Fernando Domínguez García

Curso2017-2018 1ºASIR - GBD

Proyecto BDD

Hearthstone: Heroes of Warcraft

Índice

Introducción.

Hearthstone es un videojuego de cartas coleccionables online, creado por la empresa Blizzard Entertainment. Este videojuego se basa en el universo imaginario de Warcraft, otro de los videojuegos de esta empresa, y es totalmente gratuito, aunque dentro del juego se pueden hacer compras. El juego salió para PC y se ha expandido a los dispositivos móviles, y, hoy en día, es uno de los videojuegos más conocidos del mundo, teniendo una comunidad de más de 30 millones de personas.

El juego se basa en partidas de 1vs1 por turnos en las que cada jugador elegirá un **héroe** (o comúnmente llamado clase) con el que jugar y un mazo de cartas con el que jugar ese héroe. De manera predeterminada los héroes tienen **30 puntos** de vida, y si uno de estos pierde todos sus puntos de vida, pierde la partida. Existen varios héroes que pertenecen a la misma clase, solo cambian el nombre y la estética. Cada **clase** poseen una habilidad o **poder** para la partida, la cual podrán usar una vez por turno, por lo que no todas las clases son iguales.

Cada vez que a un jugador le toque su turno poseerá una energía, llamada maná, que le permitirá jugar unas cartas u otras, dependiendo del coste de estas. Esta energía se recuperará en cada turno y aumentará hasta un máximo de 10 cristales de maná.

Los mazos que jugará cada uno serán de **30 cartas**, pudiendo mezclar los distintos tipos de cartas a tu gusto. Los tipos de carta son: **hechizo, esbirro, arma y héroe jugable**. Los hechizos son cartas que realizan una acción, los esbirros son compañeros que te acompañarán en la batalla, los cuales tienen sus propios puntos de vida, sus puntos de ataque que pueden realizar y sus habilidades correspondientes si las tienen, las armas son cartas que hacen que tu héroe pueda realizar daño, pudiendo recibir el daño también a la hora de atacar, y recientemente han sacado otro tipo de carta que se llama héroe jugable, que sustituye a tu héroe actual para ganar habilidades.

En la imagen de aquí podemos observar una carta, la cual posee un **coste** de 5 cristales de maná (situado en la parte superior izquierda), un **daño** que puede realizar de 6 puntos (parte inferior izquierda), también posee 2 puntos de **vida** (parte inferior derecha), un **nombre** (situado en la parte central), y una **descripción**, si la hay, situada justo debajo del nombre.

Además, las cartas poseen un tipo de **rareza**, que indicará la probabilidad de adquirir una. Esta clasificación se hace del siguiente modo (en orden de más rara a menos): Legendaria, Épica, Rara y Común. Dependiendo de la rareza que tengan, también las puedes crear o reciclar, adquiriendo unas cartas a partir de reciclar otras.

Estas cartas no las pueden jugar todos los héroes, algunas son exclusivas de estos, por lo que hay que hacer una distinción de las cartas que pueden jugar todos los héroes y las que no.

El juego hoy en día posee varias **expansiones**, que te ofrecen cartas nuevas para coleccionar y después usar para jugar contra tus adversarios.

Utilidad.

Hay una cantidad inmensa de cartas y cuesta mucho aprenderse lo que hacen todas ellas, como se llaman o alguna característica de estas, por lo que es muy importante tener una base de datos que refleje estos datos, a la hora de crearse un mazo, que resulte más fácil. De este modo, podremos clasificar las cartas dependiendo de qué clase queramos jugar, de qué expansión o incluso de que rareza, para saber cuál elegir.

Existen páginas web donde los usuarios pueden ver todas las cartas que existen, como **Hearthpwn**, y de aquí será de donde sacaré todos los datos sobre las cartas..

La base de datos cambia cada varios meses, cuando sacan otra expansión, que dará posibilidad a nuevas cartas e incluso a nuevos héroes. De este modo, hay que saber cómo actualizarla y modificarla para cuando esto ocurra.

La mayor parte de la comunidad es angloparlante, por este motivo la base de datos está en inglés.

He elegido hacer la base de datos sobre este videojuego porque lo suelo ver muy a menudo, me parece muy entretenido y suele plantearte retos una vez tienes buenos conocimientos.

Ejemplo de uso.

Un jugador, al que llamaremos Jesús, quiere hacerse un mazo que le dijo un compañero suyo, pero no se acuerda bien de las cartas, solamente se acuerda de algunas cosas. Se acuerda de que su compañero le dijo que usaba una carta que era de la clase de cazador, que su coste era de 9 cristales de maná y que era de una rareza legendaria, por lo que se dispone a hacer una búsqueda de esto en la base de datos.

La carta que estaba buscando era King Krush y gracias al uso de la base de datos pudo dar con la carta que estaba buscando.

Entidades e Interrelaciones.

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

- **Entidad HEROE.**
  - Atributos: codClass, name.
    - Cada héroe tendrá un nombre de clase que lo identificará.
    - Existirá un héroe que englobará a todos, llamado *Everyone*.
    - Todos los atributos son obligatorios.
    - Los héroes usan muchas cartas diferentes.

Atributo	Tipo de dato	Rango	Clave	Obligatorio
CODHEROE	Carácter	20 variable	Sí	Sí
NAME	Carácter	60 variable	-	Sí

- **Entidad HEROE POWER.**
  - Atributos: codHeroePw, descriptionHeroePower.
    - Cada clase tendrá un poder de héroe y solo uno.
    - Al igual que en la tabla **HEROE**, existe un valor que le corresponde a *Everyone.*
    - Todos los atributos son obligatorios.

