**1. Resumen Ejecutivo**

El presente informe analiza los problemas de rendimiento y escalabilidad que enfrenta una empresa de comercio electrónico debido al uso de una base de datos relacional tradicional. El crecimiento acelerado del negocio ha provocado latencias y una degradación en la experiencia del usuario durante peak de pedidos simultáneos.

Como solución, se propone la migración del sistema de gestión de pedidos a **Amazon DynamoDB**. Este cambio permitirá a la empresa manejar un volumen masivo de transacciones con una latencia constante de milisegundos, reducir los costos operativos a través de un modelo cloud y asegurar una infraestructura capaz de soportar el crecimiento futuro.

**2. Análisis del Problema**

La infraestructura de base de datos relacional actual presenta varios puntos débiles críticos que impiden el crecimiento del negocio:

* **Problemas de Escalabilidad:** Las bases de datos relacionales escalan verticalmente (aumentando la potencia de un solo servidor), una estrategia que es costosa y tiene límites técnicos. La escalabilidad horizontal (distribuir la carga en múltiples servidores) es compleja.
* **Degradación del Rendimiento:** Con el aumento de los datos, las consultas que requieren unir tablas de Pedidos, Clientes y otras se vuelven progresivamente más lentas. Esto causa mucha latencia.
* **Límites de Concurrencia:** Un alto número de pedidos simultáneos puede agotar la pool de conexiones de la base de datos, lo que provoca que las nuevas solicitudes fallen y se generen errores.

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

Se propone un diseño de **tabla única** que es la práctica recomendada para obtener el máximo rendimiento en DynamoDB.

* **Nombre de la Tabla:** PlataformaEcommerce
* **Clave Primaria:** Se utilizará una clave primaria compuesta para permitir el modelado de relaciones complejas.
  + **Clave de Partición (PK):** Un identificador genérico para la entidad (ej. CUSTOMER#<id>).
  + **Clave de Ordenación (SK):** Un identificador que define la relación o el tipo de dato (ej. PROFILE#<id>, ORDER#<id>).

**Diseño de Entidades y Patrones de Acceso**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entidad / Dato** | **PK (Clave de Partición)** | **SK (Clave de Ordenación)** | **Atributos Adicionales** | **Patrón de Acceso Soportado** |
| **Perfil de Cliente** | CUSTOMER#<CustomerID> | PROFILE#<CustomerID> | nombre, email, fecha\_registro | Obtener el perfil completo de un cliente. |
| **Pedido de un Cliente** | CUSTOMER#<CustomerID> | ORDER#<OrderID> | fecha\_pedido, estado, total, direccion | Obtener todos los pedidos de un cliente. |
| **Items de un Pedido** | ORDER#<OrderID> | ITEM#<ProductID> | nombre\_producto, cantidad, precio | Obtener todos los productos de un pedido específico. |

**Índices Secundarios Globales (GSI)**

Para soportar consultas adicionales, se proponen los siguientes GSI:

1. **GSI1: Buscar pedidos por estado.**
   * **Propósito:** Permite al equipo de logística encontrar eficientemente todos los pedidos con estado PENDIENTE.
   * **Claves:** GSI1\_PK = estado, GSI1\_SK = fecha\_pedido.
2. **GSI2: Buscar un pedido por su ID.**
   * **Propósito:** Permite al servicio de atención al cliente o al propio usuario buscar un pedido usando solo su OrderID, sin conocer el CustomerID.
   * **Claves:** GSI2\_PK = ORDER#<OrderID>, GSI2\_SK = ORDER#<OrderID>.

**4. Evaluación de la Solución**

La migración a DynamoDB presenta un balance claro de ventajas y desafíos en comparación con la solución relacional actual.

|  |  |  |
| --- | --- | --- |
| **Aspecto** | **Base de Datos Relacional (Actual)** | **Amazon DynamoDB (Propuesta)** |
| **Rendimiento** | Variable. Se degrada con el volumen de datos y la complejidad de los JOINs. | Predecible y constante. Latencia de milisegundos a cualquier escala. |
| **Escalabilidad** | Vertical. Costosa y con límites físicos. | Horizontal. Prácticamente ilimitada, automática y gestionada por AWS. |
| **Mantenimiento** | Alto. Requiere administración de servidores, parches, backups y optimización manual. | Cero. Es un servicio cloud, AWS gestiona toda la infraestructura subyacente. |
| **Modelo de Costos** | Fijo y de capital (servidores, licencias). Crece en grandes saltos. | Variable y operativo. Se paga por uso, alineando costos con el tráfico real. |
| **Flexibilidad de Consultas** | Alta. El lenguaje SQL permite consultas ad-hoc y JOINs complejos. | Baja. Las consultas se limitan a los patrones de acceso definidos por las claves primarias y los GSI. |
| **Consistencia** | Fuerte por defecto (ACID), lo que puede introducir latencia en las escrituras. | Eventual por defecto (más rápida y barata). Ofrece la opción de consistencia fuerte a un mayor costo. |

**5. Conclusiones y Recomendaciones Finales**

La migración a Amazon DynamoDB es la solución técnica y estratégica recomendada para superar los desafíos actuales de la empresa. Permitirá sostener el crecimiento del negocio con una infraestructura de datos performante, escalable y costo-eficiente.

Se proponen las siguientes recomendaciones:

1. **Adoptar un Modelo de Capacidad On Demand** Configurar la tabla de DynamoDB en modo **On Demand** para que la capacidad de lectura y escritura escale automáticamente según la demanda del tráfico.
2. **Implementar una Arquitectura Orientada a Eventos:** Utilizar **DynamoDB Streams** en conjunto con **AWS Lambda**. Esto permite ejecutar lógica de negocio de forma automática en respuesta a eventos en la base de datos (ej. al crear un nuevo pedido, una función Lambda puede enviar un correo de confirmación y notificar al sistema de inventario).
3. **Desarrollar un Plan de Migración por Fases:** Para mitigar los riesgos, la migración debe realizarse de manera gradual.