“Año del Bicentenario, de la consolidación de nuestra Independencia, y de la conmemoración de las Heroicas Batallas de Junín y Ayacucho”

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Informe de Caso: Sistema de Gestión de Inscripción y Evaluaciones en la Universidad "TechAcademy"

|  |  |  |
| --- | --- | --- |
| Curso: | Base de Datos | 39104 |
| Docente: | Núñez Huerta, José Adrián | C26008 |
| Estudiantes: | Josué Isaac Carlos Condori | U23247985 |
|  | Díaz Culqui, Nehemías | U22327597 |
|  | Jara Herrera, Jhordy Daniel | U18206822 |
|  | Ortiz Angeles, Eduard Frank | U23270119 |

viernes, 6 de Diciembre de 2024

Índice

[1 Introducción 4](#_Toc184318092)

[2 Presentación del caso 4](#_Toc184318093)

[2.1 Contextualización 4](#_Toc184318094)

[2.1.1 Objetivos 4](#_Toc184318095)

[2.2 Problemática 4](#_Toc184318096)

[3 Análisis del Caso 4](#_Toc184318097)

[3.1 Causalidad de la Problemática 4](#_Toc184318098)

[3.2 Presentación de Posibles Solución 4](#_Toc184318099)

[3.3 Planteamiento de la Propuesta 5](#_Toc184318100)

[Problemática 5](#_Toc184318101)

[Solución Propuesta 5](#_Toc184318102)

[4 Diseño del Sistema de la Propuesta 5](#_Toc184318103)

[4.1 Modelo Entidad - Relación 5](#_Toc184318104)

[4.1.1 Entidades y Atributos 5](#_Toc184318105)

[4.1.2 Relaciones 5](#_Toc184318106)

[4.1.3 Diagrama 6](#_Toc184318107)

[4.2 Modelo Lógico Relacional 7](#_Toc184318108)

[5 Normalización 8](#_Toc184318109)

[5.1 Aplicación de Formas Normales 8](#_Toc184318110)

[6 Implementación del Sistema 9](#_Toc184318111)

[6.1 Gestión de Base de Datos 9](#_Toc184318112)

[6.1.1 DDL (Data Definition Language) - Lenguaje de Definición de Datos 9](#_Toc184318113)

[6.1.2 DCL (Data Control Language) - Lenguaje de Control de Datos. TCL (Transaction Control Language) - Lenguaje de Control de Transacciones. Consultas Complementarias: Consultas de Vistas (Views), Consultas de Uniones (JOINs), Consultas de Índices 10](#_Toc184318114)

[6.1.1 Consultas de Procedimientos Almacenados (Stored Procedures) y Consultas de Funciones (Functions) 10](#_Toc184318115)

[7 Administración de usuarios 16](#_Toc184318116)

[8 Conclusión 18](#_Toc184318117)

[ANEXOS 19](#_Toc184318118)

Informe de Caso: Sistema de Gestión de Inscripción y Evaluaciones en la Universidad "TechAcademy"

# Introducción

Este informe presenta el desarrollo de funciones y procedimientos almacenados para la gestión de una base de datos académica en SQL Server en propuesta de solución a los problemas presentados en la Universidad TechAcademy presentado en el caso ficticio. Las funciones y procedimientos permiten automatizar tareas como la inscripción de estudiantes, el cálculo del promedio de notas, y la validación de prerrequisitos. Se proporcionan los scripts SQL correspondientes, junto con ejemplos de ejecución.

# Presentación del caso

## Contextualización

La Universidad "TechAcademy" enfrenta múltiples desafíos en la gestión del proceso de inscripción de estudiantes, la administración de cursos y la evaluación académica. Este informe presenta el diseño y desarrollo de un Sistema de Gestión Académica que centralice y automatice dichos procesos, optimizando la experiencia de estudiantes y personal académico.

### Objetivos

#### Objetivos Generales

Diseñar un sistema de base de datos que optimice los procesos operativos y de información.

#### Objetivos Específicos

1. Facilitar el almacenamiento y recuperación de datos.
2. Mejorar la integridad y seguridad de la información.
3. Proveer consultas rápidas y flexibles para la toma de decisiones.

## Problemática

Se explica detalladamente el problema identificado, incluyendo sus causas y las consecuencias actuales para la organización.

#### Problemas Identificados

#### Relaciones Causales

# Análisis del Caso

## Causalidad de la Problemática

Se describen escenarios específicos que ilustran el impacto de la problemática en la organización.

* Problema – causas

## Presentación de Posibles Solución

Se enumeran y analizan soluciones alternativas, destacando sus ventajas y desventajas.

## Planteamiento de la Propuesta

Se propone una solución basada en el análisis previo, destacando por qué es la opción más adecuada para abordar la problemática.

| Problemática | Solución Propuesta |
| --- | --- |
| Procesos manuales y descentralizados. | - Centralizar la inscripción en cursos. |
| Demoras en inscripciones y asignación de evaluaciones. | - Automatizar la verificación de prerrequisitos. |
| Alta probabilidad de errores en el manejo de información académica. | - Administrar las evaluaciones académicas. |

# Diseño del Sistema de la Propuesta

Para cumplir con los requerimientos de la propuesta y con los estándares de integridad de datos, planteamos el siguiente sistema de base de datos.

## Modelo Entidad - Relación

### Entidades y Atributos

**Estudiante:** ID\_Estudiante (PK), Nombre, Apellido, Email, Teléfono, Carrera.

**Curso:** ID\_Curso (PK), Nombre\_Curso, Créditos, Prerrequisitos, ID\_Profesor (FK).

**Profesor:** ID\_Profesor (PK), Nombre, Departamento, Email.

**Inscripción:** ID\_Inscripción (PK), Fecha\_Inscripción, ID\_Estudiante (FK), ID\_Curso (FK).

**Evaluación:** ID\_Evaluación (PK), Nota, Fecha\_Evaluación, ID\_Estudiante (FK), ID\_Curso (FK).

### Relaciones

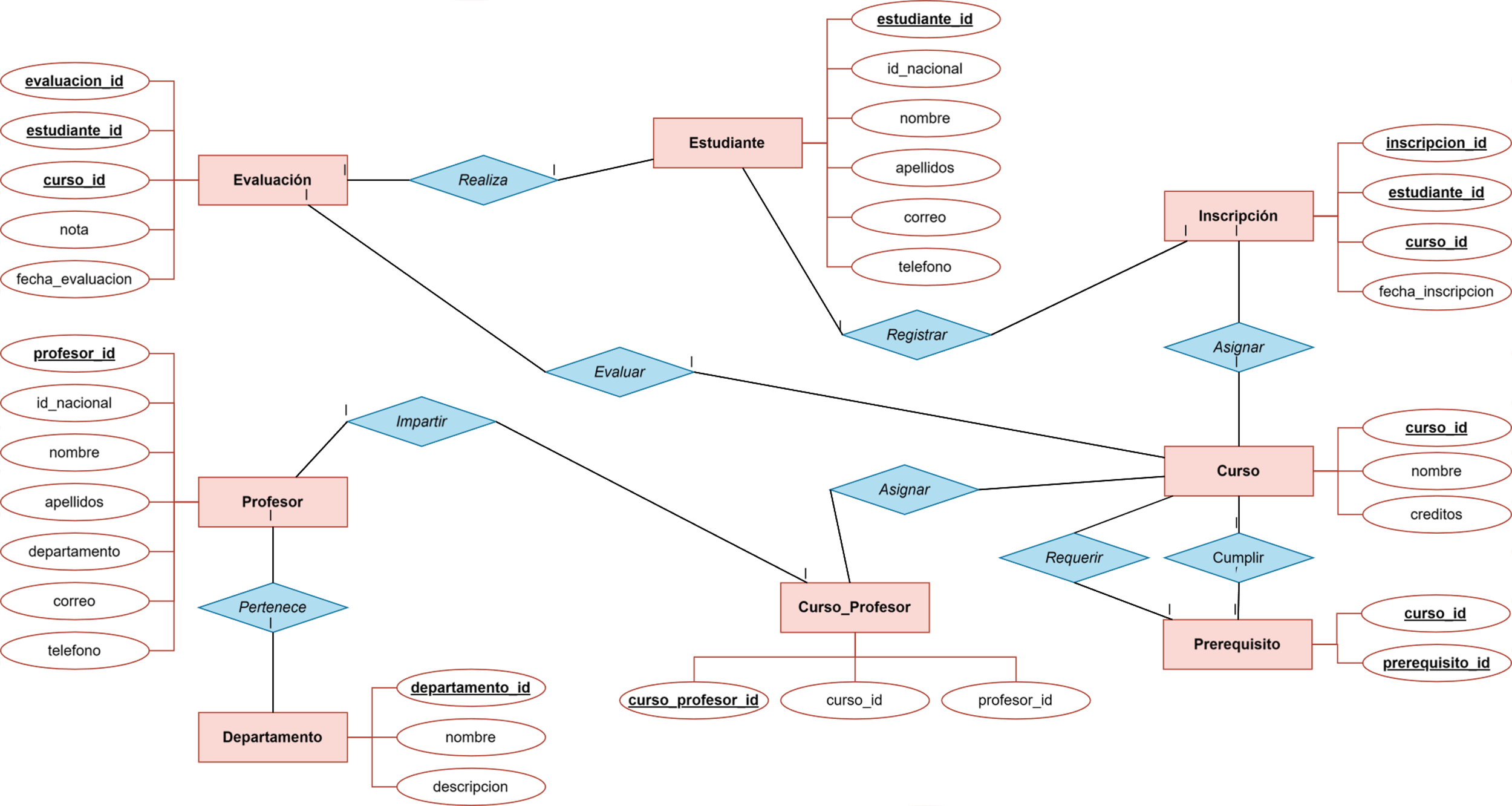
Un estudiante puede inscribirse en varios cursos (1 a N).

