@its_anaehm

Diagramas Entidad-Relación (ER)

Base de Datos: Una Base de Datos es una colección de información que está organizada para poder almacenar, gestionar, actualizar y acceder a los datos fácilmente.

Hay mucha información en movimiento en una base de datos y puede ser complicado llegar a entender cómo los elementos interactúan entre sí. Es por eso que la idea de tener una representación visual para entender fácilmente cómo todos estos elementos están relacionados y cómo trabajan en conjunto viene bien y es aquí donde entran los diagramas de entidad relación.

Conceptos Claves:

Antes de comenzar a realizar un diagrama entidad relación es conveniente entender mejor los componentes de estos mismos.

Entidad:

Una entidad es un objeto, como una persona, un lugar o una cosa, que va a ser gestionado en la base de datos.

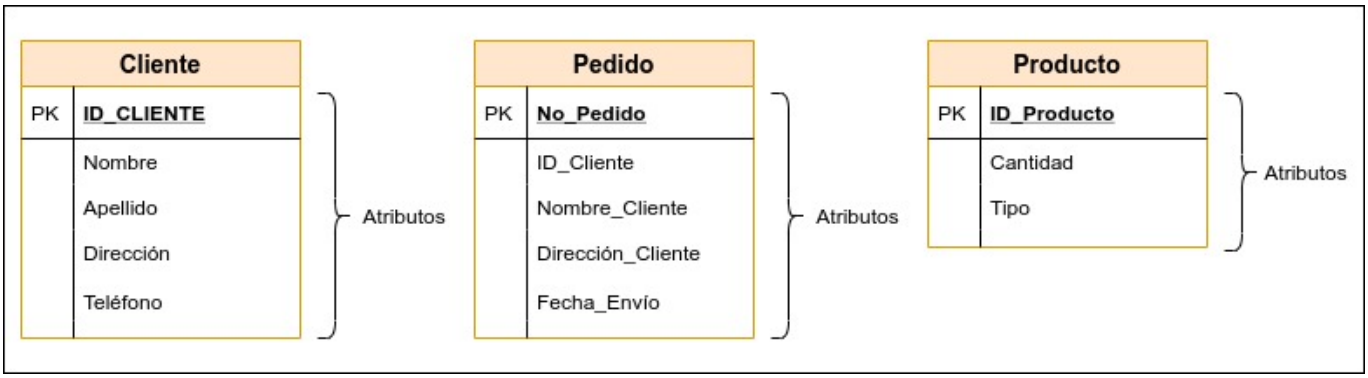
Por ejemplo: Si una persona desea realizar la compra de una laptop a través de Amazon, la entidad podría ser un cliente, un pedido o el mismo producto.

Atributos:

Cada entidad va a tener atributos y estos son las propiedades o características.

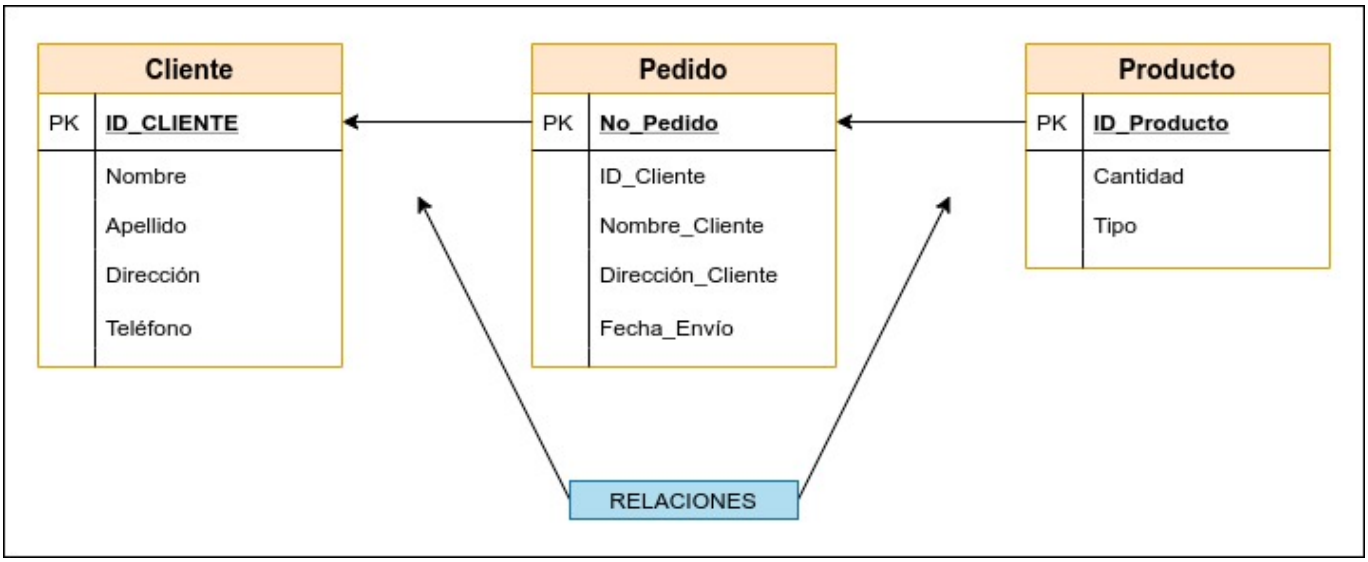
Continuando con ejemplo anterior en el caso de la entidad cliente, tenemos como posibles atributos el ID del cliente, nombre, Apellido, Dirección, No. de teléfono, etc.

