MP0372. Xestión de bases de datos.

UD 6 - Construcción de guiones

Curso académico 22/2023

IES SAN CLEMENTE

Sumario

ENUNCIADO....3
EJERCICIO 1....4
EJERCICIO 2....5
EJERCICIO 3....6
EJERCICIO 4....7
EJERCICIO 5....8
EJERCICIO 6....9
EJERCICIO 7....10
EJERCICIO 8....12
EJERCICIO 9....13
EJERCICIO 10....14

ENUNCIADO

En la base de datos TalleresFaber se van a realizar una serie de mejoras incluyendo ciertas rutinas que mejorarán la funcionalidad de la base de datos.

El modelo de la base de datos es el siguiente:

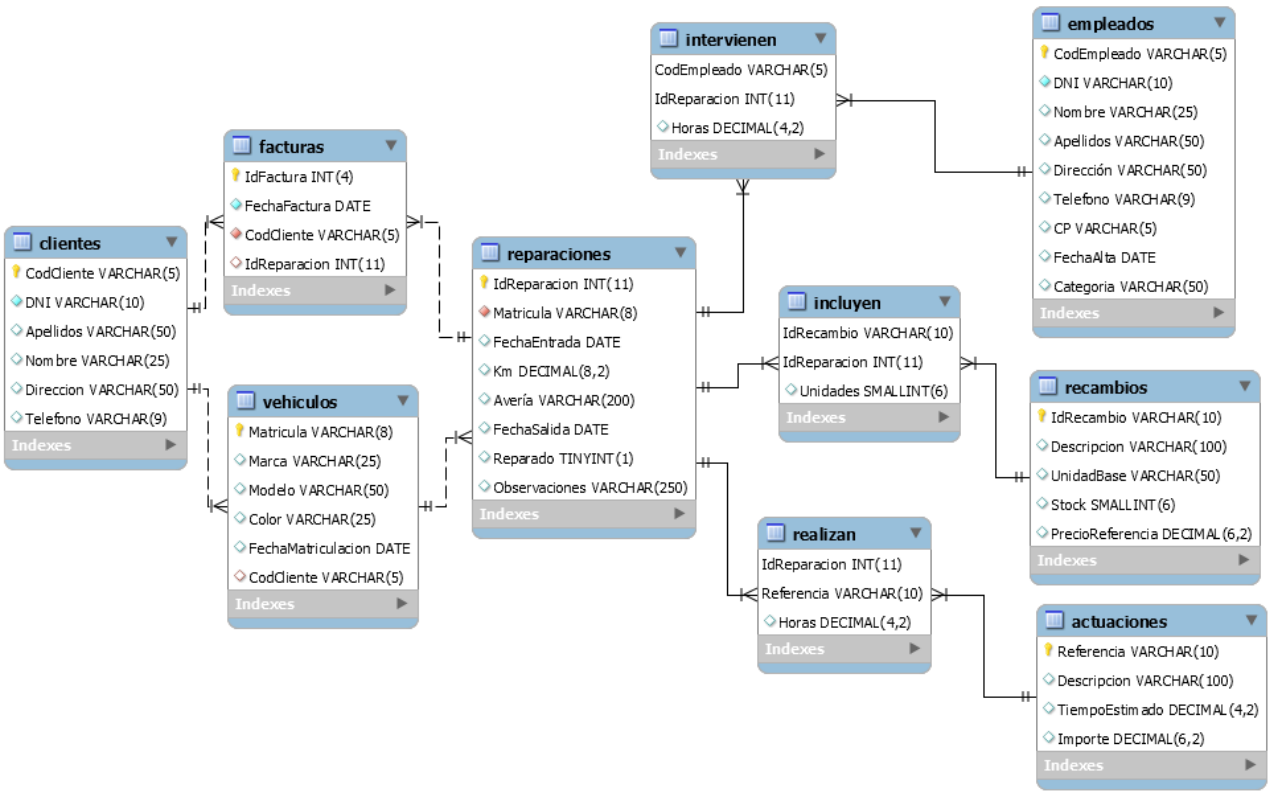

Ilustración del modelo da la base de datos talleresfaber
Workbench (Elaboración propia)

Debes cargar en tu equipo, si no la tienes aún, la base de datos talleresfaber, cuyo script de creación y de inserción de datos se proprorciona en el apartado información de interés de esta msima tarea.

