Descripción de la temática de la base de datos

La base de datos contendrá la información relacionada con Aeromexico, una empresa de aviación, la cual se dedica a transportar pasajeros desde diferentes partes de México a destinos nacionales e internacionales.

El propósito de construir una base de datos es:

- Tener la información resguardada
- Trabajar con un gran volumen de datos
- Hacer consultas rápidas
- Presentar información a los stakeholders
- Trabajar con los datos para posteriormente darles una visualización

Nuestro diagrama de Entidad-Relación se compone de la siguiente manera:

Tabla de hechos

PK	IDCLIENTE	Contiene la información del cliente
FK	IDPASAJERO	Contiene la información del pasajero
FK	IDTICKET	Contiene la información del ticket
FK	IDVUELO	Contiene la información del vuelo
FK	IDEMPLEADO	Contiene la información del empleado
FK	IDCARRY	Contiene la información de la maleta

Tablas de dimensiones

Tabla de pasajeros

PK	IDPASAJERO	Contiene la información del pasajero
	Nombre	Nombre del cliente
	Apellido	Apellido del cliente
	Correo	Correo del cliente
	Telefono	Telefono del cliente
	Tipo_de_Cliente	A que categoria pertenece (Aeromexico Rewards, Aeromexico AMEX, Aerormexico Santander)

Tabla de ticket

PK	IDTICKET	Contiene la informacion del ticket
	Costo	Costo del boleto
	Asiento	Numero de Asiento
	Puerta de abordarje	Puerta de embarque

Tabla de vuelo

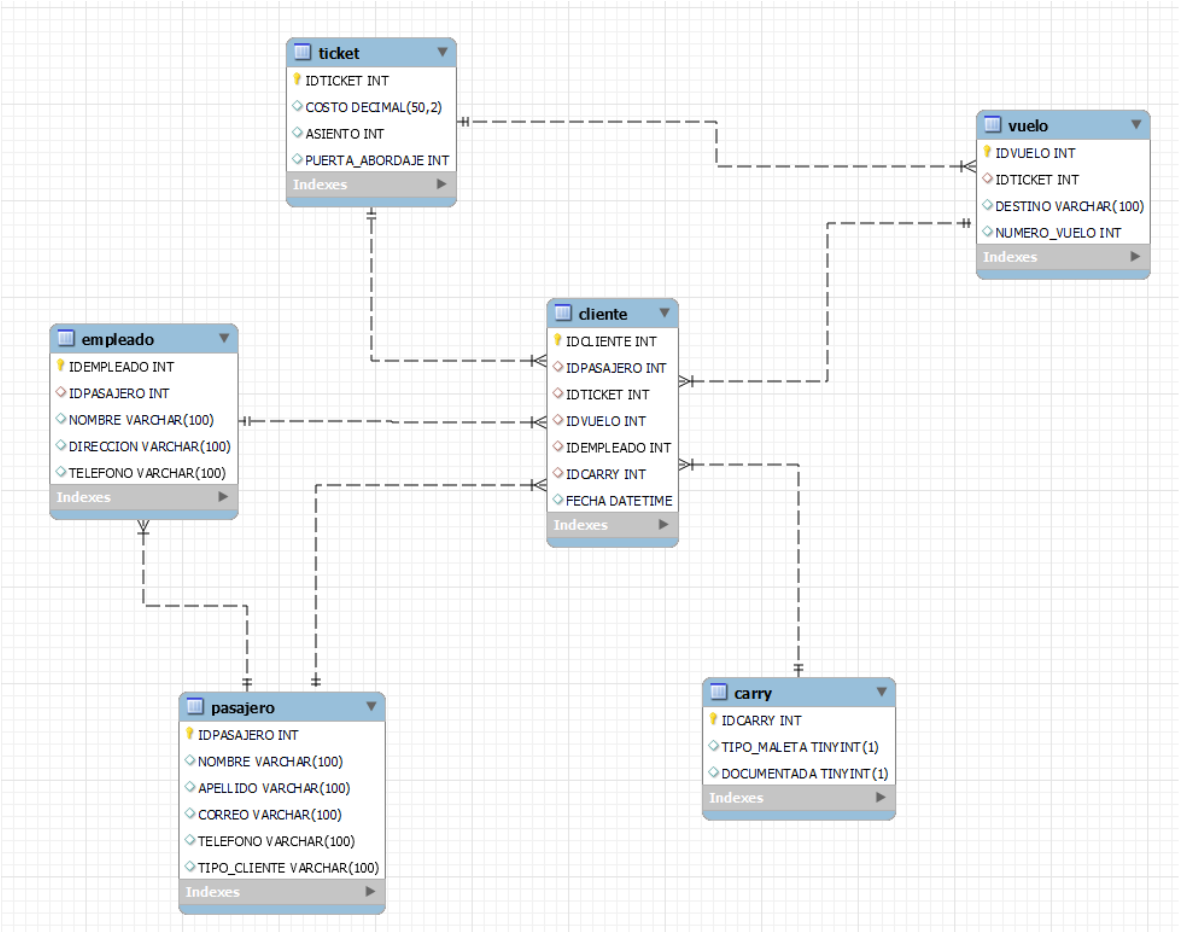
PK	IDVUELO	Contiene la información del vuelo
FK	IDTICKET	Contiene la informacion del ticket
	Destino	Destino del vuelo
	Numero de vuelo	Numero de vuelo

