**Diseño Conceptual para Componente PivotTable de Alto Rendimiento**

Entiendo que buscas una solución arquitectónica que balancee inteligentemente el procesamiento entre frontend y backend para grandes volúmenes de datos, manteniendo la flexibilidad de análisis ad-hoc. Aquí presento un diseño conceptual sin código específico:

**1. Arquitectura Híbrida Inteligente**

**Principio Fundamental**:

* **Backend para lo pesado**: Pre-procesamiento, índices, metadatos
* **Frontend para lo interactivo**: Agregaciones finales, visualización

Diagram

Code

**2. Componentes Clave**

**Capa de Frontend (Interactiva):**

* **Builder de consultas**: Traduce selecciones de usuario a árbol de operaciones
* **Cache local**: Almacena resultados intermedios
* **Worker pool**: Para procesamiento paralelo
* **Virtual scroll**: Renderizado eficiente

**Capa de Backend (Optimizadora):**

* **Query planner**: Convierte árbol en ejecución óptima
* **Adaptive sampling**: Muestreo estadístico cuando sea relevante
* **Materialized view selector**: Usa vistas precalculadas cuando aplique
* **Chunk dispatcher**: Envío de datos por bloques

**3. Flujo de Procesamiento Optimizado**

1. **Fase de Metadata**:
   * Frontend solicita valores únicos y estadísticas básicas
   * Backend responde con resúmenes comprimidos
2. **Fase de Pre-agregación**:
   * Cuando el usuario selecciona dimensiones:
     + Backend calcula combinaciones frecuentes
     + Envía múltiples niveles de granularidad
3. **Fase de Refinamiento**:
   * Frontend combina pre-agregados
   * Aplica filtros finales
   * Calcula métricas derivadas

**4. Estrategias de Optimización Clave**

**Para Datasets Masivos (>1M registros):**

* **Particionamiento inteligente**:
  + Por rangos temporales
  + Por dimensiones frecuentes
  + Por hotspots de datos
* **Doble capa de caché**:
  + Nivel 1: Resultados completos de consultas frecuentes
  + Nivel 2: Fragmentos de consultas parciales
* **Procesamiento diferido**:
  + Prioriza lo visible en pantalla
  + Precarga lo probable (predictivo)

**5. Balance Frontend/Backend**

| **Operación** | **Donde se ejecuta** | **Razón** |
| --- | --- | --- |
| Valores únicos | Backend | Requiere scan completo |
| Filtrado inicial | Backend | Reduce volumen temprano |
| Agregación básica | Backend | Optimizado con índices |
| Combinación dimensiones | Ambos | Backend para combinaciones frecuentes, Frontend para ad-hoc |
| Ordenamiento final | Frontend | Bajo costo con datos ya filtrados |
| Visualización | Frontend | Interactividad requiere latencia mínima |

**6. Patrones para Mantener Flexibilidad**

1. **Árbol de operaciones serializable**:
   * Representación JSON de la configuración pivote
   * Permite optimización en backend
2. **Adaptive Materialization**:
   * El backend decide qué pre-calcular
   * Basado en frecuencia de patrones de consulta
3. **Lazy Loading de Dimensiones**:
   * Carga valores únicos bajo demanda
   * Prioriza dimensiones más usadas

**7. Consideraciones de Implementación**

* **Protocolo de comunicación**: GraphQL para flexibilidad en consultas
* **Serialización eficiente**: Avro/Protobuf para transferencia
* **Gestión de memoria**: Pool de buffers reutilizables
* **Progreso**: Notificaciones de avance para UX

Este diseño mantiene la flexibilidad de análisis ad-hoc mientras optimiza el rendimiento mediante:

1. Delegación inteligente de operaciones
2. Pre-procesamiento estratégico
3. Transferencia selectiva de datos
4. Cómputo distribuido entre capas