Contenido

Objetivos de la Base de Datos

El principal objetivo de la base de datos es proporcionar un sistema de almacenamiento seguro y eficiente para los datos de nuestra tienda de abarrotes, esto incluye:

a) Almacenar los datos de forma estructurada en tablas relacionales

b) Capacidad para manejar grandes volúmenes de datos de manera eficiente

c) Diseño de vistas con código optimizado para consultas de datos frecuentes

d) Creación de procesos almacenados para simplificar el acceso a los datos

e) Optimización de consultas para reducir el tiempo de espera y aumentar el rendimiento de la base de datos

f) Control de acceso basado en usuarios para limitar el acceso a la información solo a usuarios autorizados.

g) Encriptación de columnas sensibles para protegerlas contra accesos no autorizados

h) Auditoría y seguimiento de actividades para registrar y controlar todos los cambios realizados

i) Creación de claves primarias y foráneas, para garantizar la coherencia de los datos

j) Uso de transacciones para asegurar la información de las operaciones realizadas

k) Validar todos los datos antes de comenzar las transacciones para evitar la inserción de información incorrecta o código malicioso

l) Diseño relacional y flexible que permita la expansión e integración de nuevas funcionalidades en la base de datos

m) Documentación completa y clara de toda la estructura y función de la base de datos para su debido mantenimiento

n) Procedimientos almacenados que creen respaldos para garantizar la disponibilidad continua de los datos y una rápida recuperación en caso de falla

Requisitos para ejecutar el programa

Para garantizar el correcto funcionamiento del sistema de facturación, se deben cumplir los siguientes requisitos:

Requerimientos mínimos de Hardware:

- **Procesador:** Se recomienda un procesador de al menos 2 GHz o más rápido, preferiblemente de múltiples núcleos para un mejor rendimiento.
- **Memoria RAM:** Se recomienda al menos 4 GB de RAM para ejecutar SQL Server y SSMS sin problemas. Sin embargo, si tu base de datos es grande o estás trabajando con grandes conjuntos de datos, es preferible tener 8 GB o más de RAM.
- **Espacio en disco duro**: se recomienda un mínimo de 6 GB de espacio en disco para la instalación base. Además, necesitarás espacio adicional para almacenar bases de datos y archivos de registro.

Requerimientos del Sistema Operativo:

Las estaciones de trabajo deben ejecutar un sistema operativo compatible con las versiones de SQL Server y SQL Server Management Studio. Se recomienda:

- **Windows:** Se necesita al menos Windows Server 2012 (64 bit) para ejecutar SQL Server y SQL Server Management Studio o Windows 10 en adelante

- **Linux:** Se necesita al menos Red Hat Enterprise Linux 7.3 para ejecutar SQL Server o versiones más recientes

Requerimientos mínimos de Software:

- **SQL Server:** Se requiere una instancia de SQL Server instalada y configurada correctamente para alojar la base de datos del sistema de facturación. La versión mínima compatible es SQL Server 2017.
- **SQL Server Management Studio (SSMS):** Todos los usuarios que necesiten interactuar con la base de datos deben tener instalado SQL Server Management Studio versión 16.x. Esto les permitirá ejecutar consultas, actualizar datos y generar informes según sea necesario.

Entidades

La base de datos se compone por 10 tablas globales y 2 tablas temporales globales

a) Tablas Globales:

Empresa: Almacena toda la información de la empresa

empleado: Registra toda la información disponible de los empleados activos

cliente: Guarda la información disponible que tenemos de nuestros clientes

proveedor: Contiene los datos sobre los proveedores que tenemos

factura_encabezado: Registra los datos que conforma el encabezado de una factura

inventario: Almacena toda la información de nuestros productos en inventario

producto: Contiene toda la información acerca de los productos que tenemos en inventario

factura_detalle: Guarda todos los datos que forma el detalle de una factura

producto_inventario: Es una tabla intermedia que rompe la relación de muchos a muchos entre la tabla de producto e inventario

b) Tablas Temporales Globales:

##factura_encabezado_temporal: Sirve como puente temporal para los datos necesarios para facturar antes de ser insertados en la tabla factura_encabezado

##factura_detalle_temporal: Sirve como puente temporal para los datos extraídos del inventario antes de insertarlos en la tabla factura_detalle

Diagrama Entidad-Relación (ER)