Un curso tiene un profesor asignado (N a 1).

Un profesor puede impartir varios cursos (1 a N).

Un estudiante puede tener varias evaluaciones en diferentes cursos (1 a N).

### Diagrama

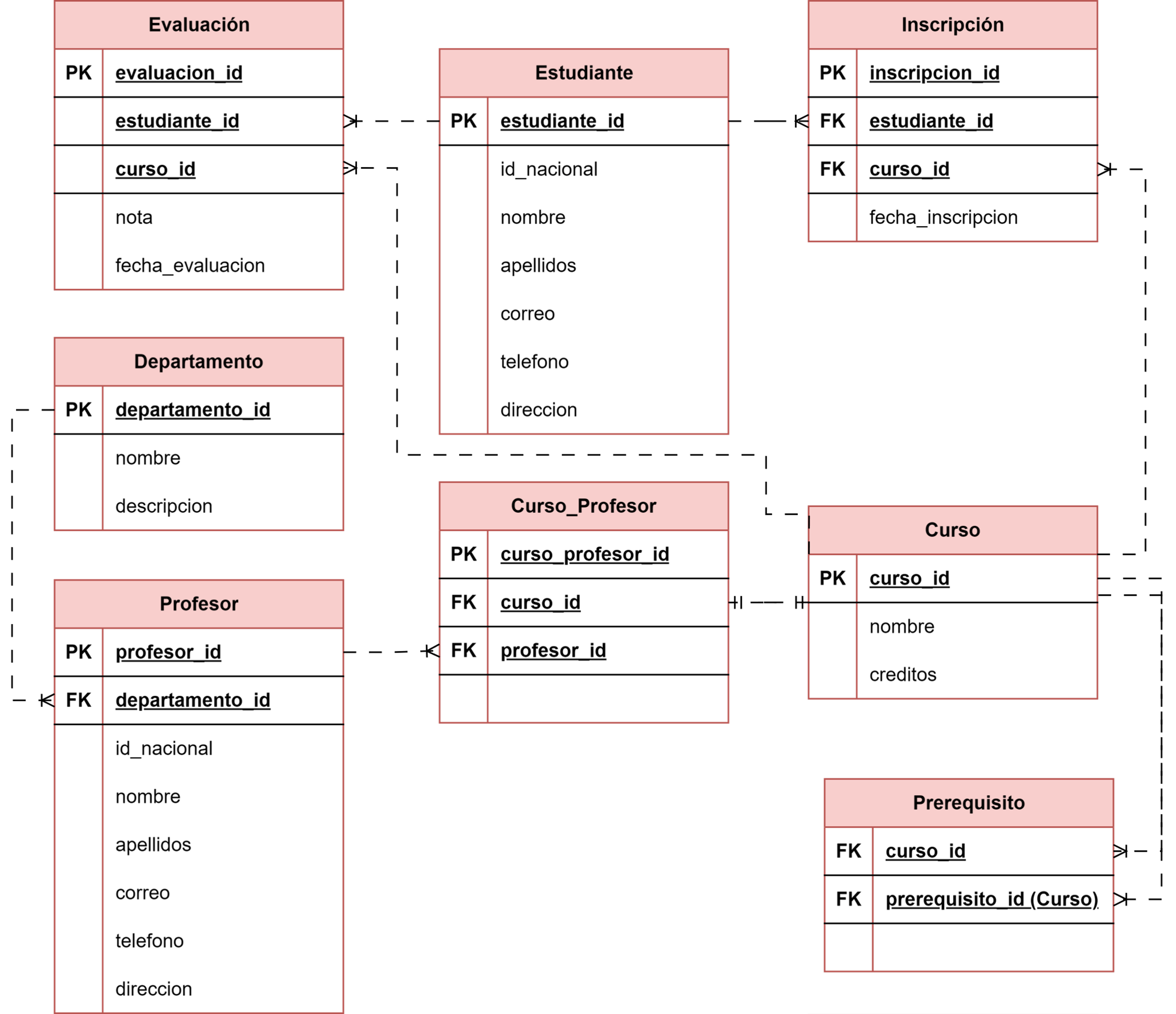


## Modelo Lógico Relacional

Definir claves primarias (PK) y foráneas (FK).

Crear tablas relacionales con dependencias claramente definidas.

Diagrama del modelo



# Normalización

## Aplicación de Formas Normales

**Primera Forma Normal (1NF):** Asegurar atributos atómicos.

**Segunda Forma Normal (2NF):** Eliminar dependencias parciales.

**Tercera Forma Normal (3NF):** Eliminar dependencias transitivas.

**BCNF:** Garantizar que cada determinante sea una clave candidata.

**Cuarta y Quinta Forma Normal:** Reducir dependencias multivaluadas y redundancias.

# Implementación del Sistema

## Gestión de Base de Datos

### DDL (Data Definition Language) - Lenguaje de Definición de Datos

-- Crear la base de datos TECHACADEMY

CREATE DATABASE TECHACADEMY;

GO

-- Cambiar al contexto de la base de datos TECHACADEMY

USE TECHACADEMY;

GO

-- Tabla para Estudiantes

CREATE TABLE Estudiante (

estudiante\_id INT IDENTITY PRIMARY KEY,

id\_nacional CHAR(9) NOT NULL UNIQUE, -- DNI

nombre NVARCHAR(100) NOT NULL,

apellidos NVARCHAR(100) NOT NULL,

correo NVARCHAR(150) UNIQUE NOT NULL,

telefono NVARCHAR(20),

direccion NVARCHAR(MAX)

);

-- Tabla de Departamentos

CREATE TABLE Departamento (

departamento\_id INT IDENTITY PRIMARY KEY,

nombre NVARCHAR(100) NOT NULL UNIQUE,

descripcion NVARCHAR(255)

);

-- Tabla para Profesores

CREATE TABLE Profesor (

profesor\_id INT IDENTITY PRIMARY KEY,

id\_nacional VARCHAR(12) NOT NULL UNIQUE, -- DNI

nombre NVARCHAR(100) NOT NULL,

apellidos NVARCHAR(100) NOT NULL,

departamento\_id INT NULL, -- Permite NULL para cumplir con ON DELETE SET NULL

correo NVARCHAR(150) UNIQUE NOT NULL,

telefono VARCHAR(9) UNIQUE NOT NULL,

direccion NVARCHAR(MAX),

FOREIGN KEY (departamento\_id) REFERENCES Departamento(departamento\_id) ON DELETE SET NULL

