TEMARIO

CONCEPTOS BÁSICOS

ACTORES DE LAS BASES DE DATOS

MODELO ENTIDAD – RELACIÓN

EJEMPLOS RESUELTOS DEL MODELO ENTIDAD – RELACIÓN

MODELO RELACIONAL

EJEMPLOS RESUELTOS DEL MODELO RELACIONAL

Introducción

Hoy en día la gestión de bases de datos ha evolucionado, siendo un tema importante para su estudio en diversas áreas de nuestro entorno, ya que hoy en día el uso de las bases de datos no solo se enfoca en resolver situaciones de índole informático, sino que se ocupa en muchas áreas como herramienta para el control y administración de la información; lo anterior expuesto se ha debido gracias a la expansión de internet y el desarrollo de tecnologías ha incrementado su uso en diversas aplicaciones.

El uso de las bases de datos ha dado origen a tener un medio general estandarizado para la administración de la información.

# BASE DE DATOS

## Importancia de las bases de datos y los sistemas de información

Una base de datos, bien diseñada y alimentada, no es más que una herramienta de consulta pura y dura. Este es el verdadero valor añadido que le da a una empresa, ya que le permite combinar cualquier información con todo tipo de consultas que se le formulen. Disponer de una buena base de datos significa controlar la amplia gama de información con la que trabaja cualquier tipo de negocio hoy en día. La franquicia no es una excepción.

Una base de datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo. La base de datos es una colección almacenada de datos interrelacionados que representa la información operacional de una organización. Sin redundancias perjudiciales o innecesarias. Su finalidad es la de servir a una aplicación ó más de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados.

## 1.2 Conceptos básicos

¿Qué es un dato?

* Un dato es la representación de una variable que puede ser cuantitativa o cualitativa que indica un valor que se le asigna a las cosas y se representa a través de una secuencia de símbolos, números o letras.
* Los datos son la representación simbólica, bien sea mediante números o letras de una recopilación de información la cual puede ser cualitativa o cuantitativa, que facilitan la deducción de una investigación o un hecho.

¿Qué es la información?

¿Qué es una organización?

Puede ser una empresa de cualquier tipo bancos, hospitales, universidades, etc

¿Qué es una base de datos?

Una base de datos es un conjunto de elementos de datos interrelacionados, administrados como unidad. Esta definición es deliberadamente amplia porque existe mucha variación entre los diferentes vendedores de software que ofrecen sistemas de bases de datos. Por ejemplo, Microsoft Access pone toda la base de datos en un solo archivo, de modo que una base de datos de Access puede definirse como el archivo que contiene los elementos de datos. Oracle Corporation define su base de datos como un conjunto de archivos físicos administrados por una instancia de su producto de software de base de datos. Una instancia es una copia del software de base de datos que se ejecuta en la memoria. Microsoft SQL Server y Sybase Adaptive Server Enterprise (ASE) definen una base de datos como un conjunto de elementos de datos que tienen un propietario común, y varias bases de datos suelen ser administradas por una sola instancia del software. Todo esto puede ser muy confuso si trabaja con varios productos; por ejemplo, una base de datos que se apega a la definición de Microsoft SQL Server o Sybase ASE es exactamente lo que Oracle Corporation llama un esquema.

Un objeto de base de datos es una estructura de datos con nombre que se guarda en una base de datos. Los tipos específicos de objetos permitidos varían de un vendedor a otro, y de un modelo a otro. El modelo de base de datos alude al modo en que una base de datos organiza su contenido para representar el mundo real.

Un archivo es un conjunto de registros relacionados que un sistema operativo guarda como unidad.

## 1.3 Ventajas de las bases de datos.

1. Independencia de datos y tratamiento.

* Cambio en datos no implica cambio en programas y viceversa (eso reduce los costos de mantenimiento).

1. Coherencia de resultados.

* Reduce redundancia:
  + Acciones lógicamente únicas.
  + Se evita inconsistencia.

1. Mejora en la disponibilidad de datos

* No hay dueño de datos (No igual a ser públicos).
* Ni aplicaciones ni usuarios.
* Guardamos descripción (Idea de catálogos).

4.Cumplimiento de ciertas normas.

* Restricciones de seguridad.
* Accesos (Usuarios a datos).
* Operaciones (Operaciones sobre datos).

5.Otras ventajas:

* Más eficiente gestión de almacenamiento

# El sistema de administración de bases de datos

El sistema de administración de bases de datos (DBMS) es el software proporcionado por el vendedor de la base de datos.

El sistema manejador de bases de datos es la porción más importante del software de un sistema de base de datos. Un DBMS es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

Existen muchos sistemas de gestión o manejadores de base de datos, existen muchos como:

* MySQL
* PosgreSQL
* Microsoft SQL Server
* Oracle
* Microsoft Access
* Microsoft Visual Fox Pro
* Firebird
* mSQL (mini SQL)
* IBM DB2
* IBM Informix
* SQLite
* Sybase ASE
* Paradox
* dBase

Claro está, que existen algunas ventajas y desventajas que los hace diferentes en la gestión de la base de datos, dichas diferencias son importantes para las grandes organizaciones y empresas pequeñas al momento de elegir el de mayor beneficio, confiabilidad y seguridad en la administración de la información que manejen.

El DBMS ofrece todos los servicios básicos requeridos para organizar y conservar una base de datos, entre ellos:

* Mover los datos de archivos de datos físicos, según sea necesario.
* Administrar la opción de que varios usuarios consulten datos de manera concurrente, e incluir medidas que eviten que las actualizaciones simultáneas tengan conflictos entre sí.
* Controlar las transacciones para que los cambios en la base de datos de cada transacción sean una unidad de trabajo tipo todo o nada. En otras palabras, si la transacción tiene éxito, todos los cambios se registran en la base de datos; si la transacción fracasa, ninguno de los cambios se registra.
* Permitir un lenguaje de consulta, que es un sistema de comandos empleado por el usuario de la base de datos para recuperar sus datos.
* Proporcionar medidas para respaldar la base de datos y recuperarla después de una falla.
* Aportar mecanismos de seguridad para evitar la consulta y modificación no autorizadas de los datos.

# Aplicaciones de los SGBD

Las bases de datos son ampliamente usadas en diversas áreas y sectores de la sociedad. Las siguientes son algunas de sus aplicaciones más representativas:

* Bancos. Para el almacenaje de información de los clientes, cuentas y préstamos, y sobre todo en la realización de transacciones bancarias.
* Líneas aéreas. Para poder efectuar reservas e información de planificación de vuelos. Las líneas aéreas fueron de los primeros en usar las bases de datos de forma distribuida geográficamente (las terminales aéreas situadas en todo el mundo accedían al sistema de bases de datos centralizado a través de las líneas telefónicas y otras redes de datos).
* Universidades. Para el almacenaje y administración de información de los estudiantes, docentes, matrículas de las asignaturas, créditos de las asignaturas, cursos, calificaciones.
* Transacciones de tarjetas de crédito. Para compras con tarjeta de crédito y generación mensual de estados de cuenta.
* Telecomunicaciones. Para guardar un registro de las llamadas realizadas, emisión mensual de facturas, para almacenar información sobre las redes de comunicaciones.
* Finanzas. Para almacenar información sobre grandes empresas, ventas y compras de índole financiero.
* Ventas. Para información de clientes, productos y compras.
* Producción. Para la gestión de la cadena de producción y para el seguimiento de la producción de elementos en las factorías, inventarios de elementos en almacenes y pedidos de elementos.
* Recursos humanos. Para la correcta administración de la información sobre los empleados, salarios, impuestos y beneficios, y para la generación de las nóminas en las instituciones o empresas.

## Objetivos fundamentales de los SGBD.

Independencia de los datos y los programas de aplicación.

Como se ha descrito anteriormente, con el uso de ficheros tradicionales la lógica de la aplicación contempla la organización de los ficheros y el método de acceso. Por ejemplo, si por razones de eficiencia se utiliza un fichero secuencial indexado, el programa de aplicaciones debe considerar la existencia de los índices y la secuencia del fichero. Entonces es imposible modificar la estructura de almacenamiento o la estrategia de acceso sin afectar el programa de aplicación (naturalmente, lo que se afecta en el programa son las partes de éste que tratan los ficheros, lo que es ajeno al problema real que el programa de aplicación necesita resolver). En un SBD sería indeseable la existencia de aplicaciones y datos dependientes entre sí, por dos razones fundamentales:

* *Diferentes aplicaciones necesitarán diferentes aspectos de los mismos datos (decimal o binario).*
* *Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios en el fenómeno o proceso de la realidad sin necesidad de modificar los programas de aplicación (también para buscar mayor eficiencia).*

Independencia de los datos: es la inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso y constituye el objetivo fundamental de los SBD.

## Modelos prevalecientes de base de datos

En esencia, un modelo de bases de datos es la arquitectura que utiliza el DBMS para guardar los objetos dentro de la base de datos y relacionarlos entre sí. Aquí se presentan los modelos más frecuentes en orden de evolución.

**Archivos simples**

Los archivos simples son archivos comunes de un sistema operativo; en ellos, los registros de un archivo no contienen información para comunicar su estructura, o cualquier relación entre los registros, a las aplicaciones que utiliza el archivo. Cualquier información acerca de la estructura o el significado de los datos en el archivo deben incluirse en cada aplicación que lo utilice o debe conocerla cada persona que lo lee. En esencia, los archivos simples no son bases de datos, porque no cumplen ninguno de los criterios analizados antes. No obstante, es importante que los comprenda por dos razones. En primer lugar, suelen ser utilizados para Guardar información de bases de datos. En este caso, el sistema operativo todavía desconoce el contenido y la estructura de los archivos, pero el DBMS tiene metadatos que le permiten convertir los archivos simples de la capa física en las estructuras de base de datos de la capa lógica.

En la Ilustración 1 se muestra un sistema de archivo simple, un subconjunto de los datos de la compañía ficticia Northwind Traders, proveedor internacional de artículos alimenticios (y la base de datos de ejemplo de Microsoft). Recuerde que los títulos de las columnas (Id de cliente, Nombre de compañía, etc.) sólo se incluyen como ejemplo; únicamente los registros de datos se guardarían en los archivos reales. Los datos del cliente se guardan en un archivo Clientes, y cada registro representa un cliente de Northwind. Cada empleado de Northwind tiene un registro en el archivo Empleados y cada producto vendido por Northwind tiene un registro en el archivo Productos. Los datos de un pedido (los pedidos hechos a Northwind por sus clientes) se guardan en otros dos archivos simples. El archivo Pedidos contiene un registro para cada pedido de cliente con los datos correspondientes, como la identificación del cliente que hizo el pedido y el nombre del empleado que aceptó el pedido del cliente. El archivo Detalles de pedido contiene un registro de cada artículo de un pedido (un pedido puede contener varios artículos de línea, uno para cada producto solicitado) incluidos datos como precio unitario y cantidad.

Un programa de aplicación es una unidad de lógica del programa computacional que efectúa una función específica dentro de un sistema de aplicación. Northwind tiene un programa de aplicación que imprime una lista de todos los pedidos. Esta aplicación debe correlacionar los datos entre los cinco archivos, al leer un pedido y realizar los pasos siguientes:

1. *Emplear Id de cliente para encontrar el nombre del cliente en el archivo Clientes.*
2. *Usar Id de cliente para hallar el nombre del empleado relacionado en el archivo Empleados.*
3. *Utilizar Id de pedido para ubicar los artículos de línea correspondientes en el archivo Detalles de pedido.*
4. *Para cada artículo de línea, aplicar Id de producto con el fin de hallar el nombre de producto correspondiente en el archivo Productos.*

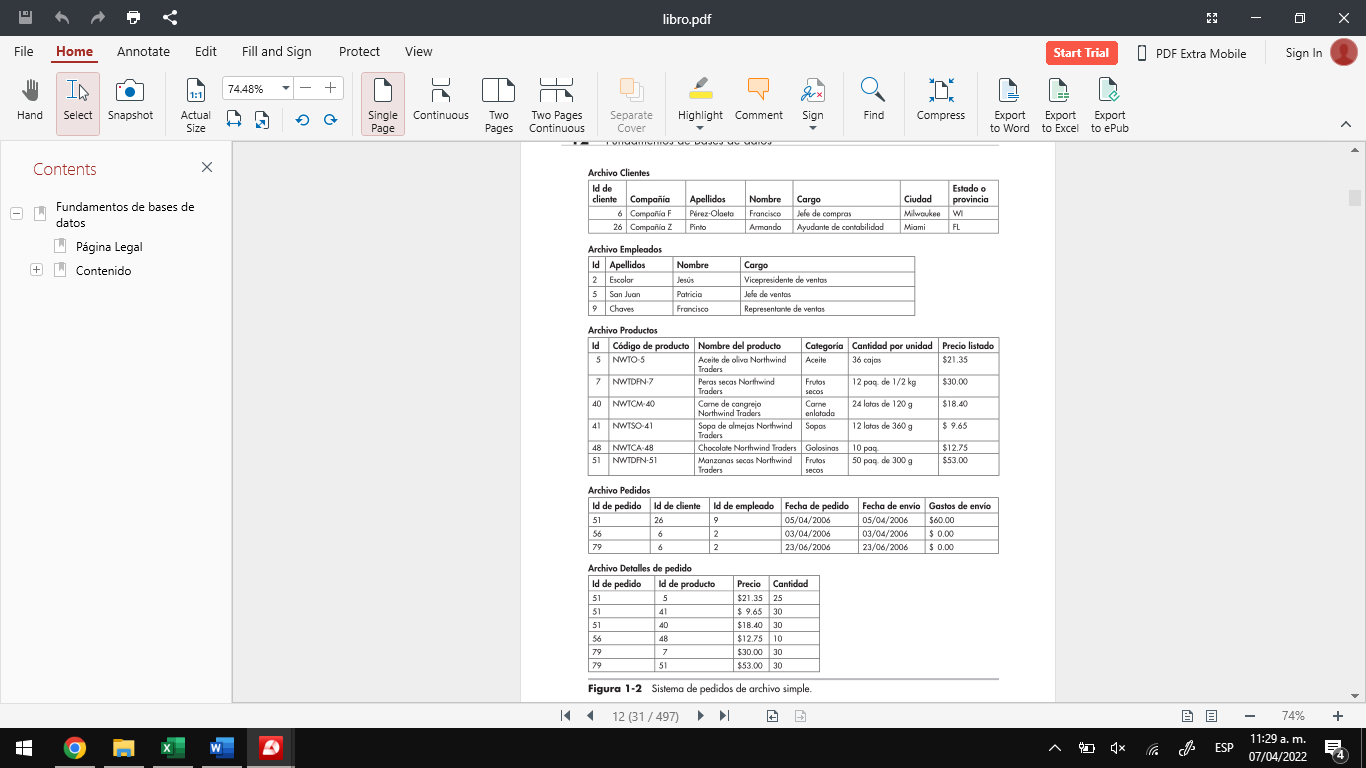


Ilustración . Ejemplo de archivos simple

**El modelo jerárquico**

Las primeras bases de datos seguían el modelo jerárquico, que evolucionó a partir de los sistemas de archivos que reemplazaron las bases de datos, con los registros ordenados en una jerarquía similar a un organigrama. Cada archivo del sistema de archivo simple se convirtió en un tipo de registro, o nodo en terminología jerárquica, pero aquí se utiliza el término registro por simplicidad. Los registros se conectaban mediante apuntadores que contenían la dirección

del registro relacionado. Los apuntadores indicaban a la computadora dónde se ubicaba físicamente el registro relacionado; igual que la dirección de una calle lo dirige a un edificio específico en una ciudad, un URL lo dirige a una página Web específica en Internet, o las coordenadas de un GPS apuntan a una ubicación determinada en el planeta. Cada apuntador establece una relación primario-secundario, también denominada relación uno a varios, en que un elemento principal puede tener muchos elementos secundarios, pero cada uno de éstos sólo puede tener un elemento primario. Esto es similar a la situación en una organización de negocios tradicional, donde cada gerente puede tener muchos empleados bajo su mando, pero cada empleado sólo tiene un gerente. El problema obvio del modelo jerárquico es que algunos datos no se ajustan exactamente a esta estructura jerárquica estricta, como cuando un pedido debe tener como elemento primario al cliente que lo generó y como otro elemento primario al empleado que capturó el pedido. La base de datos jerárquica más popular fue el Information Management System (IMS) de IBM.

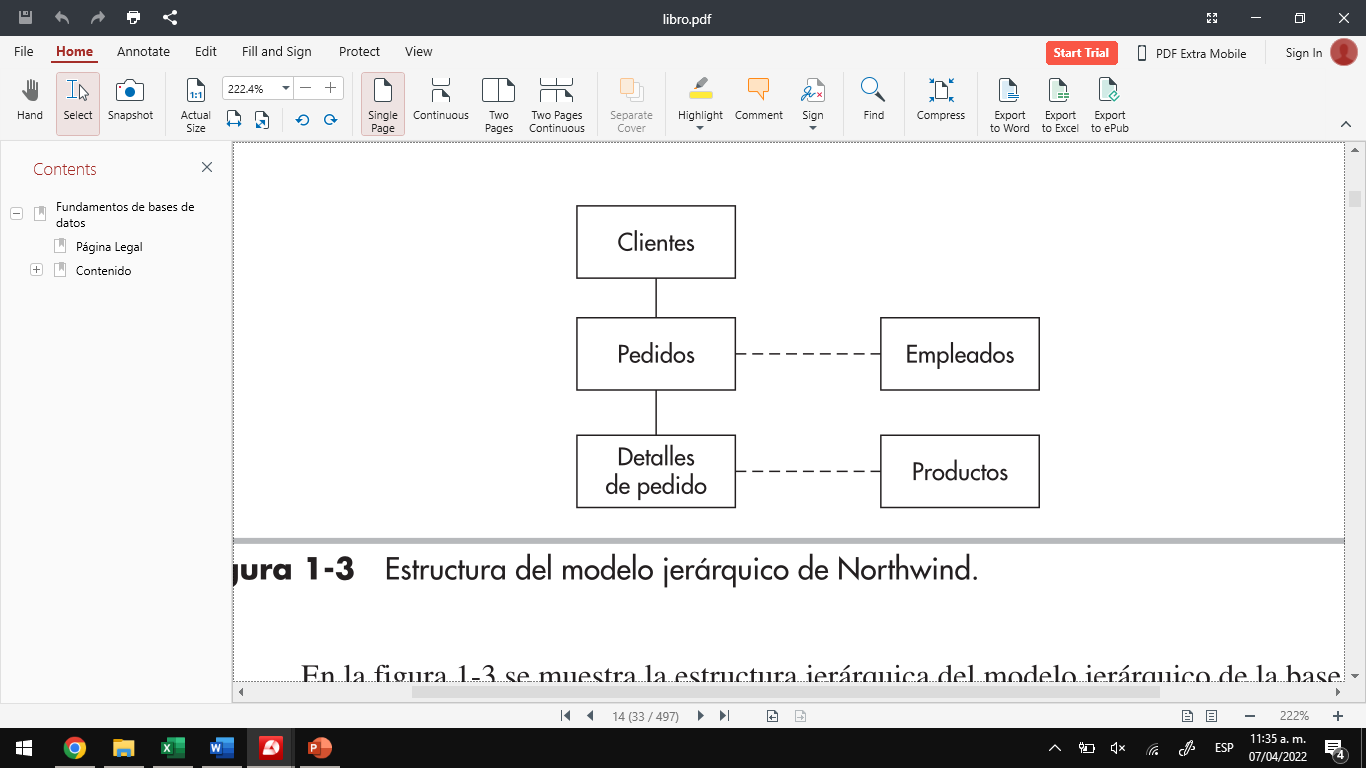


Ilustración Modelo Jerarquico de Northwind

**El modelo de red**

El modelo de base de datos de red evolucionó casi al mismo tiempo que el modelo jerárquico.

