TP GESTION DE DATOS 2024
1C - SUPERMERCADO

UTN FRBA – GRUPO 34 QUICK_SORT

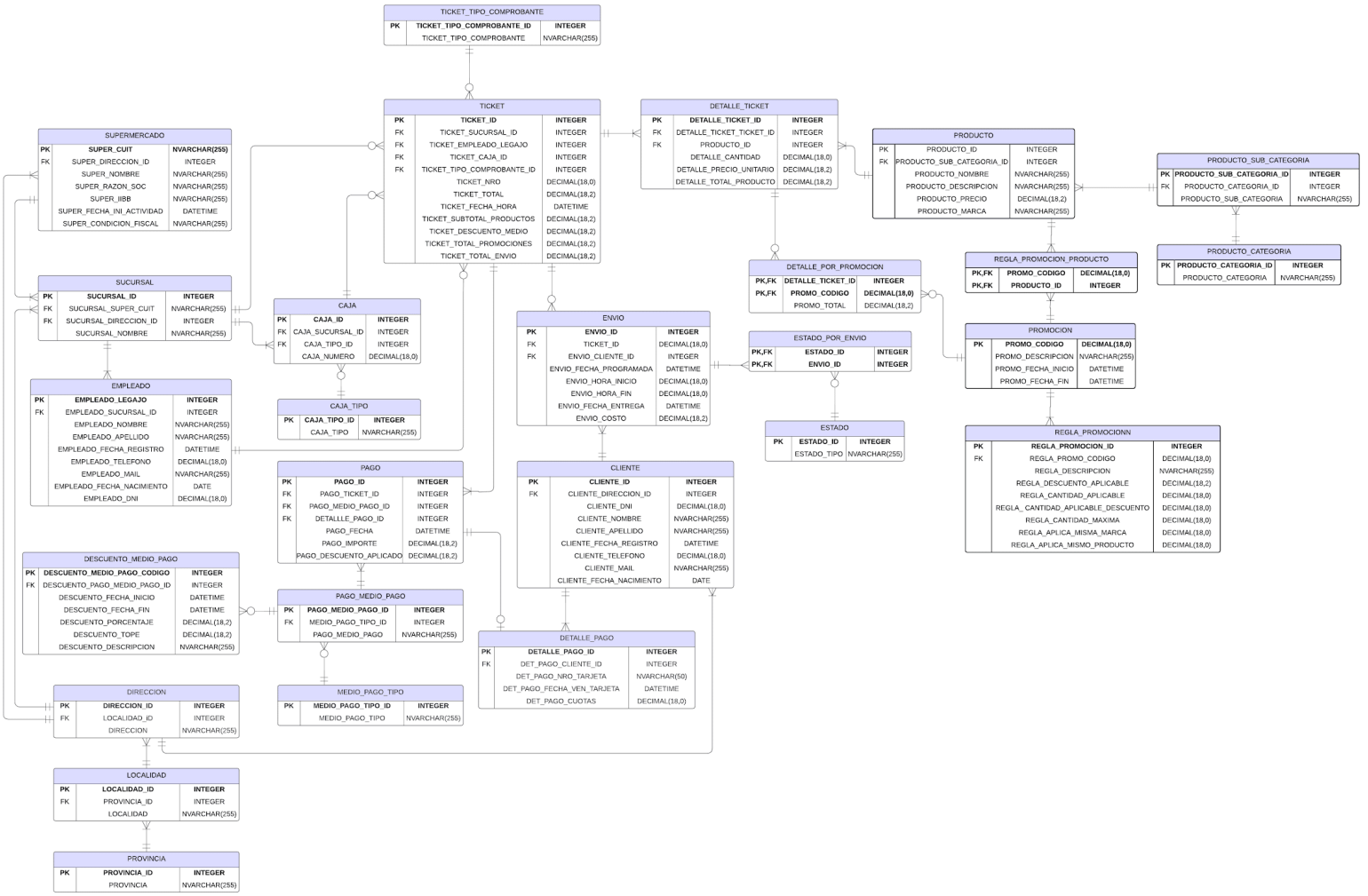
UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

LUCA SOSA CUENCA - 2040876
IGNACIO BISIO - 1726675
MARTIN GEREZ - 1416534
GIANCARLO PANDURO CALIXTO - 1405627

Índice

Diagrama de Entidad-Relación



Justificaciones de diseño

Normalización de Datos de Ubicación

Las entidades SUCURSAL, SUPERMERCADO y CLIENTE contienen información de ubicación. Para evitar la redundancia y asegurar la consistencia de los datos, se crearon las tablas DIRECCION, LOCALIDAD y PROVINCIA.

Esta decisión permite:

- Normalización: Almacenar las direcciones de manera eficiente y evitar duplicación de datos.
- Consultas eficientes: Facilitar la generación de reportes y consultas rápidas, como reportes de clientes por supermercado o análisis geográficos de ventas.
- Mantenimiento simplificado: Facilitar la actualización de datos de ubicación sin afectar múltiples registros.

