PROYECTO DB SQL

Leandro Picazo

INTRODUCCIÓN

Descripción de la Situación de Negocio:

La inmobiliaria LP se especializa en la intermediación y gestión de propiedades residenciales y comerciales en una determinada región o área geográfica. La empresa tiene como objetivo proporcionar un servicio integral a sus clientes, que incluye la compra, venta y alquiler de propiedades, así como servicios de gestión inmobiliaria, asesoramiento legal y financiero.

Actualmente, la inmobiliaria maneja una gran cantidad de información sobre propiedades, clientes, transacciones, contratos de arrendamiento y venta, y datos financieros relacionados. Sin embargo, la gestión de estos datos se ha vuelto compleja y desorganizada debido a la falta de un sistema centralizado y eficiente.

Objetivos de la Base de Datos:

- **Centralización de la Información:** Crear una base de datos centralizada que almacene información detallada sobre propiedades disponibles, clientes, transacciones pasadas y presentes, contratos y datos financieros.

- **Mejora en la Gestión de Propiedades:** Facilitar la gestión de propiedades, incluyendo detalles como ubicación, características, precios, estado legal, historial de transacciones, entre otros.

- **Optimización de Procesos:** Agilizar y optimizar los procesos de búsqueda y emparejamiento entre propiedades y clientes potenciales, así como la gestión de contratos y transacciones.

- **Análisis y Reporting:** Facilitar la generación de informes y análisis que proporcionen información valiosa para la toma de decisiones estratégicas, como tendencias del mercado inmobiliario, rendimiento financiero y preferencias de los clientes.

Problemas a Resolver en la Base de Datos:

1. **Desorganización de Datos:** La información dispersa en diferentes formatos y ubicaciones dificulta la consulta y actualización eficiente de datos.

2. **Inconsistencia de Información:** Existen duplicaciones, discrepancias y falta de integridad en los datos, lo que puede llevar a errores y decisiones erróneas.

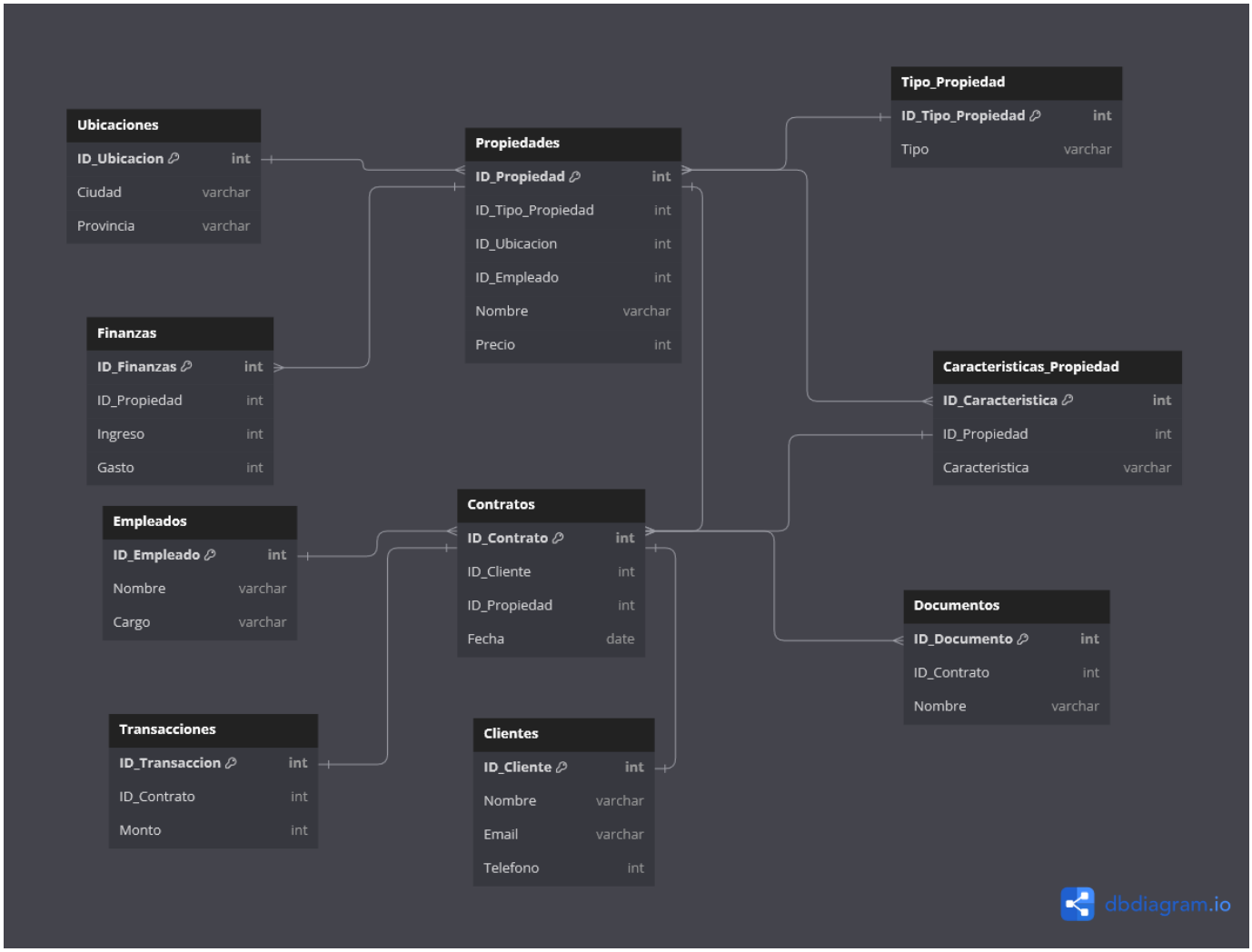
3. **Dificultad en la Búsqueda y Filtrado:** La falta de un sistema estructurado dificulta la búsqueda rápida y precisa de propiedades que se ajusten a los criterios específicos de los clientes.

