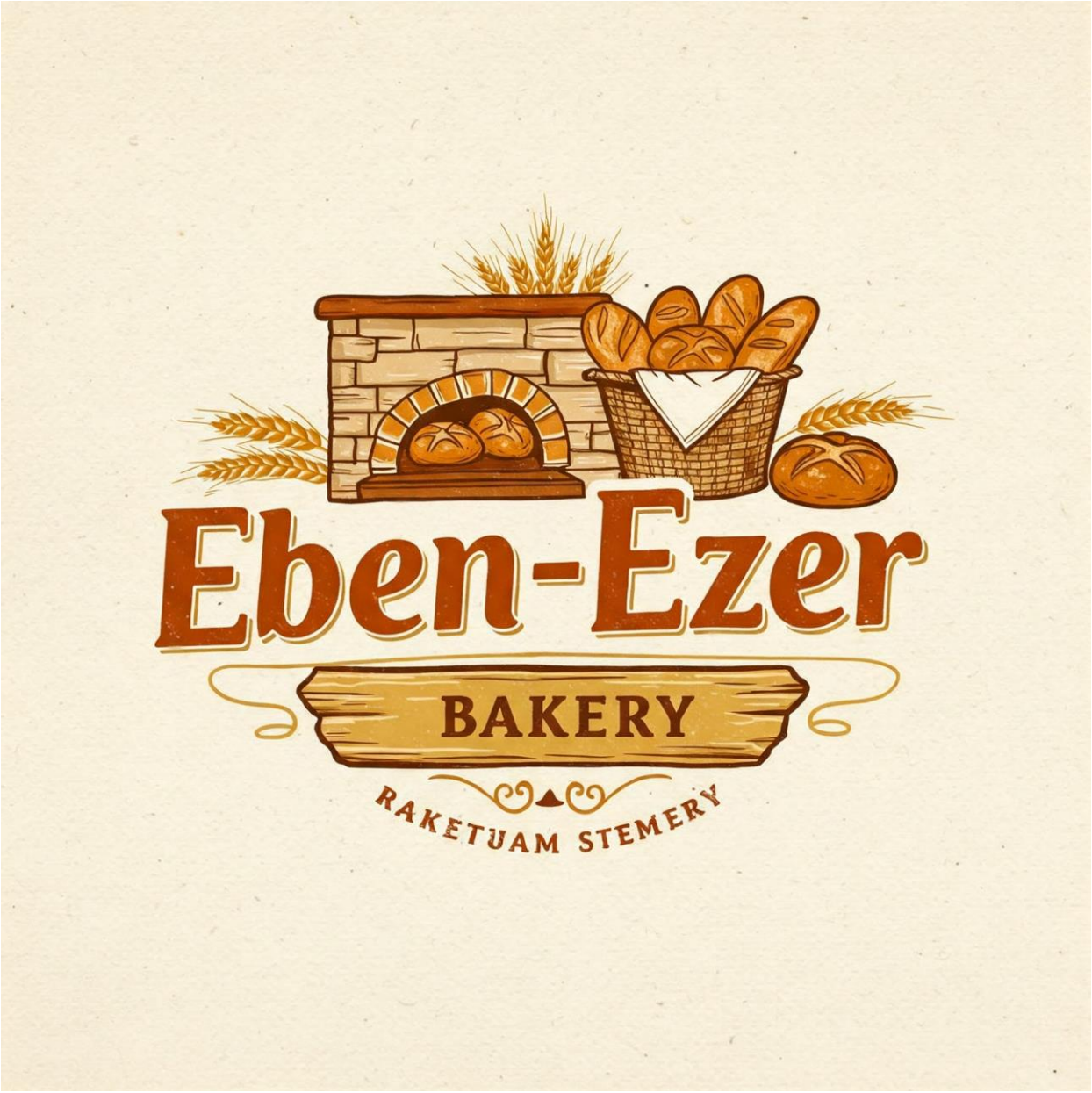
PANADERÍA EBEN-EZER

Proyecto Final SQL CoderHouse – Entrega N°1



Comisión: 72705

Profesor: Anderson Michel Torres

Alumna: Camila Sofia Abati

INTRODUCCIÓN

El proyecto esta inspirado en una panadería, que posee tres locales dentro de la ciudad de Córdoba, cuyo nombre es EBEN-EZER. Dicha empresa comenzó como un negocio familiar en el cual solo vendían un tipo de pan, pero el negocio fue evolucionando, creció notablemente en los últimos dos años y posee en la actualidad 3 locales. Cada uno de ellos posee un horno, para la cocción de los panificados, pero la producción se realiza en la sede inicial y cada mañana se reparte a los otros dos locales.

OBJETIVO

Diseñar e implementar una base de datos relacional para la Panadería EBEN-EZER, que permita una mejor gestión del negocio, aportando la información necesaria para optimizar la administración de los recursos, materiales y humanos, y las ventas.

PROBLEMÁTICA

Con el crecimiento de la empresa, se ampliaron los recursos, tanto materiales como humanos, y la producción, por lo que la misma se encuentra en un proceso de transición ya que debe haber un mayor control y organización, evolucionando ese modelo de empresa familiar que ya no es óptimo por la dimensión que ha tomado el negocio.

