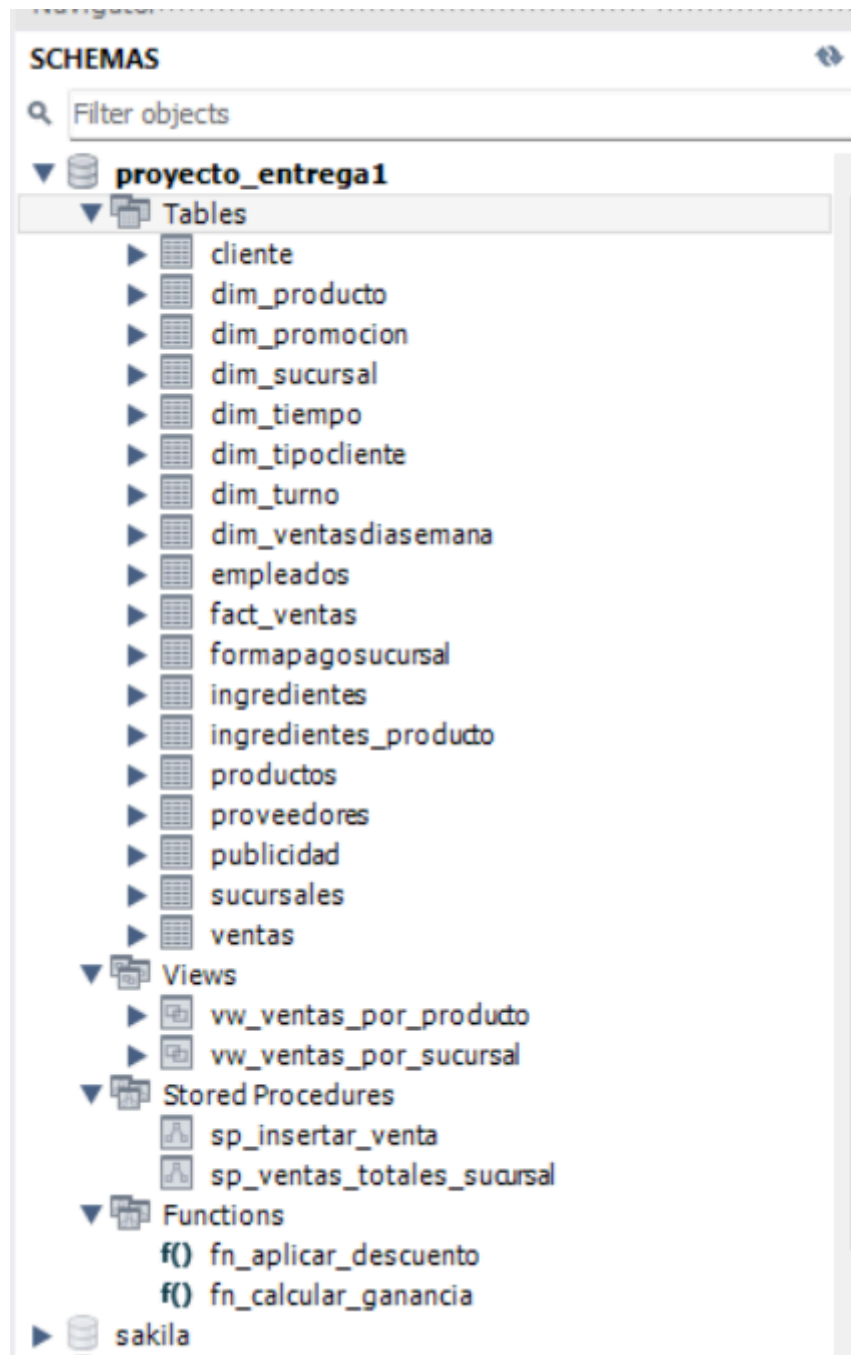


# Proyecto: BURGERS LDG



Primera entrega - Curso CoderHouse SQL  
Facundo Galbarini

## Introducción

Este proyecto trata sobre un modelo de base de datos relacional para una cadena de hamburgueserías que busca mejorar la gestión y eficiencia de su comercio. El diseño de este contiene información de ventas, clientes, productos, promociones, turnos, insumos y proveedores, así como dimensiones necesarias para enriquecer el análisis del negocio en su totalidad.

El proyecto propone una herramienta que permita el análisis y disponibilidad de información clave para la toma de decisiones, generando una estructura escalable y alineada a las necesidades reales de un negocio gastronómico.

## Objetivo del Proyecto

El objetivo principal es diseñar e implementar un esquema de base de datos que permita registrar y analizar todas las transacciones de venta realizadas por el negocio de forma integral, clara y sencilla a la vista.

A su vez, esto unifica información operativa proveniente de distintas áreas como contabilidad (ventas y pagos), logística (insumos y proveedores), análisis comercial (promociones, productos, demanda por día u horario) y gestión de recursos humanos (empleados y turnos). Para último, al generar una base sólida, permite construir reportes analíticos, indicadores de gestión (KPI's) y futuros tableros de BI.

## Situación Problemática

La cadena de hamburgueserías enfrenta varios desafíos derivados de la falta de una base de datos estructurada y centralizada:

1. **Información dispersa:** Los datos provienen de sistemas contables, registros manuales, aplicaciones de venta y proveedores externos.
2. **Dificultad para analizar ventas** por sucursal, día, turno o tipo de cliente, lo que limita la capacidad de identificar patrones de consumo o ineficiencias.
3. **Promociones sin medición clara:** No se puede evaluar el impacto real de cada campaña promocional en las ventas.
4. **Gestión logística no eficiente,** por falta de registro detallado de proveedores e insumos utilizados.

Implementar una base de datos relacional permite resolver estas brechas mediante: la integración total de la información del negocio, una visión 360° del cliente, venta y

operación, la capacidad de medir rentabilidad por producto o sucursal y por último el soporte para decisiones estratégicas basadas en datos

## **Modelo de Negocio**

El modelo de negocio representa una cadena de hamburgueserías que opera en distintas sucursales, cada una con su propia capacidad operativa y métodos de pago. La empresa vende productos como hamburguesas, acompañamientos y bebidas, cada uno compuesto por insumos provistos por distintos proveedores.

El flujo general del negocio podría resumirse de la siguiente manera:

1. Clientes realizan compras en distintas sucursales, tanto de manera presencial como por delivery o take-away.
2. Cada compra (Venta) está compuesta por uno o varios productos, registrados en el detalle de Venta.
3. Los productos se elaboran a partir de insumos, que se vinculan a proveedores específicos.
4. Las sucursales tienen empleados asignados y operan en turnos con distintos niveles de participación en ventas.
5. La empresa puede aplicar promociones que afectan el comportamiento de compra.
6. Las ventas varían según el día de la semana, permitiendo identificar patrones de demanda.

**Diagrama E-R (Entidad–Relación):** archivo SQL

## **Listado de Tablas (Detalle por Entidad)**

### **Sucursales**

- Sucursal\_ID — INT — PK
- Nombre — VARCHAR
- Ciudad — VARCHAR
- Barrio — VARCHAR
- Capacidad — INT

### **Proveedores**

- Proveedor\_ID — INT — PK
- Nombre — VARCHAR

- Telefono — VARCHAR
- Ciudad — VARCHAR

### **Productos**

- Producto\_ID — INT — PK
- Nombre — VARCHAR
- Precio — DECIMAL
- Categoria — VARCHAR

### **Ingredientes\_Producto**

- Producto\_ID — INT — FK → Productos
- Insumo\_ID — INT — FK → Insumos (*PK: Producto\_ID + Insumo\_ID*)
- Cantidad — DECIMAL

### **Dim\_Insumo**

- Insumo\_ID — INT — FK → Productos
- Nombre\_Insumo — VARCHAR
- Tipo\_Insumo — VARCHAR
- Unidad\_Medida — VARCHAR
- Proveedor\_Id — INT
- Costo\_Uitario — DECIMAL

### **Empleados**

- Empleado\_ID — INT — PK
- Nombre — VARCHAR
- Puesto — VARCHAR
- Sucursal\_ID — INT — FK → Sucursal

### **Dim\_Turno**

- Turno\_ID — INT — PK
- Nombre\_Turno — VARCHAR
- Horario — VARCHAR
- Porcentaje\_Ventas — DECIMAL

### **Forma\_Pago**

- FormaPago\_ID — INT — PK
- Metodo — VARCHAR
- Sucursal\_ID — INT — FK → Sucursal

### **Dim\_TipoCliente**

- TipoCliente\_ID — INT — PK
- Descripcion — VARCHAR
- Sucursal\_ID — INT — FK → Sucursal
- Porcentaje\_Ventas — DECIMAL

### **Dim\_Promocion**

- Promocion\_ID — INT — PK
- Nombre\_Promo — VARCHAR
- Descuento\_Porcentaje — INT
- Ventas\_Con\_Promo — INT

### **Clientes**

- Cliente\_ID — INT — PK
- Nombre — VARCHAR
- TipoCliente\_ID — INT — FK → Dim\_TipoCliente

### **Ventas**

- Venta\_ID — INT — PK
- Cliente\_ID — INT — FK → Clientes
- Fecha — DATE
- Sucursal\_ID — INT — FK → Sucursal
- Total — DECIMAL

### **Detalle\_Venta**

- Detalle\_ID — INT — PK
- Venta\_ID — INT — FK → Ventas
- Producto\_ID — INT — FK → Productos
- Cantidad — INT
- Precio\_Unitario — DECIMAL

### **Dim\_VentasDiaSemana**

- DiaSemana\_ID — INT — PK
- Sucursal\_ID — INT — FK → Sucursal
- Dia\_Semana — VARCHAR
- Ventas\_Promedio — DECIMAL
- Porcentaje\_Sobre\_Semana — DECIMAL

### **Medios\_Publicitarios**

- Medio\_ID — INT — PK

- Medio — VARCHAR
- Inversion\_Mensual — DECIMAL

## Vistas creadas

### Vista: vw\_ventas\_detalle

Vista que consolida la información principal de ventas, integrando la tabla de hechos **Ventas** con sus dimensiones asociadas.

**Objetivo de uso:**

Facilitar el análisis analítico y la generación de reportes sin necesidad de realizar múltiples JOINS complejos sobre la tabla de hechos.

Análisis de ventas por producto, sucursal, turno y tipo de cliente.

**Tablas que componen la vista:**

- Ventas
- Productos
- Sucursales
- Cliente
- Dim\_Turno
- Dim\_TipoCliente

### Vista: vw\_ventas\_por\_sucursal

Vista agregada que resume el volumen de ventas y el monto total por sucursal. **Objetivo de uso:**

Permitir un análisis rápido del desempeño comercial de cada sucursal.

Comparación de performance entre sucursales y toma de decisiones operativas.

**Tablas que componen la vista:**

- Ventas
- Sucursales

## Funciones personalizadas

### Función: fn\_calcular\_ganancia

Función escalar que calcula la ganancia de una venta a partir del monto total y el costo total.

**Objetivo de uso:**

Estandarizar el cálculo de ganancia y evitar duplicación de lógica en consultas, vistas o procedimientos.

Función aplicada sobre Valores numéricos de la tabla Ventas.

**Función: fn\_aplicar\_descuento**

Función que calcula el monto final de una venta luego de aplicar un porcentaje de descuento.

**Objetivo de uso:**

Simular escenarios de promociones y descuentos de manera consistente dentro del modelo.

Función aplicada sobre elMonto total y porcentaje de descuento asociado a promociones.

## Stored Procedures

**Stored Procedure: sp\_insertar\_venta**

**Procedimiento almacenado que inserta un nuevo registro en la tabla Ventas, calculando automáticamente métricas como ganancia y descuento aplicado.**

**Objetivo de uso:**

Centralizar la lógica de carga de ventas, asegurando integridad referencial y consistencia en los cálculos.

**Tablas impactadas:**

- Ventas

**Ventajas:**

- Reduce errores de inserción manual
- Garantiza el uso correcto de claves foráneas
- Facilita la carga masiva de datos

## Triggers

**Trigger: tr\_actualizar\_ganancia**

Trigger que se ejecuta automáticamente antes de insertar una venta, calculando la ganancia en función del monto total y el costo.

**Objetivo:**

Evitar inconsistencias en el cálculo de métricas financieras.

**Tabla sobre la que actúa:**

- Ventas (BEFORE INSERT)

Trigger: tr\_actualizar\_promocion\_ventas

**Descripción:**

Trigger que incrementa el contador de ventas asociadas a una promoción cada vez que se registra una venta con promoción aplicada.

Mantener actualizada la métrica de impacto de promociones sin intervención manual.

**Tablas involucradas:**

- Ventas
- Dim\_Promocion