

Laboratorio AutoRental

ALEJANDRA MACHUCA MOLINA

DOCENTE

PEDRO FELIPE GÓMEZ BONILLA

**CAMPUSLANDS
SANDBOX GRUPO T2
RUTA MYSQL
TIBÚ
2024**

Tabla de Contenidos

Introducción	3
Caso de Estudio	4
Planificación	4
Construcción del Modelo Conceptual	5
Descripción	5
Relaciones y cardinalidad	5
Gráfica	7
Construcción del Modelo Lógico	7
Descripción	7
Relaciones y Cardinalidad	8
Gráfica	9
Normalización del Modelo Lógico	10
Primera Forma Normal (1FN)	10
Descripción	10
Segunda Forma Normal (2FN)	10
Descripción	10
Tercera Forma Normal (3FN)	10
Descripción	10
Construcción del Modelo Físico	11
Descripción	11
Tablas	11
Construcción del Diagrama UML	14
Descripción	14
Gráfica	17
Inserciones de Datos	18

Introducción

Este proyecto busca crear una base de datos para el Laboratorio "AutoRenta", que permite gestionar de forma ordenada la información de sucursales, empleados, clientes, vehículos, alquileres, descuentos y tipos de vehículos. Con esta base de datos, el laboratorio podrá llevar un mejor control de su inventario y sus transacciones, optimizando sus procesos.

Este documento explica el diseño y desarrollo de la base de datos, desde el modelo conceptual hasta el físico. Además, se normaliza el modelo para evitar duplicaciones y asegurar la integridad de la información.

Caso de Estudio

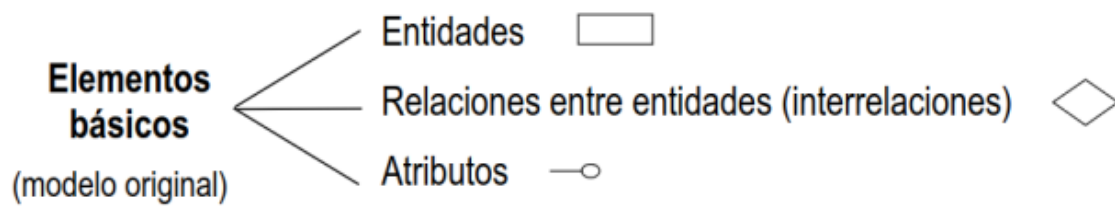
La empresa donde usted trabaja ha sido contratada para desarrollar un sistema de información para una empresa de alquiler de vehículos llamada AutoRental, y usted ha sido designado para diseñar una base de datos para ese sistema de información.

AutoRental cuenta con 5 sucursales en diferentes ciudades y se proyecta a expandirse a otras ciudades del país y cuenta con una flota propia de vehículos de diferentes tipos, modelos (año), capacidad, etc. Los clientes de AutoRental podrán alquilar un vehículo en una sucursal y entregarlo en otra sucursal. AutoRental ofrece descuentos sobre diferentes tipos de vehículos a lo largo del año. Los valores de alquiler dependen del tipo de vehículo (sedán, compacto, camioneta platón, camioneta lujo, deportivo, etc) y se cobran por días y/o semanas. Por ejemplo, si alquila un vehículo por 9 días, el valor cotizado será de 1 semana y 2 días. Si un cliente entrega el vehículo pasada la fecha de entrega contratada, se cobrarán los días adicionales con un incremento del 8%.

Planificación

1. **Identificar Necesidades:** Primero, definimos qué información necesita el laboratorio para gestionar sucursales, empleados, clientes, vehículos, alquileres, tipo de vehículos y descuentos.
2. **Diseño Conceptual:** Luego, armamos un esquema básico con las entidades principales (como Sucursales, Empleados, Clientes, Vehículos, Alquileres, Tipo_vehículos y Descuentos, etc.) y sus relaciones. Aquí creamos un diagrama E-R para visualizar cómo se conecta todo.
3. **Diseño Lógico:** Convertimos el esquema en un diseño más detallado, definiendo las llaves primarias y foráneas para asegurar que todo esté bien relacionado.
4. **Normalización:** Aplicamos las reglas de normalización (1FN, 2FN y 3FN) para evitar datos repetidos y hacer que la base de datos sea más eficiente.
5. **Diseño Físico:** Pasamos el diseño a un SGBD como MySQL, definiendo las tablas, columnas y relaciones que se usarán en la base de datos final.

Construcción del Modelo Conceptual



Las entidades representan los objetos de interés para el sistema, estas contienen atributos que describen sus características. Las relaciones representan las asociaciones entre las entidades (si aplica o si existe).

Descripción

- **Sucursales:** id_sucursal, ubicacion, telefono_fijo, celular y correo_electronico.
- **Empleados:** Id_empleado, cedula, nombres, apellidos, ubicacion, celular, correo_electronico e id_sucursal
- **Clientes:** Id_cliente, cedula, nombres, apellidos, ubicacion, celular, correo_electronico
- **Vehiculos:** id_vehiculo, placa, referencia, modelo, puertas, capacidad, sunroof, motor, color, id_tipoV
- **Alquileres:** id_alquiler, fecha_salida, fecha_llegada, fecha_esperada_llegada, valor_cotizado, valor_pagado
- **Tipo_vehiculo:** id_tipoV, valor_alquiler_semana, valor_alquiler_dia, tipo
- **Descuentos:** id_descuento, fecha_inicio, fecha_fin, porcentaje_descuento, id_tipoV

Relaciones y cardinalidad

1. Sucursales - Empleados

- **Relación:** "tiene" Una sucursal puede tener muchos empleados
- **Cardinalidad:** 1:N

2. Empleados - Alquileres

- **Relación:** "Tiene" Un empleado tiene muchos alquileres
- **Cardinalidad:** 1:N

3. Vehículos - Alquileres

- **Relación:** "Tiene" Un vehículo puede tener muchos alquileres
- **Cardinalidad:** 1:N

4. Clientes - Alquileres

- **Relación:** "Tiene" Un cliente puede tener muchos alquileres
- **Cardinalidad:** 1:N

