# TRABAJO PRÁCTICO SEGUNDO CUATRIMESTRE 2022

## Gestión Bazaar

Grupo N°34 - Código Enigma
Gestión de Datos
Curso K3751


| Nombre y Apellido | Legajo |
|---|---|
| Flores Maquera, Juan Alberto | 152234-6 |
| Berro, Camila Andrea | 172341-8 |
| Argonz, Matias | 173116-6 |

# CONFORMACIÓN GRUPO

En esta sección tenemos la intención de comunicar que uno de nuestros compañeros, **Matias Obezzi**, no participó en ninguna instancia del TP, y es por esa razón es que no se encuentra anotado en la carátula.

# MIGRACIÓN

## Resumen

El trabajo práctico consiste en implementar un nuevo sistema que se encargue de la gestión de un negocio que vende artículos de un Bazar. Se registra la información de ventas y compras de productos, información de compradores y proveedores, y a su vez información de productos en específico.

En algunas tablas, se propuso usar un id autoincremental para que sea más sencillo identificar la entidad. Por ejemplo, en la tabla Proveedor se usó este tipo de PK ya que sino se tendría que identificar usando 2 pks diferentes(proveedor_razon_social, proveedor_cuit).

Las funciones creadas que se usan en migración son de utilidad para obtener los id correspondientes a través de los campos conocidos que fueron provistos por la cátedra en la tabla maestra.

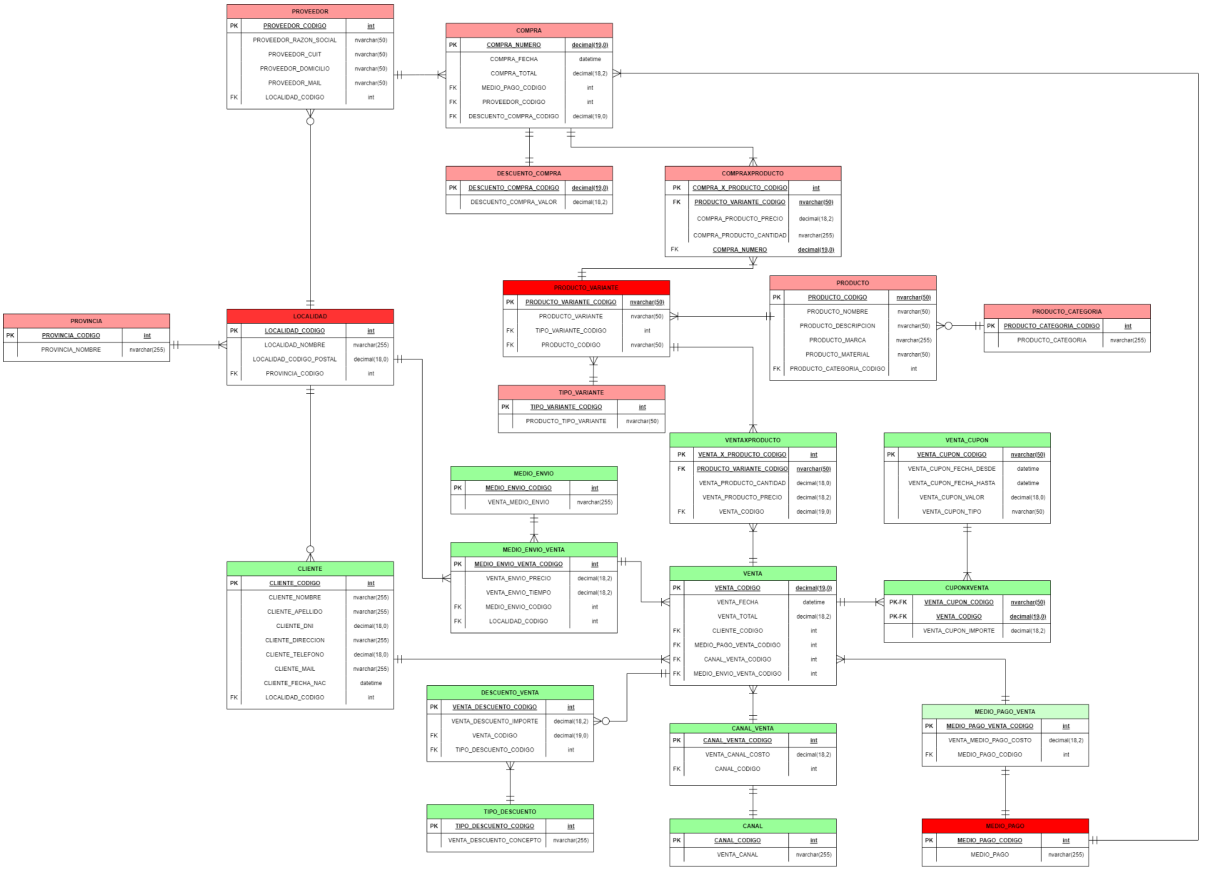
Se decidió usar una tabla llamada CompraXProducto como intermedia de la tabla compra y producto_variante (se hizo lo mismo en VentaXProducto para producto_variante y venta) para que exista la relación entre las dos tablas. En ambas tablas se decidió usar un pk autoincremental ya que el código de la compra/venta de un producto y el producto variante código se encontraba repetido en un dato, con la diferencia que la cantidad comprada era diferente.

