Caso práctico de "Base de datos para aplicaciones" con SQLite3

Planteamiento del caso de estudio

- Una papelería desea llevar el control de las ventas de sus productos, por lo que desea crear una base de datos para almacenar los datos de las ventas, las compras, los clientes, los productos y los proveedores.
- La base de datos debe tener las siguientes tablas: clietes, productos, proveedores, ventas, compras, detalle_ventas, detalle_compras.
- La base de datos debe tener realaciones entre las tablas: ventas y detalle_ventas, compras y detalle_compras, clientes y ventas, proveedores y compras, productos y detalle_compras.
- La empresa desea procesar los datos almacenados para obener información de:
    - total de venta por cada venta.

Creación de Tablas

TABLA CLIENTES

Creación de la tabla **Clientes**

```
CREATE TABLE clientes (
    id_cliente integer PRIMARY KEY AUTOINCREMENT,
    nombre varchar(50),
    email varchar(50)
);
```

Insertar datos en la tabla **Clientes**

```
INSERT INTO clientes(nombre,email)
VALUES
('Dejah','dejah@email.com'),
('Jonh','jonh@email.com');
```

Consultar datos de la tabla **Clientes**

```
SELECT * FROM clientes;
```

Resultado de la consulta

```
id_cliente  nombre      email
----------  ----------  ---------------
1           Dejah       dejah@email.com
2           Jonh        jonh@email.com
```

TABLA PRODUCTOS

Creación de la tabla **Productos**

```
CREATE TABLE productos(
    id_producto integer PRIMARY KEY AUTOINCREMENT,
    producto varchar(50),
    precio_unitario float
);
```

Insertar datos en la tabla **Productos**

```
INSERT INTO productos(producto,precio_unitario)
VALUES
('Lápiz',5),
('Libreta',20);
```

Consulta de datos de la tabla **Productos**

```
SELECT * FROM productos;
```

Resultado de la consulta

```
id_producto  producto    precio_unitario
-----------  ----------  ---------------
1            Lápiz       5.0
2            Libreta     20.0
```

TABLA VENTAS

Creación de la tabla **Ventas**

```
CREATE TABLE ventas(
    id_venta integer PRIMARY KEY AUTOINCREMENT,
    fecha date,
    id_cliente integer REFERENCES clientes(id_cliente)
);
```

Insertar datos en la tabla **Ventas**

```
INSERT INTO ventas(fecha,id_cliente)
VALUES
('2020/01/01',1),
('2020/01/02',1),
('2020/01/03',2);
```

Consulta de datos de la tabla **Ventas**

```
SELECT * FROM ventas;
```

Resultado de la consulta

```
id_venta    fecha       id_cliente
----------  ----------  ----------
1           2020/01/01  1
2           2020/01/02  1
3           2020/01/03  2
```

TABLA DETALLES_VENTAS

Creación de la tabla **"Detalle ventas"**

```
CREATE TABLE detalle_ventas(
    id_detalle_venta integer PRIMARY KEY AUTOINCREMENT,
    id_venta integer REFERENCES ventas(id_venta),
    id_producto integer REFERENCES productos(id_producto),
    cantidad_producto integer,
    precio_unitario float,
    total_x_producto float
);
```

Insertar datos en la tabla **"Detalle ventas"**

```
INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
VALUES
(1,1,2,5,10),
(1,2,10,20,200),
(2,2,1,20,20),
(3,1,10,5,50),
(3,2,10,20,200);
```

Consulta de datos de la tabla **"Detalle ventas"**

```
SELECT * FROM detalle_ventas;
```

Resultado de la consulta

```
id_detalle_venta  id_venta    id_producto  cantidad_producto  precio_unitario
total_x_producto
----------------  ----------  -----------  -----------------  ---------------  ---
-------------
1                 1           1            2                  5.0
10.0
2                 1           2            10                 20.0
200.0
3                 2           2            1                  20.0
20.0
4                 3           1            10                 5.0
50.0
5                 3           2            10                 20.0
200.0
```