NOTA: En una tabla de base de Datos las entidades son las filas y los atributos son las columnas.



Relaciones:

Las relaciones describen los vínculos o conexiones entre las entidades que definen cómo interactúan. Y en un diagrama de entidad-relación se representan a través de líneas entre ellas.



Cardinalidad:

Define la relación en un contexto numérico, por lo particular esto se hace en rangos (mínimos y máximos).

Tipos de Cardinalidad:

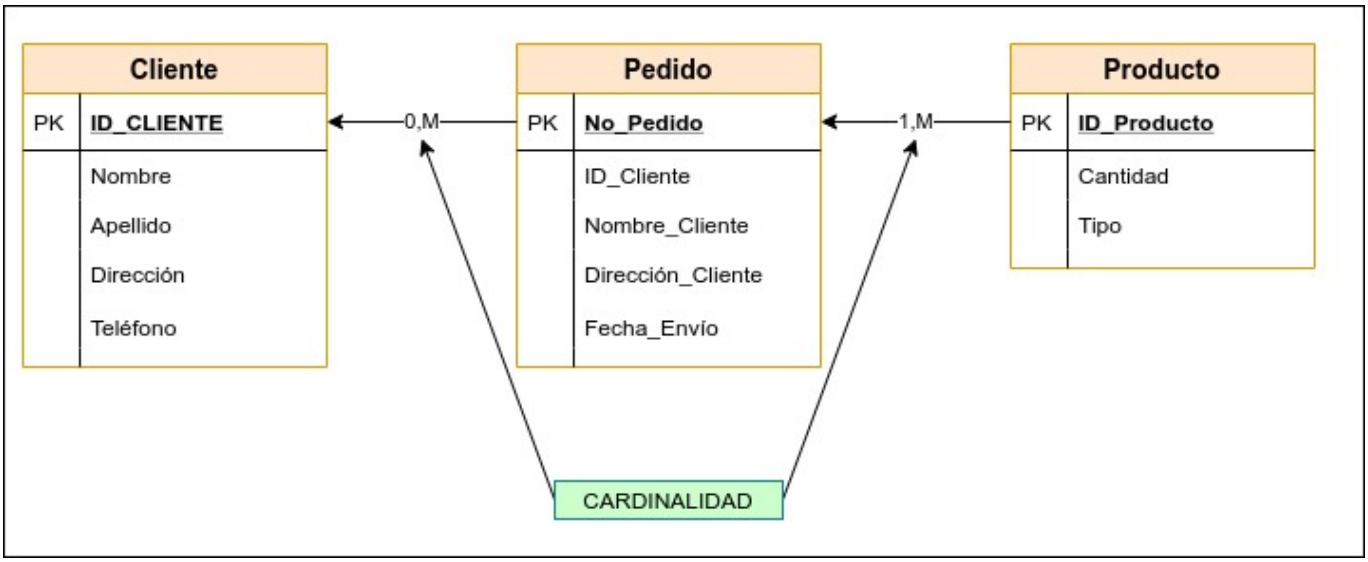
- Uno
- Muchos
- Uno y solo no
- Cero o uno
- Cero o muchos
- Uno o muchos

La mejor forma de colocar los rangos de cardinalidad es hacerlo pensando de forma lógica, pensando en cuál es el número mínimo de vínculos que pueden haber y cuál es el número máximo.

