



INSTITUTO TECNOLOGICO DE MEXICALI

Fundamentos De Base De Datos

Examen Final de Unidad 4

ALUMNO: FRANCISCO RAMOS VAZQUEZ

Numero De Control: 23490367

MAESTRO: JOSE RAMON BOGARIN VALENZUELA

CARRERA: INGENIERIA EN SISTEMAS COMPUTACIONALES

Fecha de Entrega: 23/05/2025

SEMESTRE: 4

Indice

Planteamiento del problema.....	3
La Solución a la Propuesta.....	4
Observaciones y Justificaciones.....	5
Planteamiento del problema.....	6

PLANTAMIENTO DEL PROBLEMA

Problema Técnico Gestiónando la Información de una Universidad: La universidad requiere un sistema de gestión que permita controlar información clave sobre su comunidad académica: estudiantes, profesores, cursos, departamentos, inscripciones, programas de estudio, aulas, horarios, campus y carreras. Además, debe permitir consultar datos específicos mediante queries SQL complejos. Este sistema debe estar respaldado por una base de datos relacional en PostgreSQL.

Objetivo General: Diseñar una base de datos relacional y realizar diversas operaciones para gestionar la información de la universidad. Esto incluye la creación y modificación de la estructura de las tablas, la manipulación de los datos y la realización de consultas complejas para obtener información específica.

Diseño de la Base de Datos

Se diseñó un modelo relacional compuesto por las siguientes entidades principales:

- Estudiantes**
- Cursos**
- Inscripciones**
- Profesores**
- Departamentos**
- Horarios**
- CursosProfesores** (relación muchos a muchos)
- ProgramasEstudio**
- ProgramasCursos** (relación muchos a muchos)
- Campus**
- Carreras**
- EstudiantesCarreras** (relación muchos a muchos)

Observaciones y Justificaciones

- **Relaciones Muchos a Muchos:** Se utilizaron tablas intermedias (CursosProfesores, ProgramasCursos, EstudiantesCarreras) para representar correctamente las relaciones complejas entre entidades.
- **Eliminación de Campos:** Se eliminó el campo Ciudad de la tabla Estudiantes para simplificar la información de localización, trasladando esa responsabilidad al campo Direccion.
- **Eliminación de la tabla Aulas:** Se decidió eliminarla ya que no es necesaria para los requerimientos de consultas especificadas. La tabla Horarios se adaptó para funcionar sin dicha referencia.
- **Nombres de claves foráneas y primarias:** Se siguió una convención clara para facilitar la lectura y mantenimiento del código SQL.

Capturas de Resultados de Consultas

1. **Selección Básica:** Muestra todos los nombres y apellidos de los estudiantes.

```
SELECT Nombre, Apellido FROM Estudiantes;
```

	Nombre	Apellido
1	Ana	García
2	Marta	López
3	Jorge	Martínez
4	Luis	Pérez

2. **Cláusula WHERE:** Encuentra todos los cursos que tienen 3 créditos.

```
SELECT * FROM Cursos WHERE Creditos = 3;
```

	idcurso	nombrecurso	descripcion	creditos
1	2	Psicología General	Fundamentos de psicología	3
2	3	Diseño Digital	Herramientas de diseño gráfico	3

3. **INNER JOIN:** Obtén una lista que muestre el nombre del estudiante y el nombre del curso en el que está inscrito.

```
SELECT e.Nombre AS NombreEstudiante, e.Apellido, c.NombreCurso
FROM Inscripciones i
INNER JOIN Estudiantes e ON i.IDEstudiante = e.IDEstudiante
INNER JOIN Cursos c ON i.IDCurso = c.IDCurso;
```

	nombreestudiante	apellido	nombrecurso
1	Ana	García	Programación Avanzada
2	Luis	Pérez	Psicología General
3	Jorge	Martínez	Diseño Digital
4	Marta	López	Programación Avanzada

4. LEFT JOIN: Muestra todos los estudiantes y, si están inscritos en algún curso, el nombre del curso. Si un estudiante no está inscrito en ningún curso, el campo del nombre del curso debe mostrar un valor que lo indique (ej: NULL).

```
SELECT e.Nombre AS NombreEstudiante, e.Apellido, c.NombreCurso
FROM Estudiantes e
LEFT JOIN Inscripciones i ON e.IDEstudiante = i.IDEstudiante
LEFT JOIN Cursos c ON i.IDCurso = c.IDCurso;
```

	NombreEstudiante	Apellido	NombreCurso
1	Ana	García	Programación Avanzada
2	Luis	Pérez	Psicología General
3	Jorge	Martínez	Diseño Digital
4	Marta	López	Programación Avanzada

5. RIGHT JOIN: Lista todos los cursos y, si tienen estudiantes inscritos, el nombre de los estudiantes. Muestra todos los cursos, incluso si no tienen estudiantes inscritos actualmente.

```
SELECT c.NombreCurso, e.Nombre AS NombreEstudiante, e.Apellido
FROM Cursos c
RIGHT JOIN Inscripciones i ON c.IDCurso = i.IDCurso
RIGHT JOIN Estudiantes e ON i.IDEstudiante = e.IDEstudiante
ORDER BY c.NombreCurso;
```

