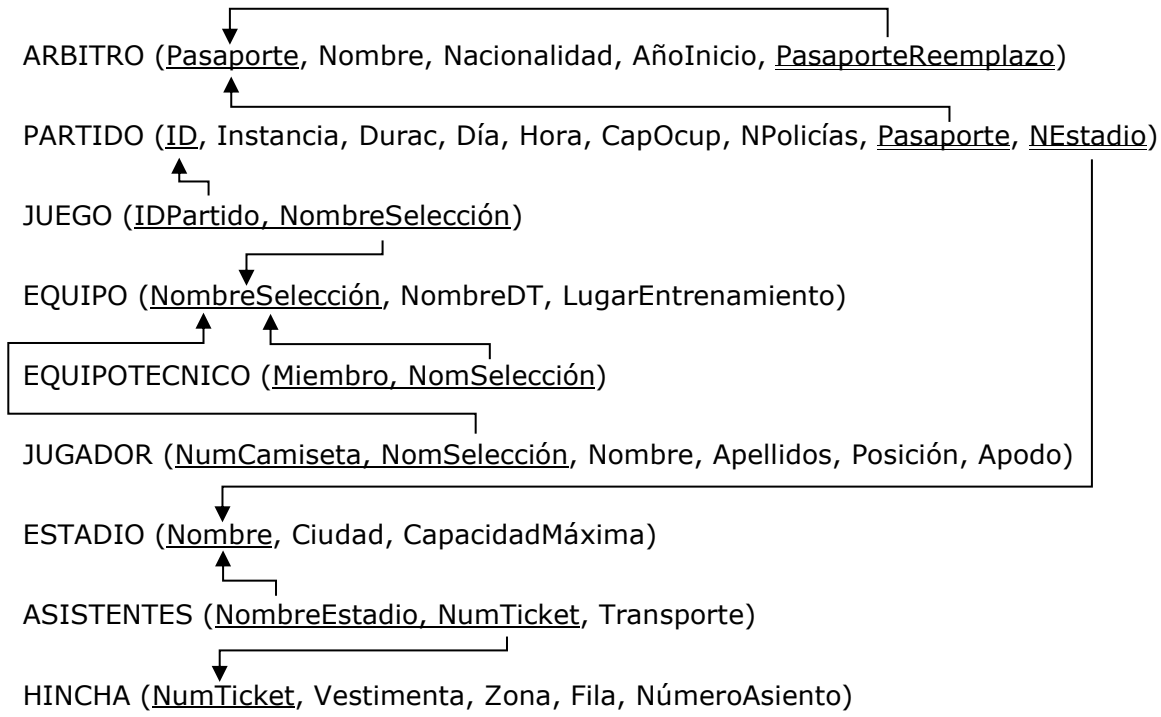
De los equipos que participan en los partidos conocemos el nombre de la selección, que es único entre el resto de las selecciones, el nombre del DT de esa selección, el lugar donde entrena y el cuerpo técnico que la compone (varios), también conocemos los distintos jugadores que forman parte de las selecciones, de los que sabemos su número de camiseta, que los distingue dentro de un equipo, la posición que ocupan en la cancha, su nombre, apellidos y el apodo de cada uno; es sabido que durante un mundial un jugador representa a un solo país y que en distintas selecciones puede haber jugadores con el mismo número. Al ser un mundial, un equipo puede jugar como mínimo tres partidos (quedo eliminado en fase de grupos) o llegar hasta siete (llegó a la final).

En este sistema los estadios son sede de distintos partidos, de los que conocemos su nombre, la ciudad en la que se encuentran, su capacidad máxima y también la capacidad ocupada y la cantidad de policías que hay presentes en cada partido; también sabemos que los hinchas pueden visitar los estadios en distintos medios de transporte (se quiere almacenar este dato). De cada hincha conocemos el número de ticket (único), la vestimenta y su ubicación (compuesta de zona, fila y número de asiento).

Adicionalmente, sabemos que un partido se juega solo en un estadio y que en un estadio pueden jugarse muchos partidos.

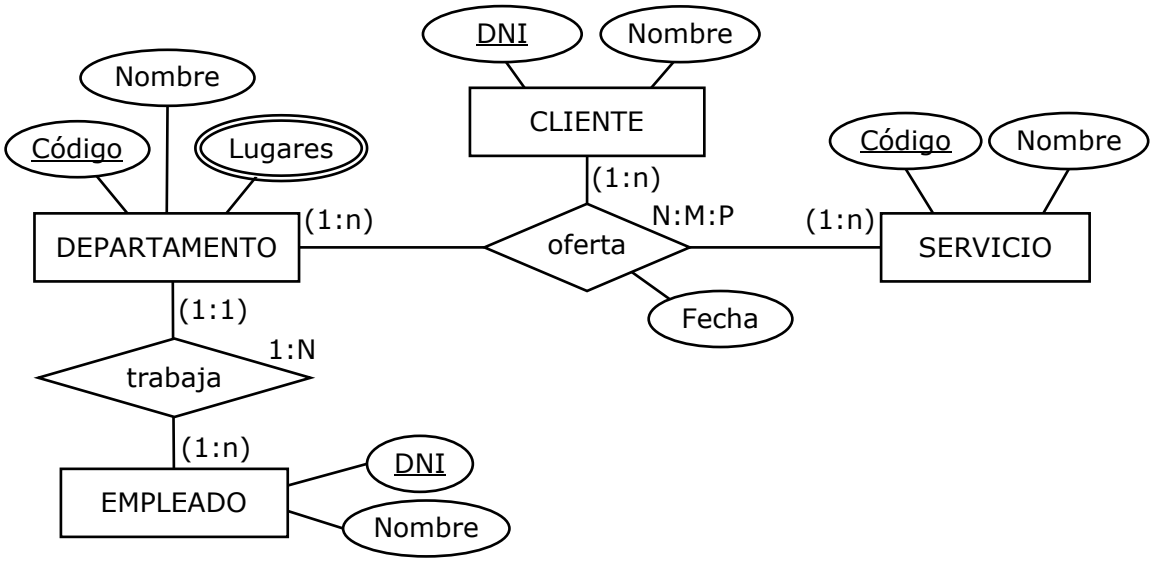
Solución





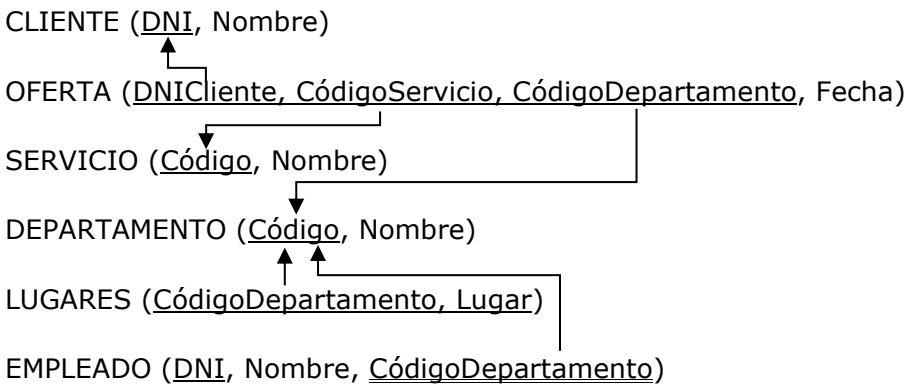
Ejercicio 13 – Modelo Relacional de un E/R dado

Transformar el siguiente modelo E/R con una relación ternaria al Modelo Relacional.



Solución

Cuando existen relaciones ternarias, cuaternarias, etc. que unen más de 2 entidades, se crea una nueva tabla con los atributos de la relación (si los hay) y con todas las claves principales de las entidades como claves ajenas. La clave principal la formarán el conjunto de claves ajenas cuya entidad tenga una cardinalidad máxima de "n". Si la cardinalidad máxima de una entidad es "1", su clave no formará parte de la clave principal y permanecerá como ajena.



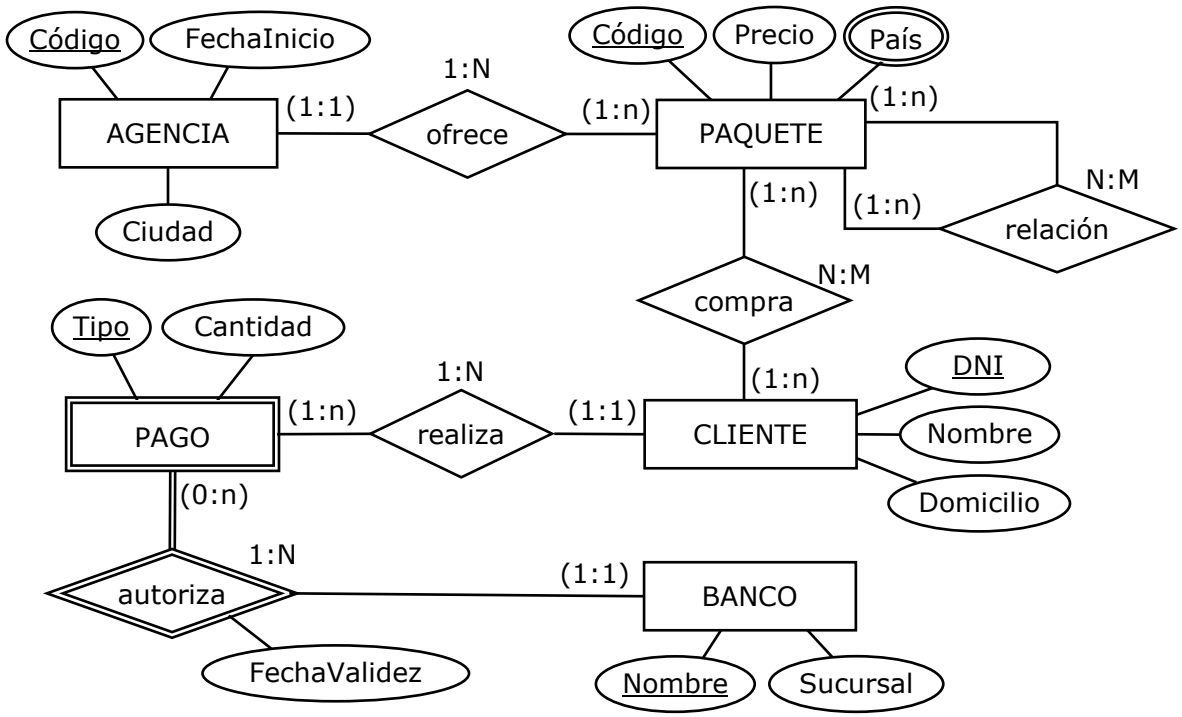
Ejercicio 14 – Modelo E/R. Agencia de viajes

Se quiere realizar una base de datos para llevar la información de varias agencias de viajes. De cada una se conoce su código, la fecha de inicio de actividades y su ciudad.

Cada agencia ofrece paquetes turísticos, los cuales se identifican por un código de paquete y tienen un precio y destinos a varios países. Un paquete puede estar relacionado con otros paquetes a modo de combo (ejemplo: viaje a Disney + crucero por el Caribe). Cada paquete es propio de su agencia, no puede ser vendido en otra.

Los paquetes son comprados por clientes, de los cuales se almacena su nombre, domicilio y son diferenciados por su DNI. Los clientes disponen de varias formas de pago, de las cuales conocemos su tipo y la cantidad a pagar. Los medios de pago son autorizados por un solo banco, de los cuales sabemos que poseen un nombre (único) y sucursal. Al autorizar los pagos, se establece una fecha de validez.

Solución



Ejercicio 15 – Modelo E/R. Biblioteca 2

Supongamos que queremos diseñar una base de datos para una biblioteca y hemos conocido que ésta funciona de la siguiente forma:

En la biblioteca hay, como es natural, una serie de libros que los empleados solicitan a las editoriales. Cuando un libro se recibe, se le da de alta, construyéndole una ficha para búsqueda por autor y otra ficha para búsqueda por tema. En ambas fichas aparecen el título de libro, el nombre del autor y su nacionalidad, la editorial a la que pertenece la publicación, el tema sobre el que trata, el ISBN y la estantería de la biblioteca en la que se encuentra. Hay que aclarar que en la biblioteca soportan como máximo 100 libros y tienen un número y un lugar asignado dentro de la biblioteca. Un empleado puede solicitar un libro escribiendo una carta de petición a la editorial correspondiente. La dirección a la que ha de dirigir la carta se encuentra en un archivo de editoriales.

Para acceder a los libros de la biblioteca es necesario tener un carné que acredita a los distintos usuarios. Este carné se confecciona a cada persona la primera vez que intenta retirar un libro. Cada usuario solo puede tener retirado un libro a la vez.

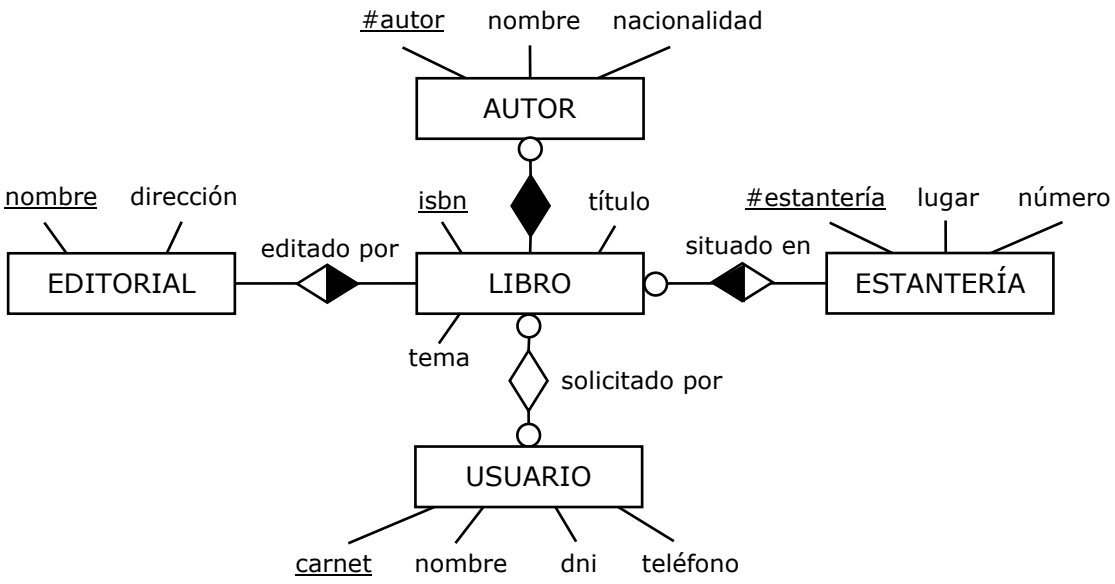
En la biblioteca les gustaría disponer de un listado que se lanzaría al final de cada día y en el que aparecería, para cada libro que se encuentra retirado, el título, el ISBN, el autor y el número de carné, nombre y el DNI del usuario que lo mantiene retirado.