En esencia, se formó un comité de representantes de la industria para crear un modelo mejorado. Un cínico diría que un camello es un caballo diseñado por un comité, y sería correcto decir eso en este caso. El modelo de base de datos más popular basado en el modelo de red fue el Integrated Database Management System (IDMS), desarrollado en un principio por Cullinane (que cambió su nombre a Cullinet). El producto fue mejorado con extensiones relacionales, se denominó IDMS/R y terminó por ser vendido a Computer Associates.

Al igual que con el modelo jerárquico, los tipos de registros (o sencillamente los registros) representan lo que serían archivos separados en un sistema de archivo simple, y esos registros se enlazan mediante relaciones uno a varios, también llamadas relaciones propietario-miembro o conjuntos, en la terminología del modelo de red. Nos apegaremos una vez más a los términos elemento primario y elemento secundario, para simplificar la exposición. Al igual que con el modelo jerárquico, se utilizan apuntadores a la dirección física para conectar los registros relacionados, y se elimina cualquier identificación de los registros principales a partir de cada registro secundario, para evitar posibles inconsistencias. En contraste con el modelo jerárquico, se da nombre a las relaciones para que el programador pueda dar instrucciones al DBMS de que utilice una relación específica para desplazarse de un registro a otro en la base de datos, permitiendo que un tipo de registro participe como elemento secundario de varias relaciones. El modelo de red aportó mayor flexibilidad, pero (como suele ocurrir con las computadoras) a costa de pérdida de la simplicidad.

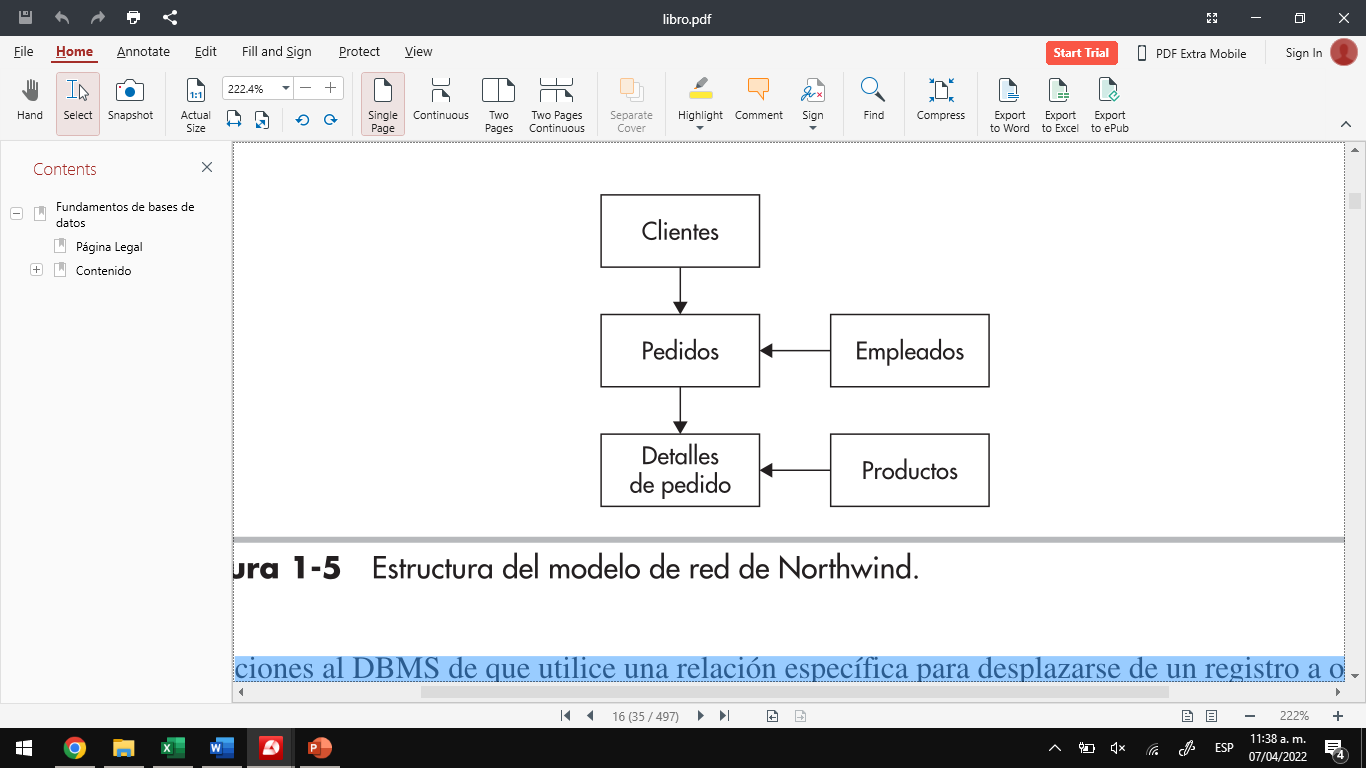


Ilustración Modelo de red de Northwind

**El modelo relacional**

Además de la complejidad, los modelos de base de datos de red y jerárquico comparten otro problema común: son inflexibles. Deben seguirse las rutas preconcebidas a través de los datos para procesarlos con eficiencia. Las consultas ad hoc, como hallar todos los pedidos enviados

en un mes específico, requieren que se explore toda la base de datos para localizarlos. Los científicos computacionales todavía buscan un mejor modo. Sólo unos cuantos elementos en la historia del desarrollo de las computadoras fueron verdaderamente revolucionarios, y el trabajo de investigación de E. F. (Ted) Codd que condujo al modelo relacional lo es.

El modelo relacional se basa en la noción de que cualquier ruta preconcebida a través de la estructura de datos es una solución demasiado restrictiva, sobre todo ante las cada vez mayores demandas para dar soporte a solicitudes ad hoc de información. Los usuarios de una base de datos no pueden pensar en cada uso posible de los datos antes de que se cree la base de datos; por lo tanto, la imposición de rutas predefinidas a través de los datos sólo genera una “prisión de datos”. El modelo relacional permite a los usuarios relacionar los registros según se requiera y no de manera predefinida, cuando se guardan los registros por primera vez en la base de datos. Además, el modelo relacional está creado de modo que las consultas funcionan con conjuntos de datos (por ejemplo, todos los clientes que tienen un saldo sobresaliente) en lugar de un registro a la vez, como ocurre con los modelos de red y jerárquico.

El modelo relacional presenta los datos en las familiares tablas bidimensionales, como lo hace una hoja de cálculo. Pero a diferencia de lo que sucede en ésta, no es necesario que los datos se guarden en forma tabular; además, el modelo también permite combinar (unir en la terminología relacional) tablas para formar vistas, que también se presentan como tablas bidimensionales. En resumen, sigue el modelo ANSI/SPARC y, por lo tanto, aporta dosis saludables de independencia física y lógica de los datos. En lugar de vincular los registros relacionados con apuntadores a una dirección física, como ocurre en los modelos jerárquico y de red, se guarda en cada tabla un elemento de datos común, como se hace en los sistemas de archivo simple.

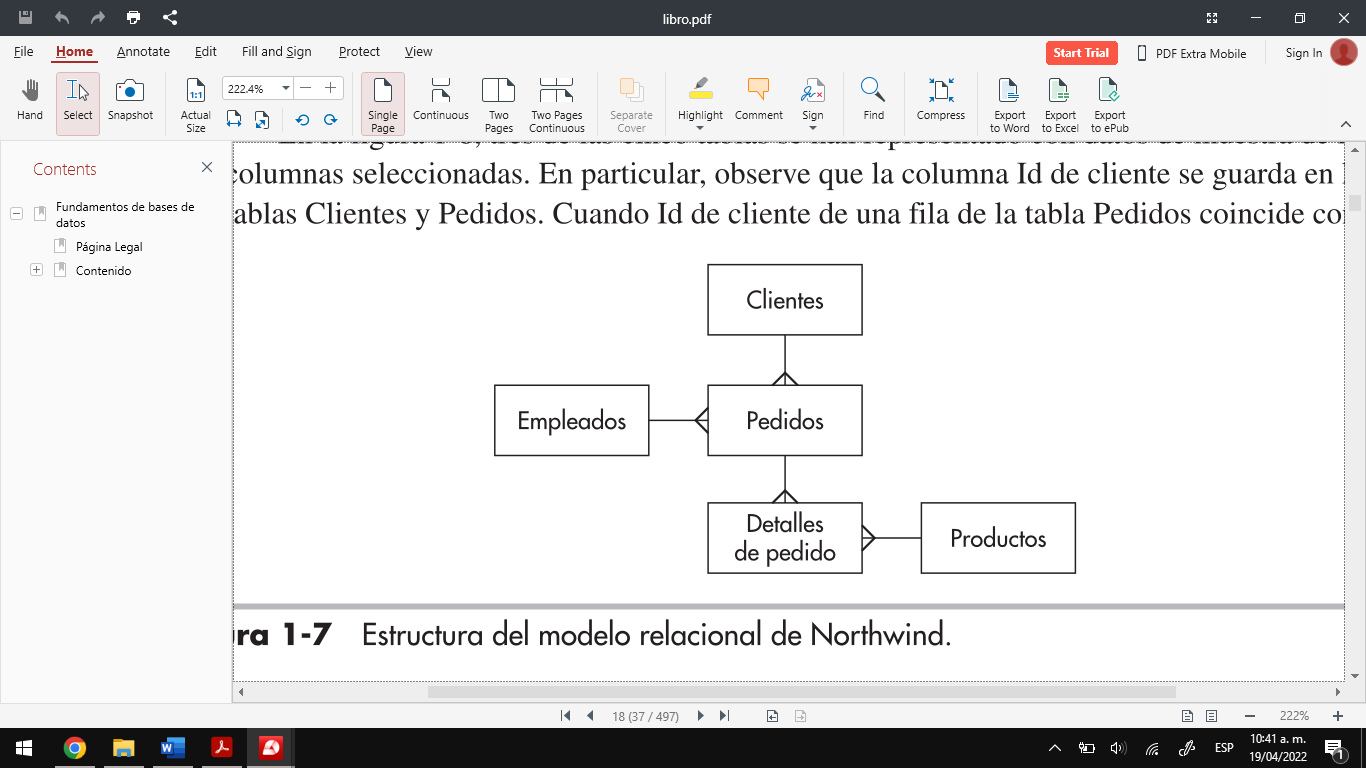


Ilustración 4. Estructura del modelo relacional de Northwind

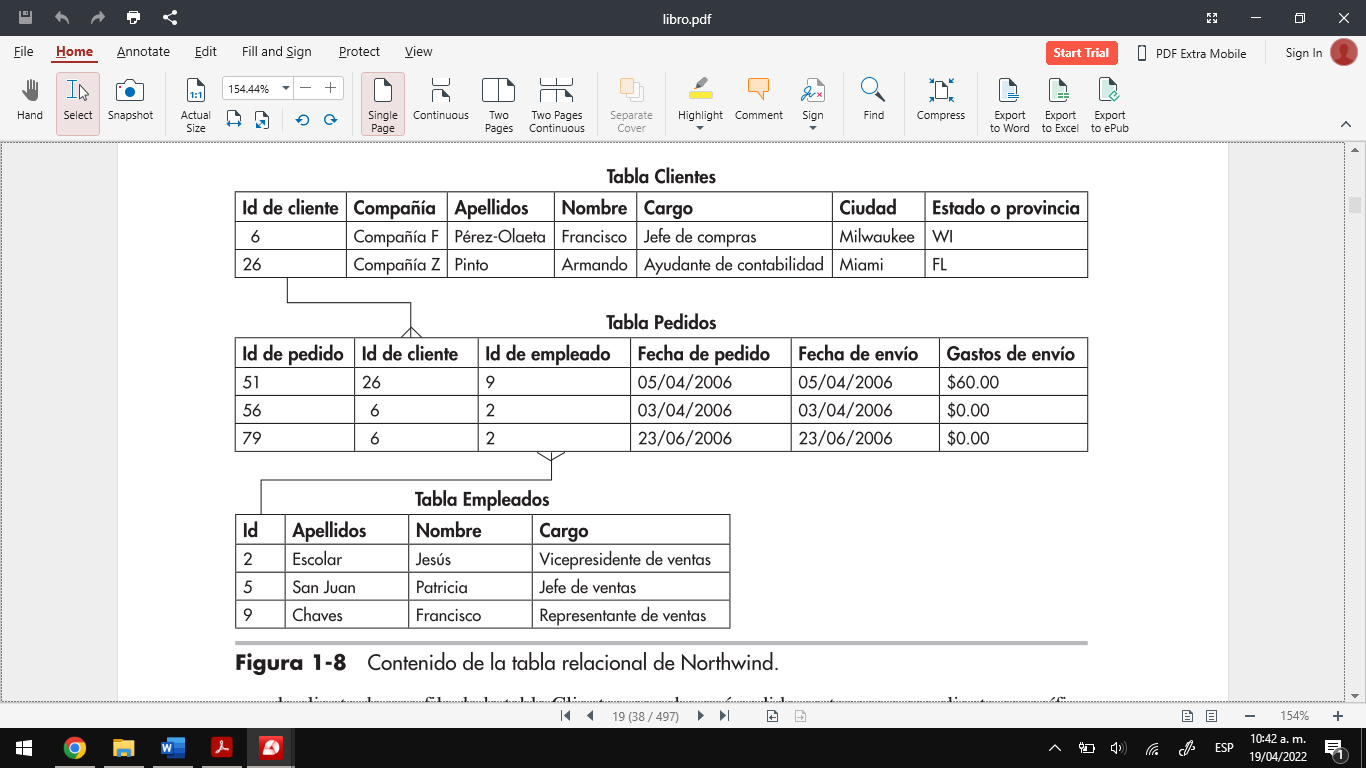


Ilustración 5. Contenido de la tabla relacional de Northwind

Tes de las cinco tablas se han representado (en la Ilustración 5) con datos de muestra de las columnas seleccionadas. En particular, observe que la columna Id de cliente se guarda en las tablas Clientes y Pedidos. Cuando Id de cliente de una fila de la tabla Pedidos coincide con Id de cliente de una fila de la tabla Clientes, se sabe qué pedido pertenece a ese cliente específico. Asimismo, la columna Id de empleado se guarda en las tablas Empleados y Pedidos, para indicar al empleado que aceptó cada pedido.

# Personas en el entorno de las bases de datos

Hay cuatro grupos de personas que intervienen en el entorno de una base de datos: el administrador de la base de datos, los diseñadores de la base de datos, los programadores de aplicaciones y los usuarios.

El *administrador de la base de datos* se encarga de la implementación física de la base de datos: escoge los tipos de los ficheros de datos y de los índices que deben crearse, determina dónde deben ubicarse ficheros e índices y, en general, toma las decisiones relativas al almacenamiento físico en función de las posibilidades que le ofrezca el SGBD con el que trabaje. Además, el administrador de la base de datos se encarga de establecer la política de seguridad y del acceso concurrente. También se debe preocupar de que el sistema se encuentre siempre operativo y procurar que los usuarios y las aplicaciones obtengan buenas prestaciones. El administrador debe conocer muy bien el SGBD con el que trabaja, así como el equipo informático sobre el que esté funcionando.

Los *diseñadores de la base de datos* realizan el diseño de la base de datos, debiendo identificar los datos, las relaciones entre ellos y las restricciones sobre los datos y sobre sus relaciones. El diseñador de la base de datos debe tener un profundo conocimiento de los datos de la empresa y también debe conocer sus reglas de negocio. Las reglas de negocio describen las características principales sobre el comportamiento de los datos tal y como las ve la empresa. Para obtener un buen resultado, el diseñador de la base de datos debe implicar en el proceso a todos los usuarios de la base de datos, tan pronto como sea posible.

Una vez se ha diseñado e implementado la base de datos, los *programadores de aplicaciones* se encargan de implementar los programas de aplicación que servirán a los usuarios finales. Estos programas de aplicación son los que permiten consultar datos, insertarlos, actualizarlos y eliminarlos. Estos programas se escriben mediante lenguajes de tercera generación o de cuarta generación.

Los *usuarios finales* son los clientes de la base de datos: la base de datos ha sido diseñada e implementada, y está siendo mantenida, para satisfacer sus requisitos en la gestión de su información.

# Modelo Conceptual de Sistemas de Bases de Datos

El diseño conceptual de una base de datos incluye el estudio y modelado de los datos de manera independiente de la tecnología. El modelo conceptual de los datos que se obtiene, en teoría, se puede implementar en cualquier base de datos o aun en un sistema de archivo simple. La persona que efectúa el diseño conceptual de una base de datos se denomina modelador de datos. El diseño lógico de una base de datos es el proceso de trasladar, o ubicar, el diseño conceptual en un diseño lógico que se ajuste al modelo de base de datos elegido (relacional, orientado a objetos, de objetos-relacional, etc.). A un especialista que desarrolla el diseño lógico de una base de datos se le conoce como diseñador de base de datos, pero el administrador de una base de datos (DBA, DataBase Administrator) realiza de manera total o parcial este paso del diseño. El paso final es el diseño físico de una base de datos, que requiere la ubicación del diseño lógico en uno o más diseños físicos, cada uno ajustado al DBMS específico que administrará la base de datos y el equipo de cómputo en particular en que funcionará la base de datos.

