Bases de datos no relacionales

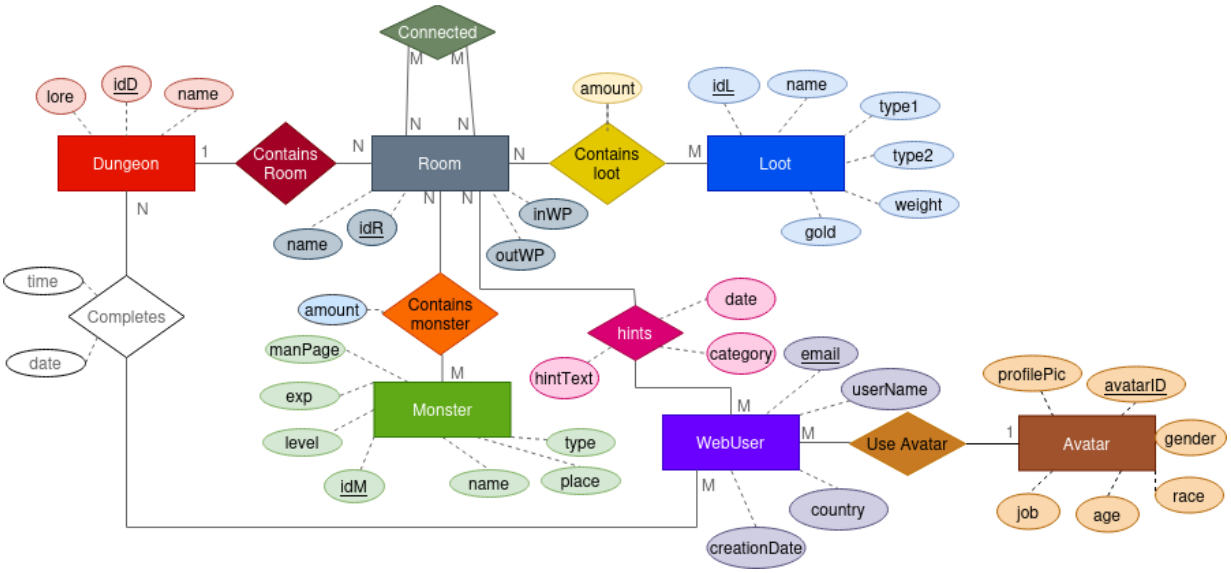
Práctica MongoDB

Introducción

Le empresa de "Norsewind Studios" tiene una wiki de su videojuego "Jotun's Lair" que contiene información y lore sobre los monstruos, el loot del juego y las mazmorras. Además, permite a los usuarios dejar comentarios sobre su experiencia con el videojuego. Los usuarios pueden dejar comentarios sobre las siguientes temáticas:

- **Lore**: Teorías e información relacionadas con la historia del juego.
- **Hint**: Trucos y consejos para otros jugadores.
- **Suggestion**: Sugerencias de los usuarios para mejorar algún aspecto del juego.
- **Bug**: Reportes de posibles fallos que ocurren durante el gameplay.

La empresa tiene una API REST para dar servicio a la wiki. Hasta ahora, se viene usando una base de datos relacional, pero últimamente, debido al alto volumen de jugadores, el sistema se está viendo sobre saturado. Por esa razón, se ha decidido migrar de una base de datos relacional a una base de datos de documentos para probar si así se solventan los problemas. A continuación, las entidades de la base de datos relacional que estarán involucradas en el prototipo



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

Objetivo

Se desea hacer un prototipo como prueba de concepto antes de decidirse a realizar la migración. Para el prototipo se usará MongoDB. La API REST actual tiene los siguientes endpoints:

- **GET /loot:** Devuelve los siguientes campos de todos los objetos del juego: *idL, name*.
- **GET /loot/{loot_id}:** Devuelve toda la información de un objeto del juego, incluidas las habitaciones donde aparece, de las habitaciones devuelve: *idR, room_name* y de la mazmorra a la que pertenece: *idM, dungeon_name*.
- **GET /monster:** Devuelve los siguientes campos de todos los monstruos del juego*: idM, name, level, type*.
- **GET /monster/{monster_id}:** Devuelve toda la información sobre un monstruo del juego, incluidas las habitaciones donde aparecen, de las habitaciones devuelve: *idR, room_name* y de la mazmorra a la que pertenece: *idM, dungeon_name*.
- **GET /dungeon:** Devuelve los siguientes campos de todas las mazmorras del juego: *idD, name.*
- **GET /dungeon/{dungeon_id}:** Devuelve información sobre una mazmorra del juego. Debe devolver: *idM, name* y *lore*. Además, este endpoint se utiliza para alimentar un grafo interactivo por lo que requiere la siguiente información: 1) el nombre e id de cada habitación de la mazmorra; 2) las conexiones entre habitaciones de la mazmorra; 3) el id y el nombre de los monstruos que aparecen en cada habitación; 4) el id y el nombre de los tesoros que aparecen en cada habitación; 5) El número de comentarios de cada categoría que hay en cada habitación.
- **GET /room/{room_id}:** Devuelve la siguiente información de una habitación*: idR, name, inWP* y *outWP*. Además, incluye la siguiente información de todos los monstruos incluidos en la habitación (*idM, name, type, level, place, exp, manPage*) y de los tesoros *(idL, name, type1, type2, weight, gold*). Por último, incluye todos los comentarios que se hayan realizado para esa habitación, cada comentario debe incluir: *email* y *userName, country, creationDate* del usuario que lo realizo y *texto*, *fecha de publicación* y *categoría* del comentario.

- **GET /user:** Devuelve los campos *email*, *username* y *country* de todos los usuarios.
- **GET /user/{email}:** Devuelve todos los campos de un usuario. Además, incluye todos los comentarios que ha realizado. De cada comentario incluye, el texto, la fecha de creación, la categoría, el id (Room.IdR) y nombre (Room.name) de la habitación a la que hace referencia el comentario, el id(Dungeon.IdD) y nombre(Dungeon.name) de la mazmorra donde está la habitación.
- **POST /comment:** Añade un nuevo comentario. Necesita de autenticación de usuario.
    - Parametros: user_email (str), room_id (int), text (str), category (str).
- **POST /monster:** Añade un nuevo monstruo al juego. Necesita de autenticación de admin.
    - Parámetros en cabecera: *name (str), type (str), level (int), place (str), exp (float), manPage (int).*
- **POST /loot:** Añade un nuevo objeto al juego. Necesita de autenticación de admin.
    - Parámetros en cabecera: *name (str), type1 (str), type2 (str), weight (str), gold (float).*
- **POST /room:** Añade una nueva habitación a una mazmorra. Necesita de autenticación de admin.
    - Parámetros de cabecera: dungeon_id (int), dungeon_name(str), dungeon_lore(str), room_*name* (str), *rooms_connected* (list[int]), [*inWP*(str)], [*outWP*(str)].
- **PUT /room/{room_id}/monster:** Sobrescribe los monstruos de una sala de una mazmorra, los monstruo debe de existir.
    - Parámetros cabecera: *monsters*(list[int])
- **PUT /room/{room_id}/loot:** Sobrescribe los tesoros de una sala de una mazmorra, los tesoros deben de existir.
    - Parámetros cabecera: *loot(list[int])*
- **PUT /room/{room_id}/connections:** Sobrescribe las conexiones de una habitación.
    - Parámetros de cabecera: *connections* (list[int])
- **DELETE /room/{room_id}/:** Borra una habitación y todos los comentarios asociados a ella.
- **DELETE /monster/{monster_id}/:** Borra un monstruo.
- **DELETE /loot/{loot_id}/:** Borra un tesoro.

Además, se está planteando extender esta API para dar servicio a otras aplicaciones internas de la compañía. Los siguientes equipos piden cambios:

- **Quality asurance:** les interesa que se les notifique automáticamente cuando un usuario deje un comentario de tipo Bug/Suggestion.
- **PR:** Los jugadores llevan tiempo pidiendo un tablón donde aparezcan todos los comentarios relativos al lore del juego separado por idiomas.
- **Marketing**: está interesado en conocer información de los jugadores como el promedio de veces que postean al mes, la longevidad de los usuarios activos, los países donde el juego se juega más ... etc.

Para llevar a cabo el prototipo se ha preparado una base de datos MongoDB con las siguientes colecciones.