4. **Gestión Ineficiente de Contratos:** La falta de seguimiento centralizado de contratos y vencimientos puede resultar en renovaciones tardías o pérdida de oportunidades.

5. **Falta de Análisis Estratégico:** La ausencia de herramientas para el análisis de datos limita la capacidad de la empresa para identificar tendencias, oportunidades de mercado y mejorar la toma de decisiones.

Entidades y relaciones.

Diagrama Entidad-relación.



Descripción de tablas.

Tabla: Propiedades.

Esta tabla almacena información sobre las propiedades disponibles para venta o alquiler, incluyendo detalles como nombre, precio, tipo de propiedad, ubicación y empleado asociado.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_PROPIEDAD	INT	Id de cada propiedad, PK, dato único y autoincrementado.
FK	ID_TIPO_PROPIEDAD	INT	ID FK que relaciona la tabla mencionada con Propiedades.
FK	ID_UBICACION	INT	ID FK que relaciona la tabla mencionada con Propiedades.
FK	ID_EMPLEADO	INT	ID FK que relaciona la tabla mencionada con Propiedades.
	NOMBRE	VARCHAR(60)	Nombre de la propiedad, identificador textual de cada propiedad.
	PRECIO	DECIMAL	Valor de la propiedad.

Tabla: Tipo Propiedad.

Esta tabla contiene los tipos de propiedades disponibles, como casas, apartamentos, terrenos, etc., asociados con un identificador único.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_TIPO_PROPIEDAD	INT	Id del tipo de la propiedad, PK, dato único y autoincrementado.
FK	TIPO	VARCHAR(50)	Tipo de propiedad (casa, depto, etc.)

Tabla: Contratos.

Registra los contratos celebrados con los clientes, incluyendo la fecha, el cliente asociado y la propiedad correspondiente.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_CONTRATO	INT	Id de cada contrato, PK, dato único y autoincrementado.
FK	ID_CLIENTE	INT	ID FK que relaciona la tabla mencionada con Contratos
FK	ID_PROPIEDAD	INT	ID FK que relaciona la tabla mencionada con Contratos
	FECHA	DATE	Fecha de cada transacción.

Tabla: Clientes.

Almacena información detallada sobre los clientes, como nombre, correo electrónico y teléfono, identificados por un número único.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_CLIENTE	INT	Id de cada cliente, PK, dato único y autoincrementado.
	NOMBRE	VARCHAR(50)	Nombre de cada cliente, identificador textual.
	EMAIL	VARCHAR(90)	Email de contacto de cada cliente.
	TELEFONO	VARCHAR(50)	Teléfono de contacto de cada cliente.

Tabla: Empleados.

Contiene datos de los empleados de la inmobiliaria, como nombre y cargo, asociados con un identificador único.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_EMPLEADO	INT	Id de cada empleado, PK, dato único y autoincrementado.
	NOMBRE	VARCHAR(50)	Nombre de cada empleado, identificador textual.
	CARGO	VARCHAR(50)	Cargo/puesto dentro de la inmobiliaria.

Tabla: Transacciones.

Registra las transacciones financieras asociadas a los contratos, incluyendo el monto de la transacción y el contrato correspondiente.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_TRANSACCIÓN	INT	Id de cada transacción, PK, dato único y autoincrementado.
FK	ID_CONTRATO	INT	ID FK que relaciona la tabla mencionada con Transacciones,
	MONTO	DECIMAL	Monto de cada transacción (Venta, alquiler, etc.)

Tabla: Ubicaciones.

Almacena información sobre las ubicaciones de las propiedades, como la ciudad y la provincia, identificadas por un número único.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_UBICACION	INT	Id de cada ubicación, PK, dato único y autoincrementado.
	CIUDAD	VARCHAR(50)	Ciudad donde está ubicada la propiedad.
	PROVINCIA	VARCHAR(50)	Provincia donde está ubicada la propiedad.

Tabla: Características propiedad.

Registra las características específicas de cada propiedad, asociando cada una con la propiedad correspondiente.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_CARACTERISTICA	INT	Id de cada Propiedad, PK, dato único y autoincrementado.
FK	ID_PROPIEDAD	INT	ID FK que relaciona la tabla mencionada con Características Propiedad.
	CARACTERISTICA	VARCHAR(100)	Característica (visual, territorial, etc) que identifica/diferencia cada propiedad.

Tabla: Finanzas.

Contiene datos financieros de las propiedades, incluyendo ingresos y gastos asociados a cada propiedad.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_FINANZAS	INT	Id de cada Finanzas, PK, dato único y autoincrementado.
FK	ID_PROPIEDAD	INT	ID FK que relaciona la tabla mencionada con Finanzas.
	INGRESO	DECIMAL	Ingresos a la inmobiliaria (Ventas, alquileres, etc.)
	GASTO	DECIMAL	Gasto de la inmobiliaria (Inversiones y demás.)

Tabla: Documentos.

Almacena documentos asociados a los contratos, como archivos PDF, vinculados a un contrato específico.