Atributo	Tipo de dato	Rango	Clave	Obligatorio
CODHEROEPW	Carácter	20 variable	Sí	Sí
DESCRIPTIONHEROEPOWER	Carácter	60 variable	-	Sí

- **Entidad CARTA.**
  - Atributos: codCarta, nameCard, rarity, type, cost damage, health, descriptionCard.
    - Las cartas se identificarán con un número exclusivo, por lo que no se podrá repetir.
    - Una carta la podrán usar todos los héroes o solamente uno.
    - Todas las cartas pertenecen a una expansión
    - No todas las cartas tienen una mecánica y, si la tienen, a veces tiene varias.

Atributo	Tipo de dato	Rango	Clave	Obligatorio
CODCARD	Número entero sin signo	Sin asignar	Sí	Sí
NAMECARD	Carácter	50 variable	-	Sí
RARITY	Carácter	20 variable	-	Sí
TYPE	Carácter	30 variable	-	Sí
COST	Número entero sin signo	De 0 a 50	-	Sí
DAMAGE	Número entero sin signo	De 0 a 50	-	Sí
HEALTH	Número entero sin signo	De 1 a 50	-	Sí
DESCRIPTIONCARD	Carácter	150 variable	-	No

- **Entidad EXPANSION.**
  - Atributos: codExpansion, nameExpansion.
    - La expansión se identificará con un código único que lo diferenciará de cada una.
    - Todas las expansiones tienen cartas.

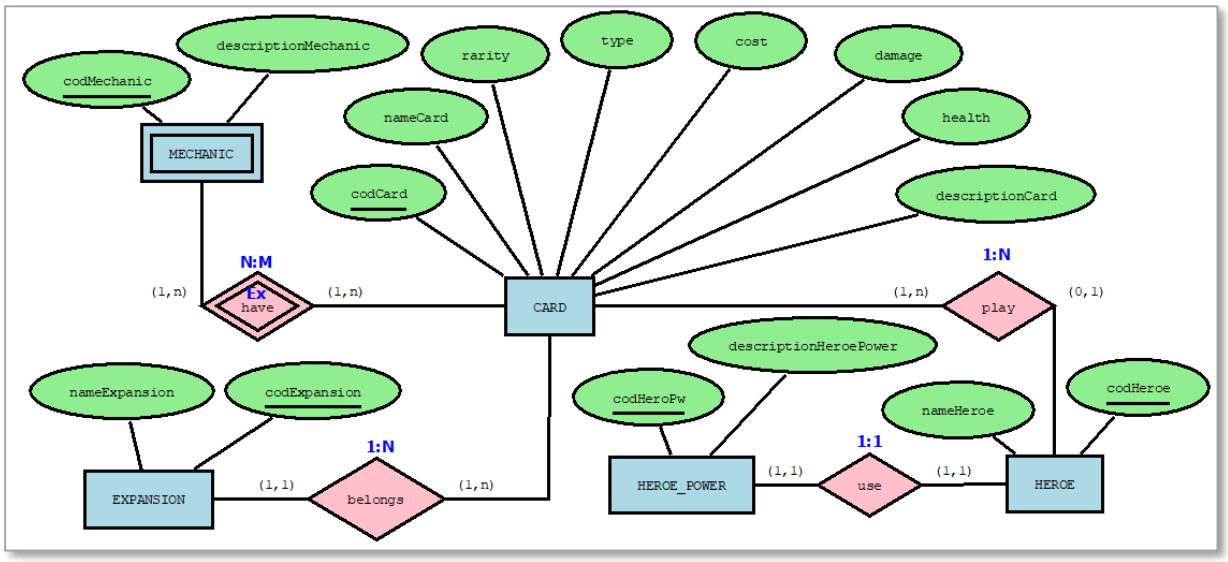
Atributo	Tipo de dato	Rango	Clave	Obligatorio
CODEXPANSION	Número entero sin signo	De 1 a 10	Sí	Sí
NAMEEXPANSION	Carácter	70 variable	-	Sí

- **Entidad MECANICA.**
  - Atributos: codMecanica, descriptionMechanic.
    - Las mecánicas se diferencian por su código de nombre.
    - Las mecánicas son asociadas a 1 o varias cartas.

Atributo	Tipo de dato	Rango	Clave	Obligatorio
CODMECHANIC	Número entero sin signo	De 1 a 30	Sí	Sí
DESCRIPTIONMECHANIC	Carácter	70 variable	-	No

Diagrama entidad-relación.

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



Esquema relacional.

Tras haber realizado el diagrama entidad-relación hay que transformarlo a un esquema relacional que nos permita la creación de las tablas de una manera adecuada.

Para empezar, se pueden observar 4 relaciones, entre ellas **MECHANIC**, que es una entidad débil.

La relación de **HEROEPOWER** y **HEROE**, tendremos que propagar la clave de **HEROE** a la otra tabla, donde será clave foránea, donde habrá dos claves primarias. Usaremos la regla 2.1, ya que la relación se da de forma completa.

