LABORATORIO

Bases de Datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un Sistema de Gestión de Bases de Datos (del inglés Database Management System o DBMS).

DATO: Un dato es una representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa que por si solos no tiene significado.

Un dato por sí mismo no constituye información, es el procesamiento de los datos lo que nos proporciona información.

Según la variabilidad de la base de datos:

- ·Bases de datos estáticas.
- ·Bases de datos dinámicas.

Sistema de Administración de Bases de Datos (DBMS)

Las bases de datos requieren de un software que permita la administración de dicha base de datos. Estos programas especializados sirven como interfaz para que los usuarios puedan, administrar como se estructura y optimiza toda la información recopilada. Un sistema de administración de bases de datos también permite un gran número de operaciones relacionadas con la administración, tal como, supervisar la productividad, ajustes, backups y restauración de los datos.

Entre los gestores de bases de datos o DBMS más conocidos se encuentran: Microsoft SQL Server, MySQL, Oracle Database, Microsoft Access, FileMaker, y dBASE.

MODELOS DE BASES DE DATOS

- **Bases de datos jerárquicas:** En este modelo los datos se organizan en forma de árbol invertido (algunos dicen raíz), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

- **Base de datos de red:** Este es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

- **Bases de datos transaccionales:** Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades.

- **Bases de datos orientadas a objetos:** Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:
  - **Encapsulación -** Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
  - **Herencia -** Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
  - **Polimorfismo -** Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

- **Bases de datos no relacionales o NoSQL:** Usan un modelo de almacenamiento que está optimizado para los requisitos específicos del tipo de datos que se almacena. Por ejemplo, los datos se pueden almacenar como pares clave/valor simple, como documentos JSON o como un grafo que consta de bordes y vértices.

- **Bases de datos relacionales o SQL:** Este es el modelo utilizado en la actualidad para representar problemas reales y administrar datos dinámicamente. En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. Virtualmente, todos los sistemas de bases de datos relacionales utilizan SQL (Structured Query Language) para consultar y mantener la base de datos.

NORMALIZACIÓN DE BASES DE DATOS RELACIONALES

En bases de datos relacionales, las formas normales son un conjunto de reglas diseñadas para organizar los datos de una manera que minimice la redundancia y dependencias no deseadas. Estas reglas son parte del proceso conocido como normalización, cuyo objetivo es asegurar la consistencia y eficiencia de los datos dentro de la base de datos.

Existen varias formas normales (NF), y cada una de ellas aborda problemas específicos en la estructura de los datos. A continuación se explican las tres primeras y más utilizadas formas normales.

TIPOS DE NORMALIZACIÓN

Primera Forma Normal (1NF)

Una tabla está en Primera Forma Normal (1NF) si:

1. Cada columna contiene valores atómicos (indivisibles), es decir, no debe contener conjuntos o listas de valores.
2. Cada valor en una columna debe ser del mismo tipo (por ejemplo, no puede haber un número en una columna que está destinada a contener fechas).
3. Debe tener una clave primaria definida, que identifique de manera única cada fila.

Ejemplo que NO está en 1NF:

ID Cliente	Nombre	Teléfonos
1	Juan	123456, 654321

En este ejemplo, la columna "Teléfonos" tiene más de un valor para el mismo cliente, lo cual viola la primera forma normal. La solución es dividir los números de teléfono en filas separadas:
Ejemplo en 1NF:

ID Cliente	Nombre	Teléfonos
1	Juan	123456
1	Juan	654321

Segunda Forma Normal (2NF)

Una tabla está en Segunda Forma Normal (2NF) si:

1. Cumple con todas las reglas de la 1NF.
2. Cada columna que no forma parte de la clave primaria depende totalmente de la clave primaria, es decir, no puede haber dependencias parciales.

Las dependencias parciales ocurren cuando solo una parte de la clave primaria compuesta determina el valor de una columna. Esto se aplica solo a las tablas que tienen claves primarias compuestas (es decir, formadas por más de una columna).

Ejemplo que NO está en 2NF:

ID Pedido	ID Producto	Cantidad	Nombre Producto
101	1	5	Notebook

Aquí, la columna "Nombre Producto" depende solo del ID Producto y no del ID Pedido. Para normalizar la tabla a 2NF, debemos eliminar esta dependencia parcial.
Ejemplo en 2NF:

1. Tabla Pedidos:

ID Pedido	ID Producto	Cantidad
101	1	5

2. Tabla Productos:

ID Producto	Nombre Producto
1	Notebook

Tercera Forma Normal (3NF)

Una tabla está en Tercera Forma Normal (3NF) si:

1. Cumple con todas las reglas de la 2NF.
2. No existen dependencias transitivas. Esto significa que las columnas que no forman parte de la clave primaria no deben depender de otras columnas no clave.

