Laboratorio 3

Antonio Jose Donis Hung - 000408397

Tablas

Colores

```
CREATE TABLE IF NOT EXISTS colores (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(45),
    CONSTRAINT colores_color_unico UNIQUE (nombre)
);
```

Modelos

```
CREATE TABLE IF NOT EXISTS modelos_vehiculos (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(45) NOT NULL,
    valor FLOAT NOT NULL,
    CONSTRAINT modelos_vehiculos_modelo_unico UNIQUE (nombre)
);
```

Constante de precio por capacidad

```
CREATE TABLE IF NOT EXISTS precio_capacidad_constante (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    valor FLOAT NOT NULL
);
```

Constante de precio por puerta

```
CREATE TABLE IF NOT EXISTS precio_puerta_constante (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    valor FLOAT NOT NULL
);
```

Constante impuesto

```
CREATE TABLE IF NOT EXISTS impuesto_constante (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    valor FLOAT NOT NULL
);
```

Constante precio cuando es descapotable

```
CREATE TABLE IF NOT EXISTS precio_descapotable_constante (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    valor FLOAT NOT NULL
);
```

Constante multiplicador de precio por dia

```
CREATE TABLE IF NOT EXISTS multiplicador_semana_consante (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    valor FLOAT NOT NULL
);
```

Ciudades

```
CREATE TABLE IF NOT EXISTS ciudades (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(45) NOT NULL,
    CONSTRAINT ciudades_ciudad_unica UNIQUE (nombre)
);
```

Empleados

```
CREATE TABLE IF NOT EXISTS empleados (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    ciudad_residencia INT NOT NULL,
    cedula VARCHAR(45) NOT NULL,
    nombres VARCHAR(255) NOT NULL,
    apellidos VARCHAR(255) NOT NULL,
    direccion VARCHAR(1000) NOT NULL,
    telefono_celular VARCHAR(45) NOT NULL,
    correo VARCHAR(255) NOT NULL,
    CONSTRAINT empleados_cedula_unica UNIQUE (cedula),
    CONSTRAINT fk_empleados_ciudad_residencia FOREIGN KEY (ciudad_residencia)
REFERENCES ciudades(id)
);
```

Sucursales

```
CREATE TABLE IF NOT EXISTS sucursales (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    ciudades_id INT NOT NULL,
    direccion VARCHAR(1000) NOT NULL,
    telefono_fijo VARCHAR(45) NOT NULL,
    telefono_celular VARCHAR(45) NOT NULL,
    correo VARCHAR(255) NOT NULL,
    CONSTRAINT fk_sucursales_ciudades_id FOREIGN KEY (ciudades_id) REFERENCES
ciudades(id)
);
```

Vehiculos

```
CREATE TABLE IF NOT EXISTS vehiculos (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    modelos_vehiculos_id INT NOT NULL,
    nombre VARCHAR(45) NOT NULL,
    colores_id INT NOT NULL,
    precio_referencia FLOAT NOT NULL,
    puertas INT NOT NULL,
    capacidad INT NOT NULL,
    es_descapotable BOOLEAN NOT NULL,
    motor VARCHAR(500) NOT NULL,
    precio_semana FLOAT,
    precio_dia FLOAT,
    CONSTRAINT vehiculos_vehiculo_unico UNIQUE (nombre, modelos_vehiculos_id,
colores_id, precio_referencia, puertas, capacidad, es_descapotable, motor),
    CONSTRAINT fk_vehiculos_modelos_vehiculos_id FOREIGN KEY
(modelos_vehiculos_id) REFERENCES modelos_vehiculos(id),
    CONSTRAINT fk_vehiculos_colores_id FOREIGN KEY (colores_id) REFERENCES
colores(id)
);
```

Inventario

```
CREATE TABLE IF NOT EXISTS inventario (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    sucursales_id INT NOT NULL,
    vehiculos_id INT NOT NULL,
    placa VARCHAR(45) NOT NULL,
    disponible  BOOLEAN NOT NULL DEFAULT true,
    CONSTRAINT inventario_placa_unica UNIQUE (placa),
    CONSTRAINT fk_inventario_sucursales_id FOREIGN KEY (sucursales_id)
REFERENCES sucursales(id),
    CONSTRAINT fk_inventario_vehiculos_id FOREIGN KEY (vehiculos_id) REFERENCES
vehiculos(id)
);
```

