ITESM Campus Santa Fe

Nombre del bloque:

TC2005B: Construcción de software y toma de decisiones (Gpo. 401)

Nombre del entregable:

3. Ejercicio de Modelación de Base de Datos Pokemon TCG

Alumno:

Diego Córdova Rodríguez, A01781166

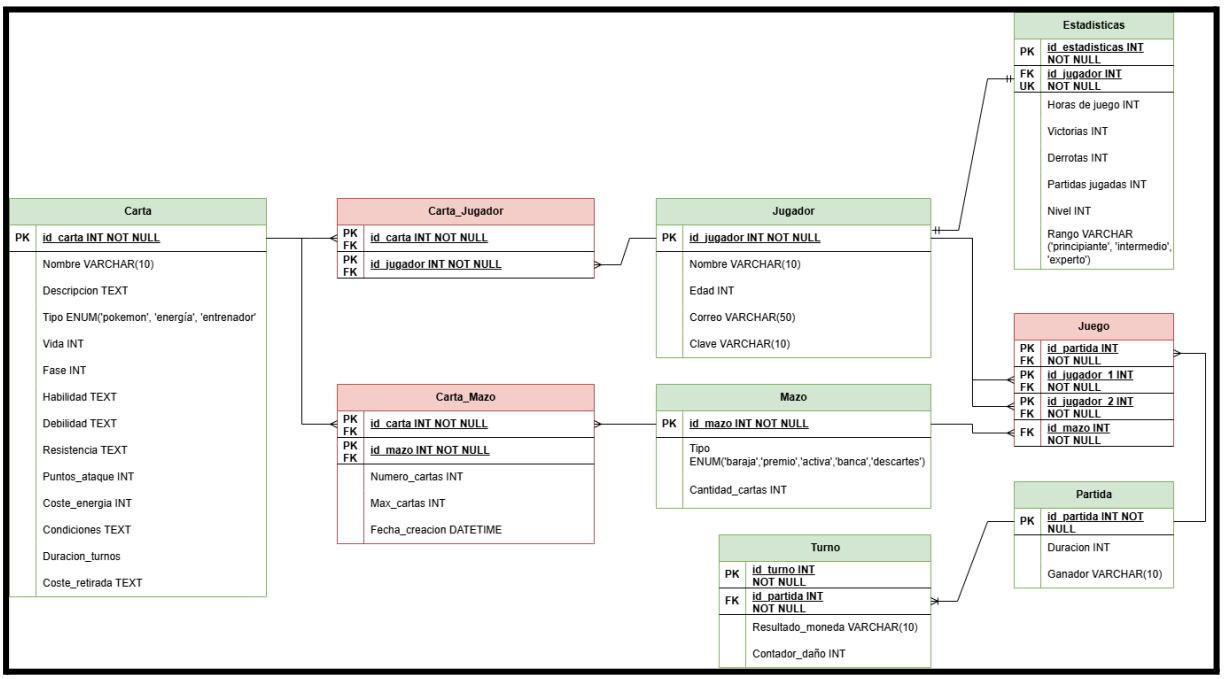
Profesor:

Esteban Castillo Juarez

Fecha de entrega:

5 de marzo de 2025

Diagrama Entidad-Relación: Base de Datos Pokemon TCG



Acceso al archivo .drawio con el Diagrama:

https://drive.google.com/file/d/16NUBp_NpAVY1EGhH7KR9xyGq1jTJ_gpG/view?usp=sharing

Justificación de Base de Datos

El modelado de la base de datos del videojuego Pokemon TCG cuenta con 6 principales entidades y 3 tablas intermedias utilizadas para evitar la duplicidad de información y falta de elementos de integridad, conectando unas entidades con otras. A continuación se especifica la función de cada una de las entidades establecidas:

Justificación de Entidades

Jugador:

Esta tabla contiene los jugadores que participarán en las partidas de Pokemon TCG:

- Cada jugador puede tener una única lista de estadísticas (relación 1 a 1) dentro de la base de datos del juego, pues solo existe una instancia de estas y se actualiza conforme el jugador juega partidas.
- Cada jugador puede tener muchas cartas en su inventario, así como una carta puede pertenecer a muchos jugadores, por lo que existe una tabla intermedia Carta_Jugador, donde únicamente se hace la relación entre estos.
- Cada jugador puede participar en una única partida del juego a la vez, por lo que existe una tabla intermedia entre Jugador y Partida llamada Juego, donde se encuentra la relación entre la partida y sus 2 jugadores.