## El modelo Entidad Relación (MER)

El MER es un modelo de datos de alto nivel. Está basado en una percepción de un mundo real que consiste en una colección de objetos básicos, denominados entidades, y de relaciones entre estos objetos, así como las características de estos objetos llamados atributos.

Este modelo fue propuesto en 1976 y ha encontrado una amplia aceptación como instrumento para modelar el mundo real en el proceso de diseño de las bases de datos.

**Entidades y conjunto de entidades**

Una entidad es un objeto que existe y se distingue de otros objetos de acuerdo a sus características llamadas atributos. Las entidades pueden ser concretas como una persona o abstractas como una fecha. Por ejemplo, una silla es una entidad, un automóvil, un empleado, un profesor, un estudiante, que son cosas concretas; pero también puede ser algo no tangible, como un suceso cualquiera, una cuenta de ahorro, o un concepto abstracto.

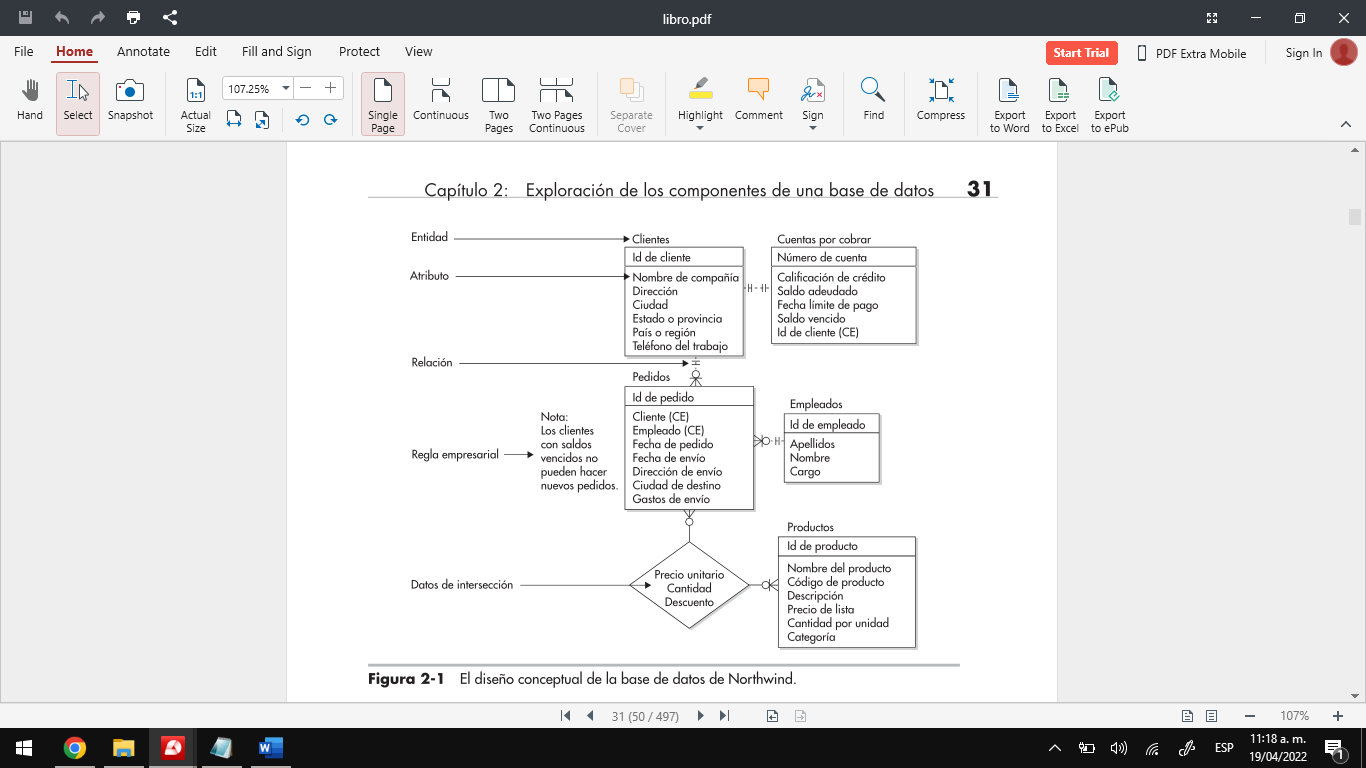


Ilustración 6. Diseño conceptual de la base de datos de Northwind

En otras palabras, las entidades son los objetos reales que nos interesan lo suficiente como para capturar y guardar sus datos en una base de datos. Una entidad se representa como un rectángulo en el diagrama. Cualquier cosa que se designe con un nombre puede ser una entidad. Sin embargo, para no incluir en el diseño de nuestra base de datos todo lo que hay en el planeta, nos limitamos a las entidades que interesan a las personas que utilizarán nuestra base de datos.

Cada entidad presentada en el modelo conceptual (ilustración 6) representa la clase completa de esa entidad. Por ejemplo, la entidad Clientes representa el conjunto de todos los clientes de Northwind. A los clientes individuales se les denomina instancias de la entidad.

Una entidad externa es la entidad con la que nuestra base de datos intercambia datos (le envía datos, los recibe de ella, o ambas cosas) pero sobre la que no se recopilan datos. Por ejemplo, casi todas las empresas que preparan cuentas de crédito para clientes adquieren informes de crédito de una o más oficinas de crédito. Envían la información de identificación del cliente a esa oficina y obtienen un informe de crédito, pero todos estos datos se refieren al cliente, no a la oficina. Al suponer que no existe una razón apremiante para que la base de datos guarde información acerca de la oficina de crédito, como su dirección, la oficina de crédito no aparecerá en el diseño conceptual de la base de datos como una entidad.

**Atributos**

Un atributo es un hecho unitario que caracteriza o describe de alguna manera a una entidad. En el diagrama de diseño conceptual presentado en la *Ilustración 6*, están incorporados como nombres dentro de un rectángulo que representa la entidad a la que pertenecen. Los atributos que aparecen en la parte superior del rectángulo (sobre la línea horizontal) forman el identificador único de la entidad. Como lo sugiere su nombre, un identificador único proporciona un valor único para cada instancia de la entidad. Por ejemplo, el atributo Id de cliente es el identificador único de la entidad Clientes, de modo que cada cliente debe tener un valor único en ese atributo. Recuerde que un identificador único puede tener varios atributos, pero cuando hay varios atributos, todavía se le considera un identificador único.

Se dice que los atributos son un hecho aislado porque deben ser indivisibles, lo que significa que no pueden dividirse en unidades más pequeñas que tengan algún significado. Por lo tanto, un atributo es la unidad de datos con nombre más pequeño que aparece en un sistema de base de datos. En este sentido, Dirección debe ser considerada un atributo sospechoso, porque fácilmente podría dividirse en Línea de dirección 1, Línea de dirección 2 y, tal vez, Línea de dirección 3, como suele ocurrir en los sistemas empresariales. Por ejemplo, este cambio tendría sentido porque facilita la impresión de etiquetas de direcciones. Por otra parte, el diseño de una base de datos no es una ciencia exacta y deben tomarse decisiones de juicio. Aunque es posible separar el atributo Teléfono del trabajo en sus atributos componentes, como Código de país, Código de área, Prefijo, Sufijo y Extensión, es necesario ponderar si ese cambio agrega sentido o valor. Aquí no existe una respuesta correcta o incorrecta, de modo que debemos confiar en la ayuda de las personas que emplearán la base de datos, o quienes financian el proyecto de base de datos, para tomar esas decisiones. Siempre recuerde que un atributo debe describir o caracterizar la entidad de alguna manera (por ejemplo, tamaño, forma, color, cantidad, ubicación).

## Diagrama Entidad Relación (DER)

Constituye la representación gráfica asociada al MER.

En un DER, cada entidad se representa mediante un rectángulo, cada relación mediante un rombo y cada dominio mediante un círculo. Mediante líneas se conectan las entidades con las relaciones, igual que las entidades con los dominios, representando a los atributos.

**Relaciones**

Las relaciones son las asociaciones entre las entidades. Como las bases de datos se concentran en guardar datos relacionados, las relaciones se vuelven el pegamento que mantiene unida la base de datos. Las relaciones se muestran en el diagrama de diseño conceptual (*Ilustración 6*) como líneas que conectan a una o más entidades. Cada extremo de una línea de relación muestra la cardinalidad máxima de la relación, que es la cantidad máxima de instancias de una entidad que se puede asociar con la entidad en el extremo opuesto de la línea.

**Cardinalidad**

La cardinalidad máxima puede ser una (la línea sin ningún símbolo especial en su extremo) o varias (la línea con una pata de gallo en el extremo). Justo antes del extremo de la línea está otro símbolo que indica la cardinalidad mínima, que es la cantidad mínima de instancias de una entidad que se pueden asociar con la entidad en el extremo opuesto de la línea. La cardinalidad mínima puede ser cero, que se representa con un círculo dibujado sobre la línea, o uno, que se señala con una breve línea perpendicular dibujada sobre la línea de la relación. Muchos modeladores de datos emplean dos líneas perpendiculares para indicar “una y sólo una”, igual que en la *Ilustración 6.*

Se requiere práctica para aprender a leer las relaciones, y definirlas y dibujarlas correctamente requiere mucha más práctica. El truco está en considerar la asociación entre las entidades en una dirección, y después invertir la vista para analizar en la dirección opuesta. Por ejemplo, para la relación entre Clientes y Pedidos, se deben formular dos preguntas: ¿cada cliente puede tener muchos pedidos? y ¿cada pedido puede tener muchos clientes? Por lo tanto, las relaciones se clasifican en tres tipos: uno a uno, uno a varios, y varios a varios, que se analizan en las secciones siguientes. Algunas personas afirman que varios a uno es un tipo de relación, pero en realidad es sólo una relación uno a varios desde una perspectiva inversa. Los tipos de relaciones se aprenden mejor mediante ejemplos. Aplicar las relaciones correctas es esencial para un diseño exitoso.

**Reglas de negocios**

Una regla de negocios es una política, procedimiento o norma adoptada por una organización. Las reglas de negocios son muy importantes en el diseño de una base de datos porque determinan los controles que deben aplicarse a los datos. En la *Ilustración* 6, se aprecia una regla de negocios que afirma que sólo se aceptarán pedidos de clientes que no tienen un saldo vencido. Casi todas las reglas de negocios se imponen mediante procedimientos manuales que se ordena seguir a los empleados o mediante la lógica de los programas de aplicación. Sin embargo, hay posibilidades de que se pase por alto a cada uno de éstos: los empleados olvidan un procedimiento manual o deciden no seguirlo, y personas autorizadas pueden actualizar directamente las bases de datos, superando los controles incluidos en los programas de aplicación. La base de datos puede funcionar muy bien como última línea de defensa. Las reglas de negocios se implementan en la base de datos como restricciones, que son las reglas formalmente definidas que limitan, de alguna manera, los valores de los datos de la base de datos.

**Relaciones y Conjunto de Relaciones**

*Una relación es la asociación que existe entre dos a más entidades.*

*Un conjunto de relaciones es un grupo de relaciones del mismo tipo.*

La cantidad de entidades en una relación determina el grado de la relación, por ejemplo, la relación ALUMNO-MATERIA es de grado 2, ya que intervienen la entidad ALUMNO y la entidad MATERIA, la relación PADRES, puede ser de grado 3, ya que involucra las entidades PADRE, MADRE e HIJO.

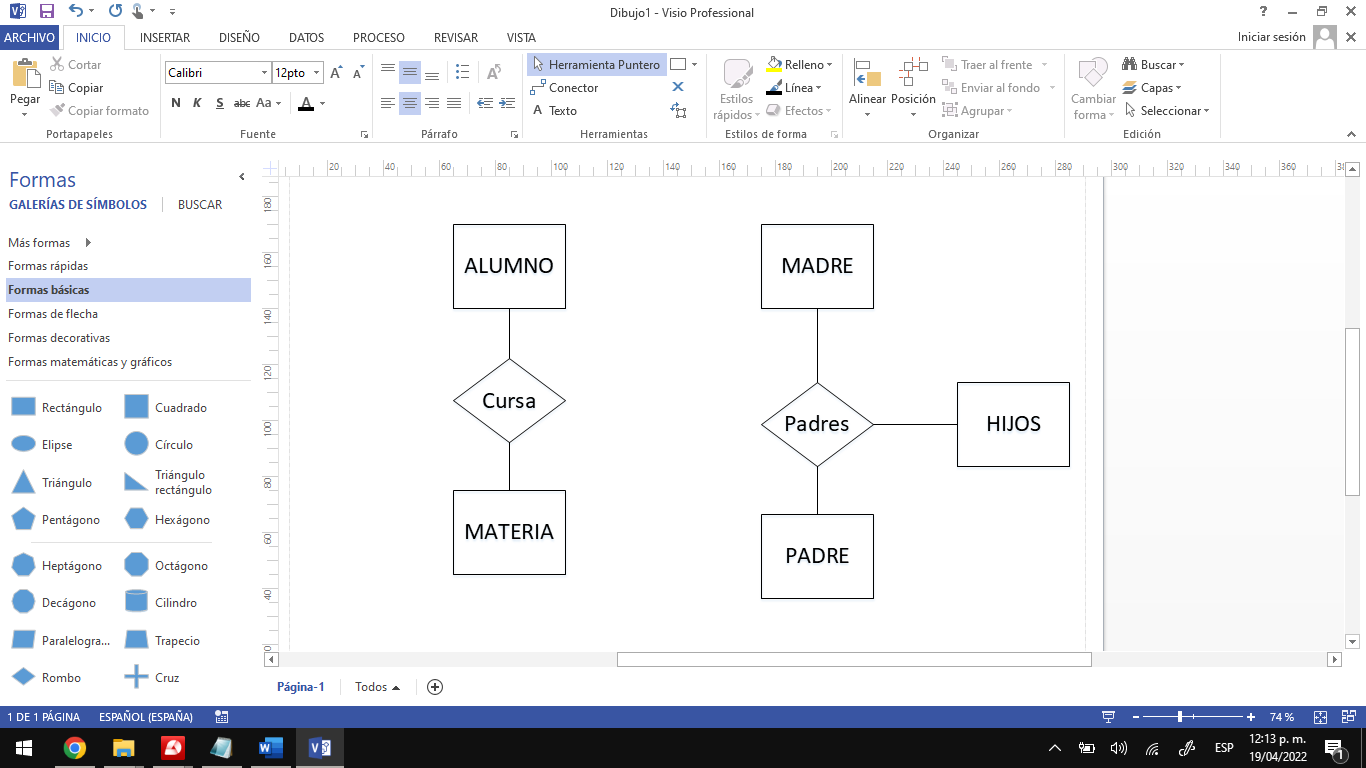


Ilustración . Ejemplo de Relación de grado 2 y Relaación de grado 3.

**Tipos de relaciones**

**Relación uno a uno. (1 a 1)**

Una relación uno a uno es una asociación en que una instancia de una entidad se puede asociar cuando mucho con una instancia de la otra entidad, y viceversa.

En la *Ilustración 6*, la relación entre las entidades Clientes y Cuentas por cobrar es uno a uno. Esto significa que un cliente puede tener cuando mucho una cuenta por cobrar asociada, y una cuenta puede tener cuando mucho un cliente asociado. La relación también es obligatoria en ambas direcciones, lo que significa que un cliente debe tener cuando menos una cuenta por cobrar asociada con él, y una cuenta por cobrar debe tener cuando menos un cliente asociado con ella. Al integrar esto, se entiende que, en la relación entre las entidades Clientes y Cuentas por cobrar, “un cliente sólo puede tener una cuenta por cobrar asociada, y una cuenta por cobrar sólo puede tener un cliente asociado”.

Otro concepto importante es la ***transferibilidad***. Se dice que una relación es transferible si el elemento principal puede cambiar con el tiempo o, dicho de otro modo, si el elemento secundario puede ser reasignado a un elemento principal distinto.

Por ejemplo: la relación asignación de automóvil que contiene a las entidades EMPLEADO, AUTO, es una relación 1 a 1, ya que asocia a un empleado con un único automóvil por lo tanto ningún empleado posee más de un automóvil asignado, y ningún vehículo se asigna a más de un trabajador.

Es representado gráficamente de la siguiente manera:

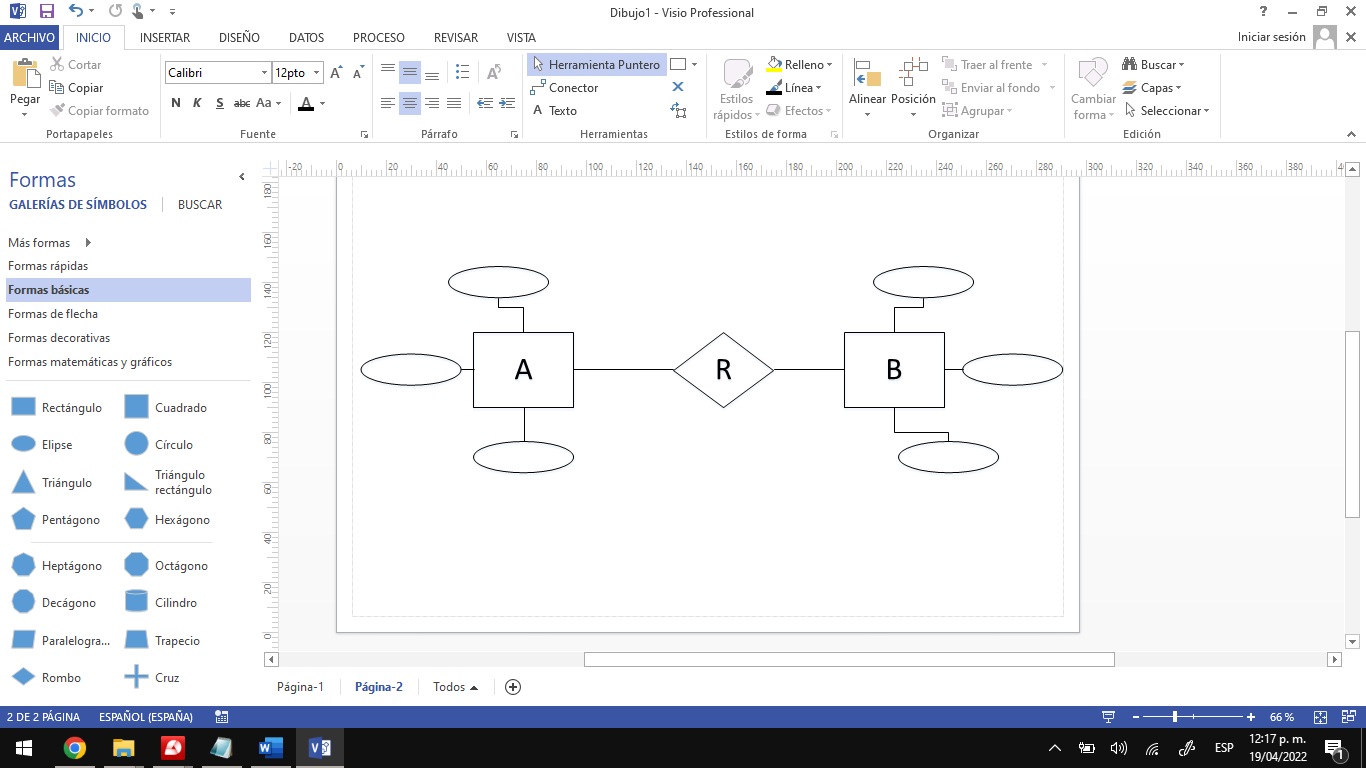


Ilustración . Relación uno a uno (1 a 1)

A: Representa a una entidad de cualquier tipo diferente a una entidad B.