Modelado de Pagos

Para modelar los pagos asociados a un ticket, se definieron las siguientes relaciones:

- TICKET a PAGO: Un TICKET puede tener múltiples pagos asociados, reflejando la posibilidad de pagar con diferentes métodos (ej: parte en efectivo y parte con tarjeta).
- PAGO a MEDIO_PAGO: Cada PAGO tiene asociado un único MEDIO_PAGO, asegurando que cada transacción se asocie con un método de pago específico.
- PAGO a DETALLE_PAGO: En los casos donde el pago se realiza con tarjeta, PAGO tiene una FK a DETALLE_PAGO para almacenar información relevante del cliente y de la tarjeta. Si el pago es en efectivo, este campo será NULL. Esto evita tener múltiples campos NULL en la entidad PAGO y mantiene el modelo limpio y eficiente.

Consistencia y Referencialidad

Se crearon tablas específicas para los distintos tipos de entidades que requieren consistencia y referencialidad:

- TICKET_TIPO_COMPROBANTE: Asegura que solo se utilicen tipos de comprobantes válidos.
- ESTADO: Garantiza que solo se puedan asignar estados predefinidos a las entidades.
- CAJA_TIPO: Establece tipos específicos de caja para las transacciones.

Esta estructura asegura que:

- Integridad de datos: Se mantenga la consistencia y validez de los datos.
- Facilidad de mantenimiento: Permitir agregar o modificar tipos de entidades sin necesidad de alterar la estructura principal de la base de datos.
- Flexibilidad y escalabilidad: Facilita la adición de nuevos tipos y categorías sin afectar las operaciones existentes.

Relación entre Entidades

Las principales entidades y sus relaciones se estructuraron para reflejar las interacciones y dependencias naturales del dominio:

- SUPERMERCADO y SUCURSAL: Un supermercado puede tener múltiples sucursales.
- SUCURSAL y EMPLEADO: Una sucursal puede tener varios empleados.
- TICKET y PAGO: Un ticket puede tener varios pagos, pero cada pago se asocia a un único ticket.
- PRODUCTO y DETALLE_TICKET: Los productos se asocian a los detalles de los tickets, reflejando los ítems comprados.

Entidades principales

TICKET: En la entidad ticket se decidió utilizar como identificador una PK de identidad, ya que el número de ticket no era suficiente para poder diferenciar a los mismos, luego de eso se referencian mediante claves foráneas a las entidades sucursal, empleado, caja y tipo de comprobante. Terminamos de completar la entidad con los datos solicitados en el enunciado, que son el número de ticket, el total del mismo, la fecha y hora, el subtotal de los productos, el descuento del medio , el total de las promociones y el total del envío.

DETALLE_TICKET: En la entidad Detalle ticket referenciamos a los productos por ticket, es decir que contamos con una clave foránea al ticket, y otra al producto al cual estamos haciendo referencia, guardando a su vez la cantidad de dicho producto, el precio por unidad de producto al momento de realizar la compra y el total del producto.

PAGO: En Pago, lo que hacemos es que cada uno tenga su propio identificador, ya que se solicitaba poder tener más de un pago a un mismo ticket, por lo que luego mediante una clave foránea referenciamos al ticket, también se va a guardar el ID del medio de pago como clave foránea, y una última FK que va a referenciar al detalle de pago, en este caso como solo se solicita para pagos con tarjeta, en caso de ser un pago realizado en efectivo, esta FK quedaria en NULL. Luego se completa la entidad con los datos pedidos que son fecha del pago, el importe y el descuento aplicado

PROMOCIÓN: Para la promoción se decidió utilizar como PK el código de la misma, luego se completó la entidad con la descripción y la fecha tanto de inicio como de fin.

ENVÍO: Para realizar la entidad envío decidimos utilizar un identificador generado como clave primaria, luego como FK se van a estar referenciando al ticket y al cliente al cual se esta realizando el envio donde alli estara la dirección a realizar el envio. Se completa la entidad con los datos restantes: fecha programada, hora de inicio, hora de fin, fecha de entrega y costo del envío.

Índices

Creamos un índice para el dni de los Clientes para poder acceder de una forma más rápida y eficiente a los datos de los clientes ya que creemos que será un campo que se utilizará bastante para generar consultas. Con el mismo fin, realizamos lo mismo para el nombre del Producto, esto nos brinda poder acceder a ellos mucho más fácil. Y por último, creamos un índice para las direcciones de forma de que si se quiere ordenar alfabéticamente, seria mucho mas facil y optimizamos su tiempo.

Creación de tablas

Dentro de un Store Procedure, creamos las tablas en orden asignando claves de Constraint Identity debido a que muchas de las entidades cuando las queremos migrar de la tabla maestra no cuentan con un dato que se pueda usar como PK. Esto se debe a que hay datos que a pesar de que pueden funcionar, se encuentran repetidos como por ejemplo TICKET_NRO. Usamos también constraints de NOT NULL para los campos de las entidades que son indispensables.

También se creó otro Stored Procedure que se ejecuta al principio del script para poder borrar todas las tablas en caso de que estas existan previamente.

Migración

Para la migración creamos otro stored procedure respetamos el mismo orden de inserción que el de creación de las tablas para garantizar que no ocurran errores entre los datos al cargar una tabla que necesite una FK de otra tabla que tiene que estar creada previamente. Vamos sacando los distintos datos que necesitamos de la tabla maestra y realizamos joins entre las tablas que creamos previamente para poder insertar los datos correctamente.