Comparación de Datos entre Colecciones en MongoDB

1 Introducción

MongoDB es una base de datos NoSQL orientada a documentos que no tiene la misma capacidad de JOIN que las bases de datos relacionales. Sin embargo, existen diferentes métodos para comparar datos entre colecciones. En este documento exploramos tres métodos principales para realizar esta tarea.

2 Métodos para comparar datos entre colecciones

2.1 Usando `$lookup` para hacer un "JOIN"

En MongoDB 3.2 y versiones posteriores, podemos usar el operador `$lookup` dentro de una consulta de agregación para realizar operaciones similares a los JOIN de SQL:

```
{
  "aggregate": "pedidos",
  "pipeline": [
    {
      "$lookup": {
        "from": "productos",
        "localField": "producto_id",
        "foreignField": "_id",
        "as": "producto_info"
      }
    },
    {
      "$match": {
        "producto_info": { "$ne": [] }
      }
    }
  ]
}
```

Este ejemplo une la colección `pedidos` con `productos` donde `pedidos.producto_id` coincide con `productos._id`. Los documentos coincidentes de la colección `productos` se añaden al array `producto_info` de cada documento de `pedidos`.

2.2 Carga de datos y comparación manual

Cuando necesitamos realizar comparaciones más complejas que no se pueden expresar fácilmente con `$lookup`, podemos cargar los datos de una colección en la aplicación y realizar la comparación manualmente. Aquí mostramos cómo podría ser la consulta JSON para obtener los datos:

```
// Primera consulta para obtener usuarios activos
{
  "find": "usuarios",
  "filter": {
    "estado": "activo"
  }
}

// Segunda consulta utilizando los IDs obtenidos
{
  "find": "transacciones",
  "filter": {
    "usuario_id": {
      "$in": ["id1", "id2", "id3"] // IDs de usuarios
   activos
    },
    "monto": { "$gt": 1000 }
  }
}
```

Este método es flexible pero menos eficiente para grandes volúmenes de datos, ya que requiere cargar una colección completa (o parte de ella) en memoria.

2.3 Variables temporales en el pipeline de agregación

Para casos avanzados donde necesitamos acceder a los datos de ambas colecciones en diferentes etapas del pipeline, podemos combinar `$lookup` con otras operaciones:

```
{
  "aggregate": "ventas",
  "pipeline": [
    {
      "$lookup": {
        "from": "productos",
        "pipeline": [],
        "as": "todosLosProductos"
      }
    },
    {
      "$project": {
        "fecha": 1,
        "producto_id": 1,
```

```
        "cantidad": 1,
        "productoInfo": {
          "$arrayElemAt": [
            {
              "$filter": {
                "input": "$todosLosProductos",
                "as": "producto",
                "cond": { "$eq": ["$$producto._id", "
   $producto_id"] }
              }
            },
            0
          ]
        }
      }
    },
    {
      "$project": {
        "fecha": 1,
        "producto_id": 1,
        "cantidad": 1,
        "nombre_producto": "$productoInfo.nombre",
        "precio_unitario": "$productoInfo.precio",
        "valor_total": { "$multiply": ["$cantidad", "
   $productoInfo.precio"] }
      }
    },
    {
      "$match": {
        "valor_total": { "$gt": 500 }
      }
    }
  ]
}
```

Este método permite realizar cálculos complejos y filtrados usando datos de ambas colecciones sin necesidad de cargar todo en memoria en la aplicación.

3 Comparación entre métodos

Criterio	**$lookup**	**Comparación manual**	**Variables en pipeline**
Eficiencia	Buena para relaciones simples	Baja para grandes colecciones	Buena para operaciones complejas
Complejidad	Baja	Media	Alta
Flexibilidad	Limitada a igualdades	Alta	Alta
Uso de memoria	Bajo (en la base de datos)	Alto (en la aplicación)	Bajo-Medio (en la base de datos)

4 Conclusión

MongoDB ofrece diferentes enfoques para comparar datos entre colecciones. La elección del método adecuado dependerá del volumen de datos, la complejidad de las relaciones y el rendimiento requerido. Para operaciones simples, `$lookup` es la opción más directa. Para casos más complejos, la carga manual o el uso de variables temporales en el pipeline ofrecen mayor flexibilidad.