Cuando un usuario intenta retirar un libro ha de presentar su carné. Si el libro que desea ha sido retirado por otro usuario, se llama a éste por teléfono indicándole que hay otro usuario que desea el libro para que lo devuelva en caso de que no lo esté utilizando.

Mensualmente se confecciona un inventario actualizado donde se indica para cada libro el nombre, el autor y el lugar de la biblioteca donde se encuentra.

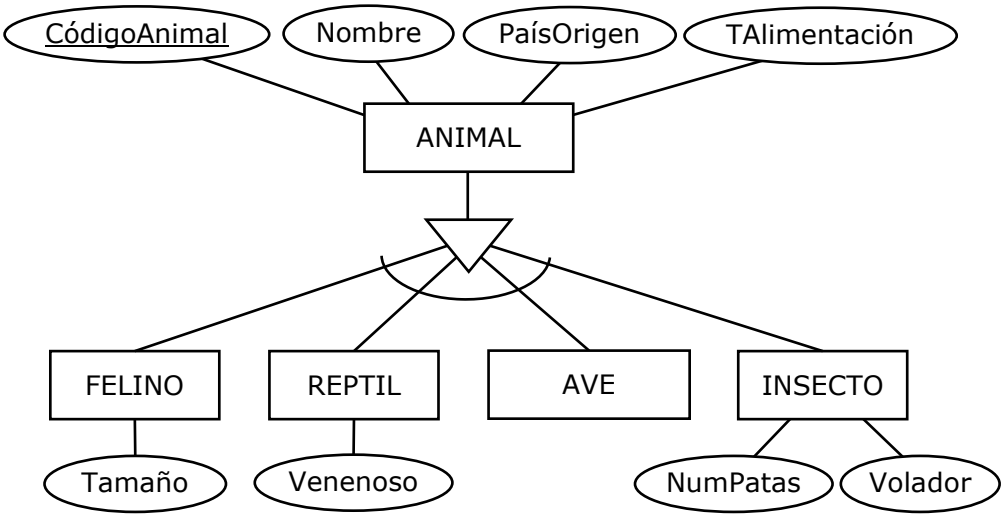
Solución

Este ejercicio se va a pasar al modelo E/R utilizando el modelo "Rein85".



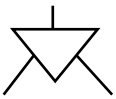
Ejercicio 16 – Modelo Relacional de un E/R extendido dado

Transformar el siguiente modelo E/R extendido al Modelo Relacional.

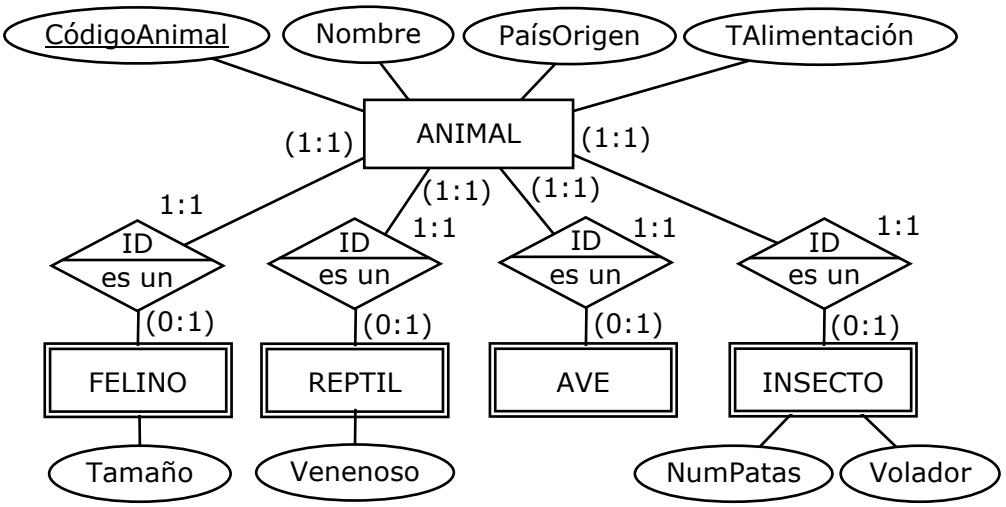


Consideraciones previas

El modelo extendido utiliza jerarquías, que es cuando una entidad supertipo representa a varias entidades subtipo. Se representan mediante un triángulo y existen 4 tipos de jerarquías:

Símbolo	Nombre	Descripción
	Solapada (Inclusiva) y parcial	Los miembros supertipo pueden ser simultáneamente varios miembros subtipo (solapada) y es posible que haya supertipos sin representación (parcial). Ejemplo: "Empleado" puede ser "Técnico", "Supervisor", ambos o ninguno, porque se dedica a otra cosa.
	Solapada (Inclusiva) y total	Los miembros supertipo pueden ser simultáneamente varios miembros subtipo (solapada) y todos los subtipos tienen representación (total). Ejemplo: "Empleado" puede ser "Técnico", "Supervisor" o ambos, pero no hay más opciones.
	Exclusiva (Disyunta) y parcial	Los miembros supertipo solo pueden ser un miembro subtipo (exclusivo) y es posible que haya supertipos sin representación (parcial). Ejemplo: "Empleado" es "Técnico", "Supervisor" o ninguno, pero no ambos.
	Exclusiva (Disyunta) y total	Los miembros supertipo solo pueden ser un miembro subtipo (exclusivo) y todos los subtipos tienen representación (total). Ejemplo: "Empleado" es "Técnico" o "Supervisor", no hay más opciones.

La jerarquía que se da en este caso es exclusiva y parcial, es decir, un animal es uno de los 4 subtipos que hay ahí o ninguno, pero no 2 o más. Por tanto, esta jerarquía equivale a 4 relaciones 1:1 con cardinalidad (1,1) en el supertipo y (1,0) en los subtipos tal y como se muestra a continuación.



Solución

ANIMAL (CodAnimal, Nombre, PaísOrigen, TipoAlimentación)

FELINO (CodAnimal, Tamaño)

REPTIL (CodAnimal, Venenoso)

AVE (CodAnimal)

INSECTO (CodAnimal, NumPatas, Volador)

Ejercicio 17 – Modelo E/R extendido. Proyecto de investigación

El departamento de Informática desea diseñar una base de datos para gestionar los profesores que participan en los proyectos de investigación. Diseña el modelo E/R usando el método extendido con la siguiente información:

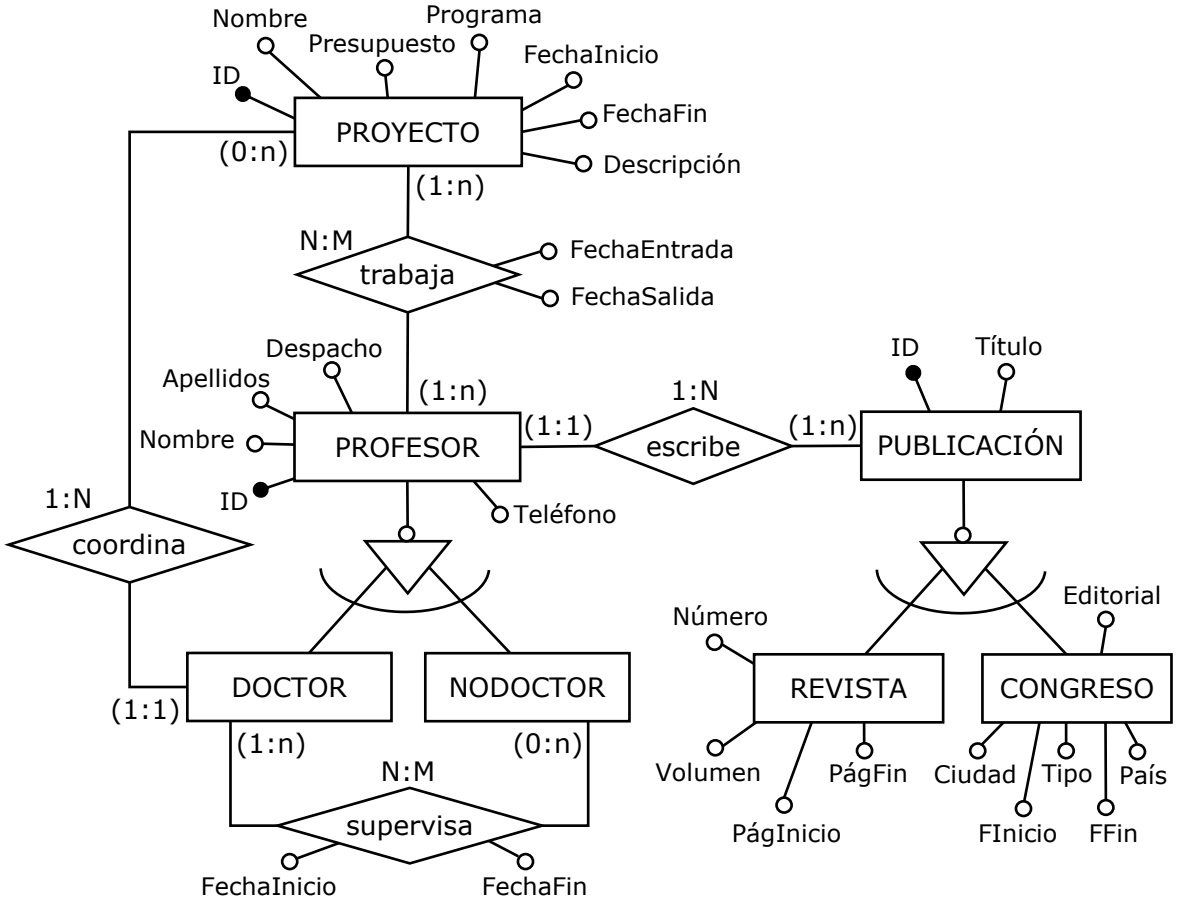
- De cada proyecto de investigación se desea almacenar un identificador único, nombre, presupuesto total, el programa de I+D que lo financia, fecha de inicio, fecha de finalización y una descripción.
- En los proyectos de investigación pueden trabajar varios profesores del departamento durante un periodo de tiempo determinado por una fecha de entrada y una fecha de salida. Tenga en cuenta que un mismo profesor puede trabajar en el mismo proyecto en diferentes épocas y en varios a la vez.
- De cada profesor se desea almacenar un identificador único, nombre, apellidos, despacho y teléfono.
- Los profesores del departamento pueden ser doctores o no doctores. Un profesor no doctor debe ser supervisado por un profesor doctor. El tiempo de supervisión viene determinado por una fecha de inicio y una fecha de fin. Debemos almacenar

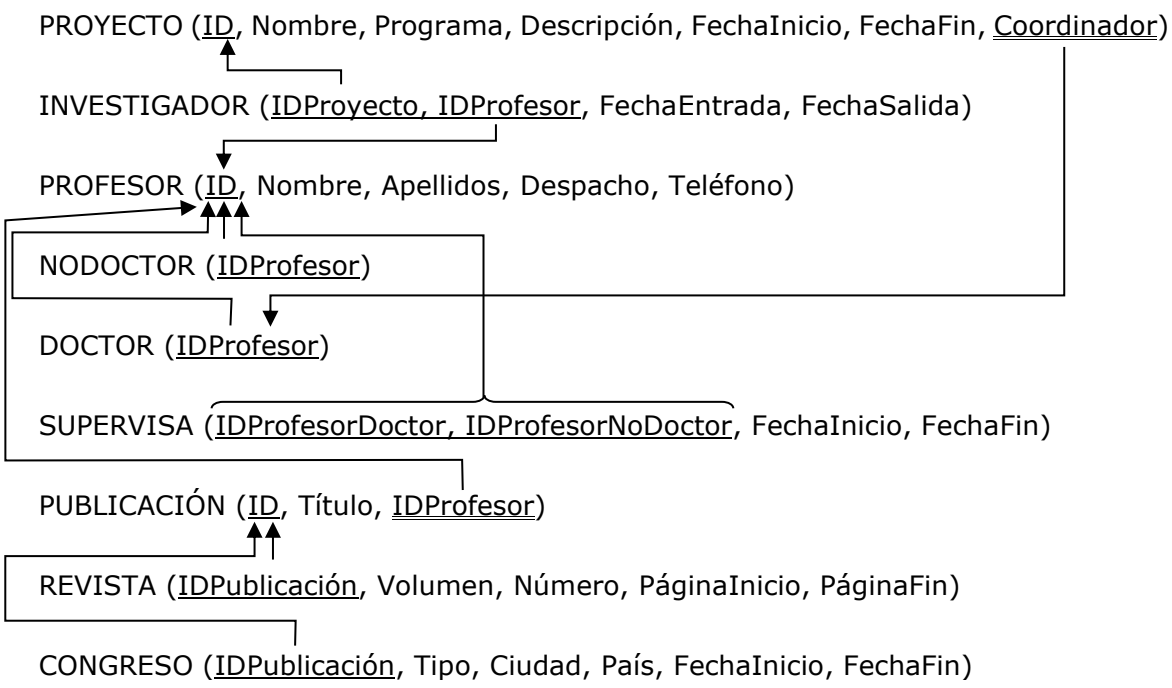
los profesores doctores que han supervisado a un profesor no doctor y durante qué periodos lo han sido.

- De todos los profesores que trabajan en el proyecto hay uno que es el investigador principal, que será el encargado de coordinar el proyecto. Es necesario almacenar quién es el investigador principal de cada uno de los proyectos. El investigador principal de un proyecto tiene que ser un profesor doctor, en ningún caso podrá serlo un profesor no doctor. Ten en cuenta que el investigador principal no puede cambiar a lo largo de la vida del proyecto y siempre será el mismo. Un doctor puede coordinar varios proyectos a la vez.
- Los profesores doctores y no doctores escriben publicaciones. Una publicación consta de un código único y un título y puede ser en una revista o en un congreso.
- Si la publicación es en una revista, además del código único y el título, almacenamos el volumen, el número, la página de inicio y la página de fin.
- Si la publicación es en un congreso, además del código único y el título, almacenamos el tipo de congreso, ciudad, país, editorial, fecha de inicio y de fin.

Solución

En este caso, existen una jerarquía exclusiva (sin solapamiento) y total, ya que un "Profesor" puede ser "Doctor" o "No doctor", pero no ambos o ninguno. Equivale a 2 relaciones 1:1 con cardinalidad (1,1) en el supertipo y (1,0) en los subtipos.





