PROYECTO BASE DE DATOS

Slay The Spire

Contenido

Introducción.

Slay the Spire es un videojuego de cartas por turnos PVE (Player vs Enviroment) donde el objetivo es ir atravesando un mapa donde podemos encontrar eventos diversos, como batallas contra monstruos, tiendas donde comprar objetos o cartas o incluso eventos especiales.

Al empezar cada partida podremos elegir un héroe, el cual posee unas características, como son una reliquia única de inicio y unas cartas de inicio. Conforme va pasando la partida, el jugador puede ir adquiriendo más cartas y reliquias para hacerse más poderoso y poder luchar contra los monstruos jefes. Conforme más fuerte sea el monstruo, mejor será la recompensa.

Respecto a las cartas que vamos adquiriendo, estas se dividen en varios tipos, ya sean ataques, habilidades, estados o incluso maldiciones, las cuales nos ayudarán o entorpecerán la batalla contra un monstruo. Estas cartas poseen un coste de energía para usarlas y algunas de ellas poseen un coste indefinido o X, ya que cuanta más energía usemos, más poderosas serán.
Debido a que la comunidad del juego es mayoritariamente angloparlante, la base de datos estará desarrollada en inglés.

Entidades e Interrelaciones.

La base de datos incluirá las siguientes entidades y correspondientes atributos:

- **Heroe:**
  - Atributos: nameHeroe, HP, descriptionHeroe.
    - Cada héroe tendrá un nombre que lo identificará.
    - Existirá un héroe que englobará a todos, llamado Everyone.
    - Todos los héroes tienen HP, excepto Everyone que los engloba a todos.
    - Un héroe puede tener varias cartas y reliquias.

Atributo	Tipo de dato	Rango	Clave	Obligatorio
nameHeroe	Carácter	10 variable	Si	Si
HP	Entero	Hasta 5 dígitos	-	No
descriptionHeroe	Carácter	100 variable	-	Si

- **Card:**
  - Atributos: nameCard, rarityCard, typeCard, cost, descriptionCard.
    - Cada carta tendrá un nombre que la identifique, por lo que no habrá dos cartas iguales.
    - Una carta la podrán usar todos los héroes o uno.
    - Hay cartas que no poseen ni coste ni rareza.

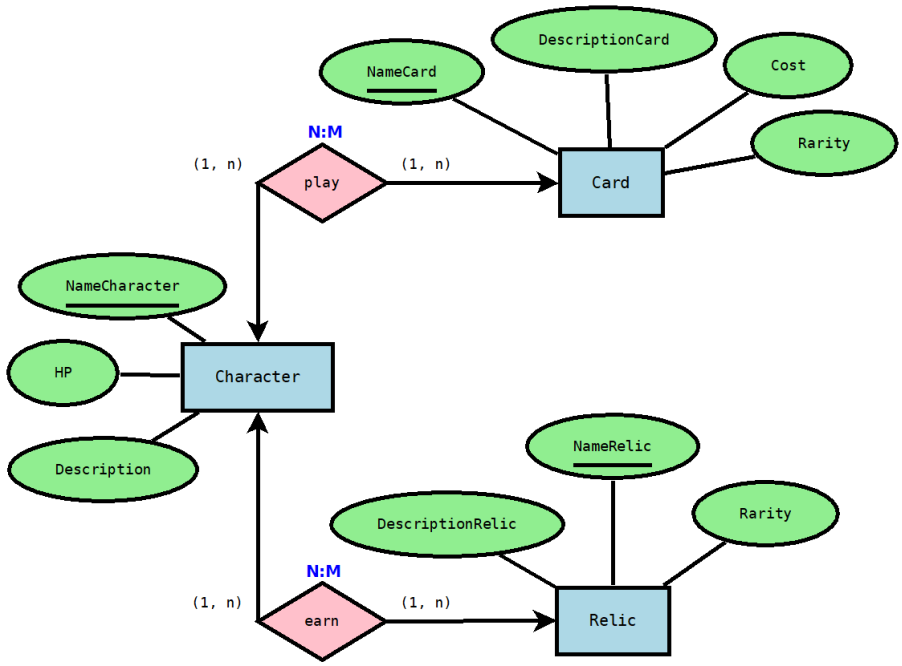
Atributo	Tipo de dato	Rango	Clave	Obligatorio
nameCard	Carácter	30 variable	Si	Si
rarityCard	Carácter	10 variable	-	No
typeCard	Carácter	10 variable	-	Si
HP	Entero	Hasta 10 dígitos	-	No
descriptionCard	Carácter	150 variable	-	Si

- **Relic:**
  - Atributos: nameRelic, rarityRelic, descriptionRelic.
    - Las reliquias son únicas, por lo que tendrán un nombre que las diferencie.
    - Una reliquia la puede obtener uno o todos los héroes.

