

SISTEMA DE GESTIÓN DE RESTAURANTE

AUDITORÍA DE BASE DE DATOS

JIMMY LUCERO - DAVID CAJAMARCA

QUITO, 4 DE AGOSTO DEL 2025

ING. LORENA CHULDE

BASES DE DATOS 2025-A

Introducción

- Motor de base de datos: MySQL
- Objetivo: Diseñar un sistema relacional completo que gestione de forma eficiente todas las operaciones de un restaurante.
- Cumple criterios avanzados de:
 - Modelado relacional
 - Integridad y normalización
 - Seguridad, auditoría y rendimiento
 - Automatización con procedimientos y triggers



Modelo Relacional – Esquema General

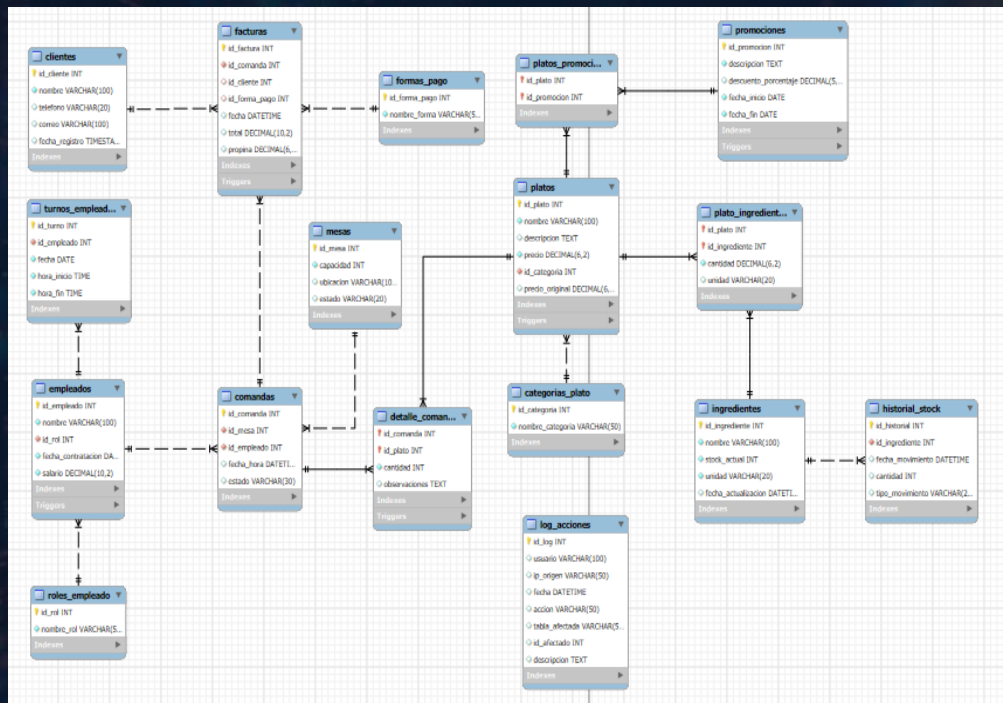
- Modelo validado en 3FN: sin redundancias, con integridad y optimización estructural.
- Herramienta: MySQL Workbench
- Esquema: restaurante_bd con 17 tablas:
- Funcionales: clientes, empleados, comandas, platos, ingredientes, facturas...
- Catálogo: roles_empleado, formas_pago, categorias_plato, platos, platos_ingredientes...
- Auditoría: log_acciones

Relaciones Clave:

1:N → empleados → comandas | clientes → facturas

N:M → platos ↔ ingredientes | platos ↔ promociones (vía tablas intermedias)

Atributos multivaluados correctamente normalizados



Procedimientos Almacenados

Se implementaron **7 procedimientos**, cumpliendo buenas prácticas:

- Validaciones lógicas y SIGNAL
- Transacciones con manejo de errores
- Generación de reportes

Ejemplos:

sp_agregar_plato_a_comanda: Inserta y actualiza platos según condiciones.

sp_aplicar_descuento_por_categoria: Aplica descuentos masivos con reversión segura.

sp_registrar_entrada_stock: Usa START TRANSACTION, SAVEPOINT, ROLLBACK, COMMIT.



