UNIVERSIDAD TECNICA FEDERICO SANTA MARIA | DEPARTAMENTO DE INFORMÁTICA

Tarea 3: MongoDB Steam++
INF-239 Bases de Datos

Profesores: Ricardo Salas (ricardo.salas@usm.cl)
Rodrigo Olavarría (rodrigo.olavarria@usm.cl)
Mauricio Figueroa (mauricio.figueroac@usm.cl)
Ayudante Coordinador: Gonzalo Alarcón(gonzalo.alarconc@usm.cl)
Octubre, 2024

1 Objetivo

Investigar y aplicar los conceptos básicos sobre el gestor de base de datos NoSQL MongoDB, utilizando herramientas como Docker para desplegar los servicios, MongoDB Query Language (MQL) y Python.

2 Especificaciones y reglas

El desarrollo de esta tarea debe cumplir las siguientes especificaciones, de lo contrario existiría un descuento en la nota final:

- Se debe ejecutar todos los procedimientos necesarios para implementar la Base de Datos MongoDB sobre Docker, bajo una arquitectura distribuida maestro-esclavo, utilizando Python como lenguaje de programación para realizar las operaciones que se requieran en esta tarea.

- La tarea debe ser entregada el 11 de noviembre hasta las 23:59 horas, cuya respectiva defensa es obligatoria.

2.1 Entregables

La tarea debe ser entregada como un archivo comprimido de la forma T3-ROL1-ROL2.zip y debe contener los siguientes archivos:

1. Archivo deploy-rol1-rol2.yml utilizado para desplegar los servicios sobre Docker con las especificaciones de arquitectura entregadas.
2. Archivo Jupyter Notebook o Google Colab pymongo-rol1-rol2.ipynb, el cuál debe venir ejecutado con todas las evidencias de los resultados documentadas en el mismo notebook, considerando una sección por cada requerimiento solicitado en la tarea.
3. Archivo usteam-rol1-rol2.json que contenga los datos actualizados y finales que den cuentas de las operaciones realizadas según los requerimientos de esta tarea.
4. Archivo README.txt el cual debe contener nombres, rol de los alumnos y las instrucciones para la correcta ejecución de su programa. Ante cualquier duda sobre algo que no aparezca especificado en la tarea y no afecte las reglas de esta, puede asumir lo que estime conveniente, pero debe ser especificado en el README, en caso de asumir eventos y no especificar habrá descuento.

2.2 Reglas

- El trabajo debe realizarse en parejas; no se aceptarán tareas individuales. Las copias serán evaluadas con nota 0 y se informará a las autoridades correspondientes. Para consultas, utilicen la plataforma oficial AULA.
- En el foro de búsqueda de pareja podrán buscar compañeros quienes no tengan, siendo esta una responsabilidad exclusiva del estudiante. En caso de problemas con su pareja, podrán contactar al profesor explicando su situación. Solo un alumno debe realizar la entrega.
- Si falla la ejecución de algún comando, no se asignará puntaje a este. Existe la posibilidad de que, durante su defensa, asista su profesor y realice preguntas. Es responsabilidad del estudiante inscribir un horario de defensa y estar presente en la fecha y hora elegida.
- Cada grupo tendrá un horario definido para su defensa; en caso de atraso, contarán con un tiempo menor para presentar su trabajo. La información respecto a la defensa se publicará eventualmente en AULA, lo que incluirá detalles sobre los descuentos. Es su obligación estar atentos a esta información y cumplir con lo establecido allí.

3 Descripción del problema

Se requiere diseñar, poblar y consultar una base de datos en **MongoDB** que almacena los datos de juegos[1] de la plataforma Steam [2], para lo cual se dispone del dataset llamado games.json. Esta fuente de datos es un JSON que recopila 85.103 documentos con los datos específicos de videojuegos para PC. La estructura de los documentos que almacena la colección de documentos games es la siguiente:


```
_id: ObjectId("6631d344cba7716b183e534f")
name: "DOOM Eternal: The Ancient Gods - Part Two"
release_date: "Mar 18, 2021"
required_age: 0
price: 19.99
dlc_count: 0
detailed_description: "The Ancient Gods – Part Two is the epic conclusion to the DOOM Slayer'..."
about_the_game: "The Ancient Gods – Part Two is the epic conclusion to the DOOM Slayer'..."
short_description: "The Ancient Gods – Part Two is the epic conclusion to the DOOM Slayer'..."
reviews: ""
header_image: "https://cdn.akamai.steamstatic.com/steam/apps/1098293/header.jpg?t=169..."
website: "https://slayersclub.bethesda.net/en"
support_url: "http://help.bethesda.net/"
support_email: ""
windows: true
mac: false
linux: false
metacritic_score: 0
metacritic_url: ""
achievements: 0
recommendations: 3860
notes: "Similar to DOOM from 2016, DOOM Eternal will contain Blood and Gore, I..."
> supported_languages: Array
> full_audio_languages: Array
> packages: Array
```


Figura 1: Estructura de documento de juego Steam

[1]Dataset y explicación de los datos en: https://www.kaggle.com/datasets/fronkongames/steam-games-dataset?resource=download

[2] https://store.steampowered.com/?l=spanish

4 Requisitos

1. Implementar la arquitectura de Cluster con un set de 3 réplicas. Una de las máquinas será el primario y las otras dos serán los secundarios. Esta configuración deberá estar operativa en su máquina local, donde se levantarán las 3 instancias de Docker con cada uno de los servicios de MongoDB con la redundancia de la Base de Datos llamada "Steam" que contendrá la colección de documentos llamada "Games". En la siguiente imagen se muestra la arquitectura requerida.

