

# Introducción a SQL

## (Structured Query Language)

**Especialista:** Junior Rodriguez



**SQL SERVER**



# MODELAMIENTO DE DATOS

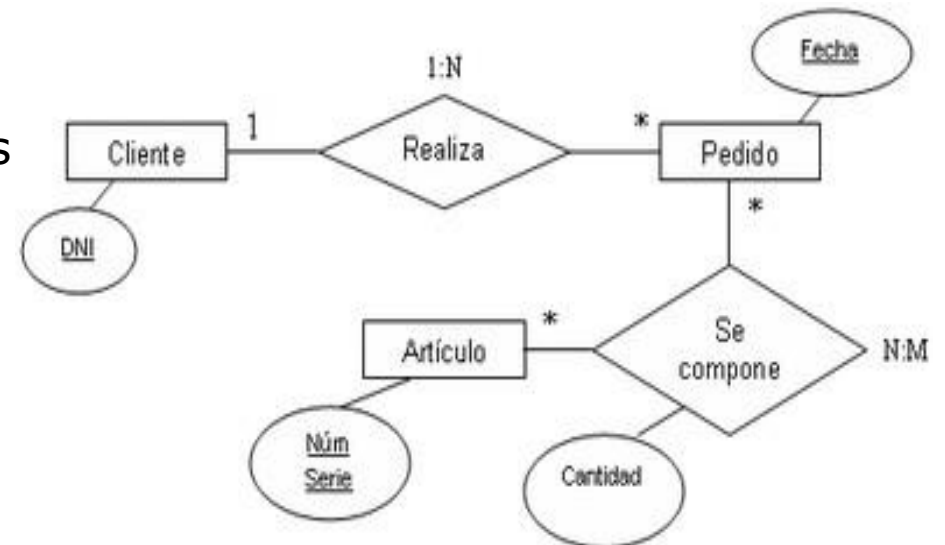
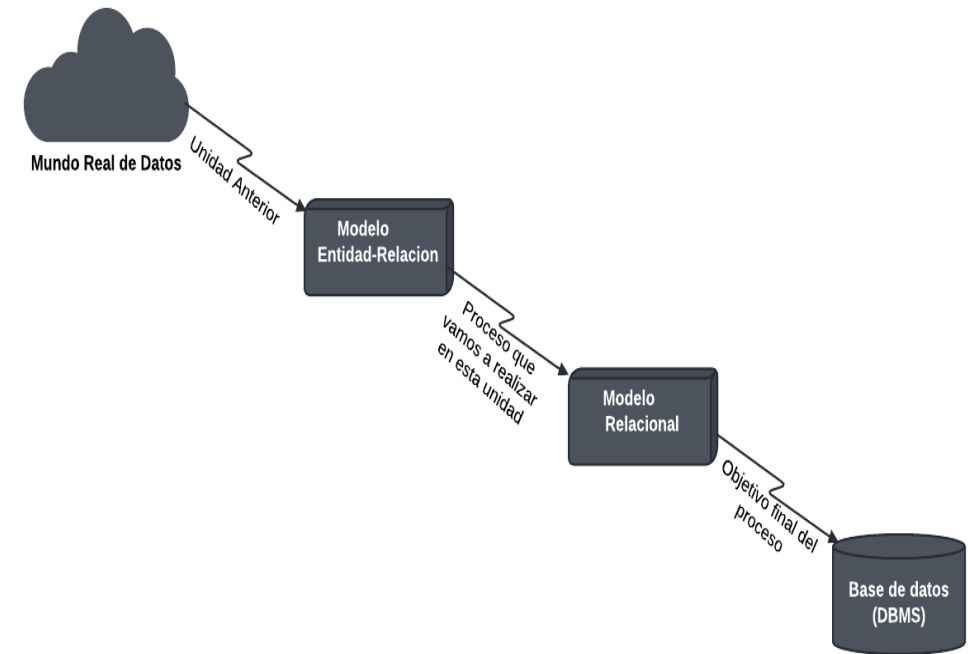
Los modelos facilitan la comunicación entre el diseñador de base de datos y los usuarios finales. Los modelos son fáciles de utilizar y cambiar, ya que son sólo una imagen muy simplificada del sistema de información que se desea desarrollar

## Conceptual

Un modelo conceptual de datos identifica las relaciones de más alto nivel entre las diferentes entidades.

## Características

- Incluye las entidades importantes y las relaciones entre ellas
- No se especifica ningún atributo.
- No se especifica ninguna clave principal.