Tipo clave	Nombre del Campo	Tipo de datos	Descripción
PK	ID_DOCUMENTO	INT	Id de cada Documento, PK, dato único y autoincrementado.
FK	ID_CONTRATO	INT	ID FK que relaciona la tabla mencionada con Documentos.
	NOMBRE	VARCHAR(60)	Nombre de cada documento, necesario para identificar/buscar documentos específicos.

SCRIPTS

Creación de tablas

```sql
-- Crear base de datos de la inmobiliaria
CREATE DATABASE IF NOT EXISTS inmobiliaria_db;

USE inmobiliaria_db;

-- Tabla propiedades
```

```sql
CREATE TABLE IF NOT EXISTS propiedades
  (
     id_propiedad      INT auto_increment PRIMARY KEY,
     id_tipo_propiedad INT,
     nombre            VARCHAR(50) NOT NULL UNIQUE,
     precio            DECIMAL(10, 2) NOT NULL,
     id_ubicacion      INT,
     id_empleado       INT,
     FOREIGN KEY (id_tipo_propiedad) REFERENCES tipo_propiedad(id_tipo_pr
opiedad
     ),
     FOREIGN KEY (id_ubicacion) REFERENCES ubicaciones(id_ubicacion),
     FOREIGN KEY (id_empleado) REFERENCES empleados(id_empleado)
  );

-- Tabla tipo de propiedad
CREATE TABLE IF NOT EXISTS tipo_propiedad
  (
     id_tipo_propiedad INT auto_increment PRIMARY KEY,
     tipo              VARCHAR(50) NOT NULL
  );

-- Tabla Contratos
CREATE TABLE IF NOT EXISTS contratos
  (
     id_contrato  INT auto_increment PRIMARY KEY,
     id_cliente   INT,
     id_propiedad INT,
     fecha        DATETIME,
     FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente),
     FOREIGN KEY (id_propiedad) REFERENCES propiedades(id_propiedad)
  );

-- Tabla Clientes
CREATE TABLE IF NOT EXISTS clientes
  (
     id_cliente     INT auto_increment PRIMARY KEY,
     nombre_cliente VARCHAR(50) NOT NULL,
     email          VARCHAR(90) NOT NULL UNIQUE,
     telefono       VARCHAR(50) NOT NULL
  );

-- Tabla Empleados
CREATE TABLE IF NOT EXISTS empleados
  (
     id_empleado     INT auto_increment PRIMARY KEY,
     nombre_empleado VARCHAR(50) NOT NULL,
     cargo           VARCHAR(50) NOT NULL
  );
```

```sql
-- Tabla Transacciones
CREATE TABLE IF NOT EXISTS transacciones
  (
     id_transaccion INT auto_increment PRIMARY KEY,
     id_contrato    INT,
     monto          DECIMAL(10, 2) NOT NULL
  );

-- Tabla Ubicaciones
CREATE TABLE IF NOT EXISTS ubicaciones
  (
     id_ubicacion INT auto_increment PRIMARY KEY,
     ciudad       VARCHAR(50) NOT NULL,
     provincia    VARCHAR(50) NOT NULL
  );

-- Tabla Caracteristicas de la propiedad
CREATE TABLE IF NOT EXISTS caracteristicas_propiedad
  (
     id_caracteristica INT auto_increment PRIMARY KEY,
     id_propiedad      INT,
     caracteristica    VARCHAR(100) NOT NULL,
     FOREIGN KEY (id_propiedad) REFERENCES propiedades(id_propiedad)
  );

-- Tabla Finanzas
CREATE TABLE IF NOT EXISTS finanzas
  (
     id_finanzas  INT auto_increment PRIMARY KEY,
     id_propiedad INT,
     ingreso      DECIMAL(10, 2) NOT NULL,
     gasto        DECIMAL(10, 2) NOT NULL,
     FOREIGN KEY (id_propiedad) REFERENCES propiedades(id_propiedad)
  );

-- Tabla Documentos
CREATE TABLE IF NOT EXISTS documentos
  (
     id_documento INT auto_increment PRIMARY KEY,
     id_contrato  INT,
     nombre_doc   VARCHAR(60) NOT NULL,
     FOREIGN KEY (id_contrato) REFERENCES contratos(id_contrato)
  );
```