- Gestión de recursos humanos: Se necesita una base de datos para saber el rol que ocupa cada persona y a qué local se encuentra asignado.
- Gestión de las ventas y cantidad de productos necesarios por sucursal: Es necesario llevar un control de las ventas para una óptima distribución de las panificaciones.
- Gestión de materia prima y proveedores: Se requiere una base de datos con la información de cada compra a los proveedores.

SOLUCIÓN

La problemática planteada será resuelta con la implementación de la base de datos diseñada, la cual tiene como finalidad unificar y normalizar la información que se maneja en cada local, con el fin de un mejor control y distribución del stock de productos.

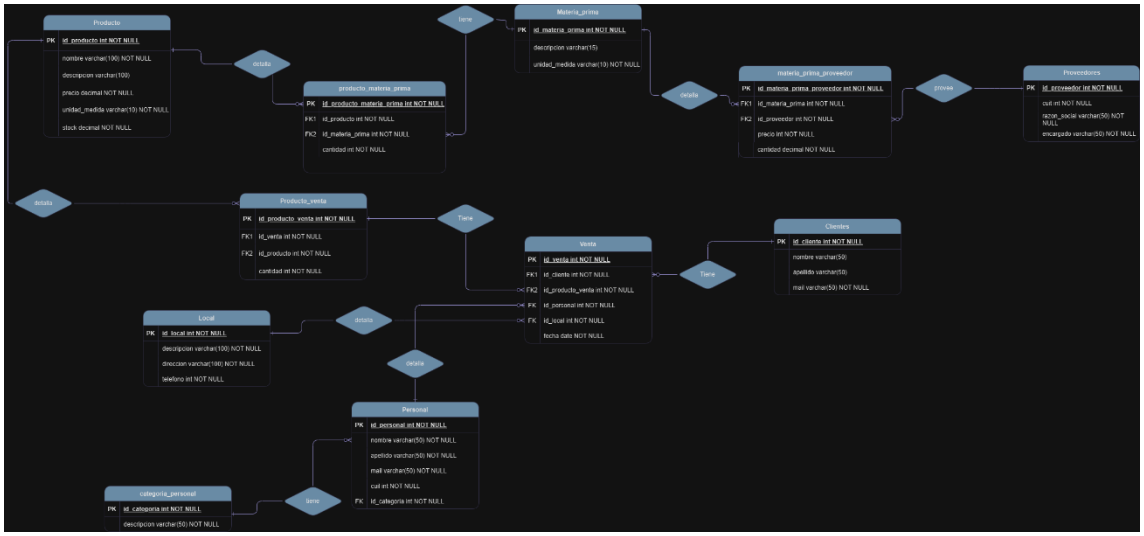
De esta manera, podrán saber qué cantidad de panificados se vende en cada local, pudiendo organizar la distribución de los productos en función de la necesidad. Por otro lado, se tendrá un control del personal, distinguiendo a cada uno de ellos por sus tareas y local al que pertenece.