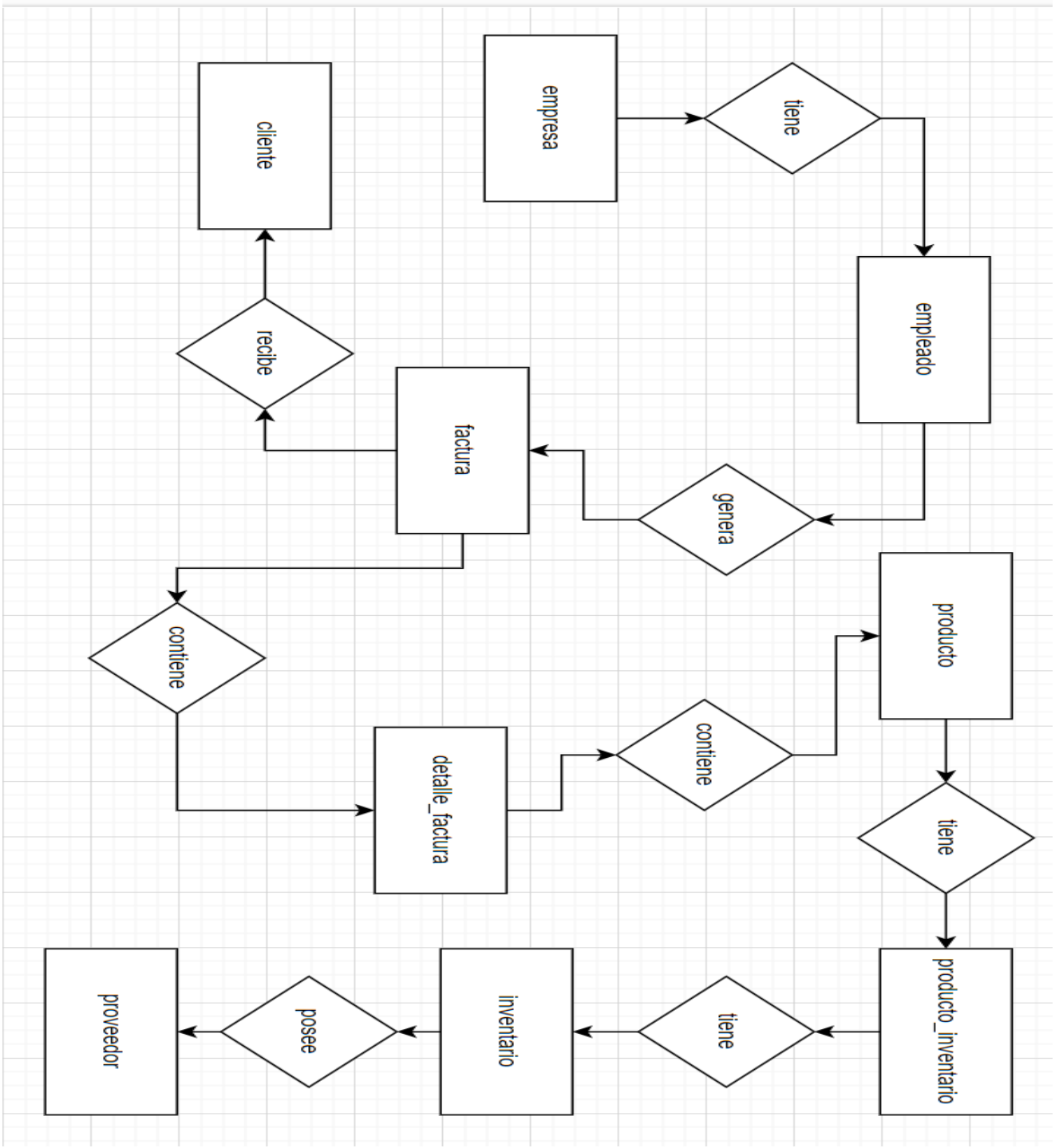
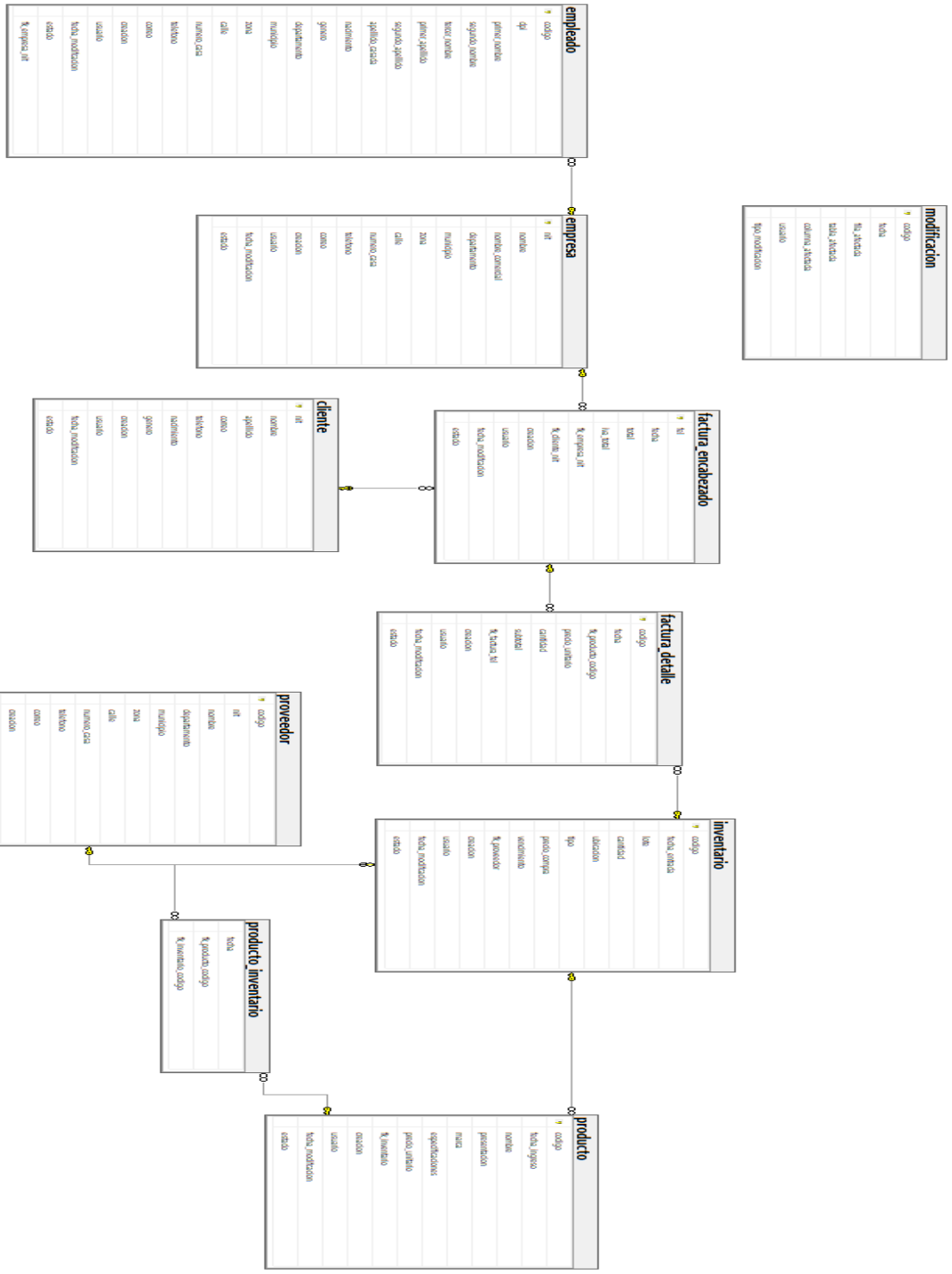


