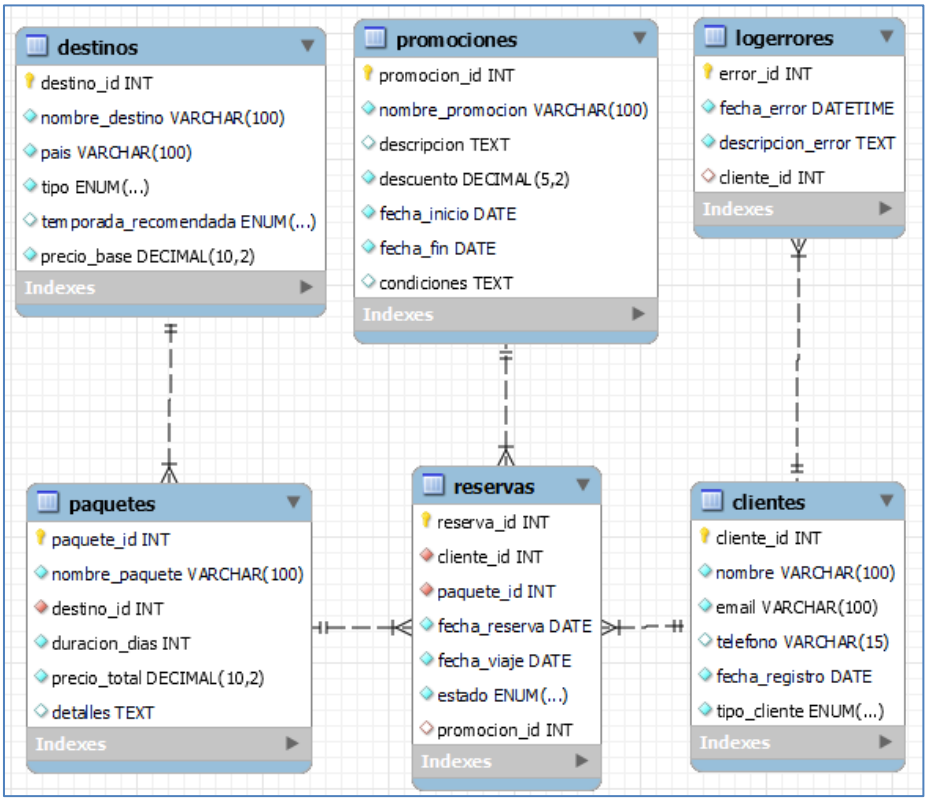
Apellido y Nombre del Estudiante:	Cantidad de hojas entregadas:	Fecha:	Calificación:

Tecnicatura Superior en Programación
Base de Datos II
Primer Parcial

- Resuelva los ejercicios en hoja separada y con lapicera.
- Como condición de aprobación se debe tener completado el 50% de cada parte para aprobar.
- Como condición para mantener la posibilidad de promoción, se debe tener completado CORRECTAMENTE al menos el 70% de cada ejercicio.
- Si la letra es ilegible se descontarán puntos.
- Ante cualquier duda en la interpretación consulte al profesor o escriba su interpretación para que sea tomada en cuenta.
- Escriba su nombre en TODAS LAS HOJAS y enumérelas. Cuando tenga el total de páginas anótelo en la parte superior de esta hoja.

Dado el siguiente DER



<u>Ejercicio 1: UDF</u>

Crear una función llamada `calcular_descuento` que reciba como parámetros el precio original de un paquete turístico y el porcentaje de descuento aplicado. La función debe devolver el precio final después de aplicar el descuento. Asegúrese de manejar casos en los que el porcentaje de descuento sea inválido (por ejemplo, menor que 0 o mayor que 100).

Ejercicio 2: VISTA

Crea una vista llamada `ReservasPorCliente` que muestre información consolidada de las reservas realizadas por cada cliente. La vista debe incluir los siguientes campos:

- ID del cliente.
- Nombre del cliente.
- Cantidad total de reservas realizadas.
- Fecha de la última reserva.

Ejercicio 3: SP

Crear un procedimiento almacenado llamado `registrar_reserva` que permita insertar una nueva reserva en la tabla `Reservas`. El procedimiento debe recibir como parámetros:

- ID del cliente.
- ID del paquete turístico.
- Fecha del viaje.

El procedimiento debe validar que el cliente y el paquete existan en sus respectivas tablas. Si alguno de ellos no existe, debe registrar un error en la tabla `LogErrores` y abortar la operación. Además, debe utilizar transacciones para garantizar la integridad de los datos.

Ejercicio 4: TRIGGER

Crear un trigger llamado `validar_fecha_reserva` en la tabla `Reservas`. El trigger debe verificar que la fecha del viaje no sea anterior a la fecha de reserva. Si esta condición no se cumple, el trigger debe:

1. Registrar un error en la tabla `LogErrores`.
2. Impedir la inserción del registro lanzando un error.

Ejercicio 5: CTE

Utilizando una CTE, generar una consulta que muestre los destinos más populares. Un destino se considera popular si tiene más de 3 reservas asociadas. La consulta debe incluir:

- ID del destino.
- Nombre del destino.
- Total de reservas realizadas.

A. PARTE TEORICA

Ejercicio 1: ¿Qué motor de almacenamiento sería ideal en los siguientes escenarios?