INSERCION DE VALORES

```
INSERT INTO Tipo\_Propiedad (ID\_Tipo\_Propiedad, Tipo)
VALUES
    (null, 'Casa'),
    (null, 'Apartamento'),
    (null, 'Terreno'),
    (null, 'Edificio'),
    (null, 'Local Comercial'),
    (null, 'Oficina'),
    (null, 'Departamento'),
    (null, 'Chalet'),
    (null, 'Duplex'),
    (null, 'Penthouse');


;INSERT INTO Clientes (ID\_Cliente, Nombre\_cliente, Email, Telefono)
VALUES
    (NULL, 'Juan Perez', 'juanperez@gmail.com', '123-456-7890'),
    (NULL, 'Maria Rodriguez', 'mariarodriguez@gmail.com', '987-654-3210'),
    (NULL, 'Carlos Gomez', 'carlosgomez@gmail.com', '111-222-3333'),
    (NULL, 'Laura Fernandez', 'laurafernandez@gmail.com', '555-444-3333'),
    (NULL, 'Pedro Sanchez', 'pedrosanchez@gmail.com', '999-888-7777'),
    (NULL, 'Ana Lopez', 'analopez@gmail.com', '777-666-5555'),
    (NULL, 'Diego Martinez', 'diegomartinez@gmail.com', '444-333-2222'),
```

```
    (NULL, 'Sofia Garcia', 'sofiagarcia@gmail.com', '222-111-0000'),
    (NULL, 'Javier Gonzalez', 'javiergonzalez@gmail.com', '888-777-6666'),
    (NULL, 'Laura Diaz', 'lauradiaz@gmail.com', '666-555-4444');

;INSERT INTO Empleados (Nombre\_empleado, Cargo)
VALUES
    ('Santiago', 'Gerente de marketing de medios'),
    ('Leandro', 'CEO'),
    ('Juan', 'Gerente finanzas'),
    ('Martin', 'Agente de Ventas'),
    ('Pablo', 'Agente de Ventas'),
    ('Catalina', 'CIO'),
    ('Julia', 'Agente de Ventas'),
    ('Maria', 'Agente de Ventas'),
    ('Gaston', 'Asistente Administrativo'),
    ('Nahuel', 'Agente de Ventas');

;INSERT INTO Ubicaciones (Ciudad, Provincia)
VALUES
    ('Buenos Aires', 'Buenos Aires'),
    ('Córdoba', 'Córdoba'),
    ('Rosario', 'Santa Fe'),
    ('Mendoza', 'Mendoza'),
    ('La Plata', 'Buenos Aires'),
    ('San Miguel de Tucumán', 'Tucumán'),
```

```
    ('Mar del Plata', 'Buenos Aires'),

    ('Salta', 'Salta'),

    ('Santa Fe', 'Santa Fe'),

    ('San Juan', 'San Juan');


;INSERT INTO Propiedades (ID\_Propiedad, Nombre, Precio, ID\_Tipo\_Propie
dad, ID\_Ubicacion, ID\_Empleado)


VALUES


  (NULL, 'Casa A', 250000, 1, 1, 1),


  (NULL, 'Casa B', 320000, 2, 2, 2),


  (NULL, 'Apto C', 180000, 3, 1, 3),


  (NULL, 'Terreno D', 150000, 3, 2, 4),


  (NULL, 'Casa E', 280000, 1, 1, 5),


  (NULL, 'Apto F', 200000, 2, 2, 1),


  (NULL, 'Terreno G', 120000, 3, 1, 2),


  (NULL, 'Casa H', 300000, 1, 2, 3),
```

```
 (NULL, 'Casa I', 260000, 1, 1, 4),

 (NULL, 'Apto J', 220000, 2, 2, 5);

;INSERT INTO Contratos (ID\_Contrato, ID\_Cliente, ID\_Propiedad, Fecha)
VALUES
   (NULL, 1, 11, '2023-01-15'),
   (NULL, 2, 12, '2023-02-20'),
   (NULL, 3, 13, '2023-03-10'),
   (NULL, 4, 14, '2023-04-05'),
   (NULL, 5, 15, '2023-05-12'),
   (NULL, 6, 16, '2023-06-18'),
   (NULL, 7, 17, '2023-07-22'),
   (NULL, 8, 18, '2023-08-30'),
   (NULL, 9, 19, '2023-09-10'),
   (NULL, 10, 20, '2023-10-15');

;INSERT INTO Transacciones (ID\_Transaccion, ID\_Contrato, Monto)
VALUES
   (NULL, 101, 250000),
   (NULL, 102, 320000),
   (NULL, 103, 180000),
   (NULL, 104, 150000),
   (NULL, 105, 280000),
   (NULL, 106, 200000),
```

```
    (NULL, 107, 120000),
    (NULL, 108, 300000),
    (NULL, 109, 260000),
    (NULL, 110, 220000);

;INSERT INTO Caracteristicas\_Propiedad (ID\_Caracteristica, ID\_Propieda
d, Caracteristica)
VALUES
    (NULL, 11, 'Jardín amplio'),
    (NULL, 12, 'Vista panorámica'),
    (NULL, 13, 'Cocina moderna'),
    (NULL, 14, 'Patio trasero'),
    (NULL, 15, 'Amplios espacios'),
    (NULL, 16, 'Balcón con vista'),
    (NULL, 17, 'Ubicación céntrica'),
    (NULL, 18, 'Piscina privada'),
    (NULL, 19, 'Diseño exclusivo'),
    (NULL, 20, 'Terraza espaciosa');

;INSERT INTO Finanzas (ID\_Finanzas, ID\_Propiedad, Ingreso, Gasto)
VALUES
    (NULL, 11, 5000, 2000),
    (NULL, 12, 7000, 2500),
    (NULL, 13, 6000, 1800),
    (NULL, 14, 8000, 3000),
```

```
    (NULL, 15, 5500, 2000),
    (NULL, 16, 7500, 2800),
    (NULL, 17, 6500, 2200),
    (NULL, 18, 9000, 3500),
    (NULL, 19, 5800, 2000),
    (NULL, 20, 7200, 2700);

;INSERT INTO Documentos (ID\_Documento, ID\_Contrato, Nombre\_Doc)
VALUES
    (NULL, 31, 'Contrato\_Venta\_Casa\_A'),
    (NULL, 32, 'Contrato\_Venta\_Casa\_B'),
    (NULL, 33, 'Contrato\_Alquiler\_Apto\_C'),
    (NULL, 34, 'Contrato\_Venta\_Terreno\_D'),
    (NULL, 35, 'Contrato\_Venta\_Casa\_E'),
    (NULL, 36, 'Contrato\_Alquiler\_Apto\_F'),
    (NULL, 37, 'Contrato\_Venta\_Terreno\_G'),
    (NULL, 38, 'Contrato\_Venta\_Casa\_H'),
    (NULL, 39, 'Contrato\_Venta\_Casa\_I'),
    (NULL, 40, 'Contrato\_Alquiler\_Apto\_J');

;
```

