GESTIÓN DE DATOS

GRUPO N° 39	**CURSO:** K3013

PROFESOR: Enrique Reinosa, Juan Zaffaroni

NOMBRE DE GRUPO: NO_SE_BAJA_NADIE

ENTREGA N°: 3

TÍTULO: Entrega de Modelo BI

INTEGRANTES	
Diego Hernandez Canetti	176.198-5
Javier Federico García	177.162-0
Federico Leonel Bietti	175.663-1

ÍNDICE

DER TRANSACCIONAL

Pedido

Dentro de los requisitos del pedido en el enunciado, se menciona que se guardan los datos de usuarios, locales y repartidores; por lo tanto, se agregaron las claves foráneas correspondientes. Además, para referir al estado del pedido, creamos la entidad **Estado_Pedido,** de la cual el mismo tiene su id como FK.

Local, Categoría, Tipo y Horarios

La entidad local cuenta con el **LOCAL_COD** para diferenciarse, y además cuenta con FK del **LOCAL_TIPO.**
Al tener la idea de **CATEGORÍA** fuertemente relacionada con el tipo del local, decidimos crear la entidad **LOCAL_CATEGORIA** relacionada al **LOCAL_TIPO**. Para referir al horario en el cual el local está abierto, creamos la entidad **HORARIO_LOCAL** la cual cuenta con un id propio y el **LOCAL_COD** como FK, además de estar relacionado con el **DIA,** ya que cada dia tiene su horario, la entidad dia, cuenta con el **HORARIO_LOCAL_COD** como FK.

Productos , Catálogo de Local y Productos pedido

Nuestro sistema debería poder diferenciar los productos de un local específico (su catálogo) y los productos en general (pizza, empanadas, helados). Entonces se crearon 2 entidades: una llamada **Catálogo Local** y otra llamada **Producto.** Con la diferencia de que, la primera mencionada, tiene el código del local como FK y el stock en el local.
La entidad **Producto_pedido**, hace referencia a los productos asociados a un pedido y está asociado al pedido y al catálogo.
La tabla Producto en un principio se quizo utilizar como PK al Producto_Local_Codigo, pero al ser un nvarchar(50) no podiamos definirlo como PK por restricciones de SQL. Para respetar los tipos de datos como decia el enunciado, creamos un Producto_ID para cumplir como PK.

Usuario

El sistema debe ser capaz de identificar a cada usuario (todos son únicos) para poder hacer esto agregamos una atributo como PK llamado **USUARIO_ID.**

Direcciones

Cada usuario puede tener varias direcciones y estas pertenecen a un solo usuario, tenemos el **ID_USUARIO** del usuario asociado a la dirección en la entidad de la misma.

Medio Pago

Sobre el medio de pago se agregó la entidad **MEDIO_PAGO_TARJETA** para separar los datos de las tarjetas y pagar en efectivo. Y lo relacionamos con el pedido dándole la FK MEDIO_PAGO_ID.

Repartidor y Tipo de movilidad

Se decidió solucionar el "problema" de la localidad del repartidor agregando un atributo **Localidad,** de esta manera solamente se relaciona con una. Además, para poder identificar correctamente al repartidor, decidimos no tomar su DNI como PK sino que creamos un nuevo atributo llamado **REPARTIDOR_ID** el cual va a tener un número único por cada repartidor registrado en la plataforma. Asociado al repartidor tenemos la entidad que refiere a su tipo de movilidad **Tipo_movilidad** de la cual el repartidor tiene el id como FK.

Envio Mensajeria y Estado del envío

En el enunciado, se menciona el hecho de que la mensajería tiene asociadas las entidades: usuario, repartidor y paquete. Por este motivo, se agregó las FKs de cada una de estas.
Un problema que encontramos es que el sistema anterior no guardó el kilometraje entre origen y destino, siempre guardó 0 cuando en realidad debiese haber puesto el km, para hacer referencia al estado del envío, creamos la entidad **Estado_Mensajeria** de la cual el envío tiene el id.

Paquete

Se normaliza esta entidad creando una tabla nueva llamada **PAQUETE_TIPO** la cual tiene como función guardar toda la información referente a cierto tipo de paquete y su precio en ese respectivo momento. Por el contexto económico actual del país, se tomó la decisión de desnormalizar el precio y dejar este atributo en el **PAQUETE** también.

