Trabajo Práctico N°3

Grupo 9. Integrantes:

- Farias Erika Aldana, 1207129
- Lee Hyo Lin, 1203815
- Marquez Diego Gabriel, 1127988

Materia: Ingeniería de Datos I

UADE

1. Análisis de dependencias funcionales

1.1 Identificar al menos cinco dependencias funcionales reales del modelo.

Dependencia Funcional 1:

```
Alumno.legajo → {nombre, apellido, usuario, documento, correo_educativo, correo_personal,  fecha_nacimiento, direccion, telefono, plan_carrera}
```

Dependencia Funcional 2:

```
Materia.codigo → {nombre, obligatoria, carga_horaria}
```

Dependencia Funcional 3:

```
Plan_Carrera.codigo → {nombre_plan, codigo_carrera}
```

Dependencia Funcional 4:

```
Curso.codigo  →  {codigo_materia,  legajo_docente,  anio,  cuatrimestre, horario, modalidad}
```

Dependencia Funcional 5:

```
(Cursada.legajo_alumno, Cursada.codigo_curso) → condicion
```

1.2 Clasificarlas como completas, parciales o transitivas y justificarlas.

Dependencia Funcional 1:

Tipo: Múltiple

Justificación: Un solo atributo (legajo) determina múltiples atributos de la tabla Alumno.

Dependencia Funcional 2:

Tipo: Múltiple

Justificación: El código de materia determina varios atributos (nombre, obligatoria y carga_horaria) de forma simultánea.

Dependencia Funcional 3:

Tipo: Simple

Justificación: El código del plan determina dos atributos relacionados.

Dependencia Funcional 4:

Tipo: Múltiple

Justificación: Un único atributo (código de curso) determina varios atributos que describen completamente la oferta académica del curso.

Dependencia Funcional 5:

Tipo: Compuesta

Justificación: Se necesita la combinación de dos atributos (legajo del alumno Y código del curso) para determinar la condición de cursada. Ninguno de los dos por separado puede determinarla.

1.3 Calcular el cierre de un conjunto de atributos (A^+).

Cierre 1: {legajo}$^+$

1. {legajo}$^+$ = {legajo}
2. Aplicamos legajo → {nombre, apellido, usuario, documento, correo_educativo, correo_personal, fecha_nacimiento, direccion, telefono, plan_carrera} entonces

```
{legajo}⁺ = {legajo, nombre, apellido, usuario, documento, correo_educativo, correo_personal, fecha_nacimiento, direccion, telefono, plan_carrera}
```

3. No hay más DFs aplicables (plan_carrera ≠ codigo_plan)
4. **Resultado**: {legajo} es clave candidata de Alumno.

Cierre 2: {codigo_curso}$^+$

1. {codigo_curso}$^+$ = {codigo_curso}
2. Aplicamos codigo_curso → {codigo_materia, legajo_docente, anio, cuatrimestre, horario, modalidad}, entonces

```
{codigo_curso}⁺ = {codigo_curso, codigo_materia, legajo_docente, anio, cuatrimestre, horario, modalidad}
```

3. Aplicamos codigo_materia → {nombre, obligatoria, carga_horaria}, entonces

```
{codigo_curso}⁺ = {codigo_curso, codigo_materia, legajo_docente, anio, cuatrimestre, horario, modalidad, nombre, obligatoria, carga_horaria}
```

4. **Resultado**: {codigo_curso} es clave candidata de Curso. Hay dependencia transitiva: codigo_curso → codigo_materia → {nombre, obligatoria, carga_horaria}

Cierre 3: {legajo_alumno, codigo_curso}⁺

1. {legajo_alumno, codigo_curso}⁺ = {legajo_alumno, codigo_curso}
2. Aplicamos (legajo_alumno, codigo_curso) → condicion, entonces

```
{legajo_alumno, codigo_curso}⁺ = {legajo_alumno, codigo_curso, condicion}
```

3. **Resultado**: {legajo_alumno, codigo_curso} es clave compuesta de Cursada.

1.4 Determinar las claves candidatas de al menos una relación principal.

Relación analizada: Alumno

La tabla Alumno tiene la siguiente estructura:

```
Alumno(legajo, usuario, documento, nombre, apellido, correo_personal, correo_educativo, fecha_nacimiento, direccion, telefono, plan_carrera)
```

Clave candidata Alumno:
Si calculamos el cierre de legajo obtenemos todos los atributos de la tabla Alumno.
Significa que conociendo el legajo de un alumno podemos determinar unívocamente todos sus datos personales, de contacto y académicos. Legajo es clave candidata y fue elegida como clave primaria de la tabla.

Clave candidata Documento:
El documento (DNI) identifica de manera única a una persona.
Si conocemos el documento de un alumno, podemos determinar quién es y por ende todos sus demás atributos. El cierre de documento incluye todos los atributos de Alumno, documento es clave candidata.

Clave candidata Usuario:
El usuario es único dentro del sistema académico.
Conociendo el usuario podemos identificar al alumno y determinar todos sus atributos. Por lo tanto usuario también es clave candidata.