Se decidió para el esquema del modelo, crear las tablas correspondientes por un lado y por otro las PKs y Fks de ellas. De esta manera, no dependemos del orden en el que se ejecuta la creación de las tablas.

Por último, la duración promedio de la migración es de aproximadamente 3 minutos y 8 segundos, dependiendo de la computadora usada.

Grupo N°34 - Código Enigma
Gestión de Datos
Curso K3752

# DER



# TABLAS

## Provincia

La tabla PROVINCIA se usa para registrar las provincias de los clientes como también de los proveedores. Se identifica con un PK autoincremental (agregada por nosotros) para garantizar el acceso unívoco a cada registro. La tabla PROVINCIA se relaciona con la tabla LOCALIDAD. Una localidad necesariamente tiene una PROVINCIA asociada. La relación es de uno a muchos, ya que varias LOCALIDADES pueden compartir la misma PROVINCIA.

## Localidad

La tabla LOCALIDAD se usa para registrar las localidades de los clientes, proveedores y, además, para el medio de envío de una venta. Se identifica mediante una PK autoincremental (agregada por nosotros) para garantizar el acceso unívoco a cada registro. La tabla LOCALIDAD se relaciona con CLIENTE y PROVEEDOR, donde ambas relaciones son de uno a muchos.También, se relaciona con MEDIO_ENVIO_VENTA, donde un medio de envío puede tener como atributo una localidad y una localidad puede tener varios medios de envío. La tabla posee, además, una FK que la relaciona con la tabla PROVINCIA.

## Proveedor

La tabla PROVEEDOR es usada para registrar cada proveedor. Se identifica mediante una PK autoincremental (agregada por nosotros) para garantizar el acceso unívoco a cada registro. La tabla se relaciona con LOCALIDAD, teniendo solo una localidad por proveedor. También, se relaciona con la tabla COMPRA. Esta última tiene una relación de uno a muchos, donde cada compra tiene un único proveedor pero un proveedor puede tener varias compras. Por último, mantiene una relación con la tabla LOCALIDAD.

## Compra

La tabla COMPRA se usa para registrar las compras existentes en el sistema. Se identifica con el número de compra (compra_numero) que ya venía cargada en la tabla maestra. La tabla posee las siguientes FK: medio_pago_codigo, proveedor_codigo y descuento_compra_codigo. La tabla se relaciona con PROVEEDOR, DESCUENTO_COMPRA, COMPRAXPRODUCTO y MEDIO_PAGO. Proveedor, como ya se ha mencionado, tiene una relación de muchos a uno, teniendo un proveedor por compra pero varias compras por proveedor. Descuento_Compra se relaciona uno a uno ya que por cada compra existe un único descuento, y por cada descuento, existe una única compra. Medio_Pago se relaciona de muchos a uno, debido a que por cada compra existe un único medio de pago pero cada medio de pago puede ser usado en diferentes compras. Por último, CompraXProducto se relaciona de muchos a uno debido a que ésta es una tabla intermedia entre compra y producto_variante, donde un producto_variante puede existir en varias compras y una compra puede tener varios producto_variante.

## Descuento_compra

La tabla DESCUENTO_COMPRA la usamos para registrar los descuentos que existen por cada compra realizada. Se identifica mediante Descuento_Compra_Codigo que ya venía cargada en la tabla maestra y se usó como PK de la tabla. Esta tabla se relaciona uno a uno con la tabla COMPRA, donde por cada compra existe un único descuento y por cada descuento existe una única compra.

## Medio_Pago

La tabla MEDIO_PAGO es usada para registrar todos los medios de pagos existentes en el sistema. Se identifica mediante un PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. Como la compra y venta comparten los mismos medios de pago, se decidió unificar este dato en el atributo Medio_Pago. Se relaciona uno a uno con la tabla MEDIO_PAGO_VENTA, debido a que se decidió separar los atributos medio_pago de la venta_medio_pago_costo. Esto se produjo a causa de que el costo no se utiliza en la Compra pero sí en la Venta de productos. Para cada medio de pago solo existe un costo y, por cada costo existe un medio pago. Por último, MEDIO_PAGO se relaciona con la tabla COMPRA, teniendo una relación de uno a muchos. Esto es porque por cada compra solo existe un medio de pago pero un medio de pago puede ser usado en varias compras.

## CompraXProducto

Esta tabla se usa para registrar las variantes de los productos vendidos en cada compra. Es una tabla intermedia entre PRODUCTO_VARIANTE y la tabla COMPRA. Por esto, en un principio se pensó utilizar como pk-fk compra_numero y producto_variante_codigo. Sin embargo, al plantearlo de esta manera nos dimos cuenta que estas pk-fks se encontraban repetidas en la tabla maestra, donde la única diferencia era el atributo cantidad de producto comprado. Por lo tanto, se decidió utilizar una PK autoincremental (creada por nosotros) para solucionar este problema. La tabla se relaciona de uno a muchos, tanto con COMPRA como con PRODUCTO_VARIANTE. Se decidió, además, relacionar esta tabla con Producto_Variante, en vez de Producto, debido a la composición de Producto_Variante, que es la combinación del producto con su variante, otorgándole la clasificación adecuada de la compra.