Rooms

```
room_id: int,
room_name: str,
dungeon_id: int,
dungeon name: str,
?in_waypoint: str,
?out_waypoint: str,
rooms_connected: [{
  room_id: int,
  room_name: str
}]
hints: [{
  creation_date: str,
  text: str,
  category: str,
  publish_by: {
    email: str,
    user_name: str,
    creation_date: str,
    country: str
  }
},...],
monsters: [{
  id: int,
  name: str,
  place: str,
  type: str,
  man_page: str,
  level:int,
  exp: float
},...],
Loot: [{
  id: int,
  name: str,
  type1: str,
  type2: str,
  weight: str,
  gold: int
},...]
```

Users

```
email: str,
user_name: str,
creation_date: str,
country: str,
hints: [{
  creation_date: str,
  text: str,
  category: str,
  references_room: {
    room_id: int,
    room_Name: str,
    dungeon_id: int
    dungeon_name: str,
  }
},...]
```

Monsters

```
id: int,
name: str,
place: str,
type: str,
man_page: str,
level:int,
exp: float,
in_rooms: [
  {
    amount: int,
    room_id: int,
    room_name: str,
    dungeon_id: int
    dungeon_name: str,
},...]
```

Loot

```
id: int,
name: str,
type1: str,
type2: str,
weight: str,
gold: int,
in_rooms: [{
  room_id: int,
  room_name: str,
  dungeon_id: int
  dungeon_name: str,
},...]
```

Tareas

Realizar las siguientes consultas desde Python o *MongoDB Compass*[1], según se especifique.

Python

Implementa una función Python que realice una consulta a mongoDB para cada uno de los endpoints de la API REST. Los campos del tipo ***{X}*** son parámetros de la función. Además, en algunos métodos se especifican ***los parámetros de cabecera*** que también son parámetros adicionales de la función. Por

[1] https://www.mongodb.com/products/tools/compass

último, si algún parámetro aparece entre corchetes ([ ]), entonces indica que es opcional. Las funciones deben devolver texto en JSON.

MongoDB Compass

1. Para dar servicio a los jugadores y el equipo de quality asurance se ha decidido crear una colección nueva en la base de datos que contenga todos los comentarios. Para ello sigue estos pasos:

A. Haz una consulta que obtengan los datos necesarios para la colección y exporta el resultado a un fichero .json. Debajo puedes encontrar la estructura de la colección.
B. A continuación, crea una nueva colección llamada Hints y usa el fichero .json para poblarla. Además, elimina el campo *hints* de las colecciones *Rooms* y *User*.
C. Por último, actualiza las funciones de los endpoints: **POST /comment, GET /dungeon/{dungeon_id}** , **GET /room/{room_id}** y **GET /user/{email}.** ¿Como se ven afectados estos endpoints?

Hints

```
Creation_date: str,
HintText: str,
Category: str,
References_room: {
  IdR: int,
  Name: str,
  IdD: int
  Dungeon: str,
}
Publish_by: {
  Email: str,
  User_name: str,
  CreationDate: str,
  Country: str
}
```

2. Realiza las siguientes consultas para el equipo de marketing

A. El número de cuentas de usuario que se crearon cada año agrupadas por país. Por ejemplo:

```
Year: 2018
Countries: [{
  UK: 3000,
  Germany: 2500
  Spain: 1129,
  …
}]
```

```
Year: 2019
Countries: [{
  UK: 2000,
  Germany: 3500
  Spain: 1689,
  …
}]
```

B. Los 20 países cuyos usuarios han realizado el mayor número de posts de tipo Lore en los últimos 5 años. Los países deben aparecen ordenados de mayor a menor número de posts.

```
country: EE.UU,
lore_posts: 150000
```

```
country: Japan,
lore_posts: 130000
```

C. Los 5 usuarios que más bugs han reportado en 2022. Deben aparecer ordenados de mayor a menor.

```
user_name:
spiderJerusalen
bugs_reported: 896
```

```
user_name: judgeDread
bugs_reported: 595
```

D. La mazmorra que más sugerencias ha recibido desglosada en países.

```
country: EE.UU,
dungeon_name: Arvalecliffe, Laboratory of the
Unsightly Ninjas
```

```
country: Spain
dungeon_name: Cobwold, Dungeon of the Jealous
Thieves
```