EJERCICIO 1.

Crea un procedimiento que muestre los vehículos (marca, modelo y color) que no estén reparados y los datos de los clientes y vehículos que han entrado a reparar hoy. (En nuestro caso ninguno).

```
DELIMITER |
CREATE PROCEDURE Exercicio1()
BEGIN
 SELECT Marca,Modelo,Color FROM vehiculos
 WHERE Matricula NOT IN (SELECT Matricula FROM reparaciones WHERE Reparado = 1);
 SELECT c.DNI,c.Nombre,c.Apellidos,V.* FROM clientes c INNER JOIN vehiculos v ON
v.CodCliente=c.CodCliente INNER JOIN reparaciones r ON r.Matricula=v.Matricula
 WHERE r.FechaEntrada = CURDATE();
END |
DELIMITER ;

CALL Exercicio1();
```

EJERCICIO 2.

a) Realiza un procedimiento que reciba la matrícula de un vehículo y escriba:

- las características del automóvil
- y el número de reparaciones que ha sufrido ese automóvil
- los empleados que han realizado esas reparaciones
- los datos de los vehículos de la misma marca.

b) Hacer una llamada al procedimiento creado.

a)

```
DELIMITER |
CREATE PROCEDURE Exercicio2(IN Matric VARCHAR(8))
BEGIN
 SELECT * FROM vehiculos
 WHERE Matricula=Matric;
 SELECT COUNT(*) AS TotalReparaciones FROM reparaciones
 WHERE Matricula=Matric;
 SELECT DISTINCT e.Nombre,e.Apellidos FROM intervienen i INNER JOIN empleados e ON
i.CodEmpleado=e.CodEmpleado
 WHERE i.IdReparacion IN (SELECT IdReparacion FROM reparaciones WHERE Matricula=Matric);
 SELECT * FROM vehiculos WHERE Marca IN (SELECT Marca FROM vehiculos
 WHERE Matricula=Matric);
END |
DELIMITER ;
```

b)

```
CALL Exercicio2('2233 ABC');
CALL Exercicio2('1212 DEF');
```

EJERCICIO 3.

Modifica el procedimiento anterior añadiendo un HANDLER que controle que si esa matrícula no está en la base de datos, el resto de instrucciones no se ejecuten.

```
DELIMITER |
CREATE PROCEDURE Exercicio3(IN Matric VARCHAR(8))
BEGIN
 DECLARE existe INT;
 SET existe = 0;
 SELECT COUNT(*) INTO existe
 FROM vehiculos
 WHERE Matricula = Matric;

 IF existe = 0
  THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A matricula non existe';
 ELSE
  SELECT * FROM vehiculos
  WHERE Matricula=Matric;
  SELECT COUNT(*) AS TotalReparaciones FROM REPARACIONES
  WHERE Matricula=Matric;
  SELECT DISTINCT E.Nombre,E.Apellidos FROM intervienen i INNER JOIN empleados E ON i.CodEmpleado=e.CodEmpleado
  WHERE i.IdReparacion IN (SELECT IdReparacion FROM reparaciones WHERE Matricula=Matric);
  SELECT * FROM vehiculos WHERE Marca IN (SELECT Marca FROM vehiculos
  WHERE Matricula=Matric);
 END IF;
END |
DELIMITER ;

CALL Exercicio3('1212 DEF');
CALL Exercicio3('9876ABC');
```

EJERCICIO 4.