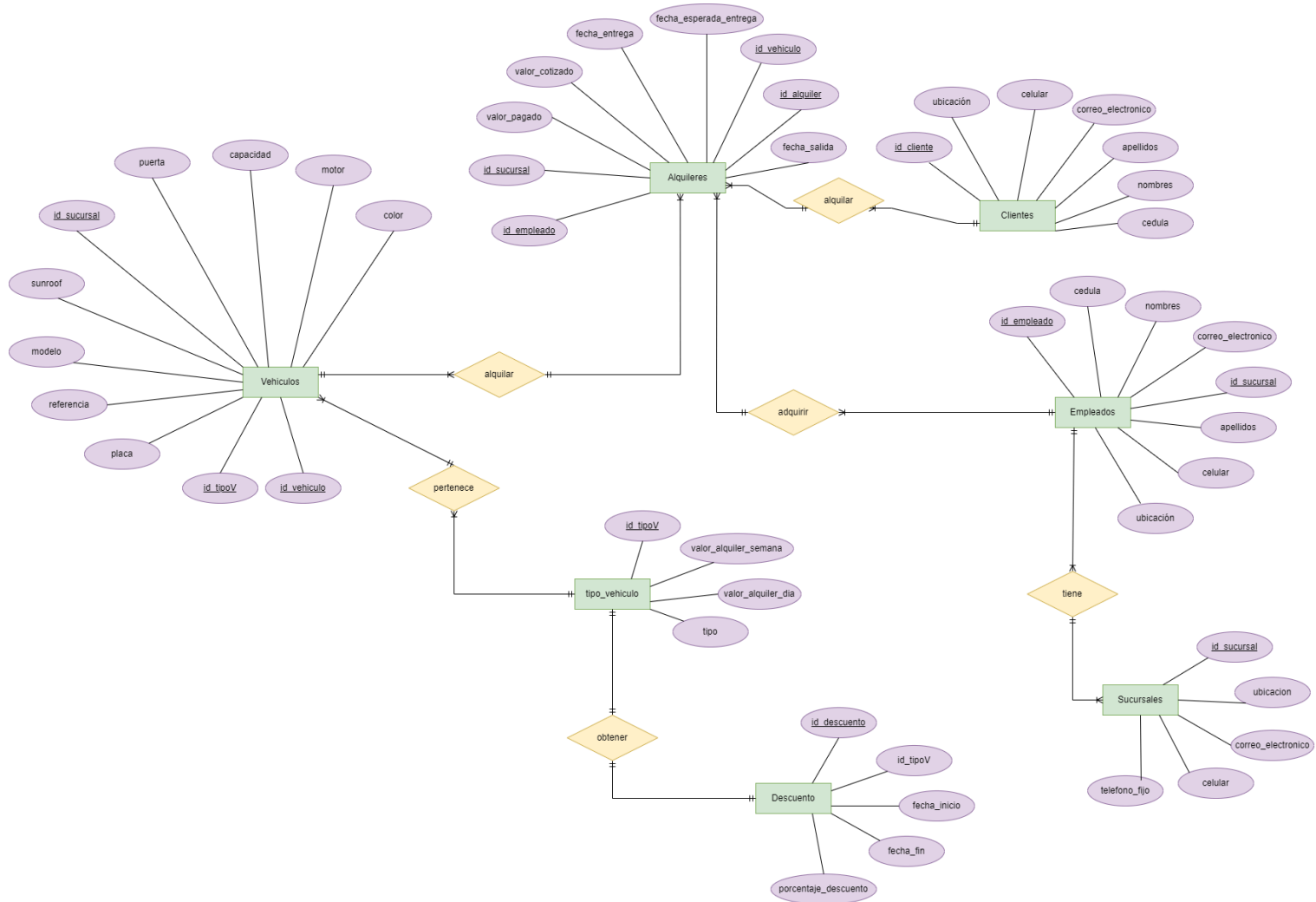
5. Tipo_vehiculo - Vehiculos

- **Relación:** "Tiene" Un tipo_vehiculo puede tener muchos Vehiculos
- **Cardinalidad:** 1:N

6. Tipo_vehiculo - Descuento

- **Relación:** "Tiene" Un tipo_vehiculo puede tener un Descuento
- **Cardinalidad:** 1:1

Gráfica



Construcción del Modelo Lógico

Contiene distintas entidades dadas en el caso de estudio. Estas entidades tienen atributos que representan sus características.

Descripción

1. Sucursales:

- Su llave principal es id_sucursal
- Sus atributos son: ubicacion, telefono_fijo, celular y correo_electronico

2. Empleados:

- Su llave principal es Id_empleado
- Sus atributos son: cedula, nombres, apellidos, ubicacion, celular, correo_electronico
- Su llave foránea es id_sucursal

3. Clientes:

- Su llave principal es Id_cliente
- Sus atributos son: cedula, nombres, apellidos, ubicacion, celular, correo_electronico

4. Vehiculos:

- Su llave principal es id_vehiculo
- Su llave foránea es id_tipoV
- Sus atributos son: placa, referencia, modelo, puertas, capacidad, sunroof, motor, color

5. Alquileres:

- Su llave principal es id_alquiler
- Sus atributos son: fecha_salida, fecha_llegada, fecha_esperada_llegada, valor_cotizado, valor_pagado

6. Tipo_vehiculo:

- Su llave principal es id_tipoV
- Sus atributos son: valor_alquiler_semana, valor_alquiler_dia, tipo

7. Descuentos:

- Su llave principal es id_descuento
- Sus atributos son: fecha_inicio, fecha_fin, porcentaje_descuento
- Su llave foránea es id_tipoV

Relaciones y Cardinalidad

1. Sucursales - Empleados

- Una sucursal puede tener muchos empleados



2. Empleados - Alquileres

- Un empleado tiene muchos alquileres



3. Vehiculos - Alquileres

- Un vehiculo puede tener muchos alquileres



4. Clientes - Alquileres

- Un cliente puede tener muchos alquileres



5. Tipo_vehiculo - Vehiculos

- Un Tipo_vehiculo puede tener muchos Vehiculos

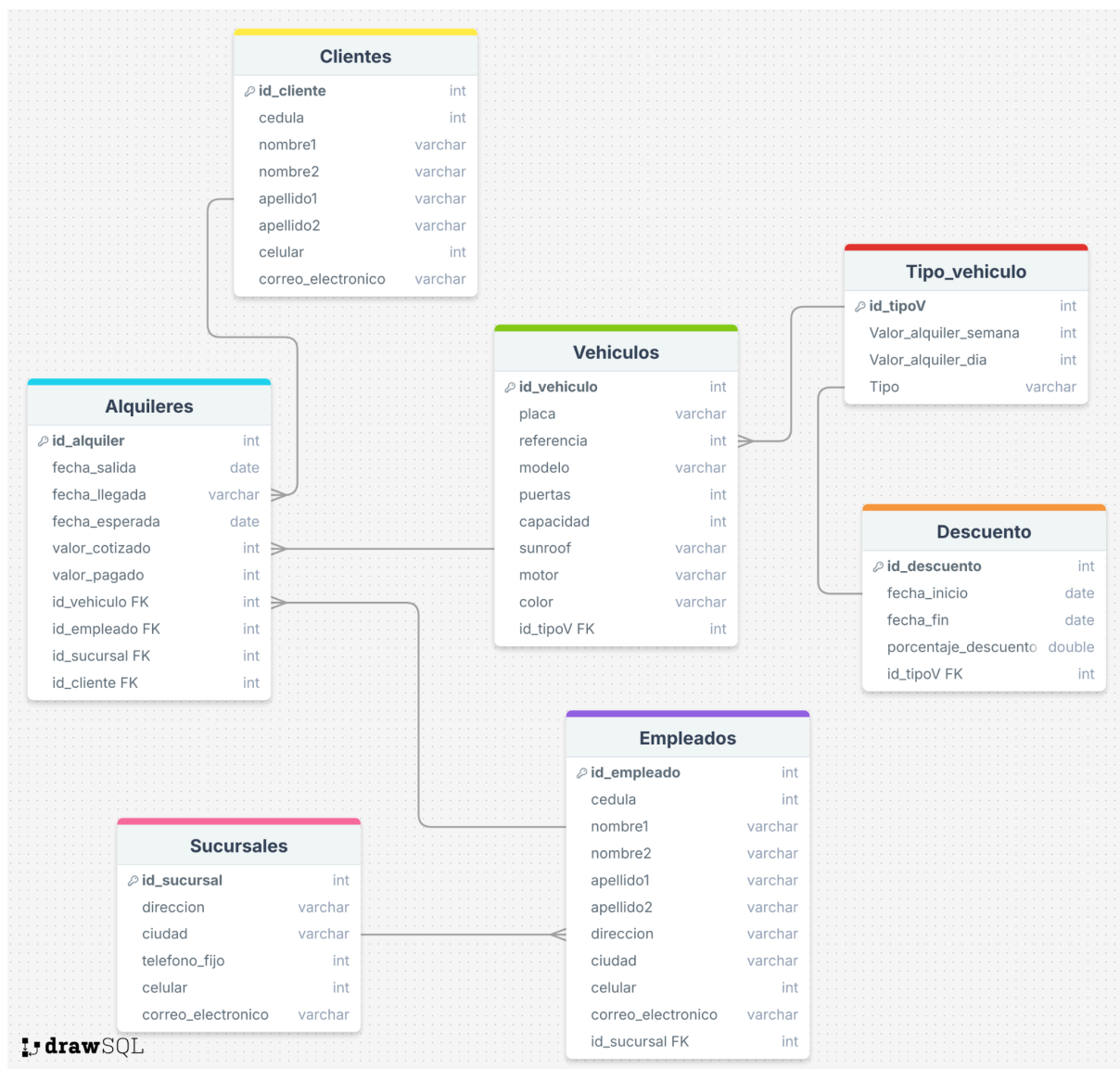


6. Tipo_vehiculo - Descuento

- Un Tipo_Vehiculo puede tener un descuento



Gráfica



Normalización del Modelo Lógico

La normalización es el proceso de organizar las tablas y relaciones en la base de datos para evitar datos repetidos y asegurar la integridad. Esto se logra aplicando tres formas normales (1FN, 2FN y 3FN), cada una con reglas que hacen que la base de datos sea más eficiente y fácil de manejar.