## Producto_Variante

Esta tabla se usa para registrar todas las posibles combinaciones existentes entre la tabla VARIANTE y PRODUCTO. Por lo tanto, tiene como FK Tipo_variante_código y Producto_codigo. Se utiliza producto_variante_codigo como PK, atributo que venía cargado en la tabla maestra.

Esta tabla se relaciona con COMPRAXPRODUCTO, de uno a muchos, debido a que solo existe un producto_variante por CompraXProducto pero un producto_variante puede existir en varias CompraXProducto. Por otro lado, también se relaciona con la tabla TIPO_VARIANTE, siendo una relación de muchos a uno. Esto es debido a que un producto_variante sólo posee un tipo_variante pero un tipo_variante puede existir en muchos producto_variantes. La tabla PRODUCTO_VARIANTE presenta otra relación de uno a muchos con VENTAXPRODUCTO, debido a que un producto_variante puede existir

en varias ventas pero una venta solo posee un producto_variante. Por último, tiene una relación de muchos a uno con PRODUCTO, donde un producto puede estar presente en muchos producto_variantes pero un producto_variante sólo tiene un producto.

## Tipo_variante

Esta tabla se utiliza para detallar los distintos tipos de variantes que existen en el sistema. Posee una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro y además, posee el atributo Producto_Tipo_Variante. Esta tabla se relaciona con PRODUCTO_VARIANTE de uno a muchos debido a que un tipo_variante puede existir en varias variantes del producto pero una variante del producto solo posee un tipo_variante.

## Producto

Esta tabla se utiliza para detallar los productos del sistema. Se utiliza, como PK, producto_codigo, un atributo que ya existía en el sistema, para identificar la entidad. Posee además la FK Producto_Categoria_Codigo, identificando al tipo de categoría a la que pertenece el producto. Esta tabla se relaciona de uno a muchos con PRODUCTO_VARIANTE, debido a que un producto puede tener muchas variantes pero una variante solo tiene un producto. Además, se relaciona de muchos a uno con PRODUCTO_CATEGORIA, donde cada producto tiene una categoría diferente pero una categoría puede estar presente en diferentes productos.

## Producto_Categoria

Esta tabla es utilizada para identificar los distintos tipos de categorías que un producto puede tener. Esta utiliza una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. PRODUCTO_CATEGORIA se relaciona con PRODUCTO de uno a muchos, donde un producto siempre tiene una categoría pero una categoría puede estar presente en diferentes productos.

## VentaXProducto

Esta tabla es utilizada para guardar las distintas ventas por producto. Se trata de una tabla intermedia, que relaciona a una venta con la variante de un producto. Se utiliza una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. VENTAxPRODUCTO tiene una relación de muchos a uno con PRODUCTO_VARIANTE y de muchos a uno con la tabla VENTA. Al igual que con COMPRAxPRODUCTO, fue necesario crear una PK ya que utilizando la PK compuesta de producto_variante_codigo y venta_codigo se producía un error en la migración ya que no permitía identificar la información de manera unívoca.

## Venta

Esta tabla es utilizada para guardar las ventas del sistema. Se utiliza como PK el atributo venta_codigo ya proporcionado por la tabla maestra. VENTA tiene una relación de uno a muchos con VENTAXPRODUCTO, de uno a muchos con CUPONxVENTA, de muchos a uno con MEDIO_PAGO_VENTA, de muchos a uno con CANAL_VENTA, de uno a

muchos con DESCUENTO_VENTA, de muchos a uno con CLIENTE, de muchos a uno con LOCALIDAD.

## CuponXVenta

Esta tabla es utilizada para guardar los cupones utilizados en una venta. Se trata de una tabla intermedia, que relaciona a una venta con sus respectivos cupones. Se utiliza, como PK, a venta_cupon_codigo y venta_codigo, que además son las FK que relaciona a CupoXVenta con las tablas VENTA_CUPON y VENTA respectivamente. Estos atributos son sacados de la tabla maestra.

## Venta_Cupon

Esta tabla es utilizada para guardar los cupones en el sistema. El atributo venta_cupon_codigo es la PK de la tabla, la cual es sacada de la tabla maestra provista. La tabla Venta_Cupon se relaciona con CuponXVenta, sabiendo que un cupón puede estar en muchos cupones por venta y que un cupón por venta únicamente tiene un cupón.

## Medio_Envio_Venta

Esta tabla es utilizada para guardar los medios de envío de una venta. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. MEDIO_ENVIO_VENTA se relaciona de uno a muchos con la tabla VENTA y de muchos a uno con la tabla LOCALIDAD.

## Medio_Envio

Esta tabla es utilizada para guardar los medios de envío existentes en el sistema. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. MEDIO_ENVIO se relaciona de uno a muchos con la tabla MEDIO_ENVIO_VENTA.

## Cliente

Esta tabla es utilizada para guardar los clientes existentes en el sistema. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. CLIENTE se relaciona de muchos a uno con la tabla LOCALIDAD y de uno a muchos con la tabla VENTA.