Crea una función que:

- actualice el estado de las reparaciones que estén finalizadas en una fecha que se indique
- y que devuelva cuantas reparaciones han finalizado en esa fecha.

```
DELIMITER |
CREATE FUNCTION Exercicio4(fecha DATE) RETURNS INT
BEGIN
 DECLARE idr INT;
 UPDATE reparaciones r SET FechaSalida=fecha,Reparado='1' WHERE r.FechaSalida = fecha and
r.Reparado='0';
 SELECT COUNT(R.IdReparacion) INTO idr FROM reparaciones r WHERE r.FechaSalida = fecha
and r.Reparado='1';
 RETURN idr;
END |
DELIMITER ;

SELECT Exercicio4('2011-01-03')
```

EJERCICIO 5.

a) Crea un procedimiento para dar de alta una nueva reparación para un vehículo y un cliente que no tenemos registrado. Llama al procedimiento ReparacionClienteNuevo.

Incluye un HANDLER que controle que si insertamos un cliente y/o un vehículo que ya existen, el resto de sentencias continúen ejecutándose, y se añade como mínimo la nueva reparación.

b) Para probar el procedimiento toma como referencia los datos siguientes.

Un cliente nuevo nos ha traído su vehículo al taller el día 03/03/2020. En recepción se registran los siguientes datos:

- Del cliente.- Código: 00011, Nombre y apellidos: Tomás Gómez Calle, Teléfono: 22334455.
- Del vehículo.- Matrícula: 3131 FGH, Modelo: Renault Scénic, matriculado el 17/03/2009, 105.000 km;
- De la reparación.- Sustitución de las lámparas delanteras.

```
DELIMITER |

CREATE PROCEDURE ReparacionClienteNuevo(cod varchar(5),dni VARCHAR(10),nombre
VARCHAR(25),apellidos VARCHAR(50),tfno VARCHAR(9),

                            matricula varchar(8),marca varchar(25),modelo
varchar(50),fechamatri DATE,fechaentrada DATE,km DECIMAL (8,2),averia VARCHAR(200))

BEGIN

 DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SELECT 'Xa existe na BBDD';

 INSERT INTO clientes (CodCliente,DNI,Nombre,Apellidos,Telefono) VALUES
(cod,dni,nombre,apellidos,tfno);

 INSERT INTO vehiculos (Matricula,Marca,Modelo,FechaMatriculacion,CodCliente) VALUES
(matricula,marca,modelo,fechamatri,cod);

 INSERT INTO reparaciones (Matricula,FechaEntrada,Km,Avería) VALUES
(matricula,fechaentrada,km,averia);

 END |

DELIMITER ;

CALL ReparacionClienteNuevo('00011','33542295S','Tomás','Gomez
Calle','22334455','3131FGH','Renault','Renault Scénic','2009-03-17','2020-03-
03','105000','Sustitución de las lámparas delanteras')
```

EJERCICIO 6.

Creación de funciones:
a) Diseña una función que calcule el importe de los recambios sustituidos en una reparación.
b) Crea una función que devuelva el importe de las actuaciones que se llevan a cabo en una reparación (para calcular el importe multiplica las horas por el importe de cada actuación).

En ambas funciones Pasar como variable el Id de la reparación.
c) Diseñar una consulta que calcule el importe total (mano de obra y recambios) de las reparaciones que se le hayan realizado al vehículo de matrícula '1313 DEF'.