Por último, se podrá llevar un control de los proveedores y la materia prima, para una mejor gestión de los mismos.

DIAGRAMA ENTIDAD – RELACIÓN

El diagrama de ER fue realizado en draw.io, por lo que además de la imagen se adjunta un link de drive a través del cual pueden realizarse comentarios.

https://drive.google.com/file/d/1c_0LCNKtHruzCYn8HVEUu_K5maYdqhRP/view?usp=drive_link



ESTRUCTURA DE LA BASE DE DATOS

Tablas y su Descripción

1- Tabla *producto*

- **Descripción**: Almacena información sobre los productos.
- **Columnas**:
  - id_producto: Identificador único del producto (clave primaria).
  - nombre: Nombre del producto.
  - descripcion: Descripción del producto.
  - precio: Precio del producto.
  - unidad_medida: Unidad en la que se mide el producto.

COLUMNA	TIPO DE DATO
id_producto(PK)	INT
nombre	varchar(100)
descripcion	varchar(100)
precio	decimal
unidad_medida	varchar(10)

2- Tabla *materia_prima*

- **Descripción**: Almacena información sobre las materias primas utilizadas para los productos de panificación.

- **Columnas**:

  - id_materia_prima: Identificador único de la materia prima (clave primaria).

  - descripcion: Descripción de la materia prima.

  - unidad_medida: Unidad de medida de la materia prima.

COLUMNA	TIPO DE DATO
id_materia_prima(PK)	INT
descripcion	varchar(100)
unidad_medida	varchar(10)

3 – Tabla *proveedor*

- **Descripción**: Almacena información sobre los proveedores.

- **Columnas**:

  - id_proveedor: Identificador único del proveedor (clave primaria).

  - cuit: Número de identificación fiscal del proveedor.

  - razon_social: Razón social del proveedor.

  - encargado: Nombre del encargado del proveedor.

COLUMNA	TIPO DE DATO
id_proveedor(PK)	INT
cuit	INT
razon_social	varchar(100)
encargado	varchar(50)

4 - Tabla *categoria_personal*

- **Descripción**: Almacena categorías del personal.

- **Columnas**:

  - id_categoria: Identificador único de la categoría (clave primaria).

  - descripcion: Descripción de la categoría.

COLUMNA	TIPO DE DATO
id_categoria(PK)	INT
descripcion	varchar(100)

5 – Tabla *personal*

- **Descripción**: Almacena datos del personal.
- **Columnas**:
  - id_personal: Identificador único del personal (clave primaria).
  - nombre: Nombre del personal.
  - apellido: Apellido del personal.
  - mail: Correo electrónico del personal.
  - cuil: Número de identificación del personal.
  - id_categoria: Relación con la tabla categoria_personal.

COLUMNA	TIPO DE DATO
id_personal(PK)	INT
nombre	varchar(50)
apellido	varchar(50)
mail	varchar(50)
cuil	INT
id_categoria(FK)	INT

6 - Tabla *local*

- **Descripción**: Almacena información sobre los locales.
- **Columnas**:
  - id_local: Identificador único del local (clave primaria).
  - descripcion: Descripción del local.
  - direccion: Dirección del local.
  - telefono: Número de teléfono del local.

COLUMNA	TIPO DE DATO
id_local(PK)	INT
descripcion	varchar(100)
direccion	varchar(100)
telefono	varchar(15)

7 – Tabla ***cliente***

- **Descripción**: Almacena información sobre los clientes.
- **Columnas**:
  - id_cliente: Identificador único del cliente (clave primaria).
  - nombre: Nombre del cliente.

- apellido: Apellido del cliente.

- mail: Correo electrónico del cliente.

