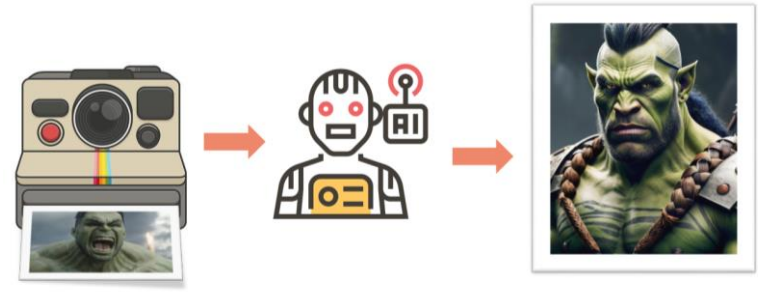
Bases de datos no relacionales

Bases de datos vectoriales

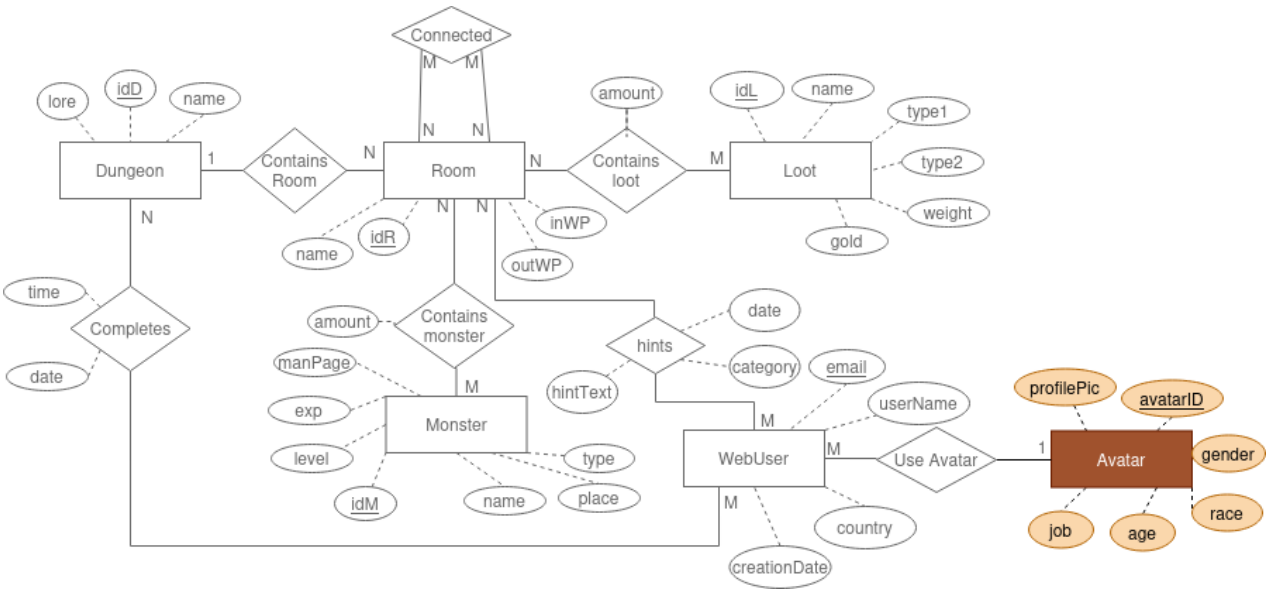


Introducción

En esta práctica crearemos un prototipo de un sistema de recomendación de avatares para el videojuego "The Jötun's Lair". Para ello vamos a crear una **base de datos vectorial** que permita buscar una imagen para el avatar del usuario usando una descripción textual o una foto. El equipo de arte ha recopilado un conjunto de retratos de avatares para usar en el prototipo.

Objetivo

Hasta el momento los avatares se almacenaban en una base de datos relacional (ver imagen inferior), para cada retrato se tenía almacenado su género, su raza, su edad y el trabajo que realiza. Usando estas anotaciones el sistema permite realizar algunos filtros para que a los usuarios les sea más sencillo elegir un avatar acorde a su personaje en el juego. Sin embargo, según ha crecido el número de avatares y el juego se ha extendido a diferentes países el sistema se ha quedado corto. Se pretende tener un sistema multilingüe que permita buscar avatares a partir de descripciones detalladas de los mismo. Además, se plantea la posibilidad de que el usuario pueda incluir una foto suya o un dibujo y que el sistema le muestre los avatares disponibles más similares. Por ello, se ha decidido realizar un pequeño prototipo usando Milvus.



