Objetivo del ejercicio:

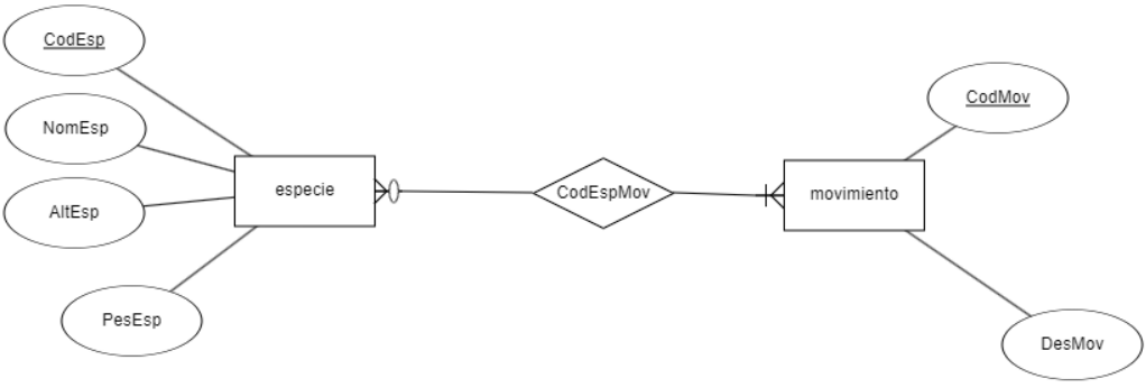
1. *Aprender el tipo de correspondencia muchos a muchos (m a n).*
2. *Repasar las cardinalidades mínimas.*
3. *Concepto de PK compuesta.*
4. *Primera Forma Normal (1FN).*
5. *En SQL: La cláusula ORDER BY de la sentencia SELECT y el tipo de datos DECIMAL.*

Se quiere diseñar una base de datos para almacenar las diferentes especies de Pokémon. Para cada especie almacenaremos su código, nombre, altura y peso. Cada especie tiene diferentes movimientos, de los que almacenaremos su código y descripción. Un movimiento lo pueden tener muchas especies y viceversa. Cuando se almacena una especie se debe indicar al menos un movimiento. Podemos tener almacenado un movimiento y que aún no pertenezca a ninguna especie.

Se pide:

1. **Modelar la base de datos. Para ello haremos:**

   a. Diseño Conceptual de Datos utilizando un Diagrama o Modelo Entidad-Relación. Lo hacemos en papel y lo pasamos a la Herramienta CASE ERD Plus.



   b. **Diseño Lógico de Datos utilizando un Diagrama de Estructura de datos (DED)**. Lo hacemos en papel y lo pasamos a la Herramienta CASE MySql Workbench. En este apartado también vamos a poner el Diagrama Referencial que genera ERD Plus a partir del Modelo Entidad-Relación. Recuerda que el Diseño Lógico de Datos es hacer el modelo relacional y para ello podemos hacer un DED o un Diagrama Referencial.

      En el Diagrama Entidad-Relación anterior hemos detectado un tipo de correspondencia M:N (m a n, o también llamada muchos a muchos). Cuando ocurre esto, en el Diseño Lógico de Datos la **relación** que une las dos entidades implicadas **se convierte en una nueva entidad**. Como esta entidad se convertirá en tabla, debemos ponerle un nombre significativo. La nueva entidad llevará las FKs correspondientes a las PKs de las entidades que relacionaba. A veces, como veremos en otros

ejercicios, llevará también otros atributos.

ELECCIÓN de la PK de la NUEVA ENTIDAD:

Esta nueva entidad al igual que todas, debe tener una PK. Por defecto, ERD Plus y MySQL Workbench ponen las dos FKs **JUNTAS** como PK compuesta (en este caso, clave primaria compuesta por dos atributos). Podemos dejar la PK compuesta de las FKs, siempre que comprobemos que la combinación de los valores que pueden llegar a tomar estas FKs, NO tiene sentido que se repita. Si sí se pueden repetir , o bien no estamos seguro, hay que añadir un nuevo atributo que designaremos como PK.