## Descuento_Venta

Esta tabla es utilizada para guardar los descuentos asociados a una venta. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. DESCUENTO_VENTA se relaciona de muchos a uno con la tabla VENTA y de muchos a uno con la tabla TIPO_DESCUENTO.

### Tipo_Descuento

Esta tabla es utilizada para guardar los tipos de descuentos existentes en el sistema. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. TIPO_DESCUENTO se relaciona de uno a muchos con la tabla DESCUENTO_VENTA.

### Canal_Venta

Esta tabla es utilizada para guardar los canales asociados a una venta. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. CANAL_VENTA se relaciona de uno a muchos con la tabla VENTA y de uno a uno con la tabla CANAL.

### Canal

Esta tabla es utilizada para guardar los canales existentes en el sistema. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. CANAL se relaciona de uno a uno con la tabla CANAL_VENTA.

### Medio_Pago_Venta

Esta tabla es utilizada para guardar los medios de pago asociados a una venta. Utilizamos una PK autoincremental (creada por nosotros) para garantizar el acceso unívoco a cada registro. MEDIO_PAGO_VENTA se relaciona de uno a uno con la tabla MEDIO_PAGO y de uno a muchos con la tabla VENTA.

## PROCEDURES

ACLARACIÓN: Las claves autoincrementales no se tienen que obtener. Además, se destaca la importancia de, en los procesos de migración, mantener la integridad referencial cargando de una manera correcta las FK.

### Migrar_Provincia

Procedure encargado de completar la tabla PROVINCIA, utilizando los datos provistos por la tabla maestra. Realiza un union para unir los datos provenientes de las provincias tanto de CLIENTE como de PROVEEDOR.

### Migrar_Localidades

Procedure encargado de completar la tabla LOCALIDAD, utilizando los datos provistos por la tabla maestra, utilizando la función OBTENER_PROVINCIA_CODIGO para completar la FK PROVINCIA_CODIGO. Realiza un union para unir los datos de las localidades tanto de CLIENTE como de PROVEEDOR.

## Migrar_Proveedores

Procedure encargado de completar la tabla PROVEEDOR, utilizando los datos provistos por la tabla maestra, utilizando la función OBTENER_LOCALIDAD_CODIGO para completar la FK LOCALIDAD_CODIGO.

## Migrar_Descuento_Compras

Procedure encargado de completar la tabla DESCUENTO_COMPRA, utilizando los datos provistos por la tabla maestra.

## Migrar_Medio_Pagos

Procedure encargado de completar la tabla MEDIO_PAGO, utilizando los datos provistos por la tabla maestra.

## Migrar_Tipo_Variantes

Procedure encargado de completar la tabla TIPO_VARIANTE, utilizando los datos provistos por la tabla maestra.

## Migrar_Producto_Categorias

Procedure encargado de completar la tabla PRODUCTO_CATEGORIA, utilizando los datos provistos por la tabla maestra.

## Migrar_Productos

Procedure encargado de completar la tabla PRODUCTO, utilizando los datos provistos por la tabla maestra, utilizando la función OBTENER_PRODUCTO_CATEGORIA_CODIGO para completar la FK PRODUCTO_CATEGORIA_CODIGO.

## Migrar_Producto_Variantes

Procedure encargado de completar la tabla PRODUCTO_VARIANTE, utilizando los datos provistos por la tabla maestra, utilizando la función OBTENER_TIPO_VARIANTE_CODIGO para completar la FK TIPO_VARIANTE_CODIGO.

## Migrar_Compras

Procedure encargado de completar la tabla COMPRA, utilizando los datos provistos por la tabla maestra, utilizando la función OBTENER_PROVEEDOR_CODIGO, OBTENER_DESCUENTO_COMPRA y OBTENER_MEDIO_PAGO_CODIGO, con el objetivo de completar la FKs PROVEEDOR_CODIGO, DESCUENTO_COMPRA_CODIGO y MEDIO_PAGO_CODIGO respectivamente.

## Migrar_CompraxProductos

Procedure encargado de completar la tabla COMPRAxPRODUCTO, utilizando los datos provistos por la tabla maestra.

## Migrar_Clientes

Procedure encargado de completar la tabla CLIENTE, utilizando los datos provistos por la tabla maestra, utilizando la función OBTENER_LOCALIDAD_CODIGO para completar la FK LOCALIDAD_CODIGO.

## Migrar_VentaXProducto

Procedure encargado de completar la tabla VENTAXPRODUCTO, utilizando los datos provistos por la tabla maestra.

## Migrar_Venta

Procedure encargado de completar la tabla VENTA, utilizando los datos provistos por la tabla maestra y las funciones OBTENER_CLIENTE_CODIGO para completar la FK CLIENTE_CODIGO, OBTENER_MEDIO_PAGO_VENTA_CODIGO para completar la FK MEDIO_PAGO_VENTA_CODIGO, OBTENER_CANAL_VENTA_CODIGO para completar la FK CANAL_VENTA_CODIGO, OBTENER_MEDIO_ENVIO_VENTA_CODIGO para completar la FK MEDIO_ENVIO_VENTA_CODIGO.

