BASE DE DATOS I

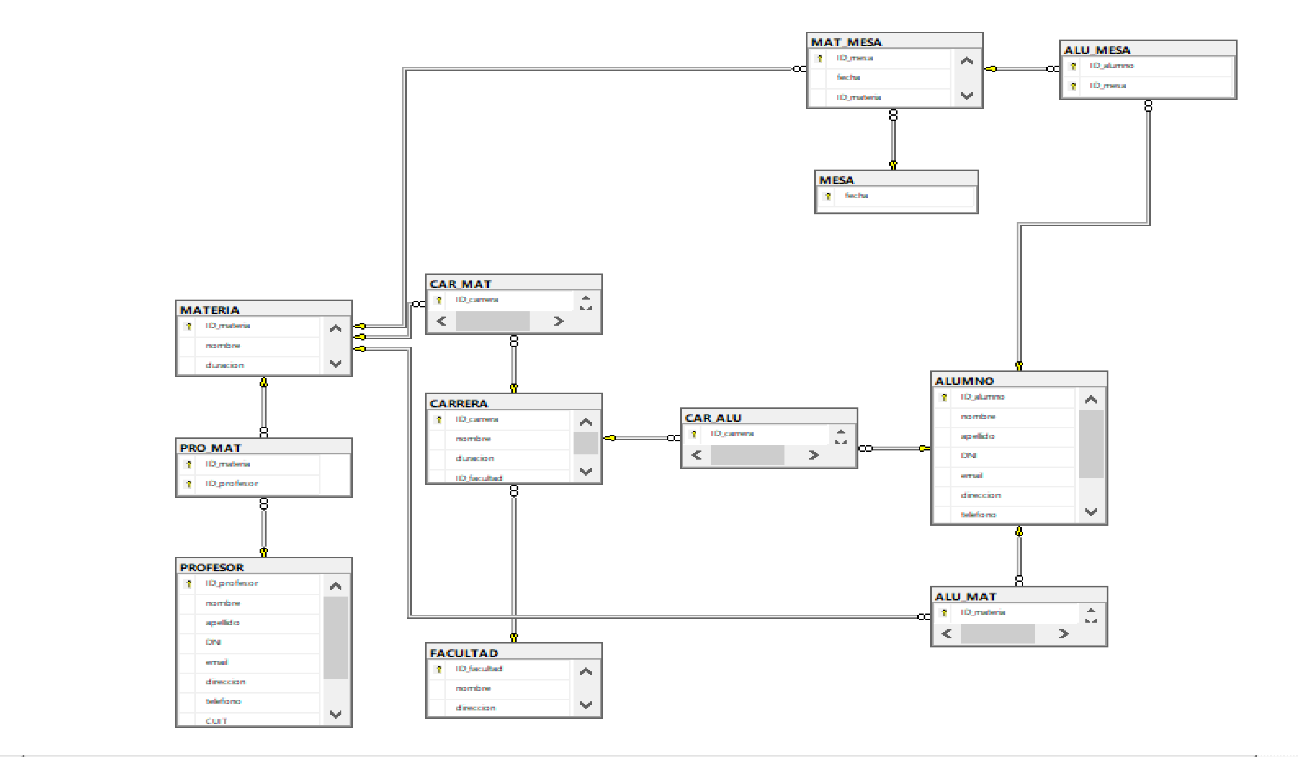
PROYECTO FINAL

## Integrantes:

* Lucas Soria.
* Phillipp Von Kesselstatt.
* Alejandro Marotta.
* Enzo Ponce.
* Franco Santander.

# INTRODUCCION:

Para el proyecto final de la materia diseño de base de datos, hemos creado un modelo de base de datos basado en la gestión de la Universidad de Mendoza. En ella están presentes todas las tablas y relaciones que permiten ingresar con facilidad y agilidad a cualquier tipo de dato que esté cargado en el sistema.