El modelo relacional inicial sería el siguiente:

R1 = **CARD** (codCard, nameCard, rarity, type, cost, damage, health, descriptionCard)

R2 = **HEROE** (codHeroe, nameHeroe)

R3 = **HEROE_POWER** (codHeroePw, descriptionHeroePower, heroepower_codHeroe (FK2))

R4 = **EXPANSION** (codExpansion, nameExpansion)

R5 = **MECHANIC** (codMechanic, descriptionMechanic)

El siguiente paso a aplicar es desglosar las relaciones que existan de N:M y 1:N, que en este caso se dan en todas las relaciones.

- Para la relación entre **CARD** y **DESCRIPTION** crearemos una 3ª tabla a partir de las claves primarias de ambas entidades, junto a los atributos de la relación, y la clave principal serán ambas claves principales, esta tabla la llamaremos **HAVE**. Ambas claves serán claves foraneas.
- Para **EXPANSION** y **CARD** se da una relación de 1:N en la que aplicamos la regla 3.1 para relaciones completas en la que debemos de propagar la clave de **EXPANSION** hacia **CARD**, donde será clave foránea.
- En el caso de **CARD** y **HEROE**, una relación 1:N parcial, usaremos la regla 3.2, en la que crearemos una tercera tabla con las claves de ambas tablas y estas serán claves primarias. Llamaremos a esta tabla **HAVE**.

De este modo, el modelo relacional quedaría así:

R1 = **EXPANSION** (codExpansion, nameExpansion)

R2 = **HEROE** (codHeroe, nameHeroe)

R3 = **HEROE_POWER** (codHeroePw, descriptionHeroePower, heroepower_codHeroe (FK2))

R4 = **CARTA** (codCard, nameCard, rarity, type, cost, damage, health, descriptionCard, card_codExpansion (FK1))

R5 = **MECHANIC** (codMechanic, descriptionMechanic)

R5 = **PLAY** (play_codHeroe (FK3), play_codCard (FK2))

R6 = **HAVE** (have_codCard (FK3), have_codMechanic (FK4))

Creación las tablas.

Para la creación de base de datos necesitaremos un editor de texto, con el que escribiremos el script para la creación de las tablas, sus entidades y relaciones. En mi caso he usado editor de texto diferente al que solemos usar en clase, que se llama "Visual Studio Code".

Visual Studio Code es un editor de código fuente de código abierto para todas las plataformas, diseñado por Microsoft.

El código para crear la base de datos y añadir las tablas es el siguiente:

```
/* Fernando Domínguez García */
/* Proyecto de base de datos de Hearthstone en MySQL */
/* Crea la base de datos */
drop database if exists hearthstone;
create database hearthstone;
use hearthstone;

/* Borrado de tablas si existieran */
drop table if exists have;
drop table if exists play;
drop table if exists mechanic;
drop table if exists deck;
drop table if exists card;
drop table if exists heroepower;
drop table if exists heroe;
drop table if exists expansion;


/* Creación de tablas */

create table expansion(
    codExpansion int auto_increment,
    nameExpansion varchar (70) not null,
    primary key (codExpansion)
)ENGINE=InnoDB;


create table heroe(
    codHeroe varchar (20),
    nameHeroe varchar (60) not null,
    primary key (codHeroe, nameHeroe)
)ENGINE=InnoDB;
```

```
create table heroepower(
    codHeroePw varchar (20),
    heroepower_codHeroe varchar (20),
    descriptionHeroePower varchar (60) not null,
    primary key (codHeroePw, heroepower_codHeroe),

    constraint fk_power_heroe
        foreign key (heroepower_codHeroe)
        references heroe (codHeroe)
    on delete cascade on update cascade
)ENGINE=InnoDB;

create table card(
    codCard int auto_increment not null,
    nameCard varchar (50) not null,
    rarity varchar (20) not null,
    type varchar (30) not null,
    cost smallint(50) not null,
    damage smallint(50) not null,
    health smallint(50) not null,
    descriptionCard varchar (150) default NULL,
    card_codExpansion int not null,
    primary key (codCard),

    constraint fk_card_expansion
        foreign key (card_codExpansion)
        references expansion (codExpansion)
    on delete cascade on update cascade
)ENGINE=InnoDB;


create table mechanic(
    codMechanic varchar (30),
    descriptionMechanic varchar (70),
    primary key (codMechanic)
)ENGINE=InnoDB;
```

```
create table play(
    play_codHeroe varchar (20),
    play_codCard int,
    primary key (play_codHeroe, play_codCard),

    constraint fk_play_heroe
        foreign key (play_codHeroe)
        references heroe (codHeroe)
    on delete cascade on update cascade,

    constraint fk_play_card
        foreign key (play_codCard)
        references card (codCard)
    on delete cascade on update cascade
)ENGINE=InnoDB;


create table have(
    have_codCard int,
    have_codMechanic varchar (30),
    primary key (have_codCard, have_codMechanic),

    constraint fk_have_card
        foreign key (have_codCard)
        references card (codCard)
    on delete cascade on update cascade,

    constraint fk_have_mechanic
        foreign key (have_codMechanic)
        references mechanic (codMechanic)
    on delete cascade on update cascade
)ENGINE=InnoDB;
```

Tabla segundaria DECK

Existe una tabla llamada deck la cual debe crearse para que algunos procedimientos funcionen. Esta tabla la usaremos para crear los mazos que veamos convenientes.

```
/* Creado de la tabla "deck" para el procedimiento */
create table deck(
    deck_codCard int not null,
    nameCard varchar(50) not null,
    Heroe varchar(20) not null,
    Rarity varchar(20) not null,

    constraint fk_deck_card
        foreign key (deck_codCard)
        references card (codCard)
    on delete cascade on update cascade
    )ENGINE=InnoDB;
```

Nota: todos los documentos se encuentran adjuntos junto al proyecto.

Actualización para los valores NULL.

Para los valores que deben ser nulos (NULL) los estableceremos gracias al comando "*Update*", con el que actualizaremos la tabla en los valores donde no hay ningún valor, estableciéndolos como NULL.

La estructura de este comando es la siguiente:

```
update [nombre de la tabla]
        set [nombre de la columna] = NULL
where [condición];
```

Consultas simples.

En este apartado expondré unos cuantos ejemplos de consultas que se pueden realizar en la base de datos:

- **Obtener el número de Cartas de Rareza Legendaria que son Esbirros.**

```
select
   rarity as Rareza,
   count(*) as NumCartas
from card where
   rarity="Legendary"
   and type="minion";
```

Rareza	NumCartas
Legendary	95

- **Se quiere saber el Nombre y el Daño que tienen las Cartas que poseen más de 10 de Vida sin ser Héroes Jugables. Ordenar por orden alfabético de las Cartas.**

```
select
   nameCard as Carta,
   health as Vida,
   damage as Daño
from card where
   health > 10
   and type=("Playable Heroe")
order by nameCard;
```

Carta	Vida	Daño
Deathwing	12	12
Lord Jaraxxus	15	3
Malygos	12	4
Sleepy Dragon	12	4
Tar Lord	11	1
The Darkness	20	20
Tyrantus	12	12
Ultrasaur	14	7
Witchwood Grizzly	12	3
Wyrmguard	11	3
Ysera	12	4

- **Contar el número de Cartas para cada Mecánica y ordenar por la columna Mecanica.**

```
select
   have_codMechanic as Mecanica,
   count(*) as NumCartas,
from have
group by have_codMechanic
order by have_codMechanic;
```

Mecanica	NumCartas
Adapt	14
Battlecry	258
Charge	20
Choose One	14
Combo	15
Deathrattle	88
Discover	17
Divine Shield	26
Echo	13
Enrage	6
Freeze	17
Immune	6
Lifesteal	23
Overload	16
Poisonous	18
Quest	9
Recruit	17
Rush	19
Secret	38
Silence	6
Spell Damage	13
Stealth	15
Taunt	100
Windfury	11

- **Listar las cartas cuya Descripción sea NULL.**

```
select
  nameCard as NomCartas,
  descriptionCard as Descripcion
from card
where descriptionCard is NULL;
```

NomCartas	Descripcion
Ultrasaur	NULL
Dire Mole	NULL
Snowflipper Penguin	NULL
Wisp	NULL
Arcanite Reaper	NULL
Assassin's Blade	NULL
Fiery War Axe	NULL
Light's Justice	NULL
Core Hound	NULL
War Golem	NULL
Boulderfist Ogre	NULL
Chillwind Yeti	NULL
Oasis Snapjaw	NULL
Magma Rager	NULL
Bloodfen Raptor	NULL
River Crocolisk	NULL
Murloc Raider	NULL

- **La suma de la Vida de todos aquellos esbirros que posean las letras “drag” en su Nombre.**

```
select sum(health) as VidaTotal
from card
where upper(nameCard) like "%DRAG%" and type="Minion";
```

VidaTotal
50

nombreCarta	vida
Dragoncaller Alanna	3
Sindragosa	8
Dragonhatcher	4
Ebon Dragonsmith	4
Sleepy Dragon	12
Hoarding Dragon	6
Dragonslayer	3
Faerie Dragon	2
Swamp Dragon Egg	3
Young Dragonhawk	1
Dragonling Mechanic	4

Consultas complejas.

Las consultas complejas son aquellas que implican el uso de varias tablas y en este apartado veremos algunos ejemplos:

- **Listar el nombre de los Héroes junto al nombre de su Poder de Héroe.**

```
select nameHeroe as NombreHeroe,
  codHeroePw as NombrePoderHeroe
  from heroepower
inner join heroe on
  heroepower_codHeroe = codHeroe;
```

NombreHeroe	NombrePoderHeroe
Malfurion Stormrage	Shapestshift
All heroes	Whatever
Rexxar	Steady Shot
Jaina Proudmoore	Fireblast
Uther Lightbringer	Reinforce
Anduin Wrynn	Lesser Heal
Valeera Sanguinar	Dagger Mastery
Thrall	Totemic Call
Gul'dan	Life Tap
Garrosh Hellscream	Armor Up!

- **Listar el Coste, Rareza y el Nombre de las Cartas que pertenecen a la clase Mago**

```
select play_codHeroe as Heroe,
  nameCard as NombreCarta,
  cost as coste,
  rarity as rareza
  from play
inner join card on
  play_codCard = codCard
where play_codHeroe="Mage"
order by cost;
```