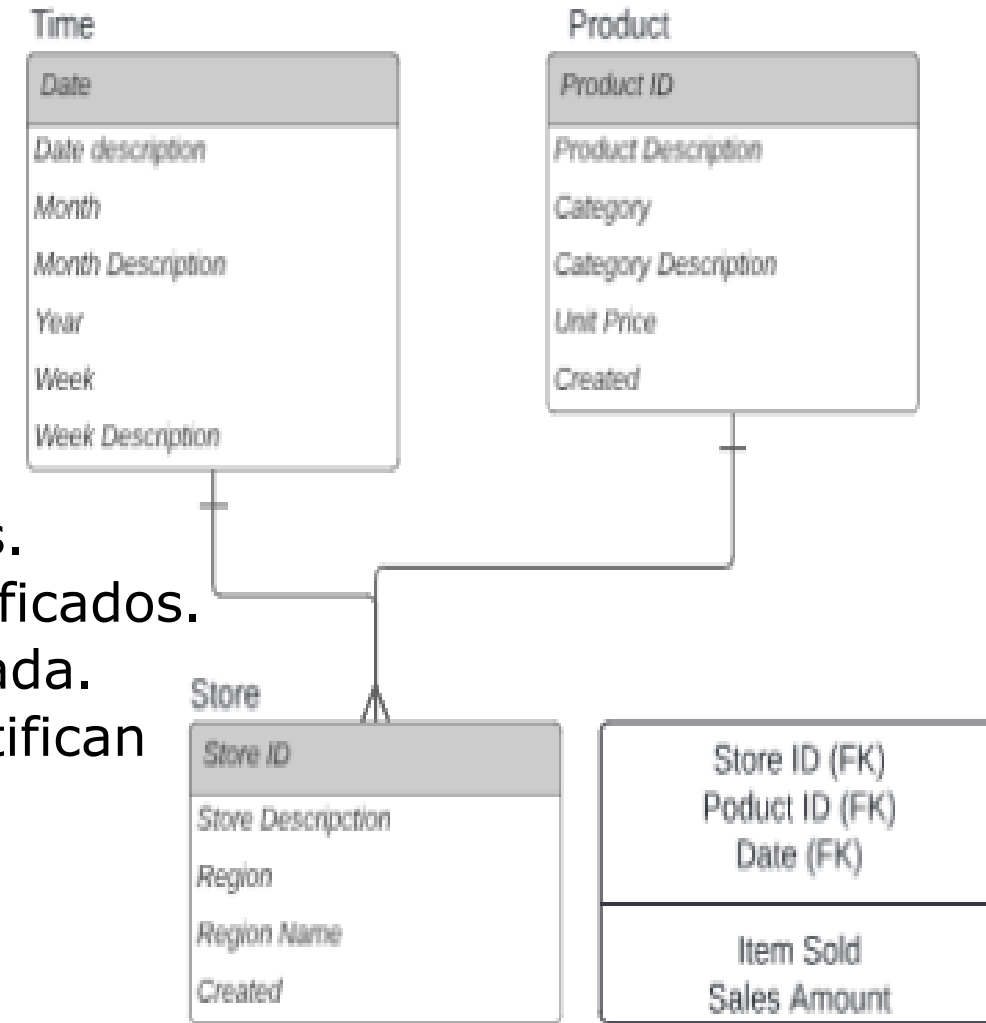


# Modelamiento Lógico

Un modelo de datos lógicos describe los datos con el mayor detalle posible, independientemente de cómo se implementarán físicamente en la base de datos.

## Características

- Incluye todas las entidades y relaciones entre ellos.
- Todos los atributos para cada entidad están especificados.
- La clave principal para cada entidad está especificada.
- Se especifican las claves externas (claves que identifican la relación entre diferentes entidades).
- La normalización ocurre en este nivel.

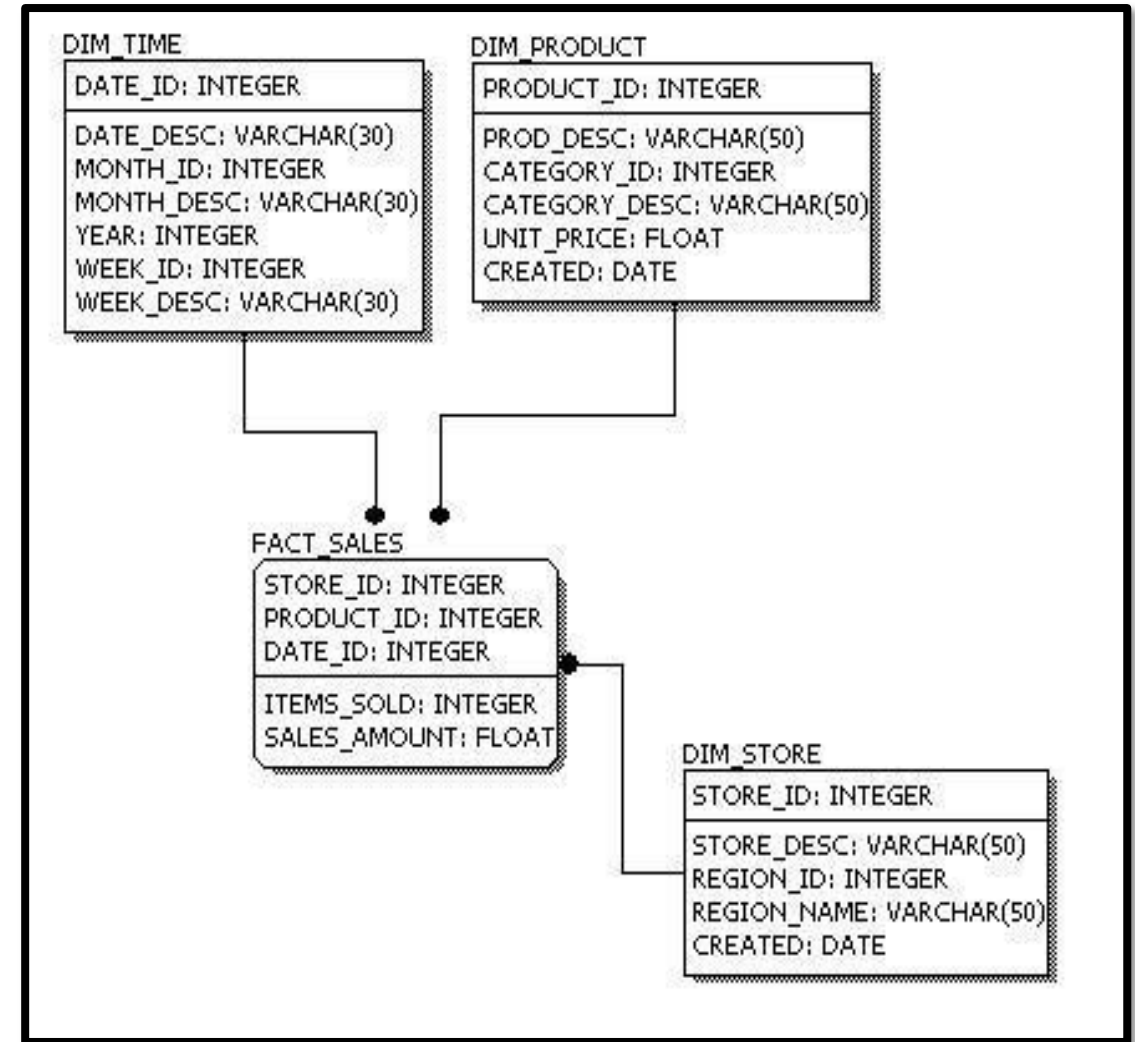


# Modelamiento Físico

El modelo de datos físicos representa cómo se construirá el modelo en la base de datos. Un modelo de base de datos física muestra todas las estructuras de tabla, incluidos el nombre de columna, el tipo de datos de columna, las restricciones de columna, la clave principal, la clave externa y las relaciones entre las tablas.

## Características

- Especificación de todas las tablas y columnas.
- Las claves externas se usan para identificar relaciones entre tablas.
- La desnormalización puede ocurrir según los requisitos del usuario.



# NORMALIZACIÓN DE LOS DATOS

La parte central de los principios del modelo relacional es el concepto de normalización, una técnica para producir un conjunto de relaciones que poseen un conjunto de ciertas propiedades que minimizan los datos redundantes y preservan la integridad de los datos almacenados tal como se mantienen (Agregados, Actualizados y Eliminados).

## **Elección de un identificador único:**

Un identificador único es un atributo o conjunto de atributos que únicamente identifican cada fila de datos en una relación. El identificador único eventualmente se convertirá en la clave principal de la tabla creada en la base de datos física desde la relación de normalización, pero muchos usan los términos identificador único y clave principal de manera intercambiable. A cada identificador potencial único se le denomina candidato clave, y cuando hay varios candidatos, el diseñador elegirá el mejor, el cual es el menos probable de cambiar valores o el más simple y/o el más corto.