);

-- Tabla para Cursos

CREATE TABLE Curso (

curso\_id INT IDENTITY PRIMARY KEY,

nombre NVARCHAR(150) NOT NULL,

creditos INT NOT NULL CHECK (creditos > 0)

);

-- Tabla para Prerequisitos de Cursos

CREATE TABLE Prerequisito (

curso\_id INT NOT NULL,

prerequisito\_id INT NOT NULL,

PRIMARY KEY (curso\_id, prerequisito\_id),

FOREIGN KEY (curso\_id) REFERENCES Curso(curso\_id) ON DELETE NO ACTION,

FOREIGN KEY (prerequisito\_id) REFERENCES Curso(curso\_id) ON DELETE NO ACTION

);

-- Tabla para asignación de Profesores a Cursos (1:1 ajustado con restricción única)

CREATE TABLE Curso\_Profesor (

curso\_profesor\_id INT IDENTITY PRIMARY KEY,

curso\_id INT NOT NULL UNIQUE, -- Garantiza que un curso solo puede estar asociado con un profesor

profesor\_id INT NOT NULL,

FOREIGN KEY (curso\_id) REFERENCES Curso(curso\_id) ON DELETE CASCADE,

FOREIGN KEY (profesor\_id) REFERENCES Profesor(profesor\_id) ON DELETE CASCADE

);

-- Tabla para Inscripciones

CREATE TABLE Inscripcion (

inscripcion\_id INT IDENTITY PRIMARY KEY,

fecha\_inscripcion DATE NOT NULL DEFAULT GETDATE(),

estudiante\_id INT NOT NULL,

curso\_profesor\_id INT NOT NULL,

FOREIGN KEY (estudiante\_id) REFERENCES Estudiante(estudiante\_id) ON DELETE CASCADE,

FOREIGN KEY (curso\_profesor\_id) REFERENCES Curso\_Profesor(curso\_profesor\_id) ON DELETE CASCADE

);

-- Tabla para Evaluaciones

CREATE TABLE Evaluacion (

evaluacion\_id INT IDENTITY PRIMARY KEY,

nota DECIMAL(5, 2) CHECK (nota BETWEEN 0 AND 100),

fecha\_evaluacion DATE NOT NULL DEFAULT GETDATE(),

estudiante\_id INT NOT NULL,

curso\_profesor\_id INT NOT NULL,

FOREIGN KEY (estudiante\_id) REFERENCES Estudiante(estudiante\_id) ON DELETE CASCADE,

FOREIGN KEY (curso\_profesor\_id) REFERENCES Curso\_Profesor(curso\_profesor\_id) ON DELETE CASCADE

);

DML (Data Manipulation Language) - Lenguaje de Manipulación de Datos

-- Crear la base de datos TECHACADEMY

CREATE DATABASE TECHACADEMY;

GO

-- Cambiar al contexto de la base de datos TECHACADEMY

USE TECHACADEMY;

GO

-- Tabla para Estudiantes

CREATE TABLE Estudiante (

estudiante\_id INT IDENTITY PRIMARY KEY,

id\_nacional CHAR(9) NOT NULL UNIQUE, -- DNI

nombre NVARCHAR(100) NOT NULL,

apellidos NVARCHAR(100) NOT NULL,

correo NVARCHAR(150) UNIQUE NOT NULL,

telefono NVARCHAR(20),

direccion NVARCHAR(MAX)

);

-- Tabla de Departamentos

CREATE TABLE Departamento (

departamento\_id INT IDENTITY PRIMARY KEY,

nombre NVARCHAR(100) NOT NULL UNIQUE,

descripcion NVARCHAR(255)

);

-- Tabla para Profesores

CREATE TABLE Profesor (

profesor\_id INT IDENTITY PRIMARY KEY,

id\_nacional VARCHAR(12) NOT NULL UNIQUE, -- DNI

nombre NVARCHAR(100) NOT NULL,

apellidos NVARCHAR(100) NOT NULL,

departamento\_id INT NULL, -- Permite NULL para cumplir con ON DELETE SET NULL

correo NVARCHAR(150) UNIQUE NOT NULL,

telefono VARCHAR(9) UNIQUE NOT NULL,

direccion NVARCHAR(MAX),

FOREIGN KEY (departamento\_id) REFERENCES Departamento(departamento\_id) ON DELETE SET NULL

);

-- Tabla para Cursos

CREATE TABLE Curso (

curso\_id INT IDENTITY PRIMARY KEY,

nombre NVARCHAR(150) NOT NULL,

creditos INT NOT NULL CHECK (creditos > 0)

);

-- Tabla para Prerequisitos de Cursos

CREATE TABLE Prerequisito (

curso\_id INT NOT NULL,

prerequisito\_id INT NOT NULL,

PRIMARY KEY (curso\_id, prerequisito\_id),

FOREIGN KEY (curso\_id) REFERENCES Curso(curso\_id) ON DELETE NO ACTION,

FOREIGN KEY (prerequisito\_id) REFERENCES Curso(curso\_id) ON DELETE NO ACTION

);

-- Tabla para asignación de Profesores a Cursos (1:1 ajustado con restricción única)

CREATE TABLE Curso\_Profesor (

curso\_profesor\_id INT IDENTITY PRIMARY KEY,

curso\_id INT NOT NULL UNIQUE, -- Garantiza que un curso solo puede estar asociado con un profesor

profesor\_id INT NOT NULL,

FOREIGN KEY (curso\_id) REFERENCES Curso(curso\_id) ON DELETE CASCADE,

FOREIGN KEY (profesor\_id) REFERENCES Profesor(profesor\_id) ON DELETE CASCADE

);

-- Tabla para Inscripciones

CREATE TABLE Inscripcion (

inscripcion\_id INT IDENTITY PRIMARY KEY,

fecha\_inscripcion DATE NOT NULL DEFAULT GETDATE(),

estudiante\_id INT NOT NULL,

curso\_profesor\_id INT NOT NULL,

FOREIGN KEY (estudiante\_id) REFERENCES Estudiante(estudiante\_id) ON DELETE CASCADE,

FOREIGN KEY (curso\_profesor\_id) REFERENCES Curso\_Profesor(curso\_profesor\_id) ON DELETE CASCADE

);

-- Tabla para Evaluaciones

CREATE TABLE Evaluacion (

evaluacion\_id INT IDENTITY PRIMARY KEY,

nota DECIMAL(5, 2) CHECK (nota BETWEEN 0 AND 20),

fecha\_evaluacion DATE NOT NULL DEFAULT GETDATE(),

estudiante\_id INT NOT NULL,

curso\_profesor\_id INT NOT NULL,

FOREIGN KEY (estudiante\_id) REFERENCES Estudiante(estudiante\_id) ON DELETE CASCADE,

FOREIGN KEY (curso\_profesor\_id) REFERENCES Curso\_Profesor(curso\_profesor\_id) ON DELETE CASCADE

);

### DCL (Data Control Language) - Lenguaje de Control de Datos. TCL (Transaction Control Language) - Lenguaje de Control de Transacciones. Consultas Complementarias: Consultas de Vistas (Views), Consultas de Uniones (JOINs), Consultas de Índices

Para cumplir con un estándar de calidad de datos es importante realizar una estructuración de consultas restringidas.