# DESARROLLO:

## PUNTO 2:

#### Forma Normal de nuestra Base de Datos:

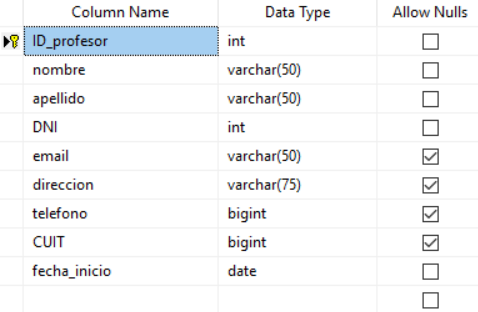
**Primera Forma Normal**: Una tabla está en primera Forma Normal si cumple con las condiciones de afinidad. Estas son: todas las celdas deben tener un valor único (no se aceptan arreglos o listas), todas las celdas de una columna deben ser del mismo tipo, los nombres de las columnas deben ser únicos, el orden de las filas y las columnas no importa y, por último, dos filas no pueden ser iguales.

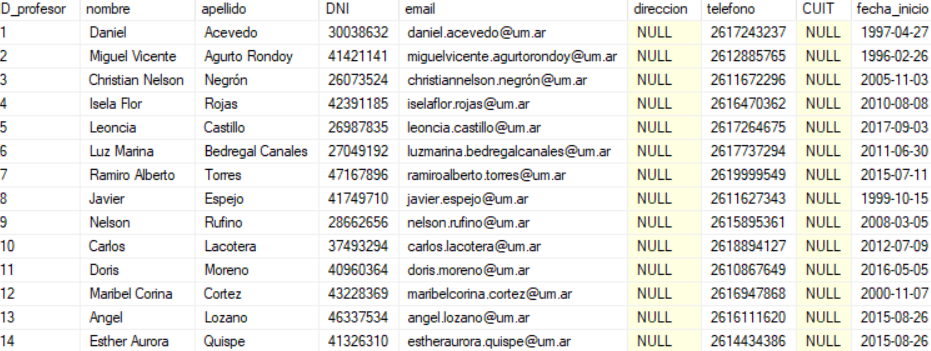
**La Segunda Forma Normal:** Una tabla está en Segunda Forma Normal si previamente esta en Primera Forma Normal y todos los atributos de una fila dependen completamente de la clave primaria de la misma.

**La Tercera Forma Normal:** Una tabla está en Tercera Forma Normal si previamente está en Segunda Forma Normal y no presenta dependencias transitivas. Esto quiere decir, un atributo debe depender únicamente de su clave primaria.

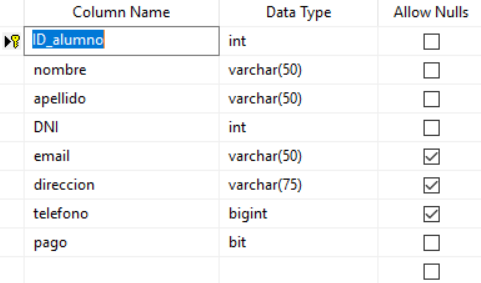
Basándonos en estas definiciones podemos concluir que nuestra base de datos se encuentra normalizada en Tercera Forma Normal, ya que todas las afinidades cumplen con esas condiciones.