## Migrar_CuponXVenta

Procedure encargado de completar la tabla CUPONXVENTA, utilizando los datos provistos por la tabla maestra.

## Migrar_Venta_Cupon

Procedure encargado de completar la tabla VENTA_CUPON, utilizando los datos provistos por la tabla maestra.

## Migrar_Medio_Envio_Venta

Procedure encargado de completar la tabla MEDIO_ENVIO_VENTA, utilizando los datos provistos por la tabla maestra y las funciones OBTENER_MEDIO_ENVIO_CODIGO para completar la FK MEDIO_ENVIO_CODIGO, OBTENER_LOCALIDAD_CODIGO para completar la FK LOCALIDAD_CODIGO.

## Migrar_Medio_Envio

Procedure encargado de completar la tabla MEDIO_ENVIO, utilizando los datos provistos por la tabla maestra.

## Migrar_Cliente

Procedure encargado de completar la tabla CLIENTE, utilizando los datos provistos por la tabla maestra y la función OBTENER_LOCALIDAD_CODIGO para completar la FK LOCALIDAD_CODIGO.

### Migrar_Descuento_Venta

Procedure encargado de completar la tabla DESCUENTO_VENTA, utilizando los datos provistos por la tabla maestra y la función OBTENER_TIPO_DESCUENTO_CODIGO para completar la FK TIPO_DESCUENTO_CODIGO.

### Migrar_Tipo_Descuento

Procedure encargado de completar la tabla TIPO_DESCUENTO, utilizando los datos provistos por la tabla maestra.

### Migrar_Canal_Venta

Procedure encargado de completar la tabla CANAL_VENTA, utilizando los datos provistos por la tabla maestra y la función OBTENER_CANAL_CODIGO para completar la FK CANAL_CODIGO.

### Migrar_Canal

Procedure encargado de completar la tabla CANAL, utilizando los datos provistos por la tabla maestra.

### Migrar_Medio_Pago_Venta

Procedure encargado de completar la tabla MEDIO_PAGO_VENTA, utilizando los datos provistos por la tabla maestra y la función OBTENER_MEDIO_PAGO_CODIGO para completar la FK MEDIO_PAGO_CODIGO.

## FUNCIONES

### Obtener_Provincia_Codigo

Función capaz de retornar el código que identifica unívocamente una provincia, en base al nombre de la misma.

### Obtener_Localidad_Codigo

Función capaz de retornar el código que identifica unívocamente una localidad, en base al nombre de la misma. y si código postal.

### Obtener_Proveedor_Codigo

Función capaz de retornar el código que identifica unívocamente un proveedor, en base a su razón social y el CUIT.

### Obtener_Medio_Pago_Codigo

Función capaz de retornar el código que identifica unívocamente un medio de pago, en base al nombre del mismo.

### Obtener_Tipo_Variante_Codigo

Función capaz de retornar el código que identifica unívocamente a un tipo de variante, en base al nombre del mismo.

### Obtener_Producto_Categoria_Codigo

Función capaz de retornar el código que identifica unívocamente una categoría de productos, en base al nombre de la misma.

### Obtener_Descuento_Compra_Codigo

Función capaz de retornar el código que identifica unívocamente un descuento de compra, en base al valor descontado de la misma.

### Obtener_Tipo_Descuento_Codigo

Función capaz de retornar el código que identifica unívocamente un tipo de descuento, en base a la descripción del tipo de descuento.

### Obtener_Medio_Envío_Codigo

Función capaz de retornar el código que identifica unívocamente un medio de envío, en base a la descripción del medio de envío.

### Obtener_Canal_Codigo

Función capaz de retornar el código que identifica unívocamente un canal, en base a la descripción del canal.

### Obtener_Cliente_Codigo

Función capaz de retornar el código que identifica unívocamente un cliente, en base al dni y mail de un cliente.

### Obtener_Medio_Envio_Venta_Codigo

Función capaz de retornar el código que identifica unívocamente un medio de envio de una venta, en base al precio, localidad y código postal.

