# REPORTE DE PRÁCTICA NO. 1.2

## GESTIÓN DE FLOTILLA DE AUTOS

ALUMNO:Israel Campos Vázquez
Dr. Eduardo Cornejo-Velázquez

# 1. Introducción

El proceso sistemático y metodológico en el modelado de bases de datos relacionales es muy importante porque nos ayudará a tener una estructura bien organizada y coherente, lo que, a su vez, nos ayudará a eficientar el tiempo de desarrollo y a evitar posibles errores de redundancia de datos, errores en las relaciones entre las entidades, errores de identificación de atributos, entre otros.

Además, hacer un buen diseño de la base de datos puede ayudarnos a darle un mejor mantenimiento al sistema, y si es necesario hacerlo más grande, un buen diseño nos garantiza una buena escalabilidad.

También, un buen diseño y un proceso sistemático nos permitirá visualizar de mejor manera los posibles casos de uso de nnuestro sistema, así como las medidas de seguridad y control de acceso que debemos implementar.

En el contexto de esta práctica (flotilla de autos), un proceso sistemático nos dará las herramientas para realizar un sistema que permita la buena administración y gestión de su flotilla a nuestro cliente. De este modo, podrá elevar la productividad, disminuir costos y obtener un mejor Retorno de Inversión (ROI)

# 2. Marco teórico

## Análisis de requerimientos

La gestión de una flotilla de autos implica tener el control de aspectos como el gasto de gasolina, mantenimiento de los autos, verificación de rutas y regulación de documentos. Por ello, debemos definir con precisión los requerimientos del sistema.

Los requisitos incluyen el registro de vehículos, la asignación de conductores, el historial de mantenimiento, disponibilidad de los autos, consumo y tipo de combustible, asignación de las rutas más rápidas. Además, debemos tener control de documentos como seguros, tenencias, tarjetas de circulación, licencias de conducir y verificaciones vehiculares. También debemos contemplar que el sistema tenga buena escalabilidad, excelente seguridad, y facil accesibilidad para el administrador.

Según Silberschatz, Korth y Sudarshan (2002) [2], un sistema de base de datos bien diseñado debe garantizar la integridad y disponibilidad de los datos, evitando redundancias y mejorando el acceso eficiente a la información. Sabiendo esto, estas garantías deben ser parte de nuestro sistema .

## Modelo Entidad - Relación

El modelo entidad-relación es un esquema que considero fundamental en el diseño de bases de datos, ya que nos permite observar gráficamente las entidades, sus atributos y los tipos de relaciones que tendrán con otras entidades. En el contexto de la práctica (flotilla de autos), he identificado las siguientes entidades:

Vehículo: Cada auto de la flotilla y tienen atributos como marca, modelo, año, placa, disponibilidad, rendimiento de combustible y qué seguro tiene.
Conductor: Choferes que conducirán la flotilla y tienen atributos como nombre, licencia, telefono, disponibilidad y CURP.
Mantenimiento: Registra el historial de mantenimientos de los autos y tiene atributos como fecha, costo y descripción del servicio.
Ruta: Información sobre los viajes realizados y tiene atributos como fecha, origen, destino, kilometraje, hora de salida y hora de llegada.
Combustible: Esta entidad nos puede ayudar a calcular automáticamente el costo que tuvo un viaje en cuestión de combustible, manteniendo actualizados sus atributos como tipo de gasolina y precio.

Estas entidades estarán relacionadas entre sí, y probablemente sea necesario agregar entidades intermedias para las relaciones muchos a muchos, pero eso lo realizaremos en el desarrollo. Por el momento identificamos relaciones como:
-Un combustible puede tener muchos vehículos asociados, pero un vehículo no pueden tener muchos combustibles.
-Un vehículo puede tener muchos mantenimientos, pero un mantenimiento no puede tener muchos vehículos.
-Un conductor puede tener muchas rutas (pero no al mismo tiempo), pero una ruta no puede tener muchos conductores a la vez.
-Un vehículo puede tener muchas rutas (pero no al mismo tiempo), pero una ruta no puede tener muchos vehículos a la vez.
Según Elmasri y Navathe (2016) [3], el modelo E-R es clave para estructurar la base de datos de manera que se minimicen inconsistencias y se facilite la recuperación de información.