Primera Forma Normal (1FN)

Se establece que los atributos deben ser atómicos, es decir que el número de valores que puede tomar un atributo debe limitarse a uno. Este se basa en las características del modelo relacional. Algunas de estas son que cada fila de la misma tabla debe ser única y que debe prevalecer un crecimiento vertical de los datos y no horizontal.

Descripción

La primera forma normal es el primer nivel de normalización en base de datos donde se le aplicará a las tablas de las bases de datos para garantizar una mejor organización donde se evita redundancias y asegurará la consistencia de la información.

Segunda Forma Normal (2FN)

Se debe tener una clave clave principal, de preferencia simple. Cada atributo de la tabla debe depender totalmente del atributo clave. Este se basa en el concepto de dependencias funcionales.

Descripción

La segunda forma normal, es el segundo nivel de normalización en el diseño de la base de datos que se aplicará a las tablas de una base de datos que ya cumplen con la primera forma normal y lleva a cabo la eliminación de dependencias parciales dentro de una tabla.

Tercera Forma Normal (3FN)

Se basa en el concepto de dependencias transitivas. Para esto, debe estar primero en 2FN y no deben existir atributos no principales que dependan transitivamente del atributo clave.

Descripción

La tercera forma normal, es el tercer nivel de normalización en el diseño de la base de datos que se aplicará a las tablas de una base de datos que ya cumplen con la segunda forma normal y se enfoca en la eliminación de dependencias transitivas, evitando que un atributo no clave dependa de otro no clave.

Construcción del Modelo Físico

Ya esta es la última etapa del proceso, se basa en el modelo lógico y se utiliza para implementar dicho modelo en una base de datos real. Este modelo consta de tablas, columnas y relaciones.

Descripción

El modelo físico se diseñó para funcionar en MySQL, donde cada entidad se representa como una tabla compuesta por sus atributos correspondientes, organizados en columnas con tipos de datos específicos según sea necesario.

Tablas

Para crear la base de datos utilice el siguiente comando:

```
CREATE DATABASE Laboratorio_T2;
```

Para utilizar la base de datos ocupe el siguiente comando:

```
USE Laboratorio_T2;
```

Comenzaremos creando las tablas junto con sus tipos de datos correspondientes. Para esto, utilicé los siguientes comandos:

1. Creación de tabla Sucursales

```
CREATE TABLE Sucursales (  
  id_sucursal INT PRIMARY KEY,  
  direccion VARCHAR(50),  
  ciudad VARCHAR(50),  
  telefono_fijo INT,  
  celular INT,  
  correo_electronico VARCHAR(50)  
);
```

2. Creación de tabla Empleados

```
CREATE TABLE Empleados (  
  id_empleado INT PRIMARY KEY,  
  cedula INT,  
  nombre1 VARCHAR(60),  
  nombre2 VARCHAR(60),  
  apellido1 VARCHAR(60),  
  apellido2 VARCHAR(60),  
  direccion VARCHAR(50),  
  ciudad VARCHAR(50),  
  celular INT,  
  correo_electronico VARCHAR(50),  
  id_sucursal INT,  
  FOREIGN KEY (id_sucursal) REFERENCES Sucursales(id_sucursal)  
);
```

3. Creación de tabla Clientes

```
CREATE TABLE Clientes (  
  id_cliente INT PRIMARY KEY,  
  cedula INT,  
  nombre1 VARCHAR(60),  
  nombre2 VARCHAR(60),  
  apellido1 VARCHAR(60),  
  apellido2 VARCHAR(60),  
  celular INT,  
  correo_electronico VARCHAR(50)  
);
```

4. Creación de tabla Tipo_vehiculo

```
CREATE TABLE Tipo_vehiculo (  
  id_tipoV INT PRIMARY KEY,  
  valor_alquiler_semana INT,  
  valor_alquiler_dia INT,  
  tipo VARCHAR(55)  
);
```

5. Creación de tabla Vehiculos

```
CREATE TABLE Vehiculos (  
  id_vehiculo INT PRIMARY KEY,  
  placa VARCHAR(50),  
  referencia INT,  
  modelo VARCHAR(50),  
  puertas INT,  
  capacidad INT,  
  sunroof VARCHAR(30),  
  motor VARCHAR(50),  
  color VARCHAR(40),  
  id_tipoV INT,  
  FOREIGN KEY (id_tipoV) REFERENCES Tipo_vehiculo(id_tipoV)  
);
```

6. Creación de tabla Alquileres

```
CREATE TABLE Alquileres (  
  id_alquiler INT PRIMARY KEY,  
  fecha_salida DATE,  
  fecha_llegada VARCHAR(55) NULL,  
  fecha_esperada DATE,  
  valor_cotizado INT,  
  valor_pagado INT,  
  id_vehiculo INT,  
  id_cliente INT,  
  id_empleado INT,  
  id_sucursal INT,  
  FOREIGN KEY (id_vehiculo) REFERENCES Vehiculos(id_vehiculo),  
  FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente),  
  FOREIGN KEY (id_empleado) REFERENCES Empleados(id_empleado),  
  FOREIGN KEY (id_sucursal) REFERENCES Sucursales(id_sucursal)  
);
```

7. Creación de tabla Descuentos

```
CREATE TABLE Descuentos (  
  id_descuento INT PRIMARY KEY,  
  fecha_inicio DATE,  
  fecha_fin DATE,  
  porcentaje_descuento INT,  
  id_tipoV INT,  
  FOREIGN KEY (id_tipoV) REFERENCES Tipo_vehiculo(id_tipoV)  
)
```

Construcción del Diagrama UML

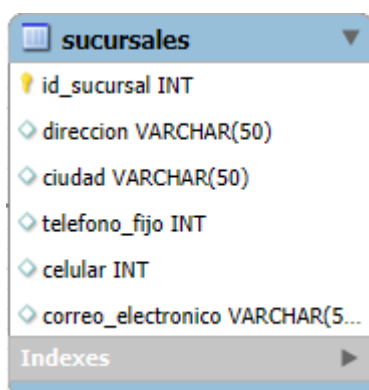
Se ha diseñado un diagrama UML tomando como referencia la normalización para entender mejor los diseños, la arquitectura del código y la implementación propuesta. Este enfoque nos permitirá tener una visión clara y detallada de cómo se manejan cada una de las consultas, funcionalidades y los usuarios en la base de datos.