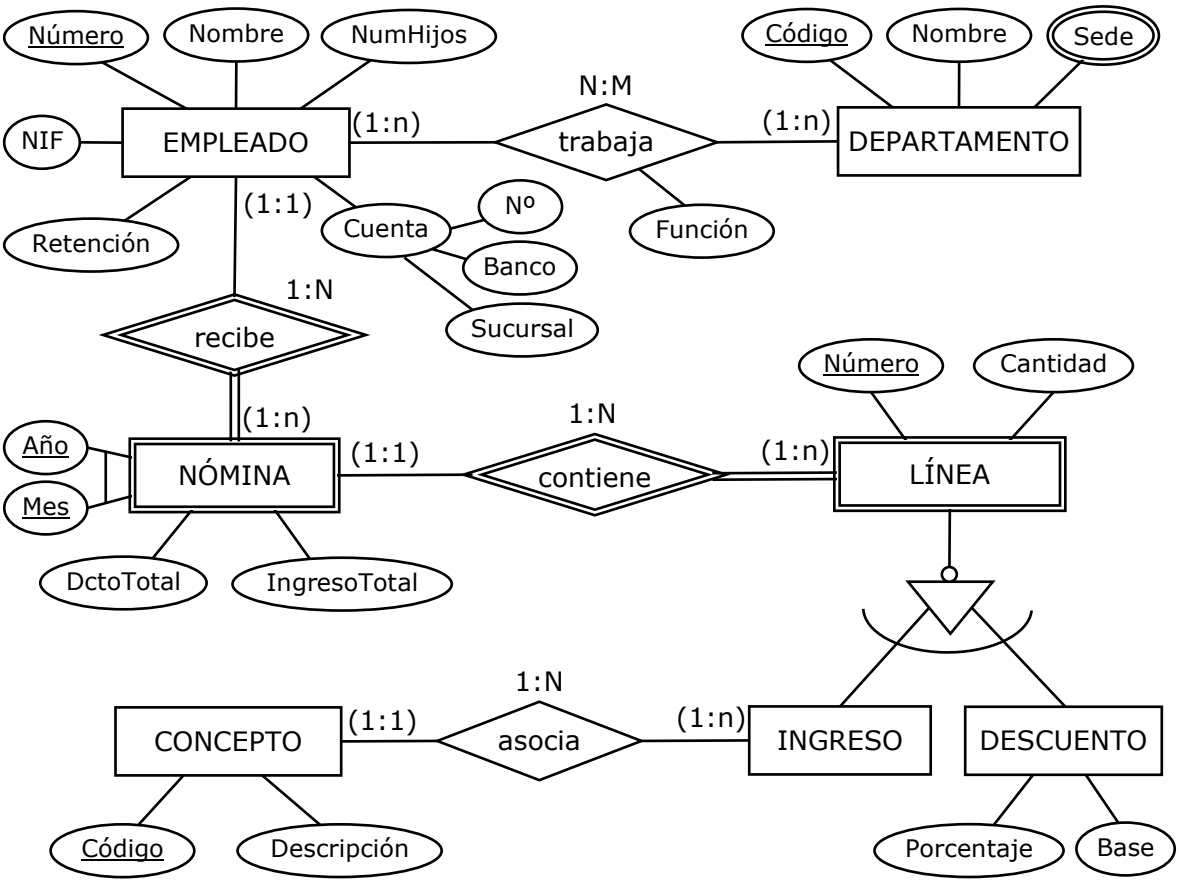
Ejercicio 18 – Modelo E/R extendido. Gestión de nominas

Una empresa decide informatizar su gestión de nóminas. Del resultado del análisis realizado, se obtiene la siguiente información:

- A cada empleado se le asigna un número de empleado cuando se incorpora en la empresa, que será usado para su identificación. Además, se registra su Número de Identificación Fiscal (NIF), nombre, número de hijos, porcentaje de retención para Hacienda, datos de cuenta corriente donde se ingresa el dinero (banco, sucursal y número de cuenta) y departamentos en los que trabaja.
- Un empleado puede trabajar en varios departamentos y en cada uno de ellos trabajará con una función distinta. De cada departamento se registra un código único, su nombre y cada una de sus posibles sedes.
- A cada empleado se le entregan múltiples nóminas a lo largo de su vida laboral y al menos una mensualmente. Son datos propios de una nómina el ingreso total percibido por el empleado y el descuento total aplicado.
- La distinción entre dos nóminas se hará, además de mediante el número de identificación del empleado, mediante el ejercicio fiscal (año) y número de mes al que pertenece.
- Cada nómina consta de varias líneas (al menos una de ingresos) y cada línea se identifica por un número de línea dentro de la correspondiente nómina. Una línea puede corresponder a un ingreso o a un descuento. En ambos casos, se recoge la cantidad que corresponde a la línea (en positivo si se trata de un ingreso o en negativo si se trata de un descuento); en el caso de los descuentos, se recoge la base sobre la cual se aplica y el porcentaje que se aplica para el cálculo de éstos.

- Cada línea de ingreso de una nómina responde a un único concepto retributivo, pero puede haber varias líneas que respondan al mismo concepto retributivo.
- De los conceptos retributivos se mantiene un código (único) y una descripción.

Solución



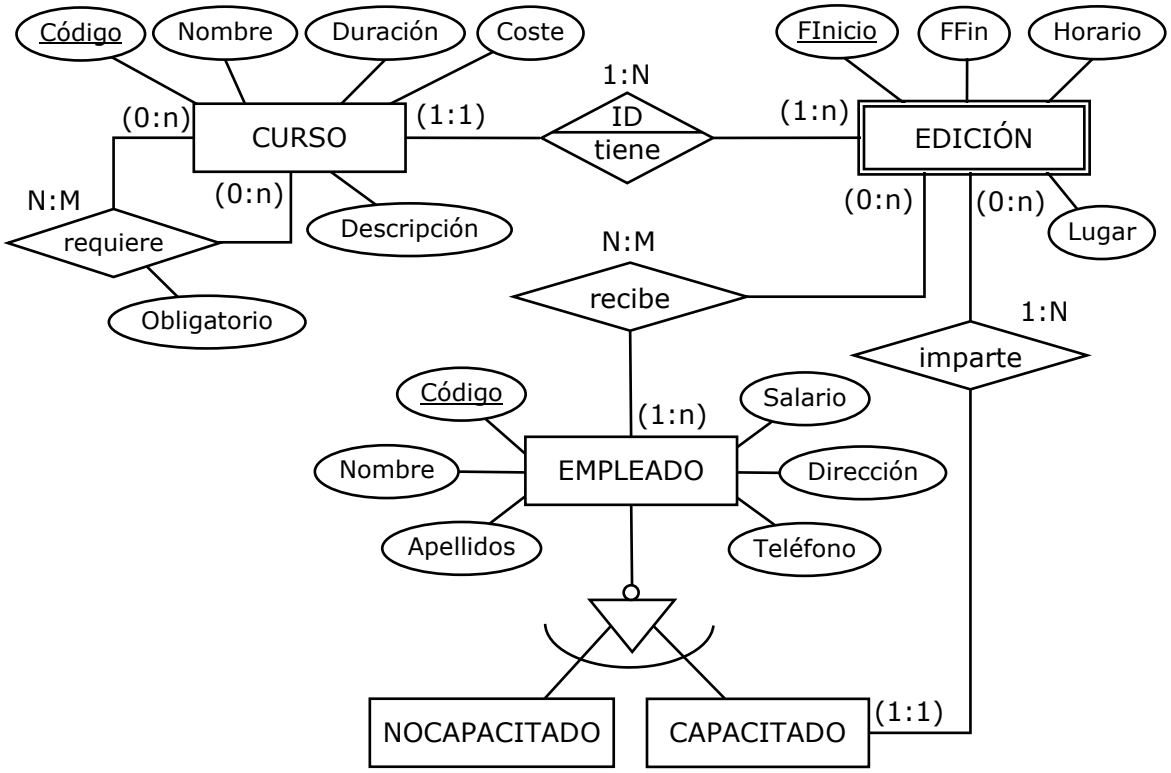
Ejercicio 19 – Modelo E/R extendido. Cursos de formación

El departamento de formación de una empresa desea diseñar una base de datos para planificar y gestionar la formación de sus empleados.

- La empresa organiza cursos internos de formación de los que se desea conocer el código de curso, el nombre, una descripción, el número de horas de duración y el coste del curso.
- Un curso puede tener como prerrequisito haber realizado otros previamente y, a su vez, la realización de un curso puede ser prerrequisito de otros. Un curso que es un prerrequisito de otro puede serlo de forma obligatoria o sólo recomendable.
- Un mismo curso tiene diferentes ediciones, es decir, se imparte en diferentes lugares, fechas y con diferentes horarios (intensivo, de mañana o de tarde). En una misma fecha de inicio sólo puede impartirse una edición de un curso.

- En la empresa hay empleados de los que se almacena su código de empleado, nombre y apellidos, dirección, teléfono y salario, así como si está capacitado para impartir cursos o no.
- Un mismo empleado puede ser docente en una edición de un curso (si está capacitado) y alumno en otra edición, pero nunca puede ser ambas cosas a la vez (en una misma edición de curso o lo imparte o lo recibe).

Solución



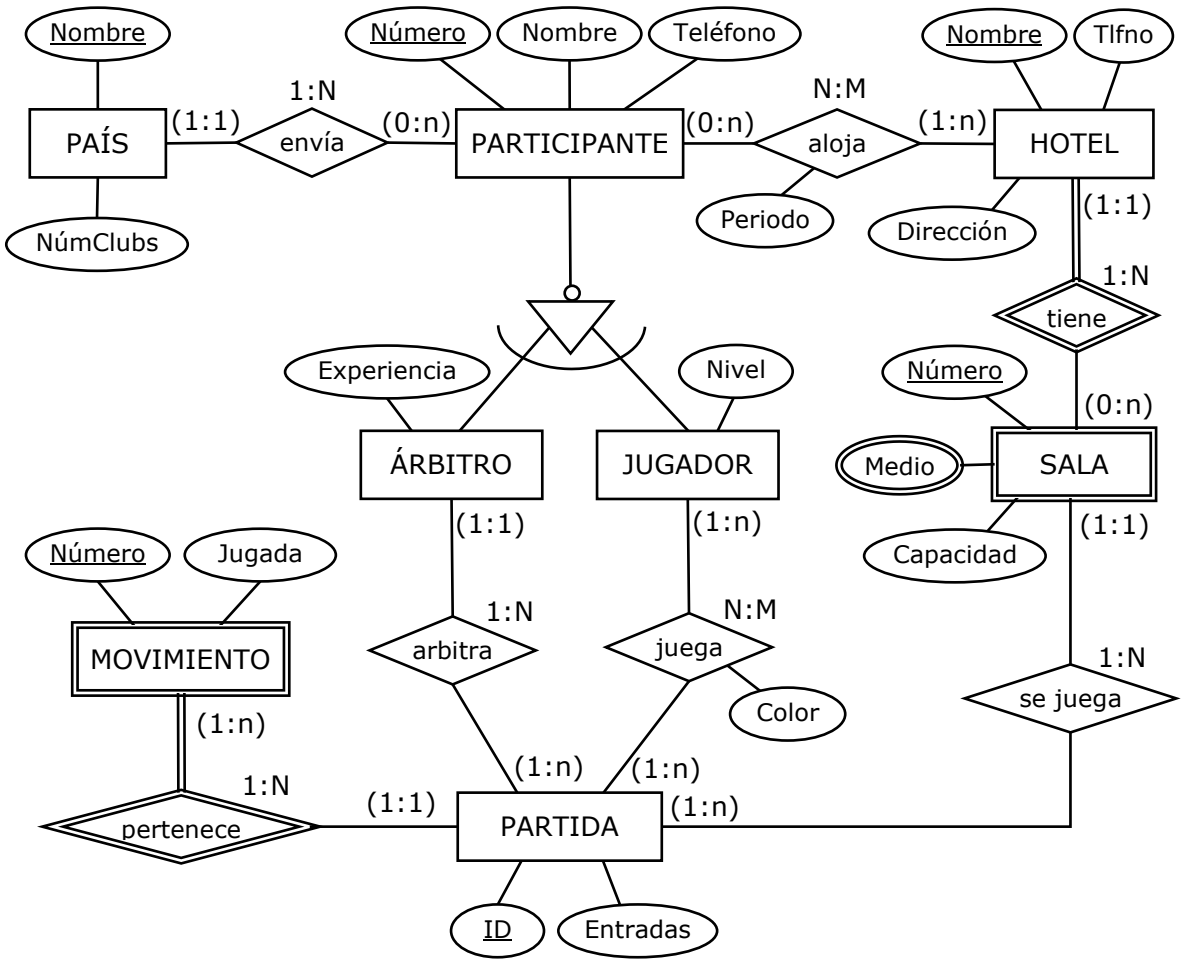
Ejercicio 20 – Modelo E/R extendido. Club de ajedrez

Un club de ajedrez ha sido encargado por la Federación Internacional de Ajedrez de la organización de los próximos campeonatos mundiales que se celebrarán en la localidad. Por este motivo, desea llevar a una base de datos toda la gestión relativa a participantes, alojamientos y partidas. Diseña el modelo E/R teniendo en cuenta lo siguiente:

- En el campeonato hay varios participantes que pueden ser jugadores o árbitros. De cada participante se requiere conocer el número de asociado, nombre y teléfono de contacto. De los jugadores se precisa, además, el nivel de juego en una escala de 1 a 10 y de los árbitros guardaremos los años de experiencia. Ningún árbitro puede participar como jugador.
- Distintos países envían al campeonato un conjunto de jugadores y árbitros, aunque no todos los países envían participantes. Cada jugador y árbitro es enviado por un único país. Cada país se identifica por su nombre e interesa conocer, además, el número de clubes de ajedrez existentes en el mismo.

- Cada partida, identificada por un ID, la juegan dos jugadores y la arbitra un árbitro. Interesa registrar las partidas que juega cada jugador y el color (blancas o negras) con el que juega. Todo participante del campeonato debe participar en al menos una partida. Para cada partida, se desea conocer el número de entradas vendidas en la sala que se ha jugado.
- Tanto jugadores como árbitros pueden alojarse en los hoteles en los que se desarrollan las partidas. Se desea conocer en qué hotel y durante qué periodo de días se ha alojado cada uno de los participantes. De cada hotel, se desea conocer el nombre (único), la dirección y el número de teléfono.
- Cada partida se celebra en una de las salas de las que pueden disponer los hoteles. De cada sala, se desea conocer su número (se puede repetir en distintos hoteles) la capacidad y medios de que dispone (radio, televisión, vídeo, etc.) para facilitar la retransmisión de los encuentros.
- De cada partida se pretende registrar todos los movimientos que la componen. Cada movimiento se identifica por su número de movimiento (se repite en las distintas partidas) y la jugada realizada.

Solución



Ejercicio 21 – Normalización

Normalizar la siguiente relación hasta 3FN, explicando detalladamente el proceso de normalización, así como las decisiones tomadas para realizar dicha normalización.

PEDIDO (Núm_Pedido, Fecha, Cod_Producto, Descripción, Precio, Cantidad, Total)

Solución

Primera Forma Normal (1FN)

Una relación está en 1FN si no tiene grupos repetitivos, es decir, si sus campos no tienen más de un valor por cada registro.