1. Una base de datos de producción que realiza muchas actualizaciones y necesita garantizar integridad referencial:
   a) MEMORY
   b) InnoDB
   c) MyISAM
   d) Ninguno de los anteriores
2. Una base de datos para un sistema de análisis temporal que necesita almacenar los datos únicamente en memoria para operaciones extremadamente rápidas:
   a) InnoDB
   b) MEMORY
   c) MyISAM
   d) CSV
3. Un sistema con tablas que realizan principalmente consultas de solo lectura y procesamiento intensivo de texto:
   a) MEMORY
   b) InnoDB
   c) MyISAM
   d) ARCHIVE

Ejercicio 2: Dado el siguiente escenario:

- La tabla `Clientes` tiene un campo `email` que es único y se busca frecuentemente en consultas de tipo igualdad.
- Los resultados se ordenan frecuentemente por `fecha_registro`.

a) ¿Qué tipo de índice recomendaría para `email` y por qué?
   a) Índice agrupado, porque mejora las búsquedas por igualdad.
   b) Índice no agrupado, porque las búsquedas por igualdad no necesitan recorrer toda la tabla.
   c) Índice de texto completo, porque mejora búsquedas por patrones como '%example.com'.
   d) No usar índice, ya que la tabla tiene pocos registros.

b) ¿Qué ocurriría si `fecha_registro` también tuviera un índice?
   a) No cambiaría nada, ambos índices son independientes.
   b) Reduciría el rendimiento de las actualizaciones de la tabla, pero las búsquedas serían más rápidas.
   c) Haría que `email` se convierta en un índice secundario automáticamente.
   d) Sería preferible usar un índice compuesto (`email`, `fecha_registro`).

Ejercicio 3: En una tabla `Promociones` con las columnas `descripcion` y `descuento`:

a) Si los usuarios buscan palabras clave en `descripcion`, ¿qué tipo de índice recomendaría?
   a) Índice B-Tree
   b) Índice HASH
   c) Índice Full-Text
   d) No se puede optimizar esta consulta.

b) Si los usuarios filtran por rangos de `descuento`, ¿qué índice sería más eficiente?
   a) Índice Full-Text
   b) Índice HASH
   c) Índice B-Tree
   d) No se requiere índice porque los rangos son muy pequeños.

Ejercicio 4 (V/F):

- a) Para consultas por igualdad, un índice HASH es más eficiente que un índice B-Tree. Verdadero / Falso
- b) Para consultas combinadas de varias columnas, un índice B-Tree es preferible a un índice HASH. Verdadero / Falso
- c) En el motor MEMORY, los índices B-Tree son la única opción. Verdadero / Falso

Ejercicio 5:

1. Para optimizar búsquedas por `estado` (con pocos valores únicos), usaría:
   a) Índice HASH
   b) Índice Full-Text
   c) Índice B-Tree
   d) No usaría índice, porque la cardinalidad es baja.
2. Si agregamos un índice en `fecha_reserva` para optimizar el ordenamiento, su impacto sería:
   a) Aumentaría el tiempo de inserción, pero aceleraría las consultas con ordenamiento.
   b) Reduciría significativamente el tiempo de ordenamiento, pero aumentaría el tamaño de la tabla.
   c) Mejoraría tanto las inserciones como las consultas de ordenamiento.
   d) No tendría impacto porque los índices no afectan el ordenamiento.

3. Si la tabla `Reservas` estuviera en MyISAM en lugar de InnoDB, ¿qué cambiaría respecto al manejo de índices agrupados?
   a) No habría cambios, porque ambos motores soportan índices agrupados.
   b) MyISAM no soporta índices agrupados, por lo que el rendimiento sería inferior.
   c) MyISAM usa índices Full-Text por defecto, eliminando la necesidad de índices agrupados. d) No se podrían usar índices en MyISAM si la tabla tiene más de un millón de filas.

Ejercicio 6: ¿Qué puede concluir de la siguiente vista?

```
CREATE VIEW VistaComplejaLocal AS
SELECT productos.nombre, pedidos.fecha, pedidos.cantidad
FROM productos
JOIN pedidos ON productos.id_producto = pedidos.id_producto
WHERE pedidos.cantidad > 5
WITH LOCAL CHECK OPTION;
```

a) Es una vista que permite manipular datos, solo si cumplen con la condición definida en el WHERE.
b) Es una vista que en la cual el uso de WITH CHECK OPTION no tiene sentido práctico.
c) El uso de WITH CHECK OPTION no garantiza que se verificaran los datos en las tablas subyacentes.
d) El uso de WITH CHECK OPTION mejora el rendimiento de las consultas realizadas a través de la vista.
e) Ninguna de las anteriores.

Ejercicio 7: ¿La siguiente UDF es deterministica/no deterministica? Justifique

```
DELIMITER //
CREATE FUNCTION Calcular(Base DECIMAL(10,2))
RETURNS DECIMAL(10,2)
NOT DETERMINISTIC
BEGIN
    DECLARE factor DECIMAL(3,2);
    SET factor = RAND() * (0.15 - 0.05) + 0.05;
    RETURN Base * (1 - factor);
END //
DELIMITER ;
```

Ejercicio 8: Complete, *si lo considera necesario*, el siguiente SP para garantizar la consistencia de los datos luego de ejecutarse el mismo. Es decir, pasar de un estado consistente a otro consistente.

```
DELIMITER //
CREATE PROCEDURE TransferirDinero(
   IN cuentaOrigen INT,
   IN cuentaDestino INT,
   IN monto DECIMAL(10,2)
)
BEGIN

   DECLARE saldoActual DECIMAL(10,2);
   SELECT saldo INTO saldoActual
   FROM cuentas
   WHERE id_cuenta = cuentaOrigen;

   IF saldoActual < monto THEN

      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Saldo insuficiente para realizar la transferencia';
      LEAVE;

   END IF;

   UPDATE cuentas
   SET saldo = saldo - monto
   WHERE id_cuenta = cuentaOrigen;

   UPDATE cuentas
   SET saldo = saldo + monto
   WHERE id_cuenta = cuentaDestino;

END //
DELIMITER ;
```