Descripción

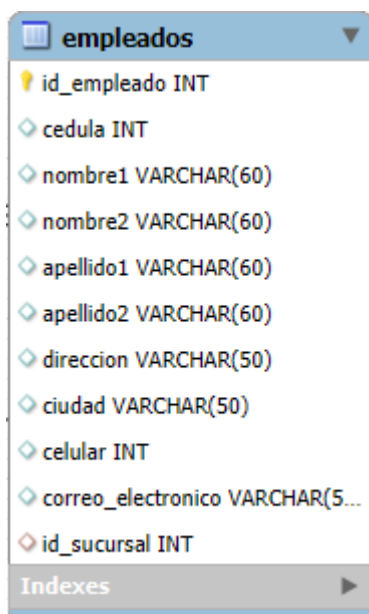
El diagrama UML se ha diseñado con el objetivo de representar detalladamente la estructura de cada tabla y sus relaciones. Este diagrama ilustra claramente el tipo de dato correspondiente a cada atributo, así como la identificación de claves primarias (primary keys) y claves foráneas (foreign keys).

Comenzaremos creando las tablas junto con sus tipos de datos correspondientes:

1. Tabla Sucursales



2. Tabla Empleados



3. Tabla Clientes

clientes	
id_cliente	INT
cedula	INT
nombre1	VARCHAR(60)
nombre2	VARCHAR(60)
apellido1	VARCHAR(60)
apellido2	VARCHAR(60)
celular	INT
correo_electronico	VARCHAR(5...
Indexes	

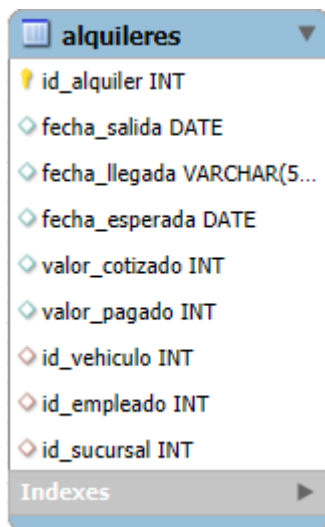
4. Tabla Tipo_vehiculo

tipo_vehiculo	
id_tipoV	INT
valor_alquiler_semana	INT
valor_alquiler_dia	INT
tipo	VARCHAR(55)
Indexes	

5. Tabla Vehiculos

vehiculos	
id_vehiculo	INT
placa	VARCHAR(50)
referencia	INT
modelo	VARCHAR(50)
puertas	INT
capacidad	INT
sunroof	VARCHAR(3...
motor	VARCHAR(50)
color	VARCHAR(40)
id_tipoV	INT
Indexes	

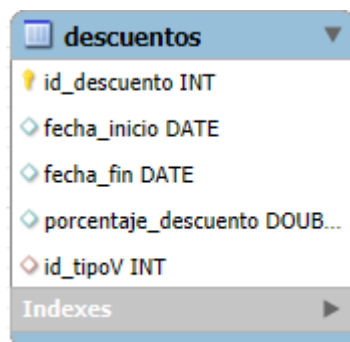
6. Tabla Alquileres



The screenshot shows the 'alquileres' table structure. It lists the following fields: 'id_alquiler' (INT, primary key), 'fecha_salida' (DATE), 'fecha_llegada' (VARCHAR(5...)), 'fecha_esperada' (DATE), 'valor_cotizado' (INT), 'valor_pagado' (INT), 'id_vehiculo' (INT), 'id_empleado' (INT), and 'id_sucursal' (INT). There is an 'Indexes' tab at the bottom.

Field	Type	Key
id_alquiler	INT	Primary Key
fecha_salida	DATE	
fecha_llegada	VARCHAR(5...)	
fecha_esperada	DATE	
valor_cotizado	INT	
valor_pagado	INT	
id_vehiculo	INT	
id_empleado	INT	
id_sucursal	INT	

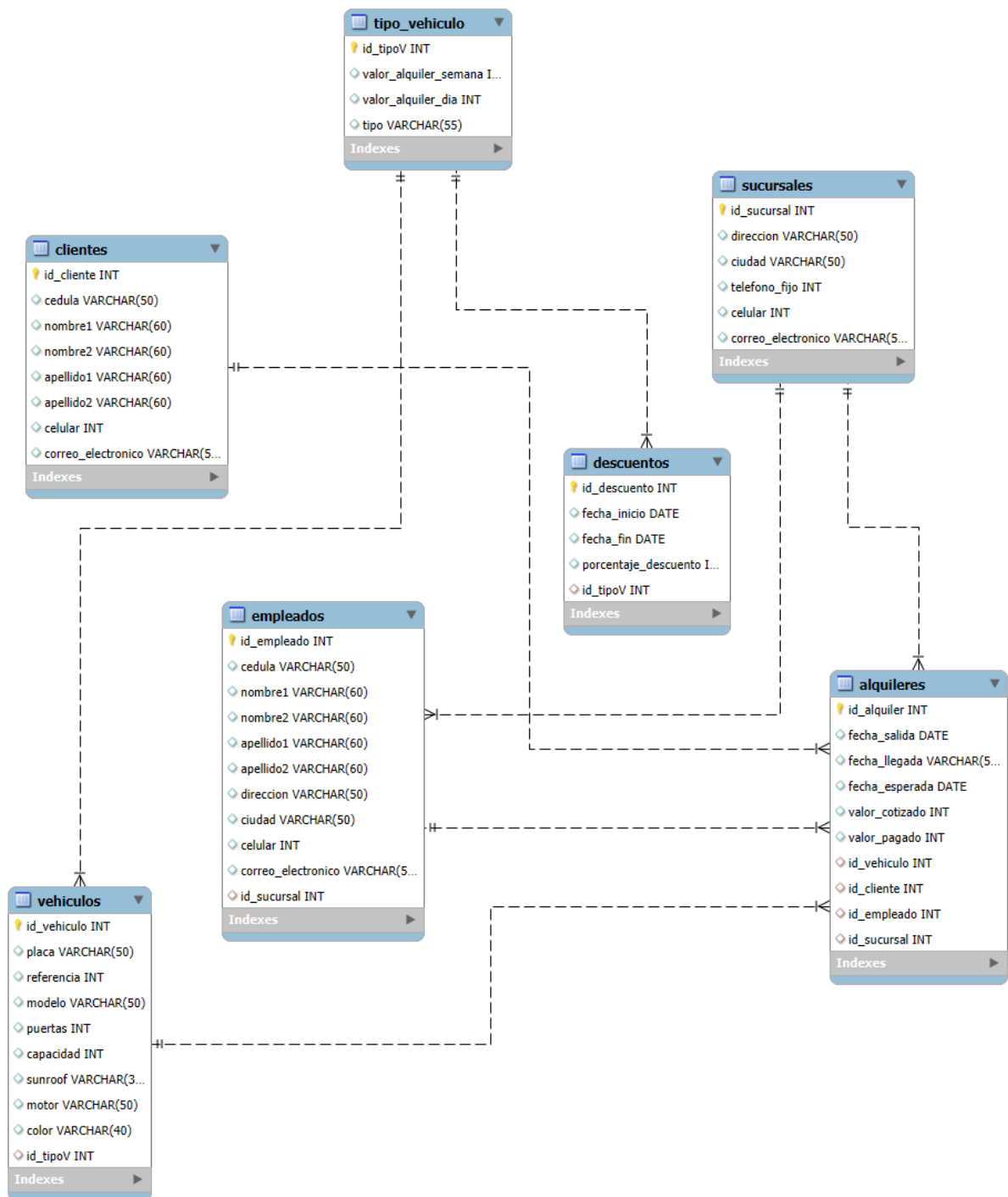
7. Tabla Descuentos



The screenshot shows the 'descuentos' table structure. It lists the following fields: 'id_descuento' (INT, primary key), 'fecha_inicio' (DATE), 'fecha_fin' (DATE), 'porcentaje_descuento' (DOUB...), and 'id_tipoV' (INT). There is an 'Indexes' tab at the bottom.

Field	Type	Key
id_descuento	INT	Primary Key
fecha_inicio	DATE	
fecha_fin	DATE	
porcentaje_descuento	DOUB...	
id_tipoV	INT	

Gráfica



Inserciones de Datos

La inserción de datos en las tablas es una parte fundamental para la prueba de operatividad.

Para insertar datos en una tabla específica, se utiliza la siguiente sintaxis:

- Empezamos colocando 'INSERT INTO' para especificar que le queremos añadir un valor a la tabla, siguiente a esto ponemos el nombre de la tabla a la cual le queremos ingresar los datos y por último le agregamos los valores que están creados en los anteriores modelos utilizando 'VALUES' y asegurándonos de que cumplan con los requisitos especificados.
- Para la mayoría de las tablas creadas se hizo una inserción de 100 datos, a excepción de las tablas "Sucursales" que tiene solo 5 inserciones y la tabla "Tipo_vehiculo" que tiene 10 inserciones. En este documento se mostrará solo 10 ejemplos de las tablas con 100 inserciones.

1. Tabla Sucursales

```
INSERT INTO Sucursales (id_sucursal, direccion, ciudad, telefono_fijo, celular, correo_electronico) VALUES
(1, 'Av. Siempre Viva 123', 'Ciudad A', 123456789, 987654321, 'sucursal1@empresa.com'),
(2, 'Calle Falsa 456', 'Ciudad B', 234567890, 876543210, 'sucursal2@empresa.com'),
(3, 'Av. de la Paz 789', 'Ciudad C', 345678901, 765432109, 'sucursal3@empresa.com'),
(4, 'Calle Principal 101', 'Ciudad D', 456789012, 654321098, 'sucursal4@empresa.com'),
(5, 'Paseo del Sol 202', 'Ciudad E', 567890123, 543210987, 'sucursal5@empresa.com');
```

2. Tabla Empleados

```
INSERT INTO Empleados (id_empleado, cedula, nombre1, nombre2, apellido1, apellido2, direccion, ciudad, celular, correo_electronico, id_sucursal) VALUES
(1, 1023456789, 'Juan', 'Carlos', 'Pérez', 'Gómez', 'Calle Libertad 123', 'Ciudad A', 987654321, 'juan.perez@empresa.com', 1),
(2, 2034567890, 'Ana', 'María', 'López', 'Rodríguez', 'Av. Siempre Viva 456', 'Ciudad B', 876543210, 'ana.lopez@empresa.com', 2),
(3, 3045678901, 'Carlos', 'Eduardo', 'Martínez', 'Sánchez', 'Calle Principal 789', 'Ciudad C', 765432109, 'carlos.martinez@empresa.com', 3),
(4, 4056789012, 'Lucía', 'Fernanda', 'González', 'Torres', 'Calle del Sol 101', 'Ciudad D', 654321098, 'lucia.gonzalez@empresa.com', 4),
(5, 5067890123, 'José', 'Luis', 'Hernández', 'Vega', 'Calle Falsa 202', 'Ciudad E', 543210987, 'jose.hernandez@empresa.com', 1),
(6, 6078901234, 'Marta', 'Elena', 'Ramírez', 'García', 'Paseo del Sol 303', 'Ciudad F', 432109876, 'marta.ramirez@empresa.com', 2),
```

(7, 7089012345, 'Raúl', 'Antonio', 'Díaz', 'Mendoza', 'Boulevard Sur 404', 'Ciudad G', 321098765, 'raul.diaz@empresa.com', 3),
 (8, 8090123456, 'Sofía', 'Isabel', 'Jiménez', 'López', 'Av. Los Andes 505', 'Ciudad H', 210987654, 'sofia.jimenez@empresa.com', 4),
 (9, 9101234567, 'David', 'Alejandro', 'Córdoba', 'Serrano', 'Calle Luna 606', 'Ciudad I', 109876543, 'david.cordoba@empresa.com', 1),
 (10, 1112345678, 'Patricia', 'Ángela', 'Castro', 'Ruiz', 'Avenida Libertad 707', 'Ciudad J', 987654321, 'patricia.castro@empresa.com', 2)

3. Tabla Clientes

```
INSERT INTO Clientes (id_cliente, cedula, nombre1, nombre2, apellido1, apellido2, celular, correo_electronico) VALUES
(1, '1234567890', 'Juan', 'Carlos', 'Pérez', 'Rodríguez', 987654321, 'juan.perez@cliente.com'),
(2, '2345678901', 'Ana', 'María', 'González', 'Sánchez', 876543210, 'ana.gonzalez@cliente.com'),
(3, '3456789012', 'Luis', 'Fernando', 'Martínez', 'López', 765432109, 'luis.martinez@cliente.com'),
(4, '4567890123', 'María', 'José', 'López', 'Gutiérrez', 654321098, 'maria.lopez@cliente.com'),
(5, '5678901234', 'Carlos', 'Alberto', 'Fernández', 'Pérez', 543210987, 'carlos.fernandez@cliente.com'),
(6, '6789012345', 'Patricia', 'Isabel', 'Hernández', 'Rodríguez', 432109876, 'patricia.hernandez@cliente.com'),
(7, '7890123456', 'David', 'Javier', 'Gómez', 'Sánchez', 321098765, 'david.gomez@cliente.com'),
(8, '8901234567', 'Carmen', 'Lucía', 'Romero', 'Vega', 210987654, 'carmen.romero@cliente.com'),
(9, '9012345678', 'José', 'Antonio', 'Jiménez', 'Molina', 109876543, 'jose.jimenez@cliente.com'),
(10, '1123456789', 'Eva', 'María', 'Serrano', 'Martínez', 987654321, 'eva.serrano@cliente.com'),
```

4. Tabla Tipo_vehiculo

```
INSERT INTO Tipo_vehiculo (id_tipoV, valor_alquiler_semana, valor_alquiler_dia, tipo)
VALUES
(1, 700, 120, 'Sedán'),
(2, 900, 150, 'SUV'),
(3, 1100, 180, 'Camioneta'),
(4, 500, 80, 'Compacto'),
(5, 600, 100, 'Hatchback'),
(6, 1500, 250, 'Deportivo'),
(7, 1800, 300, 'Convertible'),
(8, 650, 110, 'Minivan'),
(9, 750, 130, '4x4'),
(10, 850, 140, 'Coupé');
```

5. Tabla Vehiculos

INSERT INTO Vehiculos (id_vehiculo, placa, referencia, modelo, puertas, capacidad, sunroof, motor, color, id_tipoV) VALUES

(1, 'ABC123', 1001, 'Toyota Corolla', 4, 5, 'No', '1.8L', 'Blanco', 1),
(2, 'DEF456', 1002, 'Honda Civic', 4, 5, 'Sí', '2.0L', 'Negro', 2),
(3, 'GHI789', 1003, 'Mazda 3', 4, 5, 'No', '1.6L', 'Rojo', 3),
(4, 'JKL012', 1004, 'Ford Focus', 4, 5, 'Sí', '2.0L', 'Azul', 4),
(5, 'MNO345', 1005, 'Chevrolet Spark', 5, 4, 'No', '1.2L', 'Verde', 5),
(6, 'PQR678', 1006, 'Hyundai Elantra', 4, 5, 'Sí', '1.8L', 'Gris', 6),
(7, 'STU901', 1007, 'Nissan Sentra', 4, 5, 'No', '1.6L', 'Blanco', 7),
(8, 'VWX234', 1008, 'Volkswagen Jetta', 4, 5, 'Sí', '2.5L', 'Negro', 8),
(9, 'YZA567', 1009, 'Kia Forte', 4, 5, 'No', '1.6L', 'Rojo', 9),
(10, 'BCD890', 1010, 'Renault Logan', 4, 5, 'Sí', '1.6L', 'Azul', 10),

6. Tabla Alquileres

INSERT INTO Alquileres (id_alquiler, fecha_salida, fecha_llegada, fecha_esperada, valor_cotizado, valor_pagado, id_vehiculo, id_cliente, id_empleado, id_sucursal) VALUES

(1, '2024-01-10', '2024-01-15', '2024-01-14', 700, 700, 1, 1, 1, 1),
(2, '2024-02-12', '2024-02-18', '2024-02-17', 900, 900, 2, 2, 2, 2),
(3, '2024-03-05', '2024-03-07', '2024-03-06', 1100, 1100, 3, 3, 3, 3),
(4, '2024-04-01', '2024-04-08', '2024-04-07', 500, 500, 4, 4, 4, 4),
(5, '2024-05-10', '2024-05-12', '2024-05-11', 600, 600, 5, 5, 5, 1),
(6, '2024-06-15', '2024-06-20', '2024-06-19', 1500, 1500, 6, 6, 6, 2),
(7, '2024-07-01', '2024-07-05', '2024-07-04', 1800, 1800, 7, 7, 7, 3),
(8, '2024-08-10', '2024-08-12', '2024-08-11', 650, 650, 8, 8, 8, 4),
(9, '2024-09-01', '2024-09-06', '2024-09-05', 750, 750, 9, 9, 9, 1),
(10, '2024-10-12', '2024-10-16', '2024-10-15', 850, 850, 10, 10, 10, 2)

7. Tabla Descuentos

INSERT INTO Descuentos (id_descuento, fecha_inicio, fecha_fin, porcentaje_descuento, id_tipoV) VALUES

(1, '2024-01-01', '2024-01-15', 10, 1),
(2, '2024-02-01', '2024-02-15', 15, 2),
(3, '2024-03-01', '2024-03-15', 20, 3),
(4, '2024-04-01', '2024-04-15', 10, 4),
(5, '2024-05-01', '2024-05-15', 25, 5),
(6, '2024-06-01', '2024-06-15', 30, 6),
(7, '2024-07-01', '2024-07-15', 5, 7),
(8, '2024-08-01', '2024-08-15', 12, 8),
(9, '2024-09-01', '2024-09-15', 18, 9),
(10, '2024-10-01', '2024-10-15', 22, 10)

Consultas de Datos

Como se puede observar en las imágenes, cada consulta tiene especificado cuál es la función que cumple cada una. Para poderlas revisar bien se puede abrir el archivo sql para mejor visualización. Con estas consultas se nos facilitará la investigación y entendimiento de las tablas.

1. Primero podremos observar las consultas realizadas de manera sencilla
 - `SELECT direccion, ciudad, celular FROM Sucursales;`-- 1. Listar todas las sucursales con sus direcciones, ciudad y teléfonos.
 - `SELECT DISTINCT apellido1 FROM Clientes;`-- 2. Lista el primer apellido de los clientes eliminando los apellidos que estén repetidos.
 - `SELECT CONCAT_WS(' ', nombre1, ' ', nombre2, ' ', apellido1, ' ', apellido2) AS NOMBRES FROM Clientes;` -- 3. Lista el nombre y apellidos de los clientes en una única columna.
 - `SELECT * FROM Sucursales;`-- 4. Listar todas las sucursales disponibles
 - `SELECT Clientes.id_cliente, Clientes.nombre1, Clientes.apellido1 FROM Clientes JOIN Alquileres ON Clientes.id_cliente = Alquileres.id_cliente;`-- 5. Mostrar clientes con al menos un alquiler registrado
 - `SELECT * FROM Empleados WHERE id_sucursal = 8;`-- 6. Listar los empleados que trabajan en una sucursal específica
 - `SELECT V.id_vehiculo, V.placa, T.tipo FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV WHERE V.id_vehiculo NOT IN (SELECT id_vehiculo FROM Alquileres WHERE fecha_llegada IS NULL);` -- 7. Listar vehículos disponibles para alquiler (no rentados actualmente)
 - `SELECT V.placa, V.modelo, V.motor FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV WHERE T.tipo = 'Sedán';` -- 8. Listar vehículos por tipo
 - `SELECT A.id_alquiler, C.nombre1, C.apellido1, A.fecha_esperada FROM Alquileres A JOIN Clientes C ON A.id_cliente = C.id_cliente WHERE A.fecha_llegada > A.fecha_esperada;` -- 9. Mostrar alquileres que se encuentran retrasados
 - `SELECT DISTINCT tipo, valor_alquiler_semana, valor_alquiler_dia FROM Tipo_vehiculo;` -- 10. Obtener el valor de alquiler de los vehículos por tipo (semana o día)
 - `SELECT SUM(Alquileres.valor_pagado) AS Total_ingresos FROM Alquileres WHERE Alquileres.fecha_salida BETWEEN '2024-01-10' AND '2024-01-14';` -- 11. Obtener el total de ingresos generados por alquileres en un periodo específico

- `SELECT Vehiculos.placa, Vehiculos.modelo, T.tipo, T.valor_alquiler_dia, T.valor_alquiler_semana FROM Vehiculos JOIN Tipo_vehiculo T ON Vehiculos.id_tipoV = T.id_tipoV; -- 12. Listar los vehículos con su tipo, placa y el valor de alquiler correspondiente`
- `SELECT V.placa, V.referencia, V.modelo, V.puertas, V.capacidad, V.sunroof, V.motor, V.color, T.tipo FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV; -- 13. Listar los vehículos disponibles para alquiler, con sus características`
- `SELECT V.modelo, A.valor_cotizado, A.valor_pagado from Alquileres A JOIN Vehiculos V ON A.id_vehiculo = V.id_vehiculo; -- 14. Devuelve una lista con el modelo`
- `SELECT V.modelo from Alquileres A JOIN Vehiculos V ON A.id_vehiculo = V.id_vehiculo WHERE fecha_llegada > fecha_esperada; -- 15. Devuelve una lista con el modelo de los vehiculos donde su fecha de llegada es mayor a su fecha esperada de llegada`
- `SELECT C.id_cliente, C.nombre1, C.apellido1 FROM Clientes C LEFT JOIN Alquileres A ON C.id_cliente = A.id_cliente WHERE A.id_cliente IS NULL; -- 16. Obtener los clientes que no han realizado ningún alquiler`
- `SELECT C.id_cliente, C.nombre1, C.apellido1, COUNT(A.id_alquiler) AS cantidad_alquileres FROM Clientes C LEFT JOIN Alquileres A ON C.id_cliente = A.id_cliente GROUP BY C.id_cliente; -- 17. Obtener la cantidad de alquileres realizados por cada cliente`
- `SELECT V.placa, V.modelo, T.tipo, A.id_alquiler FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV JOIN Alquileres A ON V.id_vehiculo = A.id_vehiculo WHERE A.fecha_llegada IS NULL; -- 18. Listar los vehículos con su tipo y los alquileres activos (sin fecha de llegada)`
- `SELECT S.direccion, S.ciudad, E.nombre1, E.apellido1 FROM Sucursales S JOIN Empleados E ON S.id_sucursal = E.id_sucursal; -- 19. Listar las sucursales y sus empleados`
- `SELECT V.placa, V.modelo, T.valor_alquiler_semana FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV WHERE T.tipo = 'SUV' AND V.id_vehiculo NOT IN (SELECT id_vehiculo FROM Alquileres WHERE fecha_llegada IS NULL); -- 20. Obtener los vehículos disponibles de tipo "SUV" con su placa, modelo y valor de alquiler por semana`
- `SELECT C.nombre1, C.apellido1, A.fecha_esperada FROM Alquileres A JOIN Clientes C ON A.id_cliente = C.id_cliente WHERE A.fecha_llegada > A.fecha_esperada; -- 21. Obtener los clientes con alquileres retrasados y la fecha esperada de retorno`

- SELECT S.direccion, S.ciudad, COUNT(E.id_empleado) AS num_empleados FROM Sucursales S JOIN Empleados E ON S.id_sucursal = E.id_sucursal GROUP BY S.id_sucursal HAVING COUNT(E.id_empleado) > 2; -- 22. Listar las sucursales con más de 2 empleados
- SELECT T.tipo, COUNT(V.id_vehiculo) AS cantidad_vehiculos FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV WHERE V.id_vehiculo NOT IN (SELECT id_vehiculo FROM Alquileres WHERE fecha_llegada IS NULL) GROUP BY T.id_tipoV; -- 23. Mostrar la cantidad de vehículos disponibles por cada tipo de vehículo
- SELECT C.nombre1, C.apellido1, V.placa, A.fecha_salida, A.fecha_esperada FROM Alquileres A JOIN Clientes C ON A.id_cliente = C.id_cliente JOIN Vehiculos V ON A.id_vehiculo = V.id_vehiculo; -- 24. Obtener los detalles de los alquileres por cliente y vehículo
- SELECT T.tipo, COUNT(V.id_vehiculo) AS cantidad_vehiculos FROM Vehiculos V JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV GROUP BY T.tipo; -- 25. Mostrar la cantidad de vehículos de cada tipo de vehículo

Funciones

Es un objeto que se crea con la sentencia CREATE FUNCTION y se invoca con la sentencia SELECT o dentro de una expresión. Una función puede tener cero o muchos parámetros de entrada y siempre devuelve un valor, asociado al nombre de la función.

- Ahora, en cada imagen podremos observar una función y su respectiva consulta

1. Función: descuento

a. Descripción: Esta función devuelve el porcentaje de descuento disponible para un tipo de vehículo específico en un periodo de tiempo actual

b. Sintaxis:

```
DELIMITER //
CREATE FUNCTION descuento(id_tipoV INT)
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE descuento INT DEFAULT 0;

    SELECT porcentaje_descuento INTO descuento
    FROM Descuentos
    WHERE id_tipoV = id_tipoV AND CURDATE() BETWEEN fecha_inicio
    AND fecha_fin
    LIMIT 1;
    RETURN descuento;
```



```
END //
DELIMITER ;
```

c. Parámetros: id_tipoV INT

d. Retorno: La función retorna el valor del descuento

e. Ejemplo de implementación:

- SELECT T.tipo, descuento(T.id_tipoV) AS descuento_aplicado FROM Tipo_vehiculo T;

2. Función: total_alquileres_cliente

a. Descripción: Esta función devuelve el número total de alquileres realizados por un cliente específico.

b. Sintaxis:

```
DELIMITER //
CREATE FUNCTION total_alquileres_cliente(id_cliente INT)
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE total INT DEFAULT 0;

    SELECT COUNT(*) INTO total
    FROM Alquileres
    WHERE id_cliente = id_cliente;

    RETURN total;
END //
DELIMITER ;
```

c. Parámetros: id_cliente INT

d. Retorno: La función retorna el valor total de alquileres realizados

e. Ejemplo de implementación:

- SELECT C.nombre1, C.apellido1, total_alquileres_cliente(C.id_cliente) AS alquileres_realizados FROM Clientes C;

3. Función: `total_ingresos_periodo`

a. **Descripción:** Esta función devuelve el total de ingresos generados por los alquileres dentro de un rango de fechas específico.

b. **Sintaxis:**

```
DELIMITER //
CREATE FUNCTION total_ingresos_periodo(fecha_inicio DATE,
fecha_fin DATE)
RETURNS INT DETERMINISTIC
BEGIN
DECLARE total INT DEFAULT 0;
SELECT SUM(valor_pagado) INTO total
FROM Alquileres
WHERE fecha_salida BETWEEN fecha_inicio AND fecha_fin;
RETURN total;
END //
DELIMITER ;
```

c. **Parámetros:** `fecha_inicio DATE`, `fecha_fin DATE`

d. **Retorno:** La función retorna el valor total de ingresos generados

e. **Ejemplo de implementación:**

- `SELECT total_ingresos_periodo('2024-01-01', '2024-01-31') AS ingresos_enero;`

4. Función: `vehiculos_disponibles`

a. **Descripción:** Esta función devuelve el número de vehículos disponibles para alquilar

b. **Sintaxis:**

```
DELIMITER //
CREATE FUNCTION vehiculos_disponibles()
RETURNS INT DETERMINISTIC
BEGIN
DECLARE total INT DEFAULT 0;
SELECT COUNT(*) INTO total
FROM Vehiculos
WHERE id_vehiculo NOT IN (SELECT id_vehiculo FROM Alquileres
WHERE fecha_llegada IS NULL);
RETURN total;
END //
DELIMITER ;
```

c. **Retorno:** La función devuelve el número total de vehículos disponibles

d. **Ejemplo de implementación:**

- `SELECT vehiculos_disponibles() AS total_vehiculos_disponibles;`

5. Función: `vehiculo_mas_caro`

a. Descripción: Esta función devuelve el ID del vehículo con el valor de alquiler más alto.

b. Sintaxis:

```
DELIMITER //
CREATE FUNCTION vehiculo_mas_caro()
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE id_vehiculo INT;

    SELECT id_vehiculo INTO id_vehiculo
    FROM Vehiculos
    ORDER BY (SELECT valor_alquiler_dia FROM Tipo_vehiculo WHERE
id_tipoV = Vehiculos.id_tipoV) DESC
    LIMIT 1;

    RETURN id_vehiculo;
END //
DELIMITER ;
```

c. Retorno: La función retorna el valor total de ingresos generados

d. Ejemplo de implementación:

- `SELECT V.placa, V.modelo FROM Vehiculos V WHERE V.id_vehiculo = vehiculo_mas_caro();`

Procedimientos

Se han desarrollado una serie de procedimientos destinados a facilitar al usuario la inserción de datos en diversas tablas, asegurando así la integridad y consistencia de la base de datos mediante el mantenimiento adecuado de las relaciones entre ellas.

1. Procedimiento: `registrar_clientes`

a. Descripción: Este procedimiento se ha creado con el propósito de registrar un nuevo cliente

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE registrar_clientes(IN cedula_cliente VARCHAR(50),
IN nombre1_cliente VARCHAR(50), IN nombre2_cliente VARCHAR(50),
IN apellido1_cliente VARCHAR(50), IN apellido2_cliente VARCHAR(50),
IN celular_cliente VARCHAR(50), IN correo_electronico_cliente VARCHAR(50))
BEGIN
INSERT INTO Clientes(cedula, nombre1, nombre2, apellido1, apellido2, celular,
correo_electronico)
VALUES (cedula_cliente, nombre1_cliente, nombre2_cliente, apellido1_cliente,
apellido2_cliente, celular_cliente, correo_electronico_cliente);
END
// DELIMITER ;
```

c. Parámetros: `cedula_cliente`, `nombre1_cliente`, `nombre2_cliente`, `apellido1_cliente`, `apellido2_cliente`, `celular_cliente`, `correo_electronico_cliente`

d. Retorno: El procedimiento va a retornar un CREATE hacia la tabla `clientes`

e. Ejemplo de implementación:

```
CALL registrar_clientes('1234567898', 'Mayra', 'Alejandra', 'Guevara',
'Jaimes', 987654423, 'mayra.guevara@cliente.com');
```

2. Procedimiento: `descuento`

a. Descripción: Este procedimiento se ha creado con el propósito de aplicar un descuento a un tipo de vehículo

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE registrar_clientes(IN cedula_cliente VARCHAR(50),
IN nombre1_cliente VARCHAR(50), IN nombre2_cliente VARCHAR(50),
IN apellido1_cliente VARCHAR(50), IN apellido2_cliente VARCHAR(50),
IN celular_cliente VARCHAR(50), IN correo_electronico_cliente VARCHAR(50))
BEGIN
INSERT INTO Clientes(cedula, nombre1, nombre2, apellido1, apellido2, celular,
correo_electronico)
VALUES (cedula_cliente, nombre1_cliente, nombre2_cliente, apellido1_cliente,
apellido2_cliente, celular_cliente, correo_electronico_cliente);
END
// DELIMITER ;
```

c. Parámetros: `fechaInicio_descuento`, `fechaFin_descuento`,
`porcentaje_descuentoB`, `idTipoV_descuento`

d. Retorno: El procedimiento va a retornar un CREATE hacia la tabla descuentos

e. Ejemplo de implementación:

```
CALL descuento('2024-05-01', '2024-05-18', 15, 1);
```

3. Procedimiento: `consultar_historial`

a. Descripción: Este procedimiento se ha creado con el propósito de consultar el historial de un cliente

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE consultar_historial(IN id_cliente_historial INT)
BEGIN
SELECT A.id_alquiler, V.placa, V.modelo, A.fecha_salida, A.fecha_llegada,
A.valor_cotizado, A.valor_pagado FROM Alquileres A
INNER JOIN Vehiculos V ON A.id_vehiculo = V.id_vehiculo WHERE A.id_cliente =
id_cliente_historial;
END //
DELIMITER ;
```

c. Parámetros: id_cliente_historial

d. Retorno: El procedimiento va a retornar una consulta sobre el historial de un cliente

e. Ejemplo de implementación:

CALL consultar_historial(100);

4. Procedimiento: `consultar_precio`

a. Descripción: Este procedimiento se ha creado con el propósito de consultar vehículos por precio

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE consultar_precio(IN precio_minimo INT, precio_maximo INT)
BEGIN
    SELECT V.id_vehiculo, V.placa, V.modelo, T.tipo, T.valor_alquiler_dia,
    T.valor_alquiler_semana FROM Vehiculos V
    INNER JOIN Tipo_vehiculo T ON V.id_tipoV = T.id_tipoV WHERE
    (T.valor_alquiler_dia BETWEEN precio_minimo AND precio_maximo)
    OR (T.valor_alquiler_semana BETWEEN precio_minimo AND precio_maximo);
END //
DELIMITER ;
```

c. Parámetros: precio_minimo, precio_maximo

d. Retorno: El procedimiento va a retornar una consulta de vehículos por precio

e. Ejemplo de implementación:

CALL consultar_precio(50,200);

5. Procedimiento: `eliminar_descuento`

a. Descripción: Este procedimiento se ha creado con el propósito de eliminar un registro por descuento

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE eliminar_descuento(IN id_descuentos INT)
BEGIN
    DELETE FROM Descuentos WHERE id_descuento = id_descuentos;
END //
DELIMITER ;
```

c. Parámetros: id_descuentos

d. Retorno: El procedimiento va a retornar la eliminación de un registro por descuento

e. Ejemplo de implementación:

CALL eliminar_descuento(90);

6. Procedimiento: `registrar_alquiler`

a. Descripción: Este procedimiento se ha creado con el propósito de registrar un alquiler

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE registrar_alquiler (IN p_id_vehiculo INT, IN p_id_cliente INT,
IN p_id_empleado INT, IN p_id_sucursal INT, IN p_fecha_salida DATE,
IN p_fecha_esperada DATE, IN p_fecha_llegada DATE, IN p_valor_cotizado INT,
IN p_valor_pagado INT)
BEGIN
INSERT INTO Alquileres (id_vehiculo, id_cliente, id_empleado, id_sucursal,
fecha_salida, fecha_esperada, fecha_llegada, valor_cotizado, valor_pagado)
VALUES (p_id_vehiculo, p_id_cliente, p_id_empleado, p_id_sucursal,
p_fecha_salida, p_fecha_esperada, p_fecha_esperada, p_valor_cotizado, p_valor_pagado);
END //
DELIMITER ;
```

c. Parámetros: p_id_vehiculo, p_id_cliente

d. Retorno: El procedimiento va a retornar el registro de un alquiler

e. Ejemplo de implementación:

CALL registrar_alquiler(1, 100, 10, 3, '2024-12-01', '2024-12-10',
'2024-12-10', 350, 350);

7. Procedimiento: `actualizar_alquiler`

a. Descripción: Este procedimiento se ha creado con el propósito de actualizar un alquiler

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE actualizar_alquiler (IN p_id_alquiler INT, IN p_fecha_llegada
DATE, IN p_valor_pagado INT)
BEGIN
UPDATE Alquileres
SET fecha_llegada = p_fecha_llegada, valor_pagado = p_valor_pagado
WHERE id_alquiler = p_id_alquiler;
END //
```

DELIMITER ;

c. Parámetros: p_id_alquiler, p_fecha_llegada, p_valor_pagado

d. Retorno: El procedimiento va a retornar la actualización de un alquiler

e. Ejemplo de implementación:

CALL actualizar_alquiler(1, '2024-12-09', 370);

8. Procedimiento: historial_vehiculo

a. Descripción: Este procedimiento se ha creado con el propósito de consultar historial de alquileres de un vehículo

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE historial_vehiculo ( IN p_id_vehiculo INT)
BEGIN
SELECT A.id_alquiler, C.nombre1, C.apellido1, A.fecha_salida, A.fecha_llegada,
A.valor_cotizado
FROM Alquileres A
JOIN Clientes C ON A.id_cliente = C.id_cliente
WHERE A.id_vehiculo = p_id_vehiculo LIMIT 1;
END //
DELIMITER ;
```

c. Parámetros: p_id_vehiculo

d. Retorno: El procedimiento va a retornar una consulta del historial de alquileres de un vehículo

e. Ejemplo de implementación:

CALL historial_vehiculo(5);

9. Procedimiento: inicio_sesion_cliente

a. Descripción: Este procedimiento se ha creado con el propósito del inicio de sesión de un cliente

b. Sintaxis:

```
DELIMITER //
CREATE PROCEDURE inicio_sesion_cliente ( IN p_cedula VARCHAR(50), IN
p_correo_electronico VARCHAR(50))
BEGIN
IF EXISTS (SELECT 1 FROM Clientes WHERE cedula = p_cedula AND
correo_electronico = p_correo_electronico) THEN
```



```

SELECT 'Inicio de sesión exitoso' AS mensaje;
ELSE
SELECT 'Cédula o correo electrónico incorrectos' AS mensaje;
END IF;
END //
DELIMITER ;

```

c. Parámetros: p_cedula, p_correo_electronico

d. Retorno: El procedimiento va a retornar un mensaje con el inicio de sesión exitoso o equivocado

e. Ejemplo de implementación:

```
CALL inicio_sesion_cliente('1234567898', 'mayra.guevara@cliente.com');
```

10. Procedimiento: consultar_alquileres_pendientes_cliente

a. Descripción: Este procedimiento se ha creado con el propósito de consultar los alquileres pendientes de un cliente

b. Sintaxis:

```

DELIMITER //
CREATE PROCEDURE consultar_alquileres_pendientes_cliente (IN p_id_cliente
INT)
BEGIN
SELECT A.id_alquiler, V.placa, V.modelo, A.fecha_salida, A.fecha_esperada,
A.fecha_llegada, A.valor_cotizado
FROM Alquileres A
INNER JOIN Vehiculos V ON A.id_vehiculo = V.id_vehiculo
WHERE A.id_cliente = p_id_cliente
AND (A.fecha_llegada IS NULL OR A.fecha_llegada > CURDATE());
END //
DELIMITER ;

```

c. Parámetros: p_id_cliente

d. Retorno: El procedimiento va a retornar una consulta de los alquileres pendientes de un cliente

e. Ejemplo de implementación:

```
CALL consultar_alquileres_pendientes_cliente(100);
```