Procedimientos Almacenados

```
-- Procedimiento para reportes por periodo
delimiter //

create procedure sp_reporte_ventas_periodo(
    in p_fecha_inicio date,
    in p_fecha_fin date)
begin
    select
        p.nombre as plato,
        c.nombre_categoria as categoria,
        sum(dc.cantidad) as cantidad_vendida,
        sum(dc.cantidad * p.precio) as total_facturado
    from detalle_comanda dc
    inner join platos p on dc.id_plato = p.id_plato
    inner join categorias_plato c on p.id_categoria = c.id_categoria
    inner join comandas cmd on dc.id_comanda = cmd.id_comanda
    inner join facturas f on cmd.id_comanda = f.id_comanda
    where date(f.fecha) between p_fecha_inicio and p_fecha_fin
    group by p.nombre, c.nombre_categoria
    order by total_facturado desc;

end //

delimiter ;
```

```
462 • call sp_reporte_ventas_periodo('2025-01-01', '2025-12-31');
```

463

Result Grid | Filter Rows: | Export: | Wrap Cell Content: IA

	plato	categoria	cantidad_vendida	total_facturado
▶	combo ejecutivo	especial	3	30.00
	helado de vainilla	postre	3	13.50
	lomo saltado	plato fuerte	1	8.01

Procedimientos Almacenados

```
-- Procedimiento para agregar a detalle_comanda
delimiter //
create procedure sp_agregar_plato_a_comanda(
    in p_id_comanda int,
    in p_id_plato int,
    in p_cantidad int,
    in p_observaciones text
)
begin
    declare v_estado_comanda varchar(30);
    select estado into v_estado_comanda from comandas where id_comanda = p_id_comanda;
    if v_estado_comanda is null then
        signal sqlstate '45000' set message_text = 'error: la comanda especificada no existe.';
    elseif v_estado_comanda != 'solicitado' then
        signal sqlstate '45000' set message_text = 'error: no se pueden añadir platos. la comanda ya está en cocina.';
    elseif not exists (select 1 from platos where id_plato = p_id_plato) then
        signal sqlstate '45000' set message_text = 'error: el plato especificado no existe.';
    else
        insert into detalle_comanda (id_comanda, id_plato, cantidad, observaciones)
        values (p_id_comanda, p_id_plato, p_cantidad, p_observaciones)
        on duplicate key update -- si el plato ya está, suma la cantidad
        cantidad = cantidad + p_cantidad,
        observaciones = concat(observaciones, ' | adición: ', p_observaciones);
        select 'plato agregado a la comanda correctamente.' as resultado;
    end if;
end //
delimiter ;
```

```
CALL sp_agregar_plato_a_comanda(1, 5, 2, 'Extra queso, sin cebolla.');
```

resultado

plato agregado a la comanda correctamente.

```
331
332 • Select * from detalle_comanda;
```

Result Grid

	id_comanda	id_plato	cantidad	observaciones
▶	1	5	2	Extra queso, sin cebolla.
	2	2	1	sin picante
	3	3	3	extra dulce
	4	5	1	NULL
	6	1	2	NULL
*	NULL	NULL	NULL	NULL

Funciones Definidas por el Usuario

Funciones escalares que encapsulan cálculos comunes:

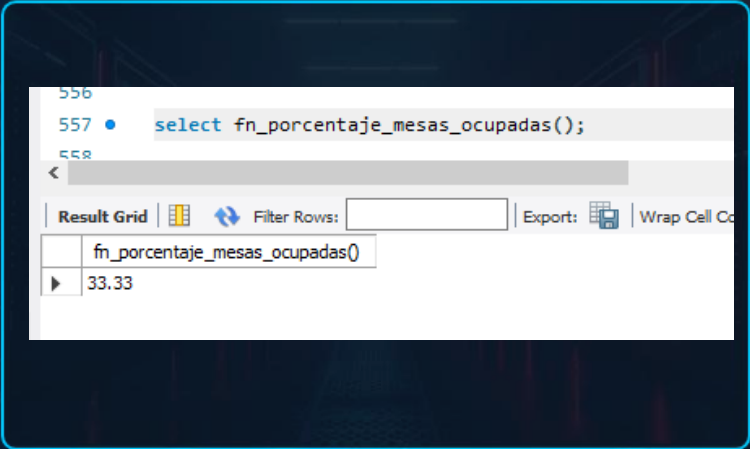
- `fn_porcentaje_mesas_ocupadas()`: Ocupación del restaurante.
- `calcularantiguedadempleado()`: Años de servicio desde la contratación.
- `calculartotalcomanda()`: Total acumulado por comanda.
- `verificardisponibilidadplato()`: Determina disponibilidad según stock de ingredientes.