### Consultas de Procedimientos Almacenados (Stored Procedures) y Consultas de Funciones (Functions)

USE TECHACADEMY;

GO

-- Crear roles de base de datos

CREATE ROLE Rol\_Estudiante;

CREATE ROLE Rol\_Profesor;

CREATE ROLE Rol\_Administrador;

-- Asignar permisos al Rol\_Estudiante

GRANT SELECT ON Estudiante TO Rol\_Estudiante; -- Permitir ver sus propios datos

GRANT SELECT ON Curso TO Rol\_Estudiante; -- Permitir ver cursos disponibles

GRANT SELECT ON Inscripcion TO Rol\_Estudiante; -- Ver sus inscripciones

GRANT SELECT ON Evaluacion TO Rol\_Estudiante; -- Ver sus evaluaciones

GRANT INSERT ON Inscripcion TO Rol\_Estudiante; -- Inscribirse en cursos

-- Asignar permisos al Rol\_Profesor

GRANT SELECT ON Profesor TO Rol\_Profesor; -- Ver datos personales del profesor

GRANT SELECT ON Curso TO Rol\_Profesor; -- Ver cursos que imparte

GRANT SELECT ON Evaluacion TO Rol\_Profesor; -- Ver evaluaciones asignadas

GRANT INSERT, UPDATE ON Evaluacion TO Rol\_Profesor; -- Calificar estudiantes

GRANT SELECT ON Inscripcion TO Rol\_Profesor; -- Ver estudiantes inscritos en sus cursos

-- Asignar permisos al Rol\_Administrador

GRANT SELECT, INSERT, UPDATE, DELETE ON Estudiante TO Rol\_Administrador; -- Gestión completa de estudiantes

GRANT SELECT, INSERT, UPDATE, DELETE ON Profesor TO Rol\_Administrador; -- Gestión completa de profesores

GRANT SELECT, INSERT, UPDATE, DELETE ON Curso TO Rol\_Administrador; -- Gestión completa de cursos

GRANT SELECT, INSERT, UPDATE, DELETE ON Inscripcion TO Rol\_Administrador; -- Gestión completa de inscripciones

GRANT SELECT, INSERT, UPDATE, DELETE ON Evaluacion TO Rol\_Administrador; -- Gestión completa de evaluaciones

GRANT SELECT, INSERT, UPDATE, DELETE ON Curso\_Profesor TO Rol\_Administrador; -- Gestión de asignación de profesores a cursos

-- Crear usuarios y asignar roles

-- Nota: Es necesario reemplazar los valores <nombre\_usuario> y <contraseña> según los requisitos

CREATE LOGIN UsuarioEstudiante WITH PASSWORD = 'Estudiante123!';

CREATE USER UsuarioEstudiante FOR LOGIN UsuarioEstudiante;

ALTER ROLE Rol\_Estudiante ADD MEMBER UsuarioEstudiante;

CREATE LOGIN UsuarioProfesor WITH PASSWORD = 'Profesor123!';

CREATE USER UsuarioProfesor FOR LOGIN UsuarioProfesor;

ALTER ROLE Rol\_Profesor ADD MEMBER UsuarioProfesor;

CREATE LOGIN UsuarioAdministrador WITH PASSWORD = 'Admin123!';

CREATE USER UsuarioAdministrador FOR LOGIN UsuarioAdministrador;

ALTER ROLE Rol\_Administrador ADD MEMBER UsuarioAdministrador;

-- Revisión de los roles asignados

SELECT r.name AS Rol, m.name AS Miembro

FROM sys.database\_role\_members rm

JOIN sys.database\_principals r ON rm.role\_principal\_id = r.principal\_id

JOIN sys.database\_principals m ON rm.member\_principal\_id = m.principal\_id;

----------------------------------------------------------------------

--1. Procedimiento para Crear un Usuario y Asignarle el Rol de Estudiante

-- Procedimiento para Crear un Usuario y Asignarle el Rol de Estudiante

-- Procedimiento para Crear un Usuario y Asignarle el Rol de Estudiante

CREATE PROCEDURE CrearUsuarioEstudiante

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(100),

@Email NVARCHAR(150),

@Telefono NVARCHAR(20),

@Direccion NVARCHAR(MAX)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Validar que el usuario no exista como login

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = @NombreUsuario)

BEGIN

THROW 50001, 'El nombre de usuario ya existe como LOGIN.', 1;

END;

-- Validar que el correo no esté duplicado

IF EXISTS (SELECT 1 FROM Estudiante WHERE correo = @Email)

BEGIN

THROW 50002, 'El correo electrónico ya está registrado en la tabla Estudiante.', 1;

END;

-- Validar que el teléfono no esté duplicado

IF EXISTS (SELECT 1 FROM Estudiante WHERE telefono = @Telefono)

BEGIN

THROW 50003, 'El número de teléfono ya está registrado en la tabla Estudiante.', 1;

END;

-- Crear el login y el usuario

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'CREATE LOGIN [' + @NombreUsuario + '] WITH PASSWORD = ''' + @Contrasena + ''', CHECK\_POLICY = ON;';

EXEC(@SQL);

SET @SQL = 'CREATE USER [' + @NombreUsuario + '] FOR LOGIN [' + @NombreUsuario + ']; ALTER ROLE Estudiante ADD MEMBER [' + @NombreUsuario + '];';

EXEC(@SQL);

-- Insertar datos en la tabla Estudiante

INSERT INTO Estudiante (nombre, correo, telefono, direccion)

VALUES (@NombreUsuario, @Email, @Telefono, @Direccion);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

/\*

--Ejemplo

EXEC CrearUsuarioEstudiante

@NombreUsuario = 'estudiante01',

@Contrasena = 'Segura123!',

@Email = 'estudiante01@techacademy.com',

@Telefono = '987654321',

@Direccion = 'Av. Siempre Viva 123';

\*/

--2. Procedimiento para Crear un Usuario y Asignarle el Rol de Profesor

CREATE PROCEDURE CrearUsuarioProfesor

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(100),

@Email NVARCHAR(150),

@Telefono NVARCHAR(20),

@Direccion NVARCHAR(MAX),

@DepartamentoId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Validar que el usuario no exista como login

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = @NombreUsuario)

BEGIN

THROW 50001, 'El nombre de usuario ya existe como LOGIN.', 1;

END;

-- Validar que el correo no esté duplicado

IF EXISTS (SELECT 1 FROM Profesor WHERE correo = @Email)

BEGIN

THROW 50002, 'El correo electrónico ya está registrado en la tabla Profesor.', 1;

END;

-- Validar que el teléfono no esté duplicado

IF EXISTS (SELECT 1 FROM Profesor WHERE telefono = @Telefono)

BEGIN

THROW 50003, 'El número de teléfono ya está registrado en la tabla Profesor.', 1;

END;

-- Validar que el Departamento exista

IF NOT EXISTS (SELECT 1 FROM Departamento WHERE departamento\_id = @DepartamentoId)

BEGIN

THROW 50004, 'El Departamento especificado no existe.', 1;

END;

-- Crear el login y el usuario

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'CREATE LOGIN [' + @NombreUsuario + '] WITH PASSWORD = ''' + @Contrasena + ''', CHECK\_POLICY = ON;';

EXEC(@SQL);

SET @SQL = 'CREATE USER [' + @NombreUsuario + '] FOR LOGIN [' + @NombreUsuario + ']; ALTER ROLE Profesor ADD MEMBER [' + @NombreUsuario + '];';

EXEC(@SQL);

-- Insertar datos en la tabla Profesor

INSERT INTO Profesor (nombre, correo, telefono, direccion, departamento\_id)

VALUES (@NombreUsuario, @Email, @Telefono, @Direccion, @DepartamentoId);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

/\*

--Ejecución

EXEC CrearUsuarioProfesor

@NombreUsuario = 'profesor01',

@Contrasena = 'DocenteSeguro!',

@Email = 'profesor01@techacademy.com',

@Telefono = '987654322',

@Direccion = 'Calle Principal 456',

@DepartamentoId = 1;

\*/

--3. Procedimiento para Crear un Usuario y Asignarle el Rol de Administrador

CREATE PROCEDURE CrearUsuarioAdministrador

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(100),

@Email NVARCHAR(150),

@Telefono NVARCHAR(20),

@Direccion NVARCHAR(MAX)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Validar que el usuario no exista como login

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = @NombreUsuario)

BEGIN

THROW 50001, 'El nombre de usuario ya existe como LOGIN.', 1;

END;

-- Validar que el correo no esté duplicado

IF EXISTS (

SELECT 1 FROM Estudiante WHERE correo = @Email

UNION ALL

SELECT 1 FROM Profesor WHERE correo = @Email

)

