Ejercicio 1: Explorando Tecnologías de Bases de Datos (Relacional vs NoSQL)

Caso de Uso:

Una startup de comercio electrónico necesita gestionar pedidos con las siguientes características:

- Cada pedido contiene múltiples productos.
- Cada pedido está asociado a un cliente.
- Los clientes pueden realizar múltiples pedidos en diferentes momentos.

Solución

1. Elección del tipo de base de datos y justificación

Se recomienda usar una **Base de datos relacional (RDBMS)**.

Justificación:

- **Estructura predecible:** Los datos de pedidos, clientes y productos tienen relaciones claras (uno a muchos, muchos a muchos) que se modelan naturalmente con tablas y claves foráneas.
- **Integridad transaccional:** Las operaciones de compra requieren ACID (ej: actualizar inventario + registrar pedido en una sola transacción).
- **Consultas complejas:** Necesidad de JOINs para reportes como "pedidos por cliente" o "productos más vendidos".
- **Caso típico:** Sistemas de comercio electrónico como Shopify o Magento usan RDBMS para gestionar órdenes.

¿Cuándo usar NoSQL? Si la startup priorizara escalabilidad horizontal o manejo de datos no estructurados (ej: logs de comportamiento de usuarios), podría considerarse MongoDB o DynamoDB.

2. Diseño del esquema relacional

Tablas propuestas:

1. **Clientes**
    - cliente_id (PK)
    - nombre
    - email
    - direccion

2. **Productos**
    - producto_id (PK)
    - nombre
    - precio
    - stock

3. **Pedidos**
    - pedido_id (PK)
    - cliente_id (FK → Clientes)
    - fecha_pedido
    - estado

4. **Detalles_Pedido** (Tabla intermedia para relación muchos a muchos)
    - detalle_id (PK)
    - pedido_id (FK → Pedidos)
    - producto_id (FK → Productos)
    - cantidad
    - precio_unitario

Diagrama conceptual:

Clientes → Pedidos (1:N)

Pedidos → Detalles_Pedido (1:N)

Productos → Detalles_Pedido (1: N)

3. Consultas SQL básicas

a) Todos los pedidos de un cliente específico (ej: cliente_id = 101):

```
SELECT p.pedido_id, p.fecha_pedido, p.estado
FROM Pedidos p
WHERE p.cliente_id = 101;
```

b) Productos de un pedido en particular (ej: pedido_id = 500):

```
SELECT pr.nombre, dp.cantidad, dp.precio_unitario
FROM Detalles_Pedido dp
JOIN Productos pr ON dp.producto_id = pr.producto_id
WHERE dp.pedido_id = 500;
```

c) Cantidad total de pedidos por cliente:

```
SELECT c.cliente_id, c.nombre, COUNT(p.pedido_id) AS total_pedidos
FROM Clientes c
LEFT JOIN Pedidos p ON c.cliente_id = p.cliente_id
GROUP BY c.cliente_id, c.nombre;
```

4. Comparación RDBMS vs NoSQL para este caso

Aspecto	RDBMS (MySQL/PostgreSQL)	NoSQL (MongoDB)
Esquema	Estructura fija (tablas). Ideal para datos estructurados.	Flexible (documentos JSON). Útil si los pedidos tienen atributos variables.
Transacciones	Soporte ACID completo (ej: actualizar stock + registrar pedido).	Transacciones limitadas (en versiones recientes de MongoDB).
Escalabilidad	Escalabilidad vertical (más costosa para grandes volúmenes).	Escalabilidad horizontal (mejor para crecimiento exponencial).
Consultas	JOIN's eficientes para relaciones complejas.	Consultas anidadas en documentos, menos eficientes para JOINs.

Conclusión: Para una startup en fase inicial con necesidades de integridad y consultas predecibles, un RDBMS es la opción más segura. NoSQL podría evaluarse en etapas posteriores si se requiere escalabilidad masiva o manejo de datos semiestructurados.