Inserción de vistas

```
CREATE VIEW VistaContratosDetalles AS
SELECT c.*, p.Nombre AS NombrePropiedad, cl.Nombre_cliente AS NombreClien
te
FROM contratos c
JOIN propiedades p ON c.id_propiedad = p.id_propiedad
JOIN clientes cl ON c.id_cliente = cl.id_cliente;

;CREATE VIEW VistaFinanzasTotales AS
SELECT f.*, p.Nombre AS NombrePropiedad
FROM finanzas f
JOIN propiedades p ON f.id_propiedad = p.id_propiedad;


;CREATE VIEW VistaContratosUbicacion AS
SELECT c.*, u.Ciudad, u.Provincia
FROM contratos c
JOIN propiedades p ON c.id_propiedad = p.id_propiedad
JOIN ubicaciones u ON p.id_ubicacion = u.id_ubicacion;


;CREATE VIEW VistaPropiedadesCaracteristicas AS
SELECT p.*, cp.Caracteristica
FROM propiedades p
JOIN caracteristicas_propiedad cp ON p.id_propiedad = cp.id_propiedad;

;CREATE VIEW VistaDocumentosContratos AS
SELECT d.*, c.Fecha AS FechaContrato, c.ID_Cliente, c.ID_Propiedad
FROM documentos d
JOIN contratos c ON d.id_contrato = c.id_contrato;

;


-Vista de Empleados con Total de Contratos y Monto Total de Transacciones

CREATE VIEW VistaEmpleadosContratosTransacciones AS
SELECT e.Nombre_empleado, COUNT(c.id_contrato) AS TotalContratos, SUM(t.M
onto) AS MontoTotalTransacciones
FROM empleados e
LEFT JOIN contratos c ON e.id_empleado = c.id_cliente
LEFT JOIN transacciones t ON c.id_contrato = t.id_contrato
GROUP BY e.id_empleado;

;
-- Vista de Propiedades con Cantidad de Contratos y Ingreso Total
```

```sql
CREATE VIEW VistaPropiedadesContratosIngreso AS
SELECT p.Nombre AS Propiedad, COUNT(c.id_contrato) AS CantidadContratos,
SUM(f.Ingreso) AS IngresoTotal
FROM propiedades p
LEFT JOIN contratos c ON p.id_propiedad = c.id_propiedad
LEFT JOIN finanzas f ON p.id_propiedad = f.id_propiedad
GROUP BY p.id_propiedad;

;
-- Vista de Contratos por Ciudad y Provincia

CREATE VIEW Vista_Contratos AS
SELECT u.Ciudad, u.Provincia, COUNT(c.id_contrato) AS CantidadContratos
FROM contratos c
JOIN propiedades p ON c.id_propiedad = p.id_propiedad
JOIN ubicaciones u ON p.id_ubicacion = u.id_ubicacion
GROUP BY u.Ciudad, u.Provincia
ORDER BY CantidadContratos DESC;

;
-- Vista de Clientes con Contratos y Monto Total de Transacciones

CREATE VIEW VistaClientesContratosTransacciones AS
SELECT cl.Nombre_cliente, COUNT(c.id_contrato) AS CantidadContratos, SUM(
t.Monto) AS MontoTotalTransacciones
FROM clientes cl
LEFT JOIN contratos c ON cl.id_cliente = c.id_cliente
LEFT JOIN transacciones t ON c.id_contrato = t.id_contrato
GROUP BY cl.id_cliente;

;
-- Vista de Documentos Agrupados por Tipo

CREATE VIEW VistaDocumentosTipo AS
SELECT SUBSTRING_INDEX(d.Nombre_Doc, '_', 1) AS TipoDocumento, COUNT(d.id
_documento) AS CantidadDocumentos
FROM documentos d
GROUP BY TipoDocumento
ORDER BY CantidadDocumentos DESC;

;
```

Vistas descripción

1. **VistaContratosDetalles:**
   - **Objetivo:** Proporciona detalles completos sobre contratos, incluyendo información de propiedades y clientes asociados.
   - **Composición de Tablas:**
     - `Contratos`
     - `Propiedades`
     - `Clientes`
2. **VistaFinanzasTotales:**
   - **Objetivo:** Ofrece información sobre finanzas relacionadas con propiedades, incluyendo el nombre de la propiedad asociada.
   - **Composición de Tablas:**
     - `Finanzas`
     - `Propiedades`
3. **VistaContratosUbicacion:**
   - **Objetivo:** Muestra detalles de contratos junto con información de ubicación de las propiedades asociadas.
   - **Composición de Tablas:**
     - `Contratos`
     - `Propiedades`
     - `Ubicaciones`
4. **VistaPropiedadesCaracteristicas:**
   - **Objetivo:** Proporciona detalles de propiedades junto con sus características específicas.
   - **Composición de Tablas:**
     - `Propiedades`
     - `Caracteristicas_Propiedad`
5. **VistaDocumentosContratos:**
   - **Objetivo:** Lista documentos relacionados con contratos, incluyendo la fecha del contrato y los identificadores de cliente y propiedad asociados.
   - **Composición de Tablas:**
     - `Documentos`
     - `Contratos`
6. **VistaEmpleadosContratosTransacciones:**
   - **Objetivo:** Muestra el nombre de los empleados con la cantidad total de contratos y el monto total de transacciones.
   - **Composición de Tablas:**
     - `Empleados`
     - `Contratos`
     - `Transacciones`
7. **VistaPropiedadesContratosIngreso:**

