Universidad ORT Uruguay

Facultad de Ingeniería

Martin Alonso y Juan Carriquiry

Bases de datos 2

Segundo obligatorio doc parte 4

Grupo M5A

Docente: Cecilia Belleti

Docente: Tatiana Bellon

Formulario de Antecedentes
Curso Materia

Nro. Estudiante: **291799**
Nombres: **Martin**
Apellidos: **Alonso**
Grupo/Turno: **5A/Matutino**

Nro. Estudiante: **310190**
Nombres: **Juan**
Apellidos: **Carriquiry**
Grupo/Turno: **5A/Matutino**

1. Tabla de contenido

1) Datos de prueba:

a) Nota:

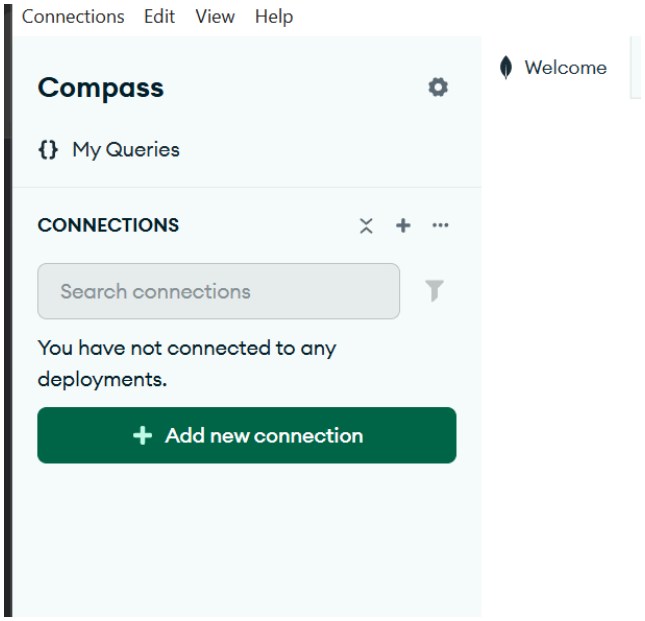
Es importante resaltar que hemos utilizado inteligencia artificial generativa como ChatGPT o Gemini para la generación de datos de prueba y darles un formato más atractivo y entendible.

b) Ubicación:

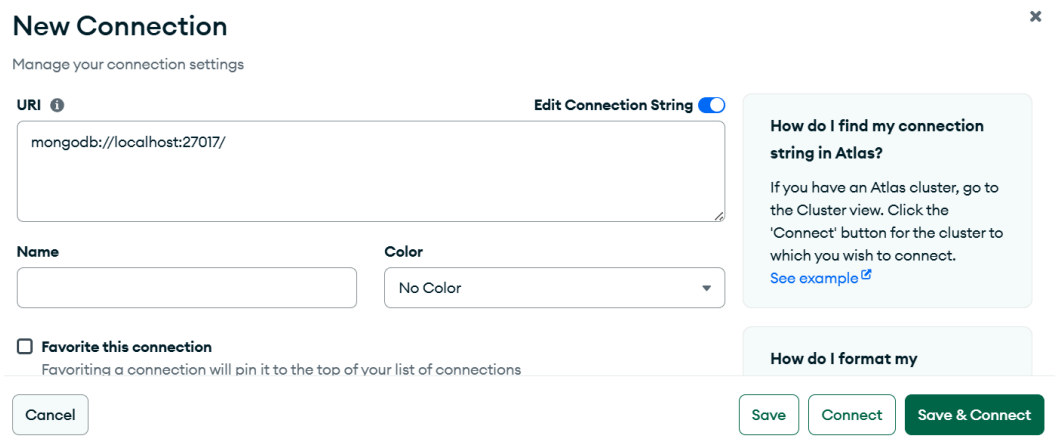
Los datos de prueba se encuentran dentro de la ruta SCRIPTS_NUEVOS/Parte4/DatosDePrueva.js

c) Creación de la conexión:

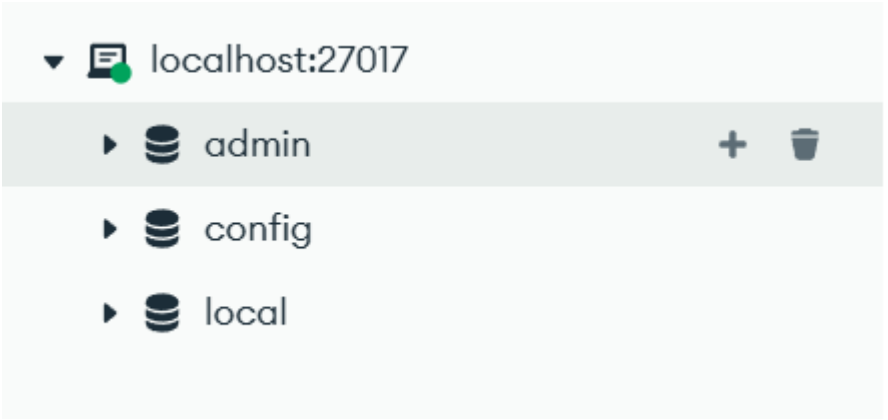
Abrimos MongoDbCompass y le damos a Add new connection:



En la ventana emergente simplemente le damos a Save & Connect:



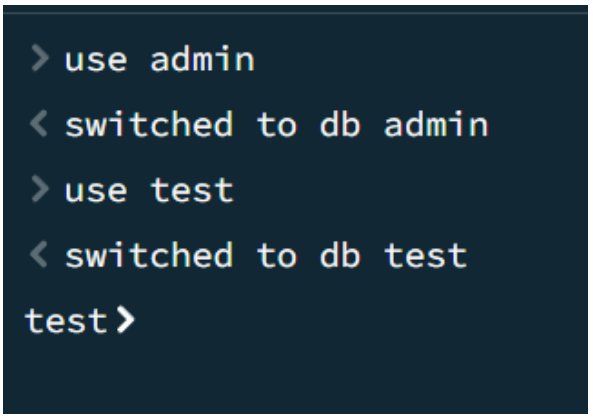
Deberíamos ver lo siguiente a la derecha debajo del menú de conexiones:



d) Uso de la conexión:

Abrimos la base de datos de admin (como se ve en gris en la foto anterior) y arriba a la derecha le damos a “Open MongoDB shell”:

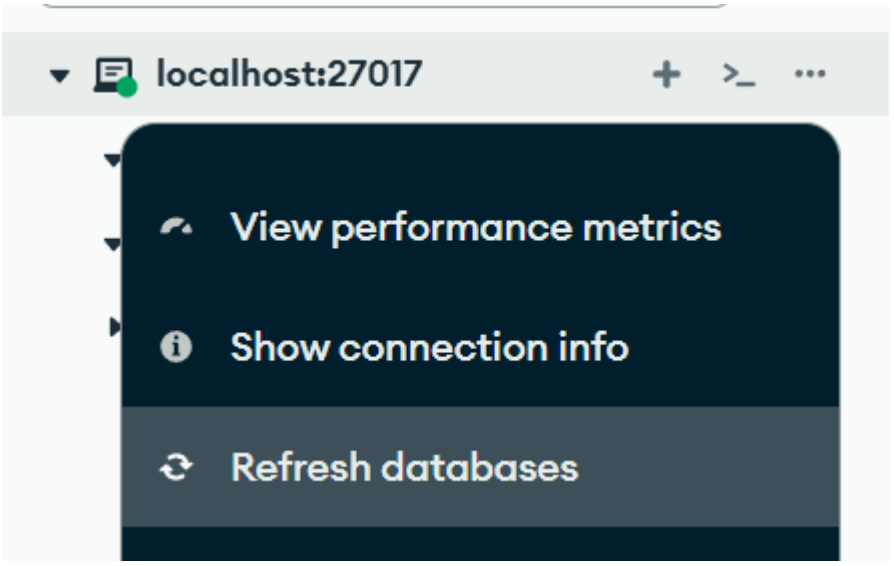
Escribimos “Use Test” para cambiarnos a la BD de testeo.



