Administración negocio Canino

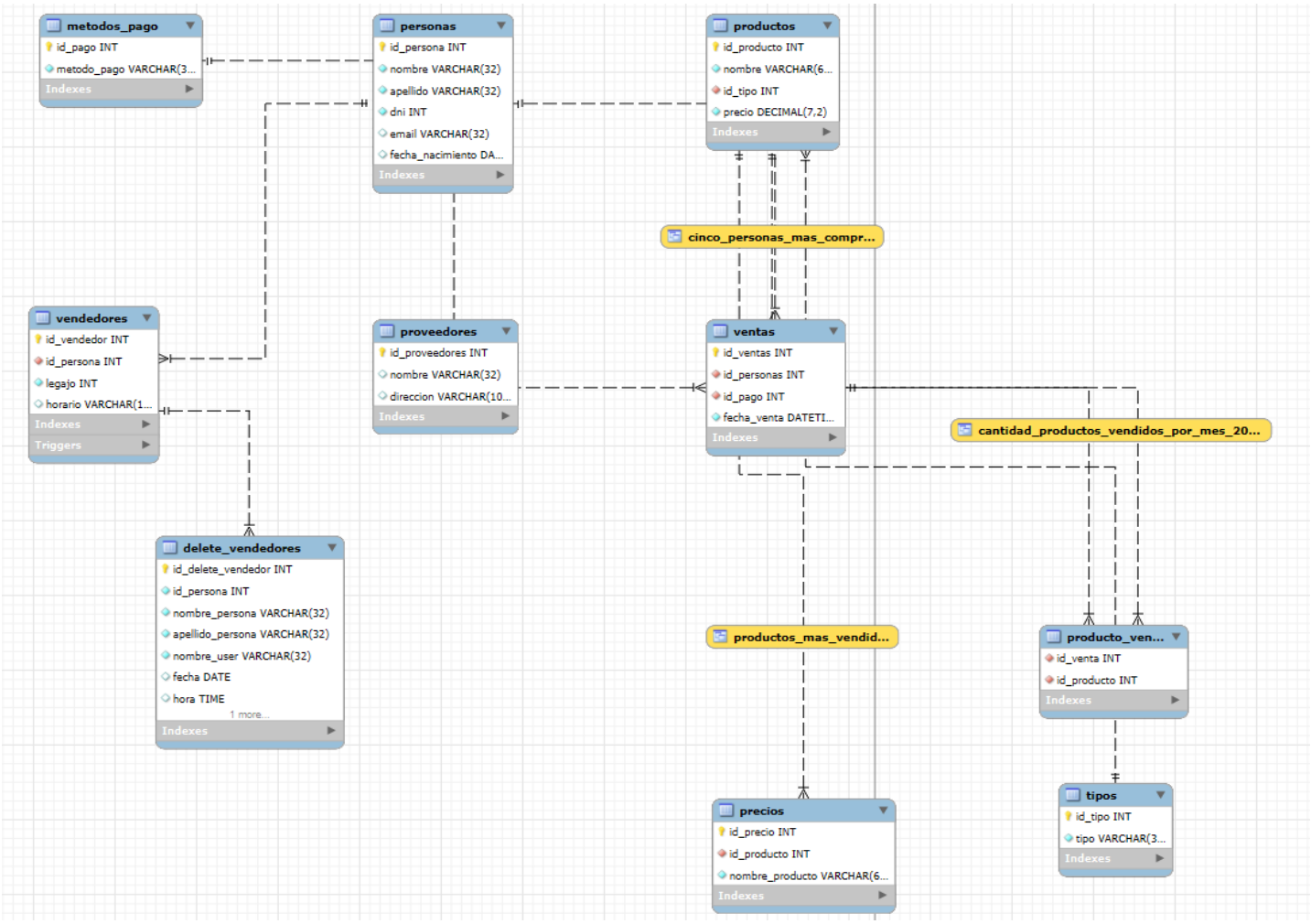
(pet shop)

- Primer entrega SQL – 9/24
- Alumno: Matias Monardez
- Comisión – 59415
- Proyecto: Administración de negocio canino.
- Curso: SQL
- Link:

El contexto

- Abrir primer negocio y llevar a cabo una organización.
- Tanto financieros y administrativos.
- Implementar una base de datos cuya función sea útil para llevar a cabo mi primer negocio.