Si observamos la relación "PEDIDO", se puede ver que los atributos "Cod_Producto", "Descripción", "Precio", "Cantidad" y "Total" pueden tener varios valores para un mismo "Núm_Pedido", por lo que no cumple la condición de la 1FN. Ejemplo:

Pedido

Núm_Pedido	Fecha	Cod_Producto		Descripción		Precio		Cantidad		Total
0001	XXX	003	004	Tomate	Pera	1,2	1	3	2	5,6
0002	YYY	003		Tomate		1,2		2		2,4

Para pasarlo a 1FN, se crea una nueva relación cuyos campos serán los atributos multivaluados, la clave principal de la tabla original y la clave de la relación que no cumple la 1FN.

PEDIDO (Núm_Pedido, Fecha, Total)
COMPRA (Núm_Pedido, Cod_Producto, Descripción, Precio, Cantidad)

Y ahora, los campos de cada registro tendrán un único valor.

Pedido

Núm_Pedido	Fecha	Total
0001	XXX	5,6€
0002	YYY	2,4€

Compra

Núm_Pedido	Cod_Producto	Descripción	Precio	Cantidad
0001	003	Tomate	1,2 €	3 kg
0001	004	Pera	1 €	2 kg
0002	003	Tomate	1,2 €	2 kg

Segunda Forma Normal (2FN)

Una relación está en 2FN si está en 1FN y, además, no tiene dependencias parciales. Una dependencia parcial se produce cuando un atributo depende funcionalmente de una parte de la clave y no de la clave completa.

Se dice que un atributo "Y" depende funcionalmente de otro atributo "X", si a todo valor de "X" le corresponde siempre el mismo valor de "Y". Se representa como "X→Y".

La relación "COMPRA" no está en 2FN porque los atributos "Descripción" y "Precio" sólo dependen funcionalmente de la clave "Cod_Producto" y no de la clave completa compuesta por "Núm_Pedido y Cód_Producto". Para pasarlo a 2FN, se crea una relación nueva con los atributos que no dependen funcionalmente de la clave completa y su clave.

PEDIDO (Núm_Pedido, Fecha, Total)
COMPRA (Núm_Pedido, Cod_Producto, Cantidad)
PRODUCTO (Cod_Producto, Descripción, Precio)

Tercera Forma Normal (3FN)

Una relación está en 3FN si está en 2FN y, además, no tiene dependencias funcionales transitivas entre la clave primaria y los atributos en las que no participe una clave candidata.

Una dependencia funcional "X→Y" es una dependencia funcional transitiva si existe un atributo "Z" donde existe una dependencia funcional "X→Z" y "Z→Y".

En este caso, se puede comprobar que no hay ninguna dependencia transitiva, por lo que las relaciones están en 3FN.

Ejercicio 22 – Normalización 2

Normalizar la siguiente relación hasta 3FN, explicando detalladamente el proceso de normalización, así como las decisiones tomadas para realizar dicha normalización.

ORDEN (Id_Orden, Fecha, Id_Cliente, Nombre_Cliente, Estado, Núm_Artículo, Nombre_Artículo, Cantidad, Precio)

Solución

Primera Forma Normal (1FN)

Analizamos la relación "ORDEN" en busca de grupos repetitivos poniendo ejemplos:

Orden

IdOrden	Fecha	IdCli	NomCli	Estado	NumArt	NomArt	Cant	Precio
2301	23/02/11	101	Martin	Casado	3786	Red	3	35,00
					4011	Raqueta	6	65,00
2303	27/02/11	110	Pedro	Soltero	4011	Raqueta	2	65,00
					3141	Funda	2	10,00

Como se puede observar, los atributos "Número_Artículo", "Nombre_Artículo", "Cantidad" y "Precio" pueden tener varios valores para un mismo "ID_Orden", por lo que no cumple con la 1FN. Lo pasamos a 1FN creando una nueva relación que contenga los atributos multivaluados, la clave principal de la tabla original y la clave de la relación que no cumple la 1FN.

ORDEN (Id_Orden, Fecha, ID_Cliente, Nombre_Cliente, Estado)
ARTÍCULOS_ORDEN (Id_Orden, Núm_Artículo, Nombre_Artículo, Cantidad, Precio)

Segunda Forma Normal (2FN)

Analizamos las relaciones en busca de dependencias parciales y se ve que los atributos "Nombre_Artículo" y "Precio" de la relación "ARTÍCULOS_ORDEN" dependen únicamente de "Núm_Artículo" y no de la clave completa, por lo que no está en 2FN. Se pasa.

ORDEN (Id_Orden, Fecha, ID_Cliente, Nombre_Cliente, Estado)
ARTÍCULOS_ORDEN (Id_Orden, Núm_Artículo, Cantidad)
ARTÍCULO (Núm_Artículo, Nombre_Artículo, Precio)

Tercera Forma Normal (3FN)

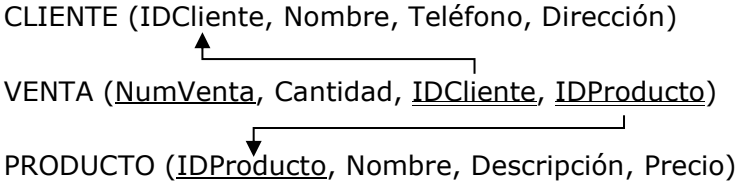
La relación ORDEN no está en 3FN porque hay una dependencia funcional transitiva con "Nombre_Cliente" y "Estado" ya que éstos dependen funcionalmente del atributo "ID_Cliente" y no de la clave "ID_Orden". En otras palabras, no está en 3FN porque en la relación "ORDEN" hay una dependencia transitiva {Id_Orden} → {Nombre_Cliente, Estado} ya que hay una dependencia funcional {Id_Orden} → {IDHuesped} y {IDHuesped} → {Nombre_Cliente, Estado}.

Lo pasamos a 3FN creando una nueva relación llamada "CLIENTE":

CLIENTE (Id_Cliente, Nombre_Cliente, Estado)
ORDEN (Id_Orden, Fecha, ID_Cliente)
ARTÍCULOS_ORDEN (Id_Orden, Núm_Artículo, Cantidad)
ARTÍCULO (Núm_Artículo, Nombre_Artículo, Precio)

Ejercicio 23 – Definición de datos (DDL) y manipulación (DML)

a) Crear mediante lenguaje de definición de datos (DDL) una base de datos llamada "Tienda" y las siguientes tablas:



b) Realiza las siguientes operaciones con lenguaje de manipulación de datos (DML):

1. Inserta el cliente "Antonio", con ID "123", teléfono "666888444" y dirección "Calle Falsa 123".
2. Inserta los productos "Plátano" y "Tomate" con ID "1" y "2" y precio "1,2" y "1,5". "Plátano" tendrá como descripción "De Canarias".
3. Modifica el precio de los plátanos a 1,3€.
4. Elimina los datos del cliente llamado "Juan".
5. Elimina los datos de los clientes cuyo nombre empieza por "A".
6. Aumenta el precio de los tomates 10 céntimos.

Solución

a) Crear mediante lenguaje de definición de datos (DDL) una base de datos llamada "Tienda" y las siguientes tablas:

Creamos la base de datos y la abrimos:

```
CREATE DATABASE TIENDA;
USE TIENDA;
```

Creamos las tablas en Microsoft SQL Server / Oracle / MS Access:

```
CREATE TABLE CLIENTE (
        IDCliente INT NOT NULL PRIMARY KEY,
        Nombre VARCHAR(20) NOT NULL,
        Teléfono VARCHAR(20) NOT NULL, -- Mejor VARCHAR(20) que INT
        Dirección VARCHAR(50) NOT NULL
);

CREATE TABLE PRODUCTO (
        IDProducto INT NOT NULL PRIMARY KEY,
        Nombre VARCHAR(20) NOT NULL,
        Descripción VARCHAR(200),
        Precio FLOAT NOT NULL
);

CREATE TABLE VENTA (
        NumVenta INT NOT NULL IDENTITY(1,1) PRIMARY KEY, -- Autoincremento
        Cantidad INT NOT NULL,
        IdCliente INT FOREIGN KEY REFERENCES CLIENTE(IDCliente),
        IDProducto INT FOREIGN KEY REFERENCES PRODUCTO(IDProducto)
);
```

Creamos las tablas en MySQL:

```
CREATE TABLE CLIENTE (
        IDCliente INT NOT NULL,
        Nombre VARCHAR(20) NOT NULL,
        Telefono VARCHAR(20) NOT NULL,
        Direccion VARCHAR(50) NOT NULL,
        PRIMARY KEY (IDCliente)
);

CREATE TABLE PRODUCTO (
        IDProducto INT NOT NULL,
        Nombre VARCHAR(20) NOT NULL,
        Descripcion VARCHAR(200),
        Precio FLOAT NOT NULL,
        PRIMARY KEY (IDProducto)
);
```

```
CREATE TABLE VENTA (
        NumVenta INT NOT NULL AUTO_INCREMENT,
        Cantidad INT NOT NULL,
        IDCliente INT NOT NULL,
        IDProducto INT NOT NULL,
        PRIMARY KEY (NumVenta),
        FOREIGN KEY (IDCliente) REFERENCES CLIENTE(IDCliente),
        FOREIGN KEY (IDProducto) REFERENCES PRODUCTO(IDProducto)
);
```

b) Realiza las siguientes operaciones con lenguaje de manipulación de datos (DML):

1. Inserta el cliente "Antonio", con ID "123", teléfono "666888444" y dirección "Calle Falsa 123".

   Forma completa:

   ```
   INSERT INTO CLIENTE (IDCliente, Nombre, Teléfono, Dirección) VALUES
       (123, 'Antonio', '666888444', 'Calle Falsa 123');
   ```

   Forma abreviada:

   Esta forma se utiliza solamente cuando la tabla tenga exactamente las mismas columnas y en el mismo orden en el que se están insertando los valores.

   ```
   INSERT INTO CLIENTE VALUES
       (123, 'Antonio', '666888444', 'Calle Falsa 123');
   ```

2. Inserta los productos "Plátano" y "Tomate" con ID "1" y "2" y precio "1,2" y "1,5". "Plátano" tendrá como descripción "De Canarias".

   ```
   INSERT INTO producto VALUES
       (1, 'Plátano', 'De Canarias', 1.2),
       (2, 'Tomate', '', 1.5);
   ```

   Otra forma de hacerlo es individualmente:

   ```
   INSERT INTO PRODUCTO VALUES
       (1, 'Plátano', 'De Canarias', 1.2);

   INSERT INTO PRODUCTO (IDProducto, Nombre, Precio) VALUES
       (2, Tomate, 1.5);
   ```

3. Modifica el precio de los plátanos a 1,3€.

   ```
   UPDATE PRODUCTO
   SET Precio=1.3
   WHERE Nombre='Plátano';
   ```

4. Elimina los datos del cliente llamado "Juan"

   ```
   DELETE FROM CLIENTE
   WHERE Nombre='Juan';
   ```

5. Elimina los datos de los clientes cuyo nombre empieza por "A".

```
DELETE FROM CLIENTE
WHERE Nombre LIKE 'A%';
```

6. Aumenta el precio de los tomates 10 céntimos.

```
UPDATE PRODUCTO
SET Precio=Precio+0.1
WHERE Nombre='Tomates';
```

Ejercicio 24 – Consultas en SQL. Consultas sobre una tabla

Realizar las consultas SQL correspondientes en cada apartado.

a) Devolver todos los campos de la tabla llamada "Cliente"

b) Devolver los campos IDCliente, Nombre y Teléfono de la tabla "Cliente"

c) Devolver los campos CP y Ciudad de la tabla "Direccion"

d) Devolver los campos CP y Ciudad de la tabla "Direccion" sin repetir registros.

e) Devolver los campos número y calle de la tabla "Direccion" cuyo campo ciudad sea "Sevilla"

f) Devolver los campos calle y ciudad de la tabla "Direccion" cuyo campo número sea "12"

g) Devolver el campo nombre de la tabla "Cliente" cuyo campo edad sea menor o igual a "32"

h) Devolver el campo nombre de la tabla "Cliente" cuya edad esté comprendida entre 20 y 35 años

i) Devolver el campo num, calle y cp de la tabla "Direccion" cuyo campo ciudad empiece por "Val", por ejemplo, Valladolid, Valencia, etc.

j) Devolver el campo num, calle y dirección de la tabla "Direccion" cuyo campo ciudad sea "Sevilla", "Córdoba" o "Huelva".

k) Devolver todos los registros de la tabla "Direccion" que tengan el código postal (cp) 41009 de Sevilla o bien que no tengan el 14010 de Córdoba.

l) Devolver todos los registros de la tabla "Ciudad" ordenados por provincia en orden ascendente y, dentro de la misma provincia, ordenados por número de habitantes (numhabitantes) en orden descendente.

m) Devolver los campos idCliente, nombre y descripción de la tabla "Cliente", pero que los resultados no se muestren con el nombre real del campo, sino con los alias "id", "cliente" y "desc" respectivamente.

n) Devolver el campo "IdPedido" de la tabla "Pedido" cuyo campo "direccionEnvio" no tengan ningún valor.

Solución

a) Devolver todos los campos de la tabla llamada "Cliente"

```
SELECT *
FROM Cliente;
```

b) Devolver los campos IDCliente, Nombre y Teléfono de la tabla "Cliente"

```
SELECT IDCliente, Nombre, Teléfono
FROM Cliente;
```

c) Devolver los campos CP y Ciudad de la tabla "Direccion"

```
SELECT CP, Ciudad
FROM Direccion;
```

d) Devolver los campos CP y Ciudad de la tabla "Direccion" sin repetir registros.

Supongamos que Pedro y Juan viven en la misma dirección (CP y Ciudad), la consulta anterior devolverá 2 registros iguales; una haciendo referencia a Pedro y otra a Juan. Para evitar registros repetidos, se utiliza DISTINCT.

```
SELECT DISTINCT CP, Ciudad
FROM Direccion;
```

e) Devolver los campos número y calle de la tabla "Direccion" cuyo campo ciudad sea "Sevilla"

```
SELECT número, calle
FROM Direccion
WHERE ciudad='Sevilla';
```

f) Devolver los campos calle y ciudad de la tabla "Direccion" cuyo campo número sea "12"

```
SELECT calle, ciudad
FROM Direccion
WHERE numero=12;
```

g) Devolver el campo nombre de la tabla "Cliente" cuyo campo edad sea menor o igual a "32"

```
SELECT nombre
FROM Cliente
WHERE edad<=32;
```

h) Devolver el campo nombre de la tabla "Cliente" cuya edad esté comprendida entre 20 y 35 años

```
SELECT nombre
FROM Cliente
WHERE edad BETWEEN 20 AND 35;
```

i) Devolver el campo num, calle y cp de la tabla "Direccion" cuyo campo ciudad empiece por "Val", por ejemplo, Valladolid, Valencia, etc.