R: en el diagrama representa a la relación que existe entre las entidades. El extremo de la flecha que se encuentra punteada indica el uno de la relación, en este caso, una entidad A ligada a una entidad B.

Las relaciones uno a uno son sorprendentemente raras entre entidades. En la práctica, las relaciones uno a uno que son obligatorias en ambas direcciones y no son transferibles representan un defecto en el diseño que debe corregirse al combinar las dos entidades. Después de todo, ¿una cuenta por cobrar no es sólo más información sobre el cliente? No vamos a recopilar datos sobre una cuenta por cobrar; en cambio, la información en la entidad Cuentas por cobrar sólo son más datos que recopilamos sobre un cliente.

**Relación uno a muchos. (1 a M)**

Una relación uno a varios es una asociación entre dos entidades en que cualquier instancia de la primera entidad puede asociarse con una o más instancias de la segunda, y cualquier instancia de la segunda entidad puede asociarse con cuando mucho una instancia de la primera. En la *Ilustración 6* se presentan dos de estas relaciones: entre las entidades Clientes y Pedidos, y entre las entidades Empleados y Pedidos. La relación entre Clientes y Pedidos, que sólo es obligatoria en una dirección, se lee así: “En cualquier momento, cada cliente puede tener de cero a varios pedidos, y cada pedido debe tener uno y sólo un cliente propietario”.

Su representación gráfica es la siguiente:

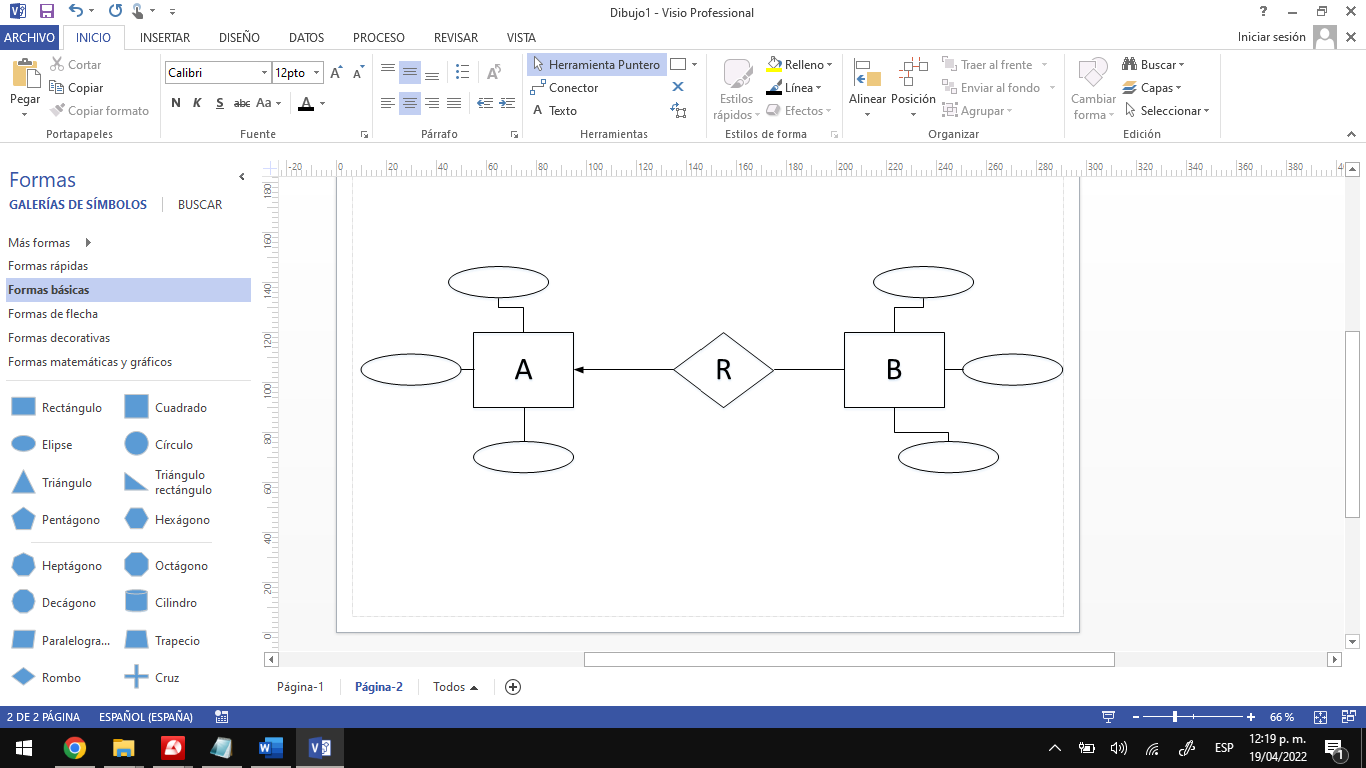


Ilustración . Relación uno a muchos (1 a M)

Nótese en este caso que el extremo punteado de la flecha de la relación de A y B, indica una entidad A conectada a muchas entidades B.

Las relaciones uno a varios son muy comunes. En realidad, son el bloque de construcción fundamental del modelo relacional de base de datos, porque todas las relaciones de una base de datos relacional se implementan como si fueran uno a varios. Es raro que sean opcionales en el lado “uno” e incluso más raro que sean obligatorias en el lado “varios”, pero estas situaciones ocurren.

**Muchos a uno. (M a 1)**

Indica que una entidad del tipo B puede relacionarse con cualquier cantidad de entidades del tipo A, mientras que cada entidad del tipo A solo puede relacionarse con solo una entidad del tipo B.

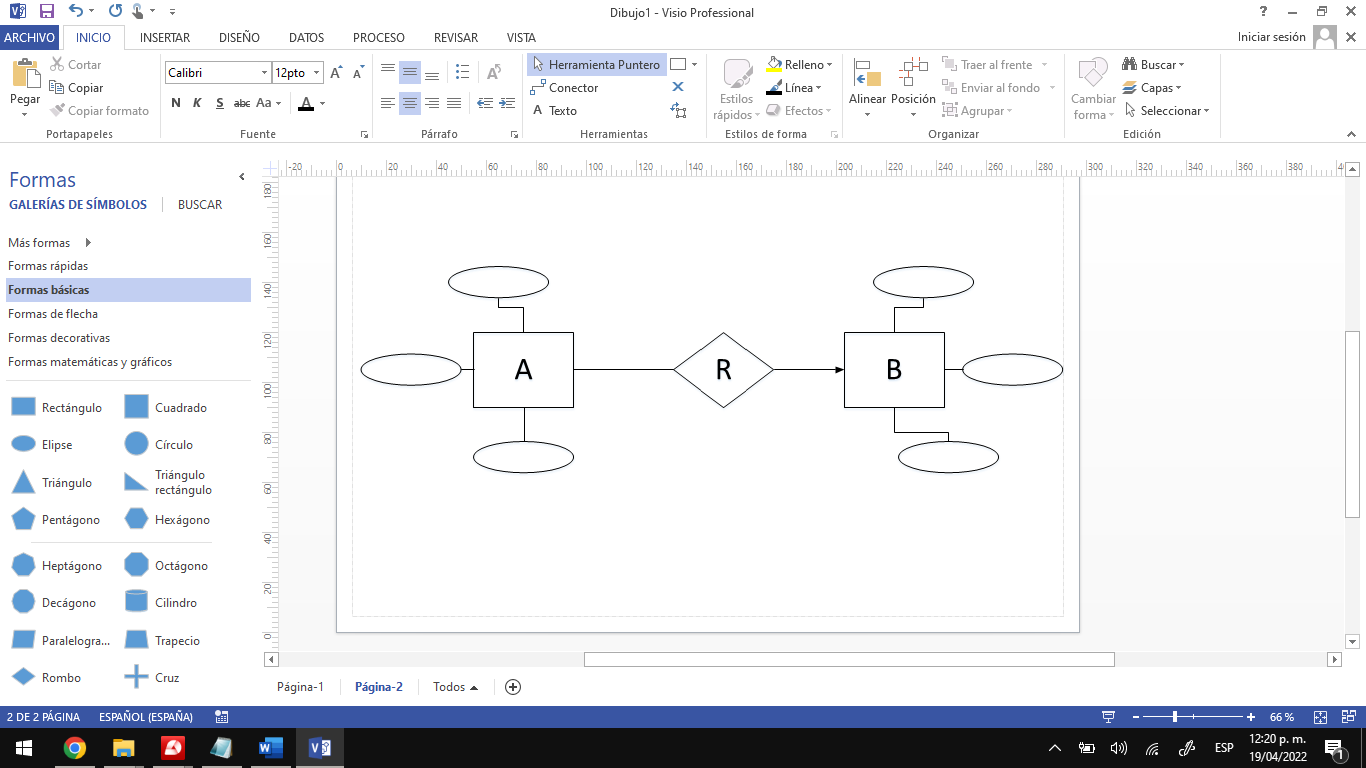


Ilustración . Relación muchos a uno (M a 1)

**Muchas a muchas. (M a M)**

Una relación varios a varios es una asociación entre dos entidades en que cualquier instancia de la primera entidad puede asociarse con cero, una o más instancias de la segunda, y viceversa. De nuevo en la *Ilustración 6*, la relación entre Pedidos y Productos es varios a varios. La relación se lee así: “En cualquier momento, cada pedido contiene de cero a varios productos, y cada producto aparece en cero a varios pedidos”.

A los datos que pertenecen a una relación varios a varios se les denomina datos de intersección. Los datos no tienen sentido a menos que los asocie con ambas entidades al mismo tiempo. Por ejemplo, Cantidad no tiene sentido a menos que sepa quién (cuál cliente) pidió qué (cuál producto).

Establece que cualquier cantidad de entidades del tipo A pueden estar relacionados con cualquier cantidad de entidades del tipo B.

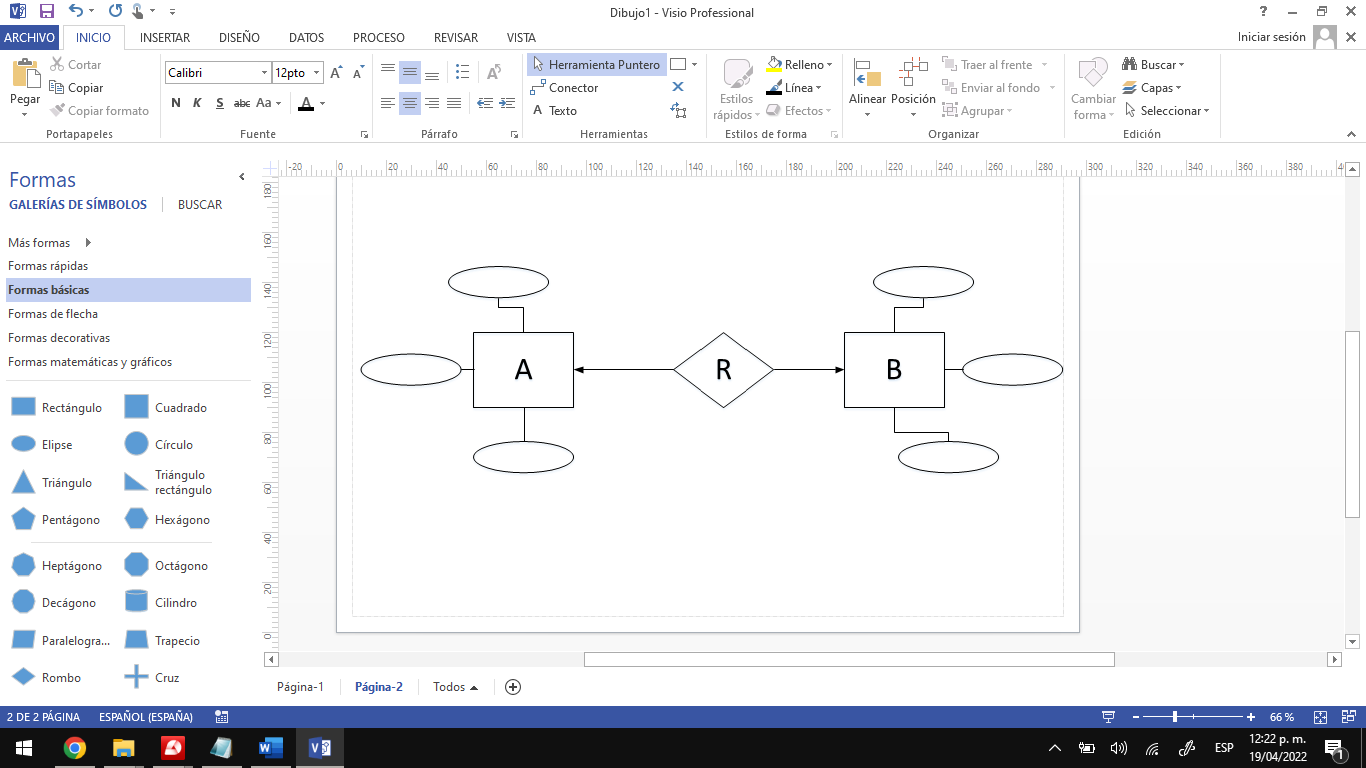


Ilustración . Relación muchos a muchos (M a N)

A los tipos de relaciones antes descritos, también se le conoce como cardinalidad.

## Cardinalidad o Llaves primarias

Como ya se ha mencionado anteriormente, la distinción de una entidad entre otra se debe a sus atributos, lo cual lo hacen único. Una llave primaria es aquel atributo el cual consideramos clave para la identificación de los demás atributos que describen a la entidad.

Por ejemplo, si consideramos la entidad ALUMNO, podríamos tener los siguientes atributos: Nombre, Semestre, Especialidad, Dirección, Teléfono, Número de control, de todos estos atributos el que podremos designar como llave primaria es el número de control, ya que es diferente para cada alumno y este nos identifica en la institución.

Claro que puede haber más de un atributo que pueda identificarse como llave primaria en este caso se selecciona la que consideremos más importante, los demás atributos son denominados **llaves secundarias.**

Una clave o llave primaria es indicada gráficamente en el modelo E-R con una línea debajo del nombre del atributo o lo que es lo mismo se subraya el atributo.

Ejemplos de modelos E-R, considerando las cardinalidades que existen entre ellos:

**Relación Uno a Uno**

Problema:

Diseñar el modelo E-R, para la relación Registro de automóvil que consiste en obtener la tarjeta de circulación de un automóvil con los siguientes datos:

- Automóvil- Modelo, Placas, Color

- Tarjeta de circulación -Propietario, No serie, Tipo.

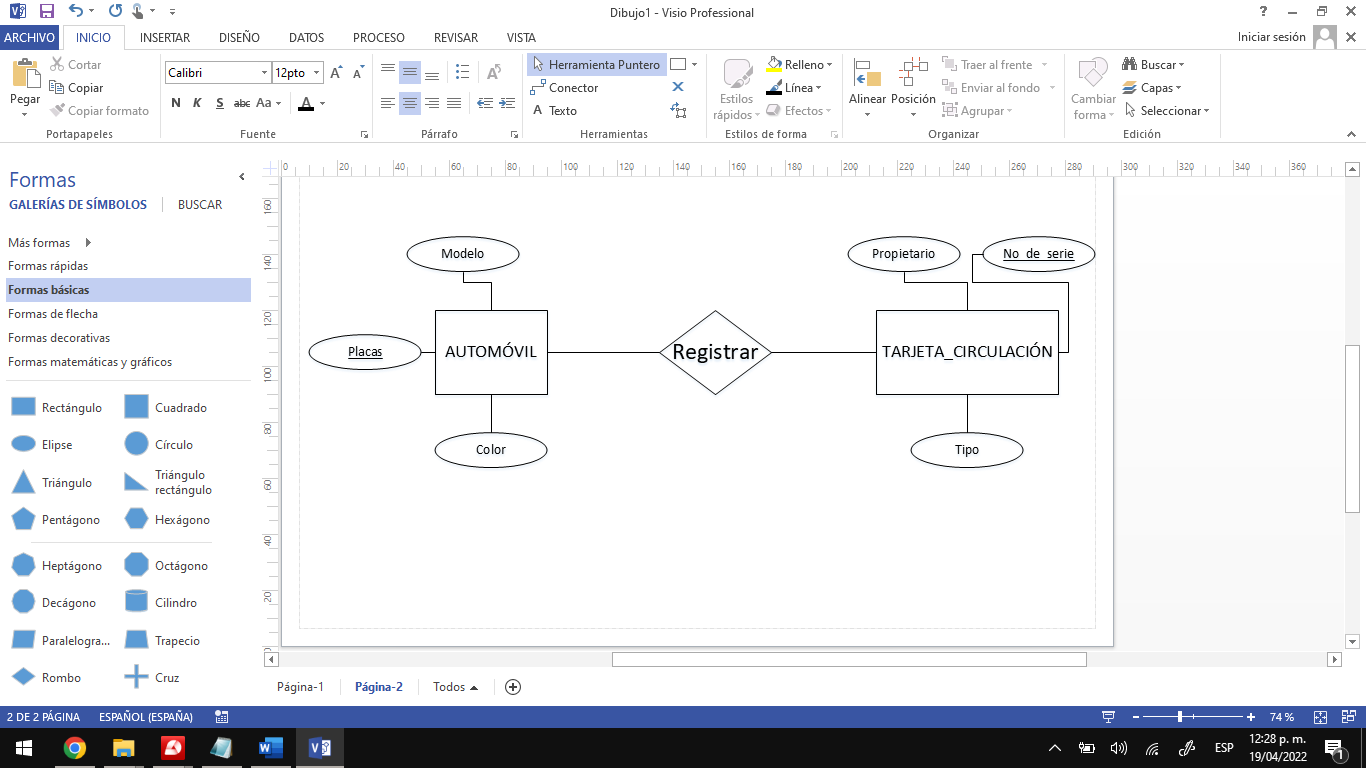


Ilustración . Ejemplo 1 de relación uno a uno (1 a 1)

Indicamos con este ejemplo que existe una relación de pertenencia de uno a uno, ya que existe una tarjeta de circulación registrada por cada automóvil.