2. Proceso de normalización

En nuestro modelo previo, Cursada tiene clave compuesta {legajo_alumno, codigo_curso}; Curso posee clave {codigo} y una clave única (UK) por oferta {codigo_materia, legajo_docente, anio, cuatrimestre, horario, modalidad}; Alumno está identificado por {legajo}.
La normalización se centró en tres tablas: Cursada (principal), Curso y Prestamos_Libros (auxiliares).

2.1. Tabla principal: Cursada

En Cursada se quitaron los datos que en realidad pertenecen al alumno (nombre, apellido, plan) y al curso (materia, docente, año, cuatrimestre, etc.). Esos campos pasan a las tablas Alumno, Curso, Materia y Docente, y Cursada se queda solo con lo que depende de toda la clave {legajo_alumno, codigo_curso} (por ejemplo, condición y nota). Así se evita repetir la misma información muchas veces y se reducen errores al actualizar.

Estructura inicial (no normalizada – hipotética)

Partimos de una versión más “cargada” de la tabla, que mezclaba datos del alumno, del curso y de la cursada:

```
Cursada(legajo_alumno, codigo_curso, nombre_alumno, apellido_alumno, plan_carrera, codigo_materia, nombre_materia, legajo_docente, nombre_docente, anio, cuatrimestre, horario, modalidad, condicion, nota_final)
```

Dependencias funcionales detectadas

La clave candidata de la tabla es {legajo_alumno, codigo_curso}.

- {legajo_alumno, codigo_curso} → condicion, nota_final (datos propios de la cursada).
- legajo_alumno → nombre_alumno, apellido_alumno, plan_carrera (atributos del alumno).
- codigo_curso → codigo_materia, anio, cuatrimestre, horario, modalidad, legajo_docente (atributos del curso).
- codigo_materia → nombre_materia (atributo de la materia).
- legajo_docente → nombre_docente (atributo del docente).

1FN

Los atributos son atómicos y no hay grupos repetidos.
Si horario tuviera múltiples franjas, se separaría (por ejemplo: dia_semana, hora_inicio, hora_fin). En nuestro caso, mantenemos un único valor atómico para el horario, por lo que la tabla cumple 1FN.

2FN (eliminación de dependencias parciales)

Hay atributos no clave que dependen solo de una parte de la clave compuesta:

- De legajo_alumno dependen nombre_alumno, apellido_alumno, plan_carrera.
- De codigo_curso dependen codigo_materia, anio, cuatrimestre, horario, modalidad, legajo_docente.

Descomposición

Mantener en Cursada solo lo que depende de toda la clave:

`Cursada(legajo_alumno, codigo_curso, condicion, nota_final)`
PK: `(legajo_alumno, codigo_curso)`

- Los atributos del alumno y del curso se obtienen mediante claves foráneas hacia Alumno(legajo, …) y Curso(codigo, …).

3FN (eliminación de dependencias transitivas)

- nombre_materia depende de codigo_materia (que depende de codigo_curso).
- nombre_docente depende de legajo_docente (que depende de codigo_curso).

Acción: no almacenar nombre_materia ni nombre_docente en Cursada; se obtienen transitivamente vía las claves foráneas de Curso hacia Materia y Docente.

Resultado 3FN final

`Cursada(legajo_alumno FK→Alumno, codigo_curso FK→Curso, condicion, nota_final)`
PK: `(legajo_alumno, codigo_curso)`

Curso y Alumno proveen el resto de los datos mediante joins.

Redundancias eliminadas / anomalías evitadas

Por el proceso de normalización se reducen varias anomalías que podían aparecer en la tabla de Cursada:

- Anomalía de actualización: si cambia el apellido de un alumno, se modifica solo en Alumno y no en todas las filas de Cursada.
- Anomalía de inserción: es posible dar de alta a un alumno en el sistema aunque aún no esté inscripto en ningún curso, porque sus datos personales se almacenan en Alumno.
- Anomalía de borrado: al eliminar una cursada determinada, no se pierden datos importantes como el nombre de la materia o del docente, que permanecen en Materia y Docente y no dependen de esa inscripción en particular.

2.2. Tabla auxiliar: Curso

En Curso se eliminaron nombre_materia y nombre_docente, porque dependen de codigo_materia y legajo_docente y no directamente de la clave codigo. Esos nombres ahora se obtienen desde Materia y Docente, lo que reduce duplicación y asegura que un cambio de nombre se haga en un solo lugar.

Estructura inicial (no normalizada – hipotética)

```
Curso(codigo, codigo_materia, nombre_materia, legajo_docente, nombre_docente, anio, cuatrimestre, horario, modalidad)
```

Dependencias funcionales detectadas

- codigo → codigo_materia, nombre_materia, legajo_docente, nombre_docente, anio, cuatrimestre, horario, modalidad (PK simple).
- codigo_materia → nombre_materia.
- legajo_docente → nombre_docente.
- La clave primaria es codigo.

1FN

La tabla Curso cumple 1FN porque todos los atributos son atómicos: cada fila representa una única oferta de curso, con una sola materia, un solo docente, un año, cuatrimestre, horario y modalidad específicos. No hay grupos repetidos ni listas de valores en una misma celda.