**Tabla profesores**



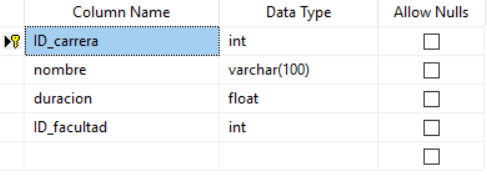


**Tabla Alumno**



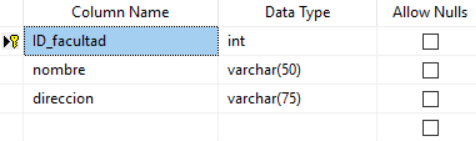


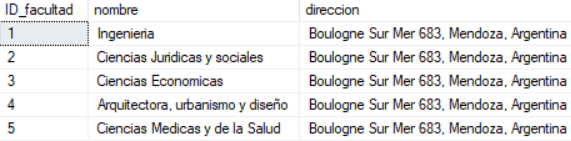
**Tabla Carrera**



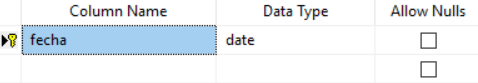


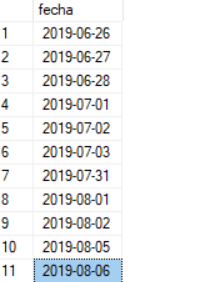
**Tabla Facultad**



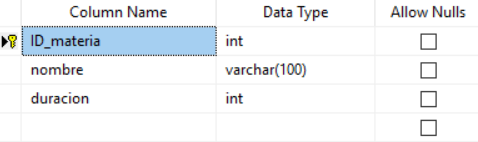


**Tabla Mesa**



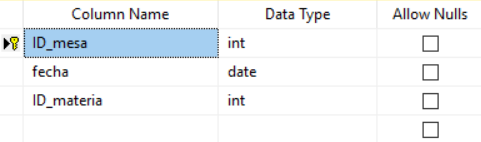


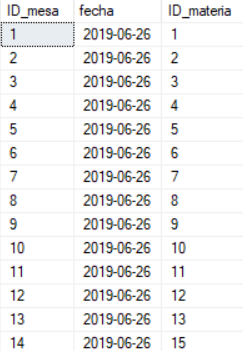
**Tabla Materia**



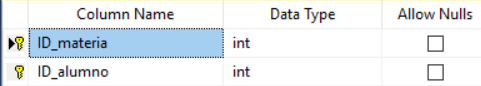


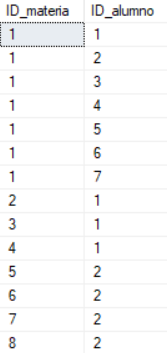
**Tabla Mesa**



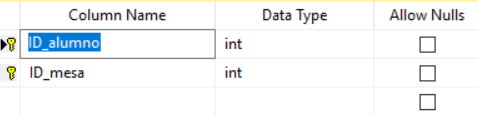


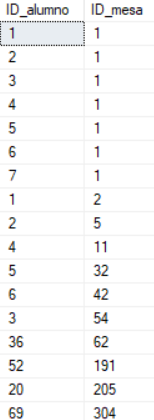
**Tabla Relación Materia Alumno**



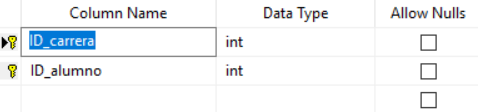


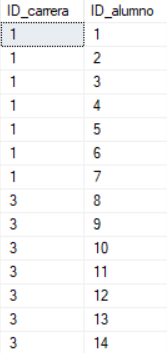
**Tabla Relación Mesa Alumno**



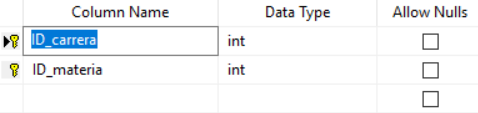


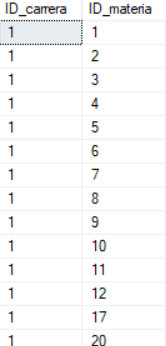
**Tabla Relación Carrera Alumno**



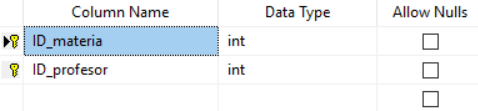


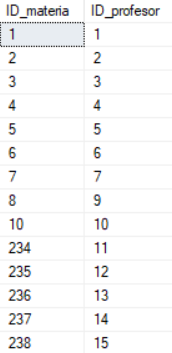
**Tabla Relación Carrera Materia**





**Tabla Relación Materia Profesor**





## PUNTO 3:

#### Consultas:

Consulta 1: ***Muestra la lista de los alumnos que están en una misma materia y no pagaron la cuota***