**Primera forma normal** La primera forma normal incluye las siguientes directrices:

- Cada atributo de una tupla contiene sólo un valor.
- Cada tupla en una relación contiene el mismo número de atributos.
- Cada tupla es diferente, lo que significa que la combinación de los valores de todos los atributos de una tupla dada no puede ser como ninguna otra tupla en la misma relación.

**La segunda tupla y la última tupla** violan la primera forma normal. En la segunda tupla, el atributo **NOMBRE\_CD** y el atributo **AÑO\_DERECHOSDEAUTOR** contienen cada una dos valores. En la última tupla, el atributo **NOMBRE\_ARTISTA** contiene tres valores.

NOMBRE_ARTISTA	NOMBRE_CD	AÑO_DERECHOSDEAUTOR
Jennifer Warnes	Famous Blue Raincoat	1991
Joni Mitchell	Blue; Court and Spark	1971; 1974
William Ackerman	Past Light	1983
Kitaro	Kojiki	1990
Bing Crosby	That Christmas Feeling	1993
Patsy Cline	Patsy Cline: 12 Greatest Hits	1988
Jose Carreras; Placido Domingo; Luciano Pavarotti	Carreras Domingo Pavarotti in Concert	1990



Para normalizar la relación, debe crear relaciones adicionales que separen los datos de modo que cada atributo contenga un solo valor, cada tupla contenga el mismo número de atributos y cada tupla sea diferente. Ahora los datos se ajustan a la primera forma normal.

ID_ARTISTA	NOMBRE_ARTISTA
10001	Jennifer Warnes
10002	Joni Mitchell
10003	William Ackerman
10004	Kitaro
10005	Bing Crosby
10006	Patsy Cline
10007	Jose Carreras
10008	Placido Domingo
10009	Luciano Pavarotti

ID_ARTISTA	ID_CD
10001	99301
10002	99302
10002	99303
10003	99304
10004	99305
10005	99306
10006	99307
10007	99308
10008	99308
10009	99308

ID_CD	NOMBRE_CD	AÑO_DERECHOSDEAUTOR
99301	Famous Blue Raincoat	1991
99302	Blue	1971
99303	Court and Spark	1974
99304	Past Light	1983
99305	Kojiki	1990
99306	That Christmas Feeling	1993
99307	Patsy Cline: 12 Greatest Hits	1988
99308	Carreras Domingo Pavarotti in Concert	1990

**Segunda forma normal:**

Para comprender la segunda forma normal, primero Debe entender el concepto de dependencia funcional. Para esta definición se usarán dos atributos arbitrarios, llamados A y B.

El atributo B es funcionalmente dependiente (dependiente para abreviar) del atributo A si en cualquier momento no hay más que un valor del atributo B asociado con el valor dado al atributo A. Si se dice que el atributo B es funcionalmente dependiente del atributo A, también estaremos diciendo que el atributo A determina al atributo B, o que A es un factor determinante (identificador único) del atributo B. En la figura,

**AÑO\_DERECHOSDEAUTOR** depende de **ID\_CD**, ya que sólo puede haber un valor de **AÑO\_DERECHOSDEAUTOR** para cualquier CD. Dicho de otra manera, **ID\_CD** es un factor determinante de **AÑO\_DERECHOSDEAUTOR**.

← Identificador único →

ID_ARTISTA	ID_CD	AÑO_DERECHOSDEAUTOR
10001	99301	1991
10002	99302	1971
10002	99303	1974
10003	99304	1983
10004	99305	1990
10005	99306	1993
10006	99307	1988



# MODELO ENTIDAD - RELACIÓN

## Entidad

Las entidades representan cosas u objetos (ya sean reales o abstractos), que se diferencian claramente entre sí. Supongamos un contexto de taller mecánico, donde se podría crear las siguientes entidades:

- **Coches** (objeto físico): contiene la información de cada taller.
- **Empleado** (objeto físico): información de los trabajadores.
- **Cargo del empleado** (cosa abstracta): información de la función del empleado.

Estas entidades se representan en un diagrama con un rectángulos.

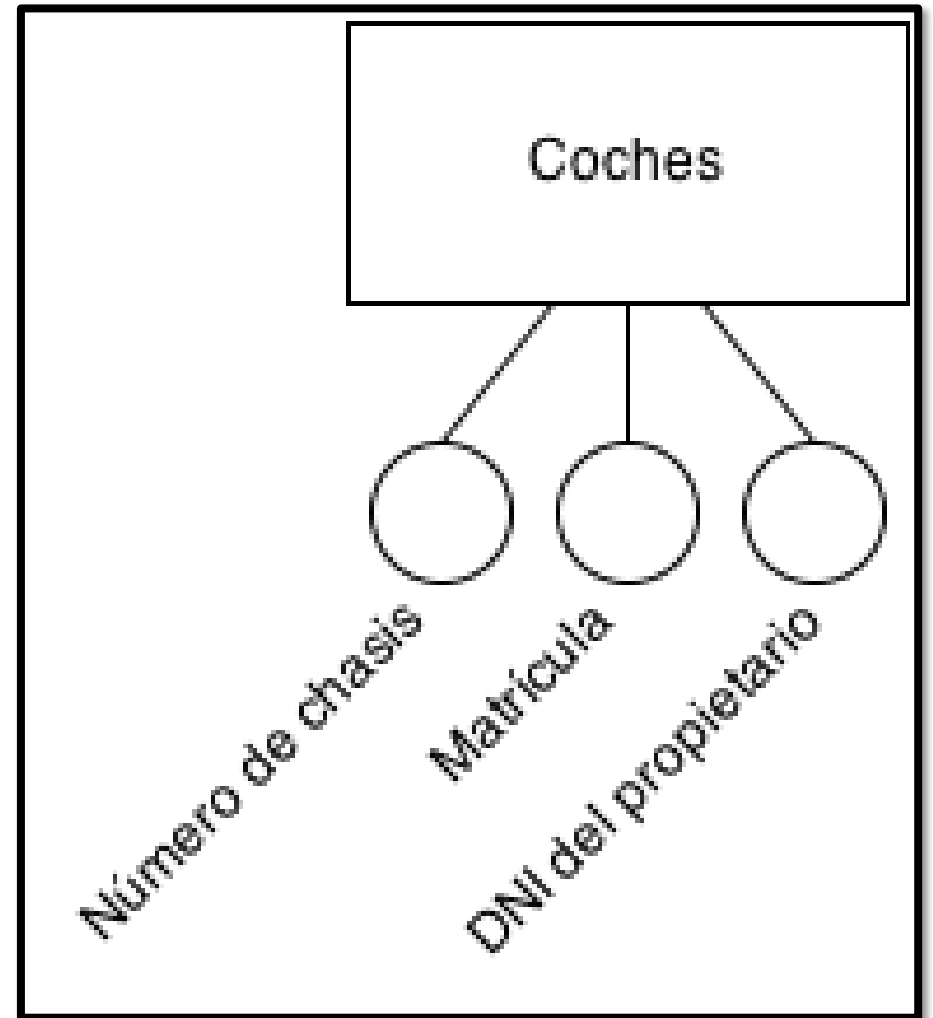


# Atributos

Los atributos definen o identifican las características de entidad (es el contenido de esta entidad). Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha...).