2FN

La clave primaria es simple (codigo), por lo que no existen dependencias parciales: todos los atributos no clave dependen funcionalmente de codigo. En consecuencia, la tabla satisface 2FN sin necesidad de descomposición.

3FN (eliminación de dependencias transitivas)

Se identifican dependencias transitivas:

- nombre_materia depende de codigo_materia.
- nombre_docente depende de legajo_docente.

Ambos (codigo_materia y legajo_docente) son atributos no clave. Para alcanzar 3FN se eliminan de Curso los atributos que dependen de ellos y se obtienen desde las tablas correspondientes.

Resultado 3FN final

```
Curso(codigo, codigo_materia, legajo_docente, anio, cuatrimestre, horario, modalidad)
```

Los nombres de materia y docente se consultan en Materia y Docente.

Redundancias eliminadas / anomalías evitadas

No se duplica nombre_materia / nombre_docente por cada curso/oferta.

Cambiar el nombre de una materia o de un docente no exige actualizar múltiples filas en Curso: basta con modificar una sola vez en la tabla correspondiente, lo que mejora la consistencia de los datos.

2.3. Tabla auxiliar: Prestamos_Libros

En Prestamos_Libros no fue necesario cambiar la estructura, porque ya cumplía con 3FN: la clave id determina a todos los demás campos y no hay dependencias parciales ni transitivas. Solo se refuerza la idea de usar claves foráneas (Libro, Alumno, Docente) y una regla para que el préstamo sea de un alumno o de un docente, pero no de ambos.

Estructura inicial

```
Prestamos_Libros(id, isbn, legajo_alumno, legajo_docente, fecha_prestamo, fecha_devolucion)
```

Dependencias funcionales detectadas

- id → isbn, legajo_alumno, legajo_docente, fecha_prestamo, fecha_devolucion (PK surrogate).
- No hay determinaciones entre atributos no clave; el XOR “alumno o docente” se maneja con una restricción de dominio (por ejemplo, un CHECK que obliga a que uno de los legajos sea nulo).

1FN

Todos los atributos son atómicos: cada fila registra un único préstamo, con un solo libro (isbn), una sola persona asociada (alumno o docente) y una única fecha de préstamo y de devolución. No hay listas de valores dentro de una celda ni grupos de columnas repetidos. Por lo tanto, la tabla cumple 1FN.

2FN

La clave primaria es simple (id). Al no tratarse de una clave compuesta, no puede haber dependencias parciales: todos los demás campos (isbn, legajo_alumno, legajo_docente, fecha_prestamo, fecha_devolucion) dependen funcionalmente de id y de ningún subconjunto de la clave. La tabla satisface 2FN.

3FN

Se revisa si existen dependencias transitivas entre atributos no clave, es decir, si algún atributo no clave determina a otro atributo no clave dentro de la misma tabla. En Prestamos_Libros no se detectan este tipo de relaciones: ninguno de los atributos no clave determina a otro (por ejemplo, isbn no determina las fechas ni los legajos, y las fechas no determinan a isbn ni a los legajos). En consecuencia, la tabla ya se encuentra en 3FN con la estructura propuesta.

Resultado 3FN final

```
Prestamos_Libros(id, isbn, legajo_alumno, legajo_docente, fecha_prestamo, fecha_devolucion)
```

Redundancias eliminadas / anomalías evitadas

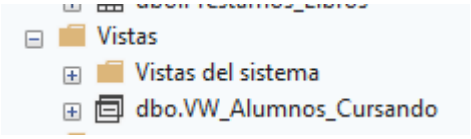
Al mantener la tabla en 3FN, cada préstamo queda representado en una sola fila sin repetir datos descriptivos de libro ni de persona. Los datos del libro y de los usuarios se guardan en tablas separadas (Libro, Alumno, Docente), lo que evita duplicaciones. Además, las claves foráneas y la regla tipo XOR garantizan que cada préstamo esté asociado correctamente a un solo tipo de usuario, preservando la integridad y coherencia de los datos.

3. SQL avanzado y seguridad

3.1 Vista

```sql
CREATE OR ALTER VIEW VW_Alumnos_Cursando
AS
SELECT
    DISTINCT a.legajo,
    a.nombre,
    a.apellido,
    a.plan_carrera,
    a.correo_educativo,
    c.anio,
    c.cuatrimestre
FROM Alumno a
JOIN Cursada cu ON cu.legajo_alumno = a.legajo
JOIN Curso c ON c.codigo = cu.codigo_curso
WHERE
    c.anio = YEAR(GETDATE())
    AND c.cuatrimestre = CASE
                            WHEN MONTH(GETDATE()) BETWEEN 1 AND 6 THEN 1
                            ELSE 2
                         END;
GO
```

La vista **VW_Alumnos_Cursando** concentra la información de alumnos activos, permitiendo identificar rápidamente quiénes están cursando al menos una materia en el cuatrimestre vigente. Es útil para reportes, seguimiento académico y decisiones administrativas. Se incluyó el correo electrónico y el plan para agilizar el envío de comunicaciones, evitando a alumnos inactivos o graduados.



```
1
2   SELECT TOP (10) *
3     FROM VW_Alumnos_Cursando;
4
```

100 % ✓ No se encontraron problemas.