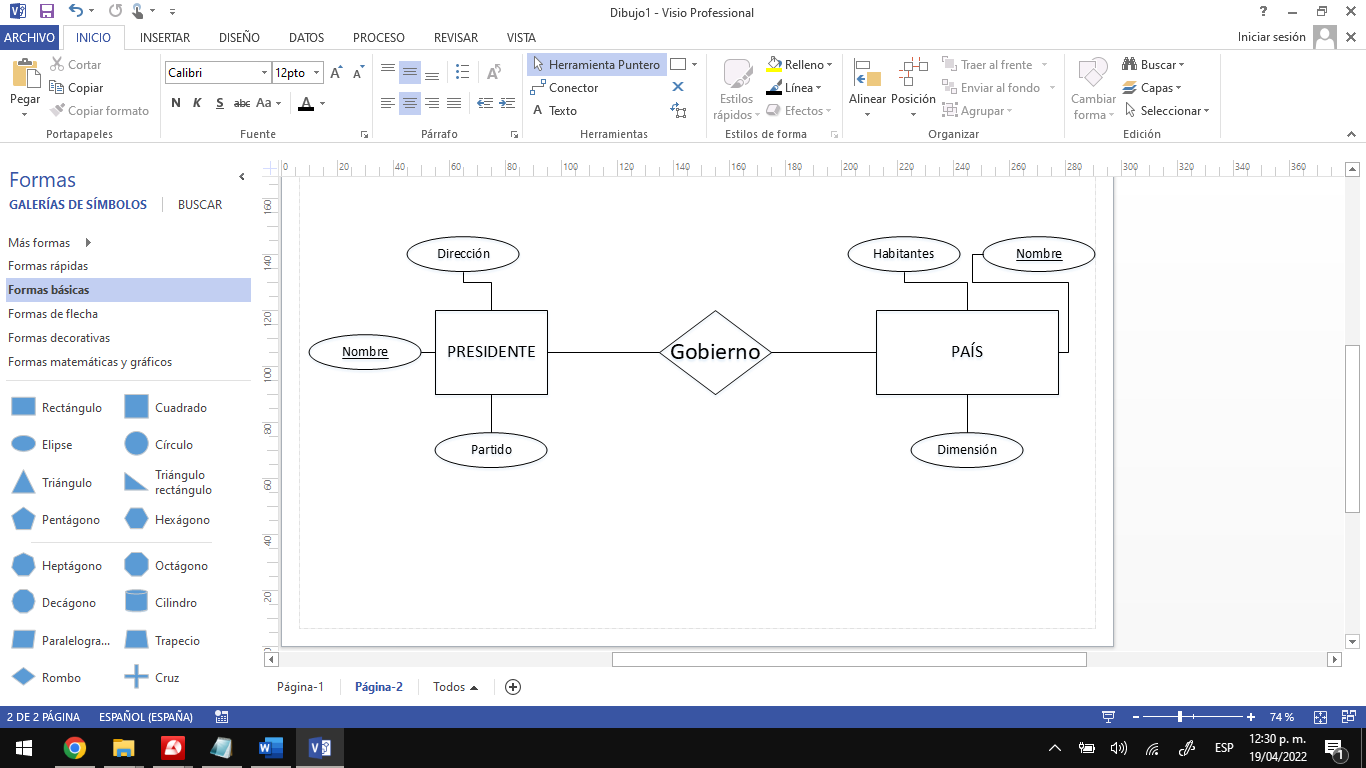


Ilustración . Ejemplo 2 de relación uno a uno (1 a 1)

**Relación uno a muchos**

El siguiente ejemplo indica que un cliente puede tener muchas cuentas, pero que una cuenta puede llegar a pertenecer a un solo cliente (Decimos puede, ya que existen cuentas registradas a favor de más de una persona).

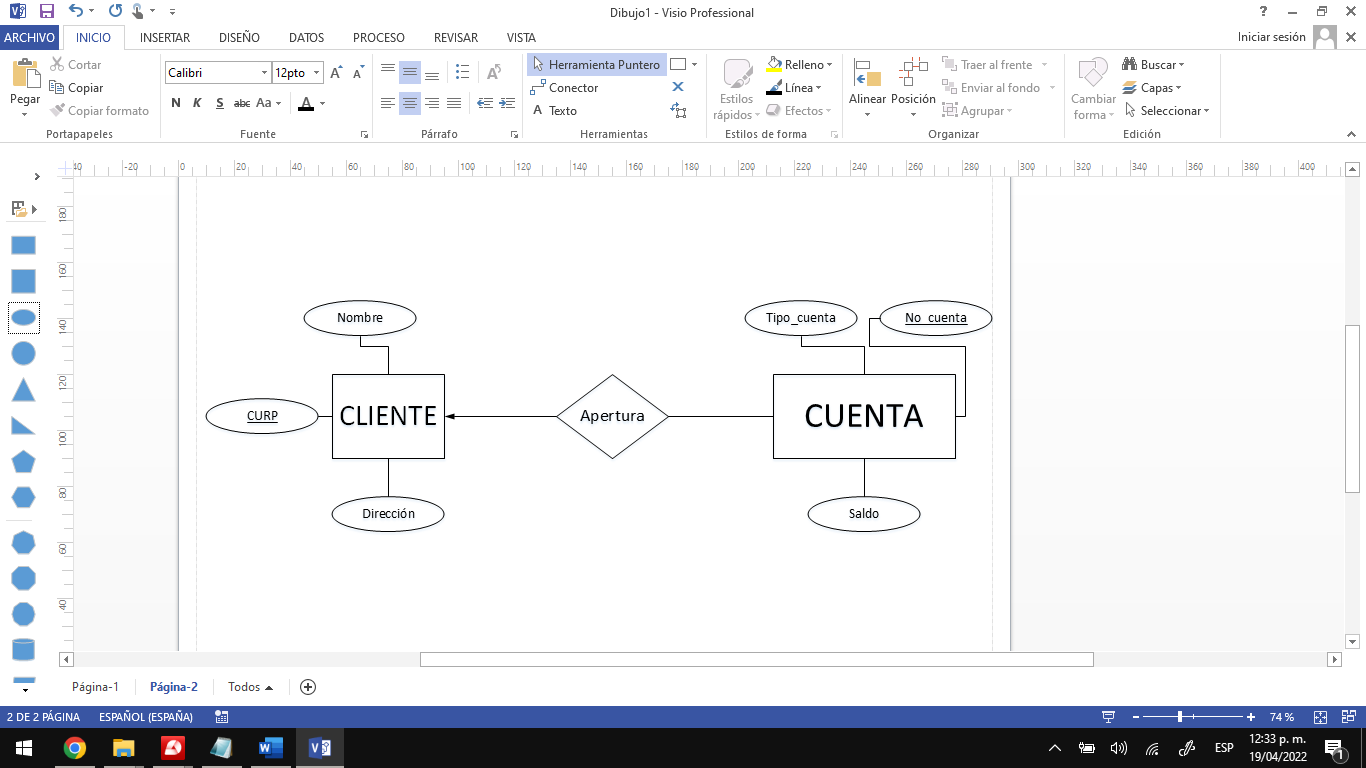


Ilustración . Ejemplo Relación Uno a Muchos (1 a M)

**Relación muchos a muchos**

El ejemplo de los estudiantes y las asignaturas donde un estudiante recibe varias asignaturas y estas a su vez la reciben varios estudiantes.

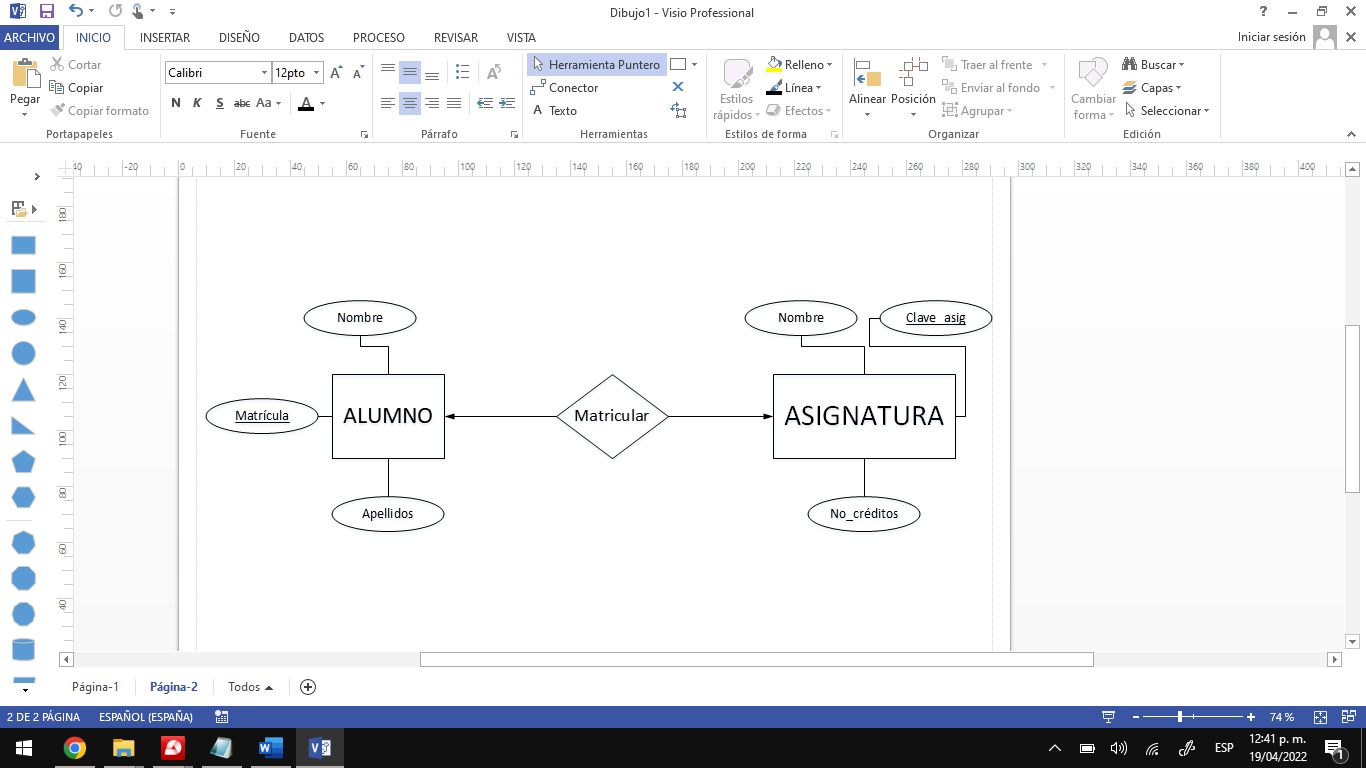


Ilustración . Ejemplo 1 de relación muchos a muchos (M a N)

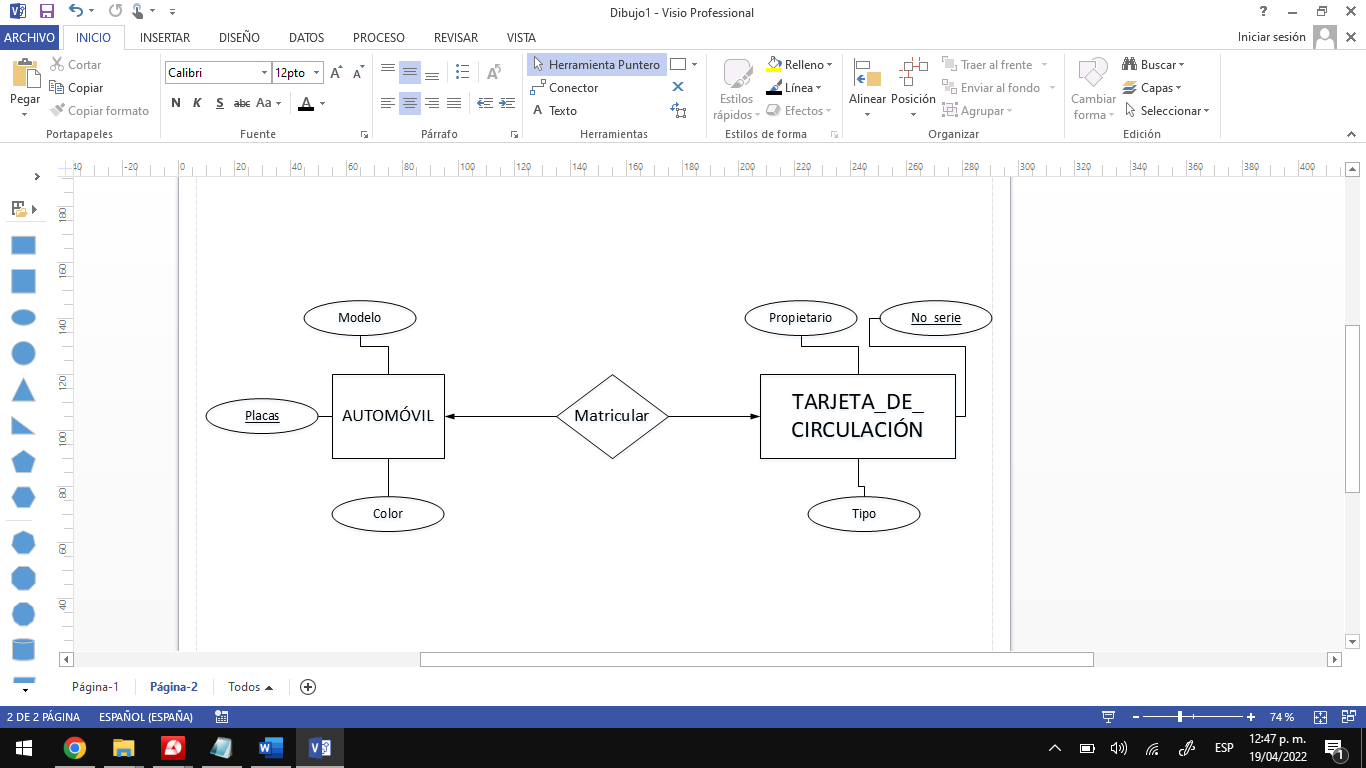


Ilustración . Ejemplo 2 de relación muchos a muchos (M a N)

Este tipo de relación de Muchos a Muchos (M a N) extiende la capacidad semántica del MER aplicando sobre sus objetos básicos (entidad y relación) diferentes operaciones. En este caso se habla de Modelo Entidad Relación extendido.

## Generalización

Es el resultado de la unión de 2 o más conjuntos de entidades (de bajo nivel) para producir un conjunto de entidades de más alto nivel.

La generalización consiste en identificar todos aquellos atributos iguales de un conjunto de entidades para formar una entidad(es) global(es) con dichos atributos semejantes, dicha entidad(es) global(es) quedara a un nivel más alto al de las entidades origen.

Ejemplo: Se tiene las entidades Cta\_Ahorro y Cta\_Cheques, ambas tienen los atributos semejantes de No\_Cta y Saldo, aunque además de estos dos atributos, Cta\_Ahorro tiene el atributo Tasa Interés y Cta\_Cheques el atributo Saldo Deudor. De todos estos atributos podemos juntar (generalizar) No\_Cta y Saldo que son iguales en ambas entidades.

Entonces tenemos:

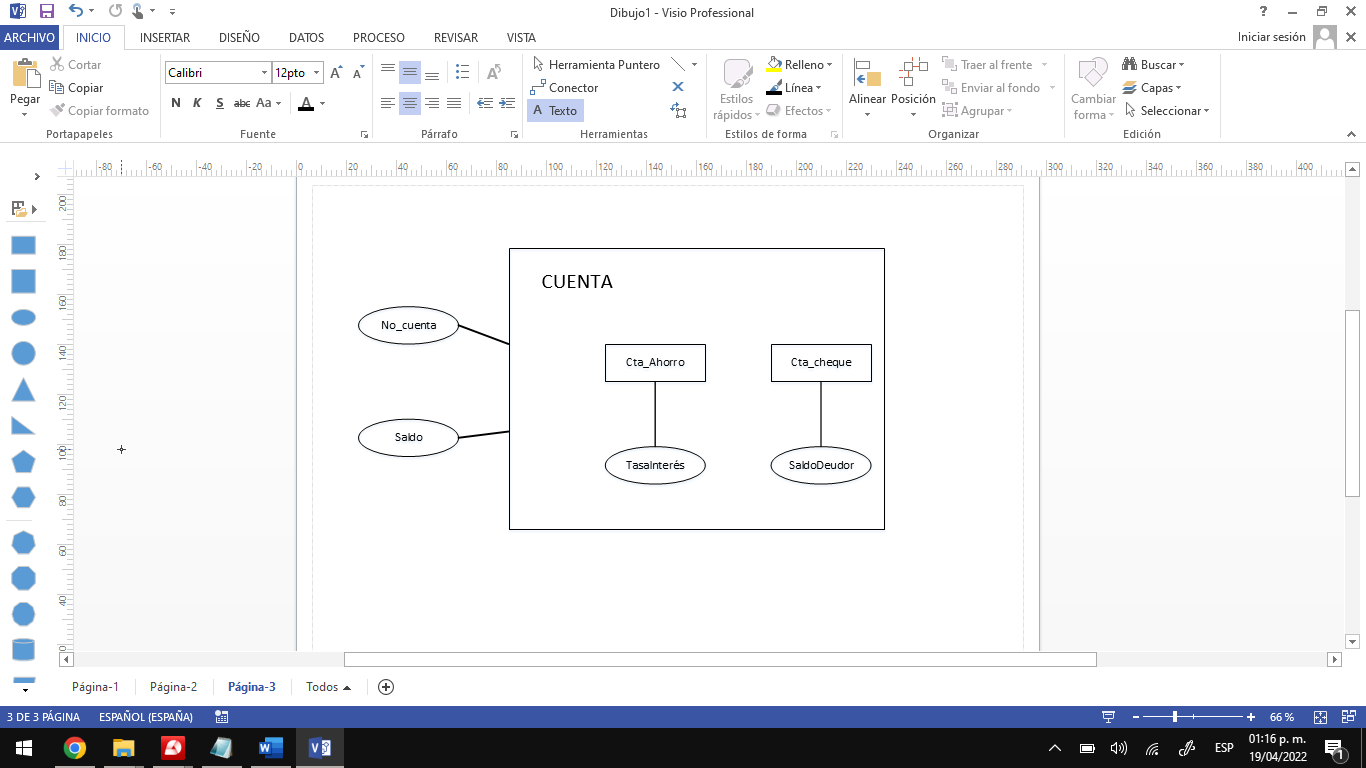


Ilustración . Generalización

Podemos leer esta gráfica como: La entidad Cta\_Ahorro hereda de la entidad CUENTA los atributos No\_Cta y saldo, además del atributo de TasaInteres, de forma semejante Cta\_cheque tiene los atributos de No\_Cta, Saldo y Saldo Deudor.

