Coder House

Curso – SQL

Proyecto: Bases de datos de un posible Lubricentro con la información del cliente, vehículo, empleados, sucursal, chequeos y metodología de pago

Profesor: Miguel Rodas

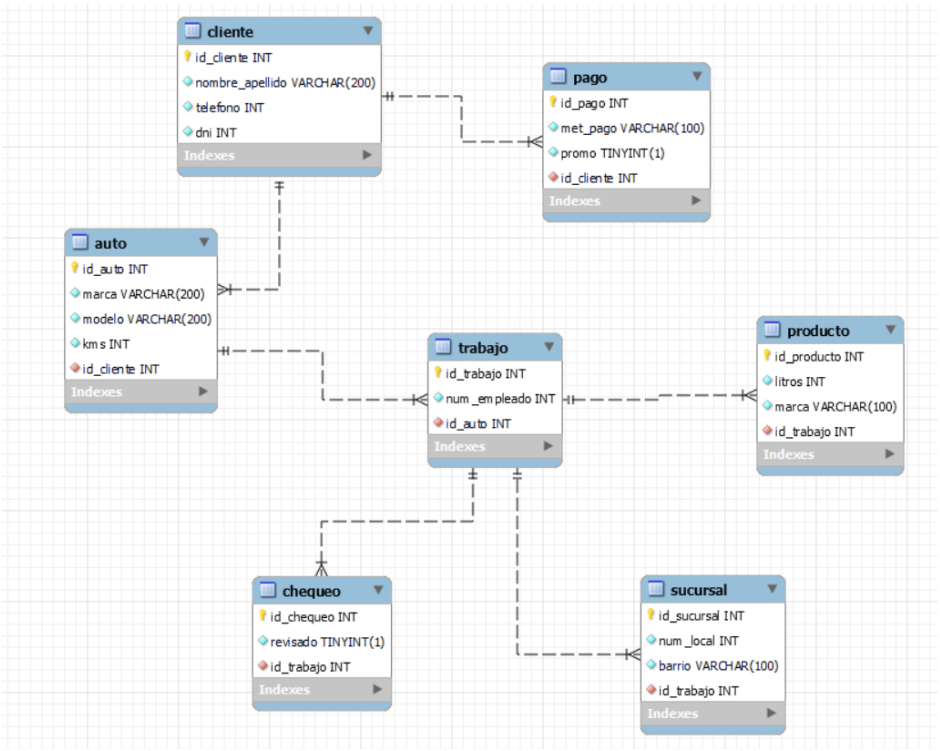
Alumno: Bruno Donato

Tutor: Natalia Arzubi

Descripción.

Un lubricentro es donde se realiza el cambio de aceite de un automóvil. En este caso estamos generando una base de datos con los detalles mas importantes que debemos conservar luego de realizar un trabajo de esta índole. Tendremos; detalles de los clientes que realicen el trabajo, así como los del vehículo (marca, modelo, etc). Incluiremos también tablas donde se reflejarán los datos de la empresa, que empleado realizo el trabajo, en cual sucursal, cual fue el método de pago, que productos se utilizaron y en que cantidades.

Diagrama ER



Detalle de tablas

Tabla	Nombre Abreviado	Nombre Completo	Tipo de Datos
Cliente	id_cliente - (PK)	id_cliente	INT AUTOINCREMENT
	nombre_apellido	nombre_apellido	VARCHAR
	dni	dni	INT
Auto	id_auto - (PK)	id_auto	INT AUTOINCREMENT
	marca	marca	VARCHAR
	modelo	modelo	VARCHAR
	kms	kilometros	INT
	id_cliente (FK)	(FK) Cliente	INT
Trabajo	id_trabajo - (PK)	id_trabajo	INT AUTOINCREMENT
	num_empleado	Numero de Empleado	INT
	id_auto (FK)	FK - Auto	INT
Producto	id_producto - (PK)	id_producto -	INT AUTOINCREMENT
	litros	litros	INT
	marca	marca	VARCHAR
	id_trabajo (FK)	FK - Trabajo	INT
Chequeo	id_chequeo - (PK)	id_chequeo	INT AUTOINCREMENT
	revisado	revisado	BOOL
	id_trabajo (FK)	FK - Trabajo	INT
Sucursal	id_sucursal	id_sucursal	INT AUTOINCREMENT
	num_local	Numero Local	INT
	barrio	barrio	VARCHAR
	id_trabajo (FK)	FK - Trabajo	INT
Pago	id_pago	id_pago	INT AUTOINCREMENT
	met_pago	Metodo de Pago	VARCHAR
	promo	Descuento 20%	BOOL
	id_cliente (FK)	(FK) Cliente	INT

Vistas

```
CREATE VIEW duenio_auto AS
SELECT cliente.id_cliente, cliente.nombre_apellido, auto.marca, auto.modelo
FROM cliente, auto
WHERE cliente.id_cliente = auto.id_cliente;
```

-Una vista que muestra el nombre del cliente junto con su auto (marca, modelo)

```sql
CREATE VIEW trabajo_empleado AS
SELECT auto.id_auto, auto.marca, auto.modelo, trabajo.num_empleado
FROM auto, trabajo
WHERE auto.id_auto = trabajo.id_auto;
```

-Muestra el auto (marca, modelo) junto con el empleado que realizó el trabajo

```sql
CREATE VIEW producto_util AS
SELECT producto.id_producto, producto.marca, producto.litros, trabajo.num_empleado
FROM producto, trabajo
WHERE producto.id_producto = trabajo.id_auto;
```

-Muestra la marca y litros del producto utilizado, junto con el empleado que realizó el trabajo

```sql
CREATE VIEW tbj_comp AS
SELECT trabajo.id_trabajo, trabajo.num_empleado, chequeo.revisado, sucursal.barrio
FROM trabajo, chequeo, sucursal
WHERE trabajo.id_trabajo = chequeo.id_trabajo = sucursal.id_trabajo;
```

-Muestra el empleado que realizó el trabajo, si el auto fue chequeado o no y la sucursal donde se llevó a cabo el servicio

```sql
CREATE VIEW completo AS
SELECT trabajo.id_trabajo, trabajo.num_empleado, producto.marca, producto.litros, chequeo.revisado
FROM trabajo, producto, chequeo
WHERE trabajo.id_trabajo = chequeo.id_trabajo = producto.id_trabajo;
```

-Muestra el producto (marca, litros) junto con el empleado que lo realizó y si este fue chequeado

Funciones

```
CREATE FUNCTION `auto_cliente`(par_id_cliente INT)
RETURNS char(255)
READS SQL DATA
BEGIN
        declare resultado char (255);
    set resultado = (select modelo from auto where id_cliente = par_id_cliente);
RETURN resultado;
```

-Devuelve el vehículo (modelo) del numero de cliente que se ingrese

```
CREATE FUNCTION `revision`(id_auto_p INT)
RETURNS char(255)
READS SQL DATA
BEGIN
        declare var1 int;
    declare var2 int;
    set var1 = (select id_trabajo from trabajo where id_auto_p = id_auto);
    set var2 = (select revisado from chequeo where var1 = id_trabajo);
    if (var2) > 0 then
                return ('El auto fue chequeado');
        else
                return ('El auto no fue chequeado');
        end if;
```

-Ingresando el id del auto, devuelve un mensaje en función de si el vehículo ya fue revisado o no

Stored Procedures

```
CREATE PROCEDURE `order_by` (in var char(50), in var2 int)
BEGIN
        if var2 = 0 then
                set @orden = concat ('order by ', var, ' desc');
    else
                set @orden = concat ('order by ', var, ' asc');
    end if;
                set @final = concat ( 'select * from cliente ', @orden);
    prepare runSQL from @final;
    execute runSQL;
    deallocate prepare runSQL;
```

-ingresando un campo de la tabla ‘cliente’ + el numero ‘1’ ordenará de manera ascendente. En caso de ingresar ‘0’ ordenará de manera descendente

```
CREATE PROCEDURE `sp_insert`(in nom char(200), in tel int, in p_dni int)
BEGIN
	insert into cliente ( nombre_apellido, telefono, dni ) values (nom, tel, p_dni);
```

-Inserta los valores ‘nombre’, ‘telefono’ y ‘dni’ en la tabla ‘cliente’

Triggers

```
CREATE TRIGGER AFT_INS_auto
AFTER INSERT ON auto
FOR EACH ROW
INSERT INTO insert_auto
VALUES (NEW.id_auto, NEW.marca, NEW.modelo, NEW.kms, NEW.id_cliente);
```

-Copia todos los datos insertados en la tabla ‘auto’ en una segunda tabla ‘insert_auto’

```
CREATE TRIGGER BEF_DEL_CLIENTE_LOGS
BEFORE DELETE ON cliente
FOR EACH ROW
INSERT INTO logs
VALUES (CURDATE(), CURTIME(), USER());
```

-Almacena en la tabla ‘logs’ la fecha, hora y usuario que elimina un campo de la tabla ‘cliente’

Herramientas y tecnología utilizada

En este caso se utilizó:

- MySQL Worckbench 8.0 CE

- Github

- Visual Studio Code

- Google Colaboratory