Resultados | Mensajes

	legajo	nombre	apellido	plan_carrera	correo_educativo	anio	cuatrimestre
1	3	Matías	Torres	2	mtorres@uade.edu.ar	2025	2
2	5	Daniel	Vargas	6	dvargas@uade.edu.ar	2025	2
3	6	Camila	Ruiz	1	cruiz@uade.edu.ar	2025	2
4	8	Valentina	Díaz	6	vdiaz@uade.edu.ar	2025	2
5	12	Julieta	Molina	6	jmolina@uade.edu.ar	2025	2

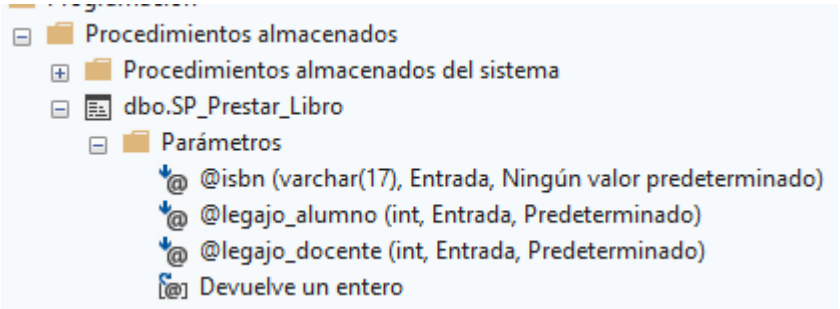
3.2 Procedimiento almacenado (Stored Procedure)

```
CREATE PROCEDURE SP_Prestar_Libro(
    @isbn VARCHAR(17),
    @legajo_alumno INT = NULL,
    @legajo_docente INT = NULL)
AS
BEGIN
    IF NOT (
        (@legajo_alumno IS NOT NULL AND @legajo_docente IS NULL) OR
        (@legajo_alumno IS NULL AND @legajo_docente IS NOT NULL)
    )
    BEGIN
        RAISERROR('Debe enviar legajo_alumno O legajo_docente, pero no
ambos ni ninguno.', 16, 1);
        RETURN;
    END;

    BEGIN TRY
        BEGIN TRAN;
        --Valida existencia de isbn ingresado
        IF NOT EXISTS (SELECT 1 FROM Libro WHERE isbn = @isbn)
        BEGIN
            RAISERROR('El ISBN indicado no existe en la tabla Libro.',
16, 1);
            ROLLBACK TRAN;
            RETURN;
        END;
        --Valida disponibilidad de ese isbn
        IF NOT EXISTS (SELECT 1 FROM Libro WHERE isbn = @isbn AND
disponibles > 0)
```

```
        BEGIN
            RAISERROR('No hay ejemplares disponibles para este isbn',
16, 1);
            ROLLBACK TRAN;
            RETURN;
        END;
        --Valida que el alumno o el docente existan
        IF @legajo_alumno IS NOT NULL
        BEGIN
            IF NOT EXISTS (SELECT 1 FROM Alumno WHERE legajo =
@legajo_alumno)
            BEGIN
                RAISERROR('El legajo alumno indicado no existe', 16, 1);
                ROLLBACK TRAN;
                RETURN;
            END;
        END
        ELSE
        BEGIN
            IF NOT EXISTS (SELECT 1 FROM Docente WHERE legajo =
@legajo_docente)
            BEGIN
                RAISERROR('El legajo docente indicado no existe', 16,
1);
                ROLLBACK TRAN;
                RETURN;
            END;
        END

        --Inserta registro en Prestamos_Libros con la información
indicada por parámetro y fecha de hoy
        INSERT INTO Prestamos_Libros (isbn, legajo_alumno,
legajo_docente, fecha_prestamo)
        VALUES (@isbn, @legajo_alumno, @legajo_docente, CAST(GETDATE()
AS DATE));

        --Resta uno a la cantidad de ejemplares disponibles para ese
isbn
        UPDATE Libro
        SET disponibles = disponibles - 1
        WHERE isbn = @isbn;

        COMMIT TRAN;
    END TRY

    BEGIN CATCH
```

```
        IF @@TRANCOUNT > 0
            ROLLBACK TRAN;
        DECLARE @msg NVARCHAR(4000) = ERROR_MESSAGE();
        RAISERROR('Error interno: %s', 16, 1, @msg);
    END CATCH
END;
GO
```

El procedimiento almacenado **SP_Prestar_Libro** se utiliza para registrar correctamente en las tablas un préstamo de un libro a un docente o alumno. Es relevante su uso por las validaciones necesarias y la encadenación de operaciones que involucran la inserción en Prestamos_Libros y la actualización del stock en Libro.
En este procedimiento se valida cada paso del proceso: verificar que solo se ingrese un alumno o docente y que éste sea válido, que el isbn exista y que la cantidad de ejemplares disponibles sea mayor a 0. Se utilizaron estructuras como TRY-CATCH y transacciones (TRAN) para el manejo de errores. En caso de producirse algún inconveniente, el bloque CATCH revierte todas las operaciones mediante ROLLBACK y genera un mensaje claro mediante RAISERROR. De esta manera, se evita que queden operaciones incompletas (por ejemplo, un INSERT sin su correspondiente UPDATE), asegurando la integridad de los datos y previniendo inconsistencias.