## Modelo relacional

El modelo relacional es el siguiente paso al modelo entidad-relación, ya que sintetiza la información en tablas, representa las relaciones con claves foráneas, entre otras diferencias.

Para el modelo relacional mantendremos la información presentada en el modelo entidad-relación.

Como mencionan Connolly y Begg (2015) [4], la normalización en el modelo relacional ayuda a reducir la redundancia de datos y mejora la eficiencia en las consultas.

### SQL

SQL (Structured Query Language) es un lenguaje utilizado para la gestión de bases de datos relacionales. Su función principal es permitir la creación, modificación y consulta de los datos almacenados en un sistema de gestión de bases de datos (SGBD).

Crearemos la Base de Datos en lenguaje SQL para gestionar la información anteriormente mencionada de nuestra flotilla de autos.

## 3. Herramientas empleadas

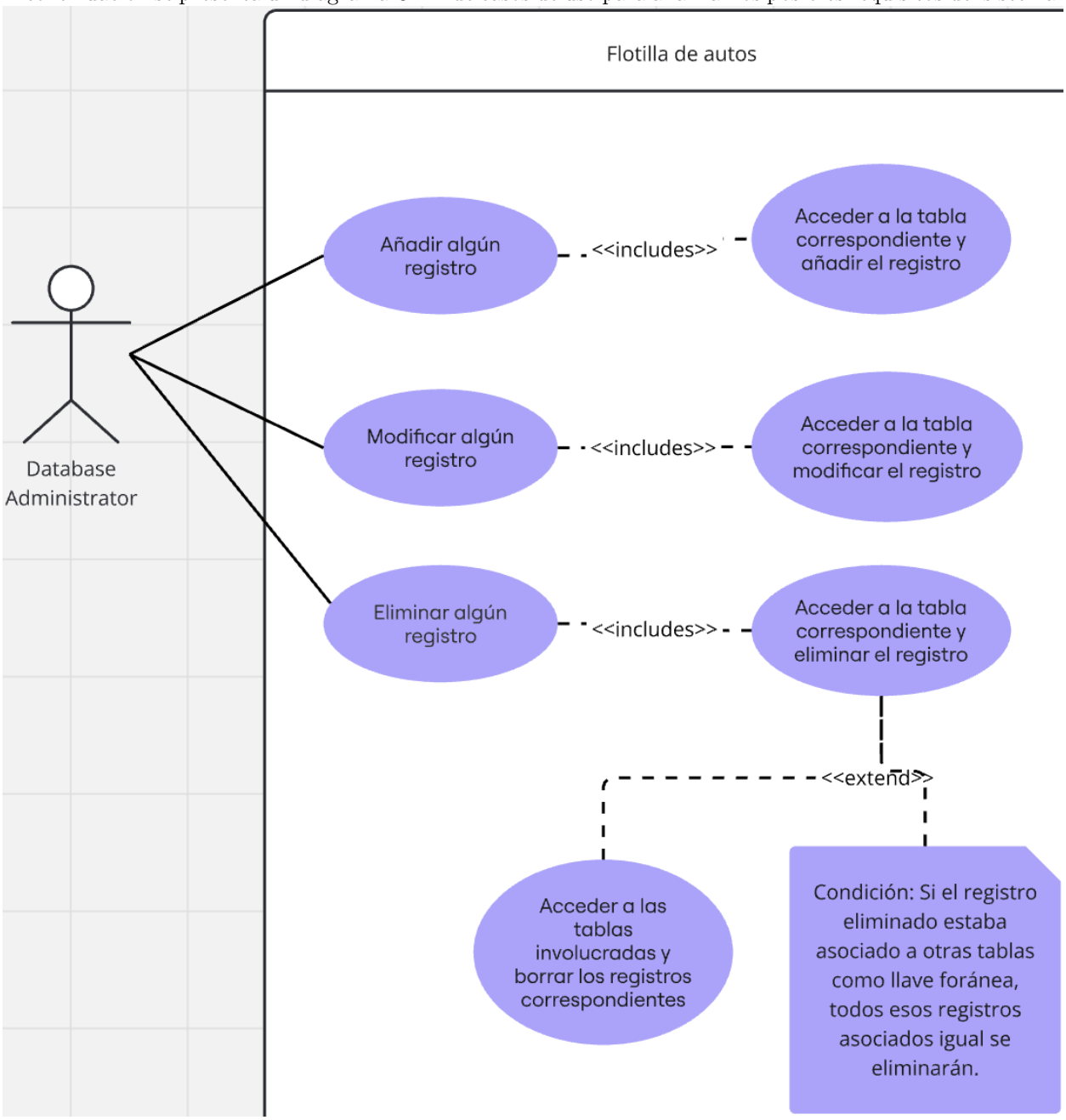
1. ERD Plus: Es una herramienta en línea que se utiliza para el diseño de bases de datos. Permite crear diagramas entidad-relación, en base a dicho diagrama generar el modelo relacional y por último, nos genera las sentencias SQL para crear nuestras tablas con sus atributos y relaciones. También nos permite exportar los diagramas a PNG o PDF.
2. MySQL Server: Es un sistema de gestión de bases de datos relacionales. Utiliza el lenguaje SQL para la administración y manipulación de datos. Se emplea para almacenar, organizar y gestionar información dependiendo el contexto en el que se utilice.