Tabla de empleado

PK	IDEMPLEADO	Contiene la información del empleado
FK	IDPASAJERO	Contiene la información del pasajero
	Nombre	
	Apellido	
	Telefono	

Tabla de carry

PK	IDCARRY	Contiene la información de la maleta
	Tipo de maleta	¿Tiene las medidas estandars?
	Documentada	¿Documento la maleta?



Segunda entrega proyecto SQL CoderHouse

Vistas

Se crearon dos vistas, a continuación se explican cada una de ellas.

1. La primera vista nos mostrara la cantidad de pasajeros registrados de acuerdo a su tipo de cliente

```
-- Vista para mostrar la cantidad de pasajeros que estan registrados de acuerdo a su tipo

CREATE VIEW
    Registro_Pasajeros AS
SELECT
    TIPO_CLIENTE,
    COUNT(IDPASAJERO) AS Cantidad_Pasajeros
FROM
    PASAJERO
GROUP BY
    TIPO_CLIENTE;
```

2. La segunda vista nos mostrara los ingresos obtenidos por cada vuelo

```
-- Vista para mostrar los ingresos que se tuvieron por cada vuelo

CREATE VIEW
    IngresosxVuelo AS
SELECT
    SUM(TICKET.COSTO) AS Ingresos
FROM
    VUELO
    LEFT JOIN TICKET ON TICKET.IDTICKET = VUELO.IDTICKET
GROUP BY
    TICKET.IDTICKET;
```

Procedures

Se creo un procedimiento, el cual nos verificara si el vuelo existe e ingresara el nombre del pasajero

```sql
-- procedimiento para verificar si el vuelo existe para insertar el nombre del pasajero

DELIMITER //

CREATE PROCEDURE crear_pasajero(
    IN p_nombre VARCHAR(100),
    IN p_telefono VARCHAR(20),
    IN p_correo VARCHAR(100),
    IN p_id_pasajero INT
)
BEGIN
    DECLARE pasajero_count INT;

    -- Verificar si el restaurante existe
    SELECT COUNT(*) INTO pasajero_count
    FROM VUELO
    WHERE IDVUELO = p_id_pasajero;

    -- Si el restaurante existe, insertar el empleado
    IF pasajero_count > 0 THEN
        INSERT INTO PASAJERO (NOMBRE, TELEFONO, CORREO, IDPASAJERO)
        VALUES (p_nombre, p_telefono, p_correo, p_id_restaurante);

        SELECT 'Pasajero creado exitosamente.';
    ELSE
        SELECT 'No existe el vuelo';
    END IF;
END //

DELIMITER ;
```

Funciones

Se crearon tres funciones.