- **Objetivo:** Proporciona detalles de propiedades junto con la cantidad de contratos y el ingreso total asociado.
- **Composición de Tablas:**
  - `Propiedades`
  - `Contratos`
  - `Finanzas`

8. **Vista_Contratos:**
   - **Objetivo:** Muestra la cantidad de contratos agrupados por ciudad y provincia.
   - **Composición de Tablas:**
     - `Contratos`
     - `Propiedades`
     - `Ubicaciones`
9. **VistaClientesContratosTransacciones:**
   - **Objetivo:** Proporciona detalles de clientes con la cantidad total de contratos y el monto total de transacciones.
   - **Composición de Tablas:**
     - `Clientes`
     - `Contratos`
     - `Transacciones`
10. **VistaDocumentosTipo:**
    - **Objetivo:** Agrupa documentos por tipo y muestra la cantidad total de documentos para cada tipo.
    - **Composición de Tablas:**
      - `Documentos`

Inserción de funciones

```
-- Esta función obtiene el precio de una propiedad según su ID.
-- Uso: SELECT ObtenerPrecioPropiedad(1);

-- Donde 1 es el ID de la propiedad.

DELIMITER //

CREATE FUNCTION ObtenerPrecioPropiedad(id_propiedad INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE precio DECIMAL(10,2);SELECT Precio INTO precio
    FROM propiedades
    WHERE id_propiedad = id_propiedad;RETURN precio;END //

DELIMITER ;
```

```
;

--
Esta función calcula la suma total de ingresos generados por contratos as
ociados a un empleado.
-- Uso: SELECT CalcularIngresosEmpleado(1);

 -- Donde 1 es el ID del empleado.DELIMITER //

CREATE FUNCTION CalcularIngresosEmpleado(id_empleado INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total_ingresos DECIMAL(10,2);SELECT SUM(f.Ingreso) INTO total
_ingresos
    FROM contratos c
    JOIN finanzas f ON c.id_contrato = f.id_contrato
    WHERE c.id_empleado = id_empleado;RETURN total_ingresos;END //

DELIMITER ;

;
```

Descripción de funciones

Funciones:

1. **ObtenerPrecioPropiedad:**
    - **Objetivo:** Esta función devuelve el precio de una propiedad dado su ID.
    - **Uso Ejemplar:** `SELECT ObtenerPrecioPropiedad(1);` (Donde 1 es el ID de la propiedad).
    - **Descripción:** La función utiliza el ID de la propiedad como parámetro de entrada y recupera el precio asociado a esa propiedad desde la tabla de propiedades. El resultado es el precio de la propiedad.
2. **CalcularIngresosEmpleado:**
    - **Objetivo:** Calcula la suma total de ingresos generados por contratos asociados a un empleado específico.
    - **Uso Ejemplar:** `SELECT CalcularIngresosEmpleado(1);` (Donde 1 es el ID del empleado).
    - **Descripción:** La función toma el ID de un empleado como parámetro y realiza una suma de los ingresos asociados a los contratos donde ese

empleado está involucrado. Utiliza las tablas `Contratos` y `Finanzas` para obtener la información necesaria.

Ambas funciones están diseñadas para proporcionar información específica y útil mediante consultas directas, facilitando la obtención del precio de una propiedad y el cálculo de los ingresos totales generados por los contratos asociados a un empleado.

Stored procedures

```
--
 Este procedimiento ordena una tabla según un campo y dirección específic
os.
--
 Uso: CALL OrdenarTabla('nombre_tabla', 'nombre_campo', 'asc' o 'desc');

DELIMITER //

CREATE PROCEDURE OrdenarTabla(
    IN tabla_nombre VARCHAR(255),
    IN campo_orden VARCHAR(255),
    IN direccion_orden VARCHAR(4)
)
BEGIN
    SET @sql = CONCAT('SELECT * FROM ', tabla_nombre, ' ORDER BY ', campo
_orden, ' ', direccion_orden);PREPARE stmt FROM @sql;EXECUTE stmt;DEALLOC
ATE PREPARE stmt;END //

DELIMITER ;

;

--
 Este procedimiento realiza operaciones de inserción o eliminación en una
 tabla.
-- Uso:
--
 Para insertar: CALL GestionarRegistros('insertar', 'nombre_tabla', valor
es);
--
 Para eliminar: CALL GestionarRegistros('eliminar', 'nombre_tabla', 'cond
icion');
```

```
DELIMITER //

CREATE PROCEDURE GestionarRegistros(
    IN operacion VARCHAR(10),
    IN tabla_nombre VARCHAR(255),
    IN valores_condicion TEXT
)
BEGIN
    IF operacion = 'insertar' THEN
        SET @sql = CONCAT('INSERT INTO ', tabla_nombre, ' VALUES ', valor
es_condicion);ELSEIF operacion = 'eliminar' THEN
        SET @sql = CONCAT('DELETE FROM ', tabla_nombre, ' WHERE ', valore
s_condicion);ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Operación no válida';END IF;PREPARE stmt FROM
 @sql;EXECUTE stmt;DEALLOCATE PREPARE stmt;END //

DELIMITER ;

;
```