Las dependencias transitivas ocurren cuando una columna depende indirectamente de la clave primaria a través de otra columna.

Ejemplo que NO está en 3NF:

ID Pedido	ID Cliente	Nombre cliente	Dirección
101	1	Juan Pérez	Calle Falsa 123

En este ejemplo, la columna "Dirección" depende del ID Cliente, no del ID Pedido. Esto significa que tenemos una dependencia transitiva a través del Nombre Cliente. Para normalizar la tabla a 3NF, debemos mover la información del cliente a una tabla separada.
Ejemplo en 3NF:

1. Tabla Pedidos:

ID Pedido	ID Cliente
101	1

2. Tabla Clientes:

ID Cliente	Nombre cliente	Dirección
1	Juan Pérez	Calle Falsa 123

Otras Formas Normales

Existen formas normales más avanzadas como la Cuarta (4NF) y la Quinta (5NF), que abordan problemas más complejos relacionados con dependencias multivaloradas y la reducción de redundancias en estructuras más avanzadas de datos. Sin embargo, en la mayoría de las aplicaciones prácticas, normalizar una base de datos hasta la Tercera Forma Normal (3NF) es suficiente para garantizar una buena estructura de datos.

ESTRUCTURA BÁSICA DE UNA TABLA DE UNA BASE DE DATOS SQL

La estructura básica de una tabla en una base de datos SQL consta de varios componentes que definen cómo se almacenan y organizan los datos en esa tabla. A continuación, se presentan los elementos clave de la estructura de una tabla:

- **Nombre de la tabla:** Es el nombre único que se le da a la tabla. Debe seguir las reglas de nomenclatura, como no contener espacios ni caracteres especiales, y generalmente se utiliza en singular (por ejemplo, "Empleado" en lugar de "Empleados").

- **Columnas o campos:** Cada tabla consta de una serie de columnas que representan los atributos o propiedades de los datos que se almacenan. Cada columna tiene un nombre único y un tipo de datos que especifica qué tipo de información se almacenará en esa columna (por ejemplo, VARCHAR para texto, INT para enteros, DATE para fechas).

- **Clave primaria (Primary Key):** Es un campo o una combinación de campos que se utiliza para identificar de manera única cada fila en la tabla. La clave primaria garantiza que no haya duplicados y permite una rápida búsqueda y recuperación de datos. Generalmente, se denota con el atributo PRIMARY KEY y se puede comprender una o más columnas.

- **Clave foránea (Foreign Key):** Es un campo que establece una relación entre dos tablas. La clave foránea se usa para vincular registros en la tabla actual con registros en otra tabla. Ayuda a mantener la integridad referencial de la base de datos. Se define con el atributo FOREIGN KEY.

- **Restricciones (Constraints):** Las restricciones son reglas que se aplican a las columnas para garantizar la integridad de los datos. Algunas restricciones comunes son NOT NULL (para garantizar que un campo no esté vacío), UNIQUE (para garantizar que los valores sean únicos) y CHECK (para imponer condiciones específicas en los valores de una columna).

- **Otros atributos:** Además de los elementos anteriores, una tabla puede tener otros atributos, como restricciones adicionales, índices (para mejorar el rendimiento de las consultas), comentarios, etc.

MySQL Workbench

MySQL es un sistema de gestión de bases de datos relacional ampliamente utilizado. MySQL Workbench es una herramienta gráfica que facilita la administración y el desarrollo de bases de datos MySQL. A continuación, te explicaremos cómo instalar MySQL Workbench y crear una base de datos para un colegio como ejemplo.

Instalación de MySQL Workbench

Descarga MySQL Workbench desde el sitio web oficial:

https://dev.mysql.com/downloads/workbench/

- Sigue las instrucciones de instalación proporcionadas para tu sistema operativo.
- Deberá hacer la instalación "FULL".
- Durante la instalación, se te pedirá configurar una contraseña para el usuario "root" de MySQL. Asegúrate de recordar esta contraseña, ya que la necesitarás para conectarte a MySQL.

Ejemplo de Base de Datos para un Colegio

En este ejemplo, crearemos una base de datos para un colegio con 4 tablas: Alumnos, Cursos, Profesores y Materias. También definiremos las relaciones entre estas tablas.

