**Sistema de Gestión Académica – Base de Datos Universidad**

  
Proyecto Final  
Universidad Bethesda  
Autor:Jose Miguel Parada Leon  
Docente: *Ing. Tito Flores Vicente*  
UNIBETH 2025  
Fecha: *03/10/2025*

**Índice**

1. Introducción
2. Objetivos
3. Diseño de la Base de Datos  
   3.1. Creación y configuración  
   3.2. Tablas  
   3.3. Relaciones y claves
4. Inserción de Datos Iniciales
5. Procedimientos Almacenados
6. Funciones
7. Triggers
8. Vistas
9. Pruebas y Resultados
10. Conclusiones
11. Anexos (diagramas, capturas, scripts completos)

**1. Introducción**

El presente proyecto tiene como objetivo desarrollar una **base de datos para la gestión académica de la Universidad Bethesda**, que permita registrar y administrar información de estudiantes, docentes, materias, inscripciones, asistencias, calificaciones y becas.

La base de datos fue diseñada utilizando **SQL Server**, implementando buenas prácticas de integridad referencial y seguridad de los datos.

**2. Objetivos**

**Objetivo general:**  
Desarrollar un sistema de base de datos relacional para gestionar la información académica de manera eficiente y confiable.

**Objetivos específicos:**

* Crear las tablas principales del sistema académico.
* Implementar procedimientos, funciones y triggers para automatizar operaciones.
* Generar vistas que simplifiquen consultas frecuentes.
* Insertar datos de prueba y validar el correcto funcionamiento del sistema.

**3. Diseño de la Base de Datos**

**3.1 Creación y configuración**

Se verifica si la base de datos ya existe, eliminándola en caso afirmativo para garantizar una instalación limpia:

IF DB\_ID('Universidad') IS NOT NULL

BEGIN

ALTER DATABASE Universidad SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

DROP DATABASE Universidad;

END

GO

Posteriormente, se crea la base de datos:

CREATE DATABASE Universidad;

GO

USE Universidad;

GO

**3.2 Tablas**

Se diseñaron las siguientes tablas principales:

* **Carreras**: nombre, facultad, duración.
* **Estudiantes**: datos personales, estado académico, carrera.
* **Docentes**: información de formación y especialidad.
* **Materias**: asignaturas y créditos asociados a la carrera.
* **Grupos**: asignación de docentes y estudiantes por materia, semestre y gestión.
* **Inscripciones**, **Asistencias**, **Calificaciones**, **Becas** y **BecasEstudiantes**: control de operaciones académicas y beneficios.

Todas las tablas tienen claves primarias y relaciones mediante claves foráneas.

**3.3 Relaciones y claves**

Se implementó integridad referencial para garantizar consistencia de datos:

* Estudiantes.IdCarrera → Carreras.IdCarrera
* Materias.IdCarrera → Carreras.IdCarrera
* Grupos.IdMateria → Materias.IdMateria
* Grupos.IdDocente → Docentes.IdDocente
* Inscripciones.IdEstudiante → Estudiantes.IdEstudiante
* Inscripciones.IdGrupo → Grupos.IdGrupo
* Asistencias.IdInscripcion → Inscripciones.IdInscripcion
* Calificaciones.IdInscripcion → Inscripciones.IdInscripcion
* BecasEstudiantes.IdBeca → Becas.IdBeca
* BecasEstudiantes.IdEstudiante → Estudiantes.IdEstudiante

**4. Inserción de Datos Iniciales**

Se insertaron datos de ejemplo para permitir pruebas funcionales:

INSERT INTO Docentes (Nombre, CI, FormacionAcademica, Especialidad, Correo, Telefono)

VALUES ('Juan Pérez', '12345678', 'Ing. de Sistemas', 'Bases de Datos', 'juan.perez@email.com', '70000001');

Se completó la inserción de **carreras, materias, grupos, estudiantes, inscripciones, asistencias, calificaciones y becas**.

**5. Procedimientos Almacenados**

Se crearon procedimientos para operaciones frecuentes, como:

* sp\_InsertarEstudiante
* sp\_InsertarDocente
* sp\_ActualizarEstudiante
* sp\_EliminarEstudiante

Estos procedimientos **garantizan consistencia y simplifican la gestión de datos**.

**6. Funciones**

Se implementaron funciones que permiten obtener información agregada:

* fn\_PromedioEstudiante: calcula el promedio de un estudiante.
* fn\_CantidadMateriasPorCarrera: obtiene el número de materias de una carrera.

**7. Triggers**

Se implementaron triggers para mantener la integridad de la base:

* trg\_NoDuplicarInscripcion: evita inscripciones duplicadas.
* trg\_ValidarNota: asegura que las notas estén en el rango 0–100.

**8. Vistas**

Se crearon vistas para simplificar consultas:

* vw\_EstudiantesCarrera
* vw\_AsistenciaEstudiantes
* vw\_CalificacionesEstudiantes
* vw\_BecasEstudiantes

Estas vistas permiten obtener información consolidada y resumida.

**9. Pruebas y Resultados**

Se realizaron pruebas de:

* Consultas básicas.
* Procedimientos y funciones.
* Triggers para validación de duplicados y rango de notas.
* Visualización de datos mediante vistas.

**Resultados:** todas las operaciones fueron exitosas, validando la correcta implementación del sistema.

**10. Conclusiones**

* La base de datos implementa todas las funcionalidades requeridas para la gestión académica.
* Las relaciones y triggers garantizan integridad y seguridad de la información.
* Los procedimientos, funciones y vistas facilitan la administración de datos.

**11. Anexos**

* Diagrama ER de la base de datos.
* Capturas de pantalla de consultas y resultados.
* Script SQL completo.

# Diagrama de Relaciones - Base de Datos Universidad

El siguiente esquema representa el modelo entidad-relación simplificado de la base de datos diseñada para la Universidad Bethesda. Muestra las principales entidades, sus relaciones y cardinalidades.

Carreras ───< Estudiantes  
 │  
 └───< Materias ───< Grupos >─── Docentes  
 │  
 └───< Inscripciones >─── Estudiantes  
 │  
 ┌───────────< Asistencias  
 └───────────< Calificaciones  
  
Becas ───< BecasEstudiantes >─── Estudiantes

## Descripción del Diagrama

- Una Carrera puede tener muchos Estudiantes y Materias.  
- Una Materia se organiza en Grupos que son dictados por Docentes.  
- Un Estudiante se inscribe en un Grupo, lo que genera Inscripciones.  
- De las Inscripciones derivan las Asistencias y Calificaciones.  
- Las Becas pueden asignarse a Estudiantes a través de la tabla intermedia BecasEstudiantes.