   Se suele utilizar para desarrollar aplicaciones web, gestionar datos de empresas, análisis de datos, entre otros.

# 4. Desarrollo

## Análisis de requisitos

A continuación se presenta un diagrama UML de casos de uso para analizar los posibles requisitos del sistema.



## Modelo Entidad - Relación

En la Tabla 1 se presenta la propuesta de Modelo Entidad - Relación para el sistema de gestión de flotilla de autos.

Table 1: Matriz de relaciones.

| Entidades | Vehiculo | Conductor | Mantenimiento | Ruta | Combustible |
|---|---|---|---|---|---|
| Vehiculo | | | X | X | X |
| Conductor | | | | X | |
| Mantenimiento | X | | | | |
| Ruta | X | X | | | |
| Combustible | X | | | | |

En la Figura 1 se presenta la propuesta de Modelo Entidad - Relación para el sistema de gestión de flotilla de autos.

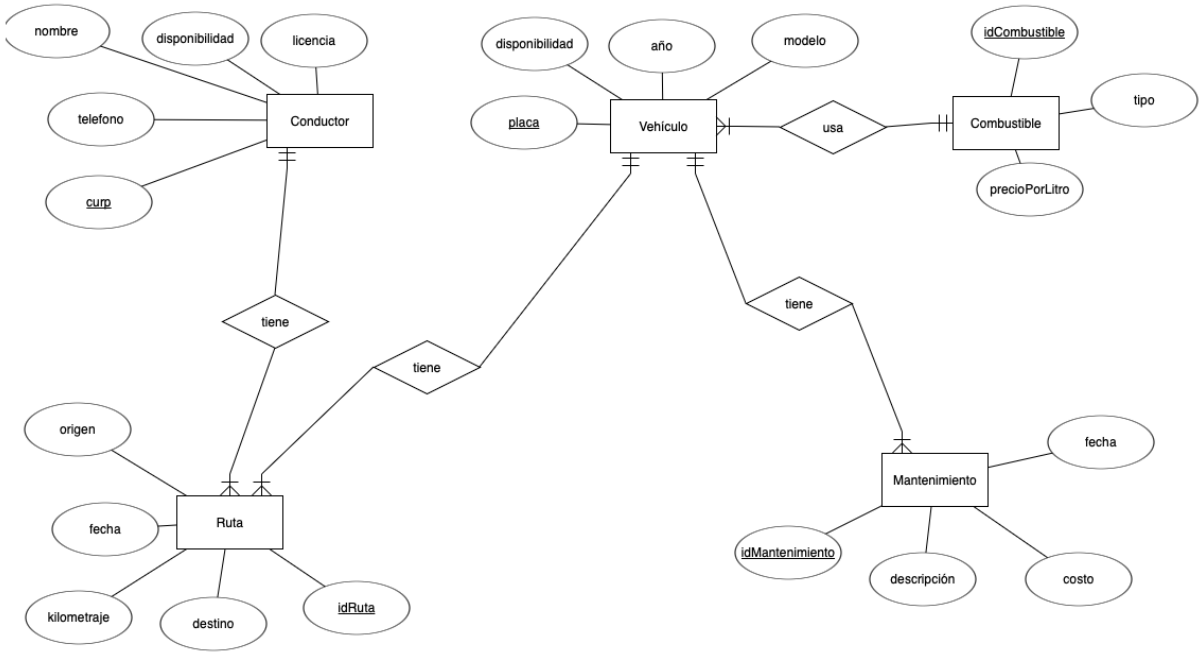