Funciones Definidas por el Usuario

```
-- Funcion para ver porcentaje de mesas ocupadas
delimiter //
create function fn_porcentaje_mesas_ocupadas()
returns decimal(5,2)
deterministic
reads sql data
begin
    declare total int;
    declare ocupadas int;
    select count(*) into total from mesas;
    select count(*) into ocupadas from mesas where estado = 'ocupada';

    if total = 0 then
        return 0;
    else
        return round((ocupadas * 100.0) / total, 2);
    end if;
end;
//
delimiter ;
```



The screenshot shows a SQL client window with a query editor and a results pane. The query editor contains the SQL statement `select fn_porcentaje_mesas_ocupadas();`. The results pane displays a single row with the value `33.33` under the column header `fn_porcentaje_mesas_ocupadas()`. The interface includes standard SQL client controls like a 'Result Grid' button, a 'Filter Rows' input, and an 'Export' button.

fn_porcentaje_mesas_ocupadas()
33.33

Funciones Definidas por el Usuario

```
-- Funcion para verificar disponibilidad de platos
delimiter $$
create function verificardisponibilidadplato(
    p_id_plato int)
returns varchar(20)
reads sql data
begin
    declare ingredientes_faltantes int;
    select count(*)
    into ingredientes_faltantes
    from plato_ingredientes pi
    join ingredientes i on pi.id_ingrediente = i.id_ingrediente
    where pi.id_plato = p_id_plato
        and i.stock_actual < pi.cantidad;

    if ingredientes_faltantes = 0 then
        return 'disponible';
    else
        return 'agotado';
    end if;
end$$
```

```
637
638 • select nombre, precio, verificardisponibilidadplato(id_plato) as estado
639 from platos;
```

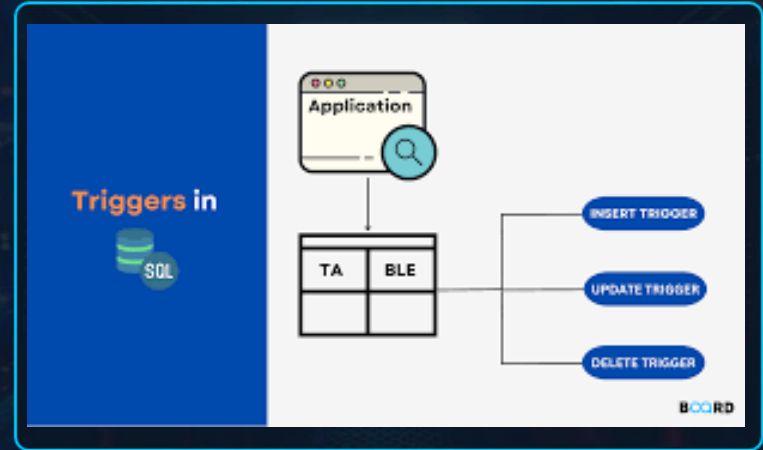
nombre	precio	estado
ensalada mixta	5.00	disponible
lomo saltado	8.01	disponible
helado de vainilla	4.50	disponible
jugo natural	2.00	disponible
combo ejecutivo	10.00	disponible
Bruschetta de Tomate y Albahaca	6.50	disponible
Calamares a la Romana	12.80	disponible
Guacamole con Totopos	9.00	disponible
Provoleta a la Parrilla	10.50	disponible
Ceviche de Pescado Clasico	14.20	disponible
Empanadas de Carne	7.80	disponible
Carpaccio de Res	15.50	disponible

Triggers (Disparadores)

Automatizan acciones sensibles y mantienen integridad:

- Auditoría: `trg_log_cambios_empleados`, `trg_insert_factura_log`
- Seguridad y control: `trg_log_actualizacion_empleados`, `trg_notificar_cambioPrecio`
- Control de stock: `trg_baja_stock_comanda` (uso de cursor y lógica avanzada)

Todos los triggers fueron validados en escenarios realistas.