La primera función nos ayudara a saber si un ticket fue cancelado por el usuario.

```sql
-- Funcion para saber si un ticket fue cancelado por el usuario

DELIMITER //

CREATE FUNCTION ticket_cancel(ticket_cancelado INT) RETURNS BOOLEAN
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE cancelacion_date DATETIME;
    DECLARE is_cancelada BOOLEAN;

    SELECT CANCELACION INTO cancelacion_date
        FROM TICKET
        WHERE IDTICKET = ticket_cancelado
        AND CANCELACION IS NOT NULL
        LIMIT 1;

    IF ticket_cancel IS NOT NULL THEN
        SET is_cancelada = TRUE;
    ELSE
        SET is_cancelada = FALSE;
    END IF;

    RETURN is_cancelada;
END //
```

La segunda función será para saber si un empleado vendio un ticket

```sql
-- funcion para saber si un empleado vendio un ticket

DELIMITER //

CREATE FUNCTION ticket_vendido(ticket_sell INT) RETURNS BOOLEAN
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE ticket_date DATETIME;
    DECLARE is_sell BOOLEAN;

    SELECT VENDIDO INTO ticket_date
        FROM TICKET
        WHERE IDTICKET = ticket_sell
        AND VENDIDO IS NOT NULL
        LIMIT 1;

    IF ticket_date IS NOT NULL THEN
        SET is_sell = TRUE;
    ELSE
        SET is_sell = FALSE;
    END IF;

    RETURN is_sell;
END //

DELIMITER ;
```

Y la ultima función, nos dará la cantidad de tickets que vendio un empleado

```sql
-- funcion para saber cuantos tickets vendio un empleado

DELIMITER //

CREATE FUNCTION cantidad_tickets(cantidad_empleado_id INT) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE ticket_count INT;

    SELECT COUNT(*) INTO ticket_count
    FROM TICKET
    WHERE IDTICKET = cantidad_empleado_id;

    RETURN ticket_count;
END //

DELIMITER ;
```

Triggers

Se crearon tres triggers.

El primer trigger nos guardara los datos si el cliente cancela su vuelo

```sql
-- TRIGGER para guardar los datos si el cliente cancela su vuelo

CREATE TABLE
    tabla_cambios (
        id_cambio          INT NOT NULL AUTO_INCREMENT PRIMARY KEY
    ,   tabla_afectada  VARCHAR (50)
    ,   accion          VARCHAR(50)
    ,   fecha           DATETIME
    ,   idcliente       INT
    ,   usuario         VARCHAR(50)
    );

DELIMITER //

CREATE TRIGGER after_insert_trigger
AFTER INSERT ON CLIENTE
FOR EACH ROW
BEGIN
    INSERT INTO tabla_cambios  (tabla_afectada, accion, fecha,idcliente,usuario)
    VALUES ('CLIENTE', 'INSERT', NOW() , NEW.IDCLIENTE);
END //

DELIMITER ;
```

El segundo trigger verificara si el correo de un cliente no es repetidos.

```
DELIMITER //

CREATE TRIGGER before_insert_cliente_trigger
BEFORE INSERT ON CLIENTE
FOR EACH ROW
BEGIN
    DECLARE correo_count INT;

    SELECT COUNT(*) INTO correo_count
        FROM CLIENTE
    WHERE CORREO = NEW.CORREO;

    IF correo_count > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El correo electrónico ya está en uso.';
    END IF;

END //

DELIMITER ;
```

Y el ultimo trigger, verifica el teléfono del cliente

```
-- Trigger para verificar que el numero de telefono de un cliente no sean repetidos

DELIMITER //

CREATE TRIGGER before_insert_cliente_trigger_telefono
BEFORE INSERT ON CLIENTE
FOR EACH ROW
BEGIN
    DECLARE tel_count INT;

    SELECT COUNT(*) INTO tel_count
        FROM CLIENTE


    WHERE TELEFONO = NEW.TELEFONO;

    IF tel_count > 0 THEN
        SIGNAL SQLSTATE '46000' SET MESSAGE_TEXT = 'El telefono ya está en uso.';
    END IF;

END //

DELIMITER ;
```