Bases de Datos I

Laboratorio 1: **M**odelo **E**ntidad-**R**elacionalidad E**X**tendido (**MERX**)

Departamento de Computación
Facultad de Matemática y Computación
Universidad de La Habana

1 Instrucciones

1. Leer la conferencia
2. Por cada sección:
   (a) Leer el ejemplo
   (b) Hacer los ejercicios
   (c) **if** no entender: **goto** 1

2 Conjuntos entidades, interrelaciones y atributos

2.1 Los clanes de Klash Royale

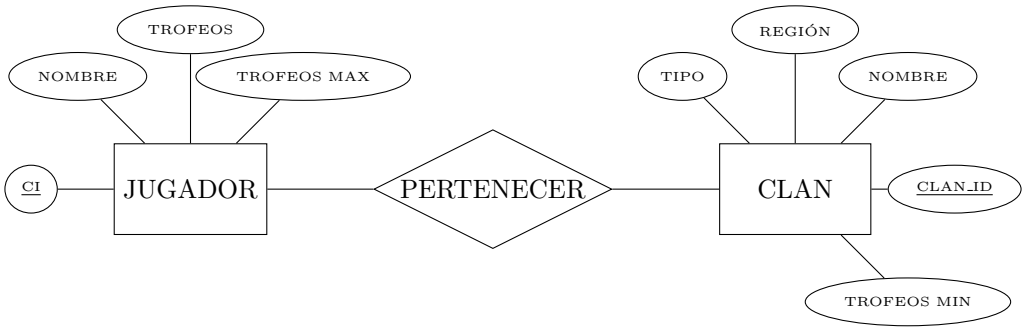
En Klash Royale los jugadores pueden pertenecer a clanes. De los jugadores se conoce su carnet de identidad, nombre, la cantidad de trofeos que tiene actualmente y la cantidad máxima de trofeos que ha alcanzado. De los clanes se conoce su identificador, nombre, región, tipo (abierto o solo invitación) y cantidad mínima de trofeos que necesita un jugador para entrar. Se conoce además que un jugador puede o no pertenecer a un clan y que para existir un clan debe tener al menos un jugador como miembro.

Solución

Primero identificamos los **conjuntos de entidades** de interés, en este caso serían los jugadores y los clanes. Luego procedemos a representarlos gráficamente como en la conferencia.

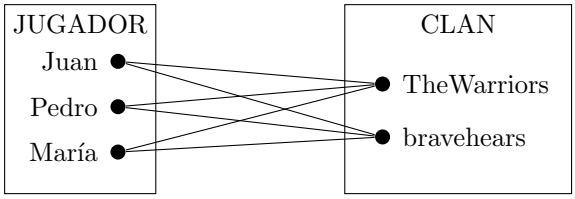
Además conocemos que entre estos conjuntos se establece la **interrelación** "pertenecer" y la representamos en el diagrama:

También podemos identificar **atributos** (o propiedades, siendo un poco informal) sobre estos conjuntos, estos representan los datos que vamos a almacenar sobre las instancias de estos conjuntos.



Ahora necesitamos una forma de diferenciar las instancias que pertenecen a estos conjuntos, osea ¿cómo diferenciamos un jugador de otro?. La idea sería seleccionar un atributo (o atributos) que podamos comparar cuando queramos saber si dos instancias son iguales. El atributo seleccionado se le denomina llave y gráficamente se subraya. En el caso de jugador seleccionamos el carnet de identidad y para los clanes seleccionamos el identificador (que la orden del ejercicio nos asegura que es único).

Finalmente añadimos la cardinalidad a la interrelación. Generalmente, si no se especifica la cardinalidad máxima de la relación se asume que es de **muchos-a-muchos**

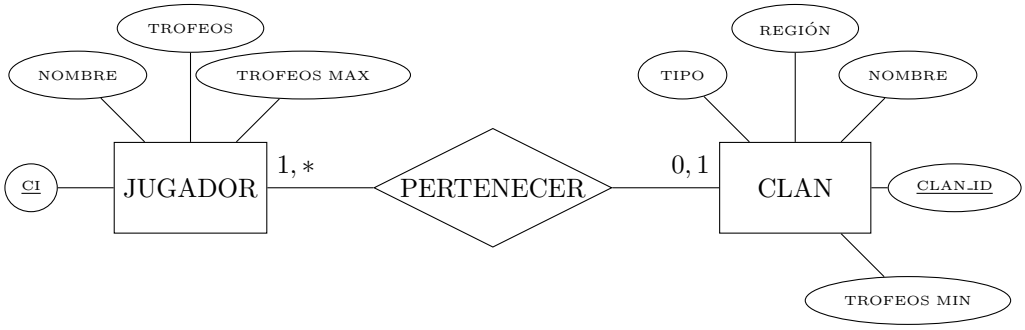


La anterior representación gráfica es un caso posible en una interrelación muchos a muchos y podemos ver que esto no es lo que buscamos, según la especificación un jugador no debería de poder pertenecer a más de un clan.

