UNIVERSIDAD PRIVADA
“FRANZ TAMAYO”
Ingeniería de Sistemas

Nombres:

ALEXANDER HEYTAN CALLISAYA VALENCIA

HENRY JAVIER HUARACHI QUISPE

FREDDY MACHACA MAMANI

DOCENTE:

Lic.William Roddy Barra Paredes

El Alto – Bolivia 2022

INDICE

1. Introducción 3
2. Análisis y diseño de la Base de Datos 3
Código SQL 5
3. Usabilidad 13
4. Conclusiones 13

1. Introducción

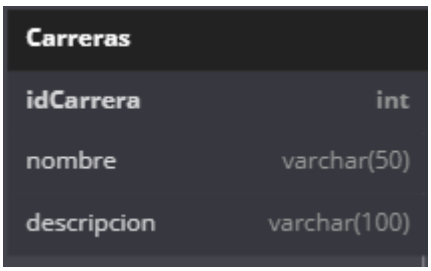
La base de datos de una universidad incluiría información relacionada con los estudiantes, docentes, carreras, materias, notas, roles, usuarios, horarios, etc. Esta información se almacenaría en tablas relacionadas entre sí para permitir el acceso y la búsqueda de información precisa. La base de datos se diseñaría para permitir la manipulación de datos, la recuperación de información y el seguimiento de los cambios en los datos con el fin de desarrollar lo mencionado se usará el MariaDb como gestor de base de datos.

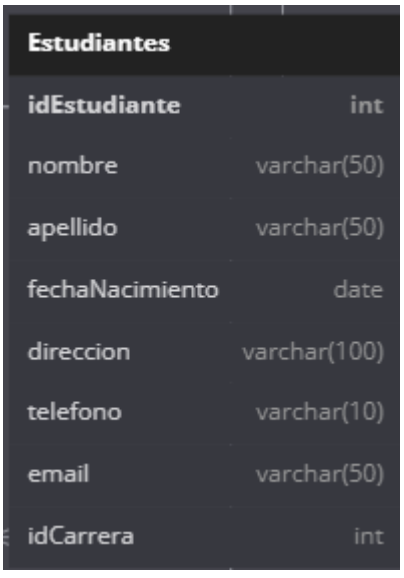
2. Análisis y diseño de la Base de Datos

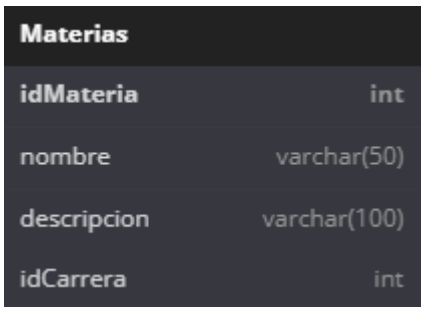
a. Contexto de la Base de Datos:

Debido a que la base de datos se trata de la información respecto a una universidad se identifica como nombre adecuado para la base de datos ser “DB_Universidad”.

b. Entidades/tablas de sistema:







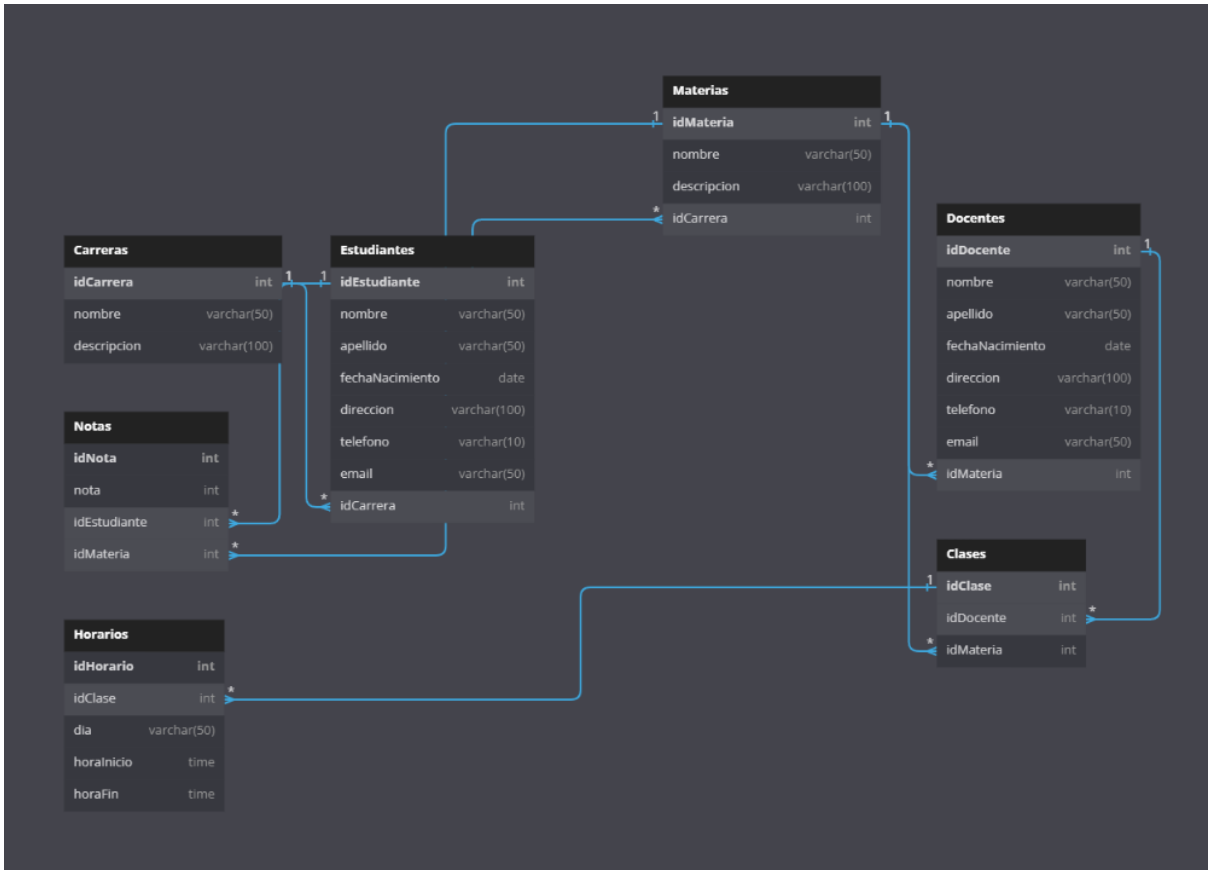
Docentes	
idDocente	int
nombre	varchar(50)
apellido	varchar(50)
fechaNacimiento	date
direccion	varchar(100)
telefono	varchar(10)
email	varchar(50)
idMateria	int

Notas	
idNota	int
nota	int
idEstudiante	int
idMateria	int

Horarios	
idHorario	int
idClase	int
dia	varchar(50)
horaInicio	time
horaFin	time

Clases	
idClase	int
idDocente	int
idMateria	int

c. Diseño de la base de datos.



Código SQL

```sql
create database DB_Universidad;
use DB_Universidad;


-- Tabla de Carreras
create table Carreras(
    idCarrera int primary key auto_increment,
    nombre varchar(50) not null,
    descripcion varchar(100) not null
);

-- Tabla de Estudiantes
create table Estudiantes(
    idEstudiante int primary key auto_increment,
    nombre varchar(50) not null,
    apellido varchar(50) not null,
    fechaNacimiento date not null,
    direccion varchar(100) not null,
    telefono varchar(10) not null,
    email varchar(50) not null,
    idCarrera int not null,
    foreign key (idCarrera) references Carreras(idCarrera)
);


-- Tabla de Materias
create table Materias(
    idMateria int primary key auto_increment,
    nombre varchar(50) not null,
    descripcion varchar(100) not null,
    idCarrera int not null,
    foreign key (idCarrera) references Carreras(idCarrera)
);

-- Tabla de docentes
create table Docentes(
    idDocente int primary key auto_increment,
    nombre varchar(50) not null,
    apellido varchar(50) not null,
    fechaNacimiento date not null,
    direccion varchar(100) not null,
    telefono varchar(10) not null,
    email varchar(50) not null,
    idMateria int not null,
    foreign key (idMateria) references Materias(idMateria)
);

-- Tabla de notas
create table Notas(
    idNota int primary key auto_increment,
    nota int not null,
    idEstudiante int not null,
    idMateria int not null,
    foreign key (idEstudiante) references Estudiantes(idEstudiante),
    foreign key (idMateria) references Materias(idMateria)
);
```

```sql
-- tabla de clases
create table Clases(
    idClase int primary key auto_increment,
    idDocente int not null,
    idMateria int not null,
    foreign key (idDocente) references Docentes(idDocente),
    foreign key (idMateria) references Materias(idMateria)
);

-- tabla de horarios
create table Horarios(
    idHorario int primary key auto_increment,
    idClase int not null,
    dia varchar(50) not null,
    horaInicio time not null,
    horaFin time not null,
    foreign key (idClase) references Clases(idClase)
);


-- insertar datos

-- insertar datos en la tabla carreras
insert into Carreras(nombre, descripcion)
values('Ingenieria en Sistemas', 'Ingenieria en Sistemas'),
      ('Ingenieria en Mecatronica', 'Ingenieria en Mecatronica'),
      ('Ingenieria en Mecanica', 'Ingenieria en Mecanica'),
      ('Ingenieria en Electrica', 'Ingenieria en Electrica'),
      ('Ingenieria en Electronica', 'Ingenieria en Electronica'),
      ('Ingenieria en Industrial', 'Ingenieria en Industrial'),
      ('Ingenieria en Mecanica Industrial', 'Ingenieria en Mecanica
Industrial'),
      ('Derecho',' Derecho'),
      ('Medicina', 'Medicina'),
      ('Psicologia', 'Psicologia');

-- insertar datos en la tabla estudiantes
insert into Estudiantes(nombre, apellido, fechaNacimiento, direccion,
telefono, email, idCarrera)
values('Juan', 'Perez', '1990-01-01', 'Calle 1', '1234567890',
'juan@gmial.com', 1),
      ('Maria', 'Gomez', '1990-01-01', 'Calle 2', '1234567890',
'maria@gmail.com', 2),
      ('Pedro', 'Gonzales', '1990-01-01', 'Calle 3', '1234567890',
'pedro@gmail.com', 3),
      ('Jose', 'Lopez', '1990-01-01', 'Calle 4', '1234567890',
'jose@gmail.com', 4),
      ('Luis', 'Martinez', '1990-01-01', 'Calle 5', '1234567890',
'luis@gmail.com', 5),
      ('Ana', 'Rodriguez', '1990-01-01', 'Calle 6', '1234567890',
'ana@gmail.com', 6),
      ('Lorena', 'Gutierrez', '1990-01-01', 'Calle 7', '1234567890',
'lorena@gmail.com', 7),
      ('Sofia', 'Hernandez', '1990-01-01', 'Calle 8', '1234567890',
'sofia@gmail.com', 8),
      ('Miguel', 'Diaz', '1990-01-01', 'Calle 9', '1234567890',
'miguel@gmail.com', 9),
      ('Carlos', 'Perez', '1990-01-01', 'Calle 10', '1234567890',
'carlos@gmail.com', 10);

-- insertar datos en la tabla materias
```

```sql
insert into Materias(nombre, descripcion, idCarrera)
values('Matematicas', 'Matematicas', 1),
      ('Fisica', 'Fisica', 2),
      ('Quimica', 'Quimica', 3),
      ('Biologia', 'Biologia', 4),
      ('Programacion', 'Programacion', 5),
      ('Base de datos', 'Base de datos', 6),
      ('Redes', 'Redes', 7),
      ('Ingles', 'Ingles', 8),
      ('Frances', 'Frances', 9),
      ('Aleman', 'Aleman', 10);

-- insertar datos en la tabla docentes
insert into Docentes(nombre, apellido, fechaNacimiento, direccion,
telefono, email, idMateria)
values('Carlos', 'Condori', '1990-01-01', 'Calle 1', '1234567890',
'carlos@gmail.com', 1),
      ('Alejandro', 'Plata', '1990-01-01', 'Calle 2', '1234567890',
'alejandro@gmail.com', 2),
      ('Abel', 'Quispe', '1990-01-01', 'Calle 3', '1234567890',
'abel@gmail.com', 3),
      ('Alex', 'Gonzales', '1990-01-01', 'Calle 4', '1234567890',
'alex@gmail.com', 4),
      ('Luis', 'Rojas', '1990-01-01', 'Calle 5', '1234567890',
'luis@gmail.com', 5),
      ('Bocaro', 'Rodriguez', '1990-01-01', 'Calle 6', '1234567890',
'bocaro@gmail.com', 6),
      ('Raul', 'Gutierrez', '1990-01-01', 'Calle 7', '1234567890',
'raul@gmail.com', 7),
      ('Misael', 'Paredes', '1990-01-01', 'Calle 8', '1234567890',
'misael@gmail.com', 8),
      ('Minerva', 'Torrez', '1990-01-01', 'Calle 9', '1234567890',
'minerva@gmail.com', 9),
      ('Estefani', 'Maldonado', '1990-01-01', 'Calle 10', '1234567890',
'estefani@gmail.com', 10);

-- insertar datos en la tabla notas
insert into Notas(idEstudiante, idMateria, nota)
values(1, 1, 15),
      (2, 2, 16),
      (3, 3, 17),
      (4, 4, 18),
      (5, 5, 19),
      (6, 6, 20),
      (7, 7, 21),
      (8, 8, 22),
      (9, 9, 23),
      (10, 10, 24);


-- insertar datos en la tabla clases
insert into Clases(idDocente, idMateria) values(1, 1);
insert into Clases(idDocente, idMateria) values(2, 2);
insert into Clases(idDocente, idMateria) values(3, 3);
insert into Clases(idDocente, idMateria) values(4, 4);
insert into Clases(idDocente, idMateria) values(5, 5);
insert into Clases(idDocente, idMateria) values(6, 6);
insert into Clases(idDocente, idMateria) values(7, 7);
insert into Clases(idDocente, idMateria) values(8, 8);
insert into Clases(idDocente, idMateria) values(9, 9);
insert into Clases(idDocente, idMateria) values(10, 10);
```

```sql
-- insertar datos en la tabla horarios
insert into Horarios(idClase, dia, horaInicio, horaFin)
values(1, 'Lunes', '08:00:00', '10:00:00'),
      (2, 'Martes', '08:00:00', '10:00:00'),
      (3, 'Miercoles', '08:00:00', '10:00:00'),
      (4, 'Jueves', '08:00:00', '10:00:00'),
      (5, 'Viernes', '08:00:00', '10:00:00'),
      (6, 'Sabado', '08:00:00', '10:00:00'),
      (7, 'Domingo', '08:00:00', '10:00:00'),
      (8, 'Lunes', '10:00:00', '12:00:00'),
      (9, 'Martes', '10:00:00', '12:00:00'),
      (10, 'Miercoles', '10:00:00', '12:00:00');

-- insertar datos en la tabla estudiantes todos en la misma carrera y
materia para probar el procedimiento almacenado
insert into Estudiantes(nombre, apellido, fechaNacimiento, direccion,
telefono, email, idCarrera)
values('Carlos', 'Condori', '1990-01-01', 'Calle 1', '1234567890', '', 6),
      ('Alejandro', 'Plata', '1990-01-01', 'Calle 2', '1234567890', '', 6),
      ('Abel', 'Quispe', '1990-01-01', 'Calle 3', '1234567890', '', 6),