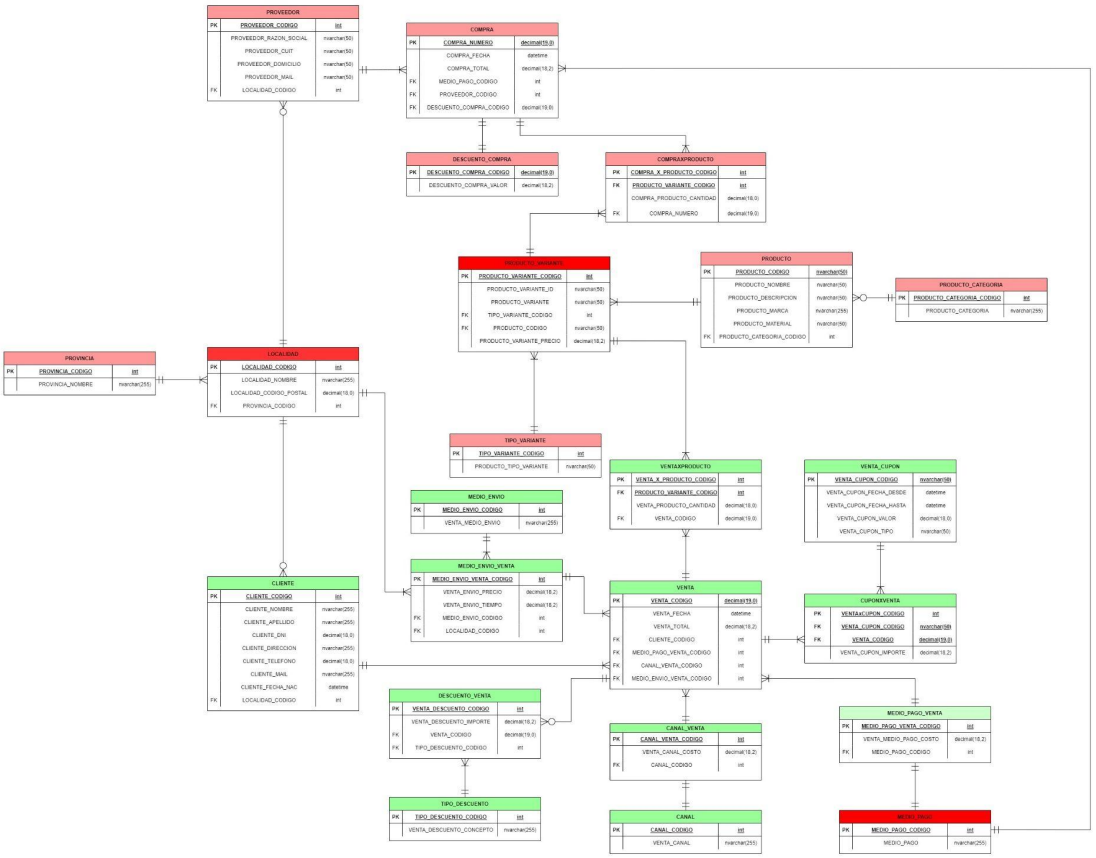
### Obtener_Medio_Pago_Venta_Codigo

Función capaz de retornar el código que identifica unívocamente un medio de pago de una venta, en base a la descripción del medio de pago.

### Obtener_Canal_Venta_Codigo

Función capaz de retornar el código que identifica unívocamente un canal de venta, en base a la descripción del canal.

# Entrega 3



# Cambios en el diagrama

Antes

| PRODUCTO_VARIANTE | | |
|---|---|---|
| PK | PRODUCTO_VARIANTE_CODIGO | int |
| | PRODUCTO_VARIANTE | nvarchar(50) |
| FK | TIPO_VARIANTE_CODIGO | int |
| FK | PRODUCTO_CODIGO | nvarchar(50) |

Después

| PRODUCTO_VARIANTE | | |
|---|---|---|
| PK | PRODUCTO_VARIANTE_CODIGO | int |
| | PRODUCTO_VARIANTE_ID | nvarchar(50) |
| | PRODUCTO_VARIANTE | nvarchar(50) |
| FK | TIPO_VARIANTE_CODIGO | int |
| FK | PRODUCTO_CODIGO | nvarchar(50) |
| | PROD_VAR_PRECIO | decimal(18,2) |

| COMPRAXPRODUCTO | | |
|---|---|---|
| PK | COMPRA_X_PRODUCTO_CODIGO | int |
| FK | PRODUCTO_VARIANTE_CODIGO | int |
| | COMPRA_PRODUCTO_PRECIO | decimal(18,2) |
| | COMPRA_PRODUCTO_CANTIDAD | nvarchar(255) |
| FK | COMPRA_NUMERO | decimal(19,0) |

| COMPRAXPRODUCTO | | |
|---|---|---|
| PK | COMPRA_X_PRODUCTO_CODIGO | int |
| FK | PRODUCTO_VARIANTE_CODIGO | int |
| | COMPRA_PRODUCTO_CANTIDAD | nvarchar(255) |
| FK | COMPRA_NUMERO | decimal(19,0) |

| VENTAXPRODUCTO | | |
|---|---|---|
| PK | VENTA_X_PRODUCTO_CODIGO | int |
| FK | PRODUCTO_VARIANTE_CODIGO | int |
| | VENTA_PRODUCTO_CANTIDAD | decimal(18,0) |
| | VENTA_PRODUCTO_PRECIO | decimal(18,0) |
| FK | VENTA_CODIGO | decimal(19,0) |

| VENTAXPRODUCTO | | |
|---|---|---|
| PK | VENTA_X_PRODUCTO_CODIGO | int |
| FK | PRODUCTO_VARIANTE_CODIGO | int |
| | VENTA_PRODUCTO_CANTIDAD | decimal(18,0) |
| FK | VENTA_CODIGO | decimal(19,0) |

Se mueve el precio de compra y precio de venta a la tabla de producto_variante. En esta tabla ahora, va a existir un producto_variante que va a tener el precio de venta y otro que va a tener el precio de compra. Por lo que las tablas de ventaxproducto y compraxproducto ya no tendrán sus valores de precio.

Respecto a la otra corrección que teníamos:

#ENVIO
Falta modelar la configuración de los medios de envio disponibles en función de la localidad/provincia, sobre la cual se configura el precio y tiempo estimado del mismo.