COLUMNA	TIPO DE DATO
id_cliente(PK)	INT
nombre	varchar(50)
apellido	varchar(50)
mail	varchar(50)

8 – Tabla ***materia_prima_proveedor***

- **Descripción:** Relación entre materias primas y proveedores.

- **Columnas:**

- id_materia_prima_proveedor: Identificador único de la relación (clave primaria).

- id_proveedor: Relación con la tabla proveedor.

- id_materia_prima: Relación con la tabla materia_prima.

- cantidad: Cantidad de materia prima suministrada por el proveedor.

COLUMNA	TIPO DE DATO
id_materia_prima_proveedor(PK)	INT
id_proveedor(FK)	INT
id_materia_prima(FK)	INT
cantidad	decimal
precio	decimal

9 – Tabla producto_materia_prima

- **Descripción**: Relación entre productos y materias primas.

- **Columnas**:

- id_producto_materia_prima: Identificador único de la relación (clave primaria).

- id_producto: Relación con la tabla producto.

- id_materia_prima: Relación con la tabla materia_prima.

- cantidad: Cantidad de materia prima utilizada en el producto.

COLUMNA	TIPO DE DATO
id_producto_materia_prima(PK)	INT
id_producto(FK)	INT
id_materia_prima(FK)	INT
cantidad	decimal

10 – Tabla producto_venta

- Descripción: Almacena información sobre la relación entre ventas y productos dentro de la venta.

- Columnas:

  - id_venta_producto: Identificador único de la venta (clave primaria).

  - id_producto: Relación con la tabla producto

  - cantidad: Cantidad de producto vendido en la unidad de medida correspondiente

COLUMNA	TIPO DE DATO
id _venta_ producto (PK)	INT
id_producto(FK)	INT
cantidad	decimal

11 – Tabla Venta

- Descripción: Almacena información sobre las ventas realizadas.

- Columnas:

  - id_venta: Identificador único de la venta (clave primaria).

  - id_venta_producto: Relación con la tabla producto.

  - id_cliente: Relación con la tabla cliente

  - id_local: Relación con la tabla local

  - id_personal: Relación con la tabla personal

COLUMNA	TIPO DE DATO
id_venta(PK)	INT
id_venta_producto(FK)	INT
id_cliente(FK)	INT
id_local(FK)	INT
id_personal(FK)	INT

OBJETOS EN LA BASE DE DATOS

Funciones

1. proveedor_mas_economico

Descripción: Devuelve el ID del proveedor que ofrece el precio más bajo para una materia prima específica.

Definición:

CREATE FUNCTION proveedor_mas_economico(id_materia_prima INT)

RETURNS INT

Parámetros:

- id_materia_prima (INT): ID de la materia prima para la cual se busca el proveedor más económico.

Detalles:

- Realiza una consulta para encontrar el proveedor con el menor precio asociado a la materia prima especificada.
- Ordena los proveedores por precio de forma ascendente y retorna el primer resultado.

---

2. total_ventas_hoy

Descripción: Calcula el total de ventas realizadas en la fecha actual.

Definición:

CREATE FUNCTION total_ventas_hoy()

RETURNS DECIMAL(10,2)

Parámetros:

- Ninguno.

Detalles:

- Realiza una suma del campo total_venta de la tabla venta para todas las ventas realizadas en la fecha actual.

Procedimientos Almacenados

1. sp_materias_primas_mas_economicas

Descripción: Genera una tabla temporal que lista las materias primas con su precio mínimo asociado.

Definición:

CREATE PROCEDURE sp_materias_primas_mas_economicas()

Parámetros:

- Ninguno.

Detalles:

- Crea una tabla temporal para almacenar los resultados.
- Agrupa las materias primas y calcula el precio mínimo de cada una.
- Retorna los datos de la tabla temporal y luego la elimina.

---

2. registrar_nueva_venta

Descripción: Registra una nueva venta, incluyendo información del producto, cliente, personal y local.

Definición:

CREATE PROCEDURE registrar_nueva_venta(

IN p_id_producto INT,

IN p_cantidad INT,

IN p_id_cliente INT,

IN p_id_personal INT,

IN p_id_local INT,

IN p_fecha DATE)

Parámetros:

- p_id_producto (INT): ID del producto vendido.
- p_cantidad (INT): Cantidad del producto vendido.
- p_id_cliente (INT): ID del cliente que realiza la compra.
- p_id_personal (INT): ID del personal que gestiona la venta.
- p_id_local (INT): ID del local donde se realiza la venta.
- p_fecha (DATE): Fecha de la venta.

Detalles:

- Inserta un nuevo registro en la tabla producto_venta y obtiene su ID generado.
- Usa el ID generado para insertar un registro en la tabla venta.

Triggers

1. validar_proveedor

Descripción: Verifica que no se inserten duplicados de proveedores con la misma razón social.

Definición:

CREATE TRIGGER validar_proveedor

BEFORE INSERT ON proveedor

Detalles:

- Se ejecuta antes de insertar un nuevo proveedor.
- Lanza un error si ya existe un proveedor con la misma razón social.

---

2. validar_cliente_duplicado

Descripción: Verifica que no se inserten duplicados de clientes con el mismo nombre y apellido.

Definición:

CREATE TRIGGER validar_cliente_duplicado

BEFORE INSERT ON cliente

Detalles:

- Se ejecuta antes de insertar un nuevo cliente.
- Lanza un error si ya existe un cliente con el mismo nombre y apellido.

Vistas

Vista: ventas_hoy

Propósito:

- Mostrar todas las ventas realizadas en el día actual.

Campos:

- Todos los campos de la tabla venta.

Cálculo:

- Filtra las ventas donde la fecha de la venta coincide con la fecha actual del sistema.

Uso:

- Permite obtener un resumen rápido de las ventas del día.
- Es útil para monitorear las ventas diarias y tomar decisiones operativas.

Vista: ventas_por_local

Propósito:

- Mostrar el número total de ventas realizadas en cada local.

Campos:

- id_local: Identificador único del local.
- count(id_venta): Cantidad total de ventas realizadas en ese local.

Cálculo:

- Agrupa las ventas por local y cuenta la cantidad de registros en cada grupo.

Uso:

- Permite comparar el desempeño de diferentes locales.
- Es útil para identificar los locales más rentables y aquellos que requieren mayor atención.

Vista: vw_ventas_por_producto

Propósito:

- Mostrar la cantidad total vendida de cada producto.

Campos:

- id_producto: Identificador único del producto.
- nombre: Nombre del producto.
- cantidad_vendida: Suma total de unidades vendidas del producto.

Cálculo:

- Se realiza un JOIN entre las tablas producto y producto_venta para relacionar los productos con sus respectivas ventas.
- Se agrupa por id_producto y se calcula la suma de las cantidades vendidas utilizando la función SUM.

Uso:

- Permite analizar el desempeño de cada producto en términos de ventas.
- Es útil para identificar los productos más vendidos, los productos menos vendidos, y para tomar decisiones sobre la gestión del inventario y la promoción de productos.

Vista: vw_ventas_por_vendedor

Propósito:

- Mostrar la cantidad total de ventas realizadas por cada vendedor.

Campos:

- id_personal: Identificador único del vendedor.
- nombre: Nombre del vendedor.
- apellido: Apellido del vendedor.
- cantidad_ventas: Número total de ventas realizadas por el vendedor.

Cálculo:

- Se realiza un JOIN entre las tablas personal y venta para relacionar los vendedores con sus respectivas ventas.
- Se agrupa por id_personal y se cuenta el número de ventas utilizando la función COUNT.

Uso:

- Permite evaluar el desempeño de cada vendedor y para identificar a los vendedores más productivos.
- Es útil para establecer metas de ventas, calcular comisiones y realizar evaluaciones de desempeño.

HERRAMIENTAS UTILIZADAS

- Para la creación de los datos se utilizó Gemini, IA de Google.