BEGIN

THROW 50002, 'El correo electrónico ya está registrado.', 1;

END;

-- Validar que el teléfono no esté duplicado

IF EXISTS (

SELECT 1 FROM Estudiante WHERE telefono = @Telefono

UNION ALL

SELECT 1 FROM Profesor WHERE telefono = @Telefono

)

BEGIN

THROW 50003, 'El número de teléfono ya está registrado.', 1;

END;

-- Crear el login y el usuario

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'CREATE LOGIN [' + @NombreUsuario + '] WITH PASSWORD = ''' + @Contrasena + ''', CHECK\_POLICY = ON;';

EXEC(@SQL);

SET @SQL = 'CREATE USER [' + @NombreUsuario + '] FOR LOGIN [' + @NombreUsuario + ']; ALTER ROLE Administrador ADD MEMBER [' + @NombreUsuario + '];';

EXEC(@SQL);

-- Opcional: Insertar datos en tabla de Administradores

-- INSERT INTO Administrador (nombre, correo, telefono, direccion)

-- VALUES (@NombreUsuario, @Email, @Telefono, @Direccion);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

/\*

--Ejecución

EXEC CrearUsuarioAdministrador

@NombreUsuario = 'admin01',

@Contrasena = 'AdminSeguro!',

@Email = 'admin01@techacademy.com',

@Telefono = '987654323',

@Direccion = 'Av. Administración 789';

\*/

/\*

7. Agregar un Nuevo Curso

Procedimiento Almacenado: Permite al administrador agregar un nuevo curso.

\*/

CREATE PROCEDURE AgregarCurso

@Nombre NVARCHAR(150),

@Creditos INT

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

-- Verificar si ya existe un curso con el mismo nombre

IF EXISTS (

SELECT 1

FROM Curso

WHERE nombre = @Nombre

)

BEGIN

PRINT 'Error: Ya existe un curso con el nombre especificado.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Validación de los créditos (ejemplo: debe ser un valor positivo)

IF @Creditos <= 0

BEGIN

PRINT 'Error: Los créditos deben ser un valor positivo.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Insertar el nuevo curso

INSERT INTO Curso (nombre, creditos)

VALUES (@Nombre, @Creditos);

COMMIT TRANSACTION;

PRINT 'Curso agregado correctamente.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al agregar el curso.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

--Ejemplo: EXEC AgregarCurso 'Matemáticas Avanzadas', 4;

/\*

8. Modificar Datos de un Curso

Procedimiento Almacenado: Permite al administrador modificar los datos de un curso.

\*/

CREATE PROCEDURE ModificarCurso

@CursoId INT,

@Nombre NVARCHAR(150),

@Creditos INT

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

-- Verificar si el curso con el ID especificado existe

IF NOT EXISTS (

SELECT 1

FROM Curso

WHERE curso\_id = @CursoId

)

BEGIN

PRINT 'Error: El curso con el ID especificado no existe.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Verificar si ya existe un curso con el mismo nombre (excluyendo el curso actual)

IF EXISTS (

SELECT 1

FROM Curso

WHERE nombre = @Nombre AND curso\_id != @CursoId

)

BEGIN

PRINT 'Error: Ya existe un curso con el mismo nombre.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Validación de los créditos (ejemplo: debe ser un valor positivo)

IF @Creditos <= 0

BEGIN

PRINT 'Error: Los créditos deben ser un valor positivo.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Actualizar el curso

UPDATE Curso

SET

nombre = @Nombre,

creditos = @Creditos

WHERE curso\_id = @CursoId;

COMMIT TRANSACTION;

PRINT 'Curso modificado correctamente.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al modificar el curso.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

--Ejemplo: EXEC ModificarCurso 1, 'Matemáticas Básicas', 3;

/\*

9. Eliminar un Curso

Procedimiento Almacenado: Permite al administrador eliminar un curso.

\*/

CREATE PROCEDURE EliminarCurso

@CursoId INT

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

-- Verificar si el curso con el ID especificado existe

IF NOT EXISTS (

SELECT 1

FROM Curso

WHERE curso\_id = @CursoId

)

BEGIN

PRINT 'Error: El curso con el ID especificado no existe.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Verificar si el curso está asociado con algún estudiante o profesor

IF EXISTS (

SELECT 1

FROM Curso\_Estudiante

WHERE curso\_id = @CursoId

)

BEGIN

PRINT 'Error: No se puede eliminar el curso porque está asociado a uno o más estudiantes.';

ROLLBACK TRANSACTION;

RETURN;

END

IF EXISTS (

SELECT 1

FROM Curso\_Profesor

WHERE curso\_id = @CursoId

)

BEGIN

PRINT 'Error: No se puede eliminar el curso porque está asociado a uno o más profesores.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Proceder con la eliminación del curso

DELETE FROM Curso

WHERE curso\_id = @CursoId;

COMMIT TRANSACTION;

PRINT 'Curso eliminado correctamente.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al eliminar el curso.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

Administración de Backups

/\*

11. Crear un Backup de la Base de Datos

Procedimiento Almacenado: Permite al administrador realizar un respaldo de la base de datos.

\*/

CREATE PROCEDURE CrearBackupBaseDatos

@RutaBackup NVARCHAR(255)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Verificar si la ruta de respaldo es válida y tiene permisos adecuados

IF @RutaBackup IS NULL OR @RutaBackup = ''

BEGIN

PRINT 'Error: La ruta de respaldo no puede estar vacía.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Ejecutar el comando de respaldo

DECLARE @SQL NVARCHAR(1000);

SET @SQL = 'BACKUP DATABASE TECHACADEMY TO DISK = @RutaBackup WITH FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD, STATS = 10';

EXEC sp\_executesql @SQL, N'@RutaBackup NVARCHAR(255)', @RutaBackup;

COMMIT TRANSACTION;

PRINT 'Respaldo de la base de datos realizado con éxito.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al realizar el respaldo de la base de datos.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

--Ejemplo: EXEC CrearBackupBaseDatos 'C:\Backups\TechAcademy.bak';

/\*

12. Restaurar la Base de Datos

Procedimiento Almacenado: Permite al administrador restaurar la base de datos desde un archivo de respaldo.

\*/

CREATE PROCEDURE RestaurarBaseDatos

@RutaBackup NVARCHAR(255)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Verificar si la ruta de respaldo es válida

IF @RutaBackup IS NULL OR @RutaBackup = ''

BEGIN

PRINT 'Error: La ruta de respaldo no puede estar vacía.';

ROLLBACK TRANSACTION;

RETURN;

END

-- Verificar si la base de datos ya existe y manejar la restauración

DECLARE @DBExists INT;

SELECT @DBExists = COUNT(\*)

FROM sys.databases

WHERE name = 'TECHACADEMY';

IF @DBExists = 1

BEGIN

PRINT 'Error: La base de datos TECHACADEMY ya existe. Se requiere un proceso de restauración con reemplazo (REPLACE).';

ROLLBACK TRANSACTION;

RETURN;

END

-- Comando de restauración

DECLARE @SQL NVARCHAR(1000);

SET @SQL = 'RESTORE DATABASE TECHACADEMY FROM DISK = @RutaBackup WITH REPLACE, STATS = 10';

EXEC sp\_executesql @SQL, N'@RutaBackup NVARCHAR(255)', @RutaBackup;

COMMIT TRANSACTION;

PRINT 'Restauración de la base de datos realizada con éxito.';

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al realizar la restauración de la base de datos.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

PROCEDIMIENTOS Y FUNCIONES PARA ESTUDIANTES

\*/

/\*

1. Ver Datos Personales del Estudiante

Procedimiento Almacenado: Permite a un estudiante consultar sus datos personales usando su id\_nacional.

\*/

CREATE PROCEDURE VerDatosPersonalesEstudiante

@IdNacional CHAR(9)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

estudiante\_id,

id\_nacional,

nombre,

apellidos,

correo,

telefono,

direccion

FROM Estudiante

WHERE id\_nacional = @IdNacional;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar los datos personales del estudiante.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerDatosPersonalesEstudiante @IdNacional = '123456789';

\*/

/\*

2. Ver Cursos Disponibles

Procedimiento Almacenado: Permite listar todos los cursos disponibles.