      ('Alex', 'Gonzales', '1990-01-01', 'Calle 4', '1234567890', '', 6),
      ('Luis', 'Rojas', '1990-01-01', 'Calle 5', '1234567890', '', 6),
      ('Bocaro', 'Rodriguez', '1990-01-01', 'Calle 6', '1234567890', '',
6),
      ('Raul', 'Gutierrez', '1990-01-01', 'Calle 7', '1234567890', '', 6),
      ('Misael', 'Paredes', '1990-01-01', 'Calle 8', '1234567890', '', 6),
      ('Minerva', 'Torrez', '1990-01-01', 'Calle 9', '1234567890', '', 6),
      ('Estefani', 'Maldonado', '1990-01-01', 'Calle 10', '1234567890', '',
6);

-- todos los estudiantes de la carrera de base de datos
select * from Estudiantes
         inner join Carreras on Estudiantes.idCarrera = Carreras.idCarrera
         inner join Materias M on Carreras.idCarrera = M.idCarrera
         where idMateria = 6;


# 1. Consultas SQL que maneja JOINS = 5 Consultas
# 2. Funciones UDF = 3 FUNCIONES
# 3. Vistas = 5 Vistas
# a. Reutiliza 3 consultas
# b. Crear 2 nuevas
# 4. Triggers = 3 TRIGGERS
# a. 1 trigger de validación
# b. 2 triggers de autoria

# 1. Consultas SQL que maneja JOINS = 5 Consultas
# 1.1. Cuales son los estudiantes que tienen notas mayores a 18
select e.nombre, e.apellido, n.nota
from Estudiantes e
inner join Notas n on e.idEstudiante = n.idEstudiante
where n.nota > 5;

# 1.2. Cuales son los estudiantes que estan en la carrera de Ingenieria de
Sistemas
select e.nombre, e.apellido, c.nombre
from Estudiantes e
inner join Carreras c on e.idCarrera = c.idCarrera
where c.nombre = 'Ingenieria en Sistemas';
```

```sql
# 1.3. que docentes imparten clases los lunes y miercoles
select d.nombre, d.apellido, h.dia
from Docentes d
inner join Clases c on d.idDocente = c.idDocente
inner join Horarios h on c.idClase = h.idClase
where h.dia = 'Lunes' or h.dia = 'Miercoles';

# 1.4. que docentes tienen a cargo la materia de Base de datos
select d.nombre, d.apellido, m.nombre
from Docentes d
inner join Clases c on d.idDocente = c.idDocente
inner join Materias m on c.idMateria = m.idMateria
where m.nombre = 'Base de datos';

# 1.5. que estudiantes tienen notas mayores a 5 en la materia de Base de
datos
select e.nombre, e.apellido, m.nombre, n.nota
from Estudiantes e
inner join Notas n on e.idEstudiante = n.idEstudiante
inner join Materias m on n.idMateria = m.idMateria
where m.nombre = 'Base de datos' and n.nota > 5;

# 2. Funciones UDF = 3 FUNCIONES
# 2.1. Funcion que retorne el promedio de notas de un estudiantes mediante
su Id
create function promedioEstudiante(idEstudiante int)
returns float
begin
    declare promedio float;
    select avg(n.nota) into promedio
    from Notas n
    where n.idEstudiante = idEstudiante;
    return promedio;
end;

select promedioEstudiante(1);

# 2.2. Funcion para obtener la informacion de un estudiante matriculado en
una materia
# la funcion debe recibir como parametro el nombre del estudiante y el
nombre de la materia
create function informacionEstudiante(nombreEstudiante varchar(50),
nombreMateria varchar(50))
returns varchar(100)
begin
    declare informacion varchar(100);
    select concat(e.nombre, ' ', e.apellido, ' ', m.nombre) into
informacion
    from Estudiantes e
    inner join Notas n on e.idEstudiante = n.idEstudiante
    inner join Materias m on n.idMateria = m.idMateria
    where e.nombre = nombreEstudiante and m.nombre = nombreMateria;
    return informacion;
end;

select informacionEstudiante('Ana', 'Base de datos');

# 2.3. Funcion para eligir el mejor estudiante de una materia
# la funcion debe recibir como parametro el nombre de la materia
create function mejorEstudiante(nombreMateria varchar(50))
```

```
returns varchar(100)