Diagrama Entidad – Relación



Personas: esta tabla contiene datos personales.

Tabla : Personas	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_personas	int		true			true	id personas
nombre	varchar	32			true		nombre de la persona
apellido	varchar	32			true		apellido de la persona
dni	int				true		numero de documento
email	varchar	32			true		correo
fecha de nacimiento	date						fecha de nacimiento

Productos: esta tabla contiene nombre del producto y el tipo de productos que se venden.

Tabla : Productos	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_productos	int					true	id personas
nombre	varchar	64					nombre del producto
tipo	int						tipo de producto
precio	decimal	7 , 2					precio del producto

Ventas: esta tabla contenderá las ventas realizadas.

Tabla : Ventas	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_ventas	int					true	id ventas
id_personas	int						personas
id_pago	int						tipo de pago
fecha_venta	datetime						fecha de la venta

Proveedores: Distribuidoras a quienes le compro la mercadería.

Tabla : Proveedores	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_proveedores	int					true	id proveedor
nombre	varchar	32					nombre del proveedor
direccion	varchar	100					direccion del proveedor

Vendedores: Se guardan los datos de los vendedores.

Tabla : Vendedores	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_vendedor	int						id vendedor
id_persona	int						id de la persona
legajo	int						legajo
horario	varchar	16					horario de trabajo

Tipos: Tipos de productos (textil, higiene)

Tabla : Tipos	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_tipo	int						id primaria del tipo de producto
tipo	varchar	32					tipo de producto

Productos por venta: Productos que pertenecen a cada venta realizada.

Tabla : Producto por venta	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_venta	int						id de la venta realizada
id_producto	int						id del producto para venta realizada

Método de pago: Tabla de las formas de pagos (debito, crédito, efectivo).

Tabla : Metodos de pago	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_pago	int						id primaria de los tipos de pagos
metodo_de_pago	varchar	32					formas de pago

Delete vendedor: Se registran los vendedores eliminados.

Tabla : Delete vendedores	tipo de dato	Len	Primary key	foreign key	Not Null	Autoincrementar	Notas
id_delete_vendedor	int						id primaria vendedores eliminados
id_persona	int						id foránea de cada persona
nombre_persona	varchar	32					nombre de vendedor eliminado
apellido_persona	varchar	32					apellido de vendedor eliminado
nombre_user	varchar	32					nombre usuario que realizado la acción
fecha	date						fecha que se realizó la acción
hora	time						hora en que se realizó la acción

Listado de vistas:

Vista: Productos más vendidos

Objetivo: muestra los nombres de los 5 productos más vendidos.

```
-- vista que muestra de 5 productos mas vendidos.
CREATE OR REPLACE VIEW productos_mas_vendidos AS
    (SELECT nombre, COUNT(nombre)
    FROM productos p JOIN producto_venta pv ON (p.id_producto = pv.id_producto)
    GROUP BY nombre ORDER BY COUNT(nombre) DESC
    LIMIT 5);

    select * from productos_mas_vendidos;
```

Vista: que muestra nombre y apellido, e email de aquellas 5 personas que compraron más productos.

```sql
CREATE OR REPLACE VIEW cinco_personas_mas_compras AS
SELECT
    p.nombre,
    p.apellido,
    p.email,
    COUNT(v.id_ventas) AS total_compras
FROM
    ventas v
JOIN
    personas p ON v.id_personas = p.id_persona
GROUP BY
    p.id_persona
ORDER BY
    total_compras DESC
LIMIT 5;

select * from cinco_personas_mas_compras;
```

Vista: que muestra la cantidad de productos vendidos por mes, en un año específico, en este caso el año 2021.

```sql
CREATE OR REPLACE VIEW cantidad_productos_vendidos_por_mes_2021 AS
SELECT
    MONTH(v.fecha_venta) AS Mes,
    COUNT(pv.id_producto) AS CantidadVendida
FROM
    ventas v
JOIN
    producto_venta pv ON v.id_ventas = pv.id_venta
WHERE
    YEAR(v.fecha_venta) = 2021
GROUP BY
    MONTH(v.fecha_venta)
ORDER BY
    Mes;

    select * from cantidad_productos_vendidos_por_mes_2021;
```

Listado de Funciones

Función que permite conocer cuantas ventas realizo por cada año cada vendedor.

```
DELIMITER $$

DROP FUNCTION IF EXISTS `ventas_por_año`$$
CREATE FUNCTION `ventas_por_año`(vendedor INT, año INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE cant_ventas INT;

    -- Contar las ventas del vendedor en el año especificado
    SELECT COUNT(v.id_ventas) INTO cant_ventas
    FROM ventas v
    JOIN vendedores ve ON v.id_personas = ve.id_vendedor
    WHERE ve.id_vendedor = vendedor AND YEAR(v.fecha_venta) = año;

    RETURN cant_ventas;
END$$

SELECT ventas_por_año(1, 2021) AS total_ventas;
```

Listado de Store Procedures.

Ingresar nuevo vendedor:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `ingresar_nuevo_vendedor`$$
CREATE PROCEDURE `ingresar_nuevo_vendedor`(
    IN nombre_param VARCHAR(32),
    IN apellido_param VARCHAR(32),
    IN dni_param INT,
    IN email_param VARCHAR(32),
    IN fecha_nacimiento_param DATE,
    IN legajo_param INT,
    IN horario_param VARCHAR(16)
)
BEGIN
    DECLARE id_ultima_persona INT;
    DECLARE email_new VARCHAR(32);
    DECLARE fecha_nacimiento_new DATE;
    DECLARE horario_new VARCHAR(16);

    IF (nombre_param <> '') AND (apellido_param <> '') THEN
        -- Manejo del email
        IF email_param = '' THEN
            SET email_new = NULL;
        ELSE
            SET email_new = email_param;
        END IF;

        IF fecha_nacimiento_param IS NULL THEN
            SET fecha_nacimiento_new = NULL;
        ELSE
            SET fecha_nacimiento_new = fecha_nacimiento_param;
        END IF;

        -- Manejo del horario
        IF horario_param = '' OR horario_param <> 'mañana' THEN
            SET horario_new = 'tarde';
        ELSE
            SET horario_new = horario_param;
        END IF;

        -- Insertar en la tabla personas
        INSERT INTO personas (id_persona, nombre, apellido, dni, email, fecha_nacimiento) VALUES
            (NULL, nombre_param, apellido_param, dni_param, email_new, fecha_nacimiento_new);
        SET id_ultima_persona = LAST_INSERT_ID();

        -- Insertar en la tabla vendedores
        INSERT INTO vendedores (id_vendedor, id_persona, legajo, horario) VALUES
            (NULL, id_ultima_persona, legajo_param, horario_new);
    END IF;
END$$
```

Listado de Triggers:

Trigger que registra aquellos vendedores que fueron eliminados y quedan almacenados en la tabla delete_vendedores.

```
DROP TRIGGER IF EXISTS `tr_delete_vendedores`$$
CREATE TRIGGER `tr_delete_vendedores` AFTER DELETE ON `vendedores` FOR EACH ROW
BEGIN
    DECLARE nomb_persona VARCHAR(32);
    DECLARE ape_persona VARCHAR(32);
    DECLARE nomb_user VARCHAR(16);

    SET nomb_persona = (SELECT nombre FROM personas as p WHERE OLD.id_persona = p.id_persona);
    SET ape_persona = (SELECT apellido FROM personas as p WHERE OLD.id_persona = p.id_persona);
    SET nomb_user = (SELECT USER());

    INSERT INTO `delete_vendedores` (id_persona, nombre_persona, apellido_persona, nombre_user, fecha, hora)
    VALUES (OLD.id_persona, nomb_persona, ape_persona, nomb_user, CURRENT_DATE(), CURRENT_TIME());
END$$
DELIMITER ;
select * from delete_vendedores;

DELETE FROM vendedores
WHERE id_vendedor = 2;
```