Diagrama Relacional



Diccionario de datos

empresa

Nombre	Tipo de dato	Descripción
código	VARCHAR (12)	Un código interno para cada empleado
dpi	VARCHAR (45)	Numero de DPI
primer_nombre	VARCHAR (45)	Primer nombre
segundo_nombre	VARCHAR (45)	Segundo nombre
tercer_nombre	VARCHAR (45)	Tercer nombre
primer_apellido	VARCHAR (45)	Primer apellido
segundo_apellido	VARCHAR (45)	Segundo apellido
apellido_casada	VARCHAR (45)	Apellido de casada
nacimiento	DATE	Fecha de nacimiento
genero	CHAR (1)	Genero sexual
departamento	VARCHAR (45)	Departamento donde vive
municipio	VARCHAR (45)	Municipio donde vive
zona	CHAR (1)	Zona donde vive
calle	CHAR (2)	Numero de calle donde vive
numero_casa	VARCHAR (8)	Numero de casa donde vive
telefono	VARCHAR (12)	Número de teléfono
correo	VARCHAR (45)	Dirección de correo electrónico
creación	DATETIME	Fecha de creación del registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo 'a', inactivo 'n')
fk_empresa_nit	VARCHAR (16)	Nit de la empresa

cliente

Nombre	Tipo de dato	Descripción
nit	VARCHAR (16)	Numero de nit
nombre	VARCHAR (45)	Nombre
apellido	VARCHAR (45)	Apellido
correo	VARCHAR (45)	Dirección de correo
telefono	VARCHAR (16)	Número de teléfono
nacimiento	DATE	Fecha de nacimiento
genero	CHAR (1)	Genero sexual
creación	DATETIME	Fecha de creación del registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo 'a', inactivo 'n')

Proveedor

Nombre	Tipo de dato	Descripción
código	VARCHAR (12)	Código interno del proveedor
nit	VARCHAR (16)	Numero de nit
nombre	VARCHAR (45)	Nombre
departamento	VARCHAR (45)	Departamento donde se ubica
municipio	VARCHAR (45)	Municipio donde se ubica
zona	CHAR (2)	Numero de zona donde se ubica
calle	CHAR (2)	Numero de calle donde se ubica
numero_casa	VARCHAR (8)	Numero de casa donde se ubica
telefono	VARCHAR (12)	Numero de teléfono
correo	VARCHAR (45)	Dirección de correo
creación	DATETIME	Fecha de creación de registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo ‘a', inactivo 'n')

factura_encabezado

Nombre	Tipo de dato	Descripción
fel	VARCHAR (140)	Numero de fel
fecha	DATETIME	Fecha de emisión
total	DECIMAL (10,2)	Monto total
iva_total	DECIMAL (10,2)	Iva total
fk_empresa_nit	VARCHAR (16)	Nit de la empresa emisora
fk_cliente_nit	VARCHAR (16)	Niit del cliente receptor
creacion	DATETIME	Fecha de creacion del registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo 'a', inactivo 'n')

Inventario

Nombre	Tipo de dato	Descripción
código	VARCHAR (16)	Código del producto
fecha_entrada	DATE	Fecha en que ingreso el producto
lote	VARCHAR (45)	Lote
cantidad	INT	Cantidad existente
ubicación	VARCHAR (45)	Ubicación
tipo	VARCHAR (45)	
precio_compra	DECIMAL (10,2)	Precio al que se compro
vencimiento	DATE	Fecha de vencimiento
fk_proveedor	VARCHAR (12)	Código del proveedor
creacion	DATETIME	Fecha de creacion del registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo 'a', inactivo 'n')

Producto

Nombre	Tipo de dato	Descripción
código	VARCHAR (16)	Numero de código
fecha_ingreso	DATE	Fecha en que ingreso
nombre	VARCHAR (45)	Nombre
presentación	VARCHAR (45)	
marca	VARCHAR (45)	Marca
especificaciones	VARCHAR (45)	Alguna especificación necesaria
precio_unitario	DECIMAL (10,2)	Precio por cada producto
fk_inventario	VARCHAR (16)	Código del producto en inventario
creacion	DATETIME	Fecha de creacion del registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo 'a', inactivo 'n')

factura_detalle

Nombre	Tipo de dato	Descripción
código	INT IDENTITY (1, 1)	Código interno
fecha	DATETIME	Fecha en que creo la factura
fk_producto_codigo	VARCHAR (16)	Código del producto
precio_unitario	DECIMAL (10,2)	Precio unitario del producto
cantidad	INT	Cantidad de productos
subtotal	DECIMAL (10,2)	Subtotal de la fila
fk_factura_fel	VARCHAR (140)	Código de la factura a donde pertenece
creacion	DATETIME	Fecha de creacion del registro
usuario	VARCHAR (60)	Usuario que creo el registro
fecha_modifcacion	DATETIME	Fecha en que se modificó el registro
estado	CHAR (1)	Estado (activo 'a', inactivo 'n')

Triggers

Un trigger es un bloque de código que se asocia a una tabla o vista, este código se ejecuta automáticamente cuando se produce un evento DML en la entidad a la que fue asociado.

Triggers de Auditoria

En nuestra base de datos, utilizamos triggers para registrar automáticamente los cambios que se realizan en las tablas. Estos triggers se activan cuando se borran o actualizan registros, y capturan la información relevante de los cambios, enviándola a una tabla de auditoría.

La tabla de auditoria se compone por
Código: El código se genera automáticamente cuando se inserta y sirve como identificados único para cada registro
Fecha: La fecha se genera automaticmante cuando se inserta y sirve para saber el momento exacto del cambio realizado
Tabla_afectada: Es enviado por el trigger e indica que tabla fue afectada
Modificación: Es enviado por el trigger y guarda el tipo de modificación que se efectúo en la tabla, ya sea un update o deleted
Fila_afectada: Es enviado por el trigger y registra la llave primaria de la fila que fue afectada, tiene el fin de saber que fila fue modificada
Columna_afectada: Es enviado por el trigger y solo aplica si la modificación realizada fue un update, guarda el nombre de la colmna que fue actualizada
Usuario: Es enviado por el trigger y guarda el nombre del usuario que realizo el cambio sobre la tabla

De esta forma creamos un registro de auditoria con todas las alteraciones realizadas a nuestros datos y poder darles un rastreo efectivo

Triggers de auditoria

Nombre	**Tabla asociada**
trEmpleadoDeleted	Empleado
trClienteDeleted	Cliente
trEmpresaDeleted	Empresa
trInventarioDeleted	Inventario
trProductoDeleted	Producto
trProveedorDeleted	Proveedor
trEmpleadoUpdated	Empleado
trClienteUpdated	Cliente
trEmpresaUpdated	Empresa
trInventarioUpdated	Inventario
trProductoUpdated	Producto
trProveedorUpdated	Proveedor

También se incluyeron 2 triggers que no permiten actualizar o borrar registros de la tabla “log”.
trImpedirUpdate y trImpedirDelted

Procesos Almacenados

Un proceso almacenado es un bloque de código que se utiliza para ser ejecutado múltiples veces sin necesidad de ser reescrito. Un proceso puede o no recibir parámetros, valores variables que se envían en la ejecución del proceso para operar adentro del mismo. De la misma forma un proceso puede devolver parámetros de salida, valores variables que resultan de la operación realizada.