En el ejemplo de la compra por Amazon podemos pensar en el número mínimo de pedidos que puede hacer un cliente y cuál es el máximo, Puede haber un cliente, pero puede haber cero pedidos y si pensamos en el máximo de pedidos que puede hacer un cliente podemos deducir que son muchos por lo que en este caso hay una cardinalidad de *cero o muchos*.

Otro ejemplo de esto continuando con el anterior es estableciendo la relación entre pedido y producto, para que exista un pedido debe haber al menos un producto, pero puede de que en ese pedido haya muchos productos también, entonces la cardinalidad en este caso sería de *uno a muchos*

NOTA 2: Cada entidad en el diagrama representa una tabla en la base de datos.

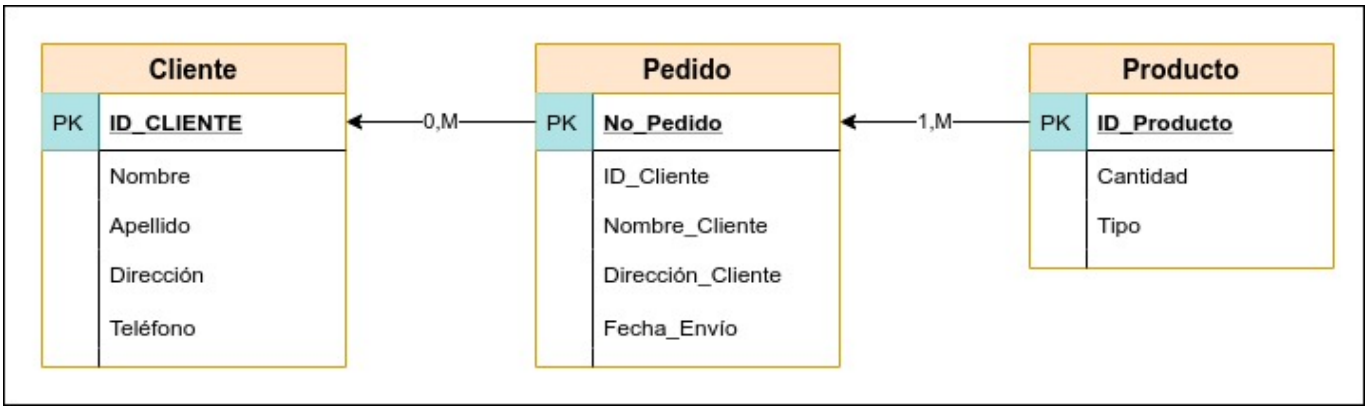


LLave Primarias (pk):

Es un atributo o campo que identifica de forma exclusiva a cada registro en una determinada tabla. Y dado que un atributo puede lograr todo esto es que sólo se necesita una llave primaria por entidad.

Reglas de LLaves Primarias:

- *Única:* Una llave primaria debe ser única de forma que identifique solo un registro en la tabla.
- *Inalterable:* Ya que de no serlo sería difícil mantener un registro preciso de datos que no son constantes.
- *Nunca Nula:* Esto significa que no hay ocasiones en las que el valor pueda dejarse en blanco.



Por ejemplo: Si trabajamos en una empresa de envíos como HUGO nos podemos encontrar con el problema de que hayan dos clientes con el mismo nombre y apellido por eso no podemos tomar estos como llaves primarias, tampoco la dirección porque puede haber dos clientes viviendo en el mismo lugar o puede mudarse alguno y esto hace que ya se altere el dato, lo mismo con el número de teléfono; por lo que la mejor opción es crear un atributo que se llame ID_Cliente este se programa para que nunca de un mismo ID a dos clientes y así nunca se repite.

Otro ejemplo de cómo se utilizan las llaves primarias es cuando queremos crear una cuenta en cualquier red social y no se nos es permitido porque el nombre de usuario con el cual nos queremos registrar ya está utilizado, esto puede suceder porque se está utilizando a los nombres de usuarios como llaves primarias en la base de datos y esta no se puede repetir.

Llaves Foráneas (FK)

Una llave foránea es una llave primaria en un lugar foráneo, esto quiere decir que cuando se tiene una llave primaria en una entidad pero es necesario o útil usar esta información en otra entidad es ahí donde se obtiene una llave foránea.

Siguiendo con el ejemplo de la empresa HUGO el atributo que se creó para identificar al cliente con su llave primaria, ese mismo atributo debe ser parte de la entidad de pedido porque para cada pedido que se registra se debe saber exactamente qué cliente realizó el pedido entonces la entidad de pedido hacer referencia a la entidad de cliente a través del atributo ID_Cliente eso lo vuelve en la tabla una llave foránea.