	NombreCurso	NombreEstudiante	Apellido
1	Diseño Digital	Jorge	Martínez
2	Programación Avanzada	Ana	García
3	Programación Avanzada	Marta	López
4	Psicología General	Luis	Pérez

6. GROUP BY y COUNT: Calcula cuántos estudiantes están inscritos en cada curso. Muestra el nombre del curso y la cantidad de estudiantes.

```
SELECT c.NombreCurso, COUNT(i.IDEstudiante) AS NumeroEstudiantes
FROM Cursos c
LEFT JOIN Inscripciones i ON c.IDCurso = i.IDCurso
GROUP BY c.NombreCurso
ORDER BY NumeroEstudiantes DESC;
```

	NombreCurso	NumeroEstudiantes
1	Programación Avanzada	2
2	Psicología General	1
3	Diseño Digital	1

7. BETWEEN: Encuentra todos los estudiantes que nacieron entre el 1 de enero de 1995 y el 31 de diciembre de 1998.

	Nombre	Apellido	FechaNacimiento
1	Ana	García	1996-04-12
2	Marta	López	1995-06-15
3	Luis	Pérez	1997-09-30

8. ORDER BY: Muestra todos los cursos ordenados alfabéticamente por su nombre.

	NombreCurso
1	Diseño Digital
2	Programación Avanzada
3	Psicología General

9. CTE: Crea una tabla de expresión común que liste el número de inscripciones por estudiante. Luego, consulta esta CTE para encontrar los 3 estudiantes con más inscripciones, mostrando el nombre del estudiante y el número de inscripciones.

```
WITH InscripcionesPorEstudiante AS (
    SELECT e.IDEstudiante, e.Nombre, e.Apellido,
    COUNT(i.IDInscripcion) AS NumInscripciones
    FROM Estudiantes e
    LEFT JOIN Inscripciones i ON e.IDEstudiante = i.IDEstudiante
    GROUP BY e.IDEstudiante, e.Nombre, e.Apellido
)
SELECT Nombre, Apellido, NumInscripciones
FROM InscripcionesPorEstudiante
ORDER BY NumInscripciones DESC
LIMIT 3;
```

	Nombre	Apellido	NumInscripciones
1	Marta	López	1
2	Jorge	Martínez	1
3	Luis	Pérez	1

- 10. Consulta Compleja 1:** Para cada departamento, muestra el nombre del departamento y el nombre del curso con la mayor cantidad de estudiantes inscritos.

```
WITH ConteoEstudiantes AS (
    SELECT c.IDCurso, c.NombreCurso, c.IDDepartamento,
    COUNT(i.IDEstudiante) AS NumEstudiantes
    FROM Cursos c
    LEFT JOIN Inscripciones i ON c.IDCurso = i.IDCurso
    GROUP BY c.IDCurso, c.NombreCurso, c.IDDepartamento
),
MaxEstudiantesPorDept AS (
    SELECT IDDepartamento, MAX(NumEstudiantes) AS MaxEstudiantes
    FROM ConteoEstudiantes
    GROUP BY IDDepartamento
)
SELECT d.NombreDepartamento, ce.NombreCurso, ce.NumEstudiantes
FROM ConteoEstudiantes ce
JOIN MaxEstudiantesPorDept med ON ce.IDDepartamento =
med.IDDepartamento AND ce.NumEstudiantes = med.MaxEstudiantes
JOIN Departamentos d ON ce.IDDepartamento = d.IDDepartamento
ORDER BY d.NombreDepartamento;
```

	idcurso	nombrecurso	iddepartamento	numestudiantes
1	2	Psicología General	2	1
2	1	Programación Avanzada	1	2
3	3	Diseño Digital	3	1

- 11. Consulta Compleja 2:** Encuentra a los profesores que imparten más de dos cursos, mostrando su nombre, apellido y la cantidad de cursos que imparten.

```
SELECT p.Nombre, p.Apellido, COUNT(cp.IDCurso) AS NumCursos
FROM Profesores p
JOIN CursosProfesores cp ON p.IDProfesor = cp.IDProfesor
GROUP BY p.IDProfesor, p.Nombre, p.Apellido
HAVING COUNT(cp.IDCurso) > 2;
```

No tengo profesores que imparten más de dos cursos, es una escuela super privada y exclusiva 😊

12. Consulta Compleja 3: Lista los nombres de los programas de estudio y, para cada programa, el nombre del curso con el promedio de calificación más alto.

```
WITH promedioscursos AS (
    SELECT
        pc.idprograma,
        pc.idcurso,
        AVG(i.calificacion) AS promedio
    FROM
        programascursos pc
    JOIN inscripciones i ON pc.idcurso = i.idcurso
    GROUP BY
        pc.idprograma, pc.idcurso
)
SELECT
    pe.nombreprograma,
    c.nombrecurso,
    p.promedio
FROM
    promedioscursos p
JOIN cursos c ON p.idcurso = c.idcurso
JOIN programastestudio pe ON p.idprograma = pe.idprograma
WHERE
    (p.idprograma, p.promedio) IN (
        SELECT
            idprograma,
            MAX(promedio)
        FROM
            promedioscursos
        GROUP BY idprograma
    );

```

	nombreprograma	nombrecurso	promedio
1	Programa Ingeniería	Programación Avanzada	90
2	Programa Diseño	Diseño Digital	92
3	Programa Psicología	Psicología General	90