Por tanto debemos de especificar que esta relación es de **muchos-a-uno**:

- A un clan pueden pertenecer **muchos** jugadores (esto no está especificado pero tienen que ser capaces de usar su **sentido común**.)
- Un jugador puede pertenecer a **un** único clan.

Las cardinalidades mínimas las especificamos de forma análoga.



2.2 Suministro de productos

Se desea modelar el suministro de productos de una tienda. De cada suministrador se conoce su identificador, su nombre, su tipo y el municipio a que pertenece. De cada producto se conoce su código, su nombre, su precio y la unidad de medida que le corresponde. Un suministrador suministra un producto en una cierta cantidad. Cada suministrador puede suministrar varios productos. Cada producto puede ser suministrado por varios suministradores.

3 Agregaciones

3.1 Coleccionar y donar cartas en Klash Royale

De la conferencia ya tenemos un MERX, obviando la falta de atributos para los conjuntos entidades ;).

Ahora debemos distinguir bien las capacidades de cada uno de los elementos que estamos utilizando aquí:

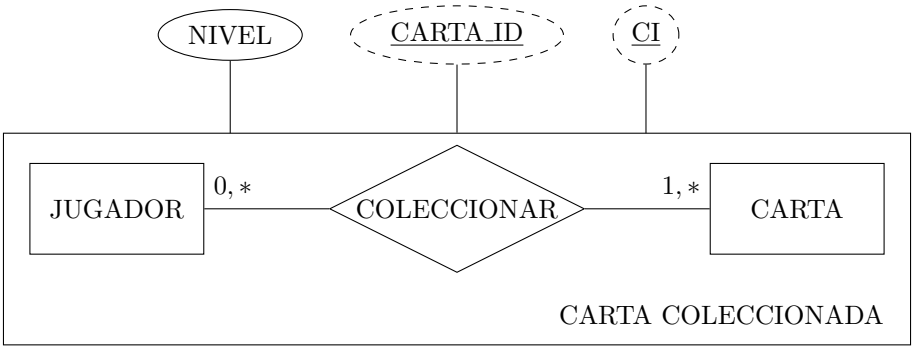
- Los conjuntos de entidades **pueden participar como extremos de una interrelación**.
- Las interrelaciones conectan conjuntos de entidades y **no pueden participar como extremo de otra interrelación**. En la conferencia vimos que **formalmente** los atributos también se representan como interrelaciones, por esto las interrelaciones **no pueden** tener atributos mientras que los conjuntos de entidades sí.

Sin embargo, existen casos donde es conveniente (e incluso obligatorio) que los atributos sean parte de la interrelación y no de los conjuntos entidades. En la conferencia hablamos de que un jugador puede subir de nivel las cartas que colecciona, por tanto debemos tener **nivel** como un atributo, el problema es ¿dónde lo colocamos?.

1. Si hacemos que nivel sea un atributo de JUGADOR entonces todas las cartas coleccionadas por un mismo jugador tendrían el mismo nivel.
2. Si hacemos que nivel sea un atributo de CARTA entonces todos los jugadores que hayan coleccionado dicha carta la tendrían al mismo nivel.

Dado que el resto de las opciones son objetivamente malas la solución sería poner nivel como atributo de la interrelación COLECCIONAR.

Así que hubo que buscar una forma de hacer que las **interrelaciones pudiesen interrelacionarse con otras interrelaciones** (recuerden que los atributos son casos especiales de una interrelación). Para esto surge el concepto de agregación, básicamente lo que hace una agregación es **decorar** (misma idea que el patrón decorador en Programación) una interrelación para añadirle la funcionalidad de poder interrelacionarse como si fuera un conjunto de entidades.



Entonces convertimos la interrelación COLECCIONAR en una agregación CARTA COLECCIONADA y le agregamos el atributo nivel. Ojo, esta agregación se utiliza como un conjunto de entidades por tanto **debe tener una llave especificada** (como cualquier otro conjunto de entidades), por eso las agregaciones heredan la llave de la interrelación que contienen.