Descripción de stored procedures

1. **OrdenarTabla:**
    - **Objetivo/Beneficio:** Este procedimiento permite ordenar una tabla específica según un campo y dirección determinados. Esto puede ser útil para obtener conjuntos de datos ordenados de manera ascendente o descendente.
    - **Uso Ejemplar:** `CALL OrdenarTabla('nombre_tabla', 'nombre_campo', 'asc' o 'desc');`
    - **Descripción:** Toma como parámetros el nombre de la tabla, el campo por el cual ordenar y la dirección (ascendente o descendente). Construye y ejecuta una consulta dinámica para realizar la ordenación. Interactúa principalmente con la tabla especificada en los parámetros.
2. **GestionarRegistros:**
    - **Objetivo/Beneficio:** Este procedimiento maneja operaciones de inserción o eliminación en una tabla. Proporciona flexibilidad para insertar nuevos registros o eliminar registros existentes según ciertas condiciones.
    - **Uso Ejemplar:**
        - Para insertar: `CALL GestionarRegistros('insertar', 'nombre_tabla', valores);`
        - Para eliminar: `CALL GestionarRegistros('eliminar', 'nombre_tabla', 'condicion');`

- **Descripción:** Selecciona la operación (insertar o eliminar) y construye y ejecuta una consulta dinámica en función de la operación seleccionada. Interactúa principalmente con la tabla especificada en los parámetros.

Ambos stored procedures proporcionan flexibilidad al usuario para realizar operaciones específicas en tablas sin tener que escribir consultas SQL directas. Esto puede simplificar la interacción con la base de datos y proporcionar una capa de abstracción para ciertas operaciones comunes, como ordenar tablas o gestionar registros.

Creación de tablas más triggers

```
-- Tabla de log para propiedades
CREATE TABLE IF NOT EXISTS log_propiedades (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    id_propiedad INT,
    accion VARCHAR(20),
    usuario VARCHAR(50),
    fecha DATE,
    hora TIME
);


-- Tabla de log para clientes
CREATE TABLE IF NOT EXISTS log_clientes (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT,
    accion VARCHAR(20),
    usuario VARCHAR(50),
    fecha DATE,
```

```
    hora TIME
);

-- Trigger que se ejecuta antes de una operación de inserción en la tabla
de log de propiedades.
-- Este trigger registra la información antes de la operación.

DELIMITER //

CREATE TRIGGER LogAntesOperacion BEFORE INSERT ON log_propiedades
FOR EACH ROW
BEGIN
    SET NEW.usuario = USER();  -- Captura el usuario que realiza la
operación.
    SET NEW.fecha = CURDATE();  -- Captura la fecha actual.
    SET NEW.hora = CURTIME();  -- Captura la hora actual.
END //

DELIMITER ;

;

-- Trigger que se ejecuta después de una operación de eliminación en la
tabla de log de clientes.
-- Este trigger registra la información después de la operación.

DELIMITER //

CREATE TRIGGER LogDespuesOperacion AFTER DELETE ON log_clientes
FOR EACH ROW
```

```
BEGIN
    INSERT INTO log_clientes (id_cliente, accion, usuario, fecha, hora)
    VALUES (OLD.id_cliente, 'Eliminacion', USER(), CURDATE(), CURTIME());
END //

DELIMITER ;
```

Sentencias

```
-- Usuario con Permiso de Lectura

-- Crear usuario con permiso de lectura
CREATE USER 'lector_1'@'%' IDENTIFIED BY 'L12345678';

-- Otorgar permisos de lectura sobre todas las tablas
GRANT SELECT ON basededatos.* TO 'lector_1'@'%';

-- En este caso, el usuario Lector_1 se crea con la capacidad de leer
datos (SELECT) en todas las tablas de la base de datos especificada.

-- Crear Usuario con Permiso de Lectura, Inserción y Modificación

-- Crear usuario con permisos de lectura, inserción y modificación
CREATE USER 'admin_1'@'%' IDENTIFIED BY 'A12345678';
```

```
-- Otorgar permisos de lectura, inserción y modificación sobre todas las tablas

GRANT SELECT, INSERT, UPDATE ON inmobiliaria_db.* TO 'admin_1'@'%';


-- Aquí, el usuario admin_1 se crea con permisos para leer (SELECT), insertar (INSERT) y actualizar (UPDATE) datos en todas las tablas de la base de datos especificada.
```