Unos posibles atributos de la entidad Coches serían los siguientes: número de chasis, matrícula, DNI del propietario, marca, modelo y muchos otros que complementen la información de cada coche.

Los atributos se representan como círculos que descenden de una entidad, y no es necesario representarlos todos, sino los más significativos.



En un modelo relacional (ya implementado en una base de datos) un ejemplo de tabla dentro de una BBDD podría ser el siguiente.

Número de chasis	Matrícula	DNI del propietario
5tfem5f10ax007210	4817 BFK	45338600L
6hsen2j98as001982	8810 CLM	02405068K
5rgsb7a19js001982	0019 GGL	40588860J

Este ejemplo es con tres atributos, pero un coche podría tener cientos (si fuese necesario) y seguirían la misma estructura de columnas, tras implementarlo en una BBDD.

# Relación

Es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable.

Por ejemplo, los empleados del taller (de la entidad "Empleados") tienen un cargo (según la entidad "Cargo del empleado"). Es decir, un atributo de la entidad "Empleados" especificará que cargo tiene en el taller, y tiene que ser idéntico al que ya existe en la entidad "Cargo del empleado".

Las relaciones se muestran en los diagramas como rombos, que se unen a las entidades mediante líneas.



Una posible representación en una BBDD sería:

### Empleados

Nombre	DNI	Cargo
Carlos Sánchez	45338600L	001
Pepe Sánchez	02405068K	002
Juan Sánchez	40588860J	002

### Cargo del empleado

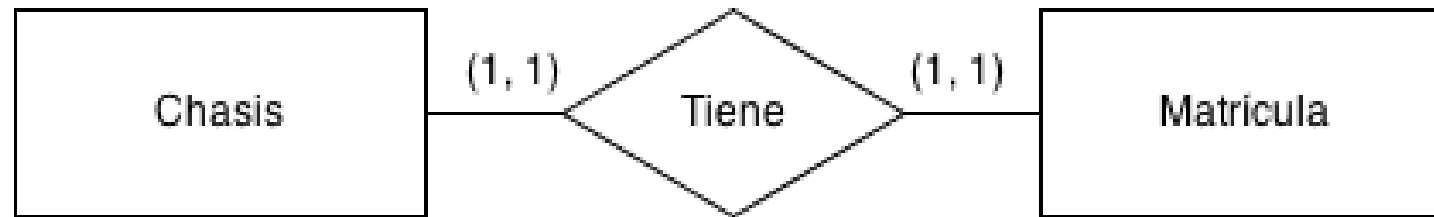
ID del cargo	Descripción
001	Jefe de taller
002	Mecánico

## Relación de cardinalidad

Podemos encontrar distintos tipos de relaciones según como participen en ellas las entidades. Es decir, en el caso anterior cada empleado puede tener un cargo, pero un mismo cargo lo pueden compartir varios empleados. Esto complementa a las representaciones de las relaciones, mediante un intervalo en cada extremo de la relación que especifica cuantos objetos o cosas (de cada entidad) pueden intervenir en esa relación.

Uno a Uno

Una entidad se relaciona únicamente con otra y viceversa. Por ejemplo, si tuviésemos una entidad con distintos chasis y otra con matrículas deberíamos de determinar que cada chasis solo puede tener una matrícula (y cada matrícula un chasis, ni más en ningún caso).





Uno a varios  
o varios a  
uno

Determina que un registro de una entidad puede estar relacionado con varios de otra entidad, pero en esta entidad existir solo una vez.



Varios a  
varios

Determina que una entidad puede relacionarse con otra con ninguno o varios registros y viceversa. Por ejemplo, en el taller un coche puede ser reparado por varios mecánicos distintos y esos mecánicos pueden reparar varios coches distintos.



## Claves

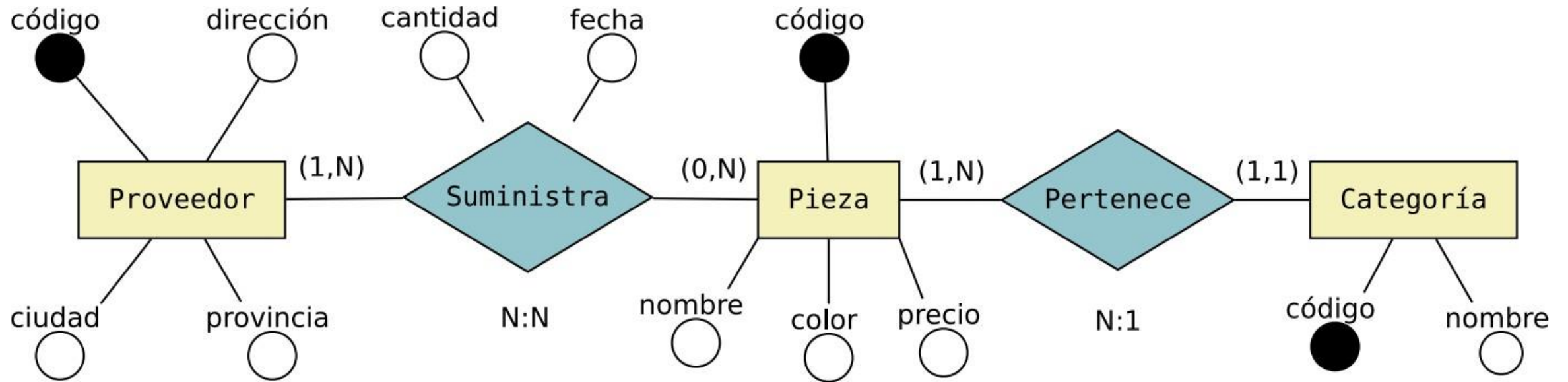
Es el atributo de una entidad, al que le aplicamos una restricción que lo distingue de los demás registros (no permitiendo que el atributo específico se repita en la entidad) o le aplica un vínculo.