En nuestra base de datos contamos con 11 procesos almacenados personalizados, se han dividido en 3 secciones:
Procesos para insertar
Procesos de transacción
Procesos de alteración

Procesos para insertar

Estos procesos los utilizamos para proteger los datos que son ingresados en nuestras entidades. Cada tabla tiene un proceso almacenado por el cual deben de pasar los datos antes de ser insertados. Cada parámetro recibe el valor solicitado por una columna, el proceso verifica que el valor enviado cumpla con los requisitos solicitados por su respectivo campo y mantenga la integridad de la columna.

Los parámetros de estos procesos reciben los valores que se enviara a las columnas de la tabla en la que insertan. Es decir, si el proceso registra en una tabla llamada 'cliente', el proceso debe tener la misma cantidad de parámetros que el número de columnas en la entidad cliente, además cada parámetro debe cumplir con los requisitos que el campo solicita. Cada parámetro pertenece al valor que debe recibir la tabla de izquierda a derecha (exceptuando las columnas que se insertan por default).

Nombre	Cantidad de parámetros	Tabla insertar
spInsertarCliente	4	Cliente
spInsertarEmpleado	8	Empleado
spInsertarInventario	8	Inventario
spInsertarProducto	8	Producto
spInsertarProveedor	8	Proveedor

Procesos de transacción

Son un conjunto de 3 procesos que se deben de ejecutar consecutivamente para crear una factura,
2 de ellos son similares a los procesos anteriores, seleccionan una tabla a insertar y cada parámetro recibe el valor para una columna. El tercero confirma que la ejecución de los 2 procesos anteriores sea correcta, por último, confirma la creación de una nueva factura

Los procesos de transacción son:
spInsertaFacturaEncabezado: 4 parametros, inserta en la tabla "##factura_encabezado_temporal",
spInsertarFacturaDetalle: 6 parametros, inserta en la tabla "##factura_detalle_temporal"

spConfirmarFactura: Extrae los datos de las tablas “##factura_encabezado_temporal” y “##factura_detalle_temporal” para se insertados en las tablas globales “factura_encabezado” y “factura_detalle”

Lea detalladamente como crear una factura en la sección “como crear una factura”. Pag 12

Procesos de alteración

Estos procesos los usaremos para las operaciones que no son de insercion, cubren tareas como: realizar backups, crear usuarios y enviar notifiaciones por correo ect.

spCrearUsuario, 3 parámetros, crea un nuevo usuario y lo inserta en la tabla “usuario”
spCrearBackUp, 0 parámetros, crea una copia de seguridad de la base de datos

Concurrencia

Concurrencia se refiere al hecho de que los Sistemas Administradores de Base de Datos permiten que muchas transacciones accedan a una misma Base de Datos al mismo tiempo.

Algunos de los problemas que surgen en los sistemas concurrentes son los siguientes:

Lectura sucia: Una transacción lee datos de otra transacción que aun no se a confirmado

Lectura no repetible: Una transacción vuelve a leer los datos que ha usado anteriormente, pero los datos ya han sido modificados por otra transacción

Lectura fantasma: Una transacción busca datos que cumplan con una condición, otra transacción modifica los datos que cumplen con esta condición antes de ser confirmada la primera transacción

Ejemplo sencillo: La tienda de abarrotes cuenta 120 cajas de leche de la marca “Vaca Feliz”, un usuario A lee el inventario, confirma que hay existencia y crea una factura con 100 cajas de leche, al mismo tiempo un usuario B lee el inventario, confirma que hay existencia y crea otra factura con 80 cajas de leche, ambos confirman la venta y han vendido 60 cajas de leche que no están en existencia

Control de concurrencia

Como ya vimos, cuando existen varios usuarios intentando modificar los mismos datos al mismo tiempo, se necesita establecer algún tipo de control para que dichas modificaciones de un usuario no interfieran en las de los otros, a este sistema se le denomina control de concurrencia.

Existe 2 tipos de control de concurrencia

Control de concurrencia pesimista: El sistema bloquea el acceso a los datos mientras una transacción los utiliza, son desbloqueados hasta que la transacción finalice.

Control de concurrencia optimista: El sistema asume que varias transacciones se pueden ejecutar al mismo tiempo sin interferir entre sí, una transacción puede acceder a los datos sin hacer bloqueos, cuando la transacción es confirmada verifica que otras transacciones no estén utilizando los mismos datos que ha leído, si los datos fueron utilizados por otra transacción, la transacción se deshace.

Como ya vimos el control de concurrencia optimista se basa en saber que los conflictos entre transacciones son raros y permiten que las transacciones se ejecuten sin bloqueos hasta el final. Por otra parte, el control de concurrencia pesimista se basa en la suposición de que los conflictos entre transacciones ocurrirán con frecuencia y utiliza bloqueos para asegurar que solo una transacción pueda utilizar lo mismo datos.

Niveles de aislamiento

READ UNCOMMITTED
Especifica que las operaciones pueden leer filas que han sido modificadas por otras transacciones, pero todavía no se han confirmado.

REPEATABLE READ
Especifica que las instrucciones no pueden leer datos que han sido modificados, pero aún no confirmados por otras transacciones y que ninguna otra transacción puede modificar los datos leídos por la transacción actual hasta que ésta finalice.

SERIALIZABLE

- Las operaciones que realiza la transacción no pueden leer registros que hayan sido modificados, pero aún no confirmados, por otras transacciones.
- Ninguna transacción puede modificar los datos leídos por la transacción actual hasta que la transacción actual finalice.
- Otras transacciones no pueden insertar filas nuevas con valores de clave que pudieran estar incluidos en el intervalo de claves leído por las instrucciones de la transacción actual hasta que ésta finalice.

ACID significa Atomicity, Consitency, Isolation, Durability, son las propiedades que aseguran que una transacción será confiable y consistente

Atomicidad: Todas las operaciones que realiza la transacción deben de ser de estado commit o rollback, es decir, se confirman todos los cambios o no se realiza ninguno

Consistencia: Todas las operaciones realizadas por la transacción deben mantener la integridad de la base de datos, si una operación dentro de la transacción intenta insertar un numero negativo donde se requiere un entero positivo, la operación debe fallar y revertirse

Aislamiento: La transacción deben de tener un nivel de aislamiento para evitar leer datos erróneos, en este caso utilizamos el nivel de aislamiento serializable, este nivel de aislamiento bloqueara las tablas donde opera hasta que la transacción no se de estado commit o rollback, si una transacción A intenta operar sobre los datos que una transacción B está modificando, la transacción A deberá de esperar hasta que la transacción B se confirme

Durabilidad: Los efectos de una transacción no deben de ser visible para otras transacciones hasta que la transacción sea de estado commit

En esta base de datos utilizamos el control de concurrencia optimista, utilizamos el nivel de aislamiento READ COMITED y seguimos las propiedades ACID en todas las transacciones

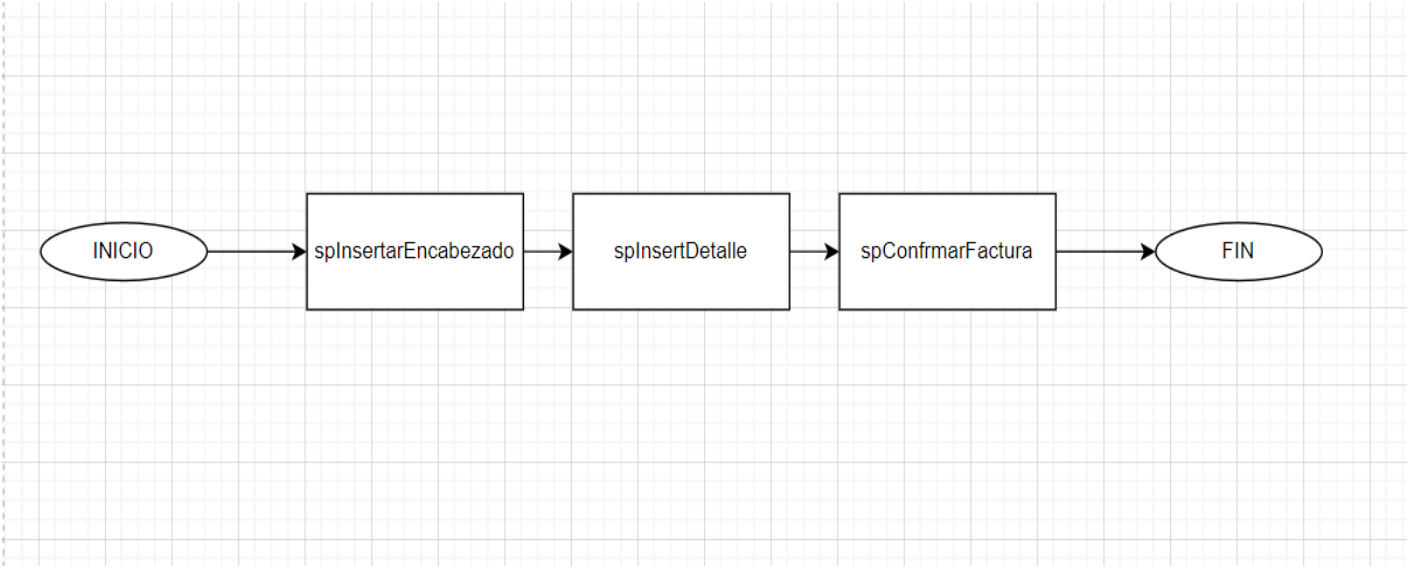
Como crear una factura

Para que sea posible guardar correctamente los datos de cada factura, seguimos los principios de normalización y dividimos los datos de una venta en 2 tablas. La primera tabla llamada “factura_encabezado” registra la información general de cada venta (Fecha, FEL, NIT emisor, NIT receptor etc.). La segunda tabla llamada “factura detalle” cubre toda la descripción de la misma venta (producto, cantidad, precio etc.).
Esta separación permite tener ventas con un numero variable de productos y únicamente crear un registro de encabezado para cada documento

Para la emisión eficiente de cada factura ejecutamos 3 procesos almacenados, cada uno contiene un bloque try-catch, dentro del bloque try se inicia una transacción, mientras la transacción no sea de estado commit o rollback las tablas que se están leyendo se mantendrán bloqueadas, de esta forma se evita que los empleados generen facturas con datos inexactos o que ya fueron alterados, cabe resaltar que cada transacción esta optimizada para ser commit o rollback de manera inmediata, de esta forma no se ralentiza las acciones de los empleados.

El primer proceso recibe los datos, los valida y coloca en la tabla temporal global ##factura_encabezado_temporal, aquí permanecerán hasta haber terminado de crear la factura, seguido a esto un segundo proceso se debe ejecutar con los datos para el detalle de la factura que estamos creando, este proceso se ejecuta cuantas veces sea necesario para abarcar todos los productos que el cliente solicita, nuevamente se inserta en una tabla temporal global ##factura_encabezado_temporal. Por último, se confirma la creación de la factura con un tercer proceso, este extrae los datos insertados anteriormente en las tablas temporales y los insertan en las tablas globales.

Diagrama del orden de ejecución obligatorio de los procesos para crear una factura:



Nombre: spInsertarFacturaEncabezado

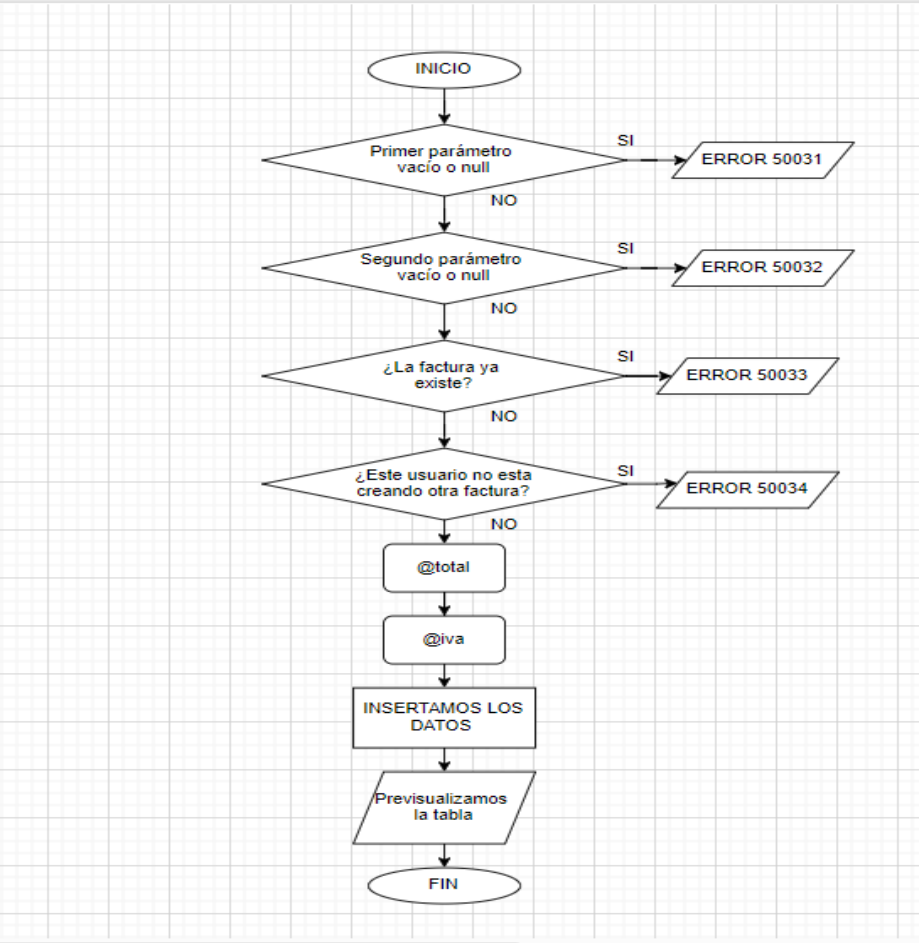
Parámetros:

1) @fel: Numero de fel para el encabezado de factura

2) @nit_cliente: Numero de NIT del cliente para el encabezado de la factura

Descripción: Este proceso recibe 2 parámetros para la creación del encabezado de 1 factura, primero verifica que los parámetros contengan valores válidos, después los coloca en una tabla temporal donde esperan a ser confirmados por un siguiente proceso

Diagrama:



Nombre: spInsertarFacturaDetalle

Parámetros:

1) @fk_factura_fel: Numero de fel de su respectivo encabezado

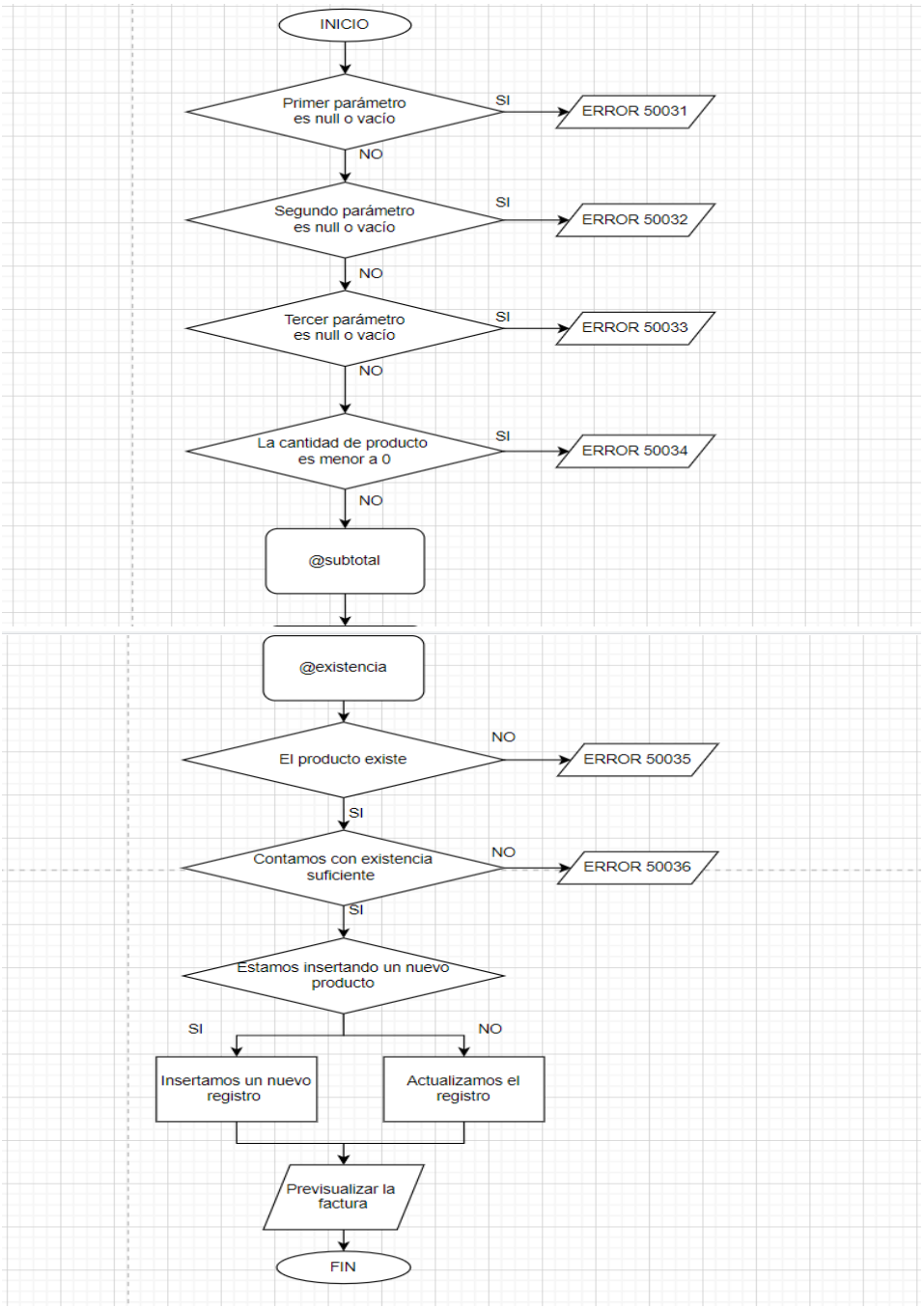
2)@codigo: El código correspondiente para al registro

3)@fk_producto: El código del producto que se va a registrar en la factura

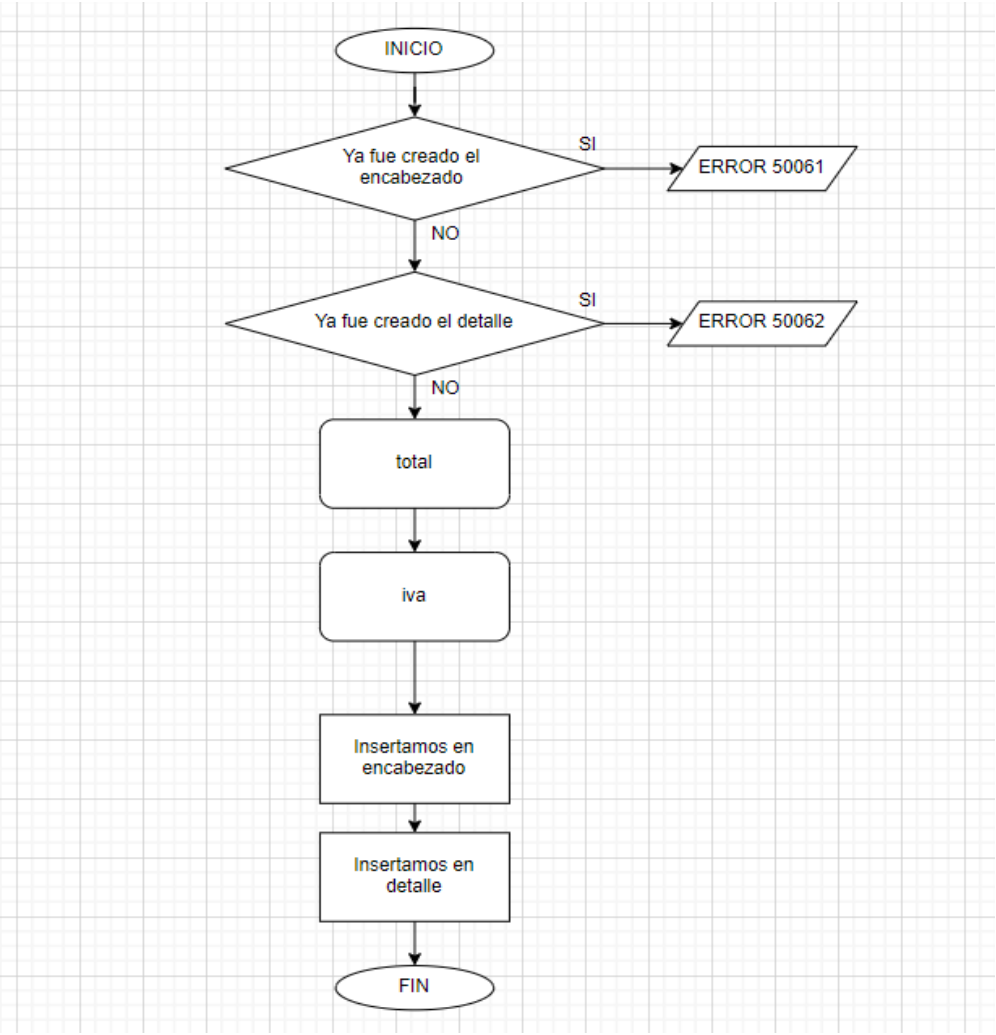
4)@cantidad: La cantidad de producto que el cliente lleva

Descripción: Este proceso recibe 4 parámetros para la creación del detalle de 1 factura, primero verifica que los parámetros contengan valores válidos, después los coloca en una tabla temporal donde esperan a ser confirmados por un siguiente proceso

Diagrama:



Nombre: spInsertarFacturaDetalle
Parámetros:
Descripción: Este proceso se ejecuta sin parámetros, analiza que los datos de encabezado y detalle sean coherentes y confirma la creación de la factura
Diagrama:



Errores

Se reservaron los codigos 50001 a 500061 para la creación de errores controlados, los errores controlados ayudaran a que el programa no.....
Error 50001 a 500030: Hechos para controlar los errores ocurridos en la inserción de datos en encabezado de la factura
Error 50031 a 50060: Hechos para controlar los errores ocurridos en la inserción de datos en detalle factura

Error 50001: Indica que el parámetro @nit contiene un valor null o vacío
Nivel: Leve
Origen: spInsertarFacturaEncabezado
Causas: El parametro @nit no fue proporcionado durante la ejecución del proceso.
El valor del parametro @nit se eliminó inadvertidamente o no se pasó correctamente a la consulta.

Error 50002: Indica que el parámetro @fel contiene un valor null o vacio
Nivel: Leve
Origen: spIsertarFacturaEncabezado
Causas: El parametro @fel no fue proporcionado durante la operación de inserción.
El valor del parametro @fel se eliminó inadvertidamente o no se pasó correctamente a la consulta.

Error 50003: Indica que el valor del parámetro @fel ya esta registrado en otra factura
Nivel: Moderado
Origen: spInsertarFacturaEncabezado
Causas: El valor del parámetro @fel fue encontrado en otro documento por lo que no se puede volver a usar

Error 50004: Indica que un mismo usuario está intentando crear 2 facturas al mismo tiempo
Nivel: Moderado
Origen: spFacturaEncabezado
Causas: Un mismo usuario esta intentando crear una nueva factura cuando aun tiene un documento sin confirmar o deshacer

Error 50031: Indica que el valor del parámetro fel contiene null o vacio
Nivel: Moderado
Origen: spFacturaDetalle
Causas: El usuario esta intentando insertar el detalle de la factura sin haber generado un numero de fel con anterioridad

Error 50032: Indica que el valor del parámetro @codigo es null o esta vacio
Nivel: Moderado
Origen: spFacturaDetalle

Causas: El parámetro @fel no fue proporcionado durante la operación anterior, nótese que si el parámetro @fel contiene un valor null o vacio en esta etapa es porque no se envio correctamente el mismo parámetro en la inserción anterior, asegúrese que el fel ingresado es el miso que inserto en el encabezado, o bien que ya se halla creado dicho encabezado para el documento

Error 50033: Indica que el valor del parámetro @fk_producto es null o esta vacio
Nivel: Moderado
Origen: spFacturaDetalle
Causas: EL parámetro @fk_productono fue proporcionado durante la ejecución del proceso
El valor del parámetro @fk_producto se elimino inadvertidamente o no se paso correctamente a la consulta

Error 50034: Indica que el valor del parámetro @cantidad es null o menor a 0
Nivel: Leve
Origen: spFacturaDetalle
Causas: El parámetro @cantidad se envio con un valor null o menor a 0, por tratarse de la cantidad de productos a comprar no puede ser un número menor a 0 o decimal

Error 50035: Indica que la cantidad de productos solicitadas no está disponible en existencia
Nivel: Critico
Origen: spFacturaDetalle
Causas: La cantidad de productos que selecciono el cliente es mayor a la cantidad del mismo producto en stock, nótese que, si el cliente pudo seleccionar una cantidad de producto que no esta disponible en stock, hay una falla grave en la resta de productos en existencia

Error 50036: Indica que el valor del parámetro @fk_producto no existe en nuestro inventario
Nivel: Critico
Origen: spFacturaDetalle
Causas: El parámetro @fk_producto contiene un valor que no existe en nuestro inventario
Puede tratarse de un error en la escritura del código o un producto que no esta registrado en la tienda

Seguridad

Como ingresar datos:

Por motivos de seguridad, todos los datos antes de ser insertados son validados por medio de 1 proceso almacenado, este proceso verifica que los datos a insertar cumplan los requisitos solicitados por los campos de cada tabla, de esta forma se evitan inyecciones SQL o inconsistencias en nuestros datos.
Proceso almacenado que se utiliza para insertar en cada tabla:

Proceso	Tabla
SpInsertarCliente	Cliente
SpInsertarEmpleado	Empleado
SpInsertarProveedor	Proveedor
SpInsertarEmpresa	Empresa
SpInsertarUsuario	Usuario
SpInsertarProducto	Producto
SpInsertarInventario	Inventario

Auditoria:

La base de datos incluye una tabla llamada "log" que registra todas las modificaciones que se realizan en las entidades. La tabla "log" almacena la fecha, el nombre del usuario, la tabla modificada, la columna modificada y el valor anterior a la modificación.

Estos datos se obtienen por medio triggers, todas las tablas cuentan con 2 triggers, que se activan cada vez que un usuario modifica o elimina datos de una entidad El primer trigger se activa cuando se eliminan registros de la tabla. El segundo trigger se activa al modificar registros. Cada trigger envía la información correspondiente a la tabla "log" para fines de auditoría. La tabla “log” esta protegida por 2 triggers que no permiten actualizar o eliminar los registros.

Para la auditoria de nuevos datos insertados, la tabla ya dispone de 3 columnas (creación, fecha_creacio, usuario)

Usuarios:

Un usuario se puede crear por medio del proceso almacenado “spCrearUsuario”, este proceso recibe como parámetros los valores para crear un nuevo usuario.

El primer parámetro es el nombre de usuario, el segundo parámetro recibe la contraseña para el usuario, un tercer proceso recibe el rol a asignar para el nuevo usuario. El proceso verifica que los 2 primeros parámetros sean cadenas alfanuméricas de mas de 12 caracteres, el tercer proceso se verfica como un rol asignar

Encriptacion de datos sensibles:

La base de datos encripta todos los datos considerados sensibles desde que se insertan en su respectiva tabla, los datos sensibles no se desencriptan en ningún proceso, reporte o vista

Para aumentar la seguridad de los datos, cada tabla ha sido encriptada con certificado y claves diferentes

Lista de todas las columnas encriptadas y su respectiva tabla

Columna	**Tabla**
nit	empresa
dpi	empleado
nacimiento	empleado
teléfono	empleado
correo	empleado
teléfono	cliente
nacimiento	cliente
teléfono	proveedor
correo	proveedor

Conclusión

La creación de esta base de datos para la tienda de abarrotes ha sido diseñada con el objetivo de optimizar la gestión de inventario, ventas, y proveedores. Facilita el manejo eficiente de los datos para las operaciones diarias del negocio. A través de una estructura bien definida y normalizada, se busca garantizar la integridad, consistencia y disponibilidad de la información.

Beneficios Clave

La base de datos permite un seguimiento preciso y actualizado del inventario, reduciendo las incidencias de productos agotados y optimizando el reabastecimiento. Los niveles de stock se pueden monitorear en tiempo real, lo que facilita una mejor toma de decisiones y planificación.

La base de datos facilita la rápida recuperación de información sobre productos y precios, agilizando el proceso de venta y mejorando la experiencia del cliente.

La inclusión de un módulo dedicado a la gestión de proveedores permite mantener un registro actualizado de los mismos, incluyendo información de contacto y detalles de los productos suministrados. Esto ayuda a establecer relaciones más sólidas y eficientes con los proveedores.

La implementación de mecanismos de seguridad asegura que solo personal autorizado pueda acceder y modificar los datos críticos. Esto protege la información sensible de accesos no autorizados y posibles manipulaciones.

La estructura de la base de datos está diseñada para adaptarse a las necesidades actuales de la tienda, con la posibilidad de escalar y expandirse según crezcan las operaciones y se incorporen nuevas funcionalidades.

Conocimientos Aplicados

Entendimiento y adaptación de requerimientos funcionales
Crear y relacionar tablas
Creación de diagramas ER Y RE
Creación de diccionarios de datos
Control de Auditoria
Creación de transacción con control de concurrencia optimista y propiedades ACID
Creación de Usuarios
Creación de Copias de seguridad
Generar errores controlados
Encriptación de datos