Cupones y Tipos

Es importante que en el pedido se sepa cuando se va a consumir un cupón, para ello decidimos crear la entidad **CUPON_PEDIDO** que media entre la entidad **CUPON** y el **PEDIDO** en la cual desnormalizamos el monto**.** Otro aspecto importante del sistema es que los cupones son personales para cada usuario, los cuales están asociados a su cuenta, para ello se agregó la FK **USARIO_ID.** Por último, es importante saber cuándo un cupón es generado por un reclamo, para esa diferenciación relacionamos los **RECLAMO** con los **CUPON** dandole una FK al reclamo con el CUPON_NRO del cupón generado. Para referir al tipo de cupón, creamos la entidad **Cupon_Tipo**, que cuenta con un id que le pasamos como FK al cupón para saber su tipo.

Operador Reclamo, Reclamo, Tipos de Reclamo y Estado

Para identificar a los operadores creamos una PK **OPERADOR_RECLAMO_ID** la cual es un número único para cada operador. Esta entidad de operador la asociamos a la de Reclamo a través de una FK. Por otro lado, el reclamo tiene asociado (según el enunciado) el pedido al cual se le realizó, por este motivo también se agregó la clave del pedido a sus atributos. Otro punto importante que tratamos fue la desnormalización de los reclamos, creamos una nueva tabla **RECLAMO_TIPO** y el ID del tipo lo pasamos como FK a la entidad **RECLAMO**. Además para referir al estado en el que se encuentra el reclamo, creamos la entidad **Estado_Reclamo,** de la cual el mismo tiene la id como FK.

Localidad

La entidad localidad simplemente contiene el código de la misma y el nombre. Y pasa el codigo como FK a todas las entidades que tengan una localidad. (**Envio_mensajeria**, **Direccion_Usuario**, **Repartidor** y **Local**).

DER BI

Hechos Pedidos

Para realizar este hecho se agregaron las dimensiones necesarias para realizar las vistas y el modelo de negocio: Tiempo, Dia, Rango Horario, Rango Etario Usuario, Rango Etario Repartidor, Localidad, Local, Medio pago Tipo, Tipo Movilidad, Tipo Local, Categoría Local, Estado Pedido y Cupón Pedido. Todas estas dimensiones comparten los siguientes atributos: cantidad de pedidos, valor promedio de envíos, total (que se calcula con la sumatoria total de cada producto incluido, más el envío, la propina y la tarifa de servicio, todo esto menos los descuentos por cupones), desvío minutos (Lo que tarda el pedido en llegar al cliente una vez iniciado el pedido en relación con el tiempo estimado) y un promedio de calificación de los pedidos. Todos estos datos le permiten al dueño del comercio tener la información necesaria y suficiente para tomar decisiones de negocio.

Cada una de las dimensiones mencionadas anteriormente, tienen solamente la información imprescindible, por ejemplo: del local nos quedamos con su descripción, de los medios de pagos solo nos quedamos con sus tipos(no agregamos información de las tarjetas).

Hechos Envio Mensajeria

Para realizar este hecho se agregaron las dimensiones necesarias para realizar las vistas y el modelo de negocio: Tiempo, Dia, Rango Horario, Rango Etario Usuario, Rango Etario Repartidor, Localidad, Tipo Movilidad, Tipo paquete, estado envío mensajeria.Todas estas dimensiones comparten los siguientes atributos: cantidad de envíos, promedio del valor asegurado, total (el cual es la sumatoria del envío, el precio por seguro y la propina), desvío en minutos (Lo que tarda el envio en llegar en comparación de lo estimado) y un promedio de calificaciones de los envíos.Todos estos datos le permiten al dueño del comercio tener la información necesaria y suficiente para tomar decisiones de negocio.

Como en el caso de los pedidos, todos los atributos de las dimensiones fueron “filtrados” para solamente quedarnos con la información que nos interesa.

Hecho Reclamo

Para realizar este hecho se agregaron las dimensiones necesarias para realizar las vistas y el modelo de negocio: Tiempo, Dia, Rango Horario, Local, Rango Etario Operador, Estado Reclamo, Tipo Reclamo, Cupón Reclamo.Todas estas dimensiones comparten los siguientes atributos: cantidad de reclamos, tiempo de resolución en minutos (que es la diferencia desde que se inició el reclamo hasta que se cerró) y un promedio de calificaciones del reclamo. Todos estos datos le permiten al dueño del comercio tener la información necesaria y suficiente para tomar decisiones de negocio.

Lo más importante para estos hechos es que todos los atributos están precalculados listos para ser consultados y así facilitar la toma de decisiones del negocio.