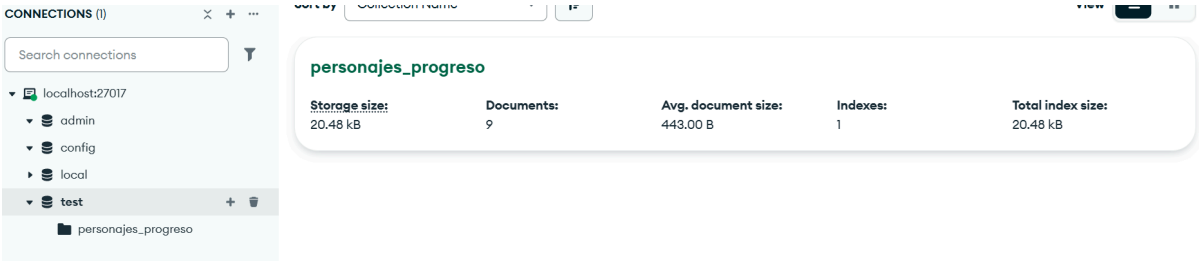
IMPORTANTE: Si es la primera vez que vamos a ejecutar este script podemos comentar la primera línea:
“db.personajes_progreso.deleteMany({});”

Y aquí podemos copiar y pegar los datos de prueba presentes en el Script. Luego de darle a enter deberíamos ver el siguiente mensaje de confirmación:

Para ver estos datos de una forma más gráfica podemos ir a la izquierda sobre la conexión de localhost:27017 a los tres puntitos de la derecha de esta coneccion y hacerle click a “Refresh databases”:



Por último vemos cómo aparece la database de “test” y le hacemos click para la colección:



Ahora ingresamos a la colección para ver los datos:

2) Explicaciones generales y supuestos:

a) Desarrollo por etapas:

Para solucionar todos los requerimientos usaremos etapas para el código, por ejemplo la etapa de descompresión ($unwind). Cada utilización de las etapas será explicada en su requerimiento específico.

b) Supuesto de horas:

Asumimos que en el requerimiento 3 “durante sus primeras 10 horas de juego” se puede traducir a que el/los jugadores objetivos tengan en total menos de 10 horas de juego. Significando que están durante sus primeras 10 horas de juego.

3) Requerimiento 1:

NOTA: Ubicación del script:

El script se encuentra presente en SCRIPTS_NUEVOS/Parte4/Parte4.1.js

a) Etapa Unwind:

Para esta etapa descomprimimos los logros de los personajes lo cual significa que el sistema generará X número de documentos para cada personaje siendo X la cantidad de logros que tiene, la diferencia con la forma en previa al unwind será que en vez de tener 1 documento de personaje con un array de X logros ahora tendremos X documentos de personajes idénticos pero con 1 solo logró por documento.

b) Etapa Match:

Ahora que tenemos todos los documentos preparados en esta etapa lo que haremos será filtrar por tipo de logro para cada documento quedándonos así solo con los documentos que tienen un logro de tipo “Exploración”

c) Etapa Group:

Posterior al Match ahora lo que se hace es volver a agrupar a los documentos resultantes basándonos en su id de personaje mientras llevamos la suma de cuantos documentos tiene cada personaje. En resumen lo que se hace es volver a agrupar para quedarnos con 1 documento por personaje (que haya pasado el filtro ya que si algún personaje no tiene logros de este tipo no tendrá documento resultante) pero ahora ese documento solo tiene los logros con tipo Exploración. Y cada vez que agrupamos 2 documentos sumamos 1 al contador de documentos (que es lo mismo que contar los logros de este tipo)

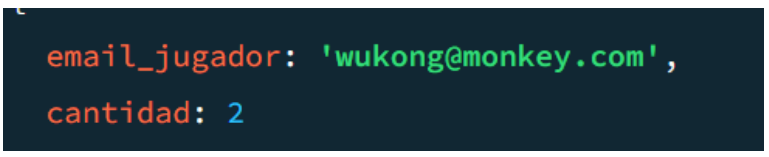
d) Etapa Project:

Por último lo que hacemos es preparar y ejecutar la salida. Para cada jugador devolvemos su id (gmail) y el contador calculado en la etapa anterior.

e) Evidencia de funcionamiento:

Según los datos de prueba presentados lo esperado sería que la consulta solo muestre que el personaje con "_id": "wukong@monkey.com" tiene 2 logros de exploración.

Ejecución:



Retorna lo esperado.

