INFORME TÉCNICO

INSTRUCCIONES DEL LENGUAJE SQL (DML Y DDL)

CRISTIAN ANDRES FIGUEROA CASTROS

Objetivos:

- Aplicación y uso del conjunto de acciones necesarias para la manipulación de la base de datos, teniendo en cuenta los requerimientos y las instrucciones del lenguaje SQL.
- Ejecución y resultado de los comandos DDL y DML aplicados a la base de datos construida, para el caso de estudio propuesto en la actividad.

Estudio de caso Import Tech SAS.

Para esta empresa es necesario implementar una base de datos para el proceso de compra de sus productos por lo que tenemos que crear el modelo de relaciones entre todas sus tablas, teniendo tablas como clientes, teléfonos, direcciones, proveedores, ventas y demás tablas para su funcionamiento. Satisfaciendo esta necesidad se creo el siguiente modelo de bases de datos:

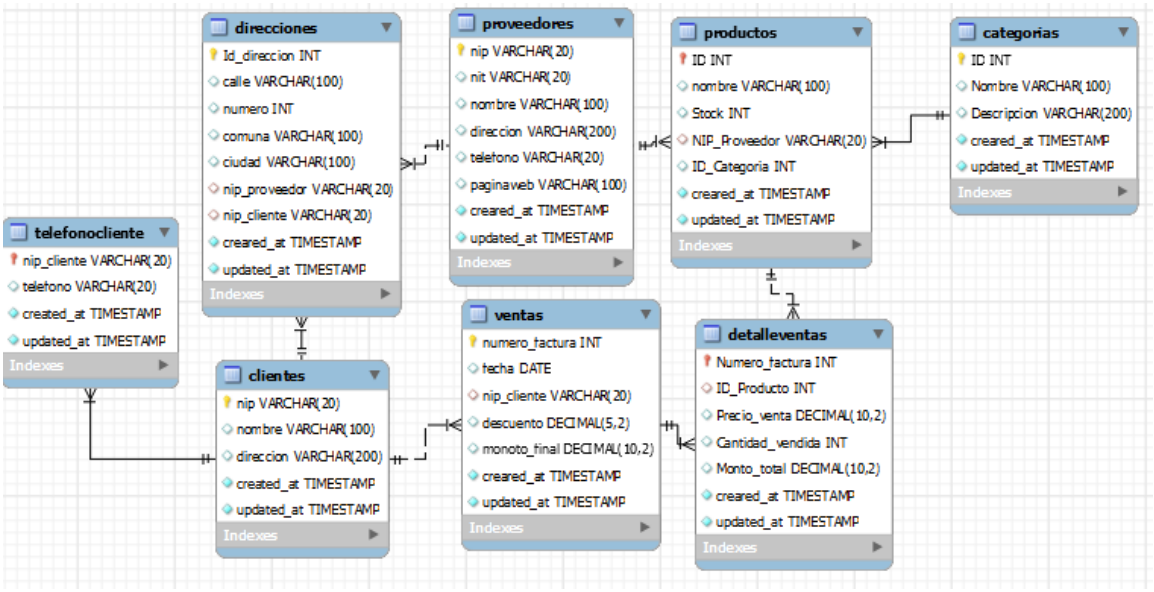

Imagen 1: Diagrama base de datos.

Para poder crear el anterior diagrama en necesario implementar los siguientes comandos:

COMANDOS DDL

```sql
CREATE DATABASE proyectosena;

USE proyectosena;

CREATE TABLE clientes(
    `nip` VARCHAR(20),
    `nombre` VARCHAR(100),
    `direccion` VARCHAR(200),
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY(nip)
)
DEFAULT CHARSET=utf8mb4 COLLATE = utf8mb4_unicode_ci;
```

Imagen 2: Creación de base de datos y tabla padre.

El primer paso primordial es la creación de la base de datos con la sentencia CREATE DATABASE, esto crea el espacio donde se van a implementar y guardar las tablas que se crearan. En el siguiente paso se crea la primera tabla o tabla padre, esta tabla tiene ese nombre porque no depende de ninguna otra y de aquí proporciona llaves foráneas para otras tablas, para crear la tabla se toma la sentencia CREATE TABLE, donde se le asigna un nombre y se crean sus columnas, al momento de crear las columnas se le debe asignar que tipo de dato van a ser, por lo que al final de el nombre de la columna va el nombre VARCHAR, así como este hay mas y los siguientes son unos de los más utilizados:

- VARCHAR: Este tipo se utiliza para almacenar cadenas de texto de longitud variable. Es útil para almacenar nombres, descripciones y otros datos de texto.
- INT: Representa números enteros. Puede ser utilizado para almacenar valores como identificadores, cantidades y otros valores enteros.
- DECIMAL / NUMERIC: Estos tipos se utilizan para almacenar números decimales precisos. Son adecuados para datos financieros y cualquier otro caso en el que la precisión decimal sea importante.
- FLOAT / DOUBLE: Estos tipos se utilizan para números en coma flotante con precisión limitada. Son apropiados para valores científicos y cálculos matemáticos.
- DATETIME / TIMESTAMP: Estos tipos se utilizan para almacenar información sobre fechas y horas. TIMESTAMP también puede incluir información sobre la zona horaria.

Al final de nombrar las columnas es necesario nombrar la llave primaria en este caso se muestra como llave primaria ‘nip’.

Continuando con las relaciones de las bases de datos creamos la tabla telefonocliente, en este lugar se crea la relación 1 a muchas ya que un cliente puede tener múltiples número, pero un número no puede tener muchos clientes, así que en la imagen 3 queda implementado el código.

```
CREATE TABLE telefonocliente(
    `nip_cliente` VARCHAR(20),
    `telefono` VARCHAR(20),
    `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY(`nip_cliente`),
    CONSTRAINT `telefonocliente_nip_cliente_foreing`
    FOREIGN KEY (`nip_cliente`) REFERENCES `clientes`(`nip`)

)DEFAULT CHARSET=utf8mb4
COLLATE = utf8mb4_unicode_ci;
```

Imagen 3. Creación tabla clientes y clave foránea.

Como se puede observar se maneja una estructura de código similar a la de la tabla cliente, creando la tabla con CREATE TABLE y asignando sus columnas, pero además creamos una clave foránea con la sentencia FOREIGN KEY (nombre_columna) REFERENCES 'Nombre_Tabla'('Nombre_columna') y además con CONSTRAINT para darle un nombre que podamos recordar.

A continuación, se mostrarán las imágenes de cada una de las tablas:

```
CREATE TABLE ventas(
    numero_factura INT,
    fecha DATE,
    nip_cliente VARCHAR(20),
    descuento DECIMAL(5,2),
    monoto_final DECIMAL(10,2),
    `creared_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY (numero_factura),
    CONSTRAINT `ventas_nip_cliente_foreing`
    FOREIGN KEY (`nip_cliente`) REFERENCES `clientes`(`nip`)
)DEFAULT CHARSET=utf8mb4
COLLATE = utf8mb4_unicode_ci;

CREATE TABLE proveedores(
    `nip` VARCHAR(20),
    `nit` VARCHAR(20),
    `nombre` VARCHAR(100),
    `direccion` VARCHAR(200),
    `telefono` VARCHAR(20),
    `paginaweb` VARCHAR(100),
    `creared_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY (nip)
)DEFAULT CHARSET=utf8mb4
COLLATE = utf8mb4_unicode_ci;
```

```sql
CREATE TABLE direcciones (
    Id_direccion INT,
    calle VARCHAR(100),
    numero INT,
    comuna VARCHAR(100),
    ciudad VARCHAR(100),
    nip_proveedor VARCHAR(20),
    nip_cliente VARCHAR(20),
    `creared_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY (Id_direccion),
    CONSTRAINT `direcciones_nip_proveedor_foreing`
    FOREIGN KEY(`nip_proveedor`) REFERENCES `proveedores`(`nip`),

    CONSTRAINT `direcciones_nip_clientes_foreint`
    FOREIGN KEY (`nip_cliente`) REFERENCES `clientes`(`nip`)
)DEFAULT CHARSET=utf8mb4
COLLATE = utf8mb4_unicode_ci;

CREATE TABLE productos (
    `ID` INT,
    nombre VARCHAR(100),
    Stock INT,
    NIP_Proveedor VARCHAR(20),
    ID_Categoria INT,
    `creared_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY (ID),
    CONSTRAINT `productos_nip_proveedor_foreint`
    FOREIGN KEY (`NIP_Proveedor`) REFERENCES `proveedores`(`nip`)
)DEFAULT CHARSET=utf8mb4
 COLLATE = utf8mb4_unicode_ci;

CREATE TABLE detalleventas (
    Numero_factura INT,
    ID_Producto INT,
    Precio_venta DECIMAL(10,2),
    Cantidad_vendida INT,
    Monto_total DECIMAL(10,2),
    `creared_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    PRIMARY KEY(Numero_factura),
    CONSTRAINT `factura_numero_ventas_foreint`
    FOREIGN KEY(`Numero_factura`) REFERENCES `ventas`(`numero_factura`),
    CONSTRAINT `detalleventas_ID_productos_foreint`
    FOREIGN KEY(`ID_Producto`) REFERENCES `productos`(`ID`)
)DEFAULT CHARSET=utf8mb4
COLLATE = utf8mb4_unicode_ci;
```

```
110 • CREATE TABLE categorias(
111       `ID` INT,
112       `Nombre` VARCHAR(100),
113       `Descripcion` VARCHAR(200),
114       `creared_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
115       `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
116
117       PRIMARY KEY(ID)
118   );
120 •  ALTER TABLE productos
121    ADD CONSTRAINT `productos_id_categoria_foreign`
122    FOREIGN KEY (`ID`) REFERENCES `categorias`(`ID`);
123
124 •  ALTER TABLE categorias
125    DEFAULT CHARSET=utf8mb4 COLLATE = utf8mb4_unicode_ci;
```

Este sería todo el código para la creación de las tablas con cada relación, para cada tabla se creo su llave primaria así como sus llaves foráneas, al final del código en las líneas 120 y 125 se puede observar que hay que hacer un llamado a la Tabla productos mediante el comando ALTER TABLE, esto porque se tenia que hacer un llamado a la llave ID de tabla categorías que en ese momento no se había creado, por lo que posterior a la creación de la tabla categorías se llama nuevamente a la tabla productos y se crea la llave foránea por medio de FOREIGN KEY.

COMANDOS DML:

Estos comando reciben este nombre por comando de manipulación de datos, ya que con esto llamaremos o modificaremos el contenido de cada tabla, con estos comando vamos a asignarle valores a diferentes tipo de tablas como en la siguiente muestra.

show tables;- Nos permite visualizar todas las tablas disponibles en la base de datos

Result Grid | Filter Rows:

Tables_in_proyectosena
categorias
clientes
detalleventas
direcciones
productos
proveedores
telefonocliente
ventas

Ahora guardaremos o enviaremos información a la tabla padre clientes, por lo que tomamos el comando INTO nombre_tabla(columnas) VALUES(valores_columna):

insert into clientes (nip, nombre, direccion) values ('5473249817', 'Kinnie Dunge', '58903 Main Way');

insert into clientes (nip, nombre, direccion) values ('4041777577', 'Eziechiele Renak', '506 Melody Center');

insert into clientes (nip, nombre, direccion) values ('9541396909', 'Linnell Ahrend', '60 Northwestern Street');

insert into clientes (nip, nombre, direccion) values ('8467826134', 'Auguste Birdis', '45 Walton Way');

insert into clientes (nip, nombre, direccion) values ('5789692311', 'Wyatan Jakeman', '57900 Hovde Pass');

Para poder visualizar los valores en la tabla se realiza con el comando:

SELECT * FROM clientes;

nip	nombre	direccion	created_at	updated_at
4041777577	Eziechiele Renak	506 Melody Center	2023-08-30 10:43:56	2023-08-30 10:43:56
5473249817	Kinnie Dunge	58903 Main Way	2023-08-30 10:41:47	2023-08-30 10:41:47
5789692311	Wyatan Jakeman	57900 Hovde Pass	2023-08-30 10:43:56	2023-08-30 10:43:56
8467826134	Auguste Birdis	45 Walton Way	2023-08-30 10:43:56	2023-08-30 10:43:56
9541396909	Linnell Ahrend	60 Northwestern Street	2023-08-30 10:43:56	2023-08-30 10:43:56
NULL	NULL	NULL	NULL	NULL

Así asignamos los diferentes valores a caga columna de la tabla clientes, a continuación, realizamos la inclusión de datos en la tabla teléfono clientes, que cuenta con una llave primaria.

insert into telefonocliente (nip_cliente, telefono) values ('5473249817', '3185149796');

insert into telefonocliente (nip_cliente, telefono) values ('4041777577', '3173775595');

insert into telefonocliente (nip_cliente, telefono) values ('9541396909', '3133333058');

insert into telefonocliente (nip_cliente, telefono) values ('8467826134', '3134330018');

insert into telefonocliente (nip_cliente, telefono) values ('5789692311', '3138702715');

Visualizamos con SELECT * FROM telefonocliente;

nip_cliente	telefono	created_at	updated_at
4041777577	3173775595	2023-08-30 11:39:17	2023-08-30 11:39:17
5473249817	3185149796	2023-08-30 11:39:17	2023-08-30 11:39:17
5789692311	3138702715	2023-08-30 11:39:17	2023-08-30 11:39:17
8467826134	3134330018	2023-08-30 11:39:17	2023-08-30 11:39:17
9541396909	3133333058	2023-08-30 11:39:17	2023-08-30 11:39:17
NULL	NULL	NULL	NULL

Como ultimo ejemplo realizaremos el mismo paso con la tabla direcciones:

```
insert into direcciones (Id_direccion, calle, numero, comuna, ciudad, nip_cliente) values (1, 'Kenwood', '00', 'Point', 'Xinli', '5473249817');

insert into direcciones (Id_direccion, calle, numero, comuna, ciudad, nip_cliente) values (2, 'Green', '76899', 'Road', 'Orocuina', '4041777577');

insert into direcciones (Id_direccion, calle, numero, comuna, ciudad, nip_cliente) values (3, 'Ronald Regan', '006', 'Junction', 'Porangaba', '9541396909');

insert into direcciones (Id_direccion, calle, numero, comuna, ciudad, nip_cliente) values (4, '8th', '0', 'Plaza', 'Belogorskīy', '8467826134');

insert into direcciones (Id_direccion, calle, numero, comuna, ciudad, nip_cliente) values (5, '3rd', '87', 'Avenue', 'Njeru', '5789692311');
```

de igual forma visualizamos con SELECT * FROM direcciones;

Id_direccion	calle	numero	comuna	ciudad	nip_proveedor	nip_cliente	creared_at	updated_at
1	Kenwood	0	Point	Xinli	NULL	5473249817	2023-08-30 12:30:20	2023-08-30 12:30:20
2	Green	76899	Road	Orocuina	NULL	4041777577	2023-08-30 12:30:20	2023-08-30 12:30:20
3	Ronald Regan	6	Junction	Porangaba	NULL	9541396909	2023-08-30 12:30:20	2023-08-30 12:30:20
4	8th	0	Plaza	Belogorskīy	NULL	8467826134	2023-08-30 12:30:20	2023-08-30 12:30:20
5	3rd	87	Avenue	Njeru	NULL	5789692311	2023-08-30 12:30:20	2023-08-30 12:30:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Es importante saber que para poder crear una dirección o un teléfono el id cliente o como en el ejemplo de la tabla direcciones nip_proveedor debe existir ya que, si no fuera así, no se pudiera crear ese dato porque es necesaria una clave primaria.

Ahora para realizar una consulta en estas tablas ya creadas, se usa el siguiente comando.

```
SELECT * FROM clientes

WHERE nip>0

ORDER BY nombre;
```

Que nos entregaría todos los datos que hace referencia al *, de la tabla clientes, donde el nip es mayor a 0(nip>0) ordenados por el nombre, como se muestra en la siguiente imagen.

nip	nombre	direccion	created_at	updated_at
8467826134	Auguste Birdis	45 Walton Way	2023-08-30 10:43:56	2023-08-30 10:43:56
4041777577	Eziechiele Renak	506 Melody Center	2023-08-30 10:43:56	2023-08-30 10:43:56
5473249817	Kinnie Dunge	58903 Main Way	2023-08-30 10:41:47	2023-08-30 10:41:47
9541396909	Linnell Ahrend	60 Northwestern Street	2023-08-30 10:43:56	2023-08-30 10:43:56
5789692311	Wyatan Jakeman	57900 Hovde Pass	2023-08-30 10:43:56	2023-08-30 10:43:56
NULL	NULL	NULL	NULL	NULL

Así es como se crean tablas con comando DDL y se les da información y se filtra esta información con comando DML.

CONCLUSIONES:

- Es muy importante estructurar y visualizar correctamente todas las relaciones y orden de las tablas, ya que al momento de crear e insertar datos, se debe cuidar la duplicidad y veracidad de los datos.
- Al momento de crear las bases de datos se debe asignar correctamente que tipo de datos se van a almacenar así como el tamaño de cada uno, esto para no ocupar ni desperdiciar espacio en la base de datos.