3.2 Administración de recursos en la investigación

En la facultad existen varios proyectos de investigación, de los cuales se conoce el identificador, el nombre y el área de relevancia. En estos proyectos trabajan investigadores de los cuales se conoce su identificador, nombre y grado científico. Además la facultad cuenta con ciertos recursos destinados a la investigación como pueden ser equipos de cómputo, APIs de pago, servidores en la nube, etc. De los recursos se conoce su identificador, nombre, tipo y descripción. Los investigadores que trabajan en un proyecto pueden solicitar todos los recursos que sean necesarios para ese proyecto en específico. Un investigador trabajando en un mismo proyecto puede solicitar un recurso una sola vez.

3.3 Comprando en Steam

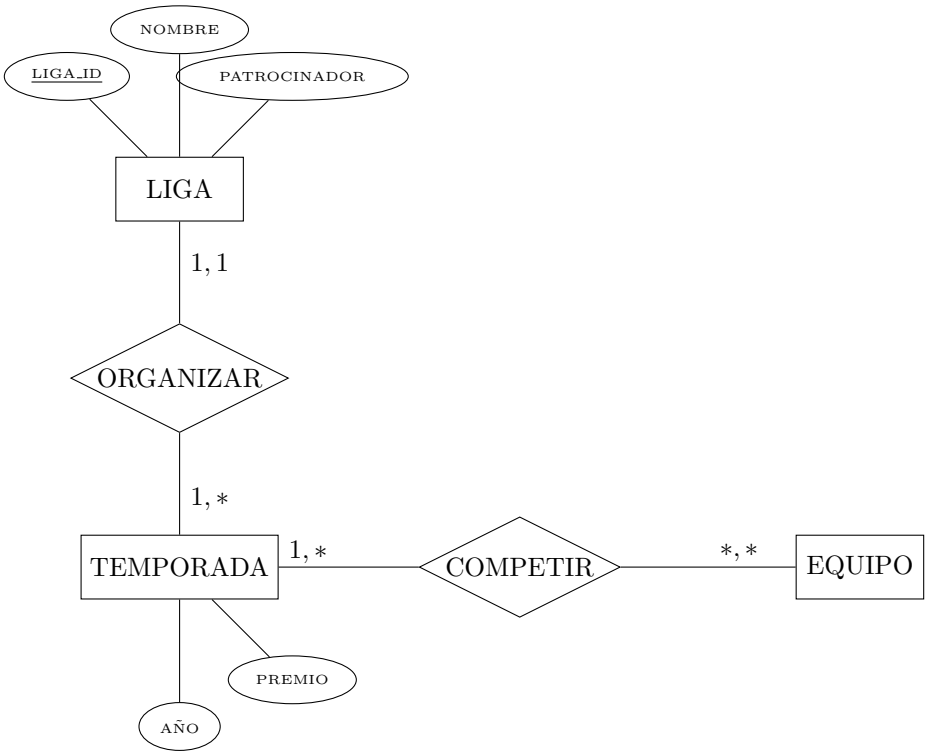
Has sido contratado por Valve para confeccionar una base de datos para el registro de las acciones en su tienda de juegos Steam. En Steam todo usuario registrado tiene un SteamID que lo identifica, un nombre de usuario, una contraseña y nacionalidad. Los juegos de la tienda tienen un código, nombre, desarrollador y fecha de salida. La mayoría de los juegos de la tienda son de pago pero existen algunos que son gratuitos. De los juegos de pago se conoce su precio y la cantidad de Steam points que otorga comprarlo. De los juegos gratuitos se almacena si son monetizados o no y el tipo de monetización aplicada en caso de tenerla (pueden ser loot boxes, micro-transacciones, gachas, etc.). Cuando un usuario compra un juego de pago se convierte en un usuario verificado, estos usuarios pueden gastar sus Steam points comprando artículos de la tienda de regalos de Steam. De los artículos en la tienda de regalos se conoce su identificador, nombre y precio (en Steam points).

4 Entidades fuertes y débiles

4.1 Ligas profesionales en Klash Royale

Las competencias para equipos profesionales se denominan ligas profesionales, de las que se conoce su identificador, nombre y patrocinador. Las ligas organizan una nueva temporada cada año, de las cuales se conoce el año en que se realizó, el premio por ganar la liga, y los quipos que participaron.