SCRIPT DE CREACIÓN Y MIGRACIÓN DE DATOS

Stored Procedures:

Se utilizaron stored procedures para la migración de la Base de Datos gd_esquema.Maestra a la nueva base. Para ello armamos para cada una de las entidades su propio Stored Procedure el cual contuviera TODOS los datos necesarios para migrar a esa entidad.

En cada uno de los procedures se fue restringiendo los datos que queríamos migrar para que nos causara problemas las FKs y PKs que definimos, como la inclusión de los is NOT NULL o también este tipo de código "SELECT TOP 1 USUARIO_ID FROM NO_SE_BAJA_NADIE.usuario WHERE DNI = M.USUARIO_DNI" para incluir las FKs que se encuentran en la nueva base de datos y evitar problemas con las subquerys. Todos estas estrategias las utilizamos para evitar insertar NULLS dentro de las FKs o PKs y se pueda migrar correctamente la base de datos.

Otro problema con el cual nos encontramos fui el tema de que no se nos permitía modificar PKs al realizar los INSERTS, pero por suerte pudimos solucionarlo, en los casos que hacía falta, utilizando

```
SET IDENTITY_INSERT NO_SE_BAJA_NADIE.X ON;
SET IDENTITY_INSERT NO_SE_BAJA_NADIE.X OFF;
```

Esto nos permitió realizar los inserts en las identidades sin ningún problema.

Con todas estas consideraciones en mente, fuimos atacando cada caso como uno totalmente diferente al próximo, armando todos los Stored Procedures sin errores y migrando todos los datos necesarios.

Índices

Decidimos armar índices duplicados y compuestos ya que de esta manera es mucho más fácil acceder a la información que solamente se puede obtener usando JOINS. Decidimos solamente armar los índices compuestos de las entidades con la información que esperamos que se releve con mayor frecuencia.

Constraints

En un principio escribimos constraints aparte, con sus alter tables definiendo primary keys y foreign keys. Al final por comodidad no lo vimos necesario y terminamos definiendo las PKs y FKs durante la creación de las tablas en las Create Table.

EXECUTE PROCEDURES

El orden tomado para la ejecución de los Stored Procedures se hizo para evitar errores. Primero las tablas sin Foreign Keys, de esta dependen las demás. Luego, tablas con FKs a tablas sin FKs, estas tablas llevan las Foreign Keys necesarias pero están ligadas a las tablas sin Foreign Keys. Por último las tablas con FKs a tablas con FKs. Ejecutando los procedures así, evitamos errores a la hora de ejecutar y migrar.

Dentro de todo el script, fuimos imprimiendo mensajes cada vez que se migran correctamente los datos; para los stored procedures no queríamos hacer la excepción. Primero intentamos poner todos los EXECUTE dentro de un try y catch,pero eso nos generaba problemas para debuggear si teníamos errores(y aún bien implementados atrapaban errores inexistentes al correrlos). Después pensamos imprimir un “Se ha migrado correctamente X” dentro de los create procedure como un RESULT, pero esto nos iba a llevar mucho tiempo implementarlo a cada uno. Por ello es que decidimos hacer al final del script un IF EXISTS que abarcara a cada una de las tablas, para ver si se había migrado “algo” en todas las tablas, si esto ocurre se lanza un “Se migró todo correctamente” y si no ocurría, es mucho más fácil encontrar el error que provoca esto.

SCRIPT CREACIÓN BI

Stored Procedures

Se crearon nuevos stored procedures para que se migren los datos requeridos para las dimensiones mínimas definidas en el enunciado, más las que creímos convenientes para el modelo OLAP como las dimensiones de cupones, tanto en general como por pedido y reclamo.
Se utilizó los datos migrados del script_creacion_inicial para extraer datos.

Decidimos unificar en un solo procedure los rangos horarios, dado que al probar por separado nos dimos cuenta que las querys devolvían los mismos horarios y era innecesario tener distintos rango horario para la apertura y cierre.

Se usaron JOINS en vez de subselects en los procedure de los hechos ya que vimos que al hacerlo de esta manera mejoraba de una manera exponencial el tiempo de migración. Todos los procedure se armaron bajo la idea de mejorar el rendimiento.

Views

Se crearon las views mínimas y necesarias definidas en el enunciado. Se realizaron después del migrado de datos, así era más fácil hacer SELECTs donde podíamos ver si estábamos obteniendo la información correcta. Esto fue clave para la realización de las mismas.