Clientes

```
CREATE TABLE IF NOT EXISTS clientes (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    ciudad_residencia INT NOT NULL,
    cedula VARCHAR(45) NOT NULL,
    nombres VARCHAR(255) NOT NULL,
    apellidos VARCHAR(255) NOT NULL,
    direccion VARCHAR(1000) NOT NULL,
    telefono_celular VARCHAR(45) NOT NULL,
    correo VARCHAR(255) NOT NULL,
    contrasena VARCHAR(128) NOT NULL,
    CONSTRAINT clientes_cedula_unica UNIQUE (cedula),
    CONSTRAINT fk_clientes_ciudad_residencia FOREIGN KEY (ciudad_residencia)
REFERENCES ciudades(id)
);
```

Alquileres

```
CREATE TABLE IF NOT EXISTS alquileres (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    clientes_id INT NOT NULL,
    empleados_id INT NOT NULL,
    sucursal_salida INT NOT NULL,
    sucursal_entrega INT,
    inventario_id INT NOT NULL,
    fecha_salida DATETIME NOT NULL,
    fecha_llegada_esperada DATETIME NOT NULL,
    fecha_llegada DATETIME,
    precio_semana_final FLOAT NOT NULL,
    precio_dia_final FLOAT NOT NULL,
    semanas_alquilado INT NOT NULL,
    dias_alquilado INT NOT NULL,
    valor_cotizado FLOAT,
    valor_pagado FLOAT,
    CONSTRAINT fk_alquileres_clientes_id FOREIGN KEY (clientes_id) REFERENCES
clientes(id),
    CONSTRAINT fk_alquileres_empleados_id FOREIGN KEY (empleados_id) REFERENCES
empleados(id),
    CONSTRAINT fk_alquileres_sucursal_salida  FOREIGN KEY (sucursal_salida)
REFERENCES sucursales(id),
    CONSTRAINT fk_alquileres_sucursal_entrega FOREIGN KEY (sucursal_entrega)
REFERENCES sucursales(id),
    CONSTRAINT fk_alquileres_inventario_id FOREIGN KEY (inventario_id)
REFERENCES inventario(id)
);
```

Vistas

Vehiculos disponibles

```
CREATE VIEW vehiculos_disponibles AS
SELECT
    inventario.id AS inventario_id,
    inventario.sucursales_id AS sucursales_id,
    inventario.vehiculos_id AS vehiculos_id,
    inventario.placa AS placa,
    vehiculos.modelos_vehiculos_id AS modelo_id,
    vehiculos.nombre AS nombre,
    vehiculos.colores_id AS color_id,
    vehiculos.precio_referencia AS precio_referencia,
    vehiculos.puertas AS puertas,
    vehiculos.capacidad AS capacidad,
    vehiculos.es_descapotable AS descapotable,
    vehiculos.motor AS motor,
    vehiculos.precio_semana AS precio_semana,
    vehiculos.precio_dia AS precio_dia
FROM
    vehiculos,
    inventario
WHERE
    inventario.vehiculos_id = vehiculos.id AND inventario.disponible;
```

Historial de alquileres

```
CREATE VIEW historial_de_alquileres AS
SELECT
    id,
    clientes_id,
    empleados_id,
    sucursal_salida,
    sucursal_entrega,
    inventario_id,
    fecha_salida,
    fecha_llegada_esperada,
    fecha_llegada,
    precio_semana_final,
    precio_dia_final,
    semanas_alquilado,
    dias_alquilado,
    valor_cotizado,
    valor_pagado
FROM
    alquileres
WHERE
    fecha_llegada != NULL;
```

Triggers

```
-- -- modelos_vehiculos -- --

CREATE TRIGGER modelos_vehiculos_after_update
AFTER UPDATE ON modelos_vehiculos
FOR EACH ROW
BEGIN
    IF OLD.valor != NEW.valor THEN
        CALL actualizar_vehiculos_por_modelo(NEW.id);
    END IF;
END; @@

-- -- precio_puerta_constante -- --
CREATE TRIGGER precio_puerta_constante_after_update
AFTER UPDATE ON precio_puerta_constante
FOR EACH ROW
BEGIN
    CALL actualizar_vehiculos();
END; @@

-- -- precio_capacidad_constante -- --
CREATE TRIGGER precio_capacidad_constante_after_update
AFTER UPDATE ON precio_capacidad_constante
FOR EACH ROW
BEGIN
    CALL actualizar_vehiculos();
END; @@

-- -- precio_descapotable_constante -- --
CREATE TRIGGER precio_descapotable_constante_after_update
AFTER UPDATE ON precio_descapotable_constante
```

```
FOR EACH ROW
BEGIN
    CALL actualizar_vehiculos();
END; @@

DELIMITER ;
```