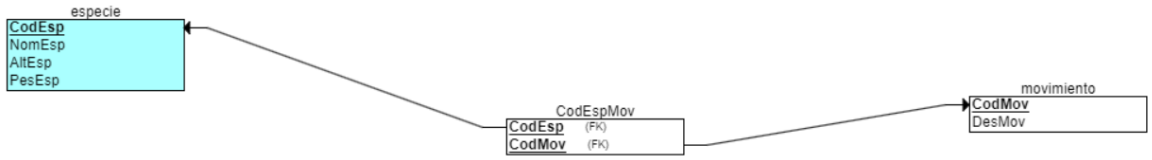
NOTA:

1ª Forma Normal (1FN). No puede existir más de un valor en un atributo.

En caso de haber elegido mal el tipo de correspondencia, y haber elegido 1:N (en cualquiera de los dos sentidos) estaríamos incumpliendo la **1FN**. Puesto que, para poder almacenar la información, tendríamos que almacenar en un atributo más de un valor. Esto haría que nuestro **modelo incumpliera las normas para ser un Modelo Relacional**.

Diagrama Referencial

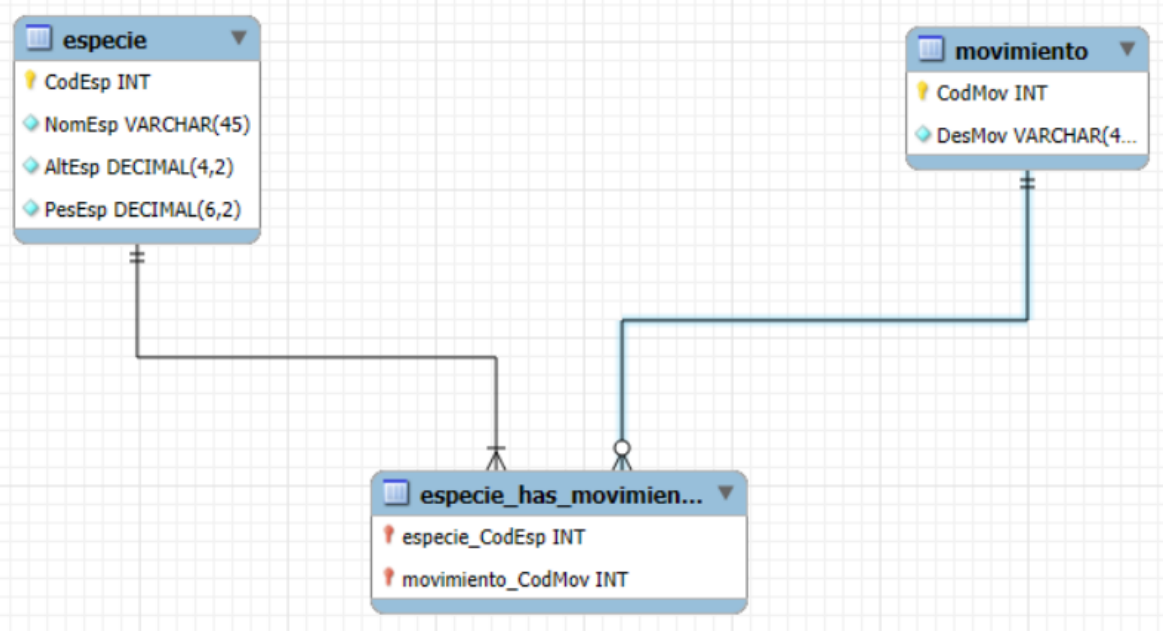
V1: la PK de la tabla creada con la relación muchos a muchos es la unión de las dos FK



