Base De Datos

Contribuidores


FreddyMachaca

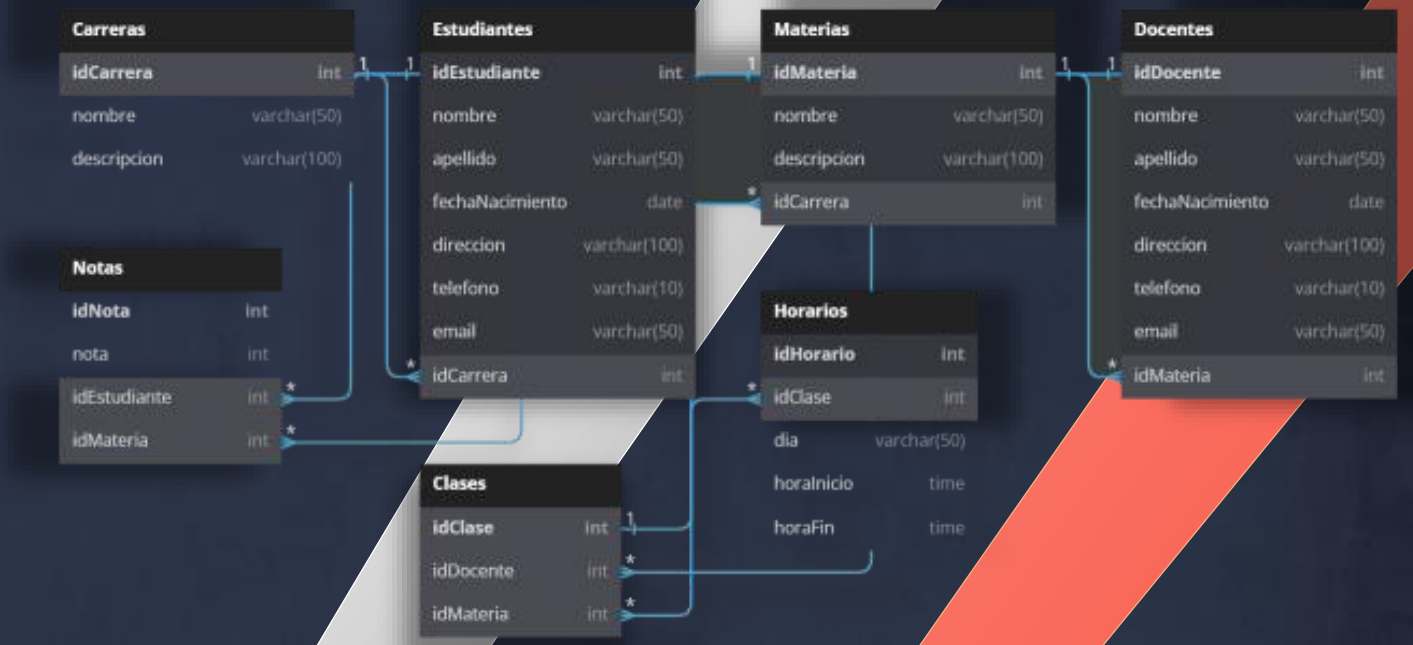

REXFOX195


Heitan99

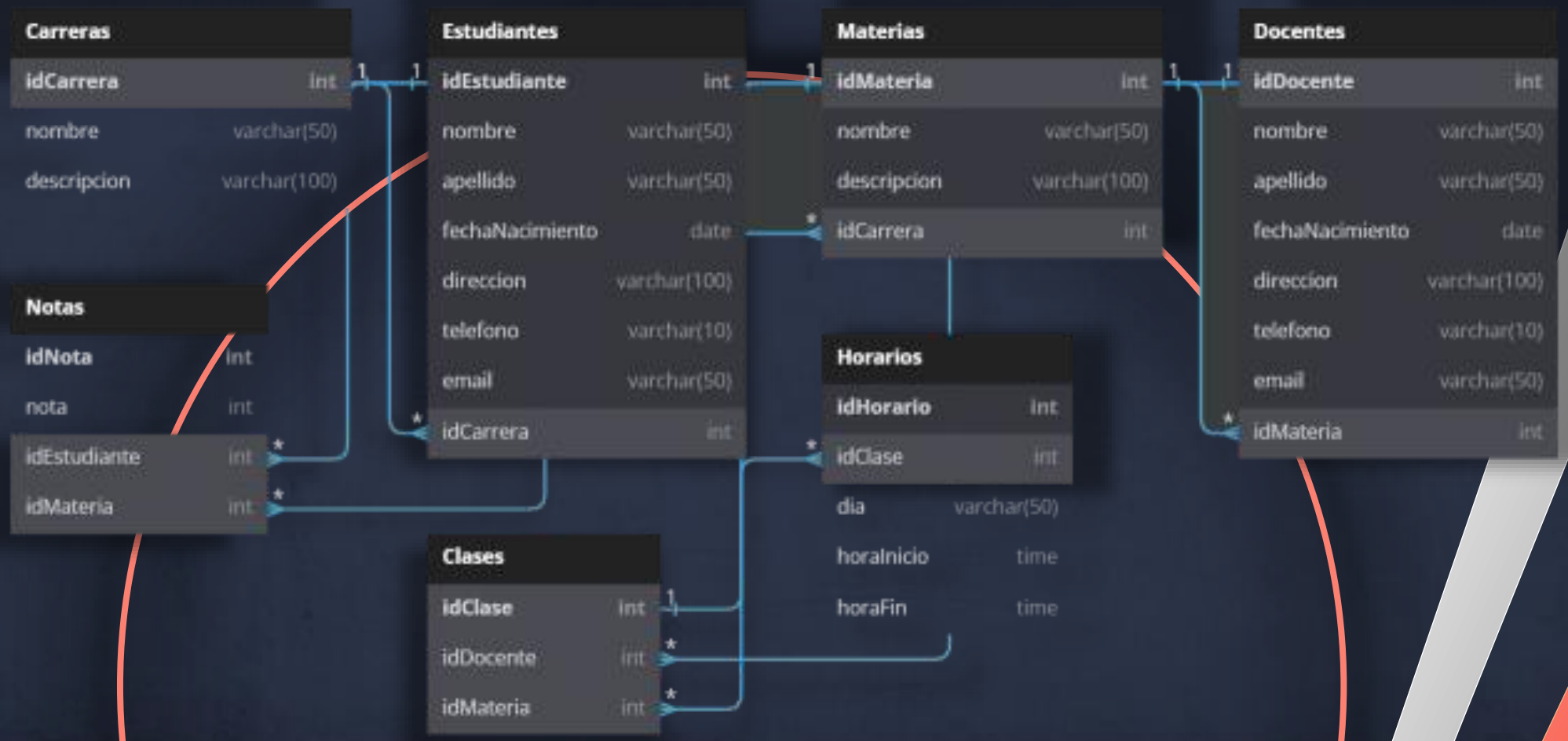

DB_Universidad

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.



Diseño Entidad Relación



E-R

DB_Universidad

Consultas SQL que maneja JOINS
5 Consultas

DB_Universidad

1.1. Cuales son los estudiantes que tienen notas mayores a 18

```sql
select e.nombre, e.apellido, n.nota
from Estudiantes e
inner join Notas n on e.idEstudiante = n.idEstudiante
where n.nota > 5;
```

E-R

DB_Universidad

1.2. Cuales son los estudiantes que están en la carrera de Ingeniería de Sistemas

```sql
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';
```

E-R

DB_Universidad

1.3. que docentes imparten clases los lunes y miércoles.

```sql
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';
```

E-R

DB_Universidad

1.4. que docentes tienen a cargo la materia de Base de datos.

```sql
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';
```

E-R

DB_Universidad

1.5. que estudiantes tienen notas mayores a 5 en la materia de Base de datos.

```sql
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;
```

E-R

DB_Universidad

Desarrollo de 3 Funciones

DB_Universidad

2.1. Función 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( idEstudiante: 1);
```

E-R

DB_Universidad

2.2. Función para obtener la información de un estudiante matriculado en una materia # la función debe recibir como parámetro el nombre del estudiante y el nombre de la materia.

```sql
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( nombreEstudiante: 'Ana', nombreMateria: 'Base de datos');
```

E-R