Características de las Llaves Foraneas:

- A diferencia de las llaves primarias, las llaves foráneas no tienen que ser únicas por lo que se pueden repetir en una tabla.

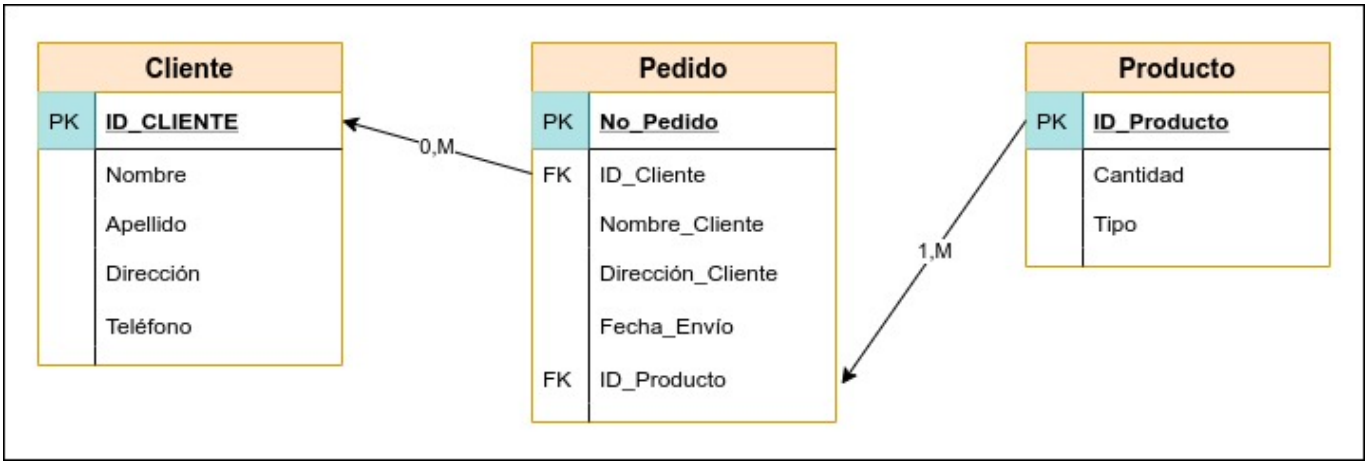
En la tabla de pedidos de la base de datos de la empresa HUGO el cliente xxx puede realizar varios pedidos.

- No puede existir una misma llave primaria para dos entidades.

No se puede utilizar el mismo tipo de identificador del cliente para identificar el producto, por el mismo motivo anterior de que un cliente puede realizar varios pedidos.

- Pueden haber múltiples llaves foráneas en una entidad.

Digamos que para cada pedido en la base de datos también se quiere saber qué producto está solicitando el cliente entonces esto haría de que en la tabla de pedidos también se necesite el ID del producto.



Llave Primaria Compuesta:

Una llave primaria compuesta es cuando se requieren dos o más atributos para identificar un registro en una tabla. Estas se utilizan cuando no se puede confiar en un solo atributo para darnos un registro único.

Si tenemos una tabla de envíos podemos ver de que en este caso el ID_Producto se puede duplicar si dos personas compran el mismo objeto, un pedido podría convertirse en un par de envíos diferentes así que el número de pedido tampoco es único, si dos personas hacen un pedido a la misma hora tampoco podría ser única así pasa también con la hora de empaquetado y la fecha de envío. No podemos confiar en ninguno de estos atributos para que por sí solos nos den un identificador único para la tabla; pero si se toman dos atributos para combinarlos y crear un atributo único nuevo como el ID_Producto y el número de pedido, si estos se unen el valor que crean ambos nunca se repetirá.

Nota 4: Las llaves foráneas pueden ser parte de una llave primaria compuesta.

Reglas de Llaves Primarias Compuestas:

- Usar la menor cantidad posible de atributos.
- No usar atributos que tiendan a cambiar.

