**Análisis de Caso: Amazon DynamoDB**

**1. Resumen Ejecutivo**

La empresa de comercio electrónico enfrenta problemas de rendimiento y latencia debido al uso de una base de datos relacional que no soporta adecuadamente picos de pedidos simultáneos. Se propone migrar la gestión de pedidos a **Amazon DynamoDB**, una base de datos NoSQL totalmente administrada, altamente escalable y con baja latencia, para garantizar disponibilidad y consistencia a gran escala, optimizando además costos de operación.

**2. Análisis del Problema**

* **Situación actual:**
  + La base de datos relacional presenta cuellos de botella en consultas concurrentes cuando los pedidos simultáneos aumentan.
  + La escalabilidad vertical de la base relacional implica costos elevados en hardware y licencias.
  + El modelo de esquema rígido limita la adaptación a cambios en los datos de pedidos y clientes.
* **Limitaciones clave:**
  + Escalabilidad limitada y no elástica.
  + Latencia alta en lecturas/escrituras con grandes volúmenes.
  + Alta dependencia de mantenimiento e infraestructura.

**3. Diseño de la Solución en DynamoDB**

* **Estructura principal:**
  + Crear una **tabla principal Pedidos** con:
    - **Clave de partición (partition key):** pedido\_id
    - **Clave de ordenamiento (sort key):** cliente\_id o fecha\_pedido para optimizar consultas por cliente o periodo.
* **Índices secundarios:**
  + **GSI (Índice Secundario Global):** por ejemplo, cliente\_id como clave de partición para consultas rápidas por cliente.
  + **LSI (Índice Secundario Local):** permitir consultas ordenadas por estado de pedido o fecha.
* **Modelado de acceso:**
  + Optimizar las consultas más comunes: pedidos por cliente, pedidos por estado, histórico de pedidos.
* **Escalabilidad:**
  + Implementar **particionamiento automático** para distribuir datos de forma uniforme y evitar hotspots.

**4. Evaluación de la Solución**

| **Aspecto** | **DynamoDB** | **Base de Datos Relacional** |
| --- | --- | --- |
| **Escalabilidad** | Escalabilidad horizontal automática y sin servidor. | Escalabilidad vertical costosa. |
| **Rendimiento** | Baja latencia, millones de transacciones por segundo. | Latencia alta bajo alta concurrencia. |
| **Costos** | Pago por uso, bajo mantenimiento. | Costos de licencias y hardware dedicados. |
| **Modelo de Consistencia** | Consistencia eventual por defecto (opcional consistencia fuerte). | Consistencia fuerte. |
| **Mantenimiento** | Totalmente administrado por AWS. | Requiere DBA y administración de servidores. |

**Desafíos:**

* Migración de datos sin afectar la operación.
* Rediseñar consultas para un modelo sin relaciones complejas.
* Controlar la consistencia eventual para operaciones críticas.

**5. Conclusiones y Recomendaciones Finales**

Migrar la gestión de pedidos a **Amazon DynamoDB** permitirá a la empresa soportar altos volúmenes de transacciones simultáneas con baja latencia y escalabilidad automática. Se recomienda:

* Realizar una **prueba piloto** para validar el rendimiento y la estructura de partición.
* Configurar políticas de autoescalado y establecer límites de costos.
* Integrar con otros servicios de AWS como **Lambda** y **API Gateway** para construir una arquitectura sin servidor y altamente disponible.