Creación de la Base de Datos

Abre MySQL Workbench y conéctate a tu servidor MySQL utilizando la contraseña que configuraste durante la instalación e ingresa el siguiente código:

```
-- Crear la base de datos
CREATE DATABASE IF NOT EXISTS escuela;

-- Seleccionar la base de datos
USE escuela;

# BORRAR DB Y TABLAS:
#ÚSENLO CON MUCHA PRECAUCIÓN.
DROP DATABASE nombre_DB;
```

Creación de las Tablas

Tabla Alumnos:

```
-- Crear la tabla Alumnos:
CREATE Table Alumnos(
    DNI INT PRIMARY KEY,
    nombre VARCHAR(255),
    apellido VARCHAR(255),
    curso VARCHAR(10)
 );
```

En esta tabla, hemos creado una columna DNI como clave primaria para identificar de manera única a cada alumno. La columna Curso servirá como clave foránea para establecer una relación con la tabla Cursos.

Tabla Cursos:

```
-- Crear la tabla Cursos:
CREATE Table Cursos(
    idCurso INT AUTO_INCREMENT PRIMARY KEY,
   curso VARCHAR(255)
 );
```

Hemos creado una columna idCurso como clave primaria para identificar los cursos de manera única y autoincremental.

Tabla Profesores:

```
-- Crear la tabla Profesores:
CREATE Table Profesores(
    DNI INT PRIMARY KEY,
    nombre VARCHAR(255),
    apellido VARCHAR(255),
    materia INT
 );
```

Al igual que en la tabla alumnos, hemos creado una columna DNI como clave primaria para identificar de manera única a cada profesor. La columna Materia servirá como clave foránea para establecer una relación con la tabla Materias.

Tabla Materias:

```
-- Crear la tabla Materias:
CREATE Table Materias(
    idMateria INT PRIMARY KEY,
    nombreMateria VARCHAR(255)
 );
```

La columna IDMateria es la clave primaria para identificar las materias de manera única.

Modificación de las Columnas

Para modificar una columna existente en una tabla, puedes utilizar la sentencia ALTER TABLE. Por ejemplo, si deseas agregar una columna "Edad" a la tabla Alumnos, puedes hacerlo de la siguiente manera:

```
-- Modificar Tabla Alumno:
ALTER TABLE Alumnos
ADD Edad INT;
```

A continuación, haremos una modificación sobre la tabla Alumnos para que sea apropiada la creación de la llave foránea:

```
-- Modificar columna curso en tabla Alumnos:
ALTER TABLE Alumnos
MODIFY curso INT;
```

A continuación, haremos una modificación sobre la tabla Alumnos para que sea apropiada la creación de la llave foránea:

```
-- Agregar FK a tabla Alumnos:
ALTER TABLE Alumnos
ADD CONSTRAINT FK_Alumnos_curso
 FOREIGN KEY (curso)
 REFERENCES Cursos(idCurso);
```

```
-- Agregar FK a tabla Profesores:
ALTER TABLE Profesores
ADD CONSTRAINT FK_Profesores_Materia
 FOREIGN KEY (materia)
 REFERENCES Materias(idMateria);
```

Ingreso de Datos en las Tablas

Puedes ingresar datos en las tablas utilizando la sentencia INSERT INTO. Aquí tienes un ejemplo para agregar datos a la tabla “Cursos”, “Alumnos”, “Materias” y “Profesores”:

```
INSERT INTO Cursos (curso)
 VALUES
 ('1º año'),
 ('2º año'),
 ('3º año'),
 ('4º año'),
 ('5º año');
```

```
INSERT INTO Alumnos (DNI, nombre, apellido, curso)
 VALUES
 (1234578, 'Juana','Perez',1),
 (2345689, 'Luis','González',1),
 (3456790, 'Ana','Rodríguez',1),
 (4567801, 'Pedro','Martínez',2),
 (5678912, 'Sofía','López',2),
 (6789023, 'Diego','Fernández',2),
 (7890134, 'María','Torres',3),
 (8901245, 'Carlos','Sanchez',3),
 (9012356, 'Laura','Gómez',3),
 (9876532, 'Andrés','Pérez',4),
 (8765421, 'Luisa','Ramírez',4),
 (7654310, 'Javier','García',4),
 (5432198, 'Marta','Díaz',5),
 (4321087, 'Joaquín','Mendoza',5),
 (3210976, 'Valentina','Hernández',5);
```

```
INSERT INTO Materias (idMateria, nombreMateria)
 VALUES
    (1, 'Matemáticas'),
    (2, 'Historia'),
    (3, 'Ciencias'),
    (4, 'Literatura'),
    (5, 'Inglés'),
    (6, 'Geografía'),
    (7, 'Química'),
    (8, 'Biología'),
    (9, 'Música'),
    (10, 'Educación Física');
```

```
INSERT INTO Profesores (DNI, nombre, apellido, materia)
 VALUES
    (12345678, 'Carlos', 'Chávez', 1),
    (23456789, 'Laura', 'Gómez', 2),
    (34567890, 'Javier', 'Hernández', 3),
    (45678901, 'Sofía', 'Martínez', 4),
    (56789012, 'Andrés', 'Pérez', 5),
    (67890123, 'Valentina', 'Díaz', 6),
    (78901234, 'Luis', 'Ramírez', 7),
    (89012345, 'María', 'Sánchez', 8),
    (90123456, 'Diego', 'García', 9),
    (98765432, 'Joaquín', 'Mendoza', 10);
```