Heroe	NombreCarta	coste	rareza
Mage	Steam Surger	4	Rare
Mage	Molten Reflection	4	Rare
Mage	Water Elemental	4	Free
Mage	Ghastly Conjurer	4	Rare
Mage	Arcane Keysmith	4	Epic
Mage	Polymorph	4	Free
Mage	Leyline Manipulator	4	Rare
Mage	Cone of Cold	4	Common
Mage	Ethereal Arcanist	4	Rare
Mage	Deck of Wonders	5	Epic
Mage	Bonfire Elemental	5	Rare
Mage	Dragon's Fury	5	Epic
Mage	Curio Collector	5	Rare
Mage	Aluneth	6	Legendary
Mage	Toki Time-Tinker	6	Legendary
Mage	Meteor	6	Epic
Mage	Blizzard	6	Rare
Mage	Archmage Antonidas	7	Legendary
Mage	Flamestrike	7	Free
Mage	Glacial Mysteries	8	Epic
Mage	Sindragosa	8	Legendary
Mage	Frost Lich Jaina	9	Legendary
Mage	Dragoncaller Alanna	9	Legendary
Mage	Pyroblast	10	Epic

63 rows in set (0.00 sec)

(No se muestran todos los resultados en la imagen debido a que son muchos)

- **Listar las Cartas que pertenecen a la Expansión "Kobolds and Catacombs" que tienen alguna Mecánica y el nombre de estas cartas. Ordenar por Nombre de las Cartas.**

```
select nameCard as NombreCarta,
  nameExpansion as NombreExpansion,
  have_codMechanic as Mecanica
  from card
inner join have on
   have_codCard = codCard
inner join expansion on
  card_codExpansion = codExpansion
where codExpansion = "4"
order by nameCard;
```

codExpansion	nameExpansion
1	Clasic
2	Journey to Ungoro
3	Knights of the Frozen Throne
4	Kobolds and Catacombs
5	The Witchwood

NombreCarta	NombreExpansion	Mecanica
Seeping Oozeling	Kobolds and Catacombs	Deathrattle
Sewer Crawler	Kobolds and Catacombs	Battlecry
Shroom Brewer	Kobolds and Catacombs	Battlecry
Silver Vanguard	Kobolds and Catacombs	Deathrattle
Silver Vanguard	Kobolds and Catacombs	Recruit
Sleepy Dragon	Kobolds and Catacombs	Taunt
Sneaky Devil	Kobolds and Catacombs	Stealth
Spiteful Summoner	Kobolds and Catacombs	Battlecry
Stoneskin Basilisk	Kobolds and Catacombs	Divine Shield
Stoneskin Basilisk	Kobolds and Catacombs	Poisonous
Sudden Betrayal	Kobolds and Catacombs	Secret
Temporus	Kobolds and Catacombs	Battlecry
The Darkness	Kobolds and Catacombs	Battlecry
The Runespear	Kobolds and Catacombs	Discover
Trogg Gloomeater	Kobolds and Catacombs	Poisonous
Trogg Gloomeater	Kobolds and Catacombs	Taunt
Twig of the World Tree	Kobolds and Catacombs	Deathrattle
Twilight Acolyte	Kobolds and Catacombs	Battlecry
Twilight's Call	Kobolds and Catacombs	Deathrattle
Val'anyr	Kobolds and Catacombs	Deathrattle
Violet Wurm	Kobolds and Catacombs	Deathrattle
Void Ripper	Kobolds and Catacombs	Battlecry
Voidlord	Kobolds and Catacombs	Deathrattle
Voidlord	Kobolds and Catacombs	Taunt
Vulgar Homunculus	Kobolds and Catacombs	Battlecry
Vulgar Homunculus	Kobolds and Catacombs	Taunt
Wandering Monster	Kobolds and Catacombs	Secret
Wax Elemental	Kobolds and Catacombs	Divine Shield
Wax Elemental	Kobolds and Catacombs	Taunt
Windshear Stormcaller	Kobolds and Catacombs	Battlecry
Woecleaver	Kobolds and Catacombs	Recruit
Zola the Gorgon	Kobolds and Catacombs	Battlecry

117 rows in set (0.00 sec)

Para que resulte más comodo, he realizado una busqueda de todas las expansiones que hay y así ver que codExpansion corresponde a "Kobolds and Catacombs"

(No se muestran todos los resultados en la imagen debido a que son muchos).

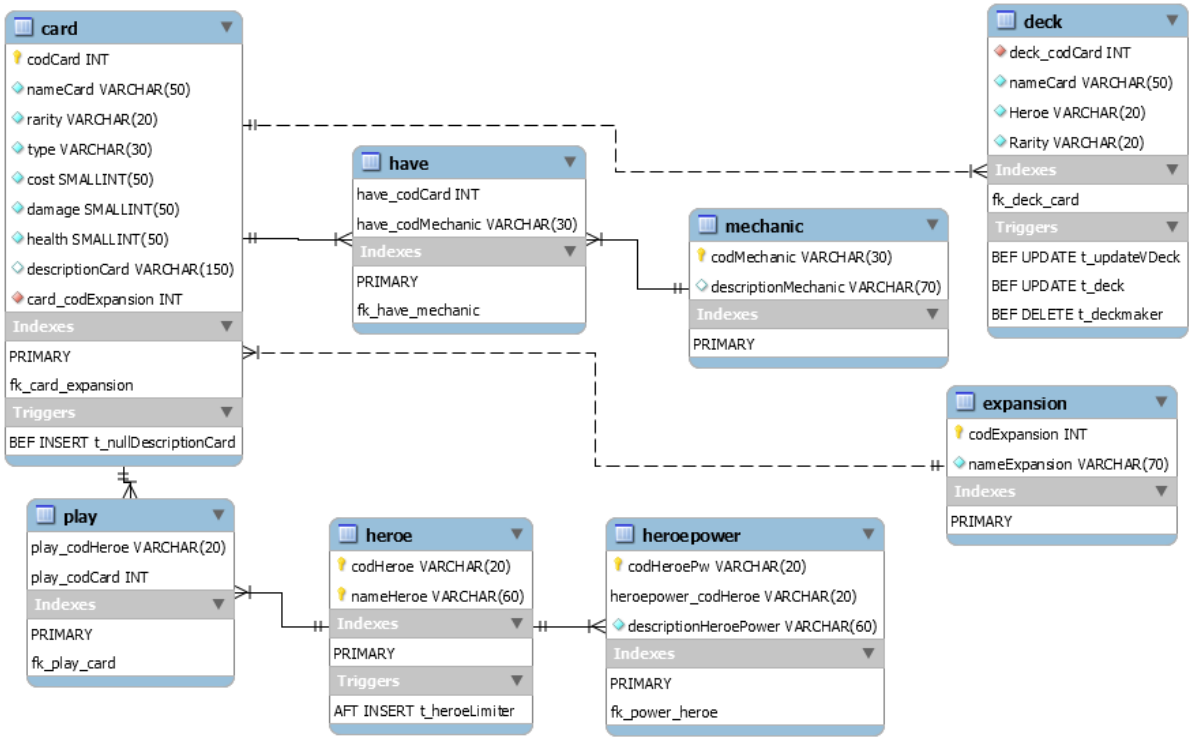
- **Queremos saber lo que hace la Carta "Darius Crowley", por lo que para ello deberemos saber las Mecánicas que posee y la Descripción de lo que hacen estas.**