\*/

CREATE PROCEDURE VerCursosDisponibles

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

curso\_id,

nombre AS Curso,

creditos

FROM Curso;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar los cursos disponibles.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerCursosDisponibles;

\*/

/\*

3. Inscribirse en un Curso

Procedimiento Almacenado: Permite registrar una inscripción.

\*/

CREATE PROCEDURE InscribirCurso

@EstudianteId INT,

@CursoId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Verificar si la inscripción ya existe

IF EXISTS (

SELECT 1

FROM Inscripcion

WHERE estudiante\_id = @EstudianteId AND curso\_profesor\_id = @CursoId

)

BEGIN

PRINT 'Error: El estudiante ya está inscrito en el curso especificado.';

RETURN;

END

INSERT INTO Inscripcion (estudiante\_id, curso\_profesor\_id, fecha\_inscripcion)

VALUES (@EstudianteId, @CursoId, GETDATE());

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al inscribir al estudiante en el curso.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC InscribirCurso @EstudianteId = 1, @CursoId = 3;

\*/

/\*

4. Ver Inscripciones Actuales

Procedimiento Almacenado: Devuelve los cursos en los que está inscrito un estudiante.

\*/

CREATE PROCEDURE VerInscripcionesEstudiante

@EstudianteId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

I.inscripcion\_id,

C.nombre AS Curso,

C.creditos,

I.fecha\_inscripcion

FROM Inscripcion I

INNER JOIN Curso C ON I.curso\_profesor\_id = C.curso\_id

WHERE I.estudiante\_id = @EstudianteId;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar las inscripciones del estudiante.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerInscripcionesEstudiante @EstudianteId = 1;

\*/

/\*

5. Consultar Notas por Curso

Procedimiento Almacenado: Permite consultar las notas de un estudiante en un curso.

\*/

CREATE PROCEDURE VerNotasPorCurso

@EstudianteId INT,

@CursoId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

E.evaluacion\_id,

E.nota,

E.fecha\_evaluacion

FROM Evaluacion E

WHERE E.estudiante\_id = @EstudianteId AND E.curso\_profesor\_id = @CursoId;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar las notas del estudiante en el curso.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerNotasPorCurso @EstudianteId = 1, @CursoId = 3;

\*/

/\*

6. Contar el Número de Inscripciones del Estudiante

Función Escalar: Devuelve el número de inscripciones activas del estudiante.

\*/

CREATE FUNCTION ContarInscripciones (@EstudianteId INT)

RETURNS INT

AS

BEGIN

DECLARE @Cantidad INT;

-- Intentar contar las inscripciones

SELECT @Cantidad = COUNT(\*)

FROM Inscripcion

WHERE estudiante\_id = @EstudianteId;

-- Devolver la cantidad

RETURN @Cantidad;

END;

/\*

--Ejemplo

SELECT dbo.ContarInscripciones(1);

\*/

/\*

7. Obtener el Promedio de Notas

Función Escalar: Calcula el promedio de notas de un estudiante.

\*/

CREATE FUNCTION PromedioNotas (@EstudianteId INT)

RETURNS DECIMAL(5, 2)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @Promedio DECIMAL(5, 2);

SELECT @Promedio = AVG(nota)

FROM Evaluacion

WHERE estudiante\_id = @EstudianteId;

COMMIT TRANSACTION;

RETURN @Promedio;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al calcular el promedio de notas del estudiante.';

PRINT ERROR\_MESSAGE();

RETURN NULL; -- Retorno en caso de error

END CATCH

END;

/\*

--Ejemplo

SELECT dbo.PromedioNotas(1);

\*/

/\*

PROCEDIMIENTOS Y FUNCIONES PARA Profesor

\*/

/\*

1. Ver Datos Personales del Profesor

Procedimiento Almacenado: Permite a un profesor consultar sus propios datos personales usando su id\_nacional.

\*/

CREATE PROCEDURE VerDatosPersonalesProfesor

@IdNacional VARCHAR(12)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

profesor\_id,

id\_nacional,

nombre,

apellidos,

departamento\_id,

correo,

telefono,

direccion

FROM Profesor

WHERE id\_nacional = @IdNacional;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar los datos personales del profesor.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerDatosPersonalesProfesor @IdNacional = '123456789';

\*/

/\*

2. Ver Cursos Asignados al Profesor

Procedimiento Almacenado: Permite a un profesor consultar los cursos que imparte.

\*/

CREATE PROCEDURE VerCursosAsignados

@ProfesorId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

C.curso\_id,

C.nombre AS Curso,

C.creditos

FROM Curso C

INNER JOIN Curso\_Profesor CP ON C.curso\_id = CP.curso\_id

WHERE CP.profesor\_id = @ProfesorId;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar los cursos asignados al profesor.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerCursosAsignados @ProfesorId = 1;

\*/

/\*

3. Asignar Calificación a un Estudiante

Procedimiento Almacenado: Permite a un profesor asignar o actualizar la calificación de un estudiante en un curso específico.

\*/

CREATE PROCEDURE AsignarCalificacion

@EstudianteId INT,

@CursoId INT,

@Nota DECIMAL(5, 2)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

IF EXISTS (

SELECT 1

FROM Evaluacion

WHERE estudiante\_id = @EstudianteId AND curso\_profesor\_id = @CursoId

)

BEGIN

UPDATE Evaluacion

SET nota = @Nota, fecha\_evaluacion = GETDATE()

WHERE estudiante\_id = @EstudianteId AND curso\_profesor\_id = @CursoId;

END

ELSE

BEGIN

INSERT INTO Evaluacion (nota, fecha\_evaluacion, estudiante\_id, curso\_profesor\_id)

VALUES (@Nota, GETDATE(), @EstudianteId, @CursoId);

END

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al asignar la calificación al estudiante.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC AsignarCalificacion @EstudianteId = 2, @CursoId = 3, @Nota = 90.50;

\*/

/\*

4. Ver Evaluaciones de un Curso

Procedimiento Almacenado: Permite a un profesor consultar todas las evaluaciones de un curso en particular.

\*/

CREATE PROCEDURE VerEvaluacionesCurso

@CursoId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

E.evaluacion\_id,

E.estudiante\_id,

E.nota,

E.fecha\_evaluacion

FROM Evaluacion E

WHERE E.curso\_profesor\_id = @CursoId;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar las evaluaciones del curso.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerEvaluacionesCurso @CursoId = 3;

\*/

/\*

5. Ver Informe de Rendimiento de Estudiantes

Procedimiento Almacenado: Permite al profesor obtener un informe de rendimiento de los estudiantes en un curso específico.

\*/

CREATE PROCEDURE VerInformeRendimientoEstudiantes

@CursoId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

SELECT

E.estudiante\_id,

S.nombre + ' ' + S.apellidos AS Estudiante,

AVG(E.nota) AS PromedioNotas

FROM Evaluacion E

INNER JOIN Estudiante S ON E.estudiante\_id = S.estudiante\_id

WHERE E.curso\_profesor\_id = @CursoId

GROUP BY E.estudiante\_id, S.nombre, S.apellidos

ORDER BY PromedioNotas DESC;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

PRINT 'Error al consultar el informe de rendimiento de los estudiantes.';

PRINT ERROR\_MESSAGE();

END CATCH

END;

/\*

--Ejemplo

EXEC VerInformeRendimientoEstudiantes @CursoId = 3;

\*/

/\*

6. Contar Estudiantes Inscritos en un Curso

Función Escalar: Devuelve el número de estudiantes inscritos en un curso específico.

\*/

CREATE FUNCTION ContarEstudiantesCurso (@CursoId INT)

RETURNS INT

AS

BEGIN

DECLARE @Cantidad INT;

-- Contar el número de estudiantes inscritos en el curso

SELECT @Cantidad = COUNT(\*)

FROM Inscripcion

WHERE curso\_profesor\_id = @CursoId;

-- Devolver la cantidad

RETURN @Cantidad;

END;

/\*

--Ejemplo

SELECT dbo.ContarEstudiantesCurso(3);

\*/

/\*

7. Calcular Promedio de Notas por Curso

Función Escalar: Calcula el promedio de notas de todos los estudiantes en un curso específico.

