* Índice:

[1. BD ventas 1](#_Toc127522784)

[**1.1** Solución 1 2](#_Toc127522785)

[**1.2** Solución 2 5](#_Toc127522786)

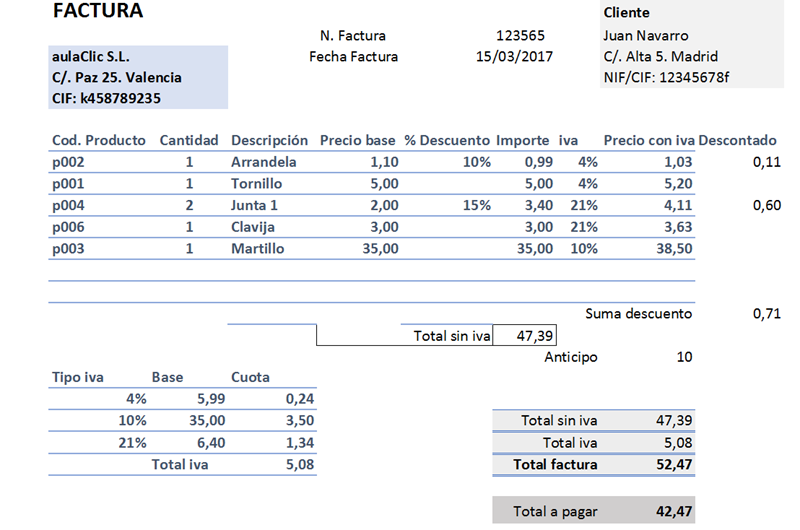
[**1.3** Soluciones: Otras soluciones OBSOLETAS 6](#_Toc127522787)

# BD ventas

Crear el modelo ER para este caso:

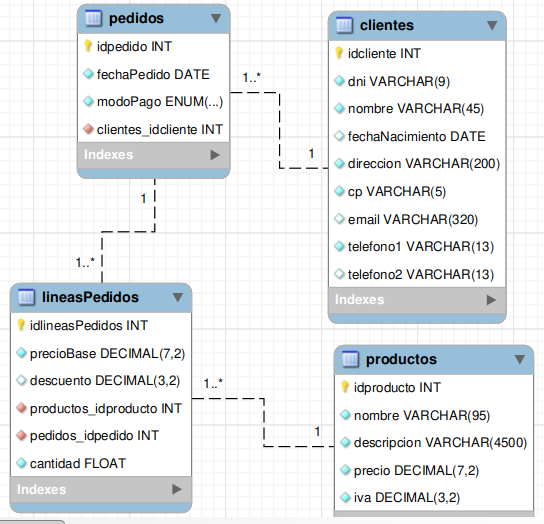
1. Se guarda información de los clientes, productos y pedidos que realizan los clientes.
2. De cada cliente: IdCliente, DNI, nombre, fecha de nacimiento (opcional), dirección, ciudad, provincia, código postal, email, y hasta dos teléfonos (uno de ellos es obligatorio).
3. Cada pedido se corresponde con un solo cliente.
4. Un pedido puede incluir varios productos (y de cada producto que se vende en ese pedido hay que almacenar cuantas unidades se venden en ese pedido), por defecto se vende una unidad de producto (el sistema guarda una unidad si se deja en blanco este campo).
5. No sólo hay que conocer el precio actual de cada producto sino también a qué precio fue vendido el producto en cada pedido.
6. En cada pedido se incluye el precio al que se ha vendido el producto, y que porcentaje de descuento se aplicó respecto al precio de venta de ese producto (opcional); además guarda la fecha del pedido y el modo de pago (contado, tarjeta..., que puede estar en blanco si todavía no se ha pagado), la fecha de pago (puede ser nula si todavía no se ha pagado). Por último, en cada pedido se puede almacenar el documento en formato *pdf* con la factura.
7. De cada producto se conoce el precio de venta actual, el nombre, descripción y el porcentaja de iva que se aplica a ese producto (no todos los productos tienen el mismo porcentaje de iva).
8. Si se borra un pedido desaparecerá toda la información relacionada con ese pedido.

Aunque no se corresponde exactamente con nuestro ejercicio, esta es aproximadamente la información que se mostraría al imprimir una factura. Se muestra para que se entienda que información global se guarda en una Base de Datos, hay que tener en cuenta que una factura muestra información de varias tablas.



## Solución 1





-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema ventas

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `ventas` ;

-- -----------------------------------------------------

-- Schema ventas

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `ventas` ;

USE `ventas` ;

-- -----------------------------------------------------

-- Table `clientes`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `clientes` ;

CREATE TABLE IF NOT EXISTS `clientes` (

`idcliente` INT NOT NULL AUTO\_INCREMENT,

`dni` VARCHAR(9) NOT NULL,

`nombre` VARCHAR(45) NOT NULL,

`fechaNacimiento` DATE NULL,

`direccion` VARCHAR(200) NOT NULL,

`cp` VARCHAR(5) NOT NULL,

`email` VARCHAR(320) NULL,

`telefono1` VARCHAR(13) NOT NULL,

`telefono2` VARCHAR(13) NULL,

PRIMARY KEY (`idcliente`),

UNIQUE INDEX `dni\_UNIQUE` (`dni` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `pedidos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `pedidos` ;