NombreCarta	Descripcion	Mecanica	DescripcionMecanica
Darius Crowley	Rush. After this attacks and kills a minion gain +2/+2.	Rush	Can attack minions immediately

```
select nameCard as NombreCarta,
  descriptionCard as Descripcion,
  codMechanic as Mecanica,
  descriptionMechanic as DescripcionMecanica
  from card
inner join have on
    have_codCard = codCar d
inner join mechanic on
  have_codMechanic = codMechanic
where nameCard="Darius Crowley";
```

Diagrama adicional.

La base de datos posee una estructura que nos permitirá ver de un vistazo como se relacionan las tablas. Esta estructura se puede representar fácilmente en forma de diagrama con una de las multiples funciones del programa "MySQL Workbench", que revertirá el proceso de creación de la base de datos a partir del código para expresar esta base de datos junto con sus relaciones. El diagrama quedará del siguiente modo:



Actualización y borrado de datos.

La base de datos puede sufrir actualizaciones para algunos valores de la tabla o suponer un aumento de los datos de la tabla, por lo que siempre se debe tener forma de arreglar estos problemas. La solución es usar el comando **UPDATE** del mismo modo que hemos hecho anteriormente para actualizar las celdas de las **descripciones** de las **cartas** que deben tener **NULL.**

La sintaxis del comando es la misma que la ya vista, solo que cambiando **NULL** por el valor deseado:

```
update [nombre de la tabla]
        set [nombre de la columna] = [Valor actualizado]
where [condición];
```

Si en algún caso se da que debemos insertar una fila de datos usaremos el comando **INSERT** con el que añadiremos valores a las distintas columnas de una misma fila.
A la hora de actualizar la tabla, hay que tener en cuenta que sigue un orden de actualización, al igual que cuando se creó la tabla con sus datos.
Del mismo modo, para eliminar una fila, lo haremos con el comando **DELETE**, y como las tablas están relacionadas entre sí, se eliminarán los valores de todas las tablas correspondientes.

```
delete from [nombre de la tabla]
where [condición];
```

Ejemplo práctico.

Se da el caso en el que debemos actualizar la base de datos debido a un cambio que han realizado en el ultimo parche del juego.
De este modo, Blizzard Entertainment nos dice en las notas del parche 10.2 que:

- La **Carta** "**Bonemare**" ahora costará 7 manas en vez de 8.
- La **Carta** "**Corridor Creeper**" ahora tendrá 5 de daño en vez de 2.
- La **Carta** "**Kingsbane**" será eliminada del juego.
- La **Carta** "**Millhouse Manastorm**" será eliminada.
- La próxima **Expansión** será "**Naxxramas**", por lo que debemos agregarla a nuestra base de datos.

Para realizar los cambios tendremos que hacer lo siguiente:

```
update card
set cost = 8
where nameCard = "Bonemare";

update card
set damage = 5
where nameCard = "Corridor Creeper";

delete from card
where nameCard = "Kingsbane";

delete from card
where nameCard = "Millhouse Manastorm";

insert into expansion
values ("6", "Naxxramas");
```

Vistas, procedimientos, funciones y disparadores.

La base de datos posee al final un apartado donde podemos encontrar el código de:

Vistas.

Existen dos vistas:

1. **V_deck** es una vista que refleja el resultado de la tabla deck agrupando las cartas repetidas y mostrándote cuantas hay.

```
Create view v_deck as select nameCard, Heroe, Rarity, count(*) as 'Number'
from deck group by nameCard;
```

2. **V_cardexpansion** Es una vista la cual relaciona cada carta con su expansión correspondiente.

```
Create view v_cardexpansion as select distinct nameCard, nameExpansion from
card inner join expansion where card_codExpansion = codExpansion;
```

Procedimientos.