Triggers (Disparadores)

```
-- 1.- Trigger para contratacion de empleados
delimiter $$
create trigger trg_log_cambios_empleados
after insert on empleados
for each row
begin
    insert into log_acciones (usuario, accion, tabla_afectada, id_afectado, descripcion)
    values (user(), 'insert', 'empleados', new.id_empleado,
           concat('se ha contratado al nuevo empleado: ', new.nombre,
                 ' con el rol id: ', new.id_rol,
                 ' y un salario de: ', new.salario));
end$$
delimiter ;
```

```
-- Contratar un nuevo empleado (activará el trigger de INSERT)
insert into empleados (nombre, id_rol, fecha_contratacion, salario)
values ('diego castro', 1, '2025-08-01', 900.00);
```

```
select * from log_acciones;
```

ip_origen	fecha	accion	tabla_afectada	id_afectado	descripcion
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	396	Cambio de precio para el plato "Goulash Hungar.
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	397	Cambio de precio para el plato "Bife de Chorizo .
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	399	Cambio de precio para el plato "Spaghetti alla C.
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	400	Cambio de precio para el plato "Paella Valencian.
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	403	Cambio de precio para el plato "Pato a la Naranj.
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	404	Cambio de precio para el plato "Salmon a la Plan.
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	405	Cambio de precio para el plato "Mole Poblano co.
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	407	Cambio de precio para el plato "Lasagna Bologn.
127.0.0.1	2025-08-03 19:22:12	eliminación	promociones	1	promoción "descuento almuerzos" eliminada.
NULL	2025-08-03 19:37:08	insert	empleados	6	se ha contratado al nuevo empleado: diego cas.
NULL	NULL	NULL	NULL	NULL	NULL

acciones 23 x

Triggers (Disparadores)

```
-- Trigger para notificar cambio de precio
delimiter $$

create trigger trg_notificar_cambioPrecio
after update on platos
for each row
begin
    if old.precio <> new.precio then
        insert into log_acciones (usuario, ip_origen, accion, tabla_afectada, id_afectado, descripcion)
        values (
            user(),
            substring_index(user(), '@', -1),
            'notificacionPrecio',
            'platos',
            new.id_plato,
            concat('cambio de precio para el plato "', new.nombre,
                '". precio anterior: ', old.precio,
                ', precio nuevo: ', new.precio)
        );
    end if;
end$$

delimiter ;
```

```
-- Actualizamos el precio de un plato específico (esto activará el trigger)
update platos
set precio = 13.90
where id_plato = 7;
```

Select * from log_acciones;

Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
ip_origen	fecha	accion	tabla_afectada	id_afectado	descripcion
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	404	Cambio de precio para el plato "Salmon a la Plan...
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	405	Cambio de precio para el plato "Mole Poblano co...
localhost	2025-08-03 19:15:30	NOTIFICACION_PRECIO	platos	407	Cambio de precio para el plato "Lasagna Bologn...
127.0.0.1	2025-08-03 19:22:12	eliminación	promociones	1	promoción "descuento almuerzos" eliminada.
NULL	2025-08-03 19:37:08	insert	empleados	6	se ha contratado al nuevo empleado: diego cas...
NULL	2025-08-03 19:37:57	update	empleados	1	se actualizo el empleado juan perez. rol anterior ...
NULL	2025-08-03 19:38:48	delete	empleados	6	se ha eliminado al empleado: diego castro (id: 6)
127.0.0.1	2025-08-03 19:42:54	inserción	facturas	8	se registró la factura #8 para la comanda #6
127.0.0.1	2025-08-03 19:53:38	eliminación	promociones	6	promoción "combo ejecutivo promo" eliminada.
localhost	2025-08-03 19:55:39	NOTIFICACION_PRECIO	platos	7	Cambio de precio para el plato "Calamares a la ...
NULL	NULL	NULL	NULL	NULL	NULL

Index

Index Simples

En campos críticos para búsquedas y filtros:

- idx_clientes_correo, idx_empleados_id_rol,
idx_facturas_fecha, idx_platos_nombre

Index Compuestos

Para combinaciones comunes:

- idx_detalle_comanda_compuesto (id_comanda, id_plato)
- idx_facturas_cliente_fecha (id_cliente, fecha)
- idx_platos_cat_precio (id_categoria, precio)

Objetivo: Mejorar el rendimiento en JOIN, WHERE, y ORDER BY.