create procedure regulares\_en\_materias @materiaID int as select ALUMNO.ID\_alumno, ALUMNO.nombre, ALUMNO.apellido, MATERIA.ID\_materia, MATERIA.nombre as nombre\_materia from ALUMNO, MATERIA, ALU\_MAT where ALUMNO.ID\_alumno = ALU\_MAT.ID\_alumno and MATERIA.ID\_materia = ALU\_MAT.ID\_materia and ALUMNO.pago = 0 and MATERIA.ID\_materia = @materiaID;

exec regulares\_en\_materias @materiaID = 1;

Consulta 2: ***Muestra la información de los alumnos que están en una misma materia***

create procedure alumnos\_en\_materias @materiaID int as select ALUMNO.\* from ALUMNO, CAR\_ALU, ALU\_MAT, MATERIA where MATERIA.ID\_materia=ALU\_MAT.ID\_materia AND MATERIA.ID\_materia = @materiaID and Alumno.ID\_alumno = CAR\_ALU.ID\_alumno and ALUMNO.ID\_alumno = ALU\_MAT.ID\_alumno;

exec alumnos\_en\_materias @materiaID = 1;

Consulta 3: ***Muestra todas las materias y su fecha de mesa***

select MATERIA.nombre, MESA.fecha from MATERIA, MESA, MAT\_MESA where MATERIA.ID\_materia = MAT\_MESA.ID\_materia and MESA.fecha = MAT\_MESA.fecha;

Consulta 4: ***Muestra todos los profesores que dan una materia***

create procedure profesores\_en\_materias @materiaID int as select \* from PROFESOR, PRO\_MAT, MATERIA where MATERIA.ID\_materia = @materiaID and PROFESOR.ID\_profesor = PRO\_MAT.ID\_profesor and PRO\_MAT.ID\_materia = MATERIA.ID\_materia;

exec profesores\_en\_materias @materiaID = 239;

Consulta 5: ***Muestra todos los alumnos que están inscriptos en una mesa***

create procedure alumnos\_en\_mesas @materiaID int, @mesa\_fecha date as select ALUMNO.ID\_alumno, ALUMNO.nombre, ALUMNO.apellido, MATERIA.nombre AS nombre\_materia, MESA.fecha from MESA,ALUMNO, ALU\_MESA, MATERIA, MAT\_MESA where MATERIA.ID\_materia = @materiaID AND MESA.fecha = @mesa\_fecha AND MESA.fecha = MAT\_MESA.fecha AND MAT\_MESA.ID\_materia = MATERIA.ID\_materia AND MAT\_MESA.ID\_mesa = ALU\_MESA.ID\_mesa AND ALUMNO.ID\_alumno = ALU\_MESA.ID\_alumno;

exec alumnos\_en\_mesas @materiaID = 1, @mesa\_fecha = '20190626';

Consulta 6: ***Muestra el email de alumnos inscritos en una materia***

create procedure email\_alumnos @materiaID int as select ALUMNO.nombre, ALUMNO.apellido, ALUMNO.email from ALUMNO, ALU\_MAT where ALUMNO.ID\_alumno = ALU\_MAT.ID\_alumno and ALU\_MAT.ID\_materia = @materiaID;

exec email\_alumnos @materiaID = 1;

Consulta 7: ***Muestra el email profesores en una materia***

create procedure email\_profesores @materiaID int as select PROFESOR.nombre, PROFESOR.apellido, PROFESOR.email from PROFESOR, PRO\_MAT where PROFESOR.ID\_profesor = PRO\_MAT.ID\_profesor and PRO\_MAT.ID\_materia = @materiaID;

exec email\_profesores @materiaID = 239;