TCL

```
-- Se haran uso de dos tablas de las presentadas en el proyecto: Propiedades y Contratos.


-- Modificaciones en la tabla Propiedades


-- Crear un Stored Procedure (SP)

DELIMITER //



CREATE PROCEDURE ModificarTablas()

BEGIN

    -- Iniciar transacción

    START TRANSACTION;



    -- Variable para contar registros

    SET @count = (SELECT COUNT(*) FROM Propiedades);
```

```
    -- Verificar si hay registros en la tabla Propiedades
    IF @count > 0 THEN
        -- Eliminar algunos registros (o reemplazar con inserción si no
hay registros)
        DELETE FROM Propiedades WHERE ID_Propiedad IN (1, 2, 3);



        -- Sentencia para rollback (descomentar si es necesario)
        -- ROLLBACK;



        -- Sentencia para commit (comentar si se utiliza rollback)
        -- COMMIT;
    ELSE
        -- Insertar nuevos registros si no hay registros
        INSERT INTO Propiedades (Nombre, Precio, ID_Tipo_Propiedad,
ID_Ubicacion, ID_Empleado)
        VALUES
            ('Nueva Propiedad 1', 300000, 1, 1, 1),
            ('Nueva Propiedad 2', 280000, 2, 2, 2),
            ('Nueva Propiedad 3', 350000, 1, 1, 3);



        -- Sentencia para rollback (descomentar si es necesario)
        -- ROLLBACK;
```

```
        -- Sentencia para commit (comentar si se utiliza rollback)
        -- COMMIT;
    END IF;


    -- Fin del Stored Procedure
END //


DELIMITER ;


-- Modificaciones en la Tabla Contratos

-- Iniciar transacción
BEGIN;

-- Insertar ocho nuevos registros en Contratos
INSERT INTO Contratos (ID_Cliente, id_contrato, Fecha)
VALUES
    (1, null, '2023-11-01'),
    (2, 42, '2023-11-02'),
    (3, 43, '2023-11-03'),
    (4, 44, '2023-11-04'),
    (5, 45, '2023-11-05'),
    (6, 46, '2023-11-06'),
    (7, 47, '2023-11-07'),
    (8, 48, '2023-11-08');
```

```
-- Guardar un savepoint después de la inserción del registro #4
SAVEPOINT after_insert_4;

-- Insertar cuatro registros adicionales
INSERT INTO Contratos (ID_Cliente, ID_contrato, Fecha)
VALUES
    (5, 9, '2023-11-09'),
    (6, 10, '2023-11-10'),
    (7, 11, '2023-11-11'),
    (8, 12, '2023-11-12');

-- Guardar un savepoint después de la inserción del registro #8
SAVEPOINT after_insert_8;

-- Sentencia para rollback (descomentar si es necesario)
-- ROLLBACK TO after_insert_4;

-- Sentencia para commit (comentar si se utiliza rollback)
-- COMMIT;
```

Backup

```
-- Backup de datos para las siguientes tablas: Propiedades, Clientes,
          Empleados, Ubicaciones, Contratos, Transacciones,
           Caracteristicas_Propiedad, Finanzas, Documentos

-- Volcar datos de las tablas

-- Tabla Propiedades
```

```
SELECT * FROM Propiedades INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL
Server 8.0.1\\Uploads\\propiedades_backup.sql';


-- Tabla Clientes

SELECT * FROM Clientes INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL Server
8.0.1\\Uploads\\clientes_backup.sql';


-- Tabla Empleados

SELECT * FROM Empleados INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL
Server 8.0.1\\Uploads\\empleados_backup.sql';


-- Tabla Ubicaciones

SELECT * FROM Ubicaciones INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL
Server 8.0.1\\Uploads\\ubicaciones_backup.sql';


-- Tabla Contratos

SELECT * FROM Contratos INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL
Server 8.0.1\\Uploads\\contratos_backup.sql';


-- Tabla Transacciones

SELECT * FROM Transacciones INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL
Server 8.0.1\\Uploads\\transacciones_backup.sql';


-- Tabla Caracteristicas_Propiedad

SELECT * FROM Caracteristicas_Propiedad INTO OUTFILE
'C:\\ProgramData\\MySQL\\MySQL Server
8.0.1\\Uploads\\c_propiedades_backup.sql';


-- Tabla Finanzas

SELECT * FROM Finanzas INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL Server
8.0.1\\Uploads\\finanzas_backup.sql';
```

```
-- Tabla Documentos

SELECT * FROM Documentos INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL
Server 8.0.1\\Uploads\\documentos_backup.sql';
```

Resumen del trabajo realizado hasta ahora:

1. **Creación de tablas:** Se han definido tablas importantes como Propiedades, Clientes, Empleados, Ubicaciones, Contratos, Transacciones, Características_Propiedad, Finanzas y Documentos.
2. **Creación de vistas y procedimientos almacenados:** Se han creado vistas como VistaContratosDetalles, VistaEmpleadosContratosTransacciones, etc., y procedimientos almacenados como ObtenerPrecioPropiedad y CalcularIngresosEmpleado para realizar operaciones específicas en la base de datos.
3. **Modificaciones controladas por transacciones:** Se han realizado operaciones de inserción, modificación y eliminación de registros controladas por transacciones en algunas tablas.
4. **Backup de la base de datos:** Se ha solicitado un backup de la base de datos, centrándose en los datos solamente.

Líneas adicionales para profundizar en la base de datos de una inmobiliaria:

1. **Implementar un sistema de usuarios y permisos:** Crea tablas para gestionar usuarios y roles, y asigna permisos específicos a cada usuario, por ejemplo, permisos de lectura, escritura o eliminación sobre ciertas tablas.
2. **Seguimiento de visitas y citas:** Crea tablas para registrar las visitas a las propiedades por parte de los clientes, así como las citas agendadas para mostrar las propiedades.
3. **Gestión de contratos de alquiler:** Amplía el modelo de datos para incluir contratos de alquiler, junto con los términos y condiciones asociados, y desarrolla funcionalidades para gestionar estos contratos.
4. **Integración con sistemas de pago:** Si la inmobiliaria procesa pagos, integra la base de datos con sistemas de pago para registrar transacciones financieras y llevar un seguimiento del flujo de efectivo.
5. **Análisis de datos y generación de informes:** Implementa consultas complejas y vistas para realizar análisis de datos, como el rendimiento de las propiedades, la rentabilidad de los contratos, etc., y genera informes basados en estos análisis.
6. **Interfaz de usuario:** Desarrolla una interfaz de usuario (front-end) para que los usuarios puedan interactuar con la base de datos de manera intuitiva, por ejemplo, buscando propiedades, registrándose para visitas, etc.