Los patrones de texto existentes son "%" para sustituir 0 o más caracteres y "_" para sustituir exactamente un carácter.

```
SELECT num, calle, cp
FROM Direccion
WHERE ciudad LIKE 'Val%';
```

j) Devolver el campo num, calle y dirección de la tabla "Direccion" cuyo campo ciudad sea "Sevilla", "Córdoba" o "Huelva".

```
SELECT num, calle, direccion
FROM Direccion
WHERE ciudad IN ('Sevilla', 'Córdoba', 'Huelva');
```

Otra forma de hacer la misma consulta es:

```
SELECT num, calle, direccion
FROM Direccion
WHERE ciudad='Sevilla' OR ciudad='Córdoba' OR ciudad='Huelva';
```

k) Devolver todos los registros de la tabla "Direccion" que tengan el código postal (cp) 41009 de Sevilla o bien que no tengan el 14010 de Córdoba.

```
SELECT *
FROM Direccion
WHERE (ciudad='Sevilla' AND cp=41009) OR
        (ciudad='Córdoba' AND (NOT cp=14010));
```

l) Devolver todos los registros de la tabla "Ciudad" ordenados por provincia en orden ascendente y, dentro de la misma provincia, ordenados por número de habitantes (numhabitantes) en orden descendente.

```
SELECT *
FROM Ciudad
ORDER BY provincia ASC, numhabitantes DESC;
```

m) Devolver los campos idCliente, nombre y descripción de la tabla "Cliente", pero que los resultados no se muestren con el nombre real del campo, sino con los alias "id", "cliente" y "desc" respectivamente.

```
SELECT idCliente AS id, nombre AS cliente, descripción AS desc
FROM Cliente;
```

n) Devolver el campo "IdPedido" de la tabla "Pedido" cuyo campo "direccionEnvio" no tengan ningún valor.

```
SELECT idPedido
FROM Pedidos
WHERE direccionEnvio IS NULL;
```

Ejercicio 25 – Consultas en SQL. Consultas agrupadas

Supongamos la tabla "Pedidos" con los atributos "IDPedido" y "Coste". Realiza las consultas SQL correspondientes en cada apartado.

a) Devolver el número de registros que tiene la tabla "Pedidos"

b) Devolver el número de registros que tiene cada pedido de la tabla "Pedidos"

c) Devolver la suma de los costes de cada pedido de la tabla "Pedidos"

d) Devolver la media de los costes de cada pedido de la tabla "Pedidos"

e) Devolver la media de los costes de cada pedido de la tabla "Pedidos". La media debe de venir redondeada con 4 decimales.

f) Devolver la media de los costes de cada pedido de la tabla "Pedidos", pero solo las medias menores o iguales que 10 y redondeadas con 4 decimales.

Consideraciones previas

Las consultas agrupadas son aquellas donde la información no proviene de un registro individual, por ejemplo, nombre de un cliente, sino de la agrupación de campos que tienen el mismo valor, por ejemplo, la cantidad de clientes llamados "Juan". Con las agrupaciones podemos contarlos, sumarlos, hacer la media, etc.

Ejemplo: Devolver la cantidad de clientes que se llaman "Juan".

```
SELECT COUNT(*) AS CantidadDeJuanes
FROM Clientes
WHERE Nombre = 'Juan';
```

Cuando se desean agrupar filas que tienen el mismo valor en una o más columnas específicas, se utiliza la cláusula GROUP BY.

Ejemplo: Devolver el nombre de cada cliente y el número de clientes que se llaman igual.

```
SELECT Nombre, COUNT(*) AS NumRegistros
FROM Cliente
GROUP BY Nombre;
```

Supongamos que la tabla Cliente tiene los siguientes valores:

ID	Nombre	Edad
1	Juan	25
2	Pedro	35
3	Juan	30
4	Ana	28
5	Juan	22
6	Ana	26

La consulta anterior nos devolverá lo siguiente:

Nombre	NumRegistros
Juan	3
Pedro	1
Ana	2

Con la cláusula HAVING podemos filtrar el resultado y mostrar aquellas filas que cumplan una condición.

Ejemplo: Devolver el nombre de cada cliente y el número de clientes que se llaman igual, pero solo si tienen 2 o más registros repetidos.

```
SELECT Nombre, COUNT(*) AS NumRegistros
FROM Cliente
GROUP BY Nombre
HAVING COUNT(*) >= 2;
```

En este caso no se mostrará el cliente "Pedro" porque solo hay 1 Pedro en la tabla.

Solución

a) Devolver el número de registros que tiene la tabla "Pedidos"

```
SELECT COUNT(*) AS NumPedidos
FROM Pedidos;
```

b) Devolver el número de registros que tiene cada pedido de la tabla "Pedidos"

```
SELECT IDPedido, COUNT(*) AS NumRegistros
FROM Pedidos
GROUP BY IDPedido;
```

c) Devolver la suma de los costes de cada pedido de la tabla "Pedidos"

```
SELECT IDPedido, SUM(Coste) AS SumaCostes
FROM Pedidos
GROUP BY IDPedido;
```

d) Devolver la media de los costes de cada pedido de la tabla "Pedidos"

```
SELECT IDPedido, AVG(Coste) AS MediaCostes
FROM Pedidos
GROUP BY IDPedido;
```

e) Devolver la media de los costes de cada pedido de la tabla "Pedidos". La media debe de venir redondeada con 4 decimales.

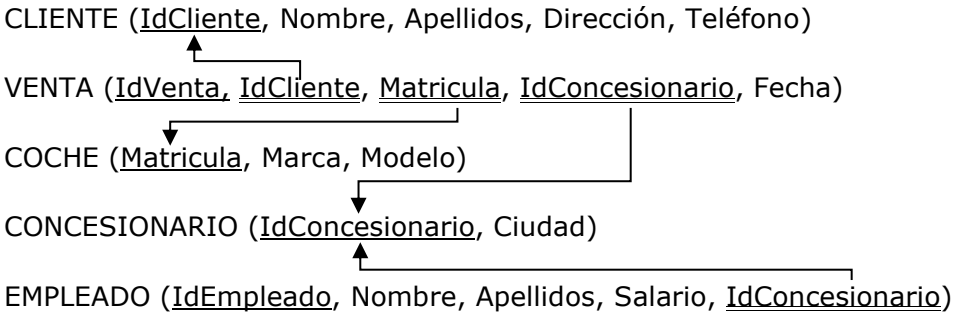
```
SELECT IDPedido, ROUND(AVG(Coste), 4) AS MediaCostes
FROM Pedidos
GROUP BY IDPedido;
```

f) Devolver la media de los costes de cada pedido de la tabla "Pedidos", pero solo las medias menores o iguales que 10 y redondeadas con 4 decimales.

```
SELECT IDPedido, ROUND(AVG(Coste), 4) AS MediaCostes
FROM Pedidos
GROUP BY IDPedido
HAVING ROUND(AVG(Coste), 4) <= 10;
```

Ejercicio 26 – Consultas en SQL. Consultas anidadas

Dado el siguiente modelo relacional de una base de datos, realiza las consultas SQL correspondientes en cada apartado usando consultas anidadas:



a) Devolver el id de los clientes que han comprado algún coche a un concesionario de Madrid.

b) Devolver la matrícula de los coches vendidos por algún concesionario de Madrid.

c) Devolver la marca y el modelo de los coches vendidos por algún concesionario de Barcelona.

d) Devolver el nombre y apellidos de los empleados cuyo salario está por encima de la media.

e) Devolver el nombre y apellidos de los clientes cuyo IdCliente sea menor que el del cliente llamado "Fulano DeTal".

f) Devolver el nombre y apellidos de los clientes cuyo IdCliente es menor que el de los clientes de Barcelona

g) Devolver el nombre y apellidos de los clientes cuyo IdCliente es mayor que el de los clientes de Madrid cuyo nombre empieza por "A".

h) Devolver el IdCliente de los clientes que sólo han comprado coches al concesionario con código "1"

i) Devolver el IdCliente de los clientes que han comprado coches al concesionario con código "1" y a algún otro.

Consideraciones previas

Una consulta anidada es aquella que utiliza en la cláusula WHERE el resultado generado por otra consulta (subconsulta). Las consultas anidadas se componen de 2 partes:

- Consulta interior o subconsulta: Es la primera consulta que se realiza. De ella obtenemos el valor o los valores necesarios para que se realice la consulta exterior.
- Consulta exterior: Utiliza el resultado de la consulta interior como parámetro y muestra el resultado final.

Ejemplo: Mostrar el nombre de las personas que tienen un coche rojo

- Consulta interior → Listado de coches rojos (lo buscamos por clave):
  - SELECT matricula FROM Coches WHERE color='rojo');
- Consulta exterior → Listar las personas cuya matrícula está en el listado anterior.
  - SELECT nombre FROM Personas WHERE matricula='resultado anterior';

Resultado final:

```
SELECT nombre FROM Personas WHERE matricula IN (
        SELECT matricula FROM Coches WHERE color='rojo'
);
```

Solución

a) Devolver el ID de los clientes que han comprado algún coche a un concesionario de Madrid.

```
SELECT IdCliente
FROM Venta
WHERE IdConcesionario IN (SELECT IdConcesionario
        FROM Concesionario
        WHERE Ciudad = 'Madrid');
```

b) Devolver la matrícula de los coches vendidos por algún concesionario de Madrid.

```
SELECT Matricula
FROM Venta
WHERE IdConcesionario IN (SELECT IdConcesionario
        FROM Concesionario
        WHERE Ciudad = 'Madrid');
```

c) Devolver la marca y el modelo de los coches vendidos por algún concesionario de Barcelona.

```
SELECT DISTINCT Marca, Modelo
FROM Coche
WHERE Matricula IN (SELECT Matricula
        FROM Venta
        WHERE IdConcesionario IN (SELECT IdConcesionario
                FROM Concesionario
                WHERE Ciudad = 'Barcelona'));
```

d) Devolver el nombre y apellidos de los empleados cuyo salario está por encima de la media.

```
SELECT Nombre, Apellidos
FROM Empleado
WHERE Salario > (
        SELECT AVG(Salario)
        FROM Empleado);
```

e) Devolver el nombre y apellidos de los clientes cuyo IdCliente sea menor que el del cliente llamado "Fulano DeTal".

```
SELECT Nombre, Apellidos
FROM Cliente
WHERE IdCliente < (SELECT IdCliente
        FROM Cliente
        WHERE Nombre='Fulano' AND Apellidos='DeTal');
```

f) Devolver el nombre y apellidos de los clientes cuyo IdCliente es menor que el de los clientes de Barcelona

```
SELECT Nombre, Apellidos
FROM Cliente
WHERE IdCliente < ALL (SELECT IdCliente
        FROM Venta
        WHERE IdConcesionario IN (SELECT IdConcesionario
                FROM Concesionario
                WHERE Ciudad = 'Barcelona'));
```

g) Devolver el nombre y apellidos de los clientes cuyo IdCliente es mayor que el de los clientes de Madrid cuyo nombre empieza por "A".

```
SELECT Nombre, Apellidos
FROM Cliente
WHERE IdCliente > ANY (SELECT IdCliente
        FROM Venta
        WHERE IdConcesionario IN (SELECT IdConcesionario
                FROM Concesionario
                WHERE Ciudad = 'Madrid') AND IdCliente IN (SELECT IdCliente
                FROM Cliente
                WHERE Nombre LIKE 'A%'));
```

h) Devolver el IdCliente de los clientes que sólo han comprado coches al concesionario con código "1"

```
SELECT IdCliente
FROM Venta VentaA
WHERE NOT EXISTS (SELECT *
        FROM Venta VentaB
        WHERE IdConcesionario<>1 AND VentaA.IdCliente=VentaB.IdCliente);
```

Otra forma:

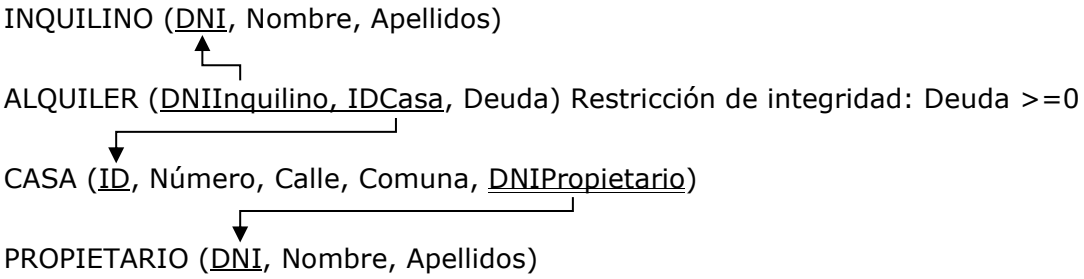
```
SELECT IdCliente
FROM Venta
WHERE IdConcesionario = 1
AND IdCliente NOT IN (SELECT IdCliente
        FROM Venta
        WHERE IdConcesionario <> 1);
```

i) Devolver el IdCliente de los clientes que han comprado coches al concesionario con código "1" y a algún otro.

```
SELECT IdCliente
FROM Venta
WHERE IdConcesionario = 1
AND IdCliente IN (SELECT IdCliente
        FROM Venta
        WHERE IdConcesionario <> 1);
```

Ejercicio 27 – Consultas SQL. Unión de tablas (JOIN)

Dado el siguiente modelo relacional de una base de datos, realiza las consultas SQL correspondientes en cada apartado usando unión de tablas:



a) Mostrar los datos de los inquilinos que viven en la casa ubicada en la calle Carrera nº 1024, Santiago

b) ¿Cuánto le deben a la propietaria María Pérez?

c) ¿Cuál es la deuda total para cada propietario?

d) Lista todas las personas de la base de datos

e) Muestra los datos de los dueños que poseen tres o más casas

f) Lista los dueños que tengan deudas en todas sus casas

Consideraciones previas

Cuando necesitamos relacionar dos o más tablas, podemos hacerlo uniendo tablas con JOIN. Existen 2 tipos de JOIN, el implícito (obsoleto) y el explícito (recomendado).

El JOIN implícito fue establecido en SQL-89 (también conocido como SQL1) como la primera forma de unir tablas. Cuando hay 2 o más tablas relacionadas en una consulta, se colocan todas las tablas separadas por comas en la cláusula FROM para devolver el producto cartesiano. Como el producto cartesiano proporciona todos los resultados posibles, se tiene que indicar en la cláusula WHERE que solo muestre aquellas filas en las que las 2 claves (primaria de una tabla y ajena de otra) coincidan.