Index

Index Simples

Tabla	Columna	Finalidad / Uso principal
clientes	correo	Búsquedas y validación rápida por correo
empleados	id_rol	Filtros por rol de empleado
empleados	nombre	Búsqueda rápida por nombre
comandas	id_mesa	Relación mesa-comanda en JOIN y WHERE
comandas	id_empleado	Filtrado por mesero
comandas	estado	Filtrado por estado de comanda (abierta, cerrada, etc.)
facturas	id_cliente	Filtrado por cliente en reportes
facturas	fecha	Consultas por rango de fechas

Index Compuestos

Tabla	Columnas	Finalidad / Uso principal
historial_stock	tipo_movimiento, fecha_movimiento	Consultas históricas por tipo y fecha
empleados	id_rol, salario	Filtros combinados por rol y rango salarial
comandas	id_empleado, estado	Filtrado conjunto por mesero y estado de comanda
facturas	id_cliente, fecha	Filtros para reportes por cliente en un periodo
turnos_empleados	id_empleado, fecha	Control de asistencia por día y empleado
detalle_comanda	id_comanda, id_plato	Optimización de JOIN y clave primaria compuesta
promociones	fecha_inicio, fecha_fin	Búsqueda de promociones vigentes por rango de fechas

Análisis de Rendimiento (EXPLAIN)

Herramientas:

EXPLAIN y SET PROFILING = 1 para analizar:

- Tipo de acceso (ALL, range, ref)
- Número de filas escaneadas
- Uso real de índices
- Antes: table scan completo (~45 rows)
- Después: uso de índice compuesto idx_platos_cat_precio (~16 rows)

Resultado: Reducción significativa del costo de consulta y mejora de tiempos.

```
SELECT id_plato, nombre, precio
FROM platos
WHERE id_categoria = 3 AND precio < 8.00;
```

Análisis de Rendimiento (EXPLAIN)

```
930
931 • show profiles;
932
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

Query_ID	Duration	Query
20	0.00075225	DROP INDEX idx_platos_cat_precio ON platos
21	0.00120025	show index from platos
22	0.00125400	show index from platos
23	0.00033275	explain select id_plato, nombre, descripcion, ...
24	0.00014200	set foreign_key_checks=1
25	0.00016625	SET profiling = 1
26	0.00009750	SHOW WARNINGS
27	0.00040800	select id_plato, nombre, descripcion, precio ...

Result 57 x

```
932
933 • explain select
934   id_plato, nombre, descripcion, precio
935   from platos
936   where id_categoria = 3 and precio < 8.00;
937
938
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	platos	NULL	range	idx_platos_cat_precio	idx_platos_cat_precio	7	NULL	16	100.00	Using index condition

```
-- Despues de index --- F2
-- Creamos un índice compuesto. El orden de las columnas es importante.
-- Ponemos primero la columna con la igualdad (=) y luego la del rango (<).
• create index idx_platos_cat_precio ON platos(id_categoria, precio);

• SET profiling = 1;

• select
  id_plato, nombre, descripcion, precio
  from platos
  where id_categoria = 3 AND precio < 8.00;

• show profiles;
```

Análisis de Rendimiento (EXPLAIN)

```
941 • explain select id_plato, nombre, precio from platos where nombre like 'tarta%';
```

```
942
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	platos	NULL	ALL	NULL	NULL	NULL	NULL	495	11.11	Using where

```
944 -- creamos el indice que tenías para el nombre
```

```
945 • create index idx_platos_nombre on platos(nombre);
```

```
946
```

```
947 • explain select id_plato, nombre, precio from platos where nombre like 'tarta%';
```

```
948
```

```
949
```

```
950
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	platos	NULL	range	idx_platos_nombre	idx_platos_nombre	402	NULL	4	100.00	Using index condition



Seguridad Implementada

Roles y Privilegios



Administrador

Acceso completo a todas las tablas y funciones del sistema



Auditor

Acceso de solo lectura a todas las tablas y registros de auditoría



Operador

Acceso limitado a funciones de ventas y gestión de comandas

Espacio para insertar diagrama de roles y privilegios



Cifrado de Datos



AES-256

Cifrado de correos electrónicos y datos personales sensibles



SHA-256

Hash seguro para contraseñas y credenciales de acceso

Espacio para insertar diagrama de cifrado

Auditoría y Trazabilidad

Tabla log_acciones

Campo	Tipo	Descripción
id_log	INT	ID único
accion	VARCHAR(50)	"INSERT", "UPDATE", "DELETE"
tabla	VARCHAR(50)	Nombre de la tabla afectada
usuario	VARCHAR(100)	Usuario que realizó la acción
fecha_hora	DATETIME	Momento de la acción