Existen 6 procedimientos los cuales deben tener precedido en el código un delimiter:

1. **P_insertaCard** es un procedimiento el cual ayuda a la inserción de cartas. Cuando lo ejecutes te pedirá una serie de parámetros de la carta y tras ello te la guardará en la tabla card, estableciéndole automáticamente el código de la carta, junto a la relación con el héroe.

```
/* Procedimiento para insertar cartas */
drop procedure if exists p_insertaCard $$
19éroe19 procedure p_insertaCard(
    in par_nameCard varchar(50),
    in par_rarity varchar(20),
    in par_type varchar(30),
    in par_cost smallint(50),
    in par_damage smallint(50),
    in par_health smallint(50),
    in par_descriptionCard varchar(150),
    in par_card_codExpansion int(11),
    in par_play_codHeroe varchar(20))
begin
    insert into card values (NULL, par_nameCard, par_rarity, par_type,
par_cost, par_damage, par_health, par_descriptionCard, par_card_codExpansion);
    insert into play values (par_play_codHeroe, (select codCard from card
where nameCard = par_nameCard));
end; $$
```

2. **P_legendaryfilter** este procedimiento filtra las legendarias del mazo, de tal modo que no te permita repetir las legendarias. Si repites una legendaria te borrará el mazo porque no puede existir.

```
/* Filtro legendarias del mazo */
drop procedure if exists p_legendaryFilter $$
20éroe20 procedure p_legendaryFilter()
begin
    create view v_legendariasrepetidas as select count(*) from deck where
Rarity like 'Legendary' group by nameCard having count(*) > 1;
if (select count(*) from v_legendariasrepetidas)>= 1 then truncate table deck;
end if;
drop view v_legendariasrepetidas;
end; $$
```

3. **P_counter** contará las cartas que estén repetidas y si existe alguna que lo esté más de 2 veces te borrará el mazo ya que no está permitido.

```
/* Filtro contador de cartas repetidas del mazo */
drop procedure if exists p_counter $$
20éroe20 procedure p_counter()
begin
    create view v_conter as select count(*) from deck group by nameCard having
count(*) > 2;
if (select count(*) from v_conter)>= 1 then truncate table deck;
end if;
drop view v_conter;
end; $$
```

4. **P_createdeck** es un procedimiento que te genera mazos aleatorios a partir de la clase que introduzcas.

```
Set max_sp_recursion_depth=255 $$ /* Cambio del limite de recursividad */

/* Procedimiento que genera un mazo aleatorio */
drop procedure if exists p_createdeck $$
create procedure p_createdeck(in par_codHeroe varchar(20))
begin

    declare par_contador smallint unsigned default 0;
    declare par_nameCard varchar(50);
    declare par_codCard int;

    truncate table deck;
        while par_contador < 30 do /* Crea el mazo */
            set par_codCard = floor((select count(*) from Card) * rand()); /*
Selecciona la carta*/
            /* Limitador de Clase */
            if ((select play_codHeroe from play where play_codCard =
par_codCard) = par_codHeroe) or
            ((select play_codHeroe from play where play_codCard = par_codCard)
= “Everyone”) then
            /* Fin limitador de clase */
                set par_nameCard = (select nameCard from Card where codCard =
par_codCard);/* Establece el nombre */
                insert into deck values (
                    par_codCard,
                    par_nameCard,
                    (select play_codHeroe from play where play_codCard =
par_codCard),
                    (select rarity from card where par_codCard = codCard) );
                set par_contador = par_contador +1;
            end if;
        end while;
    /* Filtros*/
    call p_counter();
    call p_legendaryFilter();
    if (select count(*) from deck) = 0 then call p_createdeck(par_codHeroe);
    end if;
    /* Fin de filtros*/
end; $$
```

5. **P_beaheroe** es un procedimiento que te permite sustituir a un héroe del juego por el nombre que prefieras. Puedes especificar el nombre del héroe o simplemente dejar que sea aleatorio.

```
/* Procedimiento que cambia el nombre de un 22éroe */
drop procedure if exists p_beaheroe $$
create procedure p_beaheroe(in par_nameHeroe varchar(20), in par_classHeroe
varchar (20))
begin

    declare par_randomClass smallint unsigned default 0;

    if
        par_classHeroe like ‘’ or
        par_classHeroe like ‘Aleatorio’
    then
        while par_randomClass= 0 do
            set par_randomClass = floor(9 * rand());
        end while;
        case par_randomClass
            when 1 then set par_classHeroe = “Druid”;
            when 2 then set par_classHeroe = “Hunter”;
            when 3 then set par_classHeroe = “Mage”;
            when 4 then set par_classHeroe = “Paladin”;
            when 5 then set par_classHeroe = “Priest”;
            when 6 then set par_classHeroe = “Rogue”;
            when 7 then set par_classHeroe = “Shaman”;
            when 8 then set par_classHeroe = “Warlock”;
            when 9 then set par_classHeroe = “Warrior”;
        end case;
    end if;
    update heroe set nameHeroe = par_nameHeroe
        where codHeroe = par_classHeroe;

end; $$
```

6. **P_updatevdeck** es el procedimiento que actualiza la vista del mazo.

```
/* Procedimiento que actualiza la vista del mazo*/
drop procedure if exists p_updateVDeck $$
23éroe23 procedure p_updateVDeck()
begin
    drop view if exists v_deck;
    create view v_deck as select nameCard, Heroe, Rarity, count(*) as ‘Number’
from deck group by nameCard;
end; $$


/* Evento que vacia el mazo cada semana */
23éroe23 23éroe23 ev_DelBackupCard
on 23éroe23ió every 1 week
do truncate deck $$
```

Funciones.