Préstamos_Libros antes de ejecutar el SP:

	id	isbn	legajo_alumno	legajo_docente	fecha_prestamo	fecha_devolucion
1	1	978-950-508-123-4	1	NULL	2025-03-10	2025-03-24
2	2	978-987-678-234-5	2	NULL	2025-03-15	2025-03-29
3	3	978-950-123-345-6	NULL	1	2025-03-20	2025-04-03
4	4	978-987-456-456-7	3	NULL	2025-04-01	NULL
5	5	978-950-789-567-8	4	NULL	2025-04-05	2025-04-19
6	6	978-987-234-678-9	5	NULL	2025-04-10	NULL
7	7	978-950-567-789-0	NULL	2	2025-04-12	2025-04-26
8	8	978-987-890-890-1	6	NULL	2025-04-15	2025-04-29
9	9	978-950-345-901-2	NULL	3	2025-04-18	NULL
10	10	978-950-123-345-6	10	NULL	2025-05-05	2025-05-19

Libros antes de ejecutar el SP:

Resultados | Mensajes

	isbn	nombre	autores	edicion	editorial	formato	disponibles
1	978-950-123-345-6	Análisis Matemático para Ingenieros	Fernández, Carlos	5ta	Planeta	FIS	4
2	978-950-345-901-2	Gestión de Proyectos Ágiles	Silva, Mónica	1ra	Planeta	DIG	15
3	978-950-508-123-4	Fundamentos de Programación	López, Juan Carlos	3ra	Pearson	FIS	5
4	978-950-567-789-0	Estadística Aplicada a los Negocios	Torres, Verónica	2da	McGraw-Hill	FIS	6
5	978-950-789-567-8	Marketing en la Era Digital	Rodríguez, Laura	1ra	Planeta	DIG	8
6	978-987-234-678-9	Programación Orientada a Objetos en Java	Gómez, Diego	3ra	Pearson	FIS	2
7	978-987-456-456-7	Contabilidad General	Martínez, Ana	4ta	El Ateneo	DIG	10
8	978-987-678-012-3	Derecho Empresarial Argentino	Moreno, Pablo	6ta	La Ley	FIS	4
9	978-987-678-234-5	Base de Datos: Teoría y Práctica	García, María; Pérez, Roberto	2da	McGraw-Hill	FIS	3
10	978-987-890-890-1	Sistemas de Información Gerencial	Castro, Fernando	4ta	Prentice Hall	DIG	12

Ejecutamos SP con valores: isbn = N'978-987-234-678-9', legajo_alumno = 6, legajo_docente = NULL

Parámetro	Tipo de datos	Parámetro de sal...	Pasar valor NULL	Valor
@isbn	varchar(17)	No	☐	978-987-234-678-9
@legajo_alumno	int	No	☐	6
@legajo_docente	int	No	☑	

Tabla Prestamos_Libros

Resultados | Mensajes

	id	isbn	legajo_alumno	legajo_docente	fecha_prestamo	fecha_devolucion
1	1	978-950-508-123-4	1	NULL	2025-03-10	2025-03-24
2	2	978-987-678-234-5	2	NULL	2025-03-15	2025-03-29
3	3	978-950-123-345-6	NULL	1	2025-03-20	2025-04-03
4	4	978-987-456-456-7	3	NULL	2025-04-01	NULL
5	5	978-950-789-567-8	4	NULL	2025-04-05	2025-04-19
6	6	978-987-234-678-9	5	NULL	2025-04-10	NULL
7	7	978-950-567-789-0	NULL	2	2025-04-12	2025-04-26
8	8	978-987-890-890-1	6	NULL	2025-04-15	2025-04-29
9	9	978-950-345-901-2	NULL	3	2025-04-18	NULL
10	10	978-950-123-345-6	10	NULL	2025-05-05	2025-05-19
11	11	978-987-234-678-9	6	NULL	2025-11-15	NULL

Tabla Libros:

Resultados | Mensajes

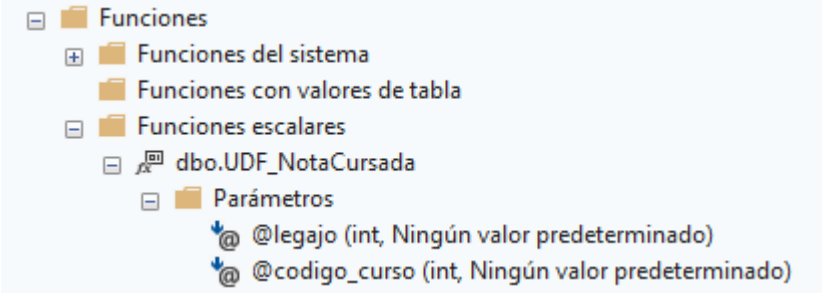
	isbn	nombre	autores	edicion	editorial	formato	disponibles
1	978-950-123-345-6	Análisis Matemático para Ingenieros	Fernández, Carlos	5ta	Planeta	FIS	4
2	978-950-345-901-2	Gestión de Proyectos Ágiles	Silva, Mónica	1ra	Planeta	DIG	15
3	978-950-508-123-4	Fundamentos de Programación	López, Juan Carlos	3ra	Pearson	FIS	5
4	978-950-567-789-0	Estadística Aplicada a los Negocios	Torres, Verónica	2da	McGraw-Hill	FIS	6
5	978-950-789-567-8	Marketing en la Era Digital	Rodríguez, Laura	1ra	Planeta	DIG	8
6	978-987-234-678-9	Programación Orientada a Objetos en Java	Gómez, Diego	3ra	Pearson	FIS	1
7	978-987-456-456-7	Contabilidad General	Martínez, Ana	4ta	El Ateneo	DIG	10
8	978-987-678-012-3	Derecho Empresarial Argentino	Moreno, Pablo	6ta	La Ley	FIS	4
9	978-987-678-234-5	Base de Datos: Teoría y Práctica	García, María; Pérez, Roberto	2da	McGraw-Hill	FIS	3
10	978-987-890-890-1	Sistemas de Información Gerencial	Castro, Fernando	4ta	Prentice Hall	DIG	12

3.3 Función definida por el usuario (UDF)

```
CREATE OR ALTER FUNCTION UDF_NotaCursada
(
    @legajo INT,
    @codigo_curso INT
)
RETURNS DECIMAL(5,2)
AS
BEGIN
    DECLARE @nota_reg DECIMAL(5,2);

--Verificar si para cada parcial desaprobado hay un recuperatorio
aprobado, caso contrario devuelve NULL
    IF EXISTS (
        SELECT 1
        FROM Examen par
        WHERE par.legajo_alumno = @legajo
          AND par.codigo_curso = @codigo_curso
          AND par.tipo = 'PAR'
          AND par.nota < 4
          AND NOT EXISTS (
                SELECT 1
                FROM Examen rec
                WHERE rec.legajo_alumno = par.legajo_alumno
                  AND rec.codigo_curso = par.codigo_curso
                  AND rec.tipo = 'REC'
                  AND rec.nota >= 4
          )
    )
        RETURN NULL;

--Toma notas parciales y de recuperatorios >= 4 para calcular el
```

```
promedio de la cursada
    SELECT @nota_reg = AVG(CAST(nota AS DECIMAL(5,2)))
    FROM Examen
    WHERE legajo_alumno = @legajo
      AND codigo_curso = @codigo_curso
      AND (
            (tipo = 'PAR' AND nota >= 4) OR
            (tipo = 'REC' AND nota >= 4)
          );

    RETURN @nota_reg;
END;
GO
```

La función **UDF_NotaCursada** tiene como objetivo determinar la nota final de cursada (regularización) de un alumno en un curso a partir de los exámenes parciales (Examen de tipo PAR) y recuperatorios (Examen de tipo REC).
Un alumno regulariza si todos los parciales tienen nota mayor o igual a 4 o bien, para cada parcial desaprobado, existe un recuperatorio con nota aprobada (también ≥ 4). Si se cumple esta condición, se calcula la nota de cursada como el promedio de notas entre parciales y recuperatorios (ambos aprobados) con la función AVG. Caso contrario, la función retorna NULL, indicando que la cursada no fue regularizada.
Su uso permite simplificar la elaboración del historial académico del alumno así como otros reportes.



Aplicación de la función UDF_NotaCursada en una consulta para determinar la nota de cursada de los alumnos que se encuentran en estado de regularizar la materia:

```sql
SELECT
    cu.legajo_alumno,
    cu.codigo_curso,
    CASE
        WHEN dbo.UDF_NotaCursada(cu.legajo_alumno, cu.codigo_curso) IS NULL
            THEN '-'
        ELSE CAST(dbo.UDF_NotaCursada(cu.legajo_alumno, cu.codigo_curso) AS VARCHAR(7))
    END AS nota_regular
FROM Cursada cu;
```

100 % 2 0

Resultados Mensajes

	legajo_alumno	codigo_curso	nota_regular
1	1	1	8.00
2	1	2	6.00
3	1	3	9.00
4	2	1	7.00
5	2	2	8.00
6	3	1	-
7	3	2	-
8	3	4	-
9	3	5	-
10	3	7	-
11	3	8	-
12	3	9	-
13	4	1	10.00
14	4	6	7.00
15	5	5	-
16	6	1	-
17	6	4	6.00
18	7	7	-
19	8	5	-
20	9	1	8.00
21	9	2	-
22	10	10	-
23	11	1	-
24	12	5	-
25	14	10	-
26	15	12	-