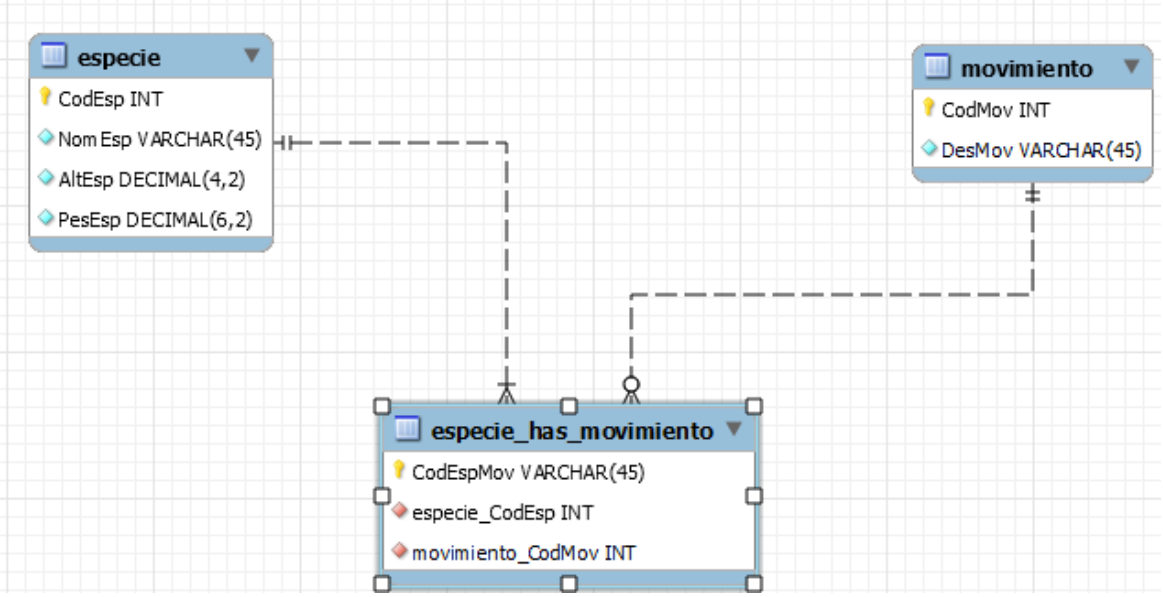
V2: la PK de la tabla creada con la relación muchos a muchos es una diferente a la unión de las dos FK

Diagrama de Estructura de Datos (DED)

V1: la PK de la tabla creada con la relación muchos a muchos es la unión de las dos FK



V2: la PK de la tabla creada con la relación muchos a muchos es una diferente a la unión de las dos FK



En este caso la combinación de CodEsp y CodMov, no tiene sentido que se repitan nunca. Por lo tanto, podemos dejar las dos juntas como PK (esto se llama PK compuesta).

Pero si queremos (NO ES OBLIGATORIO), Podemos poner un PK diferente y dejar CodEsp y CodMov solo como FKs.

c. **Diseño Físico de Datos.** Creamos la base de datos y las tablas en SQL.

```
CREATE TABLE especie
(
  CodEsp INT NOT NULL,
  NomEsp VARCHAR(30) NOT NULL,
  AltEsp INT NOT NULL,
  PesEsp INT NOT NULL,
  PRIMARY KEY (CodEsp)
);

CREATE TABLE movimiento
(
  CodMov INT NOT NULL,
  DesMov VARCHAR(200) NOT NULL,
  PRIMARY KEY (CodMov)
);

CREATE TABLE CodEspMov
(
  CodEsp INT NOT NULL,
  CodMov INT NOT NULL,
  PRIMARY KEY (CodEsp, CodMov),
  FOREIGN KEY (CodEsp) REFERENCES especie(CodEsp),
  FOREIGN KEY (CodMov) REFERENCES movimiento(CodMov)
);
```

2. Insertar datos desde phpmyadmin.

```
INSERT INTO especie (CodEsp, NomEsp, AltEsp, PesEsp)
VALUES (1, "Pikachu", 1.10, 34), (2, "Snorlax", 3.17, 289);

INSERT INTO movimiento (CodMov, DesMov)
VALUES (1, "Andar: el Pokemon andará 1 metro"), (2, "Saltar: el Pokemon saltará 1
        metro");


INSERT INTO codespmov (CodEsp, CodMov)
VALUES (1, 1), (1,2), (2,2)
```

3. Realizar las siguientes consultas en SQL:

- Muestra todas las filas y todos los campos de las tablas.

```
SELECT *
FROM especie;

SELECT *
FROM movimiento;

SELECT *
FROM codespmov;
```

- Muestra algunos campos de las tablas.
- Para cada especie, mostrar todos los movimientos que tiene almacenado. Muestra primero toda la información y posteriormente muestra solo el nombre de la especie y el nombre del movimiento. Ordenar ascendentemente por nombre de la especie.

EJERCICI

- Mostrar para cada movimiento las especies que lo tienen. Muestra primero toda la información y posteriormente muestