Entrega_Coder_Santiago_E_Sanchez_Marsili

Ver la entrega completa en el link

Trabajo de entrega Coderhouse

Introducción

Bienvenidos al proyecto. En el contexto actual, donde la eficiencia operativa y el control preciso del inventario son claves para la competitividad, contar con una infraestructura tecnológica sólida se vuelve esencial. En esta presentación, exploraremos cómo nuestro sistema de gestión permite optimizar los procesos de stock, compras y ventas, mejorar la trazabilidad de los productos y ofrecer una base escalable para futuras integraciones, como el soporte técnico especializado. Este proyecto está orientado a brindar soluciones concretas para usuarios finales y pequeñas empresas, con foco en la eficiencia, la personalización y el crecimiento sostenido.

Objetivo del trabajo

Estamos desarrollando una base de datos integral y escalable para una distribuidora de hardware, orientada a optimizar la gestión de stock, compras y ventas. Esta infraestructura tecnológica permite centralizar procesos clave, mejorar la trazabilidad y reducir costos operativos, generando métricas confiables para decisiones estratégicas. Nuestro mercado objetivo son usuarios finales y pequeñas empresas que buscan soluciones accesibles, eficientes y personalizadas en tecnología. Como parte de nuestra estrategia de crecimiento, proyectamos incorporar servicios de soporte técnico especializado, ampliando nuestra propuesta de valor y fortaleciendo la fidelización de clientes. Este proyecto representa una oportunidad de inversión en un modelo ágil, con potencial de expansión sostenida y diferenciación competitiva en un segmento dinámico y en evolución.

Problema a resolver

Al ingresar recientemente al mercado de distribución de hardware, hemos realizado una apuesta comercial significativa al adquirir un stock amplio y variado de productos. Sin embargo, la falta de un sistema de gestión estructurado dificulta el control eficiente de ese inventario, exponiéndonos a riesgos como pérdidas por errores de registro, demoras en la reposición y falta de visibilidad sobre el rendimiento comercial de cada ítem. Esta situación compromete la toma de decisiones estratégicas y limita nuestra capacidad de respuesta frente a clientes y proveedores. La implementación de una base de datos funcional y escalable se

presenta como una solución clave para ordenar, automatizar y potenciar la operación desde sus cimientos.

Descripción del modelo de negocio

El modelo de negocio se basa en la comercialización de productos de hardware con un enfoque centrado en la atención personalizada y especializada. Cada cliente, ya sea usuario final o pequeña empresa, recibe asesoramiento directo por parte de profesionales con conocimiento técnico, capaces de recomendar soluciones adaptadas a sus necesidades específicas. Esta orientación permite no solo garantizar la calidad del producto ofrecido, sino también construir relaciones de confianza sostenidas en el tiempo.

La operación se apoya en una gestión eficiente del stock, con procesos optimizados para asegurar tiempos de entrega ágiles y trazables. La combinación entre excelencia técnica, cercanía en el trato y eficiencia logística constituye el núcleo diferenciador del negocio, permitiendo posicionarse en un segmento que valora tanto el producto como la experiencia de compra. Este modelo está diseñado para escalar de forma ordenada, manteniendo la calidad en cada etapa del crecimiento.

Conclusión de esta primera entrega:

Este proyecto representa el primer paso en la construcción de una infraestructura tecnológica sólida para una distribuidora de hardware con visión estratégica. Si bien la base de datos se encuentra en una etapa inicial de desarrollo, ya se delinean los pilares fundamentales del modelo de negocio: atención personalizada, excelencia en producto y eficiencia operativa. La propuesta busca sentar las bases para un crecimiento ordenado, escalable y diferenciado, con capacidad de adaptación a las necesidades del mercado. Esta entrega inicial refleja el compromiso con una evolución constante, donde cada componente técnico responde a una lógica comercial clara y a una experiencia de usuario cuidada desde el origen.

Conclusión de esta segunda entrega

Esta segunda entrega consolida los cimientos técnicos del proyecto, transformando la visión inicial en una arquitectura relacional funcional y escalable. A través de vistas detalladas, procedimientos automatizados y funciones específicas, se refuerza la trazabilidad, la eficiencia operativa y la capacidad de análisis estratégico. Cada componente refleja una evolución consciente del modelo de negocio, donde la tecnología no solo organiza, sino que potencia la experiencia del usuario y la toma de decisiones. Este avance reafirma el compromiso con una infraestructura viva, capaz de crecer y adaptarse sin perder su esencia.

Conclusión final del proyecto *_Cuchuflito_SA_*

Lo que comenzó como una propuesta conceptual para ordenar el negocio de distribución de hardware, hoy se consolida como una arquitectura relacional viva, funcional y pedagógica.

La primera entrega delineó los pilares: atención personalizada, excelencia técnica y eficiencia operativa. La segunda los materializó en vistas, funciones, procedimientos y triggers que no solo organizan, sino que enseñan, auditan y escalan.

Esta evolución no es solo técnica: es una declaración de intenciones. Cada componente fue diseñado con conciencia comercial, lógica relacional y vocación formativa. El sistema ya no es una promesa: es una herramienta que respira, registra y responde.

*_Cuchuflito_SA_* se convierte así en un modelo de base de datos que no solo gestiona productos, sino que documenta decisiones, preserva trazabilidad y habilita crecimiento. Una infraestructura que honra su origen y proyecta su futuro.

Descripción de la base de datos

La base de datos **__cuchuflito_sa__** es un esquema que almacena datos de una firma dedicada a la venta de hardware informático. A continuación, se describen las tablas que la componen.

Tablas que componen la base de datos de `Cuchuflito_SA`

Tabla `clientes`

__Descripción__

Registra los datos personales y de contacto de cada cliente, ya sea habitual o eventual.

__Objetivo__

Permitir la identificación única de cada cliente, facilitando:

- La trazabilidad de operaciones comerciales

- La personalización de servicios

- La segmentación de usuarios

Relaciones

- Se vincula con `ventas` mediante `ID_Cliente`

- Participa en vistas como `vista_ventas_detalladas`

Tabla `proveedores`

Descripción

Almacena la información de cada proveedor registrado en el sistema.

Objetivo

Facilitar el control de abastecimiento, permitiendo:

- Identificar el origen de cada producto

- Auditar compras por proveedor

- Evaluar relaciones comerciales

Relaciones

- Se vincula con `compras` mediante `ID_Proveedor`

- Participa en vistas como `vista_inventario_por_proveedor`

Tabla `medios_de_pago`

Descripción

Define los métodos de pago disponibles, incluyendo si permiten cuotas.

Objetivo

Formalizar las condiciones de pago, permitiendo:

- Registrar transacciones con distintos medios

- Controlar pagos fraccionados

- Enseñar lógica condicional en SQL

Relaciones

- Se vincula con `ventas` y `cuotas_pago`

- Participa en procedimientos como `registrar_venta_con_cuotas`

Tabla `productos`

****Descripción****

Contiene el catálogo de productos comercializados, con descripción, precio y stock.

****Objetivo****

Centralizar la información de productos, permitiendo:

- Controlar inventario

- Asociar productos a compras y ventas

- Enseñar modelado de entidades con atributos múltiples

****Relaciones****

- Se vincula con `compras_productos` y `ventas_productos`

- Participa en vistas como `vista_control_inventario`

Tabla `compras`

****Descripción****

Registra cada operación de compra realizada a un proveedor.

****Objetivo****

Documentar el abastecimiento, permitiendo:

- Auditar compras por fecha, proveedor y empleado

- Calcular totales económicos

- Enseñar inserciones encadenadas

****Relaciones****

- Se vincula con `proveedores` y `empleados`

- Participa en procedimientos como `registrar_compra`

Tabla `ventas`

****Descripción****

Registra cada operación de venta realizada a un cliente.

****Objetivo****

Formalizar el acto comercial, permitiendo:

- Documentar cliente, vendedor, medio de pago y total

- Enseñar trazabilidad relacional

- Integrar lógica de cuotas

****Relaciones****

- Se vincula con `clientes`, `medios_de_pago`, `empleados`

- Participa en procedimientos como `registrar_venta_con_cuotas`

Tabla `compras_productos`

****Descripción****

Relaciona productos con compras específicas, detallando cantidad y precio unitario.

****Objetivo****

Descomponer cada compra en sus componentes, permitiendo:

- Calcular totales por producto

- Enseñar relaciones N:M

- Auditar el flujo de entrada de stock

****Relaciones****

- Se vincula con `compras` y `productos`

- Participa en vistas como `vista_compras_detalladas`

Tabla `ventas_productos`

****Descripción****

Relaciona productos con ventas específicas, detallando cantidad y precio unitario.

Objetivo

Descomponer cada venta en sus componentes, permitiendo:

- Calcular ingresos por producto

- Enseñar relaciones N:M

- Auditar el flujo de salida de stock

Relaciones

- Se vincula con `ventas` y `productos`

- Participa en vistas como `vista_ventas_detalladas`

Tabla `cuotas_pago`

Descripción

Registra el esquema de cuotas pactadas en una venta, incluyendo cantidad y medio de pago.

Objetivo

Formalizar compromisos financieros fraccionados, permitiendo:

- Controlar pagos mensuales

- Enseñar lógica de fragmentación

- Auditar compromisos económicos

Relaciones

- Se vincula con `medios_de_pago`

- Participa en procedimientos como `registrar_venta_con_cuotas`

Tabla `roles`

Descripción

Define los roles laborales dentro de la empresa.

Objetivo

Establecer jerarquías y funciones, permitiendo:

- Controlar permisos y responsabilidades

- Enseñar modelado de estructuras organizativas

****Relaciones****

- Se vincula con `empleados`

Tabla `empleados`

****Descripción****

Registra los datos de cada empleado, incluyendo su rol.

****Objetivo****

Identificar al personal operativo, permitiendo:

- Auditar compras y ventas por empleado

- Enseñar relaciones jerárquicas

****Relaciones****

- Se vincula con `roles`, `compras`, `ventas`

- Participa en procedimientos como `ObtenerResumenVentasPorEmpleado`

Tabla `auditoria_ventas`

****Descripción****

Registra automáticamente cada venta realizada, con metadatos de auditoría.

****Objetivo****

Preservar la trazabilidad histórica, permitiendo:

- Auditar inserciones en `ventas`

- Enseñar uso de triggers `AFTER INSERT`

****Relaciones****

- Se vincula con `ventas` y `clientes`

- Activada por el trigger `auditar_venta`

Tabla `auditoria_cuotas`

****Descripción****

Registra automáticamente cada cuota generada, con metadatos de auditoría.

****Objetivo****

Preservar la trazabilidad de pagos fraccionados, permitiendo:

- Auditar inserciones en `cuotas_pago`

- Enseñar uso de triggers `AFTER INSERT`

****Relaciones****

- Se vincula con `cuotas_pago`

- Activada por el trigger `auditar_cuota_generada`

Relaciones entre tablas resumidas

- La tabla `compras_productos` se vincula con `Compras` y `Productos` mediante `ID_Compra` y `ID_Producto`.

- La tabla `ventas_productos` se vincula con `Ventas` y `Productos` mediante `ID_Venta` y `ID_Producto`.

- La tabla `Ventas` se vincula con `Clientes` mediante `ID_Cliente`, indicando el cliente que realiza la compra.

Listado de Vistas

Vista `vista_ventas_detalladas`

****Descripción:****

Esta vista consolida de manera estructurada y precisa cada venta registrada en nuestra base de datos. Presenta información esencial como la fecha de la operación, los productos adquiridos, los datos del cliente y el medio de pago utilizado. Su construcción refleja buenas prácticas en el modelado relacional,

facilitando tanto el análisis comercial como la enseñanza de conceptos clave en gestión de datos y trazabilidad de transacciones.

****Objetivo:****

Permitir una identificación rápida y precisa de los elementos clave de cada operación de venta: quién fue el cliente, qué productos adquirió, cuántas unidades compró, en qué fecha se realizó la transacción, a qué precio se vendieron los productos y cuál fue el medio de pago utilizado. Esta vista no solo optimiza el análisis comercial, sino que también ejemplifica cómo estructurar consultas orientadas a la trazabilidad, la toma de decisiones y la enseñanza de modelos de gestión relacional.

****Tablas que componen la vista:****

- `ventas`: permite identificar la operación realizada.

- `ventas_productos`: detalla los productos vendidos en cada operación.

- `productos`: permite extraer el nombre del producto.

- `clientes`: contiene los datos del cliente, como nombre, apellido o si se trata de un cliente eventual.

- `medios_de_pago`: muestra cómo se efectuó el pago.

Vista `vista_compras_detalladas`

****Descripción:****

Esta vista permite visualizar de manera estructurada y exhaustiva cada compra registrada en nuestra base de datos. Presenta información clave como la fecha de la operación, los productos adquiridos, los datos del proveedor, la forma de contacto y otros detalles relevantes. Su diseño busca no solo facilitar el análisis técnico, sino también ofrecer una herramienta didáctica que ejemplifica cómo integrar múltiples fuentes de información en un modelo relacional coherente y funcional.

****Objetivo:****

Brindar una visualización clara y eficiente de cada operación de compra, permitiendo identificar con rapidez qué productos se adquirieron, en qué fecha, en qué cantidad, a qué precio y a qué proveedor fueron comprados. Esta vista facilita el seguimiento de decisiones comerciales y ejemplifica cómo estructurar consultas orientadas al análisis estratégico dentro de un entorno relacional.

****Tablas que componen la vista:****

- `compras`: identifica la operación.

- `compras_productos`: detalla qué se compró, en qué cantidad y a qué precio por unidad.

- `proveedores`: identifica a quién se le compró y proporciona un dato de contacto.

Vista `vista_productos_agrupado`

****Descripción:****

Esta vista permite visualizar los productos disponibles en nuestro sistema, incluyendo su descripción, costo y nivel de existencia actual. La información se presenta de forma independiente del proveedor, ya que un mismo producto puede estar asociado a múltiples proveedores. Este diseño facilita el análisis de stock y costos desde una perspectiva centralizada, ejemplificando cómo abstraer entidades clave en un modelo relacional sin perder flexibilidad operativa.

****Objetivo:****

Centralizar la información de productos disponibles, permitiendo conocer su descripción, existencia y costo, independientemente del proveedor asociado.

****Tabla que la compone:****

- `productos`: de esta tabla extraemos el producto, agrupándolo independientemente del proveedor. También obtenemos su descripción, el stock disponible y su precio.

Vista `vista_control_inventario`

****Descripción:****

Esta vista permite visualizar de manera estructurada y precisa el estado actual del inventario, consolidando información clave sobre cada producto registrado en el sistema.

Presenta datos como el nombre, la descripción, el stock actual, el total comprado, el total vendido y un cálculo del stock ajustado.

Su diseño refleja buenas prácticas en modelado relacional, integrando múltiples fuentes para ofrecer una perspectiva clara y operativa del flujo de productos.

****Objetivo:****

Brindar una visualización integral del inventario, permitiendo identificar con rapidez:

- Qué productos están disponibles

- Cuánto se ha comprado y vendido de cada uno

- Cuál sería el stock recalculado considerando el historial de movimientos

****Tablas que componen la vista:****

- `productos`: contiene el identificador, nombre, descripción y stock actual de cada producto.

- `compras_productos`: permite calcular el total de unidades compradas por producto.

- `ventas_productos`: permite calcular el total de unidades vendidas por producto.

Vista `vista_inventario_por_proveedor`

****Descripción:****

Esta vista permite visualizar de manera estructurada el inventario actual segmentado por proveedor.

Integra información clave sobre cada producto, su descripción, el total comprado y vendido, el stock actual y un cálculo del stock ajustado.

Además, vincula cada producto con su proveedor de origen, permitiendo rastrear el flujo de bienes desde su fuente hasta su estado actual.

Su diseño refleja una lectura relacional del abastecimiento, útil tanto para análisis logístico como para documentación pedagógica.

****Objetivo:****

Brindar una visualización clara y estratégica del inventario según proveedor, permitiendo identificar:

- Qué productos fueron adquiridos a cada proveedor

- Cuánto se ha comprado y vendido de cada uno

- Cuál es el stock actual y el stock recalculado considerando el historial de movimientos

****Tablas que componen la vista:****

- `productos`: contiene el identificador, nombre, descripción y stock actual de cada producto.

- `compras_productos`: vincula productos con operaciones de compra.

- `compras`: vincula cada compra con su proveedor.

- `proveedores`: identifica el origen comercial de cada producto.

- `ventas_productos`: permite calcular el total de unidades vendidas por producto.

Vista `vista_ventas_detalladas`

****Descripción:****

Esta vista permite visualizar de forma detallada cada operación de venta realizada, integrando múltiples dimensiones del proceso comercial.

Incluye información sobre la fecha de la venta, el producto vendido, la cantidad y precio unitario, el medio de pago utilizado y los datos del cliente.

Su diseño refleja una lectura relacional del ciclo de ventas, útil tanto para análisis financiero como para enseñanza de integraciones SQL complejas.

Cada fila representa un instante de intercambio, un nodo en la red de relaciones entre producto, cliente y forma de pago.

****Objetivo:****

Brindar una visualización clara y completa de las ventas realizadas, permitiendo identificar:

- Qué productos fueron vendidos, en qué cantidad y a qué precio

- Qué clientes realizaron las compras y cómo pagaron

- Cuándo se realizaron las transacciones y bajo qué condiciones

****Tablas que componen la vista:****

- `ventas`: contiene el identificador de la venta, la fecha y el vínculo con cliente y medio de pago.

- `ventas_productos`: vincula cada venta con los productos vendidos, incluyendo cantidad y precio unitario.

- `productos`: aporta el nombre del producto vendido.

- `medios_de_pago`: identifica el tipo de pago utilizado en la transacción.

- `clientes`: aporta el nombre y apellido del cliente que realizó la compra.

Lista de Stored Procedures

Procedimiento `ObtenerResumenComprasPorEmpleado`

****Descripción****

Este procedimiento almacenado permite obtener un resumen cuantitativo y financiero de las compras realizadas por cada empleado registrado como comprador.

Integra datos clave como el nombre completo del empleado, la cantidad total de compras realizadas, el monto acumulado y el promedio por operación.

Su diseño refleja una lectura relacional entre identidad laboral y comportamiento de compra, útil tanto para auditoría interna como para enseñanza de agregaciones en SQL.

****Objetivo****

Brindar una visión sintética y estratégica del comportamiento de compra por empleado, permitiendo identificar:

- Qué empleados han realizado compras y con qué frecuencia

- Cuál ha sido el monto total y el promedio de sus operaciones

- Cómo se distribuyen las compras dentro del cuerpo laboral

Este procedimiento facilita el seguimiento de responsabilidades comerciales, la evaluación de patrones internos y la enseñanza de funciones agregadas con agrupamiento.

****Tablas que componen el procedimiento****

- `compras`: contiene el total de cada operación y el identificador del comprador.

- `empleados`: aporta el nombre completo del empleado vinculado a cada compra mediante su legajo.

Procedimiento `ObtenerResumenVentasPorEmpleado`

****Descripción****

Este procedimiento almacenado permite obtener un resumen estratégico de las ventas realizadas por cada empleado registrado como vendedor.

Integra métricas clave como el nombre completo del empleado, la cantidad de ventas efectuadas, el monto total acumulado y el promedio por operación.

Su diseño refleja una lectura relacional entre identidad laboral y desempeño comercial, útil tanto para análisis de productividad como para enseñanza de funciones agregadas en SQL.

****Objetivo****

Brindar una visión sintética y comparativa del desempeño de ventas por empleado, permitiendo identificar:

- Qué empleados han participado activamente en el proceso de ventas

- Cuántas operaciones han realizado y con qué volumen económico

- Cuál es el promedio de venta por transacción individual

Este procedimiento facilita el seguimiento del rendimiento comercial, la evaluación de perfiles internos y la enseñanza de agrupamientos con funciones agregadas.

****Tablas que componen el procedimiento****

- `ventas`: contiene el total de cada operación y el identificador del vendedor.

- `empleados`: aporta el nombre completo del empleado vinculado a cada venta mediante su legajo.

Subrutina de automatización

Procedimiento `registrar_compra`

****Descripción****

Este procedimiento almacenado permite registrar una operación de compra entre el sistema y un proveedor determinado.

Integra dos momentos clave:

1. La creación del registro principal en la tabla `compras`.

2. La asociación detallada del producto adquirido en `compras_productos`.

Su diseño refleja una lectura relacional entre proveedor, producto y valor económico, útil tanto para trazabilidad operativa como para enseñanza de inserciones encadenadas en SQL.

****Objetivo****

Formalizar el acto de adquisición de productos, permitiendo documentar:

- Qué proveedor realizó la entrega

- Qué producto fue adquirido, en qué cantidad y a qué precio

- Cuál fue el total económico de la operación

Este procedimiento facilita:

- La trazabilidad de compras

- La gestión de inventario

- La enseñanza de inserciones con recuperación de claves primarias mediante `LAST_INSERT_ID()`

****Tablas que componen el procedimiento****

- `compras`: registra el proveedor, la fecha de la operación y el total económico.

- `compras_productos`: documenta el detalle del producto adquirido, su cantidad y precio unitario, vinculado a la compra principal.

Procedimiento `registrar_venta`

****Descripción****

Este procedimiento almacenado permite registrar una operación de venta entre el sistema y un cliente determinado.

Integra dos momentos clave:

1. La creación del registro principal en la tabla `ventas`, incluyendo el cliente, medio de pago, vendedor, fecha y total.

2. La asociación detallada del producto vendido en `ventas_productos`, documentando cantidad y precio unitario.

Su diseño refleja una lectura relacional entre cliente, producto y valor económico, útil tanto para trazabilidad comercial como para enseñanza de inserciones encadenadas en SQL.

****Objetivo****

Formalizar el acto de intercambio comercial, permitiendo documentar:

- Qué cliente realizó la compra

- Qué producto fue adquirido, en qué cantidad y a qué precio

- Cuál fue el total económico de la operación

- Qué vendedor facilitó la transacción y bajo qué medio de pago

Este procedimiento facilita:

- La trazabilidad de ventas

- La gestión de productos vendidos

- La enseñanza de inserciones con recuperación de claves primarias mediante `LAST_INSERT_ID()`

****Tablas que componen el procedimiento****

- `ventas`: registra el cliente, medio de pago, vendedor, fecha y total económico.

- `ventas_productos`: documenta el detalle del producto vendido, su cantidad y precio unitario, vinculado a la venta principal.

Procedimiento `registrar_venta_update` -- Creado este procedimiento para no eliminar el de registro de venta qeu es el qeu lo remplaza, dejo los dos ya que este lo cree como un UPdate no como remplazo al ser un ejercicio.

****Descripción****

Este procedimiento almacenado permite registrar una operación de venta entre el sistema y un cliente determinado, incorporando control de stock antes de confirmar la transacción. Integra tres momentos clave:

1. ****Verificación de stock****: consulta si el producto solicitado tiene suficiente cantidad disponible.

2. ****Creación del registro principal**** en la tabla `ventas`, incluyendo cliente, medio de pago, vendedor, fecha y total.

3. ****Asociación detallada del producto vendido**** en `ventas_productos`, documentando cantidad y precio unitario, seguido de la actualización del inventario.

Su diseño refleja una lectura relacional entre cliente, producto y valor económico, útil tanto para trazabilidad comercial como para enseñanza de inserciones encadenadas con validación previa en SQL.

****Objetivo****

Formalizar el acto de intercambio comercial, permitiendo documentar:

- Qué cliente realizó la compra

- Qué producto fue adquirido, en qué cantidad y a qué precio

- Cuál fue el total económico de la operación

- Qué vendedor facilitó la transacción y bajo qué medio de pago

- Si el stock disponible permite concretar la operación

Este procedimiento facilita:

- La trazabilidad de ventas

- La gestión de productos vendidos

- La enseñanza de inserciones con recuperación de claves primarias mediante `LAST_INSERT_ID()`

- La validación operativa previa a la modificación de inventario

Tablas que componen el procedimiento

- `productos`: consulta el stock disponible antes de registrar la venta.

- `ventas`: registra el cliente, medio de pago, vendedor, fecha y total económico.

- `ventas_productos`: documenta el detalle del producto vendido, su cantidad y precio unitario, vinculado a la venta principal.

Procedimiento `registrar_venta_con_cuotas`

Descripción

Este procedimiento almacenado permite registrar una operación de venta fraccionada entre el sistema y un cliente determinado.

Integra tres momentos clave:

1. La creación del registro principal en la tabla `ventas`, incluyendo cliente, medio de pago, vendedor, fecha y total.

2. La asociación detallada del producto vendido en `ventas_productos`, documentando cantidad y precio unitario.

3. La generación automática de cuotas en la tabla `pagos_cuotas`, distribuyendo el total en pagos mensuales con estado inicial `Pendiente`.

Su diseño refleja una lectura relacional entre cliente, producto, vendedor y compromiso económico extendido, útil tanto para trazabilidad comercial como para enseñanza de ciclos iterativos y funciones de fecha en SQL.

Objetivo

Formalizar el acto de intercambio comercial fraccionado, permitiendo documentar:

- Qué cliente realizó la compra

- Qué producto fue adquirido, en qué cantidad y a qué precio

- Cuál fue el total económico de la operación

- Qué vendedor facilitó la transacción y bajo qué medio de pago

- Cómo se distribuye el compromiso económico en cuotas mensuales

Este procedimiento facilita:

- La trazabilidad de ventas fraccionadas

- La gestión de compromisos financieros en cuotas

- La enseñanza de inserciones iterativas, cálculos proporcionales y funciones de fecha en SQL

****Tablas que componen el procedimiento****

- `ventas`: registra el cliente, medio de pago, vendedor, fecha y total económico.

- `ventas_productos`: documenta el detalle del producto vendido, su cantidad y precio unitario, vinculado a la venta principal.

- `pagos_cuotas`: registra cada cuota generada, con número, monto, fecha de vencimiento y estado inicial.

Lista de Funciones

Función `fn_stock_actual`

****Descripción****

Esta función calcula el stock actual de un producto específico, integrando dinámicamente las entradas por compras y las salidas por ventas.

Opera como una lectura ritual del flujo de mercancías, permitiendo obtener el saldo neto de unidades disponibles en tiempo real.

Su diseño refleja una síntesis entre movimiento histórico y estado presente, útil tanto para control logístico como para enseñanza de funciones escalares en SQL.

****Objetivo****

Brindar un cálculo preciso y actualizado del stock disponible de un producto, permitiendo:

- Consultar el saldo neto entre compras y ventas

- Integrar dinámicamente el historial de movimientos

- Utilizar la función en vistas, procedimientos o reportes que requieran trazabilidad de inventario

Esta función facilita el seguimiento de disponibilidad, la prevención de quiebres de stock y la enseñanza de funciones con lógica condicional y agregada.

Tablas que componen la función

- `compras_productos`: registra las cantidades adquiridas por producto.

- `ventas_productos`: registra las cantidades vendidas por producto.

Función `fn_total_compras_x_proveedor`

Descripción

Esta función calcula el monto total de compras realizadas a un proveedor específico, integrando el historial de transacciones registradas en la tabla `compras`.

Opera como una lectura del vínculo comercial entre la organización y sus fuentes de abastecimiento, permitiendo obtener el volumen económico asociado a cada proveedor.

Su diseño refleja una síntesis entre identidad comercial y flujo financiero, útil tanto para evaluación de proveedores como para enseñanza de funciones escalares con agregación.

Objetivo

Brindar un cálculo preciso del total de compras realizadas a un proveedor determinado, permitiendo:

- Consultar el volumen económico asociado a cada proveedor

- Integrar dinámicamente el historial de compras en reportes o vistas

- Evaluar la relevancia comercial de cada vínculo de abastecimiento

Esta función facilita el seguimiento de relaciones comerciales, la toma de decisiones estratégicas y la enseñanza de funciones con parámetros y lógica condicional.

Tablas que componen la función

- `compras`: contiene el total de cada operación y el identificador del proveedor.

Función `fn_total_cuotas_x_venta`

****Descripción****

Esta función calcula el monto total de cuotas asociadas a una venta específica, integrando los pagos fraccionados registrados en la tabla `cuotas_pago`.

Opera como una lectura del compromiso financiero asumido en cada transacción, permitiendo obtener el saldo total pactado en forma de cuotas.

Su diseño refleja una síntesis entre fragmentación de pago y totalidad económica, útil tanto para control financiero como para enseñanza de funciones escalares con agregación.

****Objetivo****

Brindar un cálculo preciso del total de cuotas correspondientes a una venta determinada, permitiendo:

- Consultar el monto total comprometido en pagos fraccionados

- Integrar dinámicamente el historial de cuotas en reportes o vistas

- Evaluar la magnitud financiera de cada transacción más allá del pago único

Esta función facilita el seguimiento de compromisos financieros, la planificación de ingresos y la enseñanza de funciones con parámetros y lógica condicional.

****Tablas que componen la función****

- `cuotas_pago`: contiene el monto de cada cuota y el identificador de la venta asociada.

Lista de Triggers

Trigger auditar_venta

****Descripción****

Este trigger se activa automáticamente después de cada inserción en la tabla `ventas`, generando un registro paralelo en la tabla `auditoria_ventas`. Su propósito es conservar una huella histórica de cada transacción comercial, asegurando trazabilidad y control sobre los datos ingresados. Opera como un mecanismo de auditoría pasiva, donde cada venta se replica en un entorno de

control sin intervención manual, integrando metadatos como el timestamp y el origen del evento.

****Objetivo****

Garantizar la integridad histórica de las operaciones de venta mediante un registro automático que permita:

- Auditar transacciones sin alterar el flujo principal de datos

- Conservar evidencia de cada inserción en la tabla `ventas`

- Facilitar el análisis posterior de ventas desde una perspectiva temporal y operativa

- Enseñar el uso de triggers como herramientas de control y automatización en bases de datos relacionales

Este trigger representa una práctica común en sistemas que requieren trazabilidad, siendo útil tanto en entornos académicos como profesionales para ilustrar el uso de eventos `AFTER INSERT`.

****Tablas que componen el trigger****

- ****ventas****: Tabla principal donde se registran las transacciones comerciales.

- ****auditoria_ventas****: Tabla auxiliar que conserva una copia de cada venta, junto con metadatos de auditoría como `Timestamp_Auditoria` y `Origen_Auditoria`.

Trigger auditar_cuota_generada

****Descripción****

Este trigger se ejecuta automáticamente después de cada inserción en la tabla `cuotas_pago`, generando un registro espejo en la tabla `auditoria_cuotas`. Su función es preservar la trazabilidad de cada cuota generada, asegurando que toda fragmentación de pago quede documentada en un entorno de control. Actúa como un mecanismo de auditoría silenciosa, donde cada cuota se replica con sus atributos esenciales, sin intervención externa.

****Objetivo****

Establecer un registro automático de cada cuota ingresada, permitiendo:

- Auditar la generación de cuotas en tiempo real

- Conservar evidencia de cada inserción en la tabla `cuotas_pago`

- Facilitar el análisis posterior de pagos fraccionados desde una perspectiva operativa

- Enseñar el uso de triggers como herramientas de automatización y control en bases de datos

Este trigger es especialmente útil en sistemas que gestionan pagos escalonados, y sirve como ejemplo pedagógico para ilustrar eventos `AFTER INSERT` en contextos de auditoría.

****Tablas que componen el trigger****

- ****cuotas_pago****: Tabla principal donde se registran las cuotas asociadas a transacciones.

- ****auditoria_cuotas****: Tabla auxiliar que conserva una copia de cada cuota generada, incluyendo su identificador, medio de pago y cantidad de cuotas.

Resumen de roles creados en el sistema

Rol Gerente

- ****Tipo****: Rol sin login, usado para agrupar permisos administrativos.