Como podemos observar la Generalización trata de eliminar la redundancia (repetición) de atributos, al englobar los atributos semejantes. La entidad(es) de bajo nivel heredan todos los atributos correspondientes.

Como se puede apreciar Cuenta es el supertipo de esta jerarquía y constituye la generalización de los subtipos de entidades Cta\_Ahorro y Cta\_Cheque y a su vez estos son una especialización del supertipo Cuenta.

## Agregación

La agregación surge de la limitación que existe en el modelado de E-R, al no permitir expresar las relaciones entre relaciones de un modelo E-R en el caso de que una relación X se quiera unir con una entidad cualquiera para formar otra relación.

La Agregación consiste en agrupar por medio de un rectángulo a la relación (representada por un rombo) junto con las entidades y atributos involucrados en ella, para formar un grupo que es considerado una entidad y ahora sí podemos relacionarla con otra entidad.

El ejemplo que representa la situación de la producción en las empresas, la relación ternaria Trab-Maq-Pieza representa la idea de que una actividad en la empresa se describe en términos de "un obrero en alguna máquina produce una pieza dada en alguna cantidad específica". Sin embargo, la misma situación puede ser vista de forma algo diferente. En la empresa las máquinas pueden estar asignadas a los obreros y estos "equipos”, producir piezas en cierta cantidad. En el MER original esta situación no hubiera podido ser modelada correctamente, ya que una relación no puede relacionarse con otra relación o entidad.

Con la operación de Agregación esta situación se resuelve fácilmente, tal y como se muestra en la figura:

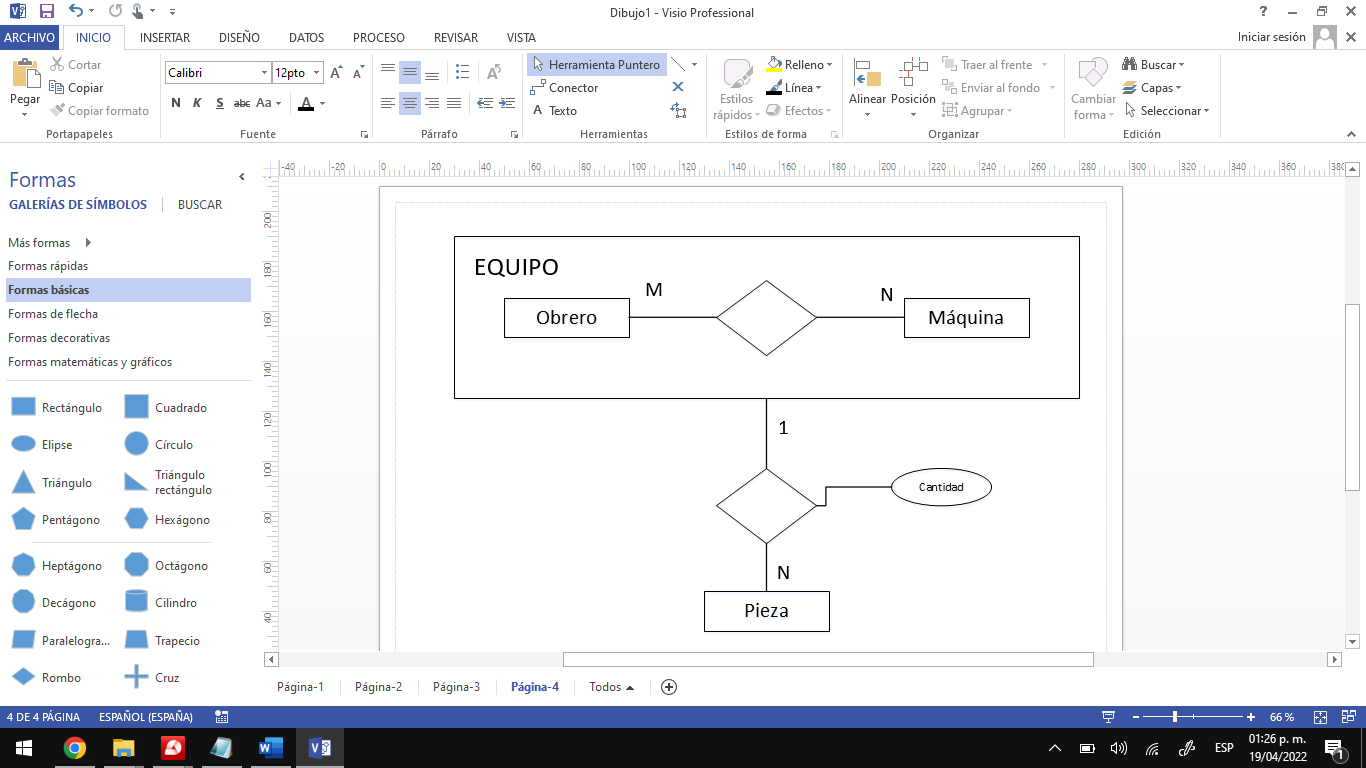


Ilustración . Agregación

## Reducción de Diagramas E-R a tablas

Un diagrama E-R, puede ser representado también a través de una colección de tablas. Para cada una de las entidades y relaciones existe una tabla única a la que se le asigna como nombre el del conjunto de entidades y de las relaciones respectivamente, cada tabla tiene un número de columnas que son definidas por la cantidad de atributos, las columnas tendrán el nombre del atributo.

Ejemplo: Venta, en la que intervienen las entidades de Empleado con los atributos RFC o CI, nombre, puesto, salario y Artículo con los atributos Clave, descripción, costo.

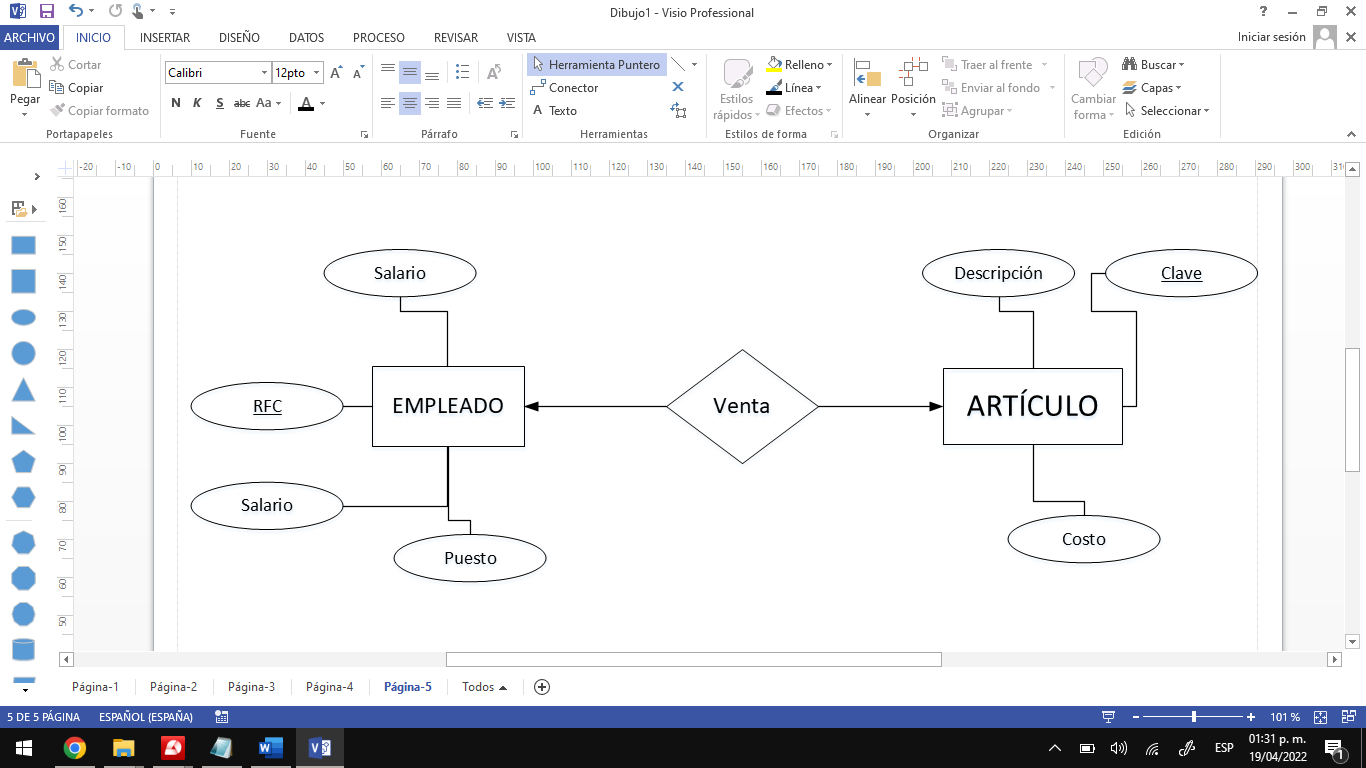


Ilustración . Diagrama E-R que será representado en tablas

Entonces las tablas resultantes siguiendo la descripción anterior son:

Tabla . Empleado

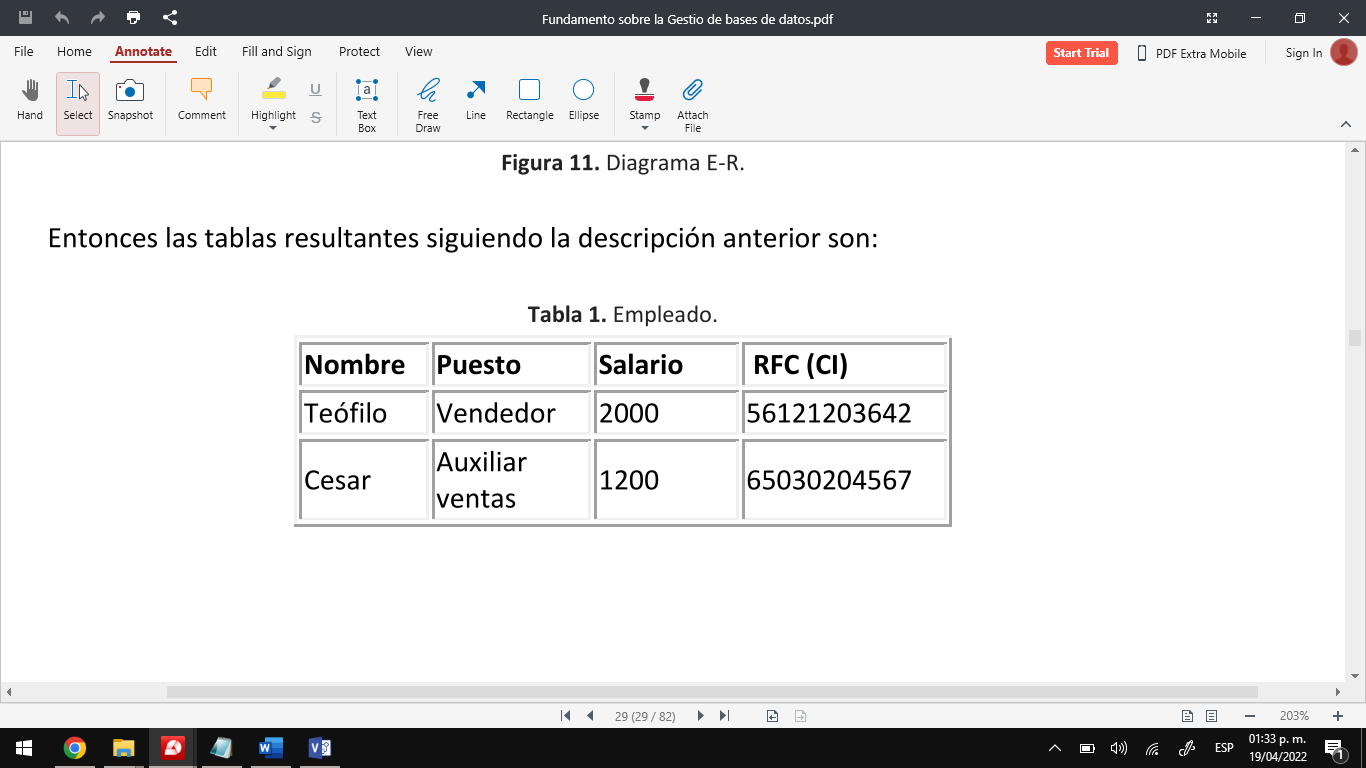


Tabla . Artículo

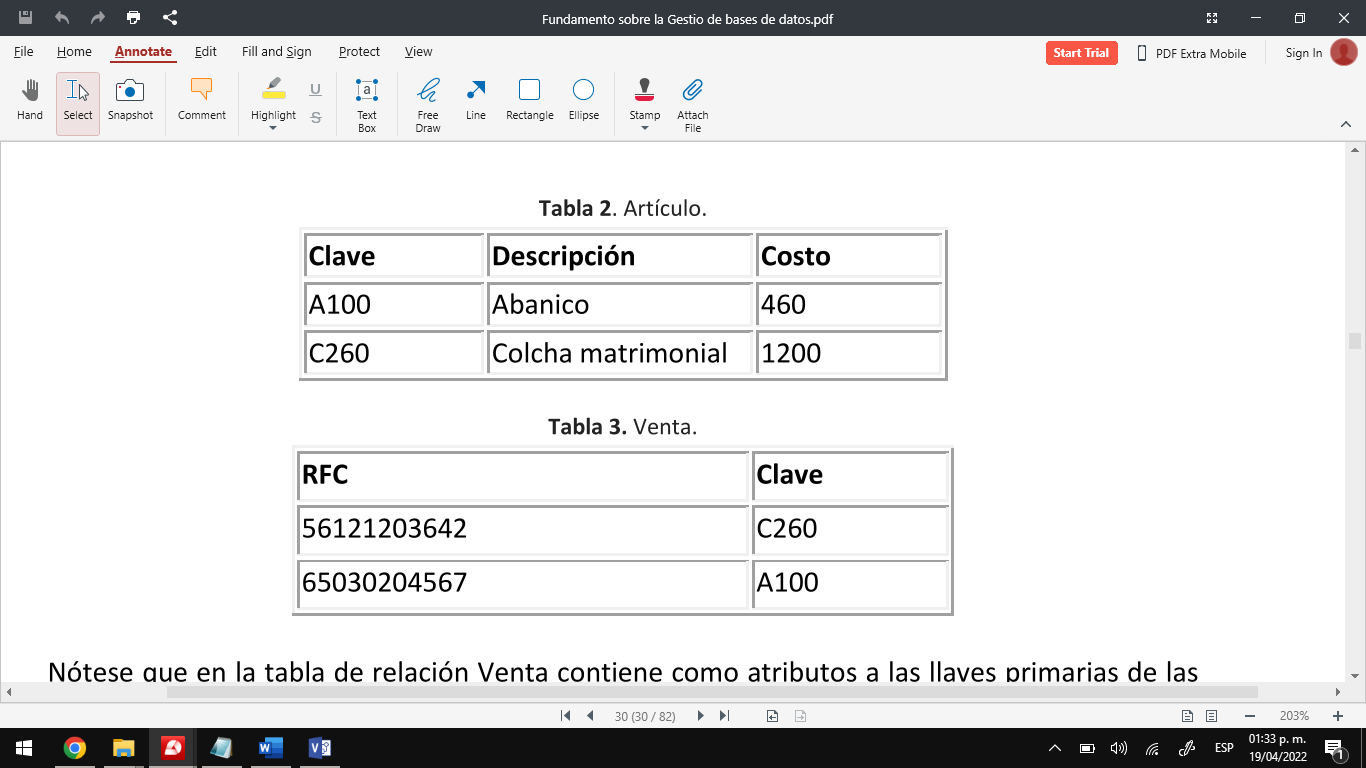
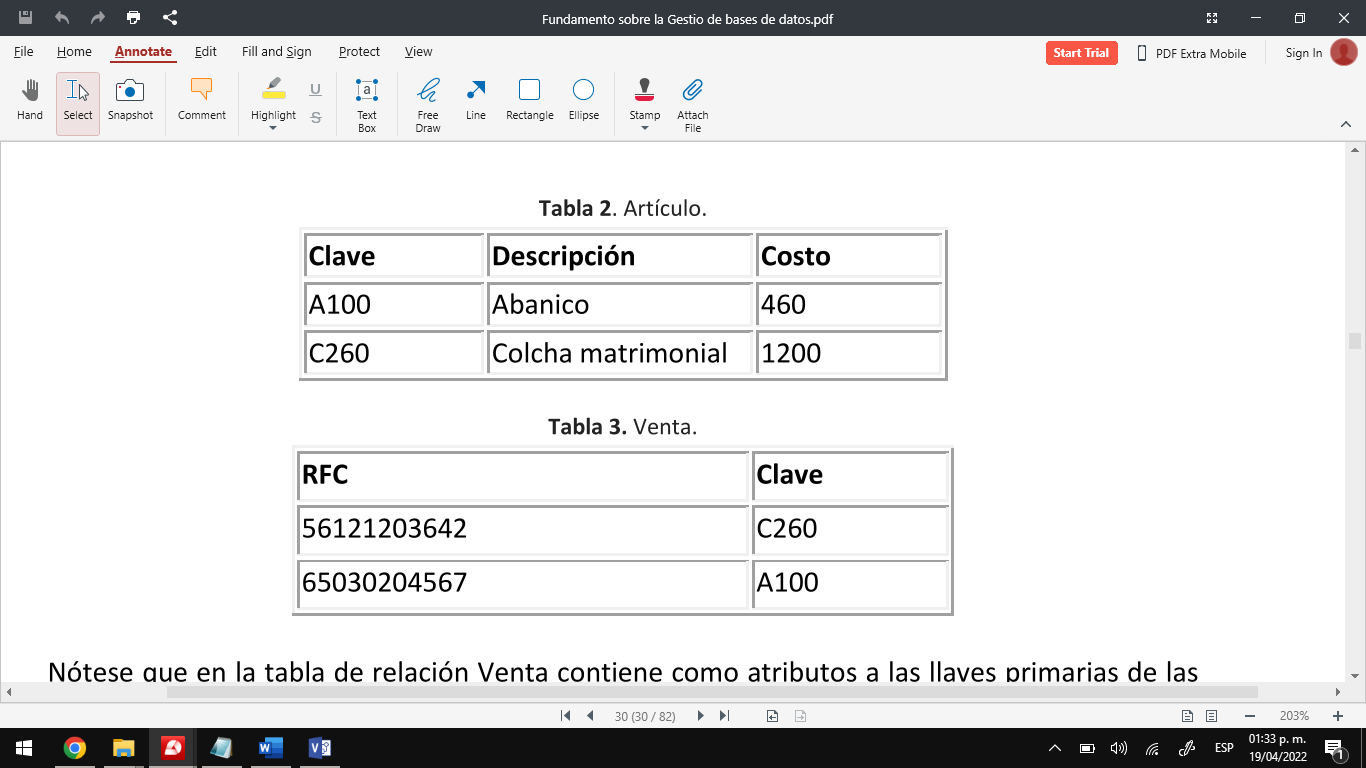