DB_Universidad

2.3. Función para elegir el mejor estudiante de una materia # la función debe recibir como parámetro el nombre de la materia.

```sql
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( nombreEstudiante: 'Ana', nombreMateria: 'Base de datos');
```

E-R

DB_Universidad

Desarrollo de 3 Vistas

DB_Universidad

3.1. Vistas de estudiantes que tienen notas mayores a 5 en la materia de Base de datos.

```sql
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;
```

E-R

DB_Universidad

3.2. Vistas de estudiantes que están en la carrera de Ingeniería de Sistemas.

```sql
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;
```

E-R

DB_Universidad

3.3. Vistas de que docentes imparten clases los lunes y miércoles.

```sql
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;
```

E-R

DB_Universidad

Desarrollo de 3 Triggers

DB_Universidad

3.4. Vistas de que docentes tienen a cargo la materia de programación.

```sql
create view docentesMateriaProgramacion as
select d.nombre as 'Nombre',
       d.apellido as 'Apellido',
       m.nombre as 'Materia'
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;
```

E-R

DB_Universidad

3.5. Vistas de estudiantes que tiene clases con el docente William barra.

```sql
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;
```

E-R

DB_Universidad

4.1. Trigger de validación 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;
```

E-R

DB_Universidad

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

```sql
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;
```

E-R

DB_Universidad

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;
```

E-R

DB_Universidad