CREATE TABLE IF NOT EXISTS `pedidos` (

`idpedido` INT NOT NULL AUTO\_INCREMENT,

`fechaPedido` DATE NOT NULL,

`modoPago` ENUM('tarjeta', 'efectivo', 'paypal') NOT NULL,

`clientes\_idcliente` INT NOT NULL,

PRIMARY KEY (`idpedido`),

INDEX `fk\_pedidos\_clientes\_idx` (`clientes\_idcliente` ASC) VISIBLE,

CONSTRAINT `fk\_pedidos\_clientes`

FOREIGN KEY (`clientes\_idcliente`)

REFERENCES `clientes` (`idcliente`)

ON DELETE NO ACTION

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `productos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `productos` ;

CREATE TABLE IF NOT EXISTS `productos` (

`idproducto` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(95) NOT NULL,

`descripcion` VARCHAR(4500) NOT NULL,

`precio` DECIMAL(7,2) NOT NULL,

`iva` DECIMAL(3,2) NOT NULL,

PRIMARY KEY (`idproducto`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `lineasPedidos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `lineasPedidos` ;

CREATE TABLE IF NOT EXISTS `lineasPedidos` (

`idlineasPedidos` INT NOT NULL,

`precioBase` DECIMAL(7,2) NOT NULL,

`descuento` DECIMAL(3,2) NULL,

`productos\_idproducto` INT NOT NULL,

`pedidos\_idpedido` INT NOT NULL,

`cantidad` FLOAT NOT NULL,

PRIMARY KEY (`idlineasPedidos`),

INDEX `fk\_lineasPedidos\_productos1\_idx` (`productos\_idproducto` ASC) VISIBLE,

INDEX `fk\_lineasPedidos\_pedidos1\_idx` (`pedidos\_idpedido` ASC) VISIBLE,

CONSTRAINT `fk\_lineasPedidos\_productos1`

FOREIGN KEY (`productos\_idproducto`)

REFERENCES `productos` (`idproducto`)

ON DELETE NO ACTION

ON UPDATE CASCADE,

CONSTRAINT `fk\_lineasPedidos\_pedidos1`

FOREIGN KEY (`pedidos\_idpedido`)

REFERENCES `pedidos` (`idpedido`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

## Solución 2

Modificar la BD de la solución 1 para que cumpla este nuevo modelo ER:



Este es el código que sin borrar la BD anterior modifica lo necesario:

USE ventas ;

CREATE TABLE IF NOT EXISTS TiposDeProductos (

idTipoProducto INT NOT NULL AUTO\_INCREMENT,

descripcionTipo VARCHAR(100) NOT NULL,

iva DECIMAL(3,2),

PRIMARY KEY (idTipoProducto));

#añadimos a la tabla existente el atributo que será FK:

ALTER TABLE productos ADD COLUMN idTipoProducto INT NOT NULL;

#modificamos la tabla existente añadiendo la restricción de FOREIGN KEY:

ALTER TABLE productos ADD CONSTRAINT fk\_productos\_idTipoProducto FOREIGN KEY (idTipoProducto) REFERENCES TiposDeProductos(idTipoProducto);

## Soluciones: Otras soluciones OBSOLETAS

Las dos primeras soluciones son muy parecidas, de hecho, al pasarlas a tablas producen las mismas tablas, salvo un detalle:

* La tabla ***LÍNEA*** del primer caso (a) incluye como PK (idLínea), que es auto numérico, y si deseamos que un pedido no pueda tener 2 líneas con el mismo producto habrá que incluir un UNIQUE (idPedido, idArtículo).
* La solución b es muy parecida a la solución a pero la entidad Linea es débil por identificación.
* La tabla ***INCLUYE*** de la solución c tendrá como PK (idPedido, idArtículo), pero de esta manera no permite que un artículo aparezca varias veces en un pedido (comportamiento típico en una compra por internet). Si quisiéramos permitir que un artículo aparezca varias veces en un pedido (en líneas distintas), tal y como sucede en un supermercado, entonces tendríamos que añadir algún atributo a la tabla ***INCLUYE*** para permitirlo, y a fin de cuentas sería preferible seleccionar la primera solución presentada.

Las dos primeras soluciones son correctas y la tercera es deficiente. La primera solución es preferible a la segunda porque:

* + La tabla Línea de pedidos puede realizar su clave primaria con un atributo auto numérico. La segunda solución requiere que la la clave primaria esté formada por dos atributos y además hay que escribir un programa para que los números de línea de los nuevos pedidos empiecen de nuevo con el valor 1.

Las dos primeras soluciones utilizan entidad débil que indica que si borramos un pedido deben borrarse las líneas asociadas a ese pedido (borrado en cascada) y así es como debe ser.

La tercera solución es deficiente y no debería usarse porque:

* + o hay una entidad débil que aclare ya en el diagrama ER que si se borra un pedido deben borrarse en cascada las líneas (incluyes) asociados.
  + No permite utilizar un atributo auto numérico para distinguir las líneas (incluyes) y requiere que la la clave primaria esté formada por dos atributos y hay que realizar un programa para que los números de línea de los nuevos pedidos empiecen de nuevo en uno