Búsquedas y Consultas

Las búsquedas de información en MySQL se realizan con el comando SELECT, y existen diversas combinaciones que se pueden realizar, así como combinar búsquedas en distintas tablas.

Aquí algunos ejemplos:

```
SELECT * FROM Alumnos WHERE curso=5;

SELECT * FROM Alumnos WHERE curso=1 OR curso=5;

SELECT apellido, nombre FROM Profesores;

SELECT * FROM Profesores;

SELECT apellido, nombre FROM Profesores WHERE materia=1 OR materia=2;

SELECT Profesores.nombre, Profesores.apellido, Materias.nombreMateria AS materia
FROM Profesores
INNER JOIN Materias ON Profesores.materia = Materias.idMateria;
```

Ejemplo de Base de Datos: BIBLIOTECA

Objetivo: Diseñar una base de datos para gestionar la información de una biblioteca, que incluya detalles sobre libros, autores, categorías y préstamos a usuarios.

Requerimientos:

1. Autores:
   - Cada autor debe tener un identificador único (AutorID), su nombre y nacionalidad.
   - No se deben permitir autores duplicados en la base de datos.
2. Libros:
   - Cada libro debe tener un identificador único (LibroID), un título, la fecha de publicación y la referencia al autor (AutorID).
   - Un libro puede tener solo un autor, pero un autor puede haber escrito varios libros.
3. Categorías:
   - Cada categoría debe tener un identificador único (CategoriaID) y un nombre que no se repita en la base de datos.
   - Un libro puede pertenecer a varias categorías, y una categoría puede incluir varios libros.
4. Usuarios:
   - Cada usuario debe tener un identificador único (UsuarioID), un nombre y un correo electrónico, que también debe ser único.
   - Los usuarios pueden tomar prestados varios libros.
5. Préstamos:
   - La base de datos debe registrar los préstamos de libros a usuarios, incluyendo un identificador único (PrestamoID), la referencia al libro (LibroID), la referencia al usuario (UsuarioID), la fecha de préstamo y la fecha de devolución.

Estructura:

- Autores: Para almacenar información sobre los autores.
- Libros: Para almacenar detalles sobre los libros y su relación con los autores.
- Categorias: Para gestionar las categorías de los libros.
- Libros_Categorias: Una tabla intermedia que relaciona libros con categorías.
- Usuarios: Para almacenar información sobre los usuarios de la biblioteca.
- Prestamos: Para registrar los préstamos de libros a los usuarios.

Normalización:

- FN1: Se garantizará que todas las tablas tengan claves primarias y que los atributos sean atómicos.
- FN2: Se evitarán las dependencias parciales en las tablas.
- FN3: No habrá dependencias transitivas entre los atributos no clave.

```sql
-- Crear base de datos
CREATE DATABASE Biblioteca;
USE Biblioteca;

-- Forma Normal 1 (FN1)
-- Tabla Autores
CREATE TABLE Autores (
    AutorID INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(100) NOT NULL,
    Nacionalidad VARCHAR(50)
);

-- Tabla Libros
CREATE TABLE Libros (
    LibroID INT PRIMARY KEY AUTO_INCREMENT,
    Titulo VARCHAR(200) NOT NULL,
    AutorID INT,
    FechaPublicacion DATE,
    FOREIGN KEY (AutorID) REFERENCES Autores(AutorID)
);

-- Forma Normal 2 (FN2)
-- Tabla Categorías
CREATE TABLE Categorias (
    CategoriaID INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(100) NOT NULL UNIQUE
);

-- Relación entre Libros y Categorías
CREATE TABLE Libros_Categorias (
    LibroID INT,
    CategoriaID INT,
    PRIMARY KEY (LibroID, CategoriaID),
    FOREIGN KEY (LibroID) REFERENCES Libros(LibroID),
    FOREIGN KEY (CategoriaID) REFERENCES Categorias(CategoriaID)
);
```

```sql
-- Forma Normal 3 (FN3)
-- Tabla Usuarios
CREATE TABLE Usuarios (
    UsuarioID INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE
);

-- Tabla Préstamos
CREATE TABLE Prestamos (
    PrestamoID INT PRIMARY KEY AUTO_INCREMENT,
    LibroID INT,
    UsuarioID INT,
    FechaPrestamo DATE NOT NULL,
    FechaDevolucion DATE,
    FOREIGN KEY (LibroID) REFERENCES Libros(LibroID),
    FOREIGN KEY (UsuarioID) REFERENCES Usuarios(UsuarioID)
);
```

La tabla ***Categorias*** está en FN1 y también en FN2 porque:

- Tiene una clave primaria.
- No tiene atributos que violen la unicidad.
- Cumple con la dependencia completa en relación con su clave primaria.

En este caso, se considera que la tabla está en ambas formas normales.

La tabla ***Usuarios*** está en:

- FN1: Por tener una clave primaria y atributos atómicos.
- FN2: Por cumplir con la dependencia completa de todos los atributos no clave respecto a la clave primaria.
- FN3: Porque no hay dependencias transitivas en la tabla.

Aunque no tiene relaciones con otras tablas, los requisitos de FN3 se cumplen en este caso.

EJERCICIOS PRÁCTICOS

En cada ejercicio, se pide crear una base de datos en MySQL con las tablas correspondientes.

- Usa CREATE TABLE para definir cada tabla.
- Las tablas en NF1 deberán tener claves primarias (PRIMARY KEY).
- Las tablas en NF2 deberán establecer relaciones usando claves foráneas (FOREIGN KEY).
- Realiza as consultas con JOIN para unir los datos de las tablas relacionadas y obtener la información solicitada.
- Se `puede cambiar el tipo de forma normal, o crear nuevas tablas con distintas relaciones.

Ejercicio 1: Gestión de Hospitales

- Tablas en NF1:
  - Departamentos: idDepartamento, nombre, númeroCamas.
  - Pacientes: idPaciente, nombrePaciente, edad, género.
- Tablas en NF2:
  - PacientesEnDepartamentos: idPaciente, idDepartamento.
  - Doctores: idDoctor, nombreDoctor, especialidad, idDepartamento.

Prácticas:

1. Crea las tablas mencionadas, respetando las reglas de 1NF, 2NF o 3NF, según crea conveniente.
2. Inserta registros de 6 departamentos, 40 pacientes, y 8 doctores.
3. Haz una consulta que muestre el departamento al que pertenece cada paciente.
4. Realiza una consulta que muestre todos los doctores y los pacientes que pertenecen a su departamento.
5. Encuentra qué doctores tienen más pacientes en sus departamentos.

Ejercicio 2: Gestión de Biblioteca

- Tablas en NF1:
  - Libros: idLibro, titulo, autor.
  - Categorías: idCategoría, nombreCategoría.
- Tablas en NF2 o NF3:
  - LibrosEnCategorías: idLibro, idCategoría.
  - Bibliotecarios: idBibliotecario, nombreBibliotecario, idCategoría.

Prácticas:

1. Crea las tablas con las relaciones correctamente definidas.
2. Inserta al menos 5 libros, 3 categorías y 2 bibliotecarios.
3. Haz una consulta para ver qué libros pertenecen a cada categoría.
4. Consulta qué bibliotecario está a cargo de qué categorías de libros.
5. Realiza una búsqueda para ver todos los libros de un bibliotecario según la categoría que gestiona.

Ejercicio 3: Gestión de Tienda de Ropa

- Tablas en NF1:
  - Productos: idProducto, nombreProducto, precio, talla.
  - CategoríasProducto: idCategoría, nombreCategoría.
- Tablas en NF2 o NF3:
  - ProductosEnCategorías: idProducto, idCategoría.
  - Vendedores: idVendedor, nombreVendedor, idCategoría.

