Presentación (Guía para presentar)

PROCEDIMIENTOS ALMACENADOS Y SUS TRIGGERS

1. sp_asignar_upgrade_vip

Descripción: Asigna automáticamente mejores habitaciones a clientes VIP cuando hay disponibilidad

Triggers que ACTIVA:

- `trg_upgrade_vip_bitacora` - Se ejecuta al UPDATE de detalle_reserva_habitacion
- `trg_habitacion_upgrade_estado` - Se ejecuta al UPDATE de habitacion
- `trg_bitacora_habitacion` - Se ejecuta al cambiar estado de habitación

Flujo del Proceso:

1. Procedimiento identifica cliente VIP
2. Busca habitación superior disponible
3. **UPDATE detalle_reserva_habitacion** → ACTIVA `trg_upgrade_vip_bitacora`
4. **UPDATE fechas_ocupadas** → Libera habitación anterior
5. **UPDATE habitacion** → ACTIVA `trg_bitacora_habitacion`

---

2. sp_cancelar_reserva

Descripción: Proceso completo de cancelación con cálculo automático de penalizaciones

Triggers que ACTIVA:

- `trg_cancelacion_reserva` - Se ejecuta al UPDATE de reserva (cambio a 'Cancelada')
- `trg_aplicar_penalizacion_cancelacion_mejorado` - Se ejecuta al INSERT en cancelacion

Flujo del Proceso:

1. Procedimiento calcula penalización (55% si <3 días)
2. **INSERT INTO cancelacion** → ACTIVA `trg_aplicar_penalizacion_cancelacion_mejorado`
3. **UPDATE reserva** SET Estado='Cancelada' → ACTIVA `trg_cancelacion_reserva`
4. Triggers automáticamente crean factura y liberan habitaciones

---

3. sp_registrar_reserva_con_temporada

Descripción: Registra nueva reserva aplicando descuentos de temporada y promociones

Triggers que ACTIVA:

- `trg_validar_fechas_reserva` - Se ejecuta BEFORE INSERT en reserva
- `trg_actualizar_contador_vip` - Se ejecuta AFTER INSERT en reserva
- `trg_cliente_potencial_vip` - Ya ejecutado al crear cliente

Flujo del Proceso:

1. Procedimiento valida fechas y promociones
2. **INSERT INTO reserva** → ACTIVA `trg_validar_fechas_reserva` (validación)
3. Mismo INSERT → ACTIVA `trg_actualizar_contador_vip` (actualiza VIP)
4. Al asignar habitaciones → ACTIVA `trg_control_inventario_habitaciones`

---

4. sp_registrar_reserva_validacion_fechas

Descripción: Registra reserva con validación estricta de disponibilidad de habitaciones

Triggers que ACTIVA:

- `trg_validar_fechas_reserva` - Validación BEFORE INSERT
- `trg_control_inventario_habitaciones` - Al INSERT detalle_reserva_habitacion
- `trg_actualizar_contador_vip` - AFTER INSERT reserva

Flujo del Proceso:

1. **INSERT reserva** → `trg_validar_fechas_reserva` + `trg_actualizar_contador_vip`

2. **INSERT detalle_reserva_habitacion** → `trg_control_inventario_habitaciones`
3. **INSERT fechas_ocupadas** → Control de inventario

---

5. sp_display_checkout_total / sp_factura_rapida

Descripción: Calcula total completo para facturación (hospedaje + servicios + penalizaciones)

Triggers que PUEDE USAR:

- No activa triggers directamente (solo consulta)
- Pero LEE datos creados por `trg_aplicar_penalizacion_cancelacion_mejorado`

Flujo del Proceso:

1. Calcula total hospedaje
2. Suma servicios no pagados
3. Suma penalizaciones pendientes (creadas por triggers)
4. Muestra desglose completo

---

6. sp_actualizar_cliente_vip

Descripción: Evalúa y promociona clientes a estatus VIP

Triggers que ACTIVA:

- No activa triggers (solo UPDATE directo)
- Pero usa datos mantenidos por `trg_actualizar_contador_vip`

Flujo del Proceso:

1. Lee datos de cliente_potencial_vip (mantenida por triggers)
2. **UPDATE cliente** SET Es_VIP=1 (sin trigger)
3. **UPDATE cliente_potencial_vip** Estado='VIP Activo'

---

7. sp_actualizar_estado_habitacion

Descripción: Cambia manualmente el estado de una habitación

Triggers que ACTIVA:

- `trg_bitacora_habitacion` - Se ejecuta al UPDATE habitacion

Flujo del Proceso:

1. **UPDATE habitacion** SET Estado=nuevo_estado → ACTIVA `trg_bitacora_habitacion`
2. **INSERT bitacora_habitacion** (manual adicional)

---

8. sp_registrar_servicio

Descripción: Registra consumo de servicios adicionales

Triggers que ACTIVA:

- No activa triggers directamente
- Pero crea datos que luego usan triggers de facturación

Flujo del Proceso:

1. **INSERT consumo_servicio** (sin triggers en esta tabla)
2. Datos quedan disponibles para sp_display_checkout_total

---

9. sp_procesar_pago_penalizacion

Descripción: Procesa pago de facturas de penalización

Triggers que ACTIVA:

- No activa triggers (solo UPDATE de estado)

Flujo del Proceso:

1. **UPDATE factura** SET Estado='Pagada' (sin triggers)
2. Actualiza método de pago y número de transacción

---

10. sp_verificar_disponibilidad_habitaciones

Descripción: Consulta habitaciones disponibles para fechas específicas

Triggers que USA:

- No activa triggers (solo consulta)

- Lee datos mantenidos por `trg_control_inventario_habitaciones`

Flujo del Proceso:

1. Consulta tabla fechas_ocupadas (poblada por triggers)
2. Filtra habitaciones disponibles

TRIGGERS: TABLA Y PROCESO DONDE SE USAN

trg_aplicar_penalizacion_cancelacion_mejorado

- **Tabla:** `cancelacion`
- **Proceso:** AFTER INSERT
- **Se usa en:** Cuando se registra una cancelación con penalización

Código para probar:

```
-- Insertar una cancelación con penalización
INSERT INTO cancelacion (
    Motivo_Cancelacion,
    Penalizacion,
    Fecha_Cancelacion,
    Codigo_Confirmacion
) VALUES (
    'Cancelación tardía del cliente',
    2500.00,
    NOW(),
    'RES-20250527-12345'
);

-- Verificar que se creó la factura automáticamente
SELECT * FROM factura WHERE Numero_Transaccion LIKE 'CANCEL-%' ORD
ER BY ID_Factura DESC LIMIT 1;
```

---

trg_actualizar_penalizacion_cancelacion

- **Tabla:** `cancelacion`
- **Proceso:** AFTER UPDATE
- **Se usa en:** Cuando se modifica el monto de penalización

Código para probar:

```
-- Actualizar monto de penalización existente
UPDATE cancelacion
SET Penalizacion = 3000.00
WHERE ID_Cancelacion = 1;

-- Verificar que se actualizó la factura correspondiente
SELECT Total FROM factura WHERE Numero_Transaccion LIKE 'CANCEL-%-
1';
```

---

trg_cliente_potencial_vip

- **Tabla:** `cliente`
- **Proceso:** AFTER INSERT
- **Se usa en:** Cuando se registra un nuevo cliente

Código para probar:

```
-- Insertar nuevo cliente
INSERT INTO cliente (
    Nombre, Apellido1, Email, RFC, Fecha_Registro, ID_Ciudad_Origen
) VALUES (
    'Juan', 'Pérez', 'juan@email.com', 'PEPJ850315', CURDATE(), 1
);

-- Verificar que se creó registro en cliente_potencial_vip
SELECT * FROM cliente_potencial_vip WHERE ID_Cliente = LAST_INSERT_ID();
```