\*/

CREATE FUNCTION PromedioNotasCurso (@CursoId INT)

RETURNS DECIMAL(5, 2)

AS

BEGIN

DECLARE @Promedio DECIMAL(5, 2);

-- Calcular el promedio de notas del curso

SELECT @Promedio = AVG(nota)

FROM Evaluacion

WHERE curso\_profesor\_id = @CursoId;

-- Devolver el promedio

RETURN @Promedio;

END;

/\*

--Ejemplo

SELECT dbo.PromedioNotasCurso(3);

\*/

# Administración de usuarios

USE TECHACADEMY;

GO

-- Crear roles de base de datos

CREATE ROLE Rol\_Estudiante;

CREATE ROLE Rol\_Profesor;

CREATE ROLE Rol\_Administrador;

-- Asignar permisos al Rol\_Estudiante

GRANT SELECT ON Estudiante TO Rol\_Estudiante; -- Permitir ver sus propios datos

GRANT SELECT ON Curso TO Rol\_Estudiante; -- Permitir ver cursos disponibles

GRANT SELECT ON Inscripcion TO Rol\_Estudiante; -- Ver sus inscripciones

GRANT SELECT ON Evaluacion TO Rol\_Estudiante; -- Ver sus evaluaciones

GRANT INSERT ON Inscripcion TO Rol\_Estudiante; -- Inscribirse en cursos

-- Asignar permisos al Rol\_Profesor

GRANT SELECT ON Profesor TO Rol\_Profesor; -- Ver datos personales del profesor

GRANT SELECT ON Curso TO Rol\_Profesor; -- Ver cursos que imparte

GRANT SELECT ON Evaluacion TO Rol\_Profesor; -- Ver evaluaciones asignadas

GRANT INSERT, UPDATE ON Evaluacion TO Rol\_Profesor; -- Calificar estudiantes

GRANT SELECT ON Inscripcion TO Rol\_Profesor; -- Ver estudiantes inscritos en sus cursos

-- Asignar permisos al Rol\_Administrador

GRANT SELECT, INSERT, UPDATE, DELETE ON Estudiante TO Rol\_Administrador; -- Gestión completa de estudiantes

GRANT SELECT, INSERT, UPDATE, DELETE ON Profesor TO Rol\_Administrador; -- Gestión completa de profesores

GRANT SELECT, INSERT, UPDATE, DELETE ON Curso TO Rol\_Administrador; -- Gestión completa de cursos

GRANT SELECT, INSERT, UPDATE, DELETE ON Inscripcion TO Rol\_Administrador; -- Gestión completa de inscripciones

GRANT SELECT, INSERT, UPDATE, DELETE ON Evaluacion TO Rol\_Administrador; -- Gestión completa de evaluaciones

GRANT SELECT, INSERT, UPDATE, DELETE ON Curso\_Profesor TO Rol\_Administrador; -- Gestión de asignación de profesores a cursos

-- Crear usuarios y asignar roles

-- Nota: Es necesario reemplazar los valores <nombre\_usuario> y <contraseña> según los requisitos

CREATE LOGIN UsuarioEstudiante WITH PASSWORD = 'Estudiante123!';

CREATE USER UsuarioEstudiante FOR LOGIN UsuarioEstudiante;

ALTER ROLE Rol\_Estudiante ADD MEMBER UsuarioEstudiante;

CREATE LOGIN UsuarioProfesor WITH PASSWORD = 'Profesor123!';

CREATE USER UsuarioProfesor FOR LOGIN UsuarioProfesor;

ALTER ROLE Rol\_Profesor ADD MEMBER UsuarioProfesor;

CREATE LOGIN UsuarioAdministrador WITH PASSWORD = 'Admin123!';

CREATE USER UsuarioAdministrador FOR LOGIN UsuarioAdministrador;

ALTER ROLE Rol\_Administrador ADD MEMBER UsuarioAdministrador;

-- Revisión de los roles asignados

SELECT r.name AS Rol, m.name AS Miembro

FROM sys.database\_role\_members rm

JOIN sys.database\_principals r ON rm.role\_principal\_id = r.principal\_id

JOIN sys.database\_principals m ON rm.member\_principal\_id = m.principal\_id;

----------------------------------------------------------------------

--1. Procedimiento para Crear un Usuario y Asignarle el Rol de Estudiante

-- Procedimiento para Crear un Usuario y Asignarle el Rol de Estudiante

-- Procedimiento para Crear un Usuario y Asignarle el Rol de Estudiante

CREATE PROCEDURE CrearUsuarioEstudiante

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(100),

@Email NVARCHAR(150),

@Telefono NVARCHAR(20),

@Direccion NVARCHAR(MAX)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Validar que el usuario no exista como login

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = @NombreUsuario)

BEGIN

THROW 50001, 'El nombre de usuario ya existe como LOGIN.', 1;

END;

-- Validar que el correo no esté duplicado

IF EXISTS (SELECT 1 FROM Estudiante WHERE correo = @Email)

BEGIN

THROW 50002, 'El correo electrónico ya está registrado en la tabla Estudiante.', 1;

END;

-- Validar que el teléfono no esté duplicado

IF EXISTS (SELECT 1 FROM Estudiante WHERE telefono = @Telefono)

BEGIN

THROW 50003, 'El número de teléfono ya está registrado en la tabla Estudiante.', 1;

END;

-- Crear el login y el usuario

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'CREATE LOGIN [' + @NombreUsuario + '] WITH PASSWORD = ''' + @Contrasena + ''', CHECK\_POLICY = ON;';

EXEC(@SQL);

SET @SQL = 'CREATE USER [' + @NombreUsuario + '] FOR LOGIN [' + @NombreUsuario + ']; ALTER ROLE Estudiante ADD MEMBER [' + @NombreUsuario + '];';

EXEC(@SQL);

-- Insertar datos en la tabla Estudiante

INSERT INTO Estudiante (nombre, correo, telefono, direccion)

VALUES (@NombreUsuario, @Email, @Telefono, @Direccion);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

/\*

--Ejemplo

EXEC CrearUsuarioEstudiante

@NombreUsuario = 'estudiante01',

@Contrasena = 'Segura123!',

@Email = 'estudiante01@techacademy.com',

@Telefono = '987654321',

@Direccion = 'Av. Siempre Viva 123';

\*/

--2. Procedimiento para Crear un Usuario y Asignarle el Rol de Profesor

CREATE PROCEDURE CrearUsuarioProfesor

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(100),

@Email NVARCHAR(150),

@Telefono NVARCHAR(20),

@Direccion NVARCHAR(MAX),

@DepartamentoId INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Validar que el usuario no exista como login

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = @NombreUsuario)

BEGIN

THROW 50001, 'El nombre de usuario ya existe como LOGIN.', 1;

END;

-- Validar que el correo no esté duplicado

IF EXISTS (SELECT 1 FROM Profesor WHERE correo = @Email)

BEGIN

THROW 50002, 'El correo electrónico ya está registrado en la tabla Profesor.', 1;

END;

-- Validar que el teléfono no esté duplicado

IF EXISTS (SELECT 1 FROM Profesor WHERE telefono = @Telefono)

BEGIN

THROW 50003, 'El número de teléfono ya está registrado en la tabla Profesor.', 1;

END;

-- Validar que el Departamento exista

IF NOT EXISTS (SELECT 1 FROM Departamento WHERE departamento\_id = @DepartamentoId)

BEGIN

THROW 50004, 'El Departamento especificado no existe.', 1;

END;

-- Crear el login y el usuario

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'CREATE LOGIN [' + @NombreUsuario + '] WITH PASSWORD = ''' + @Contrasena + ''', CHECK\_POLICY = ON;';

EXEC(@SQL);

SET @SQL = 'CREATE USER [' + @NombreUsuario + '] FOR LOGIN [' + @NombreUsuario + ']; ALTER ROLE Profesor ADD MEMBER [' + @NombreUsuario + '];';