4) Requerimiento 2:

NOTA: Ubicación del script:

El script se encuentra presente en SCRIPTS_NUEVOS/Parte4/Parte4.2.js

a) Etapa Match:

En este requerimiento podemos directamente pasar a esta etapa ya que por lo que se quiere filtrar solo hay un documento embebido por personaje. Entonces directamente aplicamos el filtro para solo quedarnos con los documentos que tienen un valor (dentro del documento embebido de estadísticas) de progreso_general < 60.

b) Etapa Sort:

Ya que tenemos los documentos que vamos a querer mostrar aplicamos un SORT con un valor de -1 para que el sistema los ordene de forma descendente.

c) Etapa Limit:

Esta etapa lo único que hace es que de los documentos resultantes de la parte anterior limita a agarrar solo los primeros 5.

d) Etapa Project:

Por último lo que hacemos es preparar y ejecutar la salida. Para cada jugador devolvemos su id (gmail) y el contador calculado en la etapa anterior. su cantidad de enemigos derrotados (dentro del documento embebido de estadísticas) y su progreso general (también dentro del documento embebido de estadísticas).

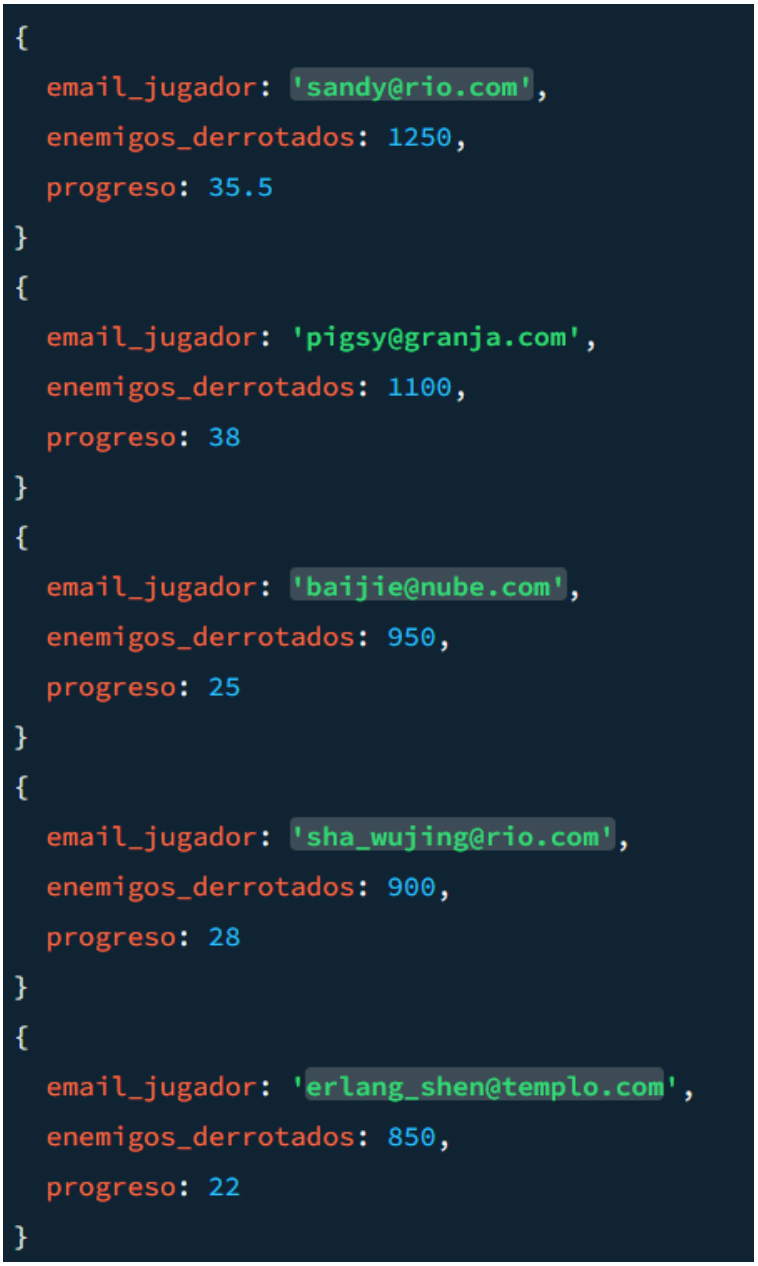
e) Evidencia de funcionamiento:

Según los datos de prueba presentados lo esperado sería que la consulta solo muestre datos requeridos de los personajes con emails (en este orden):

1) sandy@rio.com
2) pigsy@granja.com
3) baijie@nube.com
4) sha_wujing@rio.com

5) erlang_shen@templo.com

Ejecución:



Retorna lo esperado.