a)

```
DELIMITER |
CREATE FUNCTION CalcularImporteRecambios(IdRep INT) RETURNS DECIMAL(6,2) DETERMINISTIC
BEGIN
    DECLARE ImporteTotal DECIMAL(6,2);
    SELECT SUM(Unidades*PrecioReferencia) INTO ImporteTotal FROM recambios r
    INNER JOIN incluyen i ON r.IdRecambio = i.IdRecambio
    WHERE I.IdReparacion = IdRep;
    RETURN ImporteTotal;
END |
DELIMITER ;

select CalcularImporteRecambios(1);
```

b)

```
DELIMITER |
CREATE FUNCTION CalcularImporteActuaciones(IdRep INT) RETURNS DECIMAL(6,2) DETERMINISTIC
BEGIN
    DECLARE ImporteTotal DECIMAL(6,2);
    SELECT SUM(Horas*Importe) INTO ImporteTotal FROM actuaciones a
    INNER JOIN realizan r ON a.Referencia = r.Referencia
    WHERE R.IdReparacion = IdRep;
    RETURN ImporteTotal;
END |
DELIMITER ;

select CalcularImporteActuaciones(1);
```

c)

```
SELECT SUM(CalcularImporteRecambios(IdReparacion)) +
SUM(CalcularImporteActuaciones(IdReparacion))
AS ImporteTotal FROM reparaciones WHERE Matricula = '1313 DEF';
```

EJERCICIO 7.

Crea una función que reciba como parámetro de entrada el número correspondiente a un mes y devuelva el importe total facturado ese mes. Utiliza para ello las dos funciones obtenidas en la práctica anterior.

NOTAS:

- Por ejemplo para Enero, número del mes 1.
- Utilizar un cursor para recorrer cada fila de la consulta de los IdReparacion que se obtengan en ese mes.
- Controla mediante un HANDLER que la consulta haya devuelto alguna fila.

-- Sin cursor

```
DELIMITER |
CREATE FUNCTION importe_facturado_mes(mes INT) RETURNS DECIMAL(10,2) DETERMINISTIC
BEGIN
  DECLARE total_importe DECIMAL(10,2);
  SELECT SUM(IFNULL((Unidades * PrecioReferencia), 0) + IFNULL((Horas *
actuaciones.Importe), 0))
  INTO total_importe
  FROM reparaciones
  LEFT JOIN Incluyen ON REPARACIONES.IdReparacion = Incluyen.IdReparacion
  LEFT JOIN recambios ON Incluyen.IdRecambio = recambios.IdRecambio
  LEFT JOIN Realizan ON REPARACIONES.IdReparacion = Realizan.IdReparacion
  LEFT JOIN actuaciones ON Realizan.Referencia = actuaciones.Referencia
  WHERE MONTH(reparaciones.FechaEntrada) = mes;
  RETURN total_importe;
END |
DELIMITER ;

select importe_facturado_mes(1);

-- Con cursor

DELIMITER |
CREATE FUNCTION importeFacturadoMesCursor(mes INT)
RETURNS DECIMAL(10,2) DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2) DEFAULT 0.00;
    DECLARE idRep INT;
    DECLARE horas DECIMAL(4,2);
    DECLARE precioAct DECIMAL(6,2);
    DECLARE unidades SMALLINT;
    DECLARE precioRec DECIMAL(6,2);
    DECLARE done INT DEFAULT FALSE;

    DECLARE cur CURSOR FOR
    SELECT IdReparacion FROM FACTURAS WHERE MONTH(FechaFactura) = mes;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    loop_rep: LOOP
        FETCH cur INTO idRep;
        IF done THEN
            LEAVE loop_rep;
        END IF;

        SELECT SUM(Horas*Importe) INTO precioAct FROM realizan JOIN actuaciones ON
realizan.Referencia = actuaciones.Referencia WHERE realizan.IdReparacion = idRep;
```

```
        SELECT SUM(Unidades*PrecioReferencia) INTO precioRec FROM incluyen JOIN recambios
ON incluyen.IdRecambio = recambios.IdRecambio WHERE incluyen.IdReparacion = idRep;

        SELECT SUM(Horas*precioAct) INTO horas FROM realizan WHERE realizan.IdReparacion
= idRep;

        SET total = total + precioRec + horas;
    END LOOP;
    CLOSE cur;

    RETURN total;
END |
DELIMITER ;

 select importeFacturadoMesCursor(3);
```

EJERCICIO 8.