Estadísticas:

Esta tabla almacena el conjunto de estadísticas que pertenecen a los jugadores del juego:

- Un conjunto determinado de estadísticas de jugador pueden ser poseídas por uno solo (relación 1 a 1). Esta relación 1 a 1 evita que un jugador tenga múltiples registros de estadísticas dentro de la base de datos, evitando así la duplicidad de información y asegurando un fácil acceso a las estadísticas de cada usuario.
- Esta tabla cuenta con una restricción de llave única (UK) para el jugador, garantizando que cada jugador tenga un único conjunto de estadísticas almacenados en la tabla de datos.

Carta:

Esta tabla representa, de forma general, todas las cartas que pueden ser utilizadas en el juego, ya sean de tipo ‘pokemon’, ‘entrenador’ o ‘energía’. De esta forma, la tabla contiene todos los atributos que pueden llegar a contener estas cartas.

- Debido a que cada carta puede contener o no diferentes atributos, por defecto todos son nulos (NULL) hasta que son asignados. Por ejemplo, si una carta es de tipo ‘entrenador’, todos sus atributos serán NULL excepto por su tipo, nombre y descripción.
- Esta tabla se relaciona con una tabla intermedia Carta_Jugador, ya que muchas cartas pueden ser poseídas por muchos jugadores (relación muchos a muchos).
- Una carta puede tener 3 diferentes tipos: pokemon, energia y entrenador.
- Esta tabla se conecta con la tabla intermedia Mazo, definiendo en qué tipo de mazo se encuentran incluidas en una partida, ya sea una baraja, las cartas de la banca, cartas de descarte, entre otros.
- Una carta solo puede pertenecer a un mazo una vez por partida, ya sea en la baraja, en una carta de premio, activa, en la banca o en los descartes de los jugadores (relación 1 a 1).

Partida:

Esta tabla representa el núcleo del juego, una partida donde 2 jugadores se reúnen para competir con sus mazos de cartas. El propósito de esta tabla es registrar cada enfrentamiento entre los jugadores, la duración de la partida y el ganador de la misma.

- Cada partida se encuentra relacionada con una tabla intermedia Juego, donde se especifican los jugadores que participan en la partida (id_jugador_1 e id_jugador_2). Esto asegura que exista una relación clara entre quién juega contra quién y qué mazo se utiliza en la partida.
- Además, cada partida se relaciona a la tabla Turno, pues cada partida se conforma de una serie de turnos, donde los jugadores pueden tomar decisiones y utilizar su mazo de cartas (relación 1 a muchos).

- Cada partida se conforma de diferentes turnos, sin embargo, esta tabla no contiene una llave foránea id_turno, pues se tendrían que crear nuevos registros de información para especificar los turnos que fueron llevados a cabo, duplicando información de las partidas. Para evitar esto, los turnos son los que cuentan con la llave foránea id_partida, pues un turno solo puede pertenecer a una única partida.

Turno:

Esta tabla representa cada uno de los turnos que en conjunto conforman las partidas del juego. El propósito de esta tabla es registrar lo que sucede en cada turno para hacer un seguimiento del flujo del juego de cada partida.

- Un turno puede pertenecer a una única partida (id_partida) (relación 1 a 1), por lo que se puede hacer un seguimiento de todas las jugadas en cada partida del juego, estructurando estas de forma ordenada.
- Entre los datos que se registran, se encuentra el resultado de la moneda que lanza el jugador, una de las mecánicas del juego Pokemon TCG. De igual forma se registra el daño realizado por el jugador en el turno, llevando un control de los efectos de las cartas.
- Al estar relacionada con la tabla Partida, que a su vez se relaciona con la tabla Juego, se puede conocer el jugador que participa en el turno, razón por la que esta tabla no cuenta con una llave foránea de Jugador. De esta forma, para conocer el jugador que participa en el turno, se puede realizar una unión de tablas entre Turno, Partida y Juego para saber quién realiza las acciones. De esta forma se evita redundancia y duplicidad de datos.

Mazo:

Esta tabla establece la relación entre cada jugador y los diferentes mazos de cartas que un jugador puede tener durante una partida, ya sea una baraja, cartas premio, cartas activas, cartas en la banca y cartas descartadas. Su propósito es estructurar cada mazo para que sea utilizado dentro del juego en las partidas.

- Esta tabla se relaciona con la tabla intermedia Juego, donde se establece el mazo que utiliza cada jugador en la partida. De esta forma, se puede saber qué mazo pertenece a cada jugador con una unión entre las tablas Juego, Partida y Mazo. Debido a esto, no es necesario que Mazo tenga una relación directa con un jugador, teniendo una estructura más ordenada donde un mismo mazo puede ser usado por distintos jugadores en las partidas.
- La tabla intermedia Carta_Mazo define cuáles son las cartas que conforman el mazo, permitiendo que tenga diferentes cartas y que cada una pueda estar en diferentes mazos de diferentes jugadores.

Justificación de tablas intermedias

Carta_Jugador:

Esta tabla relaciona las cartas poseídas por los jugadores. De esta forma, cada jugador posee de 1 a muchas cartas, así como cada carta puede ser poseída de 1 a muchos jugadores en distintas instancias del juego. Así, se gestiona la relación muchos a muchos entre Carta y Jugador.

- Al relacionar Jugador y Cartas, se define que cada carta puede estar en posesión de muchos jugadores a lo largo del tiempo, así como también un jugador puede poseer muchas cartas.
- La tabla cuenta con la id de los jugadores y cartas como llaves primarias y foráneas (clave primaria compuesta), evitando crear una nueva llave primaria para definir la relación Carta_Jugador. Esto asegura que una carta solo pueda pertenecer una vez por un jugador en la tabla, evitando duplicidad.

Carta_Mazo:

Esta tabla existe ya que define los mazos a los que pertenecen las cartas, pues cada mazo se compone de un conjunto de cartas. El propósito de esta tabla es modelar la relación muchos a muchos entre Carta y Mazo, permitiendo que los jugadores organicen sus cartas en diferentes mazos a lo largo de las partidas.

- Una misma carta puede pertenecer a diferentes tipos de mazos en partidas diferentes (relación 1 a muchos).
- La tabla contiene una clave foránea id_carta para indicar qué carta se encuentra en el mazo.
- La llave foránea id_mazo asocia cada carta con un mazo en específico.
- id_carta e id_mazo conforman una llave primaria compuesta, asegurando que cada carta sólo pueda aparecer una vez en un mismo mazo dentro de la tabla, evitando duplicidad de información.

Juego:

Esta tabla existe con el propósito de relacionar cada partida con los 2 jugadores que participan en ella, así como el mazo específico con el que juegan.

- Al estar relacionada con una partida, se tiene una llave foránea id_partida de la tabla Partida, indicando en donde participan los jugadores y el mazo de cartas.
- La tabla contiene id_jugador_1 e id_jugador_2, llaves foráneas que identifican a los 2 jugadores que compiten en la partida con llave id_partida.
- Cada Juego contiene un id_mazo, clave foránea que referencia la tabla Mazo e indica el mazo que se está utilizando en la partida, registrando el conjunto de cartas que utiliza cada jugador.
- Se cuenta con una llave primaria compuesta conformada por id_partida, id_jugador_1, id_jugador_2 e id_mazo, asegurando que esta combinación de jugadores y mazo sea única dentro de una misma partida. A pesar de esto, cada jugador puede usar diferentes mazos en diferentes partidas, lo que no genera un conflicto con la llave primaria compuesta, pues id_partida cambia en cada juego.