Procesos y funciones

Obtener id del color

```
DELIMITER @@
CREATE FUNCTION get_color_id(color VARCHAR(45))
RETURNS INT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT id INTO @result FROM colores WHERE nombre = color;
    RETURN @result;
END; @@
DELIMITER ;
```

Obtener id del modelo

```
DELIMITER @@
CREATE FUNCTION get_modelo_id(modelo VARCHAR(45))
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT id INTO @result FROM modelos_vehiculos WHERE nombre = modelo;
    RETURN @result;
END; @@
DELIMITER ;
```

Obtener precio por capacidad

```
DELIMITER @@
CREATE FUNCTION get_precio_capacidad()
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT valor INTO @result FROM precio_capacidad_constante WHERE id = 1;
    RETURN @result;
END; @@
DELIMITER ;
```

Obtener precio por puerta

```
DELIMITER @@
CREATE FUNCTION get_precio_puerta()
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT valor INTO @result FROM precio_puerta_constante WHERE id = 1;
    RETURN @result;
END; @@
DELIMITER ;
```

Obtener impuesto

```
DELIMITER @@
CREATE FUNCTION get_impuesto()
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT valor INTO @result FROM impuesto_constante WHERE id = 1;
    RETURN @result;
END; @@
DELIMITER ;
```

Obtener precio cuando es descapotable

```
DELIMITER @@
CREATE FUNCTION get_precio_descapotable()
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT valor INTO @result FROM precio_descapotable_constante WHERE id = 1;
    RETURN @result;
END; @@
DELIMITER ;
```

Obtener multiplicador de precio por semana

```
DELIMITER @@
CREATE FUNCTION get_multiplicador_semana()
RETURNS FLOAT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT valor INTO @result FROM multiplicador_semana_consante WHERE id = 1;
    RETURN @result;
END; @@
DELIMITER ;
```

Actualizar todos los vehiculos

```
CREATE PROCEDURE
actualizar_vehiculos()
BEGIN
    SET @capacidad_valor = get_precio_capacidad();
    SET @puerta_valor = get_precio_puerta();
    SET @impuesto_valor = get_impuesto();
    SET @descapotable_precio = get_precio_descapotable();
    SET @multiplicador_semana = get_multiplicador_semana();
    UPDATE
        vehiculos,
        modelos_vehiculos
    SET
        vehiculos.precio_semana = @impuesto_valor * (
            (vehiculos.precio_referencia + modelos_vehiculos.valor) +
            (vehiculos.capacidad + @capacidad_valor) +
            (vehiculos.puertas * @puerta_valor) +
            (vehiculos.es_descapotable * @descapotable_precio)
        )
    WHERE
        vehiculos.modelos_vehiculos_id = modelos_vehiculos.id;
    UPDATE
        vehiculos,
        modelos_vehiculos
    SET
        vehiculos.precio_dia = precio_semana / @multiplicador_semana
    WHERE
        vehiculos.modelos_vehiculos_id = modelos_vehiculos.id;

END;
@@
```

Actualizar todos los vehiculos de un modelo

```
CREATE PROCEDURE
actualizar_vehiculos_por_modelo(IN id_modelo INT)
BEGIN
    SET @capacidad_valor = get_precio_capacidad();
    SET @puerta_valor = get_precio_puerta();
    SET @impuesto_valor = get_impuesto();
    SET @descapotable_precio = get_precio_descapotable();
    SET @multiplicador_semana = get_multiplicador_semana();
    UPDATE
        vehiculos,
        modelos_vehiculos
    SET
        vehiculos.precio_semana = @impuesto_valor * (
            (vehiculos.precio_referencia + modelos_vehiculos.valor) +
            (vehiculos.capacidad + @capacidad_valor) +
            (vehiculos.puertas * @puerta_valor) +
            (vehiculos.es_descapotable * @descapotable_precio)
        ),
        vehiculos.precio_dia = precio_semana / @multiplicador_semana
    WHERE
        modeloves_vehiculos.id = @id_modelo
```

```
        AND vehiculos.modelos_vehiculos_id = modelos_vehiculos.id;
END;
@@
```