3.4 Triggers

```sql
CREATE TRIGGER TR_Cursada_Desaprobada
ON Examen
AFTER INSERT
AS
BEGIN
    --Verifica que ya haya por lo menos un examen parcial para esta
cursada
    IF NOT EXISTS (SELECT 1 FROM inserted WHERE tipo = 'PAR')
        RETURN;
```

```
    --Si encuentra más de un parcial desaprobado (nota < 4) considera a
la Cursada como desaprobada (DES)
    UPDATE C
    SET C.condicion = 'DES'
    FROM Cursada C
    INNER JOIN inserted I
        ON C.legajo_alumno = I.legajo_alumno
       AND C.codigo_curso = I.codigo_curso
    WHERE I.tipo = 'PAR'
      AND (
            SELECT COUNT(*)
            FROM Examen E
            WHERE E.legajo_alumno = I.legajo_alumno
              AND E.codigo_curso = I.codigo_curso
              AND E.tipo = 'PAR'
              AND E.nota < 4
          ) > 1;
END;
GO
```

El TRIGGER **TR_Cursada_Desaprobada** es de tipo AFTER, es decir, se ejecuta luego de cierta condición, en este caso al realizar un INSERT en la tabla Examen. Valida si la cursada (relación de Alumno-Curso) tiene más de un examen parcial desaprobado y modifica la condición de la Cursada automáticamente a DES (desaprobada). Esto se debe a que sólo es posible la recuperación de un sólo examen parcial.

Alumno de legajo 6, código curso 1, con un examen parcial desaprobado

	id	legajo_alumno	codigo_curso	tipo	nota	fecha
1	1	1	1	PAR	8	2025-04-15
2	2	1	1	FIN	7	2025-07-10
3	3	1	2	PAR	6	2025-05-20
4	4	1	3	PAR	9	2025-04-25
5	5	1	3	FIN	8	2025-07-15
6	6	2	1	PAR	7	2025-04-15
7	7	2	2	PAR	8	2025-05-20
8	8	2	2	FIN	9	2025-07-12
9	9	4	1	PAR	10	2025-04-15
10	10	4	6	PAR	7	2025-04-10
11	11	4	6	FIN	8	2025-07-08
12	12	6	1	PAR	3	2025-04-15
13	13	6	4	PAR	6	2025-05-18
14	14	9	1	PAR	8	2025-04-15
15	15	9	1	FIN	7	2025-07-10

Condicion de cursada para este alumno-curso CUR (cursando):

	legajo_alumno	codigo_curso	condicion
1	1	1	APR
2	1	2	REG
3	1	3	APR
4	2	1	REG
5	2	2	APR
6	3	1	CUR
7	3	2	CUR
8	3	4	CUR
9	3	5	CUR
10	3	7	CUR
11	3	8	CUR
12	3	9	CUR
13	4	1	PRO
14	4	6	APR
15	5	5	CUR
16	6	1	CUR
17	6	4	CUR
18	7	7	CUR
19	8	5	REG
20	9	1	APR
21	9	2	CUR
22	10	10	CUR
23	11	1	REG
24	12	5	CUR
25	14	10	CUR
26	15	12	CUR

INSERT de un segundo Examen de tipo PAR, con nota menor a 4:

```
8
9   INSERT INTO Examen (legajo_alumno, codigo_curso, tipo, nota, fecha) VALUES (6, 1, 'PAR', 2, '2025-04-29');
```

100 % No se encontraron problemas.

Resultados Mensajes

	id	legajo_alumno	codigo_curso	tipo	nota	fecha
1	1	1	1	PAR	8	2025-04-15
2	2	1	1	FIN	7	2025-07-10
3	3	1	2	PAR	6	2025-05-20
4	4	1	3	PAR	9	2025-04-25
5	5	1	3	FIN	8	2025-07-15
6	6	2	1	PAR	7	2025-04-15
7	7	2	2	PAR	8	2025-05-20
8	8	2	2	FIN	9	2025-07-12
9	9	4	1	PAR	10	2025-04-15
10	10	4	6	PAR	7	2025-04-10
11	11	4	6	FIN	8	2025-07-08
12	12	6	1	PAR	3	2025-04-15
13	13	6	4	PAR	6	2025-05-18
14	14	9	1	PAR	8	2025-04-15
15	15	9	1	FIN	7	2025-07-10
16	19	6	1	PAR	2	2025-04-29

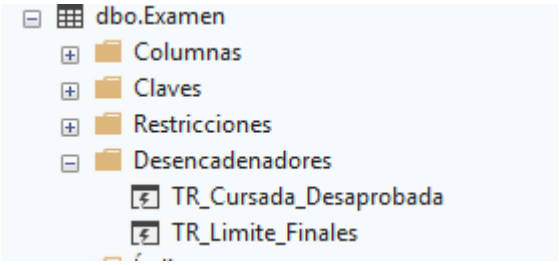
Resultado del trigger sobre la tabla Cursada:

	legajo_alumno	codigo_curso	condicion
1	1	1	APR
2	1	2	REG
3	1	3	APR
4	2	1	REG
5	2	2	APR
6	3	1	CUR
7	3	2	CUR
8	3	4	CUR
9	3	5	CUR
10	3	7	CUR
11	3	8	CUR
12	3	9	CUR
13	4	1	PRO
14	4	6	APR
15	5	5	CUR
16	6	1	DES
17	6	4	CUR
18	7	7	CUR
19	8	5	REG
20	9	1	APR
21	9	2	CUR
22	10	10	CUR
23	11	1	REG
24	12	5	CUR
25	14	10	CUR
26	15	12	CUR

```
CREATE OR ALTER TRIGGER TR_Limite_Finales
ON Examen
INSTEAD OF INSERT
AS
BEGIN
--Valida, en las inserciones de examenes finales, que no se puedan tener
más de 3 para una cursada
    IF EXISTS (
        SELECT 1
        FROM inserted I
        WHERE I.tipo = 'FIN'
        AND (
            SELECT COUNT(*)
            FROM Examen E
            WHERE E.legajo_alumno = I.legajo_alumno
              AND E.codigo_curso = I.codigo_curso
```

```
                AND E.tipo = 'FIN'
        ) >= 3
    )
    BEGIN
        RAISERROR('El alumno ya alcanzó el máximo de 3 finales
permitidos.', 16, 1);
        RETURN;
    END;

    INSERT INTO Examen (legajo_alumno, codigo_curso, tipo, nota, fecha)
    SELECT legajo_alumno, codigo_curso, tipo, nota, fecha
    FROM inserted;

END;
GO
```

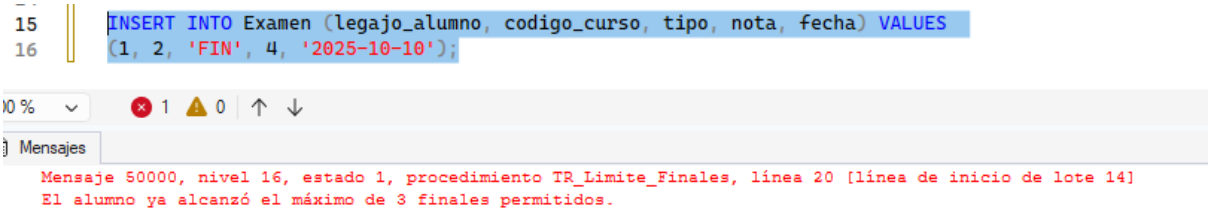
El TRIGGER TR_Limite_Finales es de tipo INSTEAD OF y actúa sobre los INSERT a la tabla EXAMEN. Verifica la cantidad de exámenes finales ingresados, teniendo como máximo 3 instancias de examen final. En caso de estar superando este límite, lanza un error y no realiza la operación.
Esta lógica no podría haber sido definida a través de un CHECK ya que depende de la verificación de múltiples filas y no sólo de los valores ingresados.



Exámenes de tipo FIN para el alumno 1 en el curso 2:

17	20	1	2	FIN	2	2025-06-09
18	21	1	2	FIN	3	2025-07-01
19	22	1	2	FIN	3	2025-08-30

INSERT de un nuevo examen final, arroja error por el control de TR_Limite_Finales:



4. Informe final

La normalización se centró en tres tablas del dominio Universidad: Cursada (principal) y las auxiliares Curso y Prestamos_Libros, llevándolas a 3FN y eliminando redundancias y dependencias parciales o transitivas.
En Cursada se conservaron únicamente los atributos que dependen de toda la clave primaria compuesta {legajo_alumno, codigo_curso} (condicion, nota_final); los datos propios del alumno y del curso se almacenan en Alumno y Curso y se recuperan mediante joins.
En Curso se eliminaron los atributos nombre_materia y nombre_docente, por depender transitivamente de codigo_materia y legajo_docente, y se mantuvo la oferta con la PK codigo y una clave única {codigo_materia, legajo_docente, anio, cuatrimestre, horario, modalidad}.
Prestamos_Libros ya se encontraba en 3FN; solo se reforzó la integridad con claves foráneas y una restricción CHECK de tipo XOR (alumno o docente). De este modo se evitan anomalías de actualización, inserción y borrado, se reduce la inconsistencia y se mantiene la coherencia de los datos. El mayor costo de joins en las consultas de lectura se compensa con menor almacenamiento duplicado y actualizaciones más simples. Este diseño, además, facilita la construcción de objetos SQL (vista, procedimiento almacenado, UDF y triggers) sobre una base consistente.

La vista *VW_Alumnos_Cursando* concentra la información de alumnos activos, lista para utilizar en reportes, seguimiento académico y otras decisiones administrativas. El procedimiento almacenado *SP_Prestar_Libro* asegura que los préstamos de libros se registren de forma consistente, evitando operaciones incompletas. La función UDF *UDF_NotaCursada* contiene lógica de cálculo de la nota de cursada, permitiendo reutilizarla en distintos procesos como reportes estadísticos por materias, historiales académicos. Por último, los triggers implementados agregan mecanismos automáticos que refuerzan reglas institucionales y evitan errores de carga manual: como la desaprobación de la cursada por acumulación de parciales desaprobados o el límite de instancias de examen final.

En conjunto, estos componentes fortalecen la confiabilidad del sistema, facilitan su mantenimiento y minimizan errores humanos, preservando la integridad de la información.