Las funciones son las siguientes:

1. **F_avgmanacostdeck** que muestra la media del coste del mazo.

```
/* Funcion que calcula la media del coste de mana del mazo */
drop function if exists f_AvgManaCostDeck $$
create function f_AvgManaCostDeck()
    returns smallint(50)
begin

    declare avgcost smallint(50);

    set avgcost = (select floor(avg(cost)) from card inner join deck where
codCard = deck_codCard);
    return avgcost;
end; $$
```

2. **F_countlegendary** que muestra el numero de legendarias que existen.

```
/* Funcion que cuenta el numero de legendarias que existen */
drop function if exists f_countLegendary $$
create function f_countLegendary()
    returns smallint(100)
begin

    declare counter smallint(100);

    set counter = (select count(*) from card where rarity like 'Legendary');
    return counter;
end; $$
```

3. **F_heroedeck** que muestra el nombre de la clase del mazo.

```
drop function if exists f_heroeDeck $$
create function f_heroeDeck()
    returns varchar(50)
begin

    declare heroeDeck varchar(50);

    set heroeDeck = (select distinct Heroe from deck where Heroe not like
'Everyone');
    return heroeDeck;
end; $$
```

4. **F_expansion** que muestra el número de expansiones.

```
/* Funcion que cuenta el numero de expansiones existentes */
drop function if exists f_expansion $$
create function f_expansion()
    returns smallint(20)
begin

    declare counter smallint(100);

    set counter = (select count(*) from expansion where nameExpansion not like
'Classic');
    return counter;
end; $$
```

5. **F_heroe** que cuenta el número de héroes.

```
drop function if exists f_heroe $$
create function f_heroe()
    returns smallint(20)
begin

    declare counter smallint(100);

    set counter = (select count(*) from heroe where codHeroe not like
'Everyone');
    return counter;
end; $$
```

Disparadores.

Los disparadores son acciones que se realizan cuando ocurre un evento concreto

1. **T_nulldescriptioncard** que cada vez que se inserte una carta que no tenga descripción lo establecerá como NULL.

```
drop trigger if exists t_nullDescriptionCard $$
create trigger t_nullDescriptionCard
    before insert on card for each row
        if new.descriptionCard like '' then set new.descriptionCard = NULL;
        end if; $$
```

2. **T_deck** que comprobará que el mazo no tenga legendarias repetidas ni cartas que se repitan más de dos veces.

```
/* Disparador que se asegura de que el mazo esta bien creado */
drop trigger if exists t_deck $$
create trigger t_deck
    before update on deck for each row
    call p_legendaryFilter();
    call p_counter();
$$
```

3. **T_deckmaker** que si se borra el mazo entero te generará uno nuevo de una clase aleatoria.

```
/* Creador de mazos */
drop trigger if exists t_deckmaker; $$
create trigger t_deckmaker
    before delete on deck for each row
        begin

        declare par_randomClass smallint unsigned default 0;
        declare par_classHeroe varchar (20);

        if (select count(*) from deck) = 0 then
            while par_randomClass= 0 do
                set par_randomClass = floor(9 * rand());
            end while;
            case par_randomClass
                when 1 then set par_classHeroe = "Druid";
                when 2 then set par_classHeroe = "Hunter";
                when 3 then set par_classHeroe = "Mage";
                when 4 then set par_classHeroe = "Paladin";
                when 5 then set par_classHeroe = "Priest";
                when 6 then set par_classHeroe = "Rogue";
                when 7 then set par_classHeroe = "Shaman";
                when 8 then set par_classHeroe = "Warlock";
                when 9 then set par_classHeroe = "Warrior";
            end case;
            call p_createdeck(par_classHeroe);
        end if;
    end;
        $$
```

4. **T_heroelimiter** solo te permite añadir nuevos héroes de las clases existentes.

```
/* Disparador que solo permite insertar heroes de las clases existentes */
drop trigger if exists t_heroeLimiter $$
create trigger t_heroeLimiter
    after insert on heroe for each row
        if
            new.codHeroe not like 'Druid' or
            new.codHeroe not like 'Hunter' or
            new.codHeroe not like 'Mage' or
            new.codHeroe not like 'Paladin' or
            new.codHeroe not like 'Priest' or
            new.codHeroe not like 'Rogue' or
            new.codHeroe not like 'Shaman' or
            new.codHeroe not like 'Warlock' or
            new.codHeroe not like 'Warrior'
        then delete from heroe where codHeroe = new.codHeroe;
    end if;
    $$
```

5. **T_updatevdeck** mantiene la vista del mazo actualizada.

```
/* Disparador que actualiza la vista del mazo */
drop trigger if exists t_updateVDeck $$
create trigger t_updateVDeck
    before update on deck for each row
    call p_updateVDeck();
$$
```

Conclusión.

Realizar este proyecto me ha resultado muy satisfactorio y me ha resultado muy útil para afianzar los conocimientos aplicados durante el curso. Además, me ha gustado realizarlo ya que es algo totalmente propio creado desde cero, que me ha supuesto un gran esfuerzo ya que me han surgido varios problemas durante el proceso que al arreglarlos me han hecho sentirme muy satisfecho de lo que he aprendido.

Esta base de datos ha hecho que me interese mucho el lenguaje SQL y que me dedique a probar cosas por mi cuenta o incluso a mantener la base de datos de este proyecto actualizada o aumentar y mejorar su estructura añadiendo más tablas.