Restricciones del modelado de base de datos

En el estado actual del modelado de la base de datos de Pokémon TCG, se representa la interacción de los jugadores en las partidas, así como los mazos de cartas que son utilizados en el juego. Sin embargo, se cuenta con diversas restricciones de integridad a nivel de base de datos para garantizar la integridad de datos del juego.

Cada tabla contiene al menos una clave primaria única para evitar datos duplicados. Mientras que algunas tablas contienen llaves primarias únicas, como la tabla Jugador (id_jugador) o Partida (id_partida), algunas otras contienen llaves primarias compuestas, principalmente aquellas que son intermedias, evitando la duplicidad de información, como es Juego (id_carta, id_ jugador_1, id_jugador_2, id_mazo) o Carta_Jugador (id_carta, id_jugador).

Las claves primarias y foráneas no pueden ser NULL en la base de datos, por lo que se debe declarar la instrucción NOT NULL. Ej: Para la id del jugador, se debe usar la instrucción id_jugador INT NOT NULL PRIMARY KEY. De la misma manera, se debe usar la instrucción UNISGNED AUTO_INCREMENT para que cada id en los registros se incremente de forma automática, asegurando que sean valores positivos únicos.

Debido a que las cartas en la tabla Carta son generales, es decir, pueden ser de diferente tipo, como ‘pokemon’, ‘energía’ o ‘entrenador’, los atributos no aplican para todas las cartas. De esta forma, todos los atributos deben ser inicializados con el valor NULL por defecto. Ej: El atributo Vida, Fase y Coste_retirada sólo aplican para las cartas tipo ‘pokemon’, por lo que los demás tipos de carta tendrán en estos atributos un valor NULL.

Para la tabla Carta, atributos como Vida, Puntos_ataque, Coste_energia o Fase no pueden ser valores numéricos negativos, por lo que se debe agregar una restricción para asegurar que sean mayores o iguales a 0.

Para asegurar que los datos en las relaciones sean válidos, se deben usar las restricciones ON DELETE CASCADE y ON UPDATE CASCADE, asegurando que si se elimina/actualiza un registro de una tabla, se eliminen/actualicen de manera automática el resto de registros relacionados en las tablas dependientes. De esta forma, se evita que existan

datos “huérfanos” o no conectados a ninguna tabla en la base de datos. Cabe mencionar que se debe tener cuidado al utilizar ON DELETE CASCADE, ya que si se utiliza en un jugador, también se eliminarán sus estadísticas, cartas y partida en la que participa.

Para asegurar que no existan duplicados en ciertos atributos específicos, se tienen las siguientes restricciones:

- El atributo Nombre en Jugador debe ser único, evitando que puedan existir usuarios duplicados en la base de datos.
- El atributo Ganador en Partida debe ser un sólo jugador de los que participan en Juego, ya sea id_jugador_1 o id_jugador_2, ya que en Pokemon TCG no pueden haber empates.
- El atributo Nombre en Carta debe ser único, ya que no pueden existir cartas con el mismo nombre. Ej: Si existe la carta Pikachu, no puede existir otra con el mismo nombre, pues ya existe.

Se utilizan restricciones en los tipos de datos usando VARCHAR(num), limitando el número de carácteres de nombres de usuario, resultados de la moneda en cada turno (cara o cruz), entre otros. Además, se utilizan los tipos de datos ENUM para aceptar determinados tipos de valores para los datos, como es en las cartas, que únicamente pueden ser de tipo ‘pokemon’, ‘energia’ y ‘entrenador’. Esta misma restricción ocurre para los tipos de mazos de carta existentes ('baraja','premio','activa','banca','descartes').

Finalmente, se utiliza la restricción de UK (Unique Key o Llave Única) para las estadísticas de los jugadores, siendo que cada jugador puede tener un único registro de conjunto de estadísticas dentro de la base de datos.