Crea un trigger que, antes de insertar una fila en la tabla Incluyen, compruebe si existen unidades en Stock en la tabla RECAMBIOS llevando a cabo las siguientes acciones:

- Si hay suficientes unidades actualiza el Stock restando las unidades que se van a insertar.
- Si no hay suficientes unidades en Stock cancela la inserción de las unidades.

```
DELIMITER |
CREATE TRIGGER Exercicio8
BEFORE INSERT ON incluyen
FOR EACH ROW
BEGIN
 DECLARE cantidad INT;
 DECLARE msg VARCHAR(255);
 SELECT stock INTO cantidad FROM recambios WHERE IdRecambio=NEW.IdRecambio;
  IF cantidad>=NEW.Unidades THEN
    UPDATE recambios
       SET Stock=Stock-NEW.Unidades WHERE IdRecambio=NEW.IdRecambio;
  ELSEIF cantidad<NEW.Unidades THEN
       SET msg='Non hai stock!';
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
  END IF;
END |
DELIMITER ;
```

EJERCICIO 9.

Crea una tabla denominada PedidoRecambios que contenga 3 columnas:

- IdRecambio.
- Descripcion.
- Stock.

Con los mismos tipos de datos que tienen esas columnas en la tabla RECAMBIOS.

Crea un trigger asociado a la tabla RECAMBIOS que después de actualizar el Stock de un recambio, si el número de unidades en Stock del recambio modificado es inferior a 4 unidades, inserte una fila en la tabla PedidoRecambios con los datos resultantes.

```
DELIMITER |
CREATE TRIGGER Exercicio9 AFTER UPDATE ON recambios FOR EACH ROW

BEGIN

DECLARE cantidad INT;
DECLARE idr VARCHAR(10);
DECLARE descr VARCHAR(200);

SELECT stock INTO cantidad
FROM recambios
WHERE IdRecambio=NEW.IdRecambio;

       IF cantidad < 4 THEN
         SELECT IdRecambio INTO idr FROM recambios WHERE IdRecambio=NEW.IdRecambio;
         SELECT Descripcion INTO descr FROM recambos WHERE IdRecambio=NEW.IdRecambio;
         INSERT INTO PedidoRecambios(IdRecambio,Descripcion,Stock)
VALUES(idr,descr,cantidad);
       END IF;

END |

DELIMITER ;
```

EJERCICIO 10.

Utilizando funciones de librerías disponibles en MySQL obtener:
a) Un listado con dos columnas:

- en la primera, en mayúsculas apellidos y nombre de todos los clientes (entre los apellidos y el nombre incluir una coma como separador)
- y en la segunda, la ciudad en la que cada cliente tiene su domicilio (únicamente la ciudad, no la dirección).

b) Un listado con 2 columnas:

- en la primera la fecha de alta de los empleados con el formato dd/mm/aaaa
- y en la segunda aparecerá 'Contrato temporal ' para aquellos empleados que lleven contratados en el taller menos de 2 años, y 'Contrato fijo' para el resto.

a)

```
DELIMITER |
CREATE PROCEDURE Exercicio10a()
BEGIN
 SELECT CONCAT(UPPER(C.Apellidos),',',C.Nombre) AS 'Apelidos,Nome',
        SUBSTRING_INDEX(C.Direccion,',','-1') AS 'Cidade'
 FROM Clientes C;
END |
DELIMITER ;

CALL Exercicio10a();
```

b)

```
DELIMITER |
CREATE PROCEDURE Exercicio10b()
BEGIN
 SELECT DATE_FORMAT(FechaAlta,'%d/%m/%Y') AS 'Data',
 IF (DATEDIFF(DATE_FORMAT(FechaAlta,'%d/%m/%Y'),CURDATE())<730,'Contrato
temporal','Contrato fixo') AS 'Contrato'
 FROM empleados e;
END |
DELIMITER ;

CALL Exercicio10b();
```