EXEC(@SQL);

-- Insertar datos en la tabla Profesor

INSERT INTO Profesor (nombre, correo, telefono, direccion, departamento\_id)

VALUES (@NombreUsuario, @Email, @Telefono, @Direccion, @DepartamentoId);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

/\*

--Ejecución

EXEC CrearUsuarioProfesor

@NombreUsuario = 'profesor01',

@Contrasena = 'DocenteSeguro!',

@Email = 'profesor01@techacademy.com',

@Telefono = '987654322',

@Direccion = 'Calle Principal 456',

@DepartamentoId = 1;

\*/

--3. Procedimiento para Crear un Usuario y Asignarle el Rol de Administrador

CREATE PROCEDURE CrearUsuarioAdministrador

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(100),

@Email NVARCHAR(150),

@Telefono NVARCHAR(20),

@Direccion NVARCHAR(MAX)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Validar que el usuario no exista como login

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = @NombreUsuario)

BEGIN

THROW 50001, 'El nombre de usuario ya existe como LOGIN.', 1;

END;

-- Validar que el correo no esté duplicado

IF EXISTS (

SELECT 1 FROM Estudiante WHERE correo = @Email

UNION ALL

SELECT 1 FROM Profesor WHERE correo = @Email

)

BEGIN

THROW 50002, 'El correo electrónico ya está registrado.', 1;

END;

-- Validar que el teléfono no esté duplicado

IF EXISTS (

SELECT 1 FROM Estudiante WHERE telefono = @Telefono

UNION ALL

SELECT 1 FROM Profesor WHERE telefono = @Telefono

)

BEGIN

THROW 50003, 'El número de teléfono ya está registrado.', 1;

END;

-- Crear el login y el usuario

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'CREATE LOGIN [' + @NombreUsuario + '] WITH PASSWORD = ''' + @Contrasena + ''', CHECK\_POLICY = ON;';

EXEC(@SQL);

SET @SQL = 'CREATE USER [' + @NombreUsuario + '] FOR LOGIN [' + @NombreUsuario + ']; ALTER ROLE Administrador ADD MEMBER [' + @NombreUsuario + '];';

EXEC(@SQL);

-- Opcional: Insertar datos en tabla de Administradores

-- INSERT INTO Administrador (nombre, correo, telefono, direccion)

-- VALUES (@NombreUsuario, @Email, @Telefono, @Direccion);

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

/\*

--Ejecución

EXEC CrearUsuarioAdministrador

@NombreUsuario = 'admin01',

@Contrasena = 'AdminSeguro!',

@Email = 'admin01@techacademy.com',

@Telefono = '987654323',

@Direccion = 'Av. Administración 789';

\*/

# 

# Conclusión

El Sistema de Gestión Académica para "TechAcademy" mejora la eficiencia y precisión en la administración de inscripciones y evaluaciones. Con una base de datos bien normalizada, consultas avanzadas y robusta gestión de transacciones, el sistema es confiable, seguro y adaptable a futuras necesidades.

ANEXOS

Diccionario de Datos

**1. Tabla: Estudiante**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| estudiante\_id | INT | Identificador único del estudiante. | PRIMARY KEY, AUTO\_INCREMENT |
| id\_nacional | CHAR(9) | Documento Nacional de Identidad del estudiante. | UNIQUE, NOT NULL |
| nombre | NVARCHAR(100) | Nombre del estudiante. | NOT NULL |
| apellidos | NVARCHAR(100) | Apellidos del estudiante. | NOT NULL |
| correo | NVARCHAR(150) | Correo electrónico del estudiante. | UNIQUE, NOT NULL |
| telefono | NVARCHAR(20) | Número de teléfono del estudiante. | NULL |
| direccion | NVARCHAR(MAX) | Dirección del estudiante. | NULL |

**2. Tabla: Departamento**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| departamento\_id | INT | Identificador único del departamento. | PRIMARY KEY, AUTO\_INCREMENT |
| nombre | NVARCHAR(100) | Nombre del departamento. | UNIQUE, NOT NULL |
| descripcion | NVARCHAR(255) | Descripción del departamento. | NULL |

**3. Tabla: Profesor**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| profesor\_id | INT | Identificador único del profesor. | PRIMARY KEY, AUTO\_INCREMENT |
| id\_nacional | VARCHAR(12) | Documento Nacional de Identidad del profesor. | UNIQUE, NOT NULL |
| nombre | NVARCHAR(100) | Nombre del profesor. | NOT NULL |
| apellidos | NVARCHAR(100) | Apellidos del profesor. | NOT NULL |
| departamento\_id | INT | Identificador del departamento al que pertenece el profesor. | NULL, FOREIGN KEY (REFERENCIA a Departamento) ON DELETE SET NULL |
| correo | NVARCHAR(150) | Correo electrónico del profesor. | UNIQUE, NOT NULL |
| telefono | VARCHAR(9) | Número de teléfono del profesor. | UNIQUE, NOT NULL |
| direccion | NVARCHAR(MAX) | Dirección del profesor. | NULL |

**4. Tabla: Curso**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| curso\_id | INT | Identificador único del curso. | PRIMARY KEY, AUTO\_INCREMENT |
| nombre | NVARCHAR(150) | Nombre del curso. | NOT NULL |
| creditos | INT | Número de créditos del curso. | CHECK (creditos > 0), NOT NULL |

**5. Tabla: Prerequisito**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| curso\_id | INT | Identificador del curso. | PRIMARY KEY, FOREIGN KEY (REFERENCIA a Curso) ON DELETE NO ACTION |
| prerequisito\_id | INT | Identificador del curso prerequisito. | PRIMARY KEY, FOREIGN KEY (REFERENCIA a Curso) ON DELETE NO ACTION |

**6. Tabla: Curso\_Profesor**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Columna | | | | Tipo de Dato | Descripción | Restricciones |
| curso\_profesor\_id | | | | INT | Identificador único de la asignación curso-profesor. | PRIMARY KEY, AUTO\_INCREMENT |
| curso\_id | | | | INT | Identificador del curso asignado. | UNIQUE, FOREIGN KEY (REFERENCIA a Curso) ON DELETE CASCADE |
| profesor\_id | | | | INT | Identificador del profesor asignado. | FOREIGN KEY (REFERENCIA a Profesor) ON DELETE CASCADE |
|  |  |  |  |

**7. Tabla: Inscripcion**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| inscripcion\_id | INT | Identificador único de la inscripción. | PRIMARY KEY, AUTO\_INCREMENT |
| fecha\_inscripcion | DATE | Fecha de inscripción del estudiante en el curso. | NOT NULL, DEFAULT GETDATE() |
| estudiante\_id | INT | Identificador del estudiante inscrito. | FOREIGN KEY (REFERENCIA a Estudiante) ON DELETE CASCADE |
| curso\_profesor\_id | INT | Identificador de la asignación curso-profesor. | FOREIGN KEY (REFERENCIA a Curso\_Profesor) ON DELETE CASCADE |

**8. Tabla: Evaluacion**

|  |  |  |  |
| --- | --- | --- | --- |
| Columna | Tipo de Dato | Descripción | Restricciones |
| evaluacion\_id | INT | Identificador único de la evaluación. | PRIMARY KEY, AUTO\_INCREMENT |
| nota | DECIMAL(5, 2) | Nota de la evaluación (rango de 0 a 20). | CHECK (nota BETWEEN 0 AND 20), NOT NULL |
| fecha\_evaluacion | DATE | Fecha en la que se realizó la evaluación. | NOT NULL, DEFAULT GETDATE() |
| estudiante\_id | INT | Identificador del estudiante que recibió la evaluación. | FOREIGN KEY (REFERENCIA a Estudiante) ON DELETE CASCADE |
| curso\_profesor\_id | INT | Identificador de la asignación curso-profesor. | FOREIGN KEY (REFERENCIA a Curso\_Profesor) ON DELETE CASCADE |

Manual del Sistema

Instrucciones para el uso y mantenimiento del sistema.