UTN · La Plata

Proyecto de Base de Datos

Gestionar Base de Datos para la organización “ImpulsaTuNota”

Integrantes

Máximo Carpignano: legajo 32971

Joaquin Montes: legajo 33459

Pedro Fiuza: legajo 33142

Comisión

S31

Año de cursada

2025

Facultad

UTN-FRLP

Fase 6 - Implementación de la Base de Datos

Durante esta fase se llevó a cabo el diseño y la implementación de la base de datos del sistema. Se partió del modelo relacional previamente definido, el cual se transformó en un conjunto de sentencias SQL para la creación de las tablas, sus atributos y las relaciones entre ellas. El motor de base de datos utilizado fue **SQLServer**.

En primer lugar, se procedió a la creación de la base de datos y de todas las tablas que conforman su estructura:

```sql
-- Creación de BD --
create database ImpulsaTuNota
use ImpulsaTuNota
```

Para cada tabla se definieron adecuadamente sus claves primarias y foráneas, con el objetivo de asegurar la integridad referencial entre las entidades. Asimismo, se establecieron restricciones como **NOT NULL**, **UNIQUE** y validaciones específicas de tipo de dato, a fin de garantizar la consistencia de la información almacenada.

```sql
-- Creación de Tablas --
--Tabla alumnos
create table Alumnos
    (nombreAlu varchar(40) not null,
    legajo int primary key identity,
    dniAlu int not null,
    apellidoAlu varchar(40) not null,
    sexoAlu varchar(10) not null check (sexoAlu IN ('hombre', 'mujer')),
    fechaNacAlu date not null,
    promedioAlu float,
    regularidadAlu bit)

--Tabla Profesores
create table Profesores
    (nombreProf varchar(40) not null,
    dniProf int primary key,
    apellidoProf varchar(40) not null,
    sexoProf varchar(10) not null check (sexoProf IN ('hombre', 'mujer')),
    fechaNacProf date not null)

--Tabla mailsAlu
create table mailsAlu
    (direccionAlu nvarchar(50) primary key,
    legajo int not null,
    foreign key (legajo) references Alumnos(legajo))
```

```sql
--Tabla mailsProfe
create table mailsProf
    (direccionProf nvarchar(50) primary key,
    dniProf int not null,
    foreign key (dniProf) references Profesores(dniProf))

--Tabla telefonosProf
create table telefonosProf
    (numero int primary key,
    dniProf int not null,
    foreign key (dniProf) references Profesores(dniProf))

--Tabla Materias
create table Materias
    (#materia int primary key identity,
    nombre varchar(40) not null,
    descripcion nvarchar(100) not null,
    fechaInicio date,
    fechaFin date,
    faltasMax int not null)

--Tabla profesorDictaMaterias
create table profesor_dicta_materia
    (dniProf int not null,
    #materia int not null,
    primary key (dniProf, #materia),
    foreign key (dniProf) references Profesores(dniProf),
    foreign key (#materia) references Materias(#materia))
```

```sql
--Tabla AlumnoCursaMateria
create table alumno_cursa_materia
    (#materia int not null,
    legajo int not null,
    notaFinal float,
    cantFaltas int,
    primary key (#materia, legajo),
    foreign key (#materia) references Materias(#materia),
    foreign key (legajo) references Alumnos(legajo))

--Tabla Evaluaciones
create table Evaluaciones
    (idEvaluacion int primary key identity,
    fecha date,
    tipo varchar(10) not null check (tipo in ('TP', 'Examen', 'Examen Oral')),
    #materia int not null,
    foreign key (#materia) references Materias(#materia))

--Tabla AlumnoRindeEvaluacion
create table alumno_rinde_evaluacion
    (legajo int not null,
    idEvaluacion int not null
    primary key (legajo, idEvaluacion),
    nota float not null,
    foreign key (legajo) references Alumnos(legajo),
    foreign key (idEvaluacion) references Evaluaciones(idEvaluacion))
```

```sql
--Tabla Grupos
create table Grupos
    (#grupo int primary key identity,
    sigla varchar(5) not null,
    #materia int,
    foreign key (#materia) references Materias(#materia))

--Tabla AlumnoInscribeGrupo
create table alumno_inscribe_grupo
    (legajo int not null,
    #grupo int not null,
    fechaInsc date not null,
    primary key (legajo, #grupo),
    estadoInsc varchar(10) not null check (estadoInsc in ('Aprobada', 'Rechazada', 'Pendiente',
    foreign key (legajo) references Alumnos(legajo),
    foreign key (#grupo) references Grupos(#grupo))

--Tabla Sedes
create table Sedes
    (idSede int primary key identity,
    nombre varchar(40) not null,
    cantAulas int not null,--derivado
    localidad nvarchar(50) not null,
    calle nvarchar (40) not null,
    numero int not null)
```