---

trg_control_inventario_habitaciones

- **Tabla:** `detalle_reserva_habitacion`
- **Proceso:** AFTER INSERT
- **Se usa en:** Cuando se asigna una habitación a una reserva

Código para probar:

```
-- Insertar detalle de reserva con habitación
INSERT INTO detalle_reserva_habitacion (
    ID_Reserva, ID_Habitacion, Precio_Noche
) VALUES (
    1, 101, 1500.00
);

-- Verificar que se creó registro en fechas_ocupadas
SELECT * FROM fechas_ocupadas WHERE ID_Habitacion = 101 AND ID_Reser
va = 1;
```

---

trg_habitacion_ocupada

- **Tabla:** `detalle_reserva_habitacion`
- **Proceso:** AFTER UPDATE
- **Se usa en:** Cuando se hace check-in (se llena Fecha_Check_In)

Código para probar:

```
-- Hacer check-in (actualizar fecha de entrada)
UPDATE detalle_reserva_habitacion
SET Fecha_Check_In = NOW()
WHERE ID_Detalle_Reserva_Habitacion = 1;

-- Verificar que la habitación cambió a 'Ocupado'
SELECT Estado FROM habitacion WHERE ID_Habitacion = 101;
```

---

trg_habitacion_disponible

- **Tabla:** `detalle_reserva_habitacion`
- **Proceso:** AFTER UPDATE
- **Se usa en:** Cuando se hace check-out (se llena Fecha_Check_Out)

Código para probar:

```
-- Hacer check-out (actualizar fecha de salida)
UPDATE detalle_reserva_habitacion
SET Fecha_Check_Out = NOW()
WHERE ID_Detalle_Reserva_Habitacion = 1;

-- Verificar que la habitación cambió a 'Disponible'
SELECT Estado FROM habitacion WHERE ID_Habitacion = 101;
```

---

trg_upgrade_vip_bitacora

- **Tabla:** `detalle_reserva_habitacion`
- **Proceso:** AFTER UPDATE
- **Se usa en:** Cuando se cambia la habitación asignada (upgrade)

Código para probar:

```
-- Cambiar habitación (simular upgrade)
UPDATE detalle_reserva_habitacion
SET ID_Habitacion = 201
WHERE ID_Detalle_Reserva_Habitacion = 1;

-- Verificar registro en bitácora
SELECT * FROM bitacora_habitacion
WHERE Motivo_cambio LIKE '%VIP%'
ORDER BY Fecha_Hora_Cambio DESC LIMIT 2;
```

---

trg_bitacora_habitacion

- **Tabla:** `habitacion`
- **Proceso:** AFTER UPDATE
- **Se usa en:** Cuando cambia el estado de una habitación

Código para probar:

```
-- Cambiar estado de habitación
UPDATE habitacion
SET Estado = 'Fuera de servicio'
WHERE ID_Habitacion = 101;

-- Verificar registro en bitácora
SELECT * FROM bitacora_habitacion
WHERE ID_Habitacion = 101
ORDER BY Fecha_Hora_Cambio DESC LIMIT 1;
```

---

trg_habitacion_upgrade_estado

- **Tabla:** `habitacion`
- **Proceso:** AFTER UPDATE
- **Se usa en:** Cuando hay cambios de estado relacionados con VIP

Código para probar:

```
-- Simular cambio de estado con reserva VIP activa-- Primero verificar que ha
y una reserva VIP para la habitación
UPDATE habitacion
SET Estado = 'Ocupado'
WHERE ID_Habitacion = 201;

-- Verificar si se registró con contexto VIP
SELECT * FROM bitacora_habitacion
WHERE ID_Habitacion = 201 AND Motivo_cambio LIKE '%VIP%'
ORDER BY Fecha_Hora_Cambio DESC LIMIT 1;
```

