INF-239 Bases de Datos
Tarea 1: Oracle

Profesores: Marcelo Mendoza - Margarita Bugueño
Ayudante de Cátedra: María Paz Morales Llopis
Ayudantes de Tareas: Jorge Ludueña Giacaman - Axel Reyes Orellana

27 de mayo de 2020

1. Emergencia de salud

El Equipo Rocket creó una nueva cepa de PokeRus la cual posee cualidades dañinas para los Pokemon, creando así una PokePandemia. Ante esto, la compañía POYO's Inc. decide crear una instalación llamada Sansanito Pokemon. Para evitar el sobre cupo de esta, requiere de un sistema que mantenga los datos de los pacientes y sus estados en una base de datos. Es por esto que POYO's Inc. le entrega a usted la lista de Pokemon existentes, facilitando así los datos necesarios para el desarrollo de la plataforma.

2. Descripción del problema

La base de datos que le entrega POYO contiene las siguientes características importantes:

- Tabla POYO
  - Número en la PokeDex (int)
  - Nombre (str)
  - Type 1 (Tipo) (str)
  - Type 2 (Tipo) (str)
  - HP Total (int)
  - Legendary (bool)

Se le pide a usted que cree una tabla que posea la siguientes características

- Tabla SANSANITO POKEMON
  - Id (int)
  - Número en la PokeDex (int)
  - Nombre (str)
  - Type 1 (Tipo) (str)
  - Type 2 (Tipo) (str)
  - HP Actual (int)

- HP Max (int)
- Legendary (bool)
- Estado (str)
- Fecha y hora de ingreso (Datetime)
- Prioridad (int)

3. Datos sobre las tablas

- Ambas tablas deben ser implementadas en una base de datos de Oracle.
- La prioridad de un Pokemon equivale a su HP Max - HP actual.
- Los estados de un Pokemon son: Envenenado, Paralizado, Quemado, Dormido y Congelado. Un estado (no null) le añade 10 de prioridad a un Pokemon.
- La capacidad máxima es de 50 Pokemon.
- Un legendario utiliza 5 de capacidad.
- No puede haber más de 1 legendario del mismo nombre.

4. To do List

- Se solicita que realice una aplicación standalone (no es necesaria una interfaz gráfica, la terminal es suficiente).
- Debe cargar el archivo de "pokemon.csv" en la base de datos con los criterios de la tabla POYO.
- Se debe crear la tabla SansanitoPokemon descrita anteriormente. Debe añadirle registros aleatorios que sigan las reglas y que sean consistentes con los datos de la tabla POYO.
- Se debe dar soporte para realizar acciones CRUD (Create, Read, Update y Delete)
- Al ingresar un nuevo Pokemon, si el Sansanito Pokemon está lleno y llega un Pokemon con mayor prioridad, se cambia por el que tiene menor prioridad.
- Su aplicación debe ser capaz de realizar las siguientes consultas sobre la tabla SANSANITO POKEMON:
  - Ingresar un Pokemon.
  - Los 10 Pokemon con mayor prioridad.
  - Los 10 Pokemon con menor prioridad.
  - Todos los Pokemon con un estado en específico.
  - Todos los Pokemon legendarios.
  - El Pokemon que lleva más tiempo ingresado.
  - Nombre del Pokemon más repetido.
  - Nombre, HP actual, HP Max y prioridad de todos los Pokemon, ordenados por prioridad.

5. Reglas y Restricciones

- **Toda interacción con la base de datos debe ser mediante Querys en Python.**
- Toda información que se requiera al añadir un Pokemon debe ser obtenida de la tabla POYO, no del CSV.
- Debe utilizar Oracle versión 11G, 12C, 18C o el Express edition (XE).
- Debe implementarse tanto una *vista* como un *trigger*, queda a su criterio en qué. **Debe ser útil**.
- El código debe ser realizado en Python utilizando una conexión **odbc** con la librería pyodbc.

6. Q&A adelantado

- Sí, pueden crear todas las funciones que quieran.
- Sí, el csv solo se debe utilizar una vez (para crear la tabla POYO).
- No, no se utilizarán inputs erróneos, la idea es evaluar su trabajo en bases de datos, no como programa.
- Olvide cualquier otro conocimiento sobre Pokemon no especificado en esta tarea.
- Sí, puede cambiarle el nombre a las tablas. Manténgalo dentro del contexto en el cual se está trabajando.
- En caso de empate en cualquier situación, la elección es arbitraria.
- Por si se lo pregunta, POYO viene del siguiente video AQUÍ

7. Sobre Entrega

- Las funciones deberán ir comentadas, explicando clara y brevemente lo que realiza, los parámetros que recibe y los returns si es que lo hace.
- Se debe trabajar de forma individual.
- **La entrega debe realizarse en .zip y debe llevar el nombre: Tarea1BD_Rol.zip**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la utilización de su programa.**
- La entrega será vía AULA y el plazo máximo de entrega es hasta el **22 de Junio 23:55 hrs**.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- Si su código no corre no se revisará.
- Consultas vía la plataforma oficial AULA y serán revisadas por ambos ayudantes. Evite utilizar otros medios, recibirá una foto de este apartado como respuesta.
- La tarea debe ser defendida mediante la plataforma Discord, el link del servidor es el siguiente AQUÍ.

- En la sección de Notas tareas podrá ver el ayudante que se le fue asignado.
- La defensa se realizará durante la semana del 22 de junio, los horarios disponibles se los dará su ayudante asignado.

8. Calificación

- Implementar correctamente las dos tablas (10 puntos)
- Correcta implementación del CRUD
  - Create (10 puntos)
  - Read (5 puntos)
  - Update (15 puntos)
  - Delete (5 puntos)
- Consultas (35 puntos (5 puntos por consulta))
- Correcta utilización de un *trigger* (10 puntos)
- Correcta utilización de una *vista* (10 puntos)
- **Descuentos:**
  - No utiliza querys (-100 puntos)
  - Código no corre (-100 puntos)
  - Falta de comentarios (-30 puntos MAX)
  - No respetar las reglas (-5 puntos por regla)
  - Por cada hora de atraso se le descontará 5 puntos.

En caso de existir nota negativa esta será reemplazada por un 0.