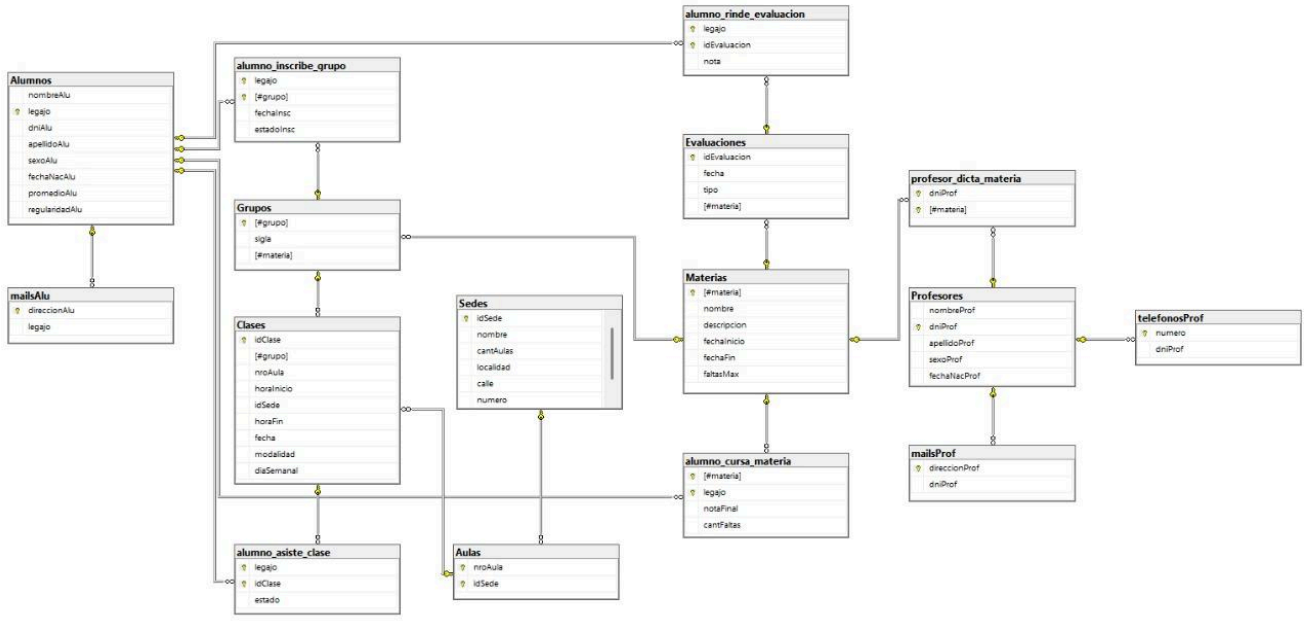
```sql
--Tabla Aulas
create table Aulas
    (nroAula int not null,
    idSede int not null,
    primary key (nroAula, idSede),
    foreign key (idSede) references Sedes(idSede))

--Tabla Clases
create table Clases
    (idClase int primary key identity,
    #grupo int not null,
    nroAula int not null,
    horaInicio time not null,
    idSede int not null,
    horaFin time not null,
    fecha date not null,
    modalidad varchar(15) not null check (modalidad in ('Presencial', 'Virtual')),
    diaSemanal varchar(15) not null,
    foreign key (#grupo) references Grupos(#grupo),
    foreign key (nroAula, idSede) references Aulas(nroAula, idSede))

--Tabla AlumnoAsisteClase
create table alumno_asiste_clase
    (legajo int not null,
    idClase int not null,
    estado varchar(15) not null check (estado in ('Asistido', 'Faltado', 'Justificado')),
    primary key (legajo, idClase),
    foreign key (legajo) references Alumnos(legajo),
    foreign key (idClase) references Clases(idClase))
```

Una vez creadas las estructuras, se realizó la carga inicial de datos, completando los campos necesarios para facilitar la posterior verificación funcional del sistema.

A continuación, se presenta el **modelo relacional** final del sistema.



Fase 7 - Resolucion de Problematicas

En esta etapa se formularon e implementaron una serie de consultas SQL orientadas a resolver problemáticas concretas del sistema. Estas consultas permiten extraer información relevante a partir de los datos almacenados en la base de datos, demostrando así la funcionalidad y utilidad del modelo relacional diseñado. A continuación, se detallan las problemáticas abordadas, junto con la descripción y justificación de cada consulta:

Problemática 1: ¿Cuál es el alumno con mayor cantidad de asistencias durante el mes de mayo?