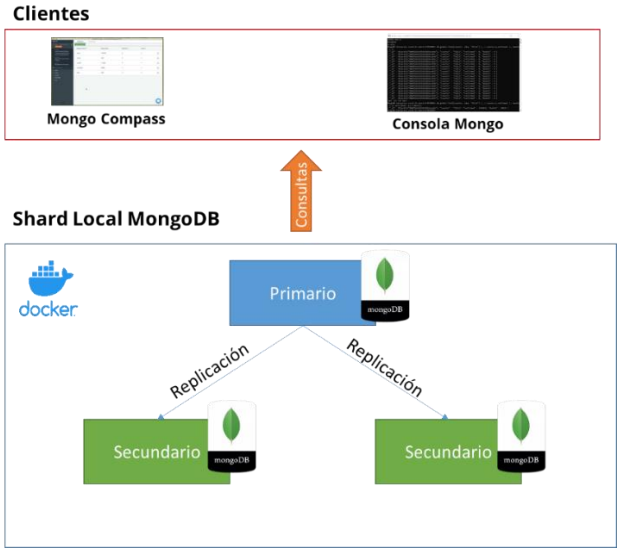

Figura 2: Arquitectura General

2. En base al dataset games.json entregado, implementar el código Python para las siguientes consultas requeridas para resolver las preguntas del negocio utilizando el cliente :
    a. Devolver los nombres y fechas de lanzamiento de juegos que se lanzaron en el año 2018. Usa una proyección para mostrar solo esos campos y limita los resultados a los primeros 20 juegos.
    b. Devolver los nombres y el número de contenidos descargables de juegos que tengan un precio entre $10 y $50 y al menos un contenido descargable. Ordena los resultados por el número de DLC de manera descendente y limita a los primeros 10.
    c. Devolver el nombre, la edad requerida y la calificación de los usuarios (si existe) para todos los juegos aptos para mayores de 12 años. Incluye una condición para que solo aparezcan aquellos juegos que tienen una calificación de usuarios mayor de 80. Limita los resultados a 15 juegos.

d. Identifica los 10 juegos más caros lanzados después del año 2015 para analizar las tendencias de precios en lanzamientos recientes.
e. Determina los 5 juegos con mejor calificación de usuarios entre los que son aptos para todas las edades (required_age = 0), para entender qué características valoran más los usuarios en estos juegos.
f. Analiza la relación entre el precio de los juegos y la calificación de los usuarios, clasificando los juegos en tres categorías de precio: bajo (hasta $10), medio (entre $11 y $30), y alto (más de $30). Muestra los 5 juegos con mejores calificaciones en cada categoría para determinar si el precio influye en la percepción de calidad del usuario.

3. Una vez realizados los puntos anteriores, realizando operaciones necesarias desde Python, se pide mostrar evidencia objetiva que permita demostrar:

a. La consistencia de los datos en los 3 nodos replicados, por ejemplo, posterior a la inserción, actualización y/o eliminación de datos, los datos deben permanecer siempre consistentes.
b. La alta disponibilidad, por ejemplo, bajando el nodo local para que responda alguno de los nodos replicados sin que el usuario se percate del problema.

5 Consideraciones técnicas

Para llevar a cabo esta tarea, se recomienda llevar a cabo las siguientes actividades técnicas:

1. Instalar el software de desarrollo y tecnología para contenedores Docker Desktop y Docker Compose. Usar como referencia este sitio https://docs.docker.com/manuals/.

2. Crear un Replica Set de MongoDB en Docker con docker-compose usando el archivo docker_compose.yml de ejemplo y subir los servicios.

3. Conectarse al Replica Set desde Python usando la biblioteca PyMongo, con la dirección de conexión de ejemplo mongodb://mongo1:30001,mongo2:30002,mongo3:30003/?replicaSet=my-replica-set&readPreference=primary&appname=MongoDB%20Compass&ssl=false. Asegúrese de modificar el archivo de hosts para que los nombres de los servidores sean mapeados a la IP local del host.

4. Descargar el dataset games.json y luego importar los datos con las instrucciones necesarias desde Python. Puede usar como referencia el siguiente Notebook PyMongo-DB.ipynb.

5. Documentación acerca de MongoDB, su arquitectura y su lenguaje de consultas puede ver en https://www.mongodb.com/docs/manual/.

6 Aclaraciones

1. Las consultas de esta tarea DEBEN realizarse en el foro de aula.
2. Es obligatorio utilizar las tecnologías y ambientes solicitados, existirá descuento en la pauta de no usarse.
3. Cualquier supuesto debe ser especificado de forma explícita en el notebook entregado.
4. Se recomienda iniciar con tiempo esta tarea.