- **Clave primaria:** identifica inequívocamente un solo atributo no permitiendo que se repita en la misma entidad. Como sería la matrícula o el número de chasis de un coche (no puede existir dos veces el mismo).
- **Clave externa o clave foránea:** este campo tiene que estar estrictamente relacionado con la clave primaria de otra entidad, para así exigir que exista previamente ese clave. Por ejemplo, un empleado indispensablemente tiene que tener un cargo, por lo cual si intentásemos darle un cargo inexistente el gestor de bases de datos nos devolvería un error.

## **Ejemplo 1**

- Tenemos que diseñar una base de datos sobre proveedores y disponemos de la siguiente información:
- De cada proveedor conocemos su nombre, dirección, ciudad, provincia y un código de proveedor que será único para cada uno de ellos.
- Nos interesa llevar un control de las piezas que nos suministra cada proveedor. Es importante conocer la cantidad de las diferentes piezas que nos suministra y en qué fecha lo hace. Tenga en cuenta que un mismo proveedor nos puede suministrar una pieza con el mismo código en diferentes fechas. El diseño de la base de datos debe permitir almacenar un histórico con todas las fechas y las cantidades que nos ha proporcionado un proveedor.
- Una misma pieza puede ser suministrada por diferentes proveedores.
- De cada pieza conocemos un código que será único, nombre, color, precio y categoría.
- Pueden existir varias categorías y para cada categoría hay un nombre y un código de categoría único.
- Una pieza sólo puede pertenecer a una categoría.

## A) DIAGRAMA E/R



## B) Modelo Lógico

Las reglas de transformación de E/R al modelo relacional nos dicen que la relación **Suministra** genera una nueva tabla porque es una relación de cardinalidad N:N. Esta nueva tabla recibe las claves primarias de las dos entidades que participan en la relación y además participan como clave primaria.

La solución teórica sería la siguiente:

- PROVEEDOR\_SUMINISTRA\_PIEZA(**código\_proveedor, código\_pieza**, fecha, cantidad)
  - código\_proveedor: FOREIGN KEY de PROVEEDOR(código)
  - código\_pieza: FOREIGN KEY de PIEZA(código)

Con esta solución podemos tener un problema en el caso de que un proveedor nos suministre piezas con el mismo código en fechas diferentes. En este caso no podríamos almacenar esta información en la tabla porque se produciría un error de claves primarias duplicadas.

Para solucionarlo podemos incluir el atributo fecha como parte de la clave primaria de la tabla, de modo que la clave primaria estaría compuesta por código\_proveedor, código\_pieza y fecha. La solución sería la siguiente:

- PROVEEDOR\_SUMINISTRA\_PIEZA(**código\_proveedor, código\_pieza, fecha**, cantidad)
  - código\_proveedor: FOREIGN KEY de PROVEEDOR(código)
  - código\_pieza: FOREIGN KEY de PIEZA(código)

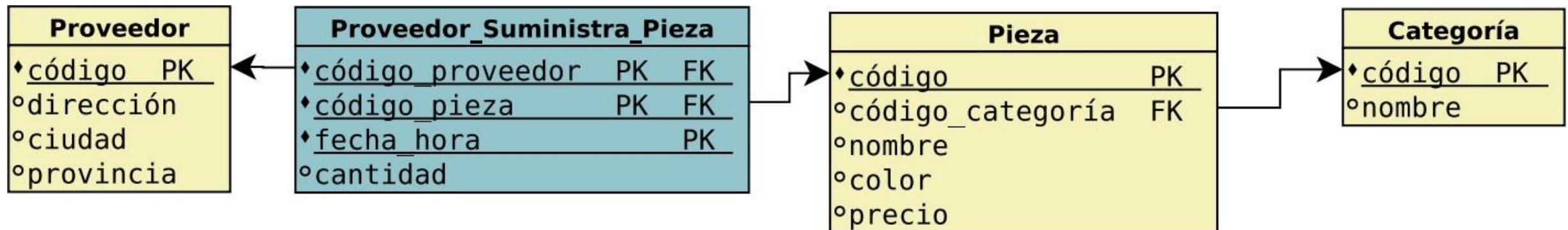
Otra forma de resolver este problema puede ser creando un nuevo atributo id que sea un valor numérico autoincrementado y que éste sea la única clave primaria de la tabla. La solución sería la siguiente:

PROVEEDOR\_SUMINISTRA\_PIEZA(**id**, código\_proveedor, código\_pieza, fecha\_hora, cantidad)

- código\_proveedor: FOREIGN KEY de PROVEEDOR(código)
- código\_pieza: FOREIGN KEY de PIEZA(código)

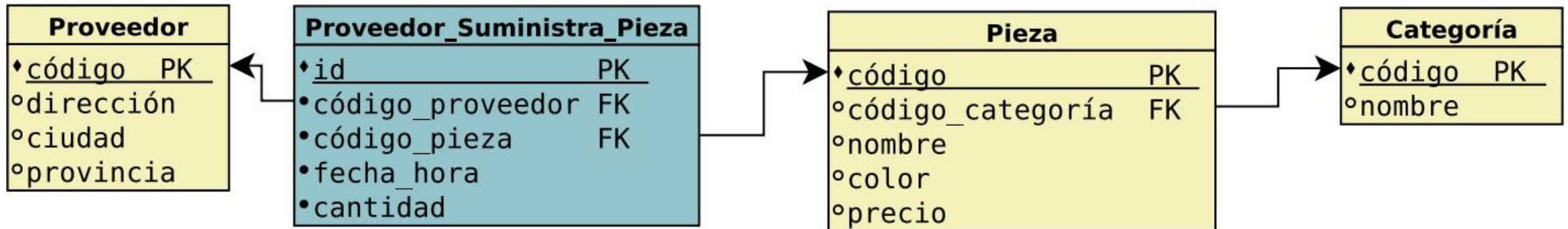
## Modelo Relacional

- PROVEEDOR(código, dirección, ciudad, provincia)
- CATEGORÍA(código, nombre)
- PIEZA(código, nombre, color, precio, código\_categoria)
  - código\_categoria: FOREIGN KEY de CATEGORÍA(código)
- PROVEEDOR\_SUMINISTRA\_PIEZA(código\_proveedor, código\_pieza, fecha\_hora, cantidad)
  - código\_proveedor: FOREIGN KEY de PROVEEDOR(código)
  - código\_pieza: FOREIGN KEY de PIEZA(código)