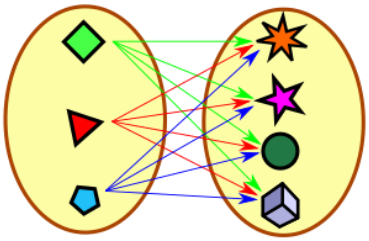
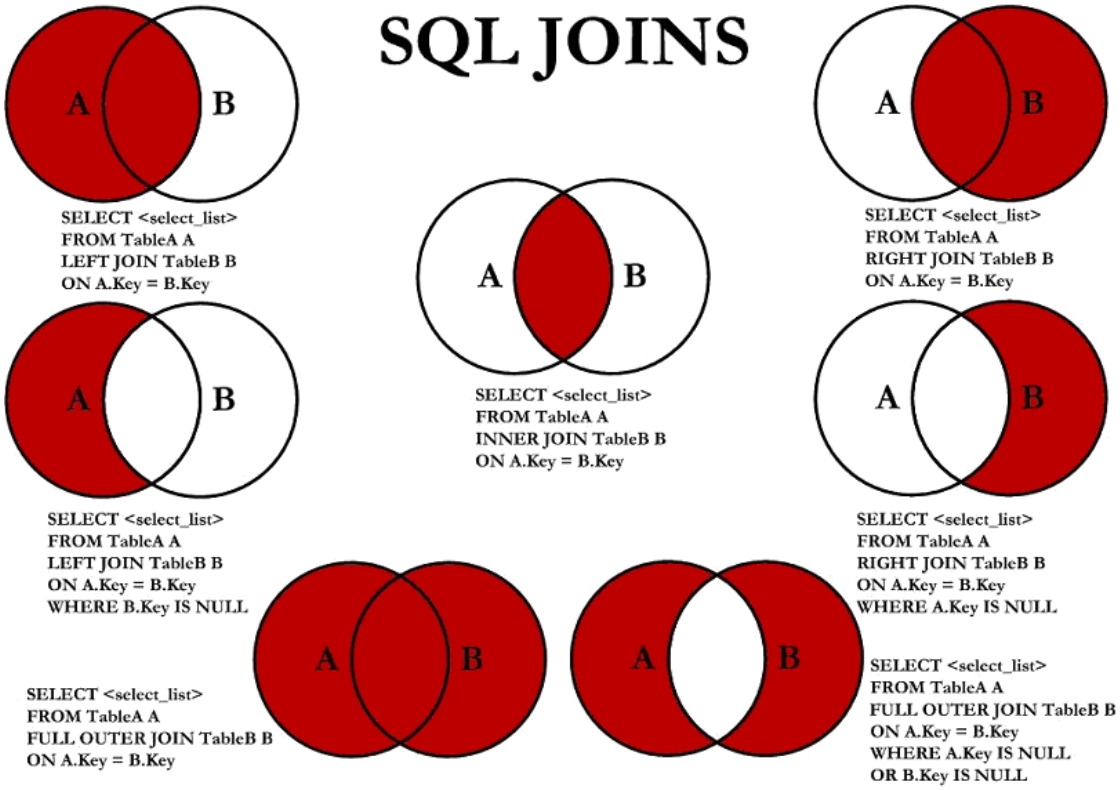


Ilustración sobre el producto cartesiano

El JOIN explícito fue establecido en SQL-92 (también conocido como SQL2) como una evolución del anterior. Cuando hay 2 o más tablas relacionadas, solo se coloca la tabla principal en la cláusula FROM y se le combinan el resto de las tablas con la sentencia JOIN. A continuación, se muestran los JOINS representados como diagramas de Venn:



Ejemplo: Mostrar el nombre de las personas que tienen un coche rojo

Con JOIN implícito establecido en SQL-89:

```
SELECT Personas.Nombre
FROM Personas, Coches
WHERE Personas.Matricula = Coches.Matricula AND Coches.Color = 'rojo';
```

Con JOIN explícito establecido en SQL-92:

```
SELECT Personas.Nombre
FROM Personas
INNER JOIN Coches ON Personas.Matricula = Coches.Matricula
WHERE Coches.Color = 'rojo';
```

Solución

a) Mostrar los datos de los inquilinos que viven en la casa ubicada en la calle Carrera nº 1024, Santiago

Con JOIN implícito:

```
SELECT I.DNI, I.Nombre, I.Apellidos
FROM Inquilino I, Alquiler A, Casa C
WHERE I.DNI = A.DNIInquilino  AND A.IDCasa=C.ID
        AND C.Calle='Carrera' AND C.Número='1024' AND C.Comuna='Santiago';
```

Con JOIN explícito (recomendado):

```
SELECT I.DNI, I.Nombre, I.Apellidos
FROM Inquilino I
INNER JOIN Alquiler A ON I.DNI = A.DNIInquilino
INNER JOIN Casa C ON A.IDCasa = C.ID
WHERE C.Calle = 'Carrera' AND C.Número = '1024' AND C.Comuna = 'Santiago';
```

b) ¿Cuánto le deben a la propietaria María Pérez?

Con JOIN implícito:

```
SELECT SUM(A.Deuda) AS Total_Deuda
FROM Alquiler A, Casa C, Propietario P
WHERE A.IDCasa = C.ID AND C.DNIPropietario = P.DNI
        AND P.Nombre = 'María' AND P.Apellidos = 'Pérez';
```

Con JOIN explícito (recomendado):

```
SELECT SUM(A.Deuda) AS Total_Deuda
FROM Alquiler A
INNER JOIN Casa C ON A.IDCasa = C.ID
INNER JOIN Propietario P ON C.DNIPropietario = P.DNI
WHERE P.Nombre = 'María' AND P.Apellidos = 'Pérez';
```

c) ¿Cuál es la deuda total para cada propietario?

Con JOIN implícito:

```
SELECT P.Nombre, P.Apellidos, SUM(A.Deuda) AS Total_Deuda
FROM Alquiler A, Casa C, Propietario P
WHERE A.IDCasa = C.ID AND C.DNIPropietario = P.DNI
GROUP BY P.DNI;
```

Con JOIN explícito (recomendado):

```
SELECT P.Nombre, P.Apellidos, SUM(A.Deuda) AS Total_Deuda
FROM Alquiler A
INNER JOIN Casa C ON A.IDCasa = C.ID
INNER JOIN Propietario P ON C.DNIPropietario = P.DNI
GROUP BY P.DNI;
```

d) Lista todas las personas de la base de datos

```
SELECT Nombre, Apellidos
FROM Inquilino
UNION
SELECT Nombre, Apellidos
FROM Propietario;
```

e) Muestra los datos de los dueños que poseen tres o más casas

Con JOIN implícito:

```
SELECT P.Nombre, P.Apellidos, COUNT(C.ID) AS NumCasas
FROM Propietario P, Casa C
WHERE P.DNI = C.DNIPropietario
GROUP BY P.DNI
HAVING COUNT (C.ID) >= 3;
```

Con JOIN explícito (recomendado):

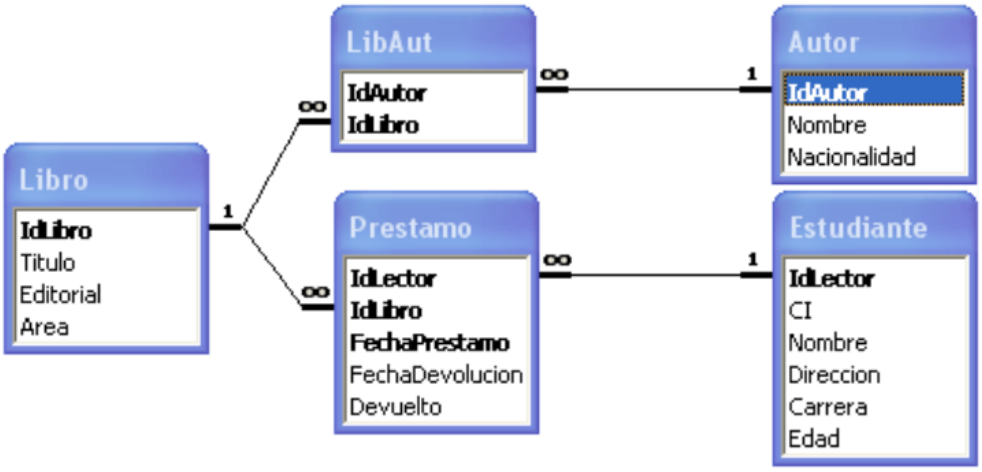
```
SELECT P.Nombre, P.Apellidos, COUNT(C.ID) AS NumCasas
FROM Casa C
INNER JOIN Propietario P ON P.DNI = C.DNIPropietario
GROUP BY P.DNI
HAVING COUNT (C.ID) >= 3;
```

f) Lista los dueños que tengan deudas en todas sus casas

```
SELECT *
FROM Propietario
WHERE DNI IN (
        SELECT DISTINCT DNIPropietario
        FROM Casa C
        WHERE NOT EXISTS (
                SELECT *
                FROM Alquiler A
                INNER JOIN Casa C1 ON A.IDCasa = C1.ID
                WHERE A.Deuda = 0
                AND C1.DNIPropietario = C.DNIPropietario);
```

Ejercicio 28 – Consultas en SQL. Repaso

Dadas las siguientes tablas relacionadas, responda a las consultas en SQL.



a) Listas los datos de los autores

b) Listar nombre y edad de los estudiantes

c) ¿Qué estudiantes pertenecen a la carrera de Informática?

d) Listar los nombres de los estudiantes cuyo nombre comience por "G".

e) ¿Qué autores son de nacionalidad USA o Francia?

f) ¿Qué libros no son del Área de Internet?

g) Listar los libros de editorial AlfayOmega

h) Listar los nombres de los autores del libro "Visual Studio Net".

i) ¿Qué libros se han prestado al lector "Raúl Valdez Alanes"?

j) Listar el nombre del estudiante de menor edad

k) Listar los estudiantes a los que se prestaron libros de Base de Datos

l) Listar los libros que pertenecen al autor Mario Benedetti

m) Listar los títulos de los libros que debían devolverse el 10/04/07

n) Hallar la suma de las edades de los estudiantes

o) Listar los datos de los estudiantes cuya edad es mayor al promedio

Solución

a) Listas los datos de los autores

```
SELECT *
FROM autor;
```

b) Listar nombre y edad de los estudiantes

```
SELECT nombre, edad
FROM estudiante;
```

c) ¿Qué estudiantes pertenecen a la carrera de Informática?

```
SELECT nombre
FROM estudiante
WHERE carrera = 'Informática';
```

d) Listar los nombres de los estudiantes cuyo apellido comience por "G".

```
SELECT nombre
FROM estudiante
WHERE nombre LIKE 'G%';
```

e) ¿Qué autores son de nacionalidad USA o Francia?

```
SELECT *
FROM autor
WHERE nacionalidad IN ('USA','Francia');
```

Otra forma:

```
SELECT *
FROM autor
WHERE nacionalidad='USA' OR nacionalidad='Francia';
```

f) ¿Qué libros no son del Área de Internet?

```
SELECT *
FROM libro
WHERE area <> 'Internet';
```

g) Listar los libros de editorial AlfayOmega

```
SELECT *
FROM libro
WHERE editorial ='AlfaOmega';
```

h) Listar los nombres de los autores del libro "Visual Studio Net".

Con consultas anidadas:

```
SELECT nombre
FROM autor
WHERE idautor IN (
        SELECT idautor
        FROM libaut
        WHERE idlibro IN (
                SELECT idlibro
                FROM libro
                WHERE título = 'Visual Studio Net'));
```

Con JOIN exclusivo:

```
SELECT nombre
FROM autor
INNER JOIN libaut ON autor.idautor = libaut.idautor
INNER JOIN libro ON libaut.idlibro = libro.idlibro
WHERE libro.título = 'Visual Studio Net';
```

i) ¿Qué libros se han prestado al lector "Raúl Valdez Alanes"?

Con consultas anidadas:

```
SELECT titulo
FROM libro
WHERE idlibro IN (
        SELECT idlibro
        FROM prestamo
        WHERE idlector IN (
                SELECT idlector
                FROM estudiante
                WHERE nombre = 'Raúl Valdez Alanes'));
```

Con JOIN exclusivo:

```
SELECT titulo
FROM libro
INNER JOIN prestamo ON libro.idlibro = prestamo.idlibro
INNER JOIN estudiante ON prestamo.idlector = estudiante.idlector
WHERE estudiante.nombre = 'Raúl Valdez Alanes';
```

j) Listar el nombre del estudiante de menor edad

```
SELECT nombre
FROM estudiante
WHERE edad IN (
        SELECT MIN (edad)
        FROM estudiante);
```

k) Listar los estudiantes a los que se prestaron libros de Base de Datos

Con consultas anidadas:

```
SELECT nombre
FROM estudiante
WHERE idlector IN (
        SELECT idlector
        FROM prestamo
        WHERE idlibro IN (
                SELECT idlibro
                FROM libro
                WHERE area = 'Base de Datos'));
```

Con JOIN exclusivo:

```
SELECT nombre
FROM estudiante
INNER JOIN prestamo ON estudiante.idlector = prestamo.idlector
INNER JOIN libro ON prestamo.idlibro = libro.idlibro
WHERE libro.area = 'Base de Datos';
```

l) Listar los libros que pertenecen al autor Mario Benedetti

Con consultas anidadas:

```
SELECT titulo
FROM libro
WHERE idlibro IN (
        SELECT idlibro
        FROM libaut
        WHERE idautor IN (
                SELECT idautor
                FROM autor
                WHERE nombre = 'Benedetti Mario'));
```

Con JOIN exclusivo:

```
SELECT titulo
FROM libro
INNER JOIN libaut ON libro.idlibro = libaut.idlibro
INNER JOIN autor ON libaut.idautor = autor.idautor
WHERE autor.nombre = 'Benedetti Mario';
```

m) Listar los títulos de los libros que debían devolverse el 10/04/07

Con consultas anidadas:

```
SELECT titulo
FROM libro
WHERE idlibro IN (
        SELECT idlibro
        FROM prestamo
        WHERE fechadevolucion = '2007-04-10' AND devuelto = False);
```

Con JOIN exclusivo:

```
SELECT titulo
FROM libro
INNER JOIN prestamo ON libro.idlibro = prestamo.idlibro
WHERE prestamo.fechadevolucion='2007-04-10' AND prestamo.devuelto=False;
```

n) Hallar la suma de las edades de los estudiantes

```
SELECT sum(edad) AS SumaEstudiantes
FROM estudiante;
```

o) Listar los datos de los estudiantes cuya edad es mayor al promedio

```
SELECT *
FROM estudiante
WHERE edad > (SELECT AVG(edad)
        FROM estudiante);
```

Ejercicio 30 – Ejercicio completo

Una cadena de hoteles desea crear una base de datos para su gestión. De cada hotel guardamos su nombre (único en toda la cadena), categoría, número de habitaciones, dirección, localidad y ciudad. De cada una de las habitaciones guardamos el número, orientación, categoría, tipo y precio.

También se guardan las reservas que se hacen en cada una de las habitaciones. De cada reserva se almacena un identificador (único en toda la cadena), fecha de entrada y de salida, régimen de alojamiento, precio de la estancia y si la estancia está pagada o no.

Cada reserva está a nombre de un huésped del que se guarda un identificador único, NIF o pasaporte, nombre, apellidos y dirección.

Por último, también se guarda el consumo que realizan los clientes en sus estancias. Para ello, el hotel va emitiendo una serie de tickets que incluyen cada uno de los productos consumidos. De cada ticket se guarda un número (único en toda la cadena), fecha, total y empleado que atendió. De cada producto se guarda un identificador, nombre y precio actual. En la composición de cada ticket se almacena la cantidad de cada producto en cada ticket y el precio del producto en el ticket.