Prácticas:

1. Crea las tablas para almacenar información sobre productos y vendedores.
2. Inserta 10 productos, 4 categorías de productos, y 3 vendedores.
3. Haz una consulta que liste los productos por categoría.
4. Muestra los vendedores y las categorías de productos que gestionan.
5. Encuentra qué categoría de productos tiene el precio promedio más alto.

Ejercicio 4: Gestión de Restaurante

- Tablas en NF1:
  - Platillos: idPlatillo, nombrePlatillo, precio.
  - CategoríasPlatillo: idCategoría, nombreCategoría.
- Tablas en NF2 o NF3:
  - PlatillosEnCategorías: idPlatillo, idCategoría.
  - Cocineros: idCocinero, nombreCocinero, idCategoría.

Prácticas:

1. Crea las tablas de platillos y categorías según las reglas de NF1 y NF2.
2. Inserta 6 platillos, 3 categorías de platillos y 2 cocineros.
3. Haz una consulta que muestre los platillos agrupados por categoría.
4. Muestra qué cocineros están asignados a qué categorías de platillos.
5. Realiza una consulta que muestre el cocinero con los platillos más caros.

Ejercicio 5: Gestión de Empresa de Software

- Tablas en NF1:
  - Proyectos: idProyecto, nombreProyecto, fechaInicio
  - Tecnologías: idTecnología, nombreTecnología
- Tablas en NF2 o NF3:
  - ProyectosConTecnologías: idProyecto, idTecnología
  - Desarrolladores: idDesarrollador, nombreDesarrollador, idTecnología

Prácticas:

1. Crea las tablas de proyectos y tecnologías de acuerdo a las formas normales.
2. Inserta 5 proyectos, 4 tecnologías y 3 desarrolladores.
3. Haz una consulta que muestre qué tecnologías se utilizan en cada proyecto.
4. Consulta los desarrolladores asignados a las tecnologías de cada proyecto.
5. Encuentra el proyecto que ha utilizado más tecnologías.

Ejercicio 6: Sistema de Gestión de Estudiantes

Diseñar una base de datos para gestionar la información de estudiantes, cursos y matrículas en una institución educativa.

Requerimientos:

1. Estudiantes:
   - Cada estudiante debe tener un identificador único (EstudianteID), nombre, fecha de nacimiento y correo electrónico.
   - No se deben permitir estudiantes duplicados.
2. Cursos:
   - Cada curso debe tener un identificador único (CursoID), nombre del curso, descripción y número de créditos.
   - Un curso puede ser tomado por varios estudiantes.
3. Matrículas:
   - La base de datos debe registrar las matrículas de los estudiantes en los cursos, incluyendo un identificador único (MatriculaID), la referencia al estudiante (EstudianteID), la referencia al curso (CursoID), la fecha de matrícula y el estado de la matrícula (activa/inactiva).

Estructura:

- La base de datos constará de las siguientes tablas:
  - Estudiantes: Para almacenar información sobre los estudiantes.
  - Cursos: Para gestionar la información de los cursos ofrecidos.
  - Matriculas: Para registrar la relación entre estudiantes y cursos.

Ejercicio 7: Sistema de Gestión de Ventas

Diseñar una base de datos para gestionar la información de productos, clientes y ventas en una tienda.

Requerimientos:

1. Productos:
   - Cada producto debe tener un identificador único (ProductoID), nombre, precio y cantidad en stock.
   - No se deben permitir productos duplicados.
2. Clientes:
   - Cada cliente debe tener un identificador único (ClienteID), nombre, dirección y número de teléfono.
   - Los clientes deben ser únicos en la base de datos.
3. Ventas:
   - La base de datos debe registrar las ventas realizadas, incluyendo un identificador único (VentaID), la referencia al cliente (ClienteID), la fecha de venta y el total de la venta.
4. Detalles de Ventas:
   - Se debe registrar la información de los productos vendidos en cada venta, incluyendo un identificador único (DetalleID), la referencia a la venta (VentaID), la referencia al producto (ProductoID) y la cantidad vendida.

Estructura:

- La base de datos constará de las siguientes tablas:
  - Productos: Para almacenar información sobre los productos disponibles en la tienda.
  - Clientes: Para gestionar la información de los clientes.
  - Ventas: Para registrar las ventas realizadas.
  - Detalles_Ventas: Para registrar los productos específicos vendidos en cada transacción.