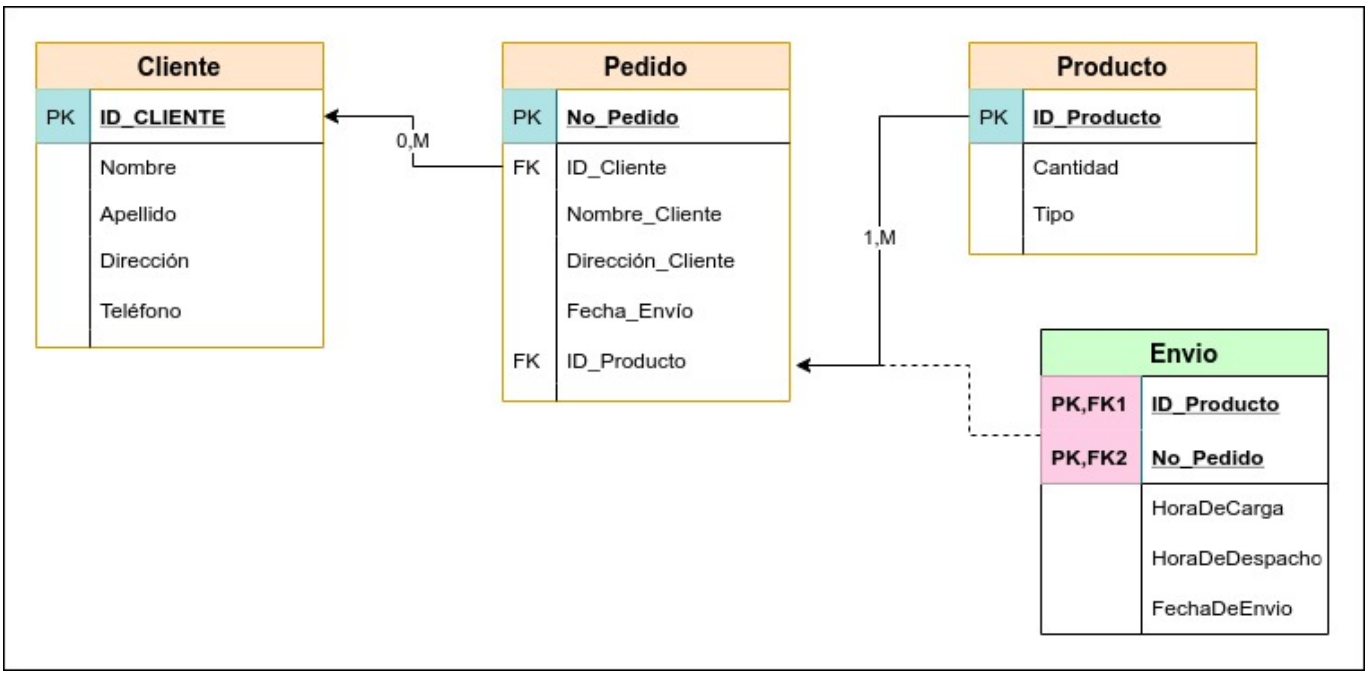
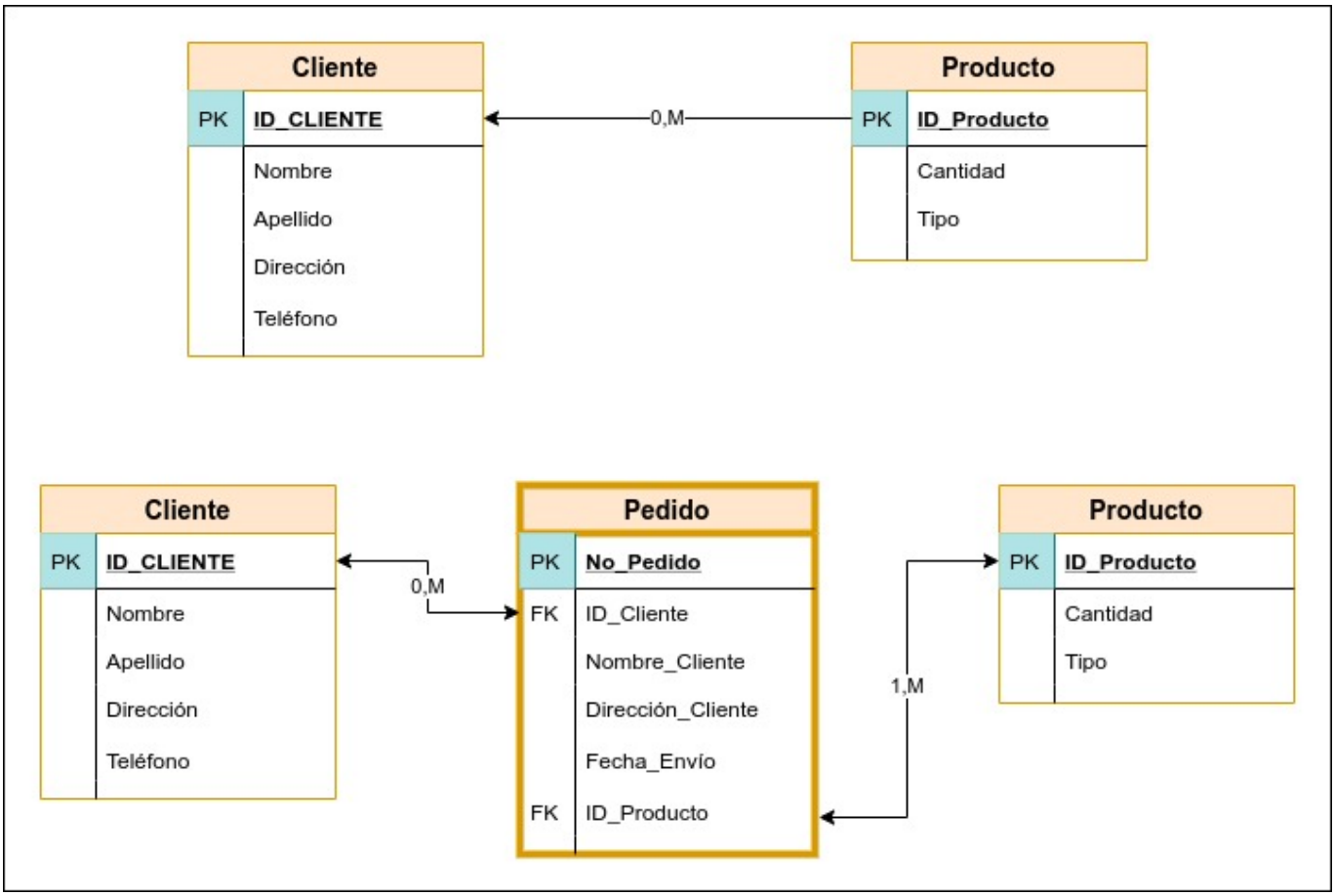