begin
    declare mejor varchar(100);
    select concat(e.nombre, ' ', e.apellido, ' ', m.nombre) into mejor
    from Estudiantes e
    inner join Notas n on e.idEstudiante = n.idEstudiante
    inner join Materias m on n.idMateria = m.idMateria
    where m.nombre = nombreMateria and n.nota = (select max(nota) from
Notas);
    return mejor;
end;

select mejorEstudiante('programacion');

# 3. Vistas = 5 Vistas
# a. Reutiliza 3 consultas
# b. Crear 2 nuevas

# 3.1. Vistas de estudiantes que tienen notas mayores a 5 en la materia de
Base de datos
create view estudiantesNotasBaseDatos as
select e.nombre as 'Nombre',
       e.apellido as 'Apellido',
       m.nombre as 'Materia',
       n.nota as 'Nota'
from Estudiantes e
inner join Notas n on e.idEstudiante = n.idEstudiante
inner join Materias m on n.idMateria = m.idMateria
where m.nombre = 'Base de datos' and n.nota > 5;

select * from estudiantesNotasBaseDatos;

# 3.2. Vistas de estudiantes que estan en la carrera de Ingenieria de
Sistemas
create view estudiantesCarreraIngenieriaSistemas as
select e.nombre as 'Nombre',
       e.apellido as 'Apellido',
       c.nombre as 'Carrera'
from Estudiantes e
inner join Carreras c on e.idCarrera = c.idCarrera
where c.nombre = 'Ingenieria en Sistemas';

select * from estudiantesCarreraIngenieriaSistemas;

# 3.3. Vistas de que docentes imparten clases los lunes y miercoles
create view docentesImpartenClasesLunesMiercoles as
select d.nombre as 'Nombre',
       d.apellido as 'Apellido',
       h.dia as 'Dia'
from Docentes d
inner join Clases c on d.idDocente = c.idDocente
inner join Horarios h on c.idClase = h.idClase
where h.dia = 'Lunes' or h.dia = 'Miercoles';

select * from docentesImpartenClasesLunesMiercoles;

# 3.4. Vistas de que docentes tienen a cargo la materia de programacion
create view docentesMateriaProgramacion as
select d.nombre as 'Nombre',
       d.apellido as 'Apellido',
       m.nombre as 'Materia'
```

```sql
from Docentes d
inner join Clases c on d.idDocente = c.idDocente
inner join Materias m on c.idMateria = m.idMateria
where m.nombre = 'Programacion';

select * from docentesMateriaProgramacion;

# 3.5. Vistas de estudiantes que tiene clases con el docente william barra
create view estudiantesClasesDocenteWilliamBarra as
select e.nombre as 'Nombre',
       e.apellido as 'Apellido',
       d.nombre as 'Docente'
from Estudiantes e
inner join Clases c on e.idEstudiante = c.idMateria
inner join Docentes d on c.idDocente = d.idDocente
where d.nombre = 'William' and d.apellido = 'Barra';

select * from estudiantesClasesDocenteWilliamBarra;

# 4. Triggers = 3 Triggers
# a. 1 trigger de validacion
# b. 2 triggers de auditoria

# 4.1. Trigger de validacion para que se elimine un estudiante solo si no
tiene notas
create trigger eliminarEstudiante
before delete on Estudiantes
for each row
begin
    if (select count(*) from Notas where idEstudiante = old.idEstudiante) >
0 then
        signal sqlstate '45000'
        set message_text = 'No se puede eliminar el estudiante porque tiene
notas';
    end if;
end;

delete from Estudiantes where idEstudiante = 1;

# 4.2. Trigger de auditoria para que se registre cuando se haga un cambio
en las notas

create table AuditoriaNotas (
    idEstudiante int,
    idMateria int,
    notaAnterior int,
    notaActual int
);

create trigger auditoriaNotas
after update on Notas
for each row
begin
    insert into AuditoriaNotas values (old.idEstudiante, old.idMateria,
old.nota, new.nota);
end;

update Notas set nota = 5 where idEstudiante = 1 and idMateria = 1;

select * from AuditoriaNotas;
```