Otras consideraciones a tener en cuenta:

- El número de habitación no tiene por qué ser único en la cadena de hoteles.
- Para asociar cada ticket con cada habitación, se hace a través de la reserva de forma que los tickets quedan ligados a una habitación y a un período de tiempo.

a) Hacer el diagrama entidad-relación.

b) Pasar a tablas (modelo relacional) el diagrama entidad-relación.

c) Reunir todos los atributos del ejercicio en una sola tabla y normalizarla hasta 3FN explicando el proceso.

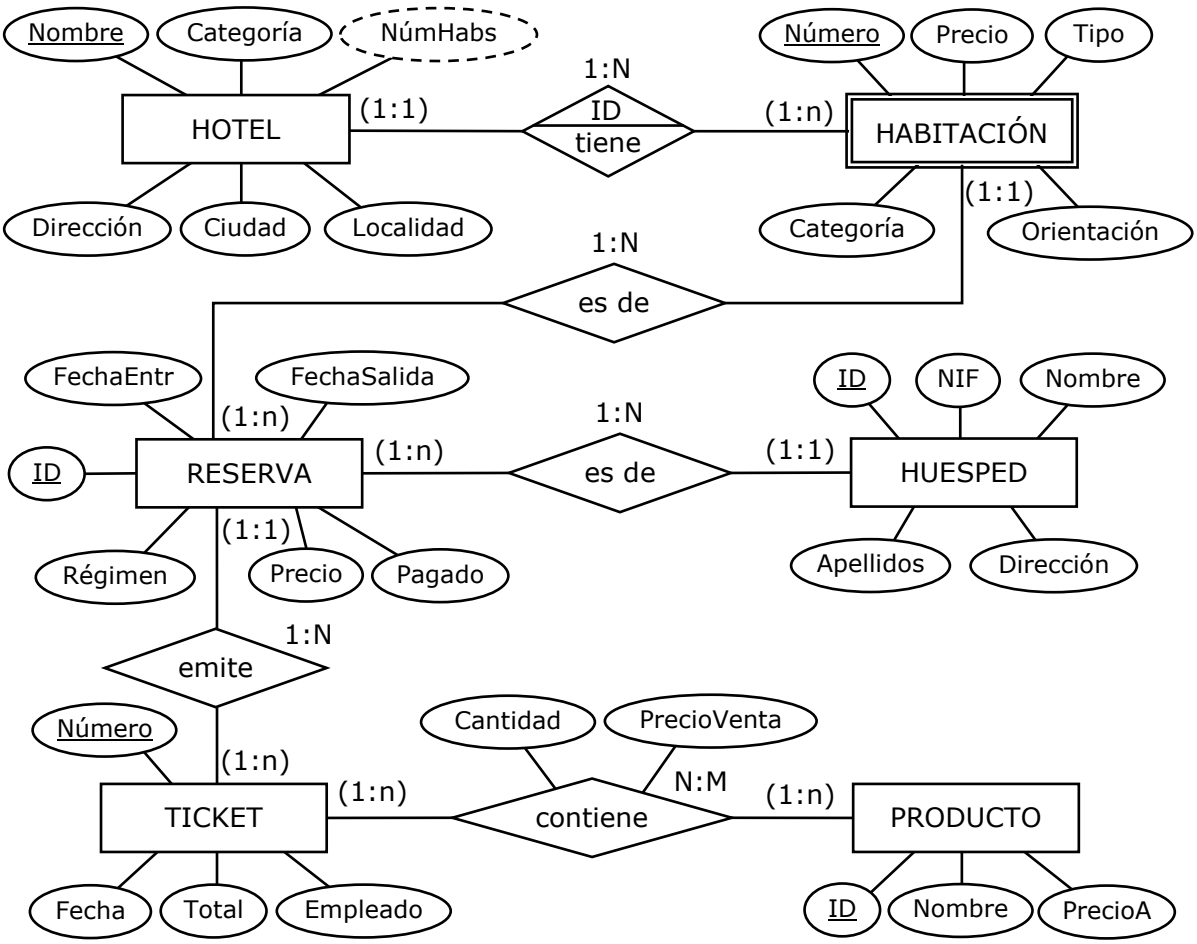
d) Crear los scripts de definición de tablas SQL.

e) Escribir las consultas SQL que realicen lo siguiente:

1. Mostrar el número de reservas hechas en el hotel H1 en el año 2007.
2. Mostrar el total a pagar por consumo (por tickets) de la habitación 111 por la reserva que acaba el 10 de diciembre de 2007.
3. Mostrar el nombre y dirección de los huéspedes que hayan realizado más de reservas en el año 2007.

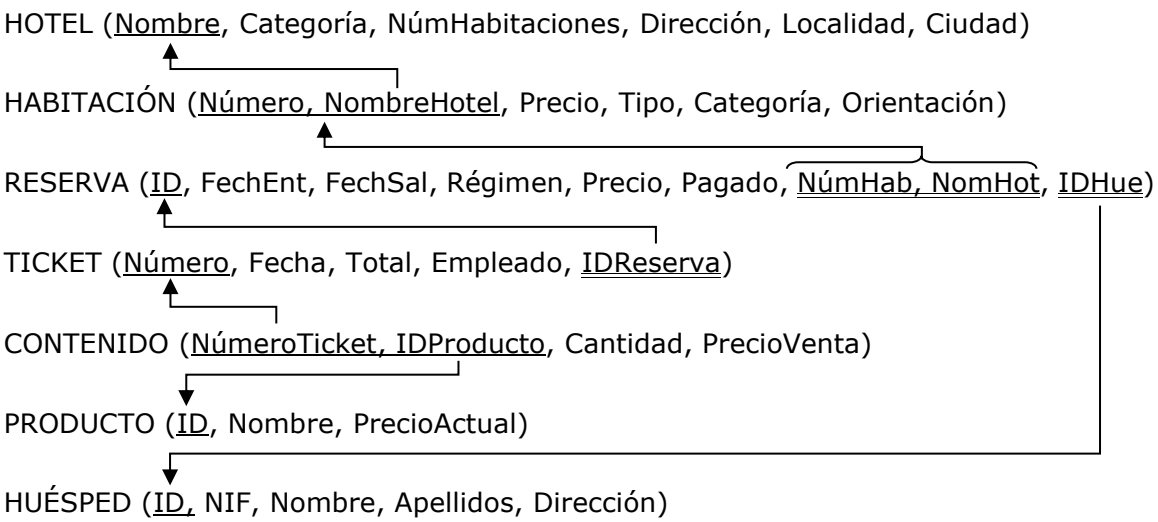
Solución

a) Hacer el diagrama entidad-relación.



b) Pasar a tablas (modelo relacional) el diagrama entidad-relación.

Aunque el atributo "NúmHabitaciones" es derivado, ya que se puede obtener a partir de la entidad "Habitación" y, por tanto, no debe ponerse en el modelo relacional, se va a poner porque en la descripción se indica que en la base de datos guardamos ese atributo.



c) Reunir todos los atributos del ejercicio en una sola tabla y normalizarla hasta 3FN explicando el proceso.

Agrupamos todos los atributos en una sola tabla llamada "RESERVA".

RESERVA (IDReserva, FechaEntrada, FechaSalida, Régimen, Precio, Pagado, NúmTicket, Total, Fecha, Empleado, Cantidad, PrecioVenta, IDProducto, NombreProducto, PrecioActual, IDHuesped, NIF, Nombre, Apellidos, Dirección, NúmeroHabitación, Orientación, Categoría, PrecioHabitación, Tipo, NombreHotel, CategoríaHotel, NúmHabitaciones, DirecciónHotel, Ciudad, Localidad)

Primera Forma Normal (1FN)

Una relación está en 1FN si no tiene grupos repetitivos, es decir, si sus atributos no tienen más de un valor por cada registro.

La relación "RESERVA" no está en 1FN porque puede tener varios registros de tickets con el mismo "IDReserva". Para pasarla a 1FN, creamos una nueva relación llamada "TICKET" que contenga los atributos multivaluados, la clave principal de la tabla original y la clave de la relación que no cumple la 1FN.

RESERVA (IDReserva, FechaEntrada, FechaSalida, Régimen, Precio, Pagado, IDHuesped, NIF, Nombre, Apellidos, Dirección, NúmeroHabitación, Orientación, Categoría, PrecioHabitación, Tipo, NombreHotel, CategoríaHotel, NúmHabitaciones, DirecciónHotel, Ciudad, Localidad)

TICKET (NúmTicket, IDReserva, Fecha, Total, Empleado, Cantidad, PrecioVenta, IDProducto, NombreProducto, PrecioActual)

Ahora analizamos la nueva relación "TICKET" y vemos que tampoco está en 1FN porque para un mismo "NúmTicket" puede haber varios productos. La pasamos a 1FN creando una nueva relación llamada "CONTENIDO" con los atributos multivaluados, la clave de la tabla original y la clave de la tabla que no cumple 1FN.

RESERVA (IDReserva, FechaEntrada, FechaSalida, Régimen, Precio, Pagado, IDHuesped, NIF, Nombre, Apellidos, Dirección, NúmeroHabitación, Orientación, Categoría, PrecioHabitación, Tipo, NombreHotel, CategoríaHotel, NúmHabitaciones, DirecciónHotel, Ciudad, Localidad)

TICKET (NúmTicket, Fecha, Total, Empleado, IDReserva)

CONTENIDO (NúmTicket, IDProducto, Cantidad, PrecioVenta, Nombre, PrecioActual)

Revisamos la nueva relación y vemos que está en 1FN.

Segunda Forma Normal (2FN)

Analizamos las 3 relaciones y vemos que la relación "CONTENIDO" no está en 2FN porque los atributos "NombreProducto" y "PrecioActual" dependen exclusivamente del atributo "IDProducto" y no de la clave completa. Lo pasamos a 2FN creando una nueva relación llamada "PRODUCTO" con estos atributos.

RESERVA (IDReserva, FechaEntrada, FechaSalida, Régimen, Precio, Pagado, IDHuesped, NIF, Nombre, Apellidos, Dirección, NúmeroHabitación, Orientación, Categoría, PrecioHabitación, Tipo, NombreHotel, CategoríaHotel, NúmHabitaciones, DirecciónHotel, Ciudad, Localidad)

TICKET (NúmTicket, Fecha, Total, Empleado, IDReserva)

CONTENIDO (NúmTicket, IDProducto, Cantidad, PrecioVenta)

PRODUCTO (IDProducto, Nombre, PrecioActual)

Tercera Forma Normal (3FN)

Una relación está en 3FN si está en 2FN y, además, no tiene dependencias funcionales transitivas entre la clave primaria y sus atributos en las que no participe una clave candidata. Vamos a ver las dependencias que existen:

En la relación "RESERVA" tenemos la dependencia transitiva {IDReserva} → {NIF, Nombre, Apellidos, Dirección} ya que hay una dependencia funcional {IDReserva} → {IDHuesped} y {IDHuesped} → {NIF, Nombre, Apellidos, Dirección} y la dependencia transitiva {IDReserva} → {Orientación, Categoría, Tipo, Precio, NombreHotel, CategoríaHotel, NúmHabitaciones, DirecciónHotel, Localidad, Ciudad} ya que esos atributos dependen funcionalmente de {NúmHabitación, NombreHotel} y no de {IDReserva}. Creamos las relaciones "HUESPED" y "HABITACIÓN":

RESERVA (IDReserva, FechaEntrada, FechaSalida, Régimen, Precio, Pagado, NúmeroHabitación, NombreHotel, IDHuesped)

HUESPED (IDHuesped, NIF, Nombre, Apellidos, Dirección)

HABITACIÓN (NúmeroHabitación, NombreHotel, Orientación, Categoría, Precio, Tipo, CategoríaHotel, NúmHabitaciones, Dirección, Ciudad, Localidad)

TICKET (NúmTicket, Fecha, Total, Empleado, IDReserva)

CONTENIDO (NúmTicket, IDProducto, Cantidad, PrecioVenta)

PRODUCTO (IDProducto, Nombre, PrecioActual)

Analizamos la relación "HABITACIÓN" y vemos que no está en 2FN porque los atributos "CategoríaHotel", "NúmHabitaciones", "Dirección", "Ciudad" y "Localidad" dependen exclusivamente del atributo "NombreHotel" y no de la clave completa. Lo pasamos a 2FN creando una nueva relación llamada "HOTEL".

RESERVA (ID, FechEnt, FechSal, Régimen, Precio, Pagado, NúmHab, NomHotel, IDHue)

HUESPED (IDHuesped, NIF, Nombre, Apellidos, Dirección)

HABITACIÓN (NúmeroHabitación, NombreHotel, Orientación, Categoría, Precio, Tipo)

HOTEL (NombreHotel, CategoríaHotel, NúmHabitaciones, Dirección, Ciudad, Localidad)

TICKET (NúmTicket, Fecha, Total, Empleado, IDReserva)

CONTENIDO (NúmTicket, IDProducto, Cantidad, PrecioVenta)

PRODUCTO (IDProducto, Nombre, PrecioActual)

d) Crear los scripts de definición de tablas SQL.

```
CREATE TABLE HOTEL (
        Nombre VARCHAR (20) NOT NULL,
        Categoría VARCHAR (50) NOT NULL,
        NúmeroHabitaciones INT NOT NULL,
        Dirección VARCHAR (200) NOT NULL,
        Localidad VARCHAR (50) NOT NULL,
        Ciudad VARCHAR (50) NOT NULL,
        PRIMARY KEY (Nombre));

CREATE TABLE HABITACIÓN (
        Número VARCHAR (10) NOT NULL,
        Orientación VARCHAR (50) NOT NULL,
        Categoría VARCHAR (50) NOT NULL,
        Tipo VARCHAR (50) NOT NULL,
        Precio FLOAT NOT NULL,
        NombreHotel VARCHAR (20) NOT NULL,
        PRIMARY KEY (Número),
        FOREIGN KEY (NombreHotel) REFERENCES HOTEL(Nombre));

CREATE TABLE HUESPED (
        ID VARCHAR (10) NOT NULL,
        NIF VARCHAR (9) NOT NULL,
        Nombre VARCHAR (50) NOT NULL,
        Apellidos VARCHAR (100) NOT NULL,
        Dirección VARCHAR (200) NOT NULL,
        PRIMARY KEY (ID));

CREATE TABLE RESERVA (
        ID VARCHAR (10) NOT NULL,
        FechaEntrada DATE NOT NULL,
        FechaSalida DATE NOT NULL,
        Régimen VARCHAR (50) NOT NULL,
        Precio FLOAT NOT NULL,
        Pagado BOOLEAN NOT NULL,
        NombreHotel VARCHAR (20) NOT NULL,
        NúmeroHabitación VARCHAR (10) NOT NULL,
        IDHuésped VARCHAR (10) NOT NULL,
        PRIMARY KEY (ID),
        FOREIGN KEY (NombreHotel) REFERENCES HOTEL(Nombre),
        FOREIGN KEY (NúmeroHabitación) REFERENCES HABITACIÓN(Número),
        FOREIGN KEY (IDHuésped) REFERENCES HUESPED(ID));

CREATE TABLE TICKET (
        Número INT NOT NULL AUTO_INCREMENT,
        Fecha DATE NOT NULL,
        Total FLOAT NOT NULL,
        Empleado VARCHAR (200) NOT NULL,
        IDReserva VARCHAR (10) NOT NULL,
```

```
        PRIMARY KEY (Número),
        FOREIGN KEY (IDReserva) REFERENCES RESERVA(ID));

CREATE TABLE PRODUCTO (
        ID VARCHAR (10) NOT NULL,
        Nombre VARCHAR (50) NOT NULL,
        PrecioActual FLOAT NOT NULL,
        PRIMARY KEY (ID));

CREATE TABLE CONTENIDO (
        NúmeroTicket VARCHAR (10) NOT NULL,
        IDProducto VARCHAR (10) NOT NULL,
        Nombre VARCHAR (50) NOT NULL,
        Cantidad INT NOT NULL,
        Precio FLOAT NOT NULL,
        PRIMARY KEY (NúmeroTicket, IDProducto),
        FOREIGN KEY (NúmeroTicket) REFERENCES TICKET(Número),
        FOREIGN KEY (IDProducto) REFERENCES PRODUCTO(ID));
```

e) Escribir las consultas SQL que realicen lo siguiente:

1. Mostrar el número de reservas hechas en el hotel H1 en el año 2007.

Opción 1: Usando BETWEEN

```
SELECT COUNT(*) AS NumReservas
FROM RESERVA
WHERE NombreHotel = 'H1'
        AND FechaEntrada BETWEEN '2007-01-01' AND '2008-01-01';
```

Opción 2: Usando YEAR

```
SELECT COUNT(*) AS NumReservas
FROM RESERVA
WHERE NombreHotel = 'H1' AND YEAR(FechaEntrada) = 2007;
```

2. Mostrar el total a pagar por consumo (por tickets) de la habitación 111 por la reserva que acaba el 10 de diciembre de 2007.