Se elaboró una consulta que contabiliza la cantidad de asistencias por alumno en clases realizadas en el mes de mayo, filtrando únicamente aquellas en las que se registró asistencia efectiva. Se selecciona el alumno con mayor cantidad

mediante **TOP 1** y orden descendente.

```sql
select top 1 a.legajo, a.apellidoAlu, a.nombreAlu, count(aac.legajo) as asistencias
from Alumnos a
join alumno_asiste_clase aac
on a.legajo = aac.legajo
join Clases c
on aac.idClase = c.idClase
where month(c.fecha) = 5 and aac.estado = 'Asistido'
group by a.legajo, a.apellidoAlu, a.nombreAlu
order by asistencias desc
```

Problemática 2: ¿Qué alumnos rindieron todas las evaluaciones de la materia "Matemática"?
Mediante una subconsulta con **NOT EXISTS**, se identifican aquellos alumnos que hayan rendido todas las evaluaciones correspondientes a la materia "Matemática", evitando registros incompletos.

```sql
select a.legajo, a.nombreAlu, a.apellidoAlu
from Alumnos a
where not exists (
    select e.idEvaluacion
    from Evaluaciones e
    join Materias m on e.#Materia = m.#Materia
    where m.nombre = 'Matemática'
    and not exists (
        select 1
        from alumno_rinde_evaluacion are
        where are.legajo = a.legajo and are.idEvaluacion = e.idEvaluacion
    )
)
```

Problemática 3: Listar los profesores que dictan materias con más de 2 evaluaciones.
Se agrupan los profesores junto con las materias que dictan, y se filtra mediante **HAVING COUNT** aquellos casos en los que la cantidad de evaluaciones supera las dos.

```sql
SELECT DISTINCT p.nombreProf, p.apellidoProf
FROM Profesores p
JOIN profesor_dicta_materia pdm ON p.dniProf = pdm.dniProf
JOIN Materias m ON pdm.#Materia = m.#Materia
JOIN Evaluaciones e ON m.#Materia = e.#Materia
GROUP BY p.dniProf, p.nombreProf, p.apellidoProf
HAVING COUNT(e.idEvaluacion) > 2
```

Problemática 4: Listar todos las evaluaciones con su materia del mes mayo de 2025 y que hayan tenido al menos 2 alumnos con (nota ≥ 6).
Se filtran las evaluaciones por mes y año, y se agrupan por evaluación, verificando que haya al menos dos alumnos con nota igual o superior a 6.

```sql
select e.idEvaluacion, m.nombre
from Evaluaciones e
join alumno_rinde_evaluacion ar
on e.idEvaluacion = ar.idEvaluacion
join Materias m
on e.#materia = m.#materia
where month(e.fecha) = 5 and year(e.fecha) = 2025 and ar.nota >= 6
group by e.idEvaluacion, m.nombre
having count(ar.legajo) >= 2
```

Problemática 5: Listar el nombre de cada alumno y la cantidad de materias que cursó en el año 2024, junto con su promedio de nota final en esas materias. Solo mostrar a los alumnos que cursaron al menos 2 materias.
Se listan los alumnos que hayan cursado como mínimo dos materias durante el año 2024, mostrando además la cantidad de materias cursadas y su promedio de notas finales.

```sql
select a.nombreAlu, count(acm.#materia) as cantidadMaterias, (
    select avg(notaFinal)
    from alumno_cursa_materia acm2
    join Materias m2 on acm2.#materia = m2.#materia
    where acm2.legajo = a.legajo and year(m2.fechaInicio) = 2024) as promedioNotas
from Alumnos a
join alumno_cursa_materia acm on a.legajo = acm.legajo
join Materias m on acm.#materia = m.#materia
where year(m.fechaInicio) = 2024
group by a.legajo, a.nombreAlu
having count(acm.#materia) >= 2;
```

Problemática 6: Cantidad de inscriptos por grupo en el primer semestre de 2025.
Se agrupan las inscripciones por grupo dentro del primer semestre del año 2025 (enero a junio), mostrando la sigla del grupo, la primera fecha de inscripción y la cantidad total de alumnos inscriptos.

```sql
select
    g.sigla, min(aig.fechaInsc) as primeraInscripcion,
    count(aig.legajo) as cantAlumnos
from alumno_inscribe_grupo aig
join Grupos g
on aig.#grupo = g.#grupo
where month(aig.fechaInsc) between 1 and 6 and year(aig.fechaInsc) = 2025
group by g.sigla
order by cantAlumnos desc
```