Consulta 8: ***Muestra todos los alumnos que no pagaron***

select \* from ALUMNO where ALUMNO.pago = 0;

Consulta 9: ***Muestra el id de un alumno especifico***

create procedure alumnoID @alumno\_nombre varchar(50), @alumno\_apellido varchar(50) as select ALUMNO.ID\_alumno from ALUMNO where ALUMNO.nombre = @alumno\_nombre and ALUMNO.apellido = @alumno\_apellido;

exec alumnoID @alumno\_nombre = 'Lucas' , @alumno\_apellido = 'Soria';

Consulta 10: ***Muestra el teléfono de un alumno y materias inscripto***

select ALUMNO.ID\_alumno, ALUMNO.nombre, ALUMNO.apellido, ALUMNO.telefono, MATERIA.nombre from ALUMNO, ALU\_MAT, MATERIA where ALUMNO.ID\_alumno = ALU\_MAT.ID\_alumno and MATERIA.ID\_materia = ALU\_MAT.ID\_materia order by ALUMNO.ID\_alumno;

Consulta 11: ***El nombre de la materia y la fecha de la mesa de esa materia que da un profesor***

create procedure mesa\_profesor @profesorID int as SELECT MATERIA.nombre,MESA.fecha FROM PROFESOR,MESA,MAT\_MESA,MATERIA,PRO\_MAT WHERE PROFESOR.ID\_profesor=PRO\_MAT.ID\_profesor AND PRO\_MAT.ID\_materia=MATERIA.ID\_materia AND PROFESOR.ID\_profesor= @profesorID AND MATERIA.ID\_materia=MAT\_MESA.ID\_materia and MAT\_MESA.fecha=MESA.fecha;

exec mesa\_profesor @profesorID= 11;

## PUNTO 4:

### TRIGGERS:

***TRIGGER QUE CREA EL MAIL DEL ALUMNO:***

CREATE TRIGGER create\_mail\_alumno ON ALUMNO AFTER INSERT AS update ALUMNO SET ALUMNO.email = lower(left(ALUMNO.Nombre, 1)) + '.' + replace(lower(ALUMNO.Apellido), ' ', '') + '@alumno.um.ar';

***TRIGGER QUE CREA EL MAIL DEL PROFESOR:***

CREATE TRIGGER create\_mail\_profesor ON PROFESOR AFTER INSERT AS update PROFESOR SET email = replace(lower(PROFESOR.nombre), ' ' , '' ) + '.' + replace(lower(PROFESOR.apellido), ' ', '') + '@um.ar';

### STORED PROCEDURE:

***STORED PROCEDURE QUE HACE UNA CONSULTA DE LOS DATOS DEL PROFESOR Y TAMBIEN LE CALCULA SU ANTIGÜEDAD:***

CREATE PROCEDURE antiguedad AS SELECT \*, DATEDIFF(YEAR, PROFESOR.fecha\_inicio, SYSDATETIME()) AS antiguedad FROM PROFESOR;

## PUNTO 5:

#### POLITICA DE BACKUP:

Para nuestra base de datos, decidimos aplicar una política de back up: full back up, el cual se haría todos los fines de semana. Y además, los días de semana complementaríamos el full back up con un differencial back up.

#### ¿Por qué tomamos esta decisión?

Se elegio la política de full back up porque los datos de la Universidad son de gran importancia y no deben ser perdidos. Por ejemplo, la inscripción a un examen de un Alumno. Si esta información se pierde, dicho alumno no tendría permitido rendir ese examen.

La política differencial back up fue elegida porque hacer un full back up toma tiempo y no puede hacerse todos los días, por esta razón esta política de back up complementa a la otra durante los días de semana.

Al mantener estas políticas, si sucede un imprevisto durante la semana, podría restaurarse la base de datos a un instante en el tiempo donde se sabe que los datos no presentan problemas.