Bases de datos relacional que da servicio al juego. La parte coloreada es la parte que usaremos en esta práctica.

Tarea

El equipo de IA proporcionará el código necesario para poder calcular los embeddings de imágenes y el texto. Usando dicho código, el equipo de base de datos debe:

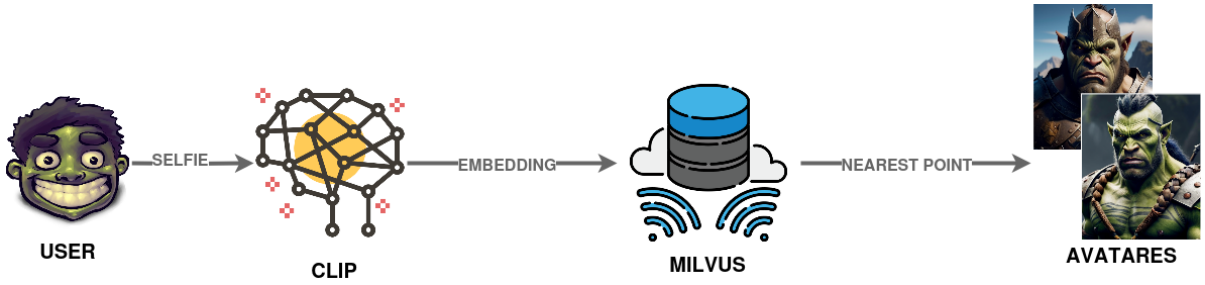
Cargar los embeddings en la base de datos vectorial

1. Calcular los embeddings de cada uno de los retratos proporcionados por el equipo de arte.
2. Configurar la conexión con la base de datos Milvus y el *SDK de Python*[1].
3. *Crear* una *base de datos*[2] y una *colección*[3] para el sistema de recomendación de avatares. En este prototipo no usaremos particiones. En este paso se debe decidir el esquema que tendrá la colección, es decir, los campos habrá en la base de datos vectorial.
4. *Introducir los datos*[4] de las imágenes de los avatares en la colección.
5. *Crea un índice*[5] que permita hacer consultas de manera eficiente.

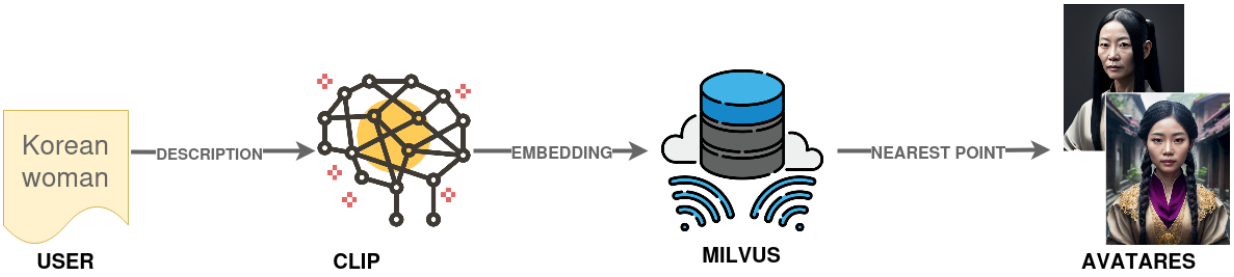
[1] https://milvus.io/docs/manage_connection.md
[2] https://milvus.io/docs/manage_databases.md
[3] https://milvus.io/docs/create_collection.md
[4] https://milvus.io/docs/insert_data.md
[5] https://milvus.io/docs/build_index.md

Programar los flujos de trabajo

Utiliza el *SDK de Python*[6] para implementar el código que realice los dos flujos de trabajo.



Flujo de trabajo 1: permitirá enviar una imagen al sistema y el sistema debe devolver las imágenes que más se le parezcan.



Flujo de trabajo 2: permitirá enviar una descripción textual al sistema y el sistema debe devolver las imágenes que más se ajusten a ella.

Probar el rendimiento

El tipo de índice y la métrica que se use para calcular las distancias tienen un impacto grande en el rendimiento del sistema, por ello es necesario probar distintas combinaciones. Crea otras colecciones que usen otros tipos de índices y otras métricas de distancias para encontrar la combinación que mejor funcione para el prototipo.

[6] https://milvus.io/docs/search.md