- ****Usuario asignado****: `gerente_user`

- ****Permisos****:

  - Acceso total a todas las tablas del esquema público (`SELECT`, `INSERT`, `UPDATE`, `DELETE`)

  - Ejecución de funciones y procedimientos

  - Lectura de vistas

- ****Alcance****: Supervisión integral del sistema, con capacidad de consulta, modificación y ejecución. Ideal para el perfil ejecutivo de la Base.

Rol Ventas

****Tipo****: Rol sin login, enfocado en operaciones comerciales.

****Usuario asignado****: `ventas_user`

Permisos:

- Inserción en tablas de ventas (`ventas`, `detalle_ventas`, `clientes`)

- Lectura de clientes (para validar antes de vender)

- Ejecución de funciones como `registrar_venta()`, `registrar_venta_update()` y `calcular_total_venta()`

- Acceso a vistas como `vista_ventas_diarias` y `vista_clientes_activos`

- **Alcance**: Gestión operativa de ventas, con acceso limitado a datos y funciones específicas. No puede modificar ni eliminar registros fuera de su ámbito.

Rol Compras

- **Tipo**: Rol sin login, orientado a abastecimiento y proveedores.

- **Usuario asignado**: `compras_user`

- **Permisos**:

  - Inserción en tablas de compras (`compras`, `detalle_compras`, `proveedores`)

  - Lectura de proveedores

  - Ejecución de funciones como `registrar_compra()` y `calcular_total_compra()`

  - Acceso a vistas como `vista_compras_mensuales` y `vista_proveedores_activos`

- **Alcance**: Control de adquisiciones, con permisos específicos para registrar y consultar operaciones de compra.

Rol RRHH

- **Tipo**: Rol sin login, centrado en gestión de personal.

- **Usuario asignado**: `rrhh_user`

- **Permisos**:

  - Lectura (`SELECT`) en todas las tablas y vistas del esquema público

  - Modificación (`UPDATE`) en todas las tablas

- **Sin permisos de creación (`INSERT`), eliminación (`DELETE`) ni ejecución (`EXECUTE`)**

- **Alcance**: Supervisión y actualización de datos relacionados con empleados, sin capacidad de alterar la estructura ni ejecutar funciones. Ideal para roles de seguimiento y control interno.

Síntesis de la Base de Datos `cuchuflito_sa`

Esta sección resume la arquitectura relacional, funcional y pedagógica del sistema desarrollado para la base, integrando sus componentes en una visión unificada.

Componentes estructurales

- **Tablas base**: `clientes`, `productos`, `empleados`, `proveedores`, `medios_de_pago`

- **Tablas operativas**: `ventas`, `compras`, `ventas_productos`, `compras_productos`

- **Tablas de compromiso financiero**: `cuotas_pago`

- **Tablas de auditoría**: `auditoria_ventas`, `auditoria_cuotas`

- **Tablas organizativas**: `roles`

Cada tabla responde a una necesidad concreta del modelo de negocio, y se vincula mediante claves foráneas que permiten trazabilidad, control y análisis.

Componentes funcionales

- **Vistas**: permiten observar el sistema desde perspectivas analíticas, como ventas por cliente, stock por producto o resumen por empleado.

- **Procedimientos almacenados**: automatizan operaciones complejas como ventas fraccionadas, compras detalladas o auditorías.

- **Funciones**: encapsulan cálculos y transformaciones específicas, como totales, fechas de vencimiento o validaciones.

- **Triggers**: garantizan la trazabilidad automática de eventos clave, como la creación de una venta o una cuota.

Lógica relacional y pedagógica

El sistema refleja una lógica relacional donde cada entidad se conecta con otras en función de su rol comercial. Esta arquitectura permite:

- **Trazabilidad completa** de operaciones

- **Escalabilidad técnica** para futuras integraciones

- **Adaptabilidad funcional** a distintos escenarios de negocio

Valor educativo (honestamente chamuyo)

Este sistema no solo organiza datos: **enseña a pensar en relaciones, procesos y decisiones**. Cada componente fue diseñado con intención del crecimiento y aprendizaje personal, permitiendo:

- Comprender el ciclo completo de una operación comercial

- Aplicar funciones de fecha, iteración y validación

- Estudiar el impacto de cada decisión técnica en la experiencia del usuario

Este módulo fue construido desde la experiencia de aprender haciendo, documentando y compartiendo, sin ánimos de atribuirle más valor pedagógico que el de una mera reflexión sobre este proceso de aprendizaje.

Conclusión

`cuchuflito_sa` es más que una base de datos: es una infraestructura viva, pensada para crecer, adaptarse y compartir esta experiencia formativa. Su diseño refleja una lectura estratégica del negocio, una ejecución técnica rigurosa y un compromiso con la documentación.