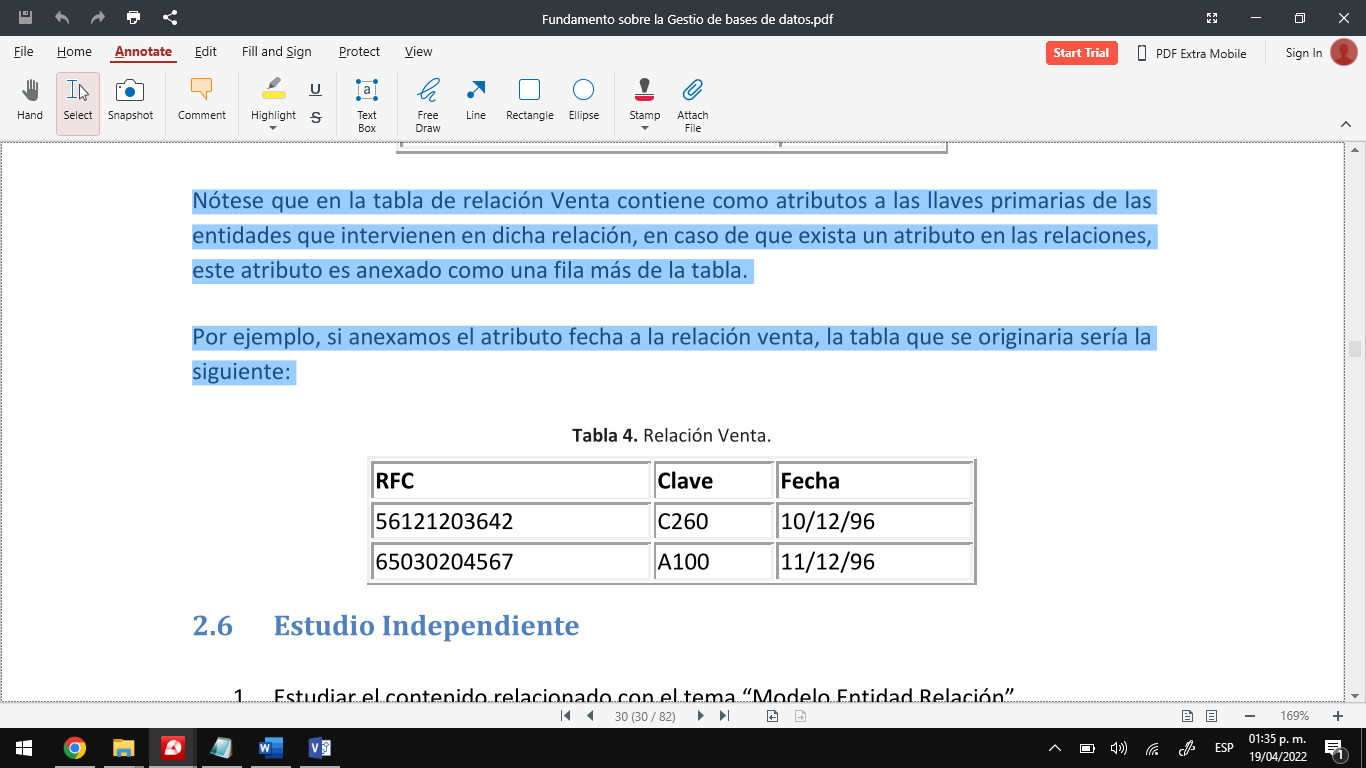
Tabla . Venta



Nótese que en la tabla de relación Venta contiene como atributos a las llaves primarias de las entidades que intervienen en dicha relación, en caso de que exista un atributo en las relaciones, este atributo es anexado como una fila más de la tabla.

Por ejemplo, si anexamos el atributo fecha a la relación venta, la tabla que se originaria sería la siguiente:

Tabla . Relación Venta



## Ejemplos resueltos del modelo entidad – relación

# Modelación Conceptual de Bases de Datos

## Modelo Relacional

En 1970, el modo en que se veían las bases de datos cambió por completo cuando E. F. Codd introdujo el modelo relacional. En aquellos momentos, el enfoque existente para la estructura de las bases de datos utilizaba punteros físicos (direcciones de disco) para relacionar registros de distintos ficheros. Si, por ejemplo, se quería relacionar un registro A con un registro B, se debía añadir al registro A un campo conteniendo la dirección en disco (un puntero físico) del registro B. Codd demostró que estas bases de datos limitaban en gran medida los tipos de operaciones que los usuarios podían realizar sobre los datos. Además, estas bases de datos eran muy vulnerables a cambios en el entorno físico. Si se añadían los controladores de un nuevo disco al sistema y los datos se movían de una localización física a otra, se requería una conversión de los ficheros de datos. Estos sistemas se basaban en el modelo de red y el modelo jerárquico, los dos modelos lógicos que constituyeron la primera generación de los SGBD.

El modelo relacional representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los SGBD de la primera generación, lo que constituye otro punto a su favor.

Dada la popularidad del modelo relacional, muchos sistemas de la primera generación se han modificado para proporcionar una interfaz de usuario relacional, con independencia del modelo lógico que soportan (de red o jerárquico).

En los últimos años, se han propuesto algunas extensiones al modelo relacional para capturar mejor el significado de los datos, para disponer de los conceptos de la orientación a objetos y para disponer de capacidad deductiva.

El modelo relacional, como todo modelo de datos, tiene que ver con tres aspectos de los datos, que son los que se presentan en los siguientes apartados de este capítulo: qué características tiene la estructura de datos, cómo mantener la integridad de los datos y cómo realizar el manejo de los mismos.

El modelo relacional se ha establecido actualmente como el principal modelo de datos para las aplicaciones de procesamiento de datos. Ha conseguido la posición principal debido a su simplicidad, que facilita el trabajo del programador en comparación con otros modelos anteriores como el de red y el jerárquico.

Se basa en la teoría matemática de las relaciones, suministrándose por ello una fundamentación teórica que permite aplicar todos los resultados de dicha teoría a problemas tales como el diseño de sublenguajes de datos y otros.

Representación de una relación

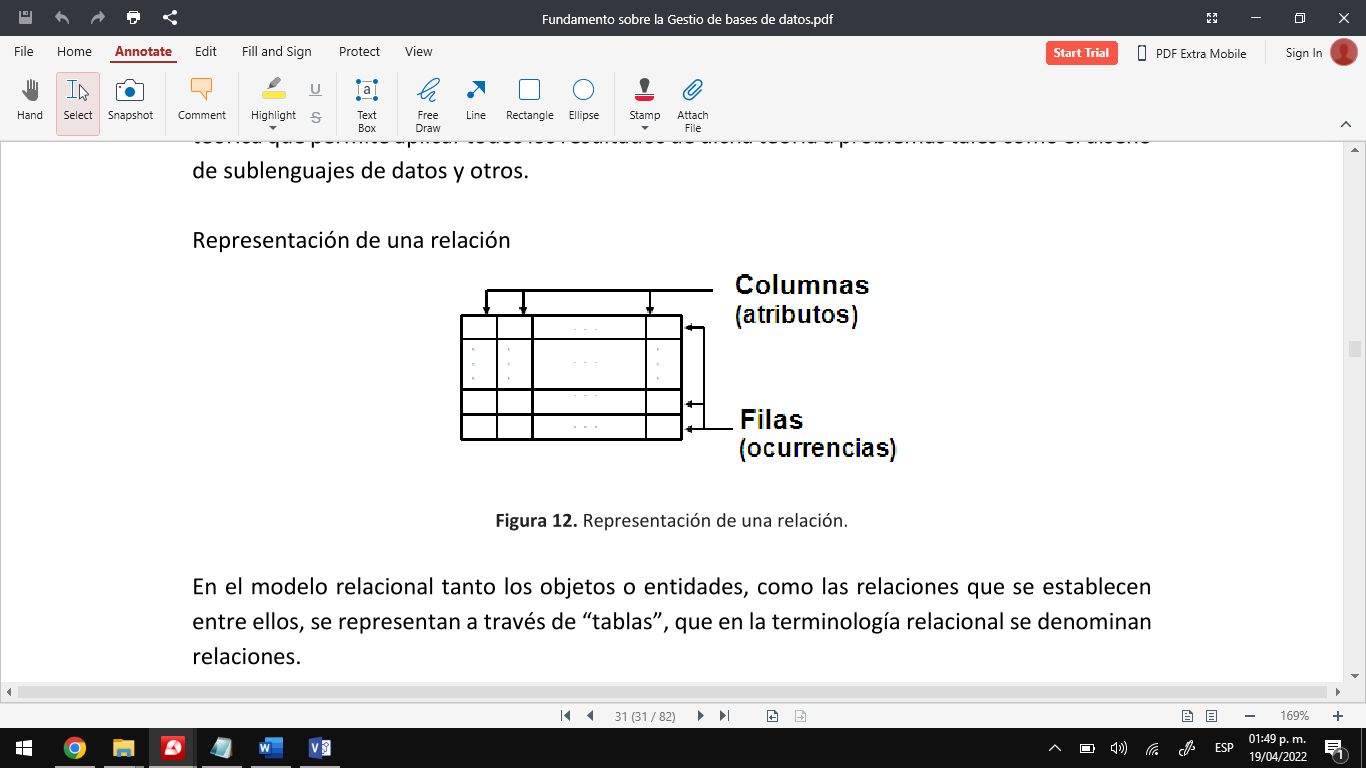


Ilustración . Representación de una relación

El modelo relacional se basa en el concepto matemático de relación, que gráficamente se representa mediante una tabla. Codd, que era un experto matemático, utilizó una terminología perteneciente a las matemáticas, en concreto de la teoría de conjuntos y de la lógica de predicados.

Una relación es una tabla con columnas y filas. Un SGBD sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas.

Esta percepción sólo se aplica a la estructura lógica de la base de datos, no se aplica a la estructura física de la base de datos, que se puede implementar con distintas estructuras de almacenamiento.

Un atributo es el nombre de una columna de una relación. En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros. Los atributos pueden aparecer en la relación en cualquier orden.

Por ejemplo, la información de los clientes de una empresa determinada se representa mediante la relación CLIENTES de la *Ilustración 21*, que tiene columnas para los atributos codcli (código del cliente), nombre (nombre y apellidos del cliente), dirección (calle y número donde se ubica el cliente), codpostal (código postal correspondiente a la dirección del cliente) y codpue (código de la población del cliente). La información sobre las poblaciones se representa mediante la relación PUEBLOS de la misma figura, que tiene columnas para los atributos codpue (código de la población), nombre (nombre de la población) y codpro (código de la provincia en que se encuentra la población).

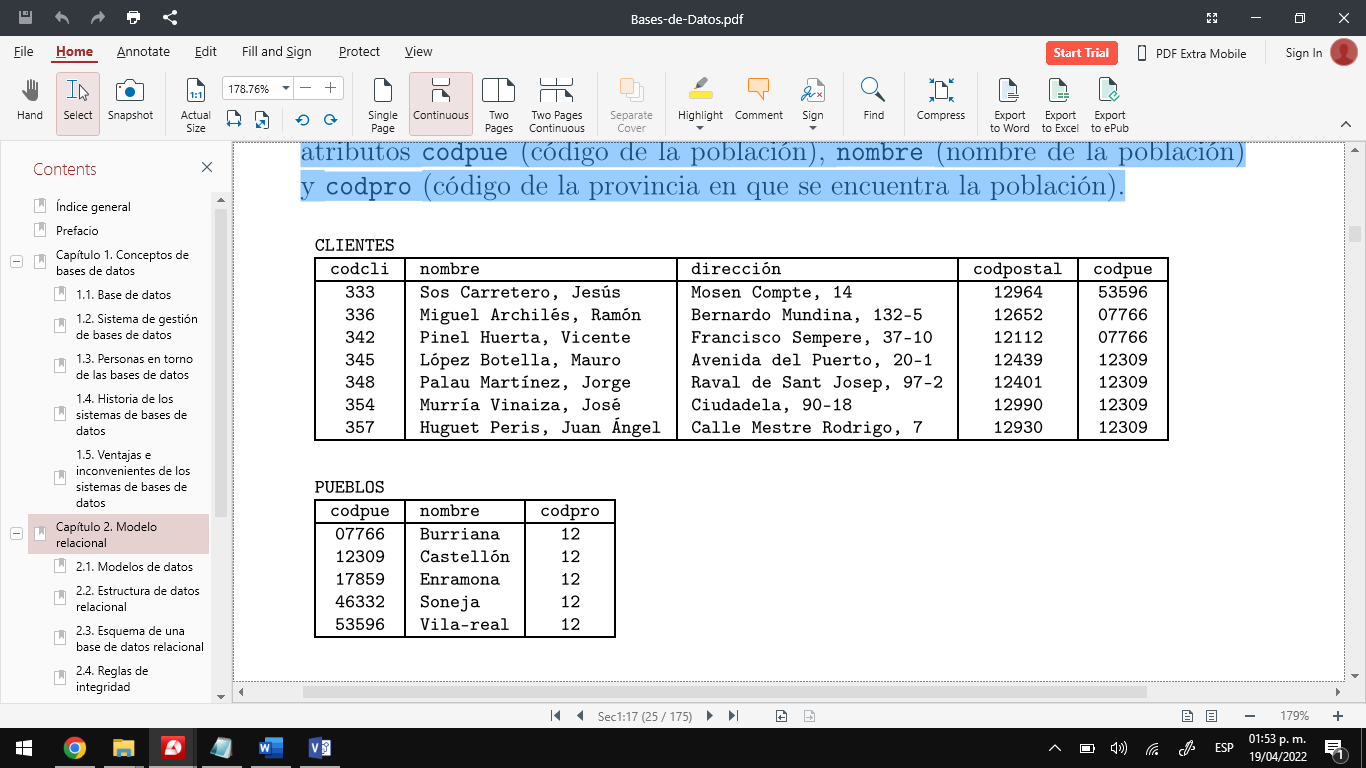


Ilustración . Relaciones que almacenan los datos de los clientes y sus poblaciones.

Un dominio es el conjunto de valores legales de uno o varios atributos. Los dominios constituyen una poderosa característica del modelo relacional. Cada atributo de una base de datos relacional se define sobre un dominio, pudiendo haber varios atributos definidos sobre el mismo dominio. La *Ilustración 22* muestra los dominios de los atributos de la relación CLIENTES.