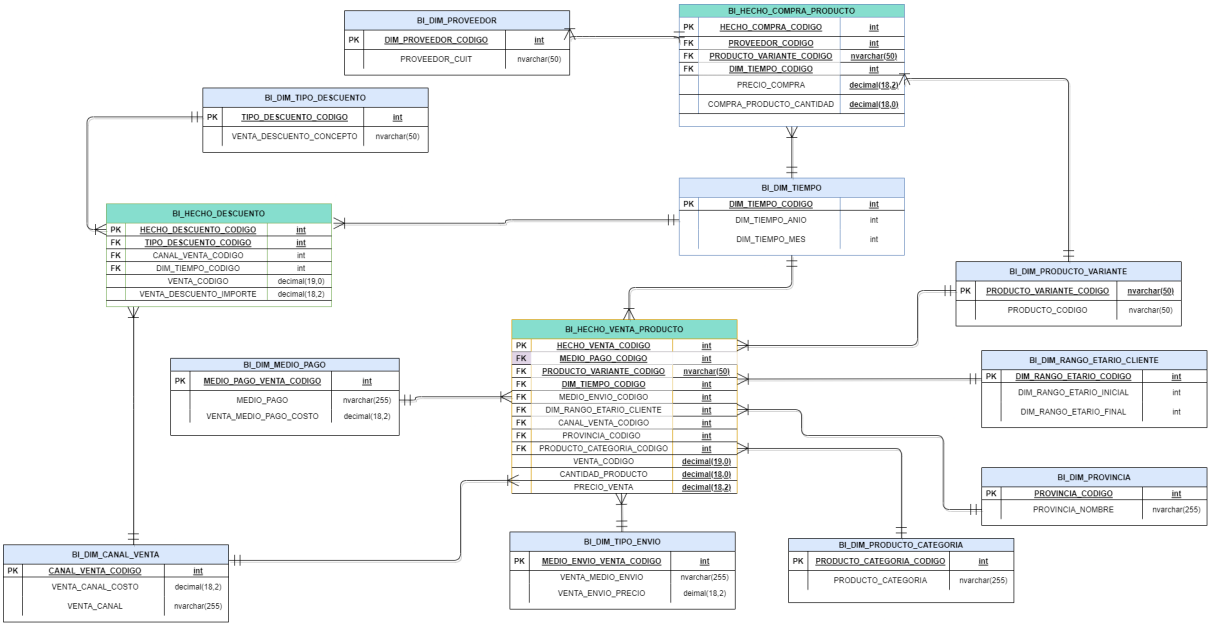
En nuestro DER tenemos configurado el precio y tiempo estimado del mismo en función de la PROVINCIA, LOCALIDAD y MEDIO_ENVIO. Como estas contienen dichos datos y no relaciones, cumplieron efectivamente con la configuración al ser tablas paramétricas.

Además, realizamos un pequeño cambio en la tabla CuponXVenta. Decidimos agregar una PK autoincremental. Esto es debido a que nos dimos cuenta de que un mismo cupón podía utilizarse en una misma venta varias veces. Esto generaba que se repitiera la PK compuesta previamente planteada de CODIGO_CUPON y CODIGO_VENTA, provocando la migración de menos filas que las correspondientes.

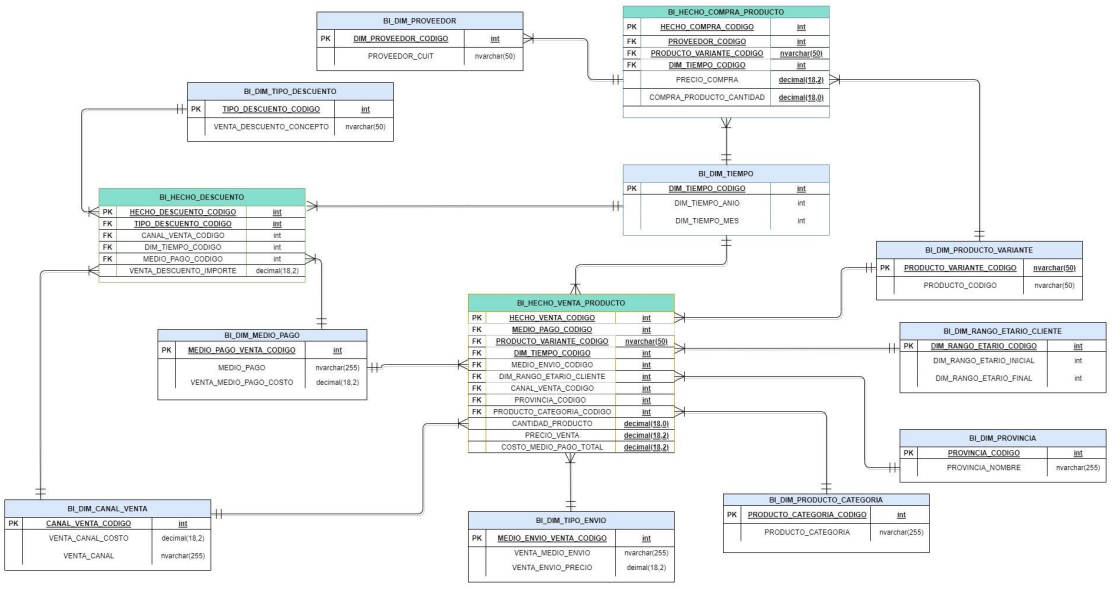
| CUPONXVENTA | | |
|---|---|---|
| PK | VENTAxCUPON_CODIGO | Int |
| FK | VENTA_CUPON_CODIGO | nvarchar(50) |
| FK | VENTA_CODIGO | decimal(19,0) |
| | VENTA_CUPON_IMPORTE | decimal(18,2) |