Registrar cliente

```
CREATE PROCEDURE
registrar_cliente(
    IN v_ciudad_residencia INT,
    IN v_cedula VARCHAR(45),
    IN v_nombres VARCHAR(255),
    IN v_apellidos VARCHAR(255),
    IN v_direccion VARCHAR(1000),
    IN v_telefono_celular VARCHAR(45),
    IN v_correo VARCHAR(255),
    IN v_contrasena VARCHAR(128)
)
BEGIN
    INSERT IGNORE INTO clientes (
        ciudad_residencia,
        cedula,
        nombres,
        apellidos,
        direccion,
        telefono_celular,
        correo,
        contrasena
    ) VALUES (
        v_ciudad_residencia,
        v_cedula,
        v_nombres,
        v_apellidos,
        v_direccion,
        v_telefono_celular,
        v_correo,
        v_contrasena
    );
END;
@@
```

Login cliente

```
CREATE FUNCTION
login_cliente(
    v_correo VARCHAR(255),
    v_password_hash VARCHAR(128)
)
RETURNS INT
LANGUAGE SQL
DETERMINISTIC
BEGIN
    SELECT id INTO @id_cliente FROM clientes WHERE clientes.correo = v_correo
AND clientes.contrasena = v_password_hash;
    RETURN @id_cliente;
END;
@@
```

Registrar vehiculo

```
CREATE PROCEDURE
registrar_vehiculo(
    v_nombre VARCHAR(45),
    v_modelo VARCHAR(45),
    v_color VARCHAR(45),
    v_placa VARCHAR(45),
    v_precio_referencia FLOAT,
    v_puertas INT,
    v_capacidad INT,
    v_es_descapotable BOOLEAN,
    v_motor VARCHAR(500),
    v_id_sucursal INT
)
BEGIN
    SET @id_modelo = get_modelo_id(v_modelo);
    SET @id_color = get_color_id(v_color);
    IF (@id_color = NULL) THEN
        INSERT INTO colores (nombre) VALUES (v_color);
        SET @id_color = get_color_id(v_color);
    END IF;
    IF @id_modelo > 0 THEN
        INSERT IGNORE INTO vehiculos (
            modelos_vehiculos_id,
            nombre,
            colores_id,
            precio_referencia,
            puertas,
            capacidad,
            es_descapotable,
            motor
        ) VALUES (
            @id_modelo,
            v_nombre,
            @id_color,
            v_precio_referencia,
            v_puertas,
            v_capacidad,
            v_es_descapotable,
            v_motor
```

```
        );
        SELECT
            id INTO @id_vehiculo
        FROM
            vehiculos
        WHERE
            modelos_vehiculos_id = @id_modelo
            AND nombre = v_nombre
            AND colores_id = @id_color
            AND precio_referencia = v_precio_referencia
            AND puertas = v_puertas
            AND capacidad = v_capacidad
            AND es_descapotable = v_es_descapotable
            AND motor = v_motor;
        IF @id_vehiculo > 0 THEN
            INSERT INTO inventario (
                sucursales_id,
                vehiculos_id,
                placa
            ) VALUES (
                v_id_sucursal,
                @id_vehiculo,
                v_placa
            );
            CALL actualizar_vehiculos();
        END IF;
    END IF;
END;
@@
```

Alquilar vehiculo

```
CREATE PROCEDURE alquilar_vehiculo(
    IN v_id_cliente INT,
    IN v_id_empleado INT,
    IN v_id_sucursal INT,
    IN v_id_vehiculo INT,
    IN v_numero_semanas INT,
    IN v_numero_dias INT
)
BEGIN
    SET @ahora = NOW();
    SELECT
        inventario_id, precio_semana, precio_dia INTO @id_inventario,
@semana_final, @dia_final
    FROM
        vehiculos_disponibles
    WHERE
        vehiculos_id = v_id_vehiculo LIMIT 1;
    INSERT INTO alquileres (
        clientes_id,
        empleados_id,
        sucursal_salida,
        inventario_id,
        fecha_salida,
        fecha_llegada_esperada,
        precio_semana_final,
```

```
        precio_dia_final,
        semanas_alquilado,
        dias_alquilado,
        valor_cotizado
    ) VALUES (
        v_id_cliente,
        v_id_empleado,
        v_id_sucursal,
        @id_inventario,
        @ahora,
        ADDDATE(@ahora, INTERVAL (v_numero_semanas * 7) + v_numero_dias DAY),
        @semana_final,
        @dia_final,
        v_numero_semanas,
        v_numero_dias,
        @semana_final * v_numero_semanas + @dia_final * v_numero_dias
    );
END;
```