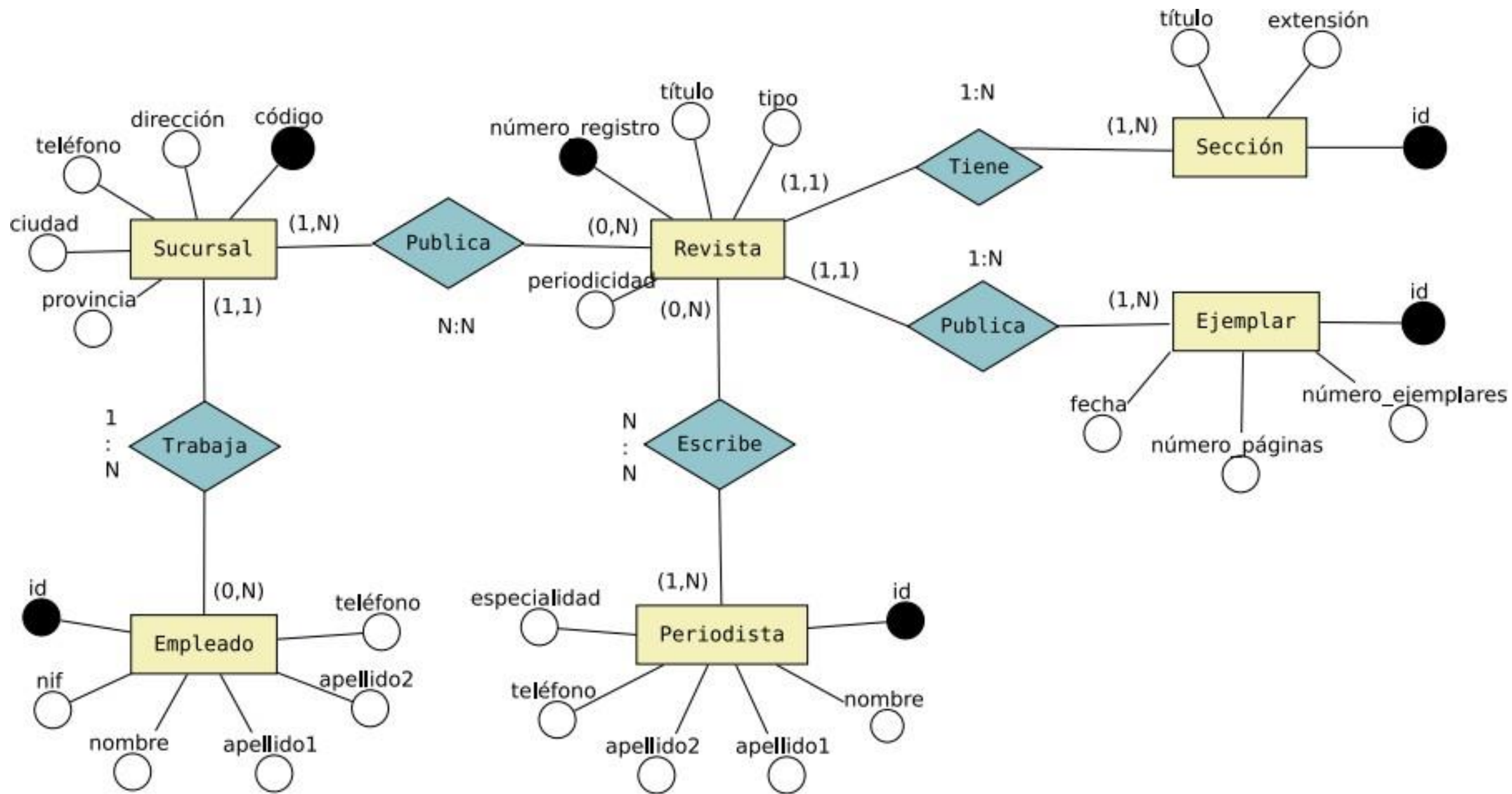
- PROVEEDOR(código, dirección, ciudad, provincia)
- CATEGORÍA(código, nombre)
- PIEZA(**código**, nombre, color, precio, código\_categoria)
  - código\_categoria: FOREIGN KEY de CATEGORÍA(código)
- PROVEEDOR\_SUMINISTRA\_PIEZA(**id**, código\_proveedor, código\_pieza, fecha\_hora, cantidad)
  - código\_proveedor: FOREIGN KEY de PROVEEDOR(código)
  - código\_pieza: FOREIGN KEY de PIEZA(código)



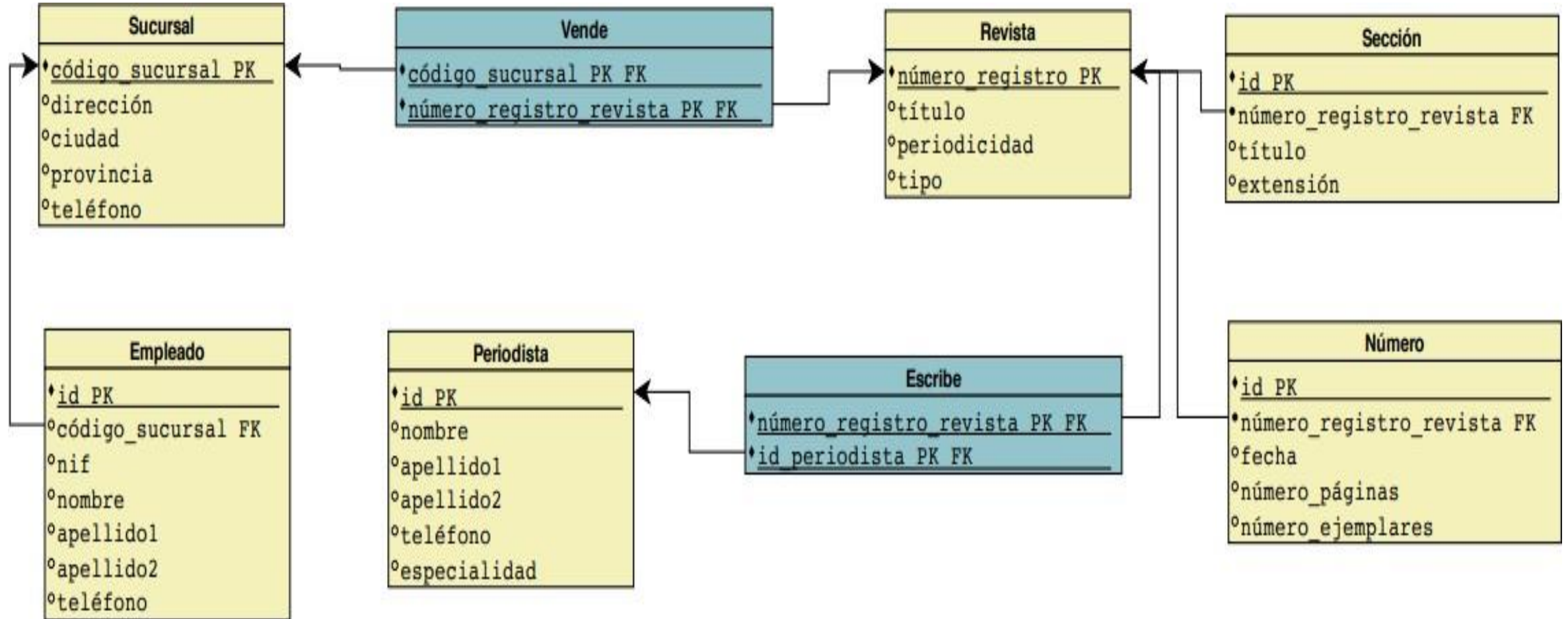
## **Ejemplo 2**

Tenemos esta información sobre una cadena editorial:

- La editorial tiene varias sucursales, con su domicilio, teléfono y un código de sucursal.
- Cada sucursal tiene varios empleados, de los cuales tendremos su nombre, apellidos, NIF y teléfono. Un empleado trabaja en una única sucursal.
- En cada sucursal se publican varias revistas, de las que almacenaremos su título, número de registro, periodicidad y tipo.
- Una revista puede ser publicada por varias sucursales.
- La editorial tiene periodistas (que no trabajan en las sucursales) que pueden escribir artículos para varias revistas. Almacenaremos los mismos datos que para los empleados, añadiendo su especialidad.
- También es necesario guardar las secciones fijas que tiene cada revista, que constan de un título y una extensión.
- Para cada revista, almacenaremos información de cada ejemplar, que incluirá la fecha, número de páginas y el número de ejemplares vendidos.



- SUCURSAL(código, dirección, teléfono, ciudad, provincia)
- EMPLEADO(id, nif, nombre, apellido1, apellido2, teléfono, código\_sucursal)
  - código\_sucursal: FOREIGN KEY de SUCURSAL(código)
- REVISTA(número\_registro, título, tipo, periodicidad)
- SUCURSAL\_PUBLICA\_REVISTA(código\_sucursal, número\_registro\_revista)
  - código\_sucursal: FOREIGN KEY de SUCURSAL(código)
  - número\_registro\_revista: FOREIGN KEY de REVISTA(número\_registro)
- PERIODISTA(id, nombre, apellido1, apellido2, teléfono, especialidad)
- PERIODISTA\_ESCRIBE\_REVISTA(id\_periodista, número\_registro\_revista)
  - id\_periodista: FOREIGN KEY de PERIODISTA(id)
  - número\_registro\_revista: FOREIGN KEY de REVISTA(número\_registro)
- SECCIÓN(id, título, extensión, número\_registro\_revista)
  - número\_registro\_revista: FOREIGN KEY de REVISTA(número\_registro)
- EJEMPLAR(id, número\_ejemplares, número\_páginas, fecha, número\_registro\_revista)
  - número\_registro\_revista: FOREIGN KEY de REVISTA(número\_registro)



# BASES DE DATOS

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos(DBMS).En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.

	<b>Representación lógica</b>	<b>Representación física</b>	<b>Modelo relacional</b>
	Tabla	Archivo secuencial	Relación
	Fila	Registro	Tupla
	Columna	Campo	Atributo

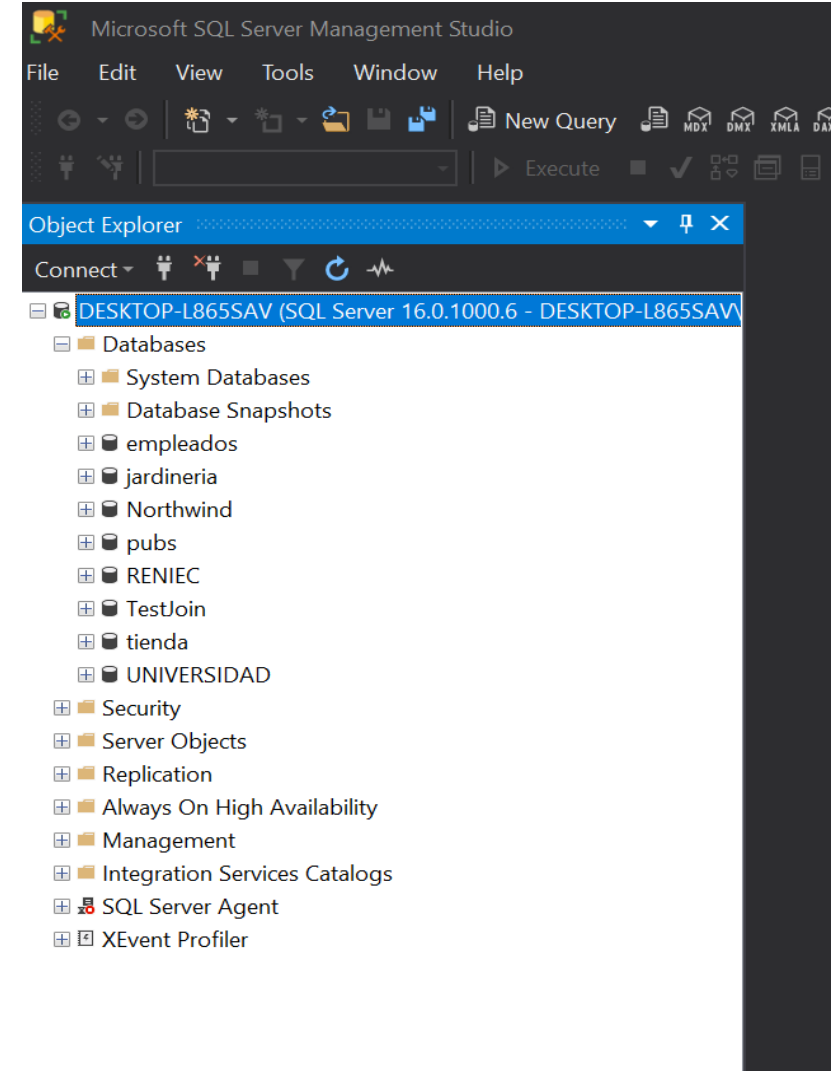


# Sistema de administración de bases de datos (DBMS)

- Un sistema de administración de bases de datos(DBMS)es un software de sistema para crear y administrar bases de datos.
- El DBMS proporciona a los usuarios y programadores una forma sistemática de crear, recuperar, actualizar y administrar datos.
- Un DBMS también permite a los usuarios finales crear, leer, actualizar y eliminar datos en una base de datos.
- El DBMS esencialmente sirve como una interfaz entre la base de datos y los usuarios finales o programas de aplicación, asegurando que los datos estén organizados de manera consistente y permanezcan fácilmente accesibles.
- El DBMS gestiona tres cosas importantes: los datos, el motor de la base de datos que permite acceder a los datos, bloquearlos y modificarlos, y el esquema de la base de datos, que define la estructura lógica de la base de datos. Estos tres elementos fundamentales ayudan a proporcionar concurrencia, seguridad, integridad de datos y procedimientos de administración uniformes.

# SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) es un entorno integrado para administrar cualquier infraestructura SQL. Utilice SSMS para acceder, configurar, gestionar, administrar y desarrollar todos los componentes de SQL Server, Azure SQL Database y Azure Synapse Analytics. SSMS proporciona una única utilidad integral que combina un amplio grupo de herramientas gráficas con varios editores de scripts enriquecidos para brindar acceso a SQL Server para desarrolladores y administradores de bases de datos de todos los niveles.





En cada instalación de SQL Server hay 4 bases de datos de sistema, y la capacidad de crear nuevas bases de datos por el usuario, en los cuales los datos están almacenados en tablas.



Las bases de datos del sistema:

- **master** - Registra toda la información del sistema para una instancia de SQL Server.
- **msdb** - La utiliza el Agente SQL Server para programar alertas y trabajos.
- **model** - Se utiliza como plantilla para todas las bases de datos creadas en la instancia de SQL Server. Las modificaciones hechas a la base de datos model, como el tamaño de la base de datos, la intercalación, el modelo de recuperación y otras opciones de base de datos, se aplicarán a las bases de datos que se creen con posterioridad.
- **tempdb** - Área de trabajo que contiene objetos temporales o conjuntos de resultados intermedios.

  System Databases

  master

  model

  msdb

  tempdb

---

# SQL

SQL es un acrónimo en inglés para Structured Query Language. Un Lenguaje de Consulta Estructurado. Un tipo de lenguaje de programación que te permite manipular y descargar datos de una base de datos. Tiene capacidad de hacer cálculos avanzados y álgebra. El SQL es un lenguaje universal que se emplea en cualquier sistema gestor de bases de datos relacional. Ha sido y sigue siendo el lenguaje de programación más usado para bases de datos relacionales.

- SQL consiste en un lenguaje de definición de datos (DDL), un lenguaje de manipulación de datos (DML) y un lenguaje de control de datos (DCL).
- El alcance de SQL incluye la inserción de datos, consultas, actualizaciones y borrado, la creación y modificación de esquemas y el control de acceso a los datos.
- También el SQL a veces se describe como un lenguaje declarativo, también incluye elementos procesales.
- Tiene un estándar definido, a partir del cual cada sistema gestor ha desarrollado su versión propia.

# TRANSACT-SQL

El SQL es un estándar definido, a partir del cual cada sistema gestor ha desarrollado su versión propia. T-SQL es fundamental para trabajar con servicios y productos de Microsoft SQL. Todas las herramientas y aplicaciones que se comunican con una base de datos SQL lo hacen enviando comandos T-SQL.

El Transact-SQL permite:

- Definir bloques de instrucciones SQL que se tratan como unidades de ejecución.
- Realizar ejecuciones condicionales.
- Realizar ejecuciones iterativas o repetitivas.
- Garantizar el tratamiento modular con la declaración de variables locales y el uso de procedimientos almacenados.
- Manipular tupla a tupla el resultado de una consulta.

Sin embargo, no permite:

- Crear interfaces de usuario.
- Crear aplicaciones ejecutables, sino elementos que en algún momento

llegarán al servidor de datos y serán ejecutados

Debido a estas restricciones se emplea generalmente para crear procedimientos almacenados, triggers y funciones de usuario.

# TIPOS DE DATOS

Para cada columna en una tabla y a cada variable o parámetro, se define un tipo de datos que sean almacenados en él, entre ellos:

1. **Números:** Números enteros y no enteros en distintos tamaños, y en diferentes niveles de precisión; y auto incremento opcional.
2. **Textos:** Cadenas de distintas longitudes, y distintas capacidades de apoyar distintas lenguas.
3. **Fechas:** Fechas en distintos niveles de precisión, desde días completos hasta fracciones menores de un segundo, que apoyan fechas a partir del principio del siglo XX o del calendario gregoriano, y la capacidad de diferenciar entre distintos usos de horarios.
4. **XML:** Datos textuales (cadenas) que representan conjuntos estándares de datos (estándar SGML).
5. **Datos binarios:** Datos almacenados como datos binarios (bits y bytes), que posibilitan el almacenamiento de archivos gráficos, etc.
6. **Geography:** Representación estándar de información geográfica, tales como estados, zonas geográficas, localidades; y las cálculos como distancias.
7. **Geometry:** Representación estándar de puntas, líneas, superficies en el plano; y las relaciones entre ellas.
8. **Hierarchid:** Representación estándar de información jerárquica como lista de materiales, relaciones de subordinación entre empleados, etc.



# **LENGUAJE DE DEFINICIÓN DE DATOS** **(DDL)**

El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Estos eventos corresponden principalmente a las instrucciones de Transact-SQL que comienzan por las palabras clave CREATE, ALTER, DROP. Algunos procedimientos almacenados del sistema que ejecutan operaciones de tipo DDL también pueden activar desencadenadores DDL.

- **CREATE (Crear):** Este comando permite crear objetos de datos, como nuevas bases de datos, tablas, vistas y procedimientos almacenados.
- **ALTER (Alterar):** Este comando permite modificar la estructura de una tabla u objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc.
- **DROP (Eliminar):** Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

# **LENGUAJE DE MANIPULACIÓN DE DATOS** **(DML)**

Lenguaje de Manipulación de Datos (Data Manipulation Language, DML) es un lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o modificación de los datos contenidos en las Bases de Datos del Sistema Gestor de Bases de Datos.

Elementos del lenguaje de manipulación de datos:

- **SELECT:** Permite recuperar datos de una o más tablas, en otras palabras, nos permite hacer consultas de los registros de las tablas.
- **INSERT:** Permite añadir o insertar registros a una tabla creada previamente, su complemento es Insert into.
- **UPDATE:** Permite modificar los datos de un conjunto de registros existentes en una tabla.
- **DELETE:** Permite eliminar o borrar todo o una parte de los datos de la tabla indicada por el argumento especificado después de la palabra clave FROM.
- **TRUNCATE:** Permite para borrar todos los registros de una tabla, pero no la tabla, es decir que quedaría una tabla vacía.

# LENGUAJE DE CONTROL DE DATOS (DCL)

- Un Lenguaje de Control de Datos (DCL - Data Control Language) es un lenguaje proporcionado por el sistema de gestión de base de datos que brinda la capacidad de manipular y adaptar los privilegios de un usuario final para hacer cumplir la seguridad y mantener una jerarquía de control sobre la información en la base de datos.

Algunos ejemplos de comandos incluidos en el DCL son los siguientes:

- **GRANT:** Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
- **REVOKE:** Permite eliminar permisos que previamente se han concedido con GRANT.
- **DENY:** Niega un permiso a un director.