La manera intuitiva de representar la especificación anterior como MERX sería la siguiente:



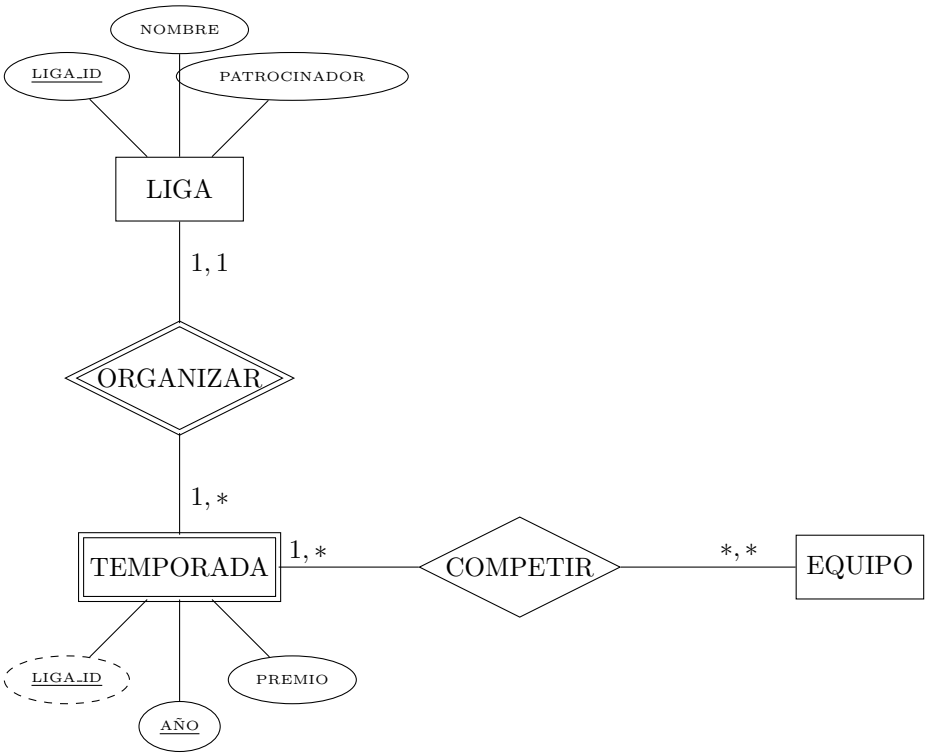
Pero tenemos el problema de escoger una llave para el conjunto de entidades TEMPORADA, la especificación no nos dice nada sobre si tiene un identificador y solo menciona los atributos año y premio.

Por lo que tenemos dos atributos para escoger como llave:

1. Utilizar año como llave, esto provocaría que no pudiesen haber dos temporadas de ligas distintas un mismo año.

2. Utilizar premio como llave, esto provocaría que no pudiesen haber dos temporadas con un mismo premio.

Ninguno de estos atributos permite identificar unívocamente las instancias que **deberían** de poder pertenecer al conjunto TEMPORADA. ¿Cómo arreglamos esto? Combinamos (heredamos) la llave del conjunto LIGA con uno de los atributos del conjunto TEMPORADA, en este caso utilizamos año porque sabemos que solo puede existir una temporada de una misma liga cada año. Si utilizamos premio como parte de la llave entonces no podemos tener dos temporadas de la misma liga con el mismo premio.



Una de las principales características de las entidades débiles es que no pueden existir sin la entidad fuerte con la que están relacionadas (la entidad de la que han heredado la llave). Por tanto al modelar entidades débiles siempre deben de preguntarse si esas entidades deberían de seguir existiendo en caso de ser eliminada la entidad fuerte de la que dependen.

4.2 Cursos optativos en la facultad

En la facultad de Matemática y Computación los profesores imparten varios cursos optativos para los estudiantes. Tanto de los profesores como de los estudiantes se almacena su identificador, su nombre y apellido. De los profesores se conoce además su categoría docente y grado científico. De los estudiantes se conoce el año que cursan actualmente. Cada curso optativo tiene un identificador, un nombre, su duración (en horas clase), el año escolar a partir del cual un estudiante puede tomar el curso y un único profesor que lo imparte. Estos cursos no son fijos en el programa de estudio sino que ofrecen varias convocatorias cada año, de estas convocatorias se conoce la fecha de inicio y el aula donde se va a impartir el curso. Un estudiante puede matricularse en varias convocatorias y en una convocatoria solo pueden matricularse aquellos estudiantes que cumplan el requisito de año escolar para el curso en cuestión.

4.3 Mejorando un poco la tienda

Se desea modelar la solicitud de productos de los clientes mediante órdenes de compra. De los clientes se conoce su número, su nombre, su dirección y el código postal. De los productos se conoce su código, su descripción y su precio unitario. De las órdenes de compra se conoce su fecha de emisión y la fecha de entrega de la solicitud esperada. Un cliente puede emitir o no varias órdenes de compra, pero una orden corresponde a un solo cliente. En una orden se pueden solicitar varios productos, especificando la cantidad de cada uno. Un producto puede solicitarse o no en varias órdenes de compra.