Bases de Datos 2020

Parcial 2: NoSQL

Sergio Canchi, Cristian Cardellino,
Ramiro Demasi, Diego Piloni

Contexto

La base de datos **analytics** contiene tres colecciones con información de servicios financieros. Las colecciones que contiene **analytics** son:

Colección	**Descripción**
accounts	Contiene información de las cuentas de los clientes.
customers	Contiene información de los clientes.
transactions	Contiene información de las transacciones de los clientes.

Para cargar la base de datos **analytics** ejecutar el siguiente comandos:

$ mongorestore --host <host> --drop --gzip --db analytics analytics/

Consignas

Operaciones CRUD

1. Listar el **username**, **email** y la dirección de los clientes que nacieron entre Febrero de 1990 y Junio de 1999, tienen exactamente cinco cuentas y sus nombres empiezan con una 'J' o 'R'.

2. Listar el **account_id** y **transaction_count** del top 10 de las cuentas con mayor cantidad de transacciones tal que al menos una de sus transacciones es una venta (**sell**) con una cantidad (**amount**) mayor a 9990.

3. Para todas las cuentas que tienen ambos productos 'Brokerage' y 'Commodity' (puede tener otros productos adicionales) se pide incrementar el límite en 500 y añadir un nuevo producto llamado 'Trackers' en el arreglo **products**.

Pipeline de Agregación

4. Crear la vista 'latest_transaction_date_per_account' que liste el account_id, y “fecha de la transacción más reciente” realizada por cada cuenta (no es necesario tener en cuenta si un account_id aparece en otro documento). Listar ordenados por fecha de transacción más reciente. (Hint: Recordar que el operador de agregación $max también puede usarse en $project para operar sobre arrays)

5. Listar el id de la cuenta junto con la suma y promedio del (monto) **total** de las transacciones de compra (**buy**) por cuenta. Limitar el resultado al top 10 de las cuentas que más transacciones de compras realizaron. (Hint: convertir el valor de campo total a double). El esquema de datos de salida debe ser el siguiente:

```
{
  account_id: "integer",
  buy_total_sum: "double",
  buy_total_average: "double"
}
```

6. Listar el username y nombre de los clientes junto con los productos asociados a sus cuentas (la lista de productos no debe contener elementos duplicados).
   Ejemplo parcial del resultado:

```
[
 {
  username: 'fmiller',
  name: 'Elizabeth Ray',
  products: [
   'Brokerage',
   'Commodity',
   'CurrencyService',
   'Derivatives',
   'InvestmentFund',
   'InvestmentStock',
   'Trackers'
  ]
 }
...
]
```

   (Hint 1 para una solución poco eficiente: El ejercicio puede ser resuelto con los stages y operadores visto en clases)
   (Hint 2 para una solución eficiente (es alternativa no es obligatoria): Se debe usar (i) el operador $reduce para combinar los distintos arreglos products relacionados a las cuentas de un cliente en combinación con (ii) el operador $setUnion para evitar nombre de productos duplicados)

Modelado de Datos

7. Evolucionar el modelo de datos de analytics de acuerdo a las siguientes necesidades:
   (i) Se debe poder almacenar reviews de productos realizados por los clientes. Los reviews contienen un conjunto de datos en común (nombre de producto, texto del review, rating, fecha del review y el cliente que realiza el review) y tienen otro datos que son propios de cada producto.
   (ii) El modelo de datos debe permitir recuperar la siguiente información realizando una sola consulta: Listar el rating promedio por producto dado un rango de fechas.
   (iii) Se debe poder agregar validación de esquema para los campos siguiendo algún criterio que considere adecuado.
   Importante: Justificar cada decisión tomada (reglas de modelado de relaciones/patrones de diseño/sentido común).

   Entonces en base a lo anterior se pide realizar los siguientes ítems:
   7.1 Crear al menos 2 reviews solo para los productos 'InvestmentStock' y 'InvestmentFund'. Imagine nombres de campos (con uno solo alcanza) para los datos que son propios de cada producto.
   7.2 Crear la consulta solicitada en (ii) con un rango de fechas concreto.
   7.3 Especificar la regla de validación de esquema usando json schema solicitado en (iii)
   7.4 Agregue un documento donde la validación sea exitosa y otro documento que falle.

Puntos a tener en cuenta

- Resolver las consultas sin vistas salvo que se lo requiera explícitamente.
- Mostrar únicamente los campos pedidos en la consigna.
- Buscar hacer la consulta de la forma más sencilla y eficiente posible.
- Se evaluará el correcto formato de las soluciones:
  - El código entregado debe ser legible.
  - Utilizar indentación de espacios de manera uniforme.

Entrega

- Se entregará un archivo comprimido `soluciones.js.gz` o `soluciones.zip` (con `soluciones.js` adentro) con todas las soluciones de todos los ejercicios. Separar las soluciones mediante comentarios de javascript.
- La entrega se hará mediante el **Aula Virtual** en el correspondiente apartado.
  - Tendrán hasta las 18:30 para que se considere una entrega completa. La recomendación es empezar a subir el archivo a las 18 hs.
  - Si se entrega después de esa hora, el límite serán las 19:00 y se descontará 1 punto por entrega tardía.
  - Después de las 19:00 se cerrará la entrega y el parcial se considerará desaprobado.