Atributo	Tipo de dato	Rango	Clave	Obligatorio
nameRelic	Carácter	30 variable	Si	Si
rarityRelic	Carácter	10 variable	-	Si
descriptionRelic	Carácter	150 variable	-	Si

Diagrama entidad-relación.

El diagrama entidad-relación de la base de datos quedaría del siguiente modo (representado con el programa DIA):



Creación de tablas.

El código de la creación de las tablas empieza creando la base de datos que se llamará “slaythespire” y creando las tablas en el orden necesario para que se establezcan adecuadamente las relaciones entre ellas.

```
/* Fernando Dominguez Garcia */
/* Base de datos Slay the Spire */
/* Creacion de la base de datos */
drop database if exists slaythespire;
create database slaythespire;
use slaythespire;

/* Creacion de las tablas */
drop table if exists play;
drop table if exists earn;
drop table if exists relic;
drop table if exists heroe;
drop table if exists card;


create table card(
    nameCard varchar(30) not null,
    rarityCard varchar(10),
    typeCard varchar(10) not null,
    cost smallint(10),
    descriptionCard varchar(150) not null,

    primary key(nameCard)

)ENGINE=InnoDB;

create table heroe(
    nameHeroe varchar(10) not null,
    hp smallint(5),
    descriptionHeroe varchar(100) not null,

    primary key(nameHeroe)
)ENGINE=InnoDB;

create table relic(
    nameRelic varchar(30) not null,
    rarityRelic varchar(10) not null,
    DescriptionRelic varchar(150)not null,

    primary key(nameRelic)
)ENGINE=InnoDB;

create table earn(
    earn_nameRelic varchar(30) not null,
    earn_nameHeroe varchar(10) not null,
```

```
    primary key(earn_nameRelic, earn_nameHeroe),

    constraint fk_earn_Relic
        foreign key (earn_nameRelic)
        references Relic (nameRelic)
    on delete cascade on update cascade,

    constraint fk_earn_Heroe
        foreign key (earn_nameHeroe)
        references Heroe (nameHeroe)
    on delete cascade on update cascade

)ENGINE=InnoDB;

create table play(
    play_nameCard varchar(30) not null,
    play_nameHeroe varchar(10) not null,

    primary key(play_nameCard, play_nameHeroe),

    constraint fk_play_Card
        foreign key (play_nameCard)
        references Card (nameCard)
    on delete cascade on update cascade,

    constraint fk_play_Heroe
        foreign key (play_nameHeroe)
        references Heroe (nameHeroe)
    on delete cascade on update cascade

)ENGINE=InnoDB;
```

Insertar datos.

Una vez las tablas están creadas, insertaremos los datos en el mismo orden en el que las creamos y una por una, de manera que si salta algún fallo será más fácil de encontrar. Tras haber insertado los datos adecuadamente, solo queda actualizar los datos de los valores nulos.

Vistas.

Existen dos vistas:

- **v_relicHeroe** nos mostrará las reliquias que corresponden a cada uno de los héroes, sin contar las que pueden usar todos, para que podamos observar cuales podemos usar con cada uno de los héroes.
- **v_starter** mostrará las cartas iniciales de los personajes, para que tengamos una idea de cuales serán las cartas más básicas.

Procedimientos.

Los procedimientos son:

- **p_cardsHeroe** que mostrará el numero de cartas que existen del héroe que indiquemos.
- **p_typecard** nos dirá el numero de cartas que existen del tipo que indiquemos, para cada clase y en total.
- **p_relic** nos mostrará el nombre de las reliquias y el héroe al que les pertenece de la rareza que indiquemos.
- **p_delHeroe** es el procedimiento que elimina al héroe junto a las reliquias y las cartas que le pertenecían.
- **p_updaterelicheroe** es el procedimiento que actualiza la vista **v_relicHeroe.**

Funciones.

Antes de empezar, debemos recordar que tenemos que establecer el permiso para poder crear funciones (ya incluido en el código).

Existen las siguientes funciones:

- **f_Curse** indica el número de maldiciones que existen.
- **f_Status** indica el número de estados que existen.
- **f_costX** indica el número de cartas cuyo coste es indeterminado, dependiendo de la energía que vayamos a gastar.
- **f_relicShop** cuenta el número de reliquias que son exclusivas en la tienda.
- **f_probEveryoneRelic** muestra la probabilidad de que cuando salga una reliquia sea apta para todos.

Disparadores.

Los disparadores serán la parte del código que regulará la base de datos.

- **t_heroe** es el disparador que resetea la tabla héroe si todos los valores son borrados (que no es lo mismo que si se vacía la tabla).
- **t_updaterelic y t_updateheroe** hacen una llamada al procedimiento **p_updaterelicheroe** cuando se actualiza una reliquia o un héroe.
- **t_earn y t_play** actualizan las respectivas tablas cuando una reliquia o una carta es añadida para establecer que cualquier héroe pueda optar a ellas.