---

trg_validar_fechas_reserva

- **Tabla:** `reserva`
- **Proceso:** BEFORE INSERT
- **Se usa en:** Antes de crear una nueva reserva (validación)

Código para probar:

```
-- Intentar insertar reserva con fechas inválidas (debe fallar)
INSERT INTO reserva (
    ID_Cliente, Fecha_Entrada, Fecha_Salida, Estado, Fecha_Creacion
) VALUES (
    1, '2025-06-15', '2025-06-10', 'Confirmada', NOW()
);
-- Error: La fecha de salida debe ser posterior a la fecha de entrada-- Insertar reserva con fechas válidas (debe funcionar)
INSERT INTO reserva (
    ID_Cliente, Fecha_Entrada, Fecha_Salida, Estado, Fecha_Creacion
) VALUES (
    1, '2025-06-10', '2025-06-15', 'Confirmada', NOW()
);
-- Quiero llorar, ya no quiero hacer nada, si ve esto quemamos demasiado tie mpo haciendo esto
```

---

trg_actualizar_contador_vip

- **Tabla:** `reserva`
- **Proceso:** AFTER INSERT
- **Se usa en:** Después de crear una reserva para cliente VIP

Código para probar:

```
-- Primero convertir cliente a VIP
UPDATE cliente SET Es_VIP = 1 WHERE ID_Cliente = 1;
```

```
-- Insertar nueva reserva para cliente VIP
INSERT INTO reserva (
    ID_Cliente, Fecha_Entrada, Fecha_Salida, Estado, Fecha_Creacion
) VALUES (
    1, '2025-07-01', '2025-07-05', 'Confirmada', NOW()
);

-- Verificar que se actualizó el contador VIP
SELECT Contador_Reservas, Ultima_Estancia
FROM cliente_potencial_vip
WHERE ID_Cliente = 1;
```

---

trg_cancelacion_reserva

- **Tabla:** `reserva`
- **Proceso:** BEFORE UPDATE
- **Se usa en:** Cuando se cambia el estado de reserva a 'Cancelada'

Código para probar:

```
-- Cancelar una reserva existente
UPDATE reserva
SET Estado = 'Cancelada'
WHERE ID_Reserva = 1;

-- Verificar que se creó registro de cancelación
SELECT * FROM cancelacion
WHERE Codigo_Confirmacion = (
    SELECT Codigo_Confirmacion FROM reserva WHERE ID_Reserva = 1
);

-- Verificar que las habitaciones se liberaron
SELECT Estado FROM habitacion
WHERE ID_Habitacion IN (
    SELECT ID_Habitacion FROM detalle_reserva_habitacion WHERE ID_Reserv
```

```
a = 1
);
```

---

trg_validar_precio_servicio

- **Tabla:** `servicio`
- **Proceso:** BEFORE INSERT
- **Se usa en:** Antes de registrar un nuevo servicio

Código para probar:

```
-- Intentar insertar servicio con precio inválido (debe fallar)
INSERT INTO servicio (
    Nombre, Tipo, Precio, Departamento
) VALUES (
    'Servicio de prueba', 'Alimentación', 0.00, 1
);
-- Error: El precio del servicio debe ser mayor que cero-- Insertar servicio con precio válido (debe funcionar)
INSERT INTO servicio (
    Nombre, Tipo, Precio, Departamento
) VALUES (
    'Servicio de prueba', 'Alimentación', 25.00, 1
);
```

---

trg_validar_precio_servicio_update

- **Tabla:** `servicio`
- **Proceso:** BEFORE UPDATE
- **Se usa en:** Antes de actualizar el precio de un servicio

Código para probar:

```
-- Intentar actualizar con precio inválido (debe fallar)
UPDATE servicio
```

```
SET Precio = -10.00
WHERE ID_Servicio = 1;
-- Error: El precio del servicio debe ser mayor que cero-- Actualizar con preci
o válido (debe funcionar)
UPDATE servicio
SET Precio = 30.00
WHERE ID_Servicio = 1;
```