5) Requerimiento 3:

NOTA: Ubicación del script:

El script se encuentra presente en SCRIPTS_NUEVOS/Parte4/Parte4.3.js

a) Etapa Match:

Para este requerimiento ya que el campo para filtrar no está dentro del array de documentos embebidos podemos primero filtrar y luego hacer el unwind para aumentar un poco la eficiencia. Por lo que filtramos primero para cada usuario por su valor de horas jugadas <= 10.

b) Etapa Unwind:

Al igual que en la etapa Unwind de la parte 1 queremos contar los logros de los personajes por lo que descomprimimos el array.

c) Etapa Group:

A diferencia de la etapa Group de la parte 1 en la cual volvíamos a construir los archivos de los personajes contando para cada personaje aquí agrupamos por el nombre de cada logro a la vez que vamos contando la cantidad de cada uno.

d) Etapa Sort:

Ya fue explicado como funciona, volvemos a ordenar en orden descendente.

e) Etapa Limit:

Ya fue explicado como funciona, volvemos a agarrar los primero 5.

f) Etapa Project:

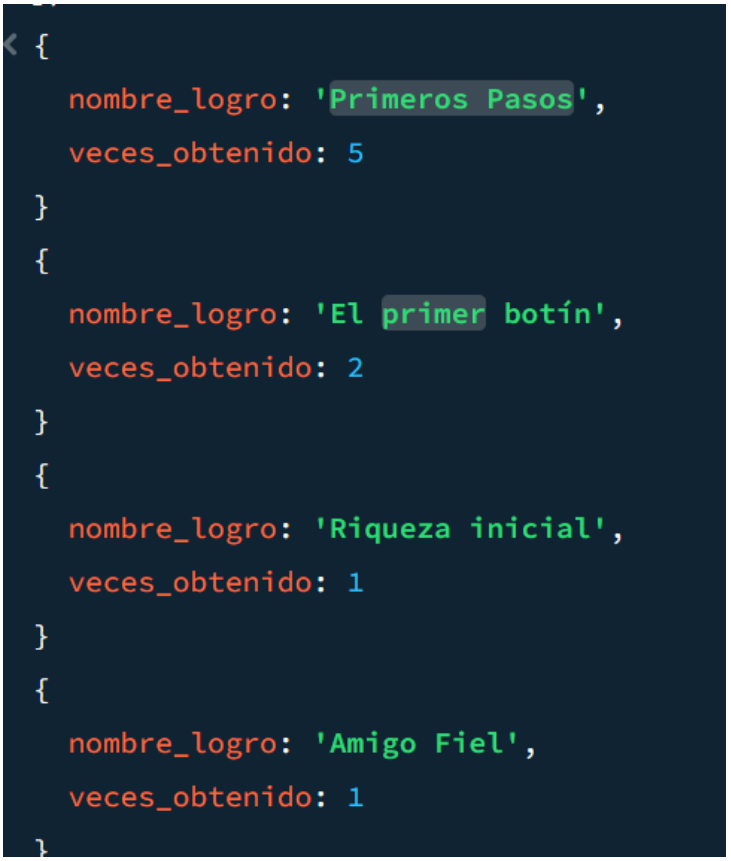
Para esta parte proyectamos el nombre del logro (que lo habíamos casteado dentro del nombre de variable “_id”) y el contador calculado en la etapa de Group.

g) Evidencia de funcionamiento:

Según los datos de prueba presentados lo esperado sería que la consulta solo muestre 4 logros debido a la limitación del filtro, pero igual mostrará datos requeridos de los logros con nombres(en este orden):

1) sandy@rio.com
2) pigsy@granja.com
3) baijie@nube.com
4) sha_wujing@rio.com
5) erlang_shen@templo.com

Ejecución:



Retorna lo esperado.