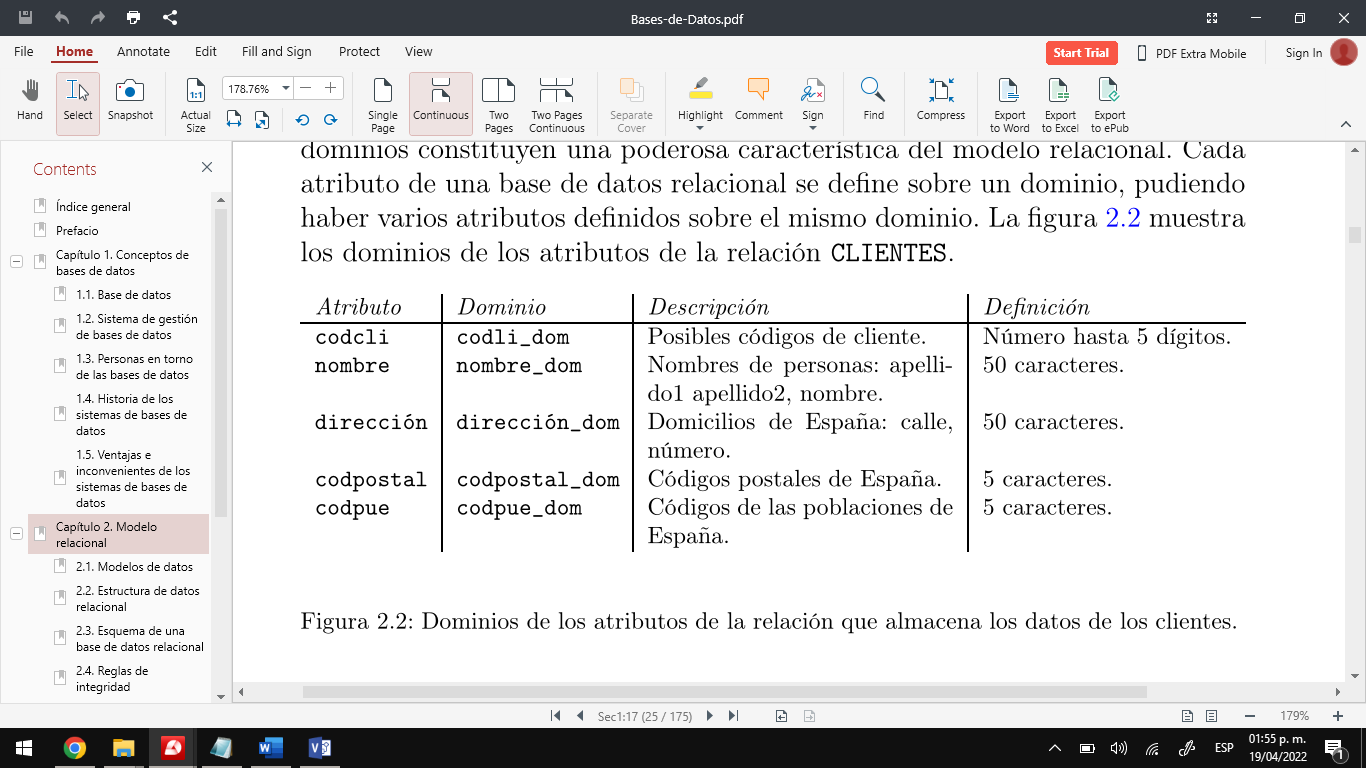


Ilustración . Dominios de los atributos de la relación que almacena los datos de los clientes

El concepto de dominio es importante porque permite que el usuario defina, en un lugar común, el significado y la fuente de los valores que los atributos pueden tomar. Esto hace que haya más información disponible para el sistema cuando éste va a ejecutar una operación relacional, de modo que las operaciones que son semánticamente incorrectas, se pueden evitar. Por ejemplo, no tiene sentido comparar el nombre de una calle con un número de teléfono, aunque los dos atributos sean cadenas de caracteres. Sin embargo, el importe mensual del alquiler de un inmueble no estará definido sobre el mismo dominio que el número de meses que dura el alquiler, pero sí tiene sentido multiplicar los valores de ambos dominios para averiguar el importe total al que asciende el alquiler. Los SGBD relacionales no ofrecen un soporte completo de los dominios ya que su implementación es extremadamente compleja.

*Una tupla es una fila de una relación*. Los elementos de una relación son las tuplas o filas de la tabla. En la relación CLIENTES, cada tupla tiene cinco valores, uno para cada atributo. Las tuplas de una relación no siguen ningún orden.

*El grado de una relación es el número de atributos que contiene*. La relación CLIENTES es de grado cinco porque tiene cinco atributos. Esto quiere decir que cada fila de la tabla es una tupla con cinco valores. El grado de una relación no cambia con frecuencia.

*La cardinalidad de una relación es el número de tuplas que contiene*. Ya que en las relaciones se van insertando y borrando tuplas a menudo, la cardinalidad de las mismas varía constantemente.

*Una base de datos relacional es un conjunto de relaciones normalizadas.* Una relación está normalizada si en la intersección de cada fila con cada columna hay un solo valor.

## Normalización

La teoría de la normalización se ha desarrollado para obtener estructuras de datos eficientes que eviten las anomalías de actualización.

La normalización es el proceso de simplificar la relación entre los campos de un registro. Por medio de la normalización un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

* Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
* Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
* Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
* Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

La teoría de normalización tiene como fundamento el concepto de formas normales; se dice que una relación está en una determinada forma normal si satisface un conjunto de restricciones.

* + Primera Forma Normal (1FN).
  + Segunda Forma Normal (2FN).
  + Tercera Forma Normal (3FN).
  + Forma Normal de Boyce -Codd (FNBC).

Existen, además, la cuarta (4FN) y la quinta (5FN) formas normales.

## Formas normales

Para evitar anomalías de actualización, es recomendable llegar al menos hasta la tercera forma normal o, mejor aún, hasta la forma normal de Boyce-Codd.

***Primera Forma Normal:***

Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valores atómicos.

Abreviada como 1FN, se considera que una relación se encuentra en la primera forma normal cuando cumple lo siguiente:

1. Las celdas de las tablas poseen valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
2. Todos los ingresos en cualquier columna (atributo) deben ser del mismo tipo.
3. Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.
4. Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no es importante.

Por lo general la mayoría de las relaciones cumplen con estas características, así que podemos decir que la mayoría de las relaciones se encuentran en la primera forma normal.

Visto de forma más sencilla para que la relación esté en 1FN no pueden existir grupos repetitivos.

Para ejemplificar como se representan gráficamente las relaciones en primera forma normal consideremos la relación alumno cursa materia cuyo diagrama E-R es el siguiente:

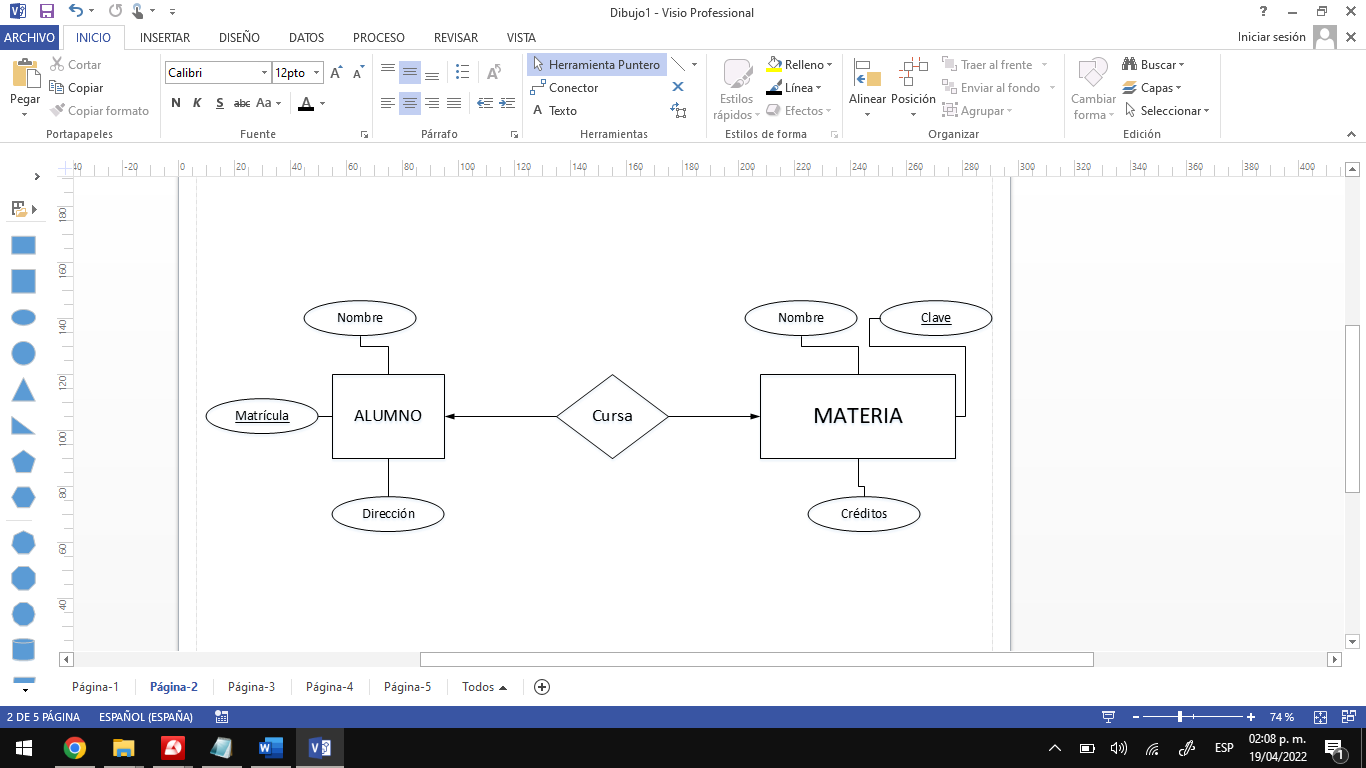
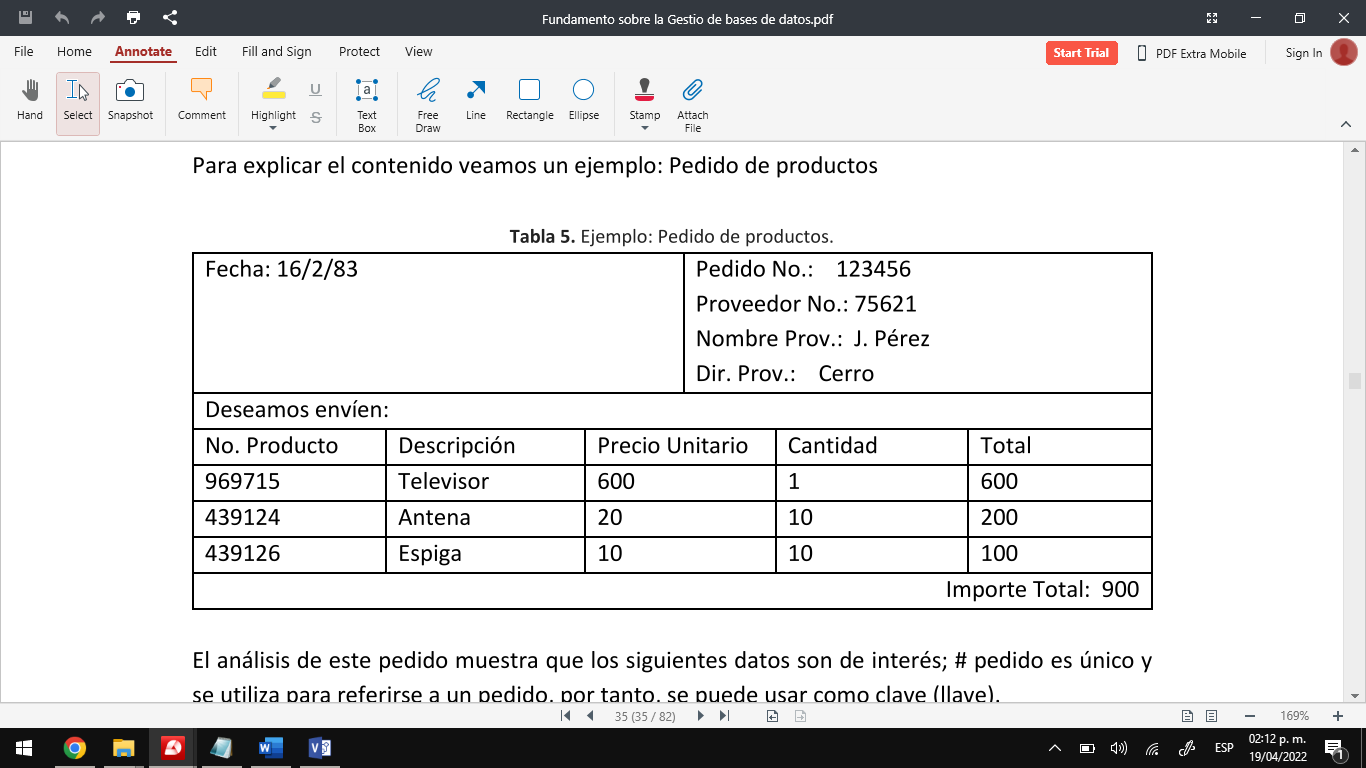


Ilustración . Primera Forma Normal

Como esta relación maneja valores atómicos, es decir un solo valor por cada uno de los campos que conforman a los atributos de las entidades, ya se encuentra en primera forma normal, gráficamente así representamos a las relaciones en 1FN.

Para explicar el contenido veamos un ejemplo: Pedido de productos

Tabla . Ejemplo pedido de productos



El análisis de este pedido muestra que los siguientes datos son de interés; # pedido es único y se utiliza para referirse a un pedido, por tanto, se puede usar como clave (llave).

En forma de relación se escribiría:

PEDIDO (nuped, fecha, nuprov, noprov, direc, nuprod, desc, prun, cant, prprod, prped)

Se puede observar que la relación PEDIDO contiene cinco grupos repetitivos: nuprod, desc, prun, cant, prprod, ya que un pedido puede contener más de una línea de pedido y, por lo tanto, puede contener varios números de producto (nuprod), varias descripciones de producto (desc), varios precios unitarios (prun), varias cantidades (cant) y varios precios por concepto del producto (prprod).

Hay que eliminar esos grupos repetitivos para que la relación esté en 1FN. Para ello se crea:

1. Una relación para los campos que sean únicos, es decir, se dejan en la relación original solo los atributos que no son repetitivos:

PEDIDO (nuped, fecha, nuprov, noprov, direc, prped)

1. Una relación para los grupos repetitivos, es decir, se extraen en una nueva relación los atributos repetitivos, además de la llave primaria de la relación original:

PED-PROD (nuped, nuprod, desc, prun, cant, prprod)

Ambos tienen como llave o parte de la llave a nuped. Pero en PED-PROD es necesaria la llave compuesta para identificar los productos individuales.

Ahora estas nuevas dos relaciones en 1FN modelan el fenómeno que nos ocupa. Los problemas de actualización mencionados anteriormente quedan resueltos con este nuevo modelo. En lugar de tener varios valores en cada campo de acuerdo a la cantidad de líneas de pedido, tal y como ocurría en la tabla PEDIDO original, se tienen varias ocurrencias en la tabla PED-PROD, una por cada producto que se solicita en el pedido. Esto permite que se soliciten tantos productos como se desee en cada pedido, pues solo significa agregar una nueva ocurrencia en la relación PED- PROD por cada producto solicitado.

Sin embargo, este modelo en 1FN tiene aún problemas de actualización, como se muestra en las siguientes operaciones:

**Creación**: la información sobre un nuevo producto no se puede insertar si no hay un pedido que lo incluya.

**Supresión**: eliminar una línea de pedido que sea la única que pida un producto implica perder la información del producto.

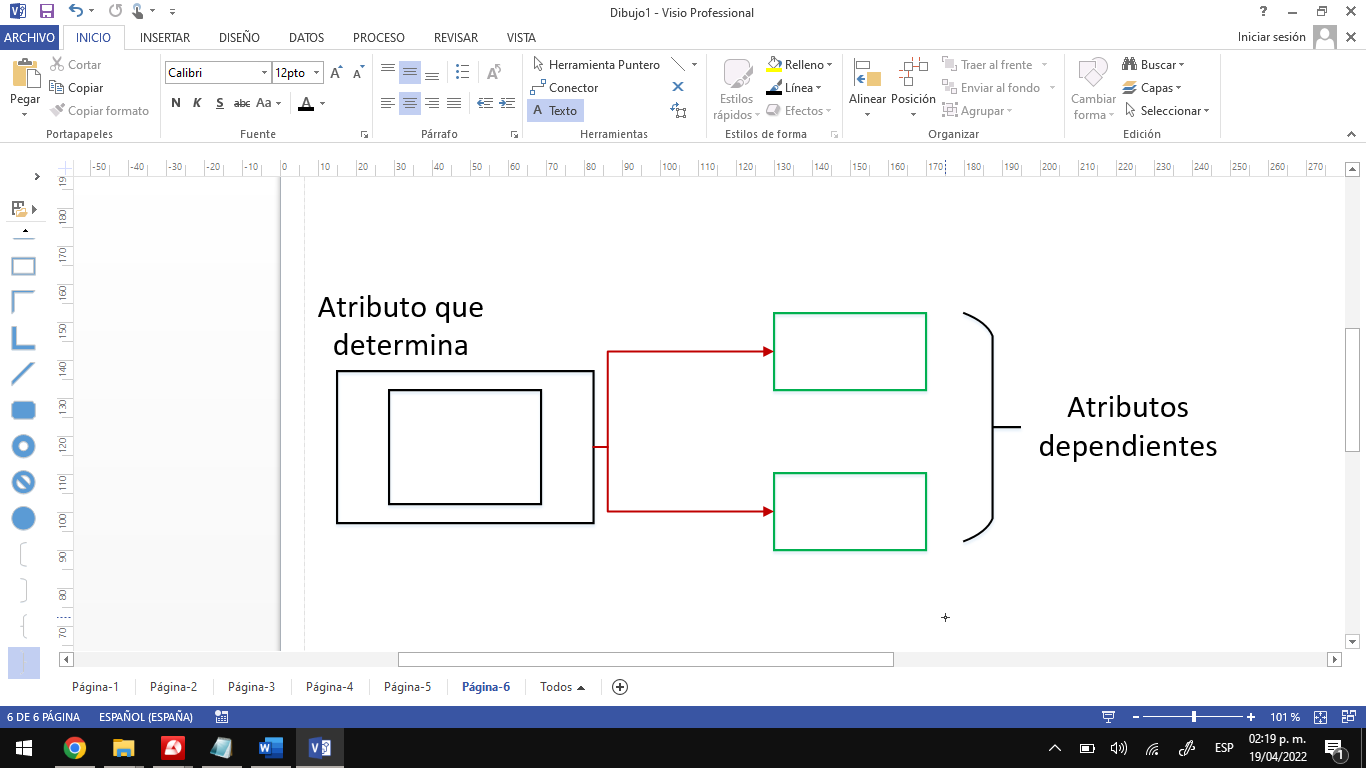
**Modificación**: por cada línea de pedido en la que se solicite determinado producto se tiene ocurrencia en PED-PROD, que repite la información sobre este. Si cambia algún atributo del producto, entonces es necesario hacer muchas actualizaciones.

Entonces será necesario aplicar formas normales más fuertes a este modelo para eliminar los problemas de actualización que presenta, como veremos a continuación.

**Segunda Forma Normal.**

Para definir formalmente la segunda forma normal requerimos saber que es una dependencia funcional.

DF: Consiste en identificar que atributos dependen de otro(s) atributo(s).



## Ejemplos resueltos del modelo relacional

FUENTES DE CONSULTA

# Bibliografía

Marqués, M. (2011). *Bases de Datos.* Universitat Jaume I.

Oppel, A. (2010). *Fundamentos de bases de datos.* México, D.F.: McGraw-Hill.

Pisco Gómez, Á., Regalado Jalca, J. J., Gutiérrez García, J., Quimis Sánchez, O., Marcillo Parrales, K., & Marcillo Merino, J. (2017). *Fundamentos sobre la gestión de base de datos.* Alicante: Área de Innovación y Desarrollo,S.L. .