Tabla Puente

Cuando estamos creando un diagrama entidad relación es importante considerar si todo lo que tenemos es lo que debemos registrar en la base de datos, ya que se pueden presentar escenarios en los que dos entidades están conectadas entre sí pero pasa algo más y eso no lo estamos representando, aquí es donde se utilizan las tablas puente.

Si para el caso anterior solo tuviésemos las entidades cliente y producto, si podemos establecer una relación ya que un cliente puede comprar cero o muchos productos y cierto producto puede llegar a cero o más clientes, pero de esta forma no se sabrán cosas como la fecha de compra del producto, si se compraron todos a la vez o si fue en varios pedidos. Nos encontraremos a oscuras sobre muchos detalles con respecto a la interacción de las dos entidades. Por lo que para este ejemplo quien actúa como entidad puente es la entidad de pedido ya que nos brinda la información que nos hace falta.



Tipos de Datos:

Un buen diagrama de entidad relación refleja la mayor cantidad de información necesaria para el entendimiento de cómo trabaja una base de datos por lo que en la mayoría de los escenarios es importante reflejar el tipo de dato que debe tener cada atributo en la entidad para ayudarnos en la programación a realizar fuera del diagrama.

- **INT:** Número entero.
- **VARCHAR:** Texto de longitud variable utilizando diversos caracteres.
- **BINARY:** Para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
- **BIT:** Valores Si/No ó True/False.
- **BYTE:** Un valor entero entre 0 y 255.
- **COUNTER:** Un número incrementado automáticamente (de tipo Long).
- **DATETIME:** Un valor de fecha u hora entre los años 100 y 9999.
- **DOUBLE:** Un valor en punto flotante de doble precisión.
- **LONG:** Un entero largo.
- **TEXT:** De cero a 255 caracteres.
- **SMALLINT:** Número entero con o sin signo.
- **TINYINT:** Es el tipo de datos numérico más chico dentro del lenguaje SQL, y su utilización siendo escasa para la mayoría de las operaciones, se debe a que no puede almacenar números negativos, solamente del 0 al 255.
- **BIGINT:** Es el tipo de dato numérico más grande dentro de SQL, se encuentra incluído en los denominados números exactos y posee un tamaño de almacenamiento de 8 bytes o bien 64 bits, ocupando el doble de lo que ocupa el tipo int.

- **TIMESTAMP:** Fecha y hora es un tipo de datos. Timestamp es un método para el control de versiones de filas. De hecho, en el server sql 2008, este tipo de columna se renombró (es decir, la timestamp está en desuso) a rowversion. Básicamente significa que cada vez que se cambia una fila, este valor aumenta.