# Diagrama Business Intelligence

ANTES:




AHORA:




Realizamos algunos cambios en las tablas HECHO_VENTA_PRODUCTO y HECHO_DESCUENTO en base a las correcciones mencionadas.

Su Trabajo de Gestión de Datos ha sido corregido.

Su TP presenta los siguientes problemas:

# No es correcto agrupar por cada venta específica tanto en las ventas como en los descuentos Deben sacar el cod_venta de los group by de las consultas.

Deben corregir los puntos mencionados anteriormente.

# Creación de tablas BI

Seguimos los siguientes pasos.

## Dropeo de objetos

Nos aseguramos de borrar cualquier objeto preexistente en nuestro chema antes de avanzar.

## Dropeo de esquema

Una vez borrados los elementos preexistentes eliminamos el esquema.

## Creación de esquema

Creamos el esquema BI_CODIGO_ENIGMA donde desarrollaremos la entrega del BI.

## Creación de Tablas BI

Creamos una tabla por cada dimensión y hecho de nuestro modelo BI. Les ponemos sus atributos y luego las modificamos para hacer referencias a las otras tablas con FKs. Las PKs de las tablas en algunos casos las migramos del modelo inicial y en otros le asignamos una PK autoincremental si no existe una PK que podamos migrar.

## Creación de Funciones Auxiliares

Creamos algunas funciones auxiliares que serán de utilidad durante la migración de las tablas del BI.

## Creación de stored procedures para migrar datos

Ahora creamos las stored procedures las cuales se encargan de migrar los datos a sus respectivas tablas. Para las dimensiones hicimos un SELECT DISTINCT  de los registros de las tablas de la primera migración e insertamos los datos seleccionados. Para las tablas de hecho hicimos un SELECT DISTINCT de los registros de las tablas de la primera migración e hicimos un JOIN con las tablas dimensión que se relaciona con la tabla de hechos. Por último hicimos un GROUP BY por dimensión e insertamos los datos seleccionados.

## Comienzo de migración

Se ejecutan todos los stored procedures, migrando las tablas de nuestro BI.

### Creación de vistas

Creamos las vistas pedidas haciendo consultas a las tablas de hechos.

### Visualización de vistas

Mostramos las vistas.

## Decisiones en la migración de Datos

### BI_DIM_PRODUCTO_VARIANTE

Para la dimensión pedía en el enunciado de “Producto” decidimos que esto representaría al Producto Variante de la migración original, siendo su PK esta misma.

### BI_DIM_TIPO_DESCUENTO

Para dicha dimensión, su tipo_descuento_codigo es asignado según si este descuento que se aplica es de envío gratis, descuento por medio de pago, cupón o descuento especial. Al migrar, nos fijamos de donde proviene dicho descuento y le asignamos su respectivo valor. Esto luego se usa para la vista 5 de nuestro modelo BI.
**Observación:**
Se pudo observar que no había forma de detectar los descuentos de ENVÍO GRATUITO, ya que no teníamos VENTA_ENVIO_PRECIO = 0 and VENTA_MEDIO_ENVIO != ‘Entrega en sucursal' o el otro caso que podría aplicarse VENTA_DESCUENTO_CONCEPTO = ‘Otros’ and VENTA_ENVIO_PRECIO = VENTA_DESCUENTO_IMPORTE.

### BI_HECHO_VENTA_PRODUCTO, BI_HECHO_COMPRA_PRODUCTO

Para estos hechos, se los consideró como modelo la tabla VENTAxPRODUCTO y COMPRAxPRODUCTO respectivamente. En él se guarda el precio de venta y compra del producto y a su vez la cantidad que se vendió. Además, en la tabla de BI_HECHO_VENTA_PRODUCTO decidimos agregar el dato calculado de costo_medio_pago_total que representa el costo de medio de pago por mes, medio pago y canal de venta. Este dato es útil para las consultas 1 y 4.

### BI_DIM_PROVEEDOR

Se agregó la dimensión de BI_DIM_PROVEEDOR con su código de proveedor y cuit para poder realizar la vista 8. Esta dimensión se relaciona con la tabla BI_HECHO_COMPRA_PRODUCTO.

### BI_VISTA_PUNTO_8

En la consigna se solicitaba el promedio de precios de cada proveedor de forma anual. Debido a que cada proveedor tiene múltiples productos (en los cuales se podía calcular por

cada producto un promedio anual) y en la consigna no se pedía un promedio por producto, decidimos hacer un promedio de los promedios de los productos, dejando así un promedio de todos los productos por proveedor.