



INNOVATECH

Data Base



ESTUDIANTE: Fabricio Papetti

Profesor: Sergio Gabriel Oberenko

Tutor/a: Gabriel Bertella

ÍNDICE

Proyecto: “INNOVATECH”

Descripción de la temática	3
Diagrama Entidad – Relación	5
Diagrama Entidad – Relación MySQL Workbench	6
Listado de tablas	7
Vistas	12
Funciones	20
Procedimientos de almacenado	25
Triggers	29
Usuarios	32
Herramientas y tecnologías utilizadas	33

DESCRIPCIÓN DE LA TEMÁTICA

INTRODUCCIÓN

Para este proyecto se realizó una base de datos de una empresa ficticia llamada “Innovatech”, dedicada a la venta de electrodomésticos y dispositivos tecnológicos con el objetivo de llevar un registro y control de clientes, sucursales, empleados, productos y ventas.

OBJETIVOS

El proyecto busca poder establecer un registro actualizado de los empleados activos en la empresa, de los productos existentes en stock, de sus clientes y principalmente de sus ventas, brindando una amplia gama de información en cada uno de los elementos mencionados. Todo ello, con el fin de convertir a dicho registro en la principal fuente de información del hombre de negocios en su día a día.

SITUACIÓN PROBLEMÁTICA

El ejercicio de continuo de la actividad comercial trae aparejada la constante recepción de información, ya sean datos de productos, de empleados, de clientes, de ventas, etc. Por ende, ¿Cuál es el método más seguro de registro y almacenado de tales datos? La implementación de una base de datos en una empresa de estas características resulta esencial a fin de tener una fuente de información confiable y segura, desde la cual se registre el total del volumen de negocios que se desprenda del ejercicio habitual de la actividad mencionada

MODELO DE NEGOCIO

La organización del modelo planteado para la base de datos de "Innovatech" busca recopilar, almacenar, y administrar información sobre sus miembros, ya sean clientes, productos o empleados; almacenando y llevando un registro de sus respectivos datos personales y características como así también de la ventas que sean concretadas.

Como se ha mencionado anteriormente el objetivo de la presente base de datos es la utilización de la información recopilada a raíz de la ocurrencia de los actos típicos de la actividad comercial, sirviendo de referencia a fin de poder visualizar aquellos empleados que concreten más ventas, aquellos productos que sean más vendidos como así también los menos vendidos, las sucursales con mayor o menor cantidad de empleados, los ingresos obtenidos en un determinado periodo de tiempo, etc.

DIAGRAMA ENTIDAD – RELACIÓN

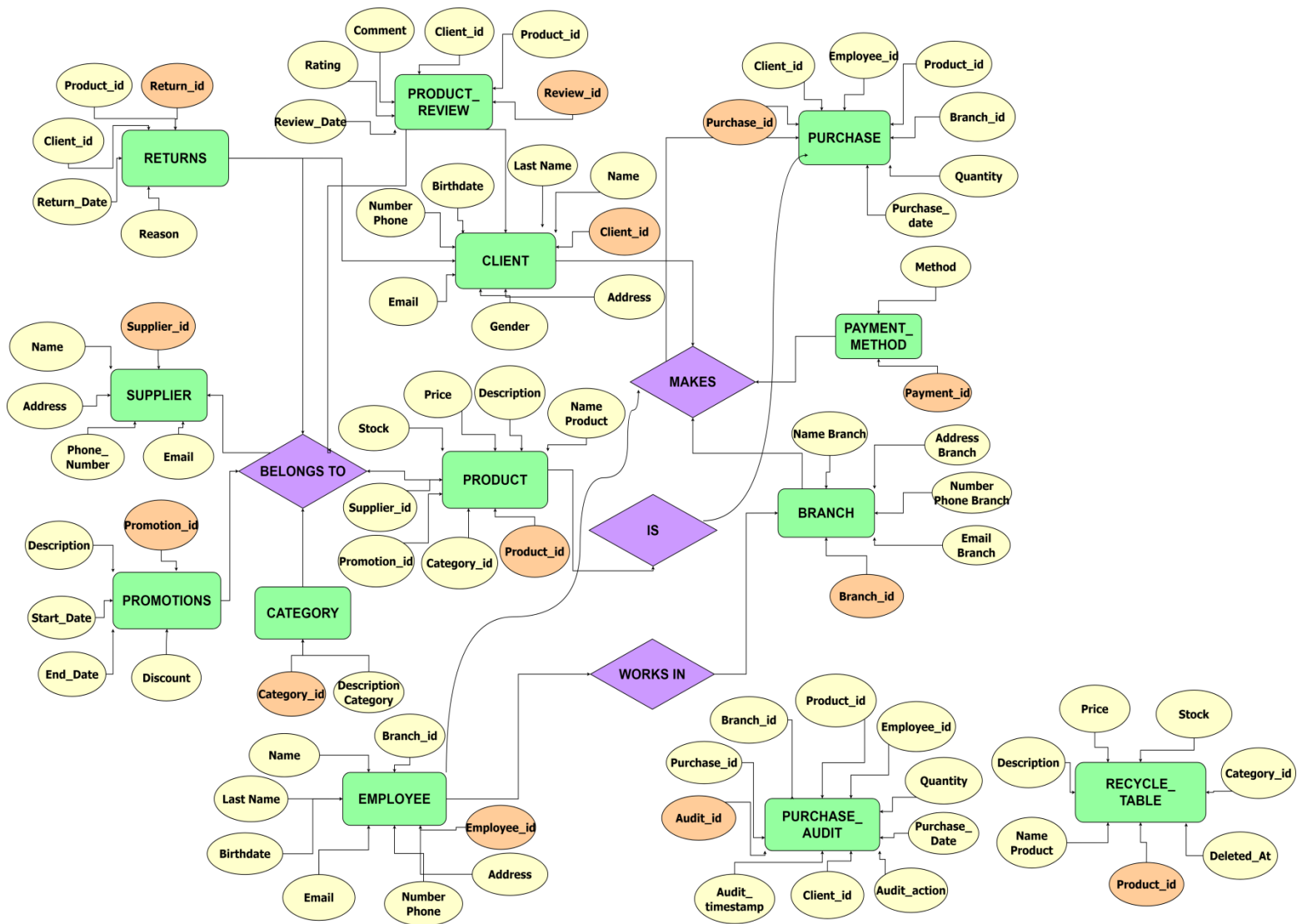
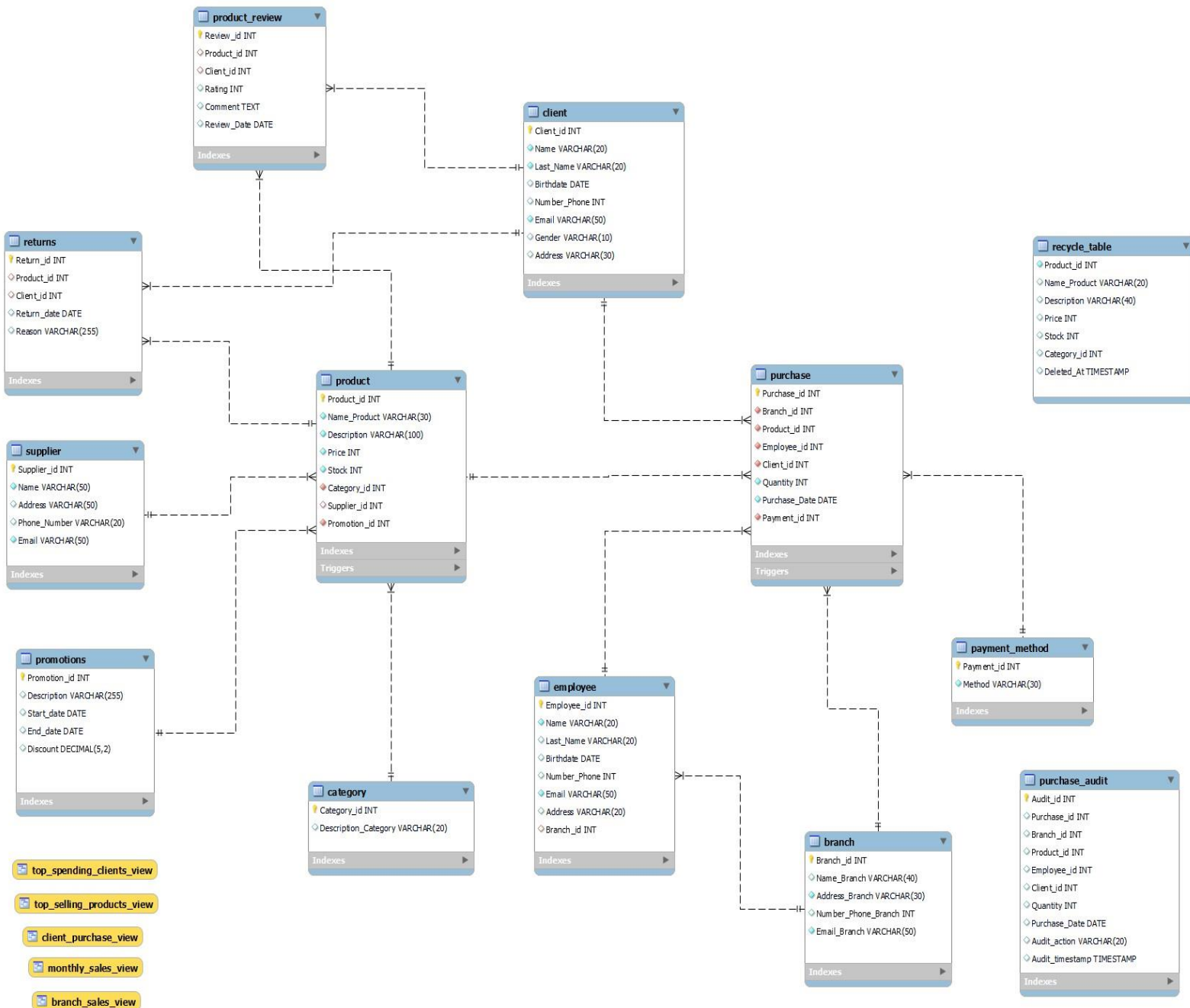







DIAGRAMA ENTIDAD – RELACIÓN

MySQL WORKBENCH







LISTADO DE TABLAS

Tabla	Descripción	Campos	Tipos de datos	Claves Primarias	Claves Foráneas
Client	Registra todos los clientes de la empresa.	1) Name 2) Last_Name 3) BirthDate 4) Number_Phone 5) Email 6) Gender 7) Address 8) Client_id	1) varchar 2) varchar 3) date 4) Int 5) varchar 6) varchar 7) varchar 8) int	Client_id	X
Product	Registra todos los productos existentes.	1) Name_Product 2) Description 3) Price 4) Stock 5) Category_id 6) Product_id	1) varchar 2) varchar 3) int 4) int 5) int 6) int	Product_id	Category_id Reference Table  category (Category_id) <hr/> Supplier_id Reference Table  supplier (Supplier_id) <hr/>

					Promotion_id Reference Table  promotions (Promotion_id)
Employee	Registra todos los empleados de la empresa.	1) Name 2) Last_Name 3) Bitthdate 4) Number_Phone 5) Email 6) Address 7) Employee_id	1) varchar 2) varchar 3) date 4) int 5) varchar 6) varchar 7) int	Employee_id	Branch_id Reference Table  branch (Branch_id)
Category	Registra los diferentes tipos de categorías de los productos.	1) Description_Category 2) Category_id	1) varchar 2) int	Category_id	X
Branch	Registra las diferentes sucursales de la empresa.	1) Name_Branch 2) Address_Branch 3) Number_Phone_Branch 4) Email_Branch 5) Branch_id	1) varchar 2) varchar 3) int 5) varchar 6) int	Branch_id	X
Purchase	Registra las ventas realizadas.	1) Branch_id 2) Product_id 3) Employee_id 4) Client_id 5) Quantity 6) Purchase_Date 7) Purchase_id	1) int 2) int 3) int 4) int 5) int 6) date 7) Int	Purchase_id	Branch_id Reference Table  branch (Branch_id) <hr/>

					<div>Product_id Reference Table ↓ product (Product_id)</div> <hr/> <div>Employee_id Reference Table ↓ employee (Employee_id)</div> <hr/> <div>Client_id Reference Table ↓ client (Client_id)</div> <hr/> <div>Payment_id Reference Table ↓ payment_method (Payment_id)</div>
Payment_Method	Registrar los diferentes métodos de pago.	1) Payment_id 2) Method	1) Int 2) varchar	Payment_Id	X

Supplier	Registrar los diferentes proveedores.	1) Supplier_id 2) Name 3) Address 4) Phone_Number 5) Email	1) Int 2) varchar 3) varchar 4) Int 5) varchar	Supplier_id	X
Product – review	Registrar los comentarios o valoraciones de los clients sobre un producto.	1) Review_id 2) Product_id 3) Client_id 4) Rating 5) Coment 6) Review_Date	1) Int 2) Int 3) Int 4) Int 5) Text 6) Date	Review_id	Product_id Reference Table  product (Product_id) <hr/> Client_id Reference Table  client (Client_id)
Returns	Registrar las devoluciones de productos realizadas por los clientes.	1) Return_id 2) Product_id 3) Client_id 4) Return_Date 5) Reason	1) Int 2) Int 3) Int 4) Date 5) varchar	Return_id	Product_id Reference Table  product (Product_id) <hr/> Client_id Reference Table  client (Client_id)
Promotions	Registrar las diferentes promociones	1) Promotion_id 2) Description 3) Start_Date 4) End_Date 5) Discount	1) Int 2) varchar 3) Date 4) Date 5) Decimal	Promotion_id	X

Recycle_Table	Registrar los productos eliminados de la table de productos	1) Product_id 2) Name_Product 3) Description 4) Price 5) Stock 6) Category_id 7) Deleted_At	1) Int 2) varchar 3) varchar 4) Int 5) Int 6) Int 7) TimeStamp	Product_id	X
Purchase_audit	Registrar los movimientos de la tabla de ventas de la empresa	1) Audit_id 2) Purchase_id 3) Branch_id 4) Product_id 5) Employee_id 6) Client_id 7) Quantity 8) Purchase_Date 9) Audit_action 10) Audit_timestamp	1) Int 2) Int 3) Int 4) Int 5) Int 6) Int 7) Int 8) Date 9) varchar 10) TimeStamp	Audit_id	X

VISTAS

Vista N° 1: 'client_purchase_view'

La vista `client_purchase_view` es diseñada para proporcionar una vista consolidada y fácil de entender que combina datos relevantes de clientes, sus compras, y los productos adquiridos. Esta vista facilita la consulta de información sin necesidad de escribir consultas complejas que involucren múltiples uniones cada vez que se necesita esta información. Aquí te detallo su objetivo y las tablas que la componen:

Objetivo de la Vista

El principal objetivo de `client_purchase_view` es ofrecer una visión integral que relaciona a los clientes con sus compras específicas y los detalles de los productos comprados. Esta vista es útil para:

- **Análisis de Comportamiento del Cliente:** Entender qué productos son comprados por cuáles clientes, lo que puede ayudar en el análisis de patrones de compra, preferencias de productos, y segmentación de clientes.
- **Reportes de Ventas:** Generar informes detallados sobre las ventas, incluyendo quién compró qué y cuándo, lo que es esencial para el seguimiento de desempeño de ventas y la planificación de inventario.
- **Marketing y Promociones:** Identificar oportunidades para campañas de marketing dirigidas, promociones, y ofertas personalizadas basadas en el historial de compras de los clientes.

Tablas Compuestas

La vista `client_purchase_view` se compone de las siguientes tablas de una base de datos relacional:

1. `client (c)`: Esta tabla almacena información sobre los clientes, como identificadores únicos, nombres, y otros datos personales. La vista incluye todas las columnas de la tabla `client (c.*)`, lo que significa que cada fila de la vista proporcionará una visión completa de los datos del cliente.
2. `purchase (pu)`: Contiene los registros de las compras realizadas, incluyendo el identificador del cliente (`Client_id`) que realizó la compra, el identificador del producto (`Product_id`) comprado, y la fecha de la compra (`Purchase_Date`). La tabla `purchase` sirve como el enlace entre los clientes y los productos que han adquirido.

3. product (p): Esta tabla guarda los detalles de los productos, como el identificador único del producto (Product_id), el nombre del producto (Name_Product), y el precio (Price). Estos detalles se incorporan en la vista para proporcionar información completa sobre los productos comprados en cada transacción.

Funcionamiento de la Vista

La vista realiza dos uniones (JOIN):

- Primero, une la tabla client con la tabla purchase utilizando el Client_id para relacionar a cada cliente con sus compras.
- Luego, une el resultado con la tabla product usando el Product_id para incluir detalles de los productos comprados en cada compra.

Al final, la vista client_purchase_view presenta una fila por cada compra realizada, incluyendo la información completa del cliente, los detalles del producto adquirido, y la fecha de la compra. Esto hace que la vista sea una herramienta valiosa para consultas rápidas y eficientes sobre las transacciones de compra de clientes sin necesidad de manejar directamente las complejidades de las uniones entre tablas.

Vista N° 2: 'branch_sales_view'

La vista branch_sales_view está diseñada para ofrecer una perspectiva consolidada de las ventas realizadas en diferentes sucursales de la empresa, detallando los productos vendidos, sus precios, y las fechas de venta. Esta vista facilita el acceso y análisis de las ventas por sucursal sin la necesidad de ejecutar múltiples y complejas consultas SQL que impliquen varias uniones entre tablas. A continuación, se detalla el objetivo de esta vista y las tablas que la componen:

Objetivo de la Vista

El principal objetivo de branch_sales_view es proporcionar una visión clara y accesible de las ventas de productos en cada sucursal, lo que permite:

- Análisis de Ventas por Sucursal: Comprender cómo se desempeñan las diferentes sucursales en términos de ventas, identificando qué productos son más populares y cómo varían las ventas en distintas ubicaciones.
- Gestión de Inventario: Ayudar en la gestión del inventario al proporcionar datos sobre los productos más vendidos en cada sucursal, permitiendo ajustes más precisos en la planificación del stock.

- Estrategias de Precios: Evaluar el desempeño de diferentes estrategias de precios entre las sucursales, analizando cómo el precio de los productos afecta las ventas en diferentes áreas geográficas.

Tablas Compuestas

La vista `branch_sales_view` integra datos de las siguientes tablas:

1. `branch (b)`: Contiene información sobre las sucursales de la empresa, incluyendo un identificador único (`Branch_id`) y el nombre de la sucursal (`Name_Branch`). La vista incluye el nombre de la sucursal para identificar de dónde provienen las ventas.
2. `purchase (pu)`: Registra las ventas realizadas, indicando la sucursal donde se efectuó la venta (`Branch_id`), el producto vendido (`Product_id`), y la fecha de la venta (`Purchase_Date`). Esta tabla sirve como enlace entre las sucursales y los productos vendidos.
3. `product (p)`: Almacena detalles sobre los productos, como el identificador único del producto (`Product_id`), el nombre (`Name_Product`), y el precio (`Price`). Estos detalles son incluidos en la vista para mostrar qué productos se venden y a qué precio.

Funcionamiento de la Vista

La vista realiza las siguientes operaciones de unión (JOIN):

- Primero, une la tabla `branch` con `purchase`, utilizando `Branch_id` para relacionar cada venta con la sucursal correspondiente.
- Luego, une el resultado con la tabla `product` mediante `Product_id` para añadir información detallada sobre los productos vendidos en cada transacción.

Como resultado, `branch_sales_view` presenta una fila por cada venta realizada, mostrando el nombre de la sucursal (`Name_Branch`), el nombre y precio del producto vendido (`Name_Product`, `Price`), y la fecha de la venta (`Purchase_Date`). Esta estructura hace que la vista sea una herramienta invaluable para realizar consultas rápidas sobre las ventas en las sucursales, facilitando la toma de decisiones basada en datos sobre gestión de inventario, estrategias de precios, y desempeño de ventas.

Vista N° 3: 'top_selling_products_view'

La vista `'top_selling_products_view'` está diseñada para proporcionar un análisis de los productos más vendidos, agrupando las ventas por producto y ordenándolos en función de su volumen total de ventas. Este tipo de vista es particularmente útil para la toma de decisiones estratégicas relacionadas con la gestión de inventario, la planificación de la producción, y la formulación de

estrategias de marketing y promociones. A continuación, se detalla el objetivo de esta vista y las tablas que la componen:

Objetivo de la Vista

El principal propósito de ``top_selling_products_view`` es ofrecer una visión clara de los productos que tienen un mayor éxito de ventas, permitiendo a la empresa:

- **Identificar Productos Populares:** Reconocer cuáles son los productos más vendidos y, por lo tanto, más populares entre los consumidores. Esta información es vital para asegurar que estos productos estén siempre disponibles y bien abastecidos.
- **Optimizar la Gestión de Inventario:** Ayudar en la planificación del inventario al centrar los esfuerzos de reabastecimiento en los productos más vendidos, evitando así el exceso de stock de productos menos populares.
- **Informar Estrategias de Marketing:** Orientar las campañas de marketing y promociones hacia los productos más vendidos para maximizar el retorno de inversión en actividades promocionales.
- **Planificación de la Producción:** Para empresas que fabrican sus propios productos, esta vista puede informar decisiones de producción, enfocando recursos en la fabricación de productos de alta demanda.

Tablas Compuestas

La vista ``top_selling_products_view`` integra datos de las siguientes tablas:

1. `product (`p`)`: Esta tabla contiene detalles sobre los productos ofrecidos por la empresa, incluyendo un identificador único del producto (``Product_id``) y el nombre del producto (``Name_Product``). Estos detalles son fundamentales para identificar los productos en la vista.
2. `purchase (`pu`)`: Registra las ventas de productos, especificando qué producto fue vendido (``Product_id``) y la cantidad vendida (``Quantity``). Esta tabla es esencial para calcular el volumen total de ventas de cada producto.

Funcionamiento de la Vista

La vista realiza las siguientes operaciones:

- Realiza una unión (``JOIN``) entre la tabla ``product`` y ``purchase`` utilizando ``Product_id`` para relacionar cada producto con sus ventas correspondientes.
- Agrupa los resultados por ``Product_id`` y ``Name_Product`` para consolidar las ventas de cada producto.
- Suma la cantidad vendida de cada producto (``SUM(pu.Quantity)``) para obtener el volumen total de ventas, denominado ``Total_Sales``.

- Ordena los resultados en función del volumen total de ventas (`Total_Sales`) de manera descendente, lo que permite que los productos más vendidos aparezcan primero en la lista.

Como resultado, `top_selling_products_view` presenta una lista de productos ordenada por su éxito de ventas, mostrando el identificador y nombre del producto (`Product_id`, `Name_Product`), junto con el volumen total de ventas (`Total_Sales`). Esta estructura facilita la identificación rápida de los productos más populares y es una herramienta valiosa para la gestión estratégica en diversas áreas de la empresa.

Vista N° 4: 'top_spending_clients_view'

La vista `top_spending_clients_view` está diseñada para ofrecer un análisis detallado de los clientes que han gastado más en compras, calculando el gasto total por cliente y ordenándolos desde el mayor al menor gasto. Esta vista es una herramienta esencial para entender mejor el comportamiento de compra de los clientes y para la toma de decisiones en áreas como marketing, ventas, y servicio al cliente. A continuación, se detalla el objetivo de esta vista y las tablas que la componen:

Objetivo de la Vista

El propósito principal de `top_spending_clients_view` es identificar a los clientes que representan el mayor valor en términos de gasto total en productos, permitiendo a la empresa:

- Reconocer Clientes Valiosos: Identificar a los clientes que contribuyen significativamente a los ingresos de la empresa a través de sus compras.
- Personalizar la Atención al Cliente: Dirigir esfuerzos para retener a estos clientes valiosos ofreciendo promociones, descuentos personalizados, o programas de lealtad.
- Orientar Estrategias de Marketing: Desarrollar campañas de marketing dirigidas específicamente a estos clientes, basadas en su historial de compras y preferencias.
- Análisis de Comportamiento de Compra: Comprender los patrones de compra de los clientes que gastan más, lo que puede ayudar a prever tendencias de ventas y ajustar la estrategia de stock.

Tablas Compuestas

La vista `top_spending_clients_view` se compone de las siguientes tablas:

1. `client`(`c`)`: Contiene información sobre los clientes, incluyendo un identificador único (``Client_id``), nombre (``Name``), y apellido (``Last_Name``). Estos detalles son esenciales para identificar y personalizar el análisis hacia los clientes individuales.
2. `purchase`(`pu`)`: Registra las ventas, especificando qué cliente realizó la compra (``Client_id``), qué producto fue comprado (``Product_id``), y la cantidad comprada (``Quantity``). Esta tabla es crucial para calcular el gasto total de cada cliente.
3. `product`(`p`)`: Proporciona detalles sobre los productos vendidos, incluido el precio (``Price``). Estos detalles son necesarios para calcular el valor monetario total de las compras de cada cliente.

Funcionamiento de la Vista

La vista realiza las siguientes operaciones:

- Une (``JOIN``) las tablas ``client``, ``purchase``, y ``product`` para relacionar cada compra con el cliente correspondiente y el producto comprado.
- Agrupa los resultados por cliente (``Client_id``, ``Name``, ``Last_Name``) para consolidar todas las compras realizadas por cada cliente.
- Calcula el gasto total (``SUM(p.Price * pu.Quantity)``) de cada cliente para determinar ``Total_Spending``, sumando el producto del precio de cada artículo por la cantidad comprada en todas sus compras.
- Ordena los clientes por su gasto total (``Total_Spending``) de forma descendente, lo que permite identificar rápidamente a los clientes que han gastado más.

Como resultado, ``top_spending_clients_view`` proporciona una lista detallada de clientes, mostrando su identificador, nombre, apellido, y el gasto total realizado (``Total_Spending``). Esta estructura ofrece una visión clara de quiénes son los clientes más valiosos en términos de ingresos generados para la empresa, facilitando la implementación de estrategias focalizadas en retener y satisfacer a estos clientes clave.

Vista N° 5: ‘monthly_sales_view’

La vista ``monthly_sales_view`` está diseñada para proporcionar un resumen mensual de las ventas, incluyendo tanto el total de productos vendidos como el ingreso total generado en cada mes. Es una herramienta valiosa para análisis de tendencias de ventas, planificación de inventario y estrategias de marketing. A continuación, se detalla el propósito de esta vista y las tablas que la componen:

Objetivo de la Vista

El propósito principal de `monthly_sales_view` es ofrecer una visión clara y organizada de la actividad de ventas de la empresa desglosada por mes y año, permitiendo a la empresa:

- Analizar Tendencias de Ventas: Observar cómo varían las ventas a lo largo del tiempo, identificando patrones estacionales o tendencias de crecimiento/decrecimiento.
- Planificación Financiera y de Inventario: Utilizar los datos de ventas históricas para prever la demanda futura, ajustar los niveles de stock adecuadamente y optimizar la planificación financiera.
- Evaluación de Estrategias de Marketing y Promociones: Medir el impacto de campañas de marketing y promociones específicas analizando cambios en el volumen de ventas y los ingresos generados en diferentes periodos.
- Establecimiento de Metas y Objetivos: Basar los objetivos de ventas futuros en el desempeño histórico, estableciendo metas realistas para crecimiento o recuperación.

Tablas Compuestas

La vista `monthly_sales_view` se compone de las siguientes tablas:

1. purchase (`pu`): Esta tabla registra las transacciones de venta, incluyendo la fecha de compra (`Purchase_Date`) y la cantidad de productos vendidos (`Quantity`). La fecha de la compra es esencial para agrupar las ventas por mes y año.
2. product (`p`): Proporciona detalles de los productos, incluido el precio (`Price`). El precio es crucial para calcular el ingreso total generado por las ventas de productos.

Funcionamiento de la Vista

La vista realiza las siguientes operaciones:

- Realiza una unión (`JOIN`) entre las tablas `purchase` y `product` para correlacionar cada venta con el producto correspondiente.
- Agrupa los resultados por año y mes de la fecha de compra (`YEAR(pu.Purchase_Date), MONTH(pu.Purchase_Date)`) para organizar las ventas y los ingresos de forma cronológica.
- Calcula el total de productos vendidos (`SUM(pu.Quantity)`) y el ingreso total (`SUM(p.Price * pu.Quantity)`) por cada grupo de año y mes, ofreciendo una medida de volumen y valor de las ventas respectivamente.
- Ordena los resultados por año y mes de forma ascendente, facilitando el análisis secuencial de las tendencias de ventas a lo largo del tiempo.

Como resultado, `monthly_sales_view` ofrece una vista consolidada que muestra el año, el mes, el total de productos vendidos (`Total_Products_Sold`), y el ingreso total generado (`Total_Revenue`) en cada periodo. Esta estructura proporciona una herramienta poderosa para el análisis financiero y operativo, ayudando a la empresa a tomar decisiones informadas basadas en el desempeño de ventas mensual.

FUNCIONES

Function N° 1: 'CalculateTotalSpent'

La función 'CalculateTotalSpent' está diseñada para calcular el total de dinero gastado por un cliente específico en todas sus compras. Es una herramienta útil para análisis de comportamiento de clientes, segmentación, y personalización de ofertas. A continuación, se detalla el propósito de esta función, las tablas que manipula e implementa, y cómo funciona:

Objetivo de la Función

El propósito principal de 'CalculateTotalSpent' es ofrecer una manera eficiente y directa de determinar el gasto total de un cliente individual en la compra de productos. Este cálculo permite a la empresa:

- Entender el Valor del Cliente: Identificar cuánto ha gastado cada cliente permite clasificarlos por su valor económico para la empresa, lo que es crucial para estrategias de marketing focalizado y gestión de relaciones con clientes (CRM).
- Segmentación de Clientes: Agrupar clientes basándose en su gasto total puede ayudar a diseñar estrategias de marketing y promociones específicas, maximizando la efectividad de estas acciones.
- Personalización de Ofertas: Basar las ofertas y promociones en el historial de gastos del cliente puede mejorar la experiencia del cliente y aumentar la lealtad y satisfacción del mismo.
- Análisis Financiero: Evaluar el gasto total de los clientes contribuye al análisis financiero global, ayudando a prever ingresos futuros y a planificar estrategias de negocio adecuadas.

Tablas Manipuladas e Implementadas

La función 'CalculateTotalSpent' interactúa con las siguientes tablas:

1. purchase ('pu'): Esta tabla contiene los registros de todas las compras realizadas por los clientes, incluyendo la cantidad de cada producto comprado ('Quantity') y la identificación del cliente ('Client_id'). La tabla es crucial para identificar las compras realizadas por un cliente específico.
2. product ('p'): Proporciona detalles sobre los productos, incluido el precio de cada uno ('Price'). Esta información es necesaria para calcular el costo total de los productos comprados por el cliente.

Funcionamiento de la Función

- La función recibe un `clientId` como parámetro, que se utiliza para filtrar las compras realizadas por ese cliente específico en la tabla `purchase`.
- Realiza una unión (`JOIN`) entre las tablas `purchase` y `product` para obtener el precio de cada producto comprado por el cliente.
- Calcula el total gastado (`SUM(p.Price * pu.Quantity)`) multiplicando el precio de cada producto por la cantidad comprada y sumando estos productos para todas las compras realizadas por el cliente.
- Almacena el resultado del cálculo en la variable `totalSpent` y luego lo retorna.

Esta función devuelve el total de dinero gastado por el cliente especificado como un valor decimal con dos decimales, lo que refleja la precisión monetaria necesaria para análisis financieros. Al implementar esta función, se facilita la realización de consultas complejas relacionadas con el gasto de los clientes de manera sencilla y eficiente, proporcionando una herramienta valiosa para la toma de decisiones basada en datos.

Function N° 2: 'CheckStock'

Como La función `CheckStock` está diseñada para verificar si hay suficiente stock de un producto específico para satisfacer una cantidad solicitada. Este tipo de función es útil en sistemas de gestión de inventarios, ventas en línea, y cualquier aplicación donde sea crucial asegurar la disponibilidad de productos antes de comprometerse a una venta o distribución. A continuación, se explica el propósito de esta función, las tablas que manipula e implementa, y cómo funciona:

Objetivo de la Función

El propósito principal de `CheckStock` es ofrecer una manera rápida y eficiente de determinar si el stock actual de un producto es suficiente para cubrir una demanda específica. Los usos principales incluyen:

- Validación de Pedidos: Asegurar que los pedidos solo se acepten si hay suficiente inventario para cumplirlos.
- Prevención de Sobreventa: Evitar vender más productos de los que están disponibles en stock, lo que puede llevar a problemas de satisfacción del cliente y logística.
- Planificación de Inventario: Identificar necesidades de reabastecimiento de stock basándose en la demanda actual y futura.

Tablas Manipuladas e Implementadas

La función `CheckStock` interactúa principalmente con la siguiente tabla:

1. `product`: Esta tabla contiene los registros de todos los productos disponibles, incluyendo el stock actual de cada uno (`Stock`). La tabla es esencial para verificar la disponibilidad del producto deseado.

Funcionamiento de la Función

- La función recibe dos parámetros: `productId`, que identifica el producto a verificar, y `quantityNeeded`, que especifica la cantidad requerida del producto.
- Realiza una consulta a la tabla `product` para obtener el stock actual del producto especificado por `productId`.
- Utiliza una estructura `CASE` para determinar si el stock disponible (`Stock`) es mayor o igual a la `quantityNeeded`. Si es así, establece el mensaje `Sufficient stock`. De lo contrario, establece el mensaje `Insufficient stock`.
- Almacena el resultado de esta evaluación en la variable `message`.
- Retorna el mensaje, que indica si hay suficiente stock disponible para satisfacer la cantidad necesitada.

Esta función devuelve un mensaje en forma de cadena de texto (`VARCHAR(50)`), lo que hace que sea fácilmente interpretable tanto por aplicaciones automatizadas como por operadores humanos. Al implementar `CheckStock`, las empresas pueden mejorar la eficiencia de sus procesos de ventas y gestión de inventario, asegurando que las decisiones se basen en la disponibilidad real de productos.

Function N° 3: 'CalculateTotalPrice'

La función `CalculateTotalPrice` está diseñada para calcular el precio total de un número especificado de unidades de un producto, basándose en su precio unitario. Esta función es particularmente útil en escenarios como sistemas de punto de venta, comercio electrónico, y gestión de inventarios, donde es necesario determinar el costo total de una cantidad específica de artículos para procesos de facturación, cotización, y análisis financiero. A continuación, se detalla el propósito de la función, las tablas que manipula, y cómo funciona:

Objetivo de la Función

El propósito principal de `CalculateTotalPrice` es facilitar el cálculo del costo total asociado con la compra de una cantidad determinada de un producto. Esto incluye:

- **Facilitación de Transacciones:** Proveer el cálculo inmediato del costo total durante transacciones de compra o venta.
- **Cotización Rápida:** Permitir la generación rápida de cotizaciones para clientes interesados en adquirir múltiples unidades de un producto.
- **Análisis Financiero:** Contribuir a la evaluación de costos en la planificación de compras y análisis de rentabilidad de productos.

Tablas Manipuladas e Implementadas

La función interactúa con la siguiente tabla:

1. **product:** Esta tabla almacena los detalles de cada producto, incluido su precio unitario (``Price``). La función consulta esta tabla para encontrar el precio unitario del producto especificado por ``productId``.

Funcionamiento de la Función

- La función acepta dos parámetros: ``productId``, que es el identificador único del producto cuyo precio total se desea calcular, y ``quantity``, que es la cantidad de dicho producto.
- Realiza una consulta a la tabla ``product`` para obtener el precio unitario del producto especificado por ``productId``.
- Multiplica el precio unitario (``Price``) por la cantidad (``quantity``) para calcular el precio total.
- El resultado del cálculo se almacena en la variable ``totalPrice``.
- Finalmente, la función retorna el valor de ``totalPrice``, representando el costo total de la cantidad especificada del producto.

Al utilizar ``CalculateTotalPrice``, las organizaciones y usuarios pueden obtener de manera eficiente y precisa el costo total de múltiples unidades de un producto, simplificando el proceso de venta y permitiendo una mejor planificación financiera y gestión de inventarios.

Function N° 4: 'GetEmployeeFullName'

La función ``GetEmployeeFullName`` está diseñada para obtener el nombre completo de un empleado a partir de su ID. Esto puede ser útil en diversas situaciones, como en la generación de informes, visualización de datos, o cualquier otro escenario en el que se necesite mostrar el nombre completo de un empleado en lugar de solo su nombre o apellido.

Objetivo de la Función

El propósito principal de ``GetEmployeeFullName`` es proporcionar el nombre completo de un empleado, concatenando su nombre y apellido, a partir de su

ID de empleado. Esto facilita la presentación de información clara y comprensible sobre los empleados en diversas aplicaciones y contextos.

Tablas Manipuladas e Implementadas

La función interactúa con la siguiente tabla:

1. `employee`: Esta tabla almacena la información de los empleados, incluidos sus nombres y apellidos. La función consulta esta tabla para obtener el nombre y apellido del empleado especificado por ``employeeid``.

Funcionamiento de la Función

- La función acepta un parámetro, ``employeeid``, que es el identificador único del empleado del que se desea obtener el nombre completo.
- Realiza una consulta a la tabla ``employee`` para obtener el nombre y apellido del empleado especificado por ``employeeid``.
- Concatena el nombre y el apellido del empleado utilizando la función ``CONCAT`` de MySQL.
- Almacena el nombre completo resultante en la variable ``fullName``.
- Finalmente, la función retorna el valor de ``fullName``, que representa el nombre completo del empleado.

Al utilizar ``GetEmployeeFullName``, las organizaciones pueden mostrar fácilmente el nombre completo de un empleado en lugar de solo su nombre o apellido, lo que puede mejorar la presentación de datos y la experiencia del usuario en diversas aplicaciones y sistemas.

PROCEDIMIENTOS DE ALMACENADO

Stored Procedure N° 1: 'MakeSale'

A El proceso de almacenado 'MakeSale' es un procedimiento almacenado que se utiliza para realizar una venta en el sistema. Aporta varios beneficios al proyecto al encapsular la lógica de negocio relacionada con la venta de productos, lo que simplifica el código en la capa de aplicación y garantiza consistencia en la forma en que se realizan las ventas en el sistema. Además, proporciona una capa de seguridad al verificar el stock disponible antes de realizar la venta y al manejar los errores de manera adecuada.

Objetivos y Beneficios

- Encapsulación de la lógica de negocio: El procedimiento encapsula la lógica de negocio relacionada con la venta de productos, lo que facilita su reutilización y mantenimiento.
- Consistencia en la aplicación: Al utilizar el procedimiento almacenado para realizar ventas, se garantiza que todas las ventas se realicen de acuerdo con la misma lógica, lo que ayuda a mantener la consistencia en el sistema.
- Seguridad: El procedimiento verifica si hay suficiente stock disponible antes de realizar la venta, lo que evita ventas de productos que no están en stock.
- Manejo de errores: El procedimiento maneja de manera adecuada los errores, como la falta de stock, utilizando la instrucción ' SIGNAL SQLSTATE ' para indicar un error específico al cliente.

Tablas Manipuladas e Interactuadas

El procedimiento interactúa principalmente con las siguientes tablas:

- product: Se utiliza para verificar el stock disponible del producto que se va a vender y para actualizar el stock después de la venta.
- purchase: Se utiliza para registrar la venta realizada, incluidos los detalles como el producto vendido, la sucursal, el empleado, el cliente, la cantidad vendida y la fecha de compra.
- payment_method: Se utiliza para especificar el método de pago utilizado en la venta, que se registra junto con la venta en la tabla de compras.

Funcionamiento del Procedimiento

- **Parámetros de Entrada:** El procedimiento acepta varios parámetros de entrada, incluidos el ID del producto, el ID del cliente, el ID de la sucursal, el ID del empleado, la cantidad vendida y el ID del método de pago.
- **Verificación de Stock:** Se verifica si hay suficiente stock disponible del producto utilizando la tabla `product`.
- **Registro de la Venta:** Si hay suficiente stock, se calcula el precio total de la venta y se registra en la tabla `purchase` junto con el método de pago utilizado.
- **Actualización del Stock:** Se actualiza el stock del producto en la tabla `product` después de la venta.
- **Retorno de Información:** Finalmente, se devuelve el precio total de la venta como resultado del procedimiento.

Al utilizar este procedimiento almacenado, las ventas en el sistema se pueden realizar de manera consistente y segura, lo que contribuye a una mejor gestión de inventario y una experiencia de usuario más fluida.

Stored Procedure N° 2: 'AddNewClient'

El procedimiento almacenado `AddNewClient` se utiliza para agregar un nuevo cliente a la base de datos. A continuación, se proporciona una descripción detallada del procedimiento, su objetivo y las tablas que interactúa:

Objetivo

- **Agregar un Nuevo Cliente:** El procedimiento tiene como objetivo agregar un nuevo cliente a la base de datos con la información proporcionada.

Beneficios

- **Simplificación del Código:** Al encapsular la lógica de inserción de un nuevo cliente en un procedimiento almacenado, se simplifica el código en la capa de aplicación.
- **Seguridad:** Al utilizar un procedimiento almacenado, se evitan posibles ataques de inyección SQL al insertar directamente los valores proporcionados como parámetros.
- **Consistencia de Datos:** Al utilizar el procedimiento almacenado, se garantiza que todos los nuevos clientes se agreguen de la misma manera, lo que ayuda a mantener la consistencia de los datos en la base de datos.

Tablas Manipuladas e Interactuadas

El procedimiento interactúa con la siguiente tabla:

1. client: Se utiliza para insertar un nuevo registro de cliente con la información proporcionada.

Funcionamiento del Procedimiento

- Parámetros de Entrada: El procedimiento acepta varios parámetros de entrada que representan la información del nuevo cliente, como el nombre, apellido, fecha de nacimiento, número de teléfono, correo electrónico, género y dirección.
- Inserción en la Tabla: Utilizando la instrucción ``INSERT INTO``, el procedimiento inserta un nuevo registro en la tabla ``client`` con la información proporcionada en los parámetros.
- Finalización del Procedimiento: Una vez que se ha realizado la inserción, el procedimiento finaliza.

Este procedimiento proporciona una forma segura y consistente de agregar nuevos clientes a la base de datos, lo que contribuye a una mejor gestión de la información de los clientes en el sistema.

Stored Procedure N° 3: 'applyPromotion'

El procedimiento almacenado `'applyPromotion'` se utiliza para aplicar un descuento de promoción a un producto específico, actualizando su precio en la base de datos.

Objetivo

- aplicar un descuento de promoción a un producto específico, actualizando su precio en la base de datos. Este procedimiento toma dos parámetros de entrada: el `Product_id` del producto al que se aplicará la promoción y el `Promotion_id` de la promoción que se aplicará.

Funcionamiento del procedimiento:

- Obtiene la fecha actual (`@current_date`).
- Obtiene la fecha de inicio (`@promo_start_date`), fecha de fin (`@promo_end_date`) y descuento (`@promo_discount`) de la promoción con el `Promotion_id` proporcionado.
- Inicia una transacción con `START TRANSACTION`.

- Verifica si la fecha actual está dentro del rango de fechas de la promoción.
- Si es así, calcula el nuevo precio del producto con el descuento aplicado y actualiza el precio en la tabla product para el producto especificado.
- Confirma la transacción con COMMIT y devuelve un mensaje indicando que la transacción se completó con éxito.
- Si la fecha actual no está dentro del rango de fechas de la promoción, revierte la transacción con ROLLBACK y devuelve un mensaje indicando que no hay promoción vigente para el producto.

Beneficios:

- Permite aplicar descuentos de promoción de manera automática y consistente.
- Reduce la posibilidad de errores al realizar actualizaciones manuales en la base de datos.
- Proporciona un mecanismo para mantener un registro de las promociones aplicadas a los productos.

Tablas manipuladas e interactuadas

El procedimiento interactúa con la siguiente tabla:

1. La tabla promotions, de la cual se obtienen los datos de la promoción.
2. La tabla product, donde se actualiza el precio del producto con el descuento de la promoción aplicado.

Este procedimiento simplifica y automatiza el proceso de aplicar descuentos de promoción a productos, garantizando que se realicen de manera consistente y segura en la base de datos.

TRIGGERS

Trigger N° 1: 'MoveToRecycleTable'

El trigger `MoveToRecycleTable` se utiliza para mover los registros de productos eliminados a una tabla de reciclaje llamada `recycle_table`. A continuación, se proporciona una descripción detallada del trigger y su funcionalidad:

Objetivo

- Mover Registros Eliminados: El trigger tiene como objetivo mover los registros de productos que se eliminan de la tabla `product` a la tabla `recycle_table`, para mantener un historial de los productos eliminados.

Beneficios

- Historial de Eliminaciones: Al utilizar el trigger, se mantiene un historial de los productos eliminados, lo que puede ser útil para fines de auditoría o seguimiento.
- Recuperación de Datos: Al mover los registros a una tabla de reciclaje en lugar de eliminarlos permanentemente, se pueden recuperar los datos si es necesario en el futuro.

Tablas Manipuladas e Interactuadas

El trigger interactúa con las siguientes tablas:

1. product: El trigger se dispara antes de eliminar un registro de esta tabla.
2. recycle_table: Se utiliza para insertar los registros de productos eliminados.

Funcionamiento del Trigger

- Disparador de Antes de Eliminar: El trigger se dispara antes de que se elimine un registro de la tabla `product`.
- Obtención de Datos: Utilizando la variable `OLD`, el trigger obtiene los valores del registro que se va a eliminar, como el `Product_id`, `Name_Product`, `Description`, `Price`, `Stock` y `Category_id`.
- Inserción en la Tabla de Reciclaje: Utilizando la instrucción `INSERT INTO`, el trigger inserta los valores obtenidos en la tabla `recycle_table`.
- Finalización del Trigger: Una vez que se ha realizado la inserción en la tabla de reciclaje, el trigger finaliza.

Este trigger proporciona una forma de mantener un historial de los productos eliminados y permite una posible recuperación de datos en el futuro si es necesario.

Trigger N° 2: 'MoveToRecycleTable'

El conjunto de triggers ``PurchaseInsertAuditTrigger``, ``PurchaseUpdateAuditTrigger`` y ``PurchaseDeleteAuditTrigger`` se utilizan para auditar las operaciones de inserción, actualización y eliminación en la tabla ``purchase``, registrando los cambios en la tabla ``purchase_audit``. A continuación, se proporciona una descripción detallada de cada trigger y su funcionalidad:

Objetivo

- Auditoría de Operaciones: Los triggers tienen como objetivo auditar las operaciones de inserción, actualización y eliminación en la tabla ``purchase``, registrando los detalles de cada operación en la tabla ``purchase_audit``.

Beneficios

- Registro de Cambios: Al utilizar los triggers, se registra de manera automática cada operación realizada en la tabla ``purchase``, lo que proporciona un historial detallado de cambios.
- Seguimiento de Operaciones: Los triggers permiten realizar un seguimiento de quién realizó cada operación y en qué fecha y hora.

Tablas Manipuladas e Interactuadas

Los triggers interactúan con las siguientes tablas:

1. `purchase`: La tabla principal sobre la cual se auditan las operaciones.
2. `purchase_audit`: La tabla donde se registran los detalles de las operaciones auditadas.

Funcionamiento de los Triggers

1. PurchaseInsertAuditTrigger (Después de Insertar):

- Disparador: Se dispara después de insertar un nuevo registro en la tabla ``purchase``.
- Acción: Registra los detalles de la nueva inserción en la tabla ``purchase_audit``, incluyendo el ``Purchase_id``, ``Branch_id``, ``Product_id``, ``Employee_id``, ``Client_id``, ``Quantity``, ``Purchase_Date`` y la acción de auditoría como ``INSERT``.

2. PurchaseUpdateAuditTrigger (Después de Actualizar):

- Disparador: Se dispara después de actualizar un registro en la tabla `purchase`.
- Acción: Registra los detalles de la actualización en la tabla `purchase_audit`, incluyendo los mismos campos que el trigger de inserción, pero con la acción de auditoría como `UPDATE`.

3. PurchaseDeleteAuditTrigger (Después de Eliminar):

- Disparador: Se dispara después de eliminar un registro de la tabla `purchase`.
- Acción: Registra los detalles de la eliminación en la tabla `purchase_audit`, incluyendo los mismos campos que el trigger de inserción, pero con la acción de auditoría como `DELETE`.

Consideraciones Adicionales

- Registros de Cambios: Cada trigger registra los cambios utilizando la palabra clave `NEW` para los valores nuevos (en caso de inserción y actualización) y `OLD` para los valores antiguos (en caso de actualización y eliminación).
- Seguridad y Auditoría: Estos triggers proporcionan una capa adicional de seguridad y auditoría para las operaciones realizadas en la tabla `purchase`, lo que permite un seguimiento detallado de los cambios realizados en los registros.

USUARIOS

Usuario 1

- **Nombre:** usuarioCoder1
- Creado con la contraseña "123456coder".

Se le otorgan permisos de 'SELECT' sobre tablas específicas ('client', 'payment_method', 'product', 'category', 'product_review', 'promotions') de la base de datos Innovatech. Esto significa que este usuario puede leer datos de estas tablas, pero no puede modificarlos, insertar nuevos datos, ni eliminar datos existentes. Este perfil de usuario es típicamente utilizado para roles que requieren acceso de solo lectura para reportes o análisis de datos.

Usuario 2

- **Nombre:** usuarioCoder2
- Creado con la contraseña "654321coder".

Se le otorgan permisos de 'SELECT', 'INSERT', 'UPDATE' sobre todas las tablas de todas las bases de datos (*.*). Esto permite al usuario no solo leer datos de cualquier tabla sino también insertar nuevos registros y actualizar registros existentes en todas las bases de datos del servidor MySQL. Es un perfil con mayores privilegios, posiblemente destinado a usuarios con roles de mantenimiento de datos o administración parcial de la base de datos.

Usuario 3

- **Nombre:** usuarioCoder3
- Creado con la contraseña "123321coder".

Se le conceden permisos de 'SELECT', 'DELETE' sobre todas las tablas de todas las bases de datos. Este usuario puede leer todos los datos y eliminar registros de cualquier tabla. Sin embargo, no puede modificar los registros existentes ni insertar nuevos. Este tipo de permisos podría ser adecuado para roles encargados de limpieza o archivado de datos.

HERRAMIENTAS Y TECNOLOGIAS UTILIZADAS

Para realizar el diagrama de entidad-relación, se utilizó la herramienta en línea <https://app.diagrams.net>, que permitió diseñar de forma visual y detallada la estructura y relaciones entre las entidades de la base de datos. Este diagrama sirvió como una guía fundamental para la creación y organización de las tablas en la base de datos, asegurando una correcta normalización y definición de relaciones.

Para la creación de la base de datos, se utilizó MySQL Workbench, una herramienta visual de diseño de bases de datos que permite modelar, diseñar y generar scripts SQL. Con MySQL Workbench, se pudo crear el esquema de la base de datos de manera intuitiva, diseñando las tablas, estableciendo relaciones entre ellas y configurando las propiedades de las columnas, claves primarias y claves foráneas.

Además de MySQL Workbench, también se utilizó SQL para escribir consultas y procedimientos almacenados que interactúan con la base de datos. Con SQL, se pudo definir la estructura de las tablas, insertar datos, realizar consultas para obtener información específica y crear procedimientos almacenados para aplicar descuentos a los productos.

Adicionalmente, se emplearon triggers para automatizar acciones, como la creación de un registro automáticos de las transacciones realizadas en la tabla 'purchase', funciones para realizar cálculos específicos o aplicar lógica personalizada a los datos, y vistas para proporcionar una capa de abstracción sobre los datos, mostrando información de varias tablas en una sola vista.