Opción 1: Mediante consultas anidadas

```
SELECT SUM(TICKET.Total) AS TotalAPagar
FROM TICKET
WHERE IDReserva IN (
        SELECT ID
        FROM RESERVA
        WHERE NúmeroHabitación = '111'
        AND FechaSalida = '2007-12-10';
```

Opción 2: Mediante JOIN implícito

```
SELECT SUM(TICKET.Total) AS TotalAPagar
FROM TICKET, RESERVA
WHERE TICKET.IDReserva = RESERVA.ID
        AND RESERVA.Número = '111'
        AND RESERVA.FechaSalida = '2007-12-10';
```

Opción 3: Mediante JOIN explícito

```
SELECT SUM(TICKET.Total) AS PagoTotal
FROM TICKET
INNER JOIN RESERVA ON TICKET.IDReserva = RESERVA.ID
WHERE RESERVA.Número = '111'
        AND RESERVA.FechaSalida = '2007-12-10';
```

3. Mostrar el nombre y dirección de los huéspedes que hayan realizado más de 3 reservas en el año 2007.

Opción 1: Mediante consultas anidadas

```
SELECT Nombre, Dirección
FROM HUESPED
WHERE ID IN (
        SELECT IDHuésped
        FROM RESERVA
        WHERE RESERVA.FechaEntrada BETWEEN '2007-01-01' AND '2008-01-01'
        GROUP BY IDHuésped
        HAVING COUNT(*)>3);
```

Opción 2: Mediante JOIN explícito

```
SELECT H.Nombre, H.Dirección
FROM HUESPED H
INNER JOIN RESERVA R ON H.ID = R.IDHuésped
WHERE R.FechaEntrada BETWEEN '2007-01-01' AND '2008-01-01'
GROUP BY H.ID
HAVING COUNT(*) > 3;
```

Ejercicio 31 – Examen oposiciones PES 2021

Apartado 1

Una gran multinacional ha comprado varias compañías de salud y desea diseñar una base de datos que recoja la información relativa a las distintas compañías y a los asegurados. Para el diseño de dicha base de datos se debe tener en cuenta lo siguiente:

Existen varias compañías de salud de las que nos interesa almacenar el cif, un nombre y un conjunto de teléfonos que estarán disponible para que los clientes contacten con ellos. Además, cada compañía dispone de un conjunto de seguros que ofrecerá a sus asegurados.

Las compañías de salud contratan convenios con determinados hospitales, al menos con uno, aunque a las compañías les interesa tener muchos hospitales para poder convencer a los clientes. Los distintos hospitales podrán trabajar con varias compañías (al menos una) y de los hospitales se debe almacenar el nombre, que será único, y la razón social o dirección, así como la fecha en la que contrató el convenio con cada una de las compañías de salud con las que las que tenga convenio.

De los asegurados se debe almacenar su dni o cif, el nombre, apellidos y la dirección. Los asegurados podrán contratar varios seguros (salud, dental, con copago, etc), y cada seguro será de una única compañía. De los distintos seguros se deberá almacenar un nombre, que será único, y una descripción de las condiciones. Cada seguro es contratado por un único asegurado.

Existen dos tipos de asegurados, los que disponen de una mutualidad, de los que interesa saber el nombre de la mutua y los clientes que han contratado el seguro de forma privada, de los que necesitaremos conocer su número de cuenta y el coste de la mensualidad. Además, podemos tener asegurados que dependen de otro asegurado y por lo tanto tendrán los mismos seguros, por ejemplo, los hijos que están incluido en el seguro de uno de sus progenitores.

Diseñar el esquema E/R empleando el modelo extendido.

Apartado 2

Se consideran las siguientes tablas en una aplicación en la que se tienen los datos de una persona, de la empresa en la que trabaja y de los padres de la misma:

PERSONA (dni, nombre, apellidos, empresa, salario)
PK(dni)
FK(empresa/Empresa(cif))

PADREMADRE(dniPadreMadre, dniHijo)
PK(dniPadreMadre, dniHijo)
FK(dniPadreMadre/Persona(dni)
FK(dniHijo/Persona(dni)

EMPRESA(cif, nombre, direccion)
PK(cif)

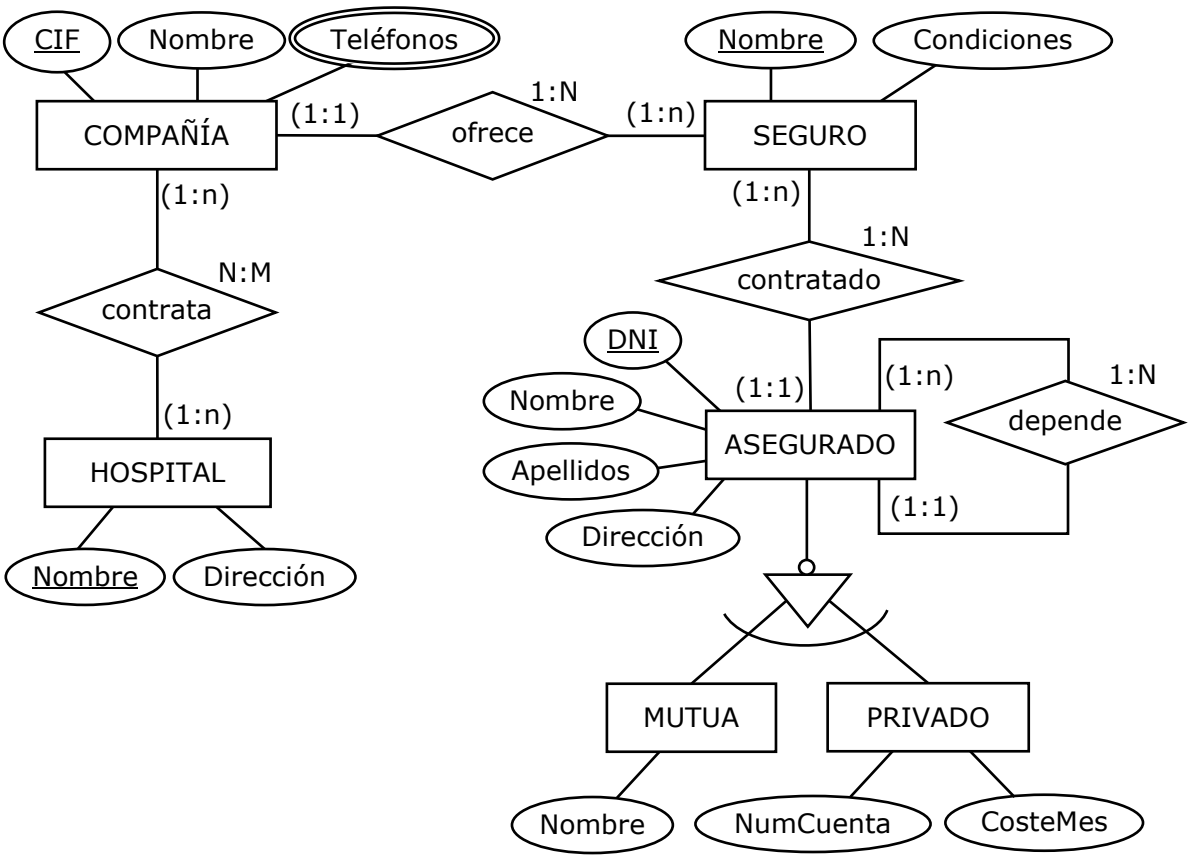
Escriba las sentencias SQL para realizar las siguientes consultas:

1. Obtener el nombre y apellido del padre y de la madre de "Pepe Pérez Pérez"
2. Obtener un listado con el nombre de las empresas y el total de los salarios de todos los empleados de la empresa ordenados de forma que las empresas que gasten más dinero en sus salarios se muestren al principio.
3. Subir el salario un 10% a todas las personas que trabajen en las empresas de Sevilla (Considere que en el campo dirección se encuentra entre otros datos la provincia).

4. Mostrar el nombre y apellido de las personas que cobran más que la media del salario de todas personas que están recogidas en la base de datos.

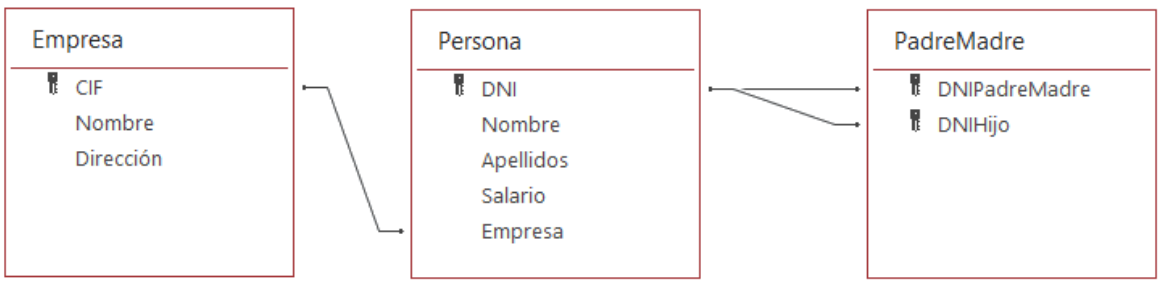
Solución

Apartado 1



Apartado 2

Las tablas y sus relaciones son las siguientes:



1. Obtener el nombre y apellido del padre y de la madre de "Pepe Pérez Pérez"

   Opción 1: Mediante consultas anidadas

   SELECT Persona.Nombre, Persona.Apellidos

```
FROM Persona
WHERE Persona.DNI IN (
        SELECT PadreMadre.DNIPadreMadre
        FROM PadreMadre
        WHERE PadreMadre.DNIHijo IN (
                SELECT Persona.DNI
                FROM Persona
                WHERE Persona.Nombre = 'Pepe'
                AND Persona.Apellidos = 'Pérez Pérez'));
```

Opción 2: Mediante JOIN explícito

```
SELECT P1.Nombre, P1.Apellidos
FROM Persona P1
INNER JOIN PadreMadre PM ON P1.DNI = PM.DNIPadreMadre
INNER JOIN Persona P2 ON PM.DNIPadreMadre = P2.DNI
WHERE P2.Nombre = 'Pepe'
AND P2.Apellidos = 'Pérez Pérez';
```

Opción 3: Mediante una mezcla de los 2

```
SELECT Persona.Nombre, Persona.Apellidos
FROM Persona
INNER JOIN PadreMadre ON Persona.DNI = PadreMadre.DNIPadreMadre
WHERE PadreMadre.DNIHijo IN (
        SELECT Persona.DNI
        FROM Persona
        WHERE Persona.Nombre = 'Pepe'
        AND Persona.Apellidos = 'Pérez Pérez');
```

2. Obtener un listado con el nombre de las empresas y el total de los salarios de todos los empleados de la empresa ordenados de forma que las empresas que gasten más dinero en sus salarios se muestren al principio.

Opción 1: Mediante JOIN implícito

```
SELECT Empresa.Nombre, SUM(Persona.Salario) AS Salarios
FROM Persona, Empresa
WHERE Persona.Empresa = Empresa.CIF
GROUP BY Empresa.Nombre
ORDER BY Salarios DESC;
```

Opción 2: Mediante JOIN explícito

```
SELECT Empresa.Nombre, SUM(Persona.Salario) AS Salarios
FROM Persona
INNER JOIN Empresa ON Persona.Empresa = Empresa.CIF
GROUP BY Empresa.Nombre
ORDER BY Salarios DESC;
```

3. Subir el salario un 10% a todas las personas que trabajen en las empresas de Sevilla (Considere que en el campo dirección se encuentra entre otros datos la provincia).

   Opción 1: Mediante consultas anidadas

```
UPDATE Persona
SET Salario = (Salario * 1.1)
WHERE Empresa IN (SELECT CIF
        FROM Empresa
        AND Dirección LIKE '%Sevilla%');
```

   Opción 2: Mediante JOIN implícito

```
UPDATE Persona, Empresa
SET Persona.Salario = (Persona.Salario * 1.1)
WHERE Persona.Empresa = Empresa.Cif
AND Empresa.Dirección LIKE '%Sevilla%';
```

   Opción 3: Mediante JOIN explícito

```
UPDATE Persona
INNER JOIN Empresa ON Empresa.CIF = Persona.Empresa
SET Persona.Salario = (Persona.Salario * 1.1)
WHERE Empresa.Dirección LIKE '%Sevilla%';
```

4. Mostrar el nombre y apellido de las personas que cobran más que la media del salario de todas personas que están recogidas en la base de datos.

   Opción 1: Mediante consultas anidadas

```
SELECT Nombre, Apellidos
FROM Persona
WHERE Salario > (SELECT AVG(Salario) FROM Persona);
```

   Opción 2: Mediante JOIN explícito

```
SELECT Nombre, Apellidos
FROM Persona
INNER JOIN (SELECT AVG(Salario) AS SalarioMedio
        FROM Persona) AS TablaGenerada
        WHERE Persona.Salario > TablaGenerada.SalarioMedio;
```