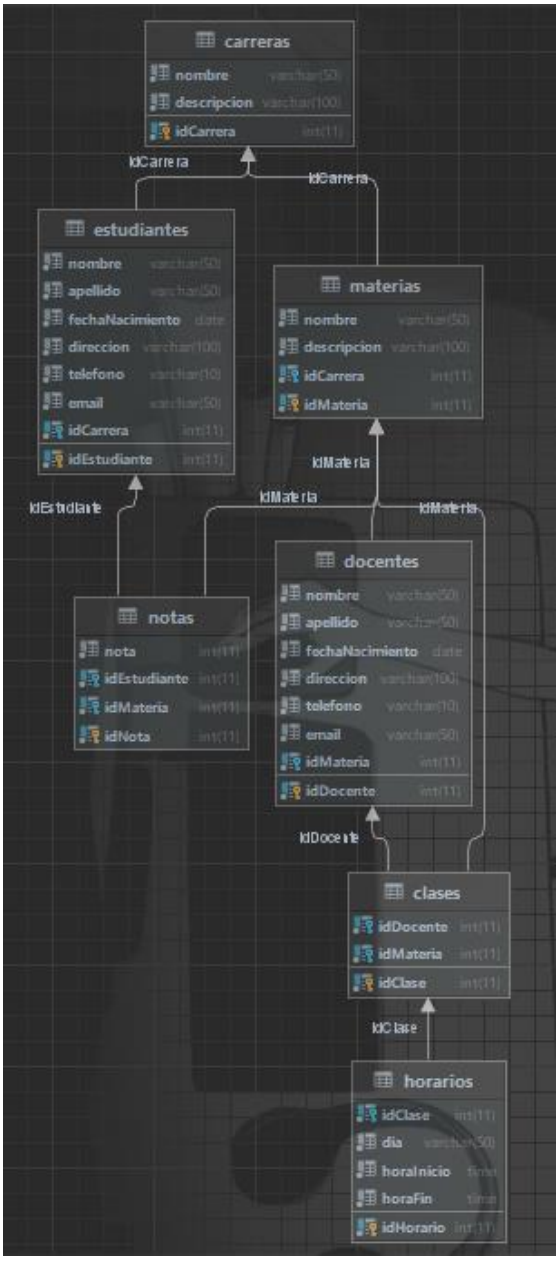
```sql
# 4.3. Trigger de auditoria para que se registre cuando se haga un cambio
en las materias

create or replace table AuditoriaMaterias (
    idMateria int,
    materiaAnterior varchar(50),
    materiaActual varchar(50)
);

create or replace trigger auditoriaMaterias
after update on Materias
for each row
begin
    insert into AuditoriaMaterias values (old.idMateria, old.nombre,
new.nombre);
end;

update Materias set nombre = 'Base de datos' where idMateria = 2;

select * from AuditoriaMaterias;
```



3. Usabilidad

Video del sistema: https://youtu.be/sXkik8vP3Dc

4. Conclusiones

- Programar en una base de datos para una universidad proporciona una gran cantidad de beneficios, como una mejor organización de los datos, una mayor seguridad de los datos, una mejor eficiencia en el procesamiento de los datos y una mayor capacidad de recuperar los datos si se producen problemas.

- La programación en una base de datos para una universidad también puede proporcionar una mejor comunicación entre los departamentos y mejorar la gestión de los recursos humanos.

- La programación en una base de datos para una universidad también puede ayudar a mejorar la seguridad de los datos y reducir el riesgo de pérdida o alteración de datos.