TABLA PROVEEDORES

Creación de la tabla **Proveedores**

```
CREATE TABLE proveedores (
    id_proveedor integer PRIMARY KEY AUTOINCREMENT,
    proveedor varchar(50),
    nombre_contacto varchar(100),
```

```
    email_contacto varchar(50)
);
```

Insertar datos en la tabla **Proveedores**

```
INSERT INTO proveedores(proveedor,nombre_contacto,email_contacto)
VALUES
('ACME','Bruce Wayne','bruce@acme.com'),
('Cloud9','Diana Prince','diana@cloud9.com');
```

Consulta de datos de la tabla **Proveedores**

```
SELECT * FROM proveedores;
```

Resultado de la consulta

```
id_proveedor  proveedor   nombre_contacto  email_contacto
------------  ----------  ---------------  --------------
1             ACME        Bruce Wayne      bruce@acme.com
2             Cloud9      Diana Prince     diana@scribe.c
```

TABLA COMPRAS

Creación de la tabla **Compras**

```
CREATE TABLE compras(
    id_compra integer PRIMARY KEY AUTOINCREMENT,
    fecha date,
    id_proveedor integer REFERENCES proveedores(id_proveedor)
);
```

Insertar datos en la tabla **Compras**

```
INSERT INTO compras(fecha,id_proveedor)
VALUES
('2020/01/01',1),
('2020/02/02',2),
('2020/03/03',2);
```

Consulta de datos de la tabla **Compras**

```
SELECT * FROM compras;
```

Resultado de la consulta

```
id_compra   fecha       id_proveedor
----------  ----------  ------------
1           2020/01/01  1
2           2020/02/02  2
3           2020/03/03  2
```

TABLA DETALLES_COMPRAS

Creación de la tabla **Detalle de Compras**

```
CREATE TABLE detalle_compras(
    id_detalle_compra integer PRIMARY KEY AUTOINCREMENT,
    id_compra integer REFERENCES compras(id_compra),
    id_producto integer REFERENCES productos(id_producto),
    cantidad_producto integer,
    precio_unitario float,
    total_x_producto float
);
```

Insertar datos en la tabla **Detalle de Compras**

```
INSERT INTO
detalle_compras(id_compra,id_producto,cantidad_producto,precio_unitario,total_x_pr
oducto)
VALUES
(1,1,100,5,500),
(1,2,150,20,3000),
(2,1,200,5,1000),
(2,2,250,20,5000),
(3,1,300,5,600);
```

Consulta de datos de la tabla **Detalle de Compras**

```
SELECT * FROM detalle_compras;
```

Resultado de la consulta

```
id_detalle_compra  id_compra   id_producto  cantidad_producto  precio_unitario
total_x_producto
-----------------  ----------  -----------  -----------------  ---------------  --
--------------
1                  1           1            100                5.0
500.0
2                  1           2            150                20.0
3000.0
3                  2           1            200                5.0
1000.0
4                  2           2            250                20.0
5000.0
5                  3           1            300                5.0
600.0
```

Consultas Ventas

Consulta 01:

Mostrar id_cliente, nombre, email, fecha, id_venta, id_producto, producto, cantidad_producto,precio_unitario,total_producto para cada detalle_venta

Script SQL

```
select clientes.id_cliente, clientes.nombre, clientes.email, ventas.fecha,
ventas.id_venta, productos.id_producto,
productos.producto,detalle_ventas.id_detalle_venta,
detalle_ventas.cantidad_producto,detalle_ventas.precio_unitario,detalle_ventas.tot
al_x_producto

from clientes,ventas,detalle_ventas, productos

WHERE clientes.id_cliente = ventas.id_cliente AND
ventas.id_venta = detalle_ventas.id_venta AND
detalle_ventas.id_producto = productos.id_producto;
```

Resultado del script

```
id_cliente  nombre      email            fecha       id_detalle_venta  id_venta
id_producto  producto    cantidad_producto  precio_unitario  total_x_producto
----------  ----------  ---------------  ----------  ----------------  ----------
----------  ----------  -----------------  ---------------  ----------------
```

```
1           Dejah       dejah@email.com  2020/01/01  1                1
1            Lápiz       2                  5.0             10.0
1           Dejah       dejah@email.com  2020/01/01  2                1
2            Libreta     10                 20.0            200.0
1           Dejah       dejah@email.com  2020/01/02  3                2
2            Libreta     1                  20.0            20.0
2           Jonh        jonh@email.com   2020/01/03  4                3
1            Lápiz       10                 5.0             50.0
2           Jonh        jonh@email.com   2020/01/03  5                3
2            Libreta     10                 20.0            200.0
```

Consulta 02:

Mostrar el total_venta por cada venta

Script SQL

```
select detalle_ventas.id_venta, sum(total_x_producto) as total_venta
FROM detalle_ventas
GROUP BY id_venta;
```

Resultado del script

```
id_venta    total_venta
----------  -----------
1           210.0
2           20.0
3           250.0
```

Consulta 03:

Mostrar el nombre del cliente y total_venta por cada venta

Script SQL

```
SELECT clientes.nombre, ventas.id_venta, sum(detalle_ventas.total_x_producto)  as
total_venta
FROM clientes,ventas,detalle_ventas
WHERE clientes.id_cliente=ventas.id_cliente
AND ventas.id_venta=detalle_ventas.id_venta
GROUP BY ventas.id_venta;
```

Resultado del script

```
nombre      id_venta    total_venta
----------  ----------  -----------
Dejah       1           210.0
Dejah       2           20.0
Jonh        3           250.0
```

Consulta 04:

Mostrar el nombre del cliente y el total que pagado

Script SQL

```
SELECT clientes.nombre, sum(detalle_ventas.total_x_producto) as total_venta
FROM clientes,ventas, detalle_ventas
WHERE clientes.id_cliente = ventas.id_cliente AND ventas.id_venta =
detalle_ventas.id_venta
GROUP BY  clientes.nombre;
```

Resultado del script

```
nombre      total_venta
----------  -----------
Dejah       230.0
Jonh        250.0
```

Consulta 05:

Mostrar la cantidad total de productos vendida por cada producto

Script SQL

```
SELECT productos.producto, sum(detalle_ventas.cantidad_producto) as
cantidad_producto
FROM productos,detalle_ventas
WHERE detalle_ventas.id_producto= productos.id_producto
GROUP BY productos.producto;
```

Resultado del script

```
producto    cantidad_producto
----------  -----------------
```

```
Libreta     21
Lápiz       12
```

Consulta 06:

Mostrar el total vendido por dia

Script SQL

```
SELECT ventas.fecha, sum(detalle_ventas.total_x_producto) as total_venta
FROM ventas, detalle_ventas
WHERE ventas.id_venta = detalle_ventas.id_venta
GROUP BY ventas.id_venta;
```

Resultado del script

```
fecha       total_venta
----------  -----------
2020/01/01  210.0
2020/01/02  20.0
2020/01/03  250.0
```

Consulta 07:

Mostrar el dia que menos se ha vendido

Script SQL

```
SELECT ventas.fecha, sum(detalle_ventas.total_x_producto) as total_venta
FROM ventas, detalle_ventas
WHERE ventas.id_venta = detalle_ventas.id_venta
GROUP BY ventas.id_venta ORDER BY total_venta LIMIT 1;
```

Resultado del script

```
fecha       total_venta
----------  -----------
2020/01/02  20.0
```

Consultas Compras

Consulta 08:

Mostrar id_proveedor, proveedor, nombre_contacto, email_contacto, fecha, id_compra, id_producto, producto, cantidad_producto,precio_unitario,total_producto para cada **detalle_compra**

Script SQL

```
SELECT proveedores.id_proveedor, proveedores.proveedor,
proveedores.nombre_contacto, proveedores.email_contacto, compras.fecha,
compras.id_compra, detalle_compras.id_producto, productos.producto,
detalle_compras.cantidad_producto,detalle_compras.precio_unitario,detalle_compras.
total_x_producto
FROM proveedores, compras, productos, detalle_compras
WHERE
proveedores.id_proveedor = compras.id_proveedor AND
compras.id_compra = detalle_compras.id_compra AND
detalle_compras.id_producto = productos.id_producto;
```

Resultado del script

```
id_proveedor  proveedor   nombre_contacto  email_contacto  fecha       id_compra
id_producto  producto       cantidad_producto  precio_unitario  total_x_producto
------------  ----------  ---------------  --------------  ----------  ----------
-----------  -------------  -----------------  ---------------  ----------------
1             ACME        Bruce Wayne      bruce@acme.com  2020/01/01  1
1            Lápiz acme 2H  100                5.0              500.0
1             ACME        Bruce Wayne      bruce@acme.com  2020/01/01  1
2            Libreta scrib  150                20.0             3000.0
2             Cloud9      Diana Prince     diana@scribe.c  2020/02/02  2
1            Lápiz acme 2H  200                5.0              1000.0
2             Cloud9      Diana Prince     diana@scribe.c  2020/02/02  2
2            Libreta scrib  250                20.0             5000.0
2             Cloud9      Diana Prince     diana@scribe.c  2020/03/03  3
1            Lápiz acme 2H  300                5.0              600.0
```

Consulta 09:

Mostrar el total_compra por cada **compra**

Script SQL

```
select detalle_compras.id_compra, sum(total_x_producto) as total_compra
FROM detalle_compras
GROUP BY id_compra;
```

Resultado del script

```
id_compra   total_compra
----------  ------------
1           3500.0
2           6000.0
3           600.0
```

Consulta 10:

Mostrar el proveedor, nombre_contacto, email_contacto y total_compra por cada compra

Script SQL

```
SELECT proveedores.proveedor, proveedores.nombre_contacto,
proveedores.email_contacto ,compras.id_compra,
sum(detalle_compras.total_x_producto) as total_compra
FROM proveedores,compras, detalle_compras
WHERE proveedores.id_proveedor= compras.id_proveedor AND compras.id_compra =
detalle_compras.id_compra
GROUP BY compras.id_compra;
```

Resultado del script

```
proveedor   nombre_contacto  email_contacto  id_compra   total_compra
----------  --------------  --------------  ----------  ------------
ACME        Bruce Wayne     bruce@acme.com  1           3500.0
Cloud9      Diana Prince    diana@scribe.c  2           6000.0
Cloud9      Diana Prince    diana@scribe.c  3           600.0
```

Consulta 11:

Mostrar el proveedor y el total que se le ha comprado

Script SQL

```
SELECT proveedores.proveedor, sum(detalle_compras.total_x_producto) as
total_compra
FROM proveedores,compras, detalle_compras
WHERE proveedores.id_proveedor= compras.id_proveedor AND compras.id_compra =
detalle_compras.id_compra
GROUP BY  proveedores.proveedor;
```

Resultado del script

```
proveedor   total_compra
----------  ------------
ACME        3500.0
Cloud9      6600.0
```

Consulta 12:

Mostrar la cantidad total de productos comprados por cada producto

Script SQL

```
SELECT productos.producto, sum(detalle_compras.cantidad_producto) as
cantidad_producto
FROM productos,detalle_compras
WHERE detalle_compras.id_producto= productos.id_producto
GROUP BY productos.producto;
```

Resultado del script

```
producto                    cantidad_producto
--------------------------  -----------------
Libreta scribe profesional  400
Lápiz acme 2H               600
```

Consulta 13:

Mostrar el total comprado por día

Script SQL

```
SELECT compras.fecha, sum(detalle_compras.total_x_producto) as total_compra
FROM compras, detalle_compras
WHERE compras.id_compra = detalle_compras.id_compra
GROUP BY compras.id_compra;
```

Resultado del script

```
fecha       total_compra
----------  ------------
```

```
2020/01/01  3500.0
2020/02/02  6000.0
2020/03/03  600.0
```

Consulta 14:

Mostrar el dia que más se ha comprado

Script SQL

```
SELECT compras.fecha, sum(detalle_compras.total_x_producto) as total_compra
FROM compras, detalle_compras
WHERE compras.id_compra = detalle_compras.id_compra
GROUP BY compras.id_compra ORDER BY total_compra DESC LIMIT 1;
```

Resultado del script

```
fecha       total_compra
----------  ------------
2020/02/02  6000.0
```

Consultas Triggers

Consulta 15:

Modificar mediante sql la estructura de la tabla **productos** e insertar el campo existencias de tipo entero y con un valor default de 100.

Tabla productos antes de modificar su estructura

Script SQL

```
SELECT * from productos;
```

Resultado del script

```
id_producto  producto       precio_unitario
-----------  -------------  ---------------
1            Lápiz acme 2H  5.0
2            Libreta scrib  20.0
```

Script SQL

```
ALTER TABLE productos
ADD existencias integer default 100;
```

Resultado del script

Tabla productos después de insertar existencias

```
id_producto  producto       precio_unitario  existencias
-----------  -------------  ---------------  -----------
1            Lápiz acme 2H  5.0              100
2            Libreta scrib  20.0             100
```

Consulta 16:

Crear un trigger que después de insertar un producto en detalle_ventas, dejando preci_unitario y total_x_producto con un valor de 0, actualice el precio_unitario del producto insertado trayendolo directmente de la tabla productos.

Ejemplo consulta SQL:

```
INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
VALUES (1,1,2,0,0);
```

Script SQL

```
CREATE TRIGGER actualizar_precio_unitario
   AFTER INSERT ON detalle_ventas
BEGIN
    UPDATE detalle_ventas
    SET precio_unitario = (SELECT precio_unitario FROM productos where
detalle_ventas.id_producto = productos.id_producto)
    where id_producto = new.id_producto;
END;

SELECT * FROM detalle_ventas;

INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
```

```
VALUES (1,1,2,0,0);

SELECT * FROM detalle_ventas;
```

Resultado del script

```
id_detalle_venta  id_venta    id_producto  cantidad_producto  precio_unitario
total_x_producto
----------------  ----------  -----------  -----------------  ---------------  ---
-------------
1                 1           1            2                  5.0
10.0
2                 1           2            10                 20.0
200.0
3                 2           2            1                  20.0
20.0
4                 3           1            10                 5.0
50.0
5                 3           2            10                 20.0
200.0
6                 1           1            2                  5.0              0.0
```

Consulta 17

Crear un trigger que después de insertar un producto en detalle_ventas, actualice las existencias de productos:

Script SQL

```
CREATE TRIGGER actualizar_inventario
   AFTER INSERT ON detalle_ventas
BEGIN
    UPDATE productos
    SET existencias = existencias - new.cantidad_producto
    where id_producto = new.id_producto;
END;

SELECT * FROM productos;

INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
VALUES (1,1,2,0,0);

SELECT * FROM productos;
```

Existencias de productos:

```
id_producto  producto       precio_unitario  existencias
-----------  -------------  ---------------  -----------
1            Lápiz acme 2H  5.0              100
2            Libreta scrib  20.0             100
```

Nuevo detalle_venta

```
INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
VALUES (1,1,2,0,0);
```

Existencias actualizadas

```
id_producto  producto       precio_unitario  existencias
-----------  -------------  ---------------  -----------
1            Lápiz acme 2H  5.0              98
2            Libreta scrib  20.0             100
```

Consulta 18

Crear un trigger que después de actualizar el detalle_ventas, actualice las total_x_prodcutos con la operación total_x_producto = cantidad_producto * precio_unitario:

```
CREATE TRIGGER actualizar_total_producto
   AFTER update ON detalle_ventas
BEGIN
    UPDATE detalle_ventas
    SET total_x_producto = cantidad_producto * precio_unitario
    WHERE id_detalle_venta = new.id_detalle_venta;
END;

SELECT * FROM detalle_ventas;

INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
VALUES (1,1,10,0,0);

SELECT * FROM detalle_ventas;
```

Detalle ventas antes del TRIGGER

```
id_detalle_venta  id_venta    id_producto  cantidad_producto  precio_unitario
total_x_producto
----------------  ----------  -----------  -----------------  ---------------  ---
-------------
1                 1           1            2                  5.0
10.0
2                 1           2            10                 20.0
200.0
3                 2           2            1                  20.0
20.0
4                 3           1            10                 5.0
50.0
5                 3           2            10                 20.0
200.0
6                 1           1            2                  5.0              0.0
7                 1           1            2                  5.0              0.0
```

Insertar nuevo detalle_ventas

```
INSERT INTO
detalle_ventas(id_venta,id_producto,cantidad_producto,precio_unitario,total_x_prod
ucto)
VALUES (1,1,10,0,0);
```

Tabla detalle_ventas actualizada con el trigger.

```
id_detalle_venta  id_venta    id_producto  cantidad_producto  precio_unitario
total_x_producto
----------------  ----------  -----------  -----------------  ---------------  ---
-------------
1                 1           1            2                  5.0
10.0
2                 1           2            10                 20.0
200.0
3                 2           2            1                  20.0
20.0
4                 3           1            10                 5.0
50.0
5                 3           2            10                 20.0
200.0
6                 1           1            2                  5.0
10.0
7                 1           1            2                  5.0
10.0
8                 1           1            10                 5.0
50.0
```

Transactions

COMMIT

```
CREATE TABLE clientes (
    id_cliente integer PRIMARY KEY AUTOINCREMENT,
    nombre varchar(50),
    email varchar(50)
);

INSERT INTO clientes(nombre,email)
VALUES
('Dejah','dejah@email.com'),
('Jonh','jonh@email.com');
```

SELECT * FROM clientes;

```
1|Dejah|dejah@email.com
2|Jonh|jonh@email.com
```

Transaction COMMIT

```
BEGIN TRANSACTION;

INSERT INTO clientes(nombre,email)
VALUES
('Jane','jane@email.com');

COMMIT;
```

SELECT * FROM clientes;

```
1|Dejah|dejah@email.com
2|Jonh|jonh@email.com
3|Jane|jane@email.com
```

Nota: COMMIT aplica la transaccion

ROLLBACK

```
CREATE TABLE clientes (
    id_cliente integer PRIMARY KEY AUTOINCREMENT,
    nombre varchar(50),
    email varchar(50)
);

INSERT INTO clientes(nombre,email)
VALUES
('Dejah','dejah@email.com'),
('Jonh','jonh@email.com');
```

SELECT * FROM clientes;

```
1|Dejah|dejah@email.com
2|Jonh|jonh@email.com
```

Transaction ROLLBACK

```
BEGIN TRANSACTION;

INSERT INTO clientes(nombre,email)
VALUES
('Jane','jonh@email.com');

ROLLBACK;
```

SELECT * FROM clientes;

```
1|Dejah|dejah@email.com
2|Jonh|jonh@email.com
```

Nota: ROLLBACK anula la transacion

PERMISSIONS

```
create database demo_users;

use demo_users;


create table clientes(
id int primary key auto_increment,
```

```
name varchar(50) not null
);

insert into clientes(name)
values
('Jane'),
('John');

select * from clientes;
```

Crear usuario

```
CREATE USER 'invitado'@'localhost' IDENTIFIED BY '123456';
```

Asignar permisos

```
GRANT ALL PRIVILEGES ON demo_users.* TO 'administrador'@'localhost';
```

Aplicar cambios

```
FLUSH PRIVILEGES;
```

Tipos de permisos

- **ALL** PRIVILEGES- asigna todos los permisos
- **CREATE** Permite crear nuevas bases o tablas.
- **DROP** Permite eliminar bases o tablas.
- **DELETE** Permite borrar bases o tablas.
- **INSERT** Permite insertar nuevos registros en las tablas.
- **SELECT** Permite seleccionar registros de las tablas.
- **UPDATE** Permite actualizar datos de los registros de las tablas.
- **GRANT OPTION** Permite dar o quitar permisos a los usuarios.

Quitar permisos

```
REVOKE Select ON demo_users.* TO 'invitado'@'localhost';
```

Mostrar lista de permisos

```
SHOW GRANTS FOR 'invitado'@'localhost';
```

Eliminar usuario

```
DROP USER 'invitado'@'localhost';
```