Figure 1: Modelo Entidad - Relación propuesto.

## Modelo relacional

En la Figura 2 se presenta la propuesta de Modelo Relacional para el sistema de gestión de flotilla de autos.

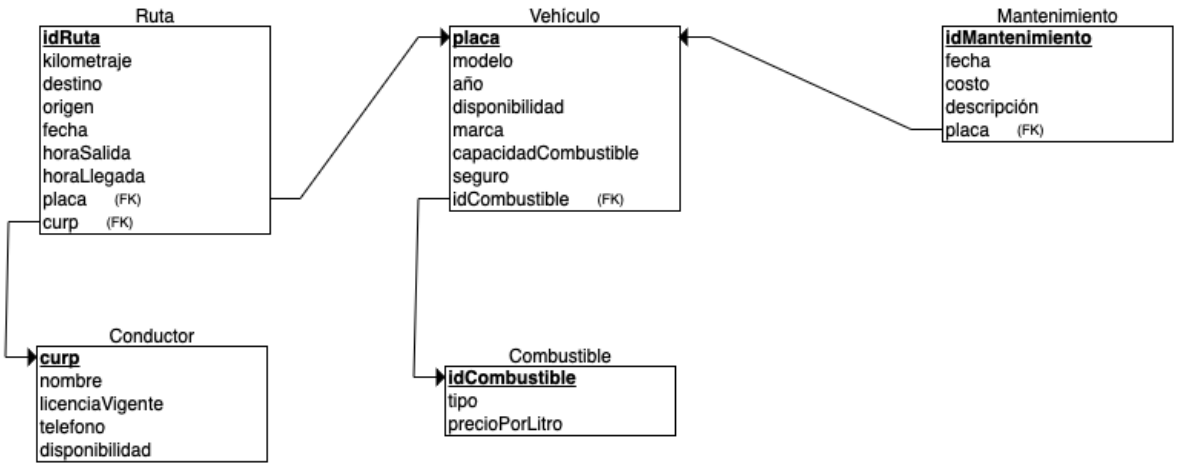



Figure 2: Modelo Relacional propuesto.

## Sentencias SQL

En el Listado 1 se presenta la sentencia SQL para crear la base de datos flotilla de autos.

En el Listado 2 se presentan las sentencias SQL para crear las tablas de la Base de Datos.

En el Listado 3 se presentan las sentencias SQL para insertar registros en las tablas de la Base de Datos.

En el Listado 4 se presentan las sentencias SQL para validar si la licencia del conductor que se quiere asignar a una ruta está vigente.

En el Listado 5 se presentan las sentencias SQL para validar si el conductor y el vehículo que se quieren asignar a una ruta están disponibles.

En el Listado 6 se presentan las sentencias SQL para validar si el conductor o el vehículo no se encuentran en viaje en el momento en que se quieren asignar a otra ruta.

En el Listado 7 se presentan las sentencias SQL para realizar una función que, enviandole como parámetro el idRuta, calcule el costo que tuvo dicho viaje en cuestión de combustible.

En el Listado 8 se presentan las sentencias SQL para crear una vista que muestre el resumen de cada viaje.

Listing 1: Crear base de datos flotilla de autos.

```
CREATE DATABASE flotilla;
```

Listing 2: Crear tablas de las bases de datos flotilla de autos.

```
CREATE TABLE Conductor
(
  nombre VARCHAR(80) NOT NULL,
  licenciaVigente BOOLEAN NOT NULL,
  telefono CHAR(15) NOT NULL,
  curp VARCHAR(20) NOT NULL,
  disponibilidad BOOLEAN NOT NULL DEFAULT 1,
  PRIMARY KEY (curp)
);

CREATE TABLE Combustible
(
  idCombustible INT NOT NULL,
  tipo VARCHAR(80) NOT NULL,
  precioPorLitro FLOAT NOT NULL,
  PRIMARY KEY (idCombustible)
);

CREATE TABLE Vehículo
(
  modelo VARCHAR(80) NOT NULL,
  placa VARCHAR(15) NOT NULL,
  año YEAR NOT NULL,
  disponibilidad BOOLEAN NOT NULL DEFAULT 1,
  marca VARCHAR(80) NOT NULL,
  capacidadCombustible FLOAT NOT NULL,
  seguro VARCHAR(80) NOT NULL,
  idCombustible INT NOT NULL,
  PRIMARY KEY (placa),
  FOREIGN KEY (idCombustible) REFERENCES Combustible(idCombustible)
);

CREATE TABLE Mantenimiento
(
  fecha DATE NOT NULL,
  costo FLOAT NOT NULL,
  idMantenimiento INT NOT NULL,
  descripción VARCHAR(100) NOT NULL,
  placa VARCHAR(15) NOT NULL,
  PRIMARY KEY (idMantenimiento),
  FOREIGN KEY (placa) REFERENCES Vehículo(placa)
);

CREATE TABLE Ruta
(
  kilometraje FLOAT NOT NULL,
  destino VARCHAR(80) NOT NULL,
  origen VARCHAR(80) NOT NULL,
  fecha DATE NOT NULL,
  idRuta INT NOT NULL,
  horaSalida TIME NOT NULL,
  horaLlegada TIME NOT NULL,
  placa VARCHAR(15) NOT NULL,
```

```
  curp VARCHAR(20) NOT NULL,
  PRIMARY KEY (idRuta),
  FOREIGN KEY (placa) REFERENCES Vehículo(placa),
  FOREIGN KEY (curp) REFERENCES Conductor(curp)
);
```

Listing 3: Inserción de registros.

```
INSERT INTO Combustible VALUES (1, 'Magna', 24.79),
                               (2,'Premium',25.55),
                               (3,'Diesel',26.80);

INSERT INTO Conductor VALUES ('Alejandro-Gonzalez-Cruz',1,'5546986133','GOCA040227HDFMZSA1
                             ('Ruben-Ruiz-Diaz',1,'5546236512','RUDR040227HDFMZSA1',1),
                             ('Luis-Torres-Lopez',0,'5598683214','TOLL040227HDFMZSA1',1),
                             ('Antonio-Cruz-Rosas',1,'5590436712','CRRA040227HDFMZSA1',1),
                             ('Pedro-Fuentes-Herrera',0,'5553681243','FUHP040227HDFMZSA1',

INSERT INTO Vehiculo VALUES ('Rifter','AS12-AS3',2020,1,'Peugeot','Qualitas',1,20),
                            ('Saveiro','FDS32-12',2021,1,'Volkswagen','Qualitas',2,18),
                            ('Oroch','FD3-45G',2020,1,'Renault','GNP',3,16),
                            ('RAM-1200','FDL-42K',2023,1,'RAM','GNP',1,14),
                            ('Rifter','SDF-12',2024,1,'Peugeot','Qualitas',2,15);

INSERT INTO Ruta VALUES (200,'Cuautitlan','Toreo','2025-02-07',10,'17:55:59','21:55:59','A
                        (100,'Coapa','Toreo','2025-02-06',2,'14:31:45','15:55:59','FD
                        (40,'Santa-Fe','Toreo','2025-02-06',3,'13:55:12','14:55:59','I
                        (50,'Lomas-Verdes','San-Mateo','2025-02-08',4,'18:51:39','19:5
                        (60,'Satelite','San-Mateo','2025-02-08',5,'19:52:13','20:55:59

INSERT INTO Mantenimiento VALUES ('2024-10-12',5000,1,'Servicio-completo','AS12-AS3'),
                                 ('2024-12-20',3000,2,'Servicio-parcial','FDS32-12'),
                                 ('2025-01-10',4000,3,'Servicio-completo','FD3-45G'),
                                 ('2024-08-29',3500,4,'Servicio-parcial','FDL-42K'),
                                 ('2025-02-01',4500,5,'Servicio-completo','SDF-12');
```

Listing 4: Validar licencia vigente.

```
CREATE TRIGGER licenciaNoVigente
BEFORE INSERT ON Ruta
FOR EACH ROW
BEGIN
    DECLARE vigente tinyint(1);
    SELECT licenciaVigente INTO vigente FROM Conductor WHERE curp=NEW.curp;
    IF vigente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No-puedes-asignar-sin-licencia-vigente-a-una-ruta'
    end if;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER licenciaNoVigenteUpdate
BEFORE UPDATE ON Ruta
FOR EACH ROW
```

```
BEGIN
    DECLARE vigente tinyint(1);
    SELECT licenciaVigente INTO vigente FROM Conductor WHERE curp=NEW.curp;
    IF vigente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No puedes asignar sin licencia vigente a una ruta'
    end if ;
END //
DELIMITER ;
```

Listing 5: Conductor y Vehículo disponibles.

```
DELIMITER //
CREATE TRIGGER disponibilidadChofer
BEFORE INSERT ON Ruta
FOR EACH ROW
BEGIN
    DECLARE disponibilidadChofer tinyint(1);
    DECLARE llegada time default null;
    DECLARE fecha1 date default null;
    DECLARE horaActual time;
    DECLARE fechaActual date;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT disponibilidad INTO disponibilidadChofer FROM Conductor WHERE curp=N
    IF disponibilidadChofer=0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE curp=NEW.curp ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'El chofer no esta disponible';
        END IF;
        UPDATE Conductor SET disponibilidad = 1 WHERE curp=NEW.curp;
    end if ;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER disponibilidadChoferUpdate
BEFORE UPDATE ON Ruta
FOR EACH ROW
BEGIN
    DECLARE disponibilidadChofer tinyint(1);
    DECLARE llegada time default null;
    DECLARE fecha1 date default null;
    DECLARE horaActual time;
    DECLARE fechaActual date;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT disponibilidad INTO disponibilidadChofer FROM Conductor WHERE curp=N
    IF disponibilidadChofer=0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE curp=NEW.curp ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
```

```
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'El chofer no esta disponible';
        END IF;
        UPDATE Conductor SET disponibilidad = 1 WHERE curp=NEW.curp;
    end if ;
END //
DELIMITER ;


DELIMITER //
CREATE TRIGGER disponibilidadVehiculo
BEFORE INSERT ON Ruta
FOR EACH ROW
BEGIN
    DECLARE disponibilidadVehiculo tinyint(1);
    DECLARE llegada time default null;
    DECLARE fecha1 date default null;
    DECLARE horaActual time;
    DECLARE fechaActual date;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT disponibilidad INTO disponibilidadVehiculo FROM Veh culo WHERE plac
    IF disponibilidadVehiculo=0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE placa=NEW.placa ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'El vehiculo no esta disponible';
        END IF;
        UPDATE Veh culo SET disponibilidad = 1 WHERE placa=NEW.placa;
    end if ;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER disponibilidadVehiculoUpdate
BEFORE UPDATE ON Ruta
FOR EACH ROW
BEGIN
    DECLARE disponibilidadVehiculo tinyint(1);
    DECLARE llegada time default null;
    DECLARE fecha1 date default null;
    DECLARE horaActual time;
    DECLARE fechaActual date;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT disponibilidad INTO disponibilidadVehiculo FROM Veh culo WHERE plac
    IF disponibilidadVehiculo=0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE placa=NEW.placa ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
```

```
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'El vehiculo no esta disponible';
        END IF;
        UPDATE Vehículo SET disponibilidad = 1 WHERE placa=NEW.placa;
    end if ;
END //
DELIMITER ;
```

Listing 6: Validar que vehiculo o conductor no estén en ruta.

```
DELIMITER //
CREATE TRIGGER choferEnRuta
BEFORE INSERT ON Ruta
FOR EACH ROW
BEGIN
    DECLARE horaActual time;
    DECLARE llegada time default null;
    DECLARE fechaActual date;
    DECLARE fecha1 date default null;
    DECLARE rutas int default 0;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT COUNT(*) INTO rutas FROM Ruta WHERE curp=NEW.curp;
    IF rutas > 0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE curp=NEW.curp ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'En estos momentos el conductor esta en ruta';
        end if ;
        IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
            UPDATE Conductor SET disponibilidad = 0 WHERE curp = NEW.curp;
        end if ;
    end if ;
    IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
        UPDATE Conductor SET disponibilidad = 0 WHERE curp = NEW.curp;
    end if ;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER choferEnRutaUpdate
BEFORE UPDATE ON Ruta
FOR EACH ROW
BEGIN
    DECLARE horaActual time;
    DECLARE llegada time;
    DECLARE fechaActual date;
    DECLARE fecha1 date;
    DECLARE rutas int;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT COUNT(*) INTO rutas FROM Ruta WHERE placa=NEW.placa ;
```

```
    IF rutas > 0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE placa=NEW.placa ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'En estos momentos el vehiculo esta en ruta';
        end if;
        IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
            UPDATE Conductor SET disponibilidad = 0 WHERE curp = NEW.curp;
        end if;
    end if;
    IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
        UPDATE Conductor SET disponibilidad = 0 WHERE curp = NEW.curp;
    end if;
END //
DELIMITER ;


DELIMITER //
CREATE TRIGGER vehiculoEnRuta
BEFORE INSERT ON Ruta
FOR EACH ROW
BEGIN
    DECLARE horaActual time;
    DECLARE llegada time;
    DECLARE fechaActual date;
    DECLARE fecha1 date;
    DECLARE rutas int;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT COUNT(*) INTO rutas FROM Ruta WHERE curp=NEW.curp;
    IF rutas > 0 THEN
        SELECT horaLlegada,fecha INTO llegada,fecha1 FROM Ruta
        WHERE curp=NEW.curp ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'En estos momentos el conductor esta en ruta';
        end if;
        IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
            UPDATE Veh culo SET disponibilidad = 0 WHERE placa = NEW.placa;
        end if;
    end if;
    IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
        UPDATE Veh culo SET disponibilidad = 0 WHERE placa = NEW.placa;
    end if;
END //
DELIMITER ;

DELIMITER //
CREATE TRIGGER vehiculoEnRutaUpdate
BEFORE UPDATE ON Ruta
FOR EACH ROW
BEGIN
```

```
    DECLARE horaActual time;
    DECLARE llegada time;
    DECLARE fechaActual date;
    DECLARE fecha1 date;
    DECLARE rutas int;
    SET time_zone = 'America/Mexico_City';
    SELECT CURTIME() INTO horaActual;
    SELECT CURDATE() INTO fechaActual;
    SELECT COUNT(*) INTO rutas FROM Ruta WHERE curp=NEW.curp;
    IF rutas > 0 THEN
        SELECT horaLlegada INTO llegada FROM Ruta
        WHERE curp=NEW.curp ORDER BY fecha DESC, horaLlegada DESC LIMIT 1;
        IF llegada > horaActual and fecha1 = fechaActual THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'En-estos-momentos-el-conductor-esta-en-ruta';
        end if ;
        IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
            UPDATE Vehculo SET disponibilidad = 0 WHERE placa = NEW.placa;
        end if ;
    end if ;
    IF NEW.horaLlegada > horaActual and NEW.fecha = fechaActual THEN
        UPDATE Vehculo SET disponibilidad = 0 WHERE placa = NEW.placa;
    end if ;
END //
DELIMITER ;
```

Listing 7: Función calcular el costo de viaje.

```
DELIMITER //
CREATE FUNCTION costoViaje(ruta INT) RETURNS FLOAT
NOT DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE costo FLOAT DEFAULT 0;
    DECLARE rendimiento FLOAT DEFAULT 0;
    DECLARE precioLitro FLOAT DEFAULT 0;
    DECLARE kilometros FLOAT DEFAULT 0;
    DECLARE vehiculo VARCHAR(15) DEFAULT NULL;
    DECLARE combustible INT DEFAULT NULL;
    SELECT kilometraje, placa INTO kilometros, vehiculo FROM Ruta WHERE idRuta
    SELECT idCombustible, rendimientoCombustible INTO combustible, rendimiento
    SELECT precioPorLitro INTO precioLitro FROM Combustible WHERE idCombustible
    SET costo = (kilometros / rendimiento) * precioLitro;
    RETURN costo;
END //
DELIMITER ;
```

Listing 8: Vista resumen de viaje.

```
SELECT C.nombre, V.placa, R.idRuta, R.fecha, R.horaLlegada, R.horaSalida, R.des
FROM Ruta R
JOIN Vehculo V ON R.placa = V.placa
JOIN Conductor C ON R.curp = C.curp;
```

## 5. Conclusiones

Al realizar esta práctica, pude recordar la síntaxis y funcionamiento de setencias SQL que había utilizado en clases anteriores. Además de utilizar estas sentencias para crear la base de datos, crear tablas, y hacer inserciones de registros en ellas, me propuse de reto expandir mi conocimiento e investigar el funciionamiento de triggers, functions, y views para poder automatizar la base de datos.

Es decir, me propuse hacer validaciones antes de hacer inserciones en la tabla ruta, como checar que el conductor tuviera licencia vigente, validar que el vehículo y conductor solicitados para una nueva ruta estén disponibles y que no estén en otra ruta en ese momento.

Fue un reto gratificante conocer nuevas herramientas que nos brinda SQL y recordas aquellas que ya había visto en otras clases.

# Referencias Bibliográficas

# References


[1] Grabowska, S.; Saniuk, S. (**2022**). Business models in the industry 4.0 environment—results of web of science bibliometric analysis. *J. Open Innov. Technol. Mark. Complex*, *8(1)*, 19.

[2] FUNDAMENTOS DE BASES DE DATOS (4.a ed.). (**2002**). Silberschatz, Korth y Sudarshan. http://biblioteca.univalle.edu.ni/files/original/01aebde3cc06dce33f2538aa2724eb2541cb9473.pdf

[3] Fundamentos de Sistemas de Bases de Datos (5.a ed.). (2007). Elmasri y Navathe. https://gc.scalahed.com/recursos/files/r161r/w24566w/FundamentosDeSistemasDeBasesDeDatos.-1-69.pdf

[4] Database Systems (6.a ed.). (2015). Conolly, Begg. https://dl.ebooksworld.ir/motoman/Pearson.Database.Systems.A.Pr