3 Key Benefits of Data Analytics and AI in Auditing



Big Data
Modernization for
Operational Efficiency



Advanced
Data Visualization
Capabilities



Intelligent
Document
Processing

Trigger de Ejemplo

```
DELIMITER //

CREATE TRIGGER after_cliente_update
AFTER UPDATE ON clientes
FOR EACH ROW
BEGIN
    -- Registrar la acción en la tabla de auditoría
    INSERT INTO log_acciones (accion, tabla, usuario,
    fecha_hora)

    VALUES ('UPDATE', 'clientes',
    CURRENT_USER (), NOW());
END //
```

Protección ante ataques

Vulnerabilidades Corregidas



SQL Injection

Implementación de consultas parametrizadas y validación de entrada para prevenir inyecciones SQL en formularios de búsqueda y login.



XSS (Cross-Site Scripting)

Sanitización de datos de entrada y salida para prevenir la ejecución de scripts maliciosos en el navegador.



Exposición de información sensible

Ocultamiento de mensajes de error detallados y cifrado de datos sensibles en la base de datos.



Prueba de SQL Injection

Código Vulnerable

```
$username = $_POST['username'];  
$query = "SELECT * FROM usuarios  
        WHERE username = '$username'";  
  
// Vulnerable a: ' OR '1'='1
```

Código Seguro

```
$username = $_POST['username'];  
$stmt = $conn->prepare("SELECT * FROM usuarios  
                      WHERE username = ?");
```


Procedimientos Clave

Procedimientos Almacenados



sp_generar_factura

Genera automáticamente una factura a partir de una comanda, calculando subtotales, impuestos y aplicando descuentos si corresponde.



sp_eliminar_cliente_seguro

Elimina un cliente verificando primero que no tenga facturas pendientes o comandas activas, garantizando la integridad referencial.



sp_reporte_ventas_periodo

Genera un reporte detallado de ventas por período, categoría de producto y mesero, con cálculos de rendimiento.



Ejemplo de Procedimiento

```
DELIMITER //
```

```
CREATE PROCEDURE sp_generar_factura(  
    IN p_comanda_id INT,  
    IN p_cliente_id INT,  
    OUT p_factura_id INT  
)  
  
BEGIN  
    DECLARE v_subtotal DECIMAL(10,2);  
    DECLARE v_impuestos DECIMAL(10,2);  
    DECLARE v_total DECIMAL(10,2);
```

Capturas del Entorno

Conexión SSL



Configuración de conexión segura mediante SSL para proteger la transmisión de datos.

✔ Certificados X.509

✔ Cifrado TLS 1.3



Verificación de conexión segura mediante herramientas de diagnóstico.

Workpulse

The Role of AI
in modern restaurant
management systems



Backup Completo



Proceso automatizado de backup completo con verificación de integridad.

✔ Backups diarios

✔ Verificación automática

Código Fuente

Implementaciones SQL



Definición de Tablas (DDL)

Estructura normalizada de tablas con relaciones, claves primarias y foráneas, e índices para optimizar consultas.



Transacciones (DML)

Operaciones ACID con control de transacciones para mantener la integridad de los datos en operaciones críticas.



Consultas Optimizadas

Consultas complejas con JOINS optimizados y subconsultas eficientes para reportes y análisis de datos.



Control de Acceso

Implementación de roles y permisos granulares a nivel de tabla y columna para seguridad de datos.

** El código mostrado ha sido validado con herramientas de análisis estático para detectar vulnerabilidades y optimizar rendimiento.*

Ejemplos de Código

DDL



Espacio para insertar código SQL de definición de tablas

DML



Espacio para insertar código SQL de manipulación de datos

Conclusión

Fortalezas Destacables



Diagrama ER validado

Estructura normalizada y validada con herramientas profesionales (MySQL Workbench).



Relaciones complejas bien implementadas

Relaciones N:M para gestión de recetas y uso de ON DELETE CASCADE en dependencias críticas.



Automatización y lógica de negocio

Procedimientos almacenados robustos para facturación automática y eliminación segura con validación.



Seguridad básica implementada

Roles bien definidos y cifrado demostrativo para datos sensibles.



Recomendaciones



Optimización de consultas

Implementar índices adicionales para mejorar el rendimiento en tablas de alta frecuencia.



Reforzar seguridad

Implementar cifrado completo para todos los datos sensibles y auditoría exhaustiva.

El sistema cumple con los requisitos funcionales básicos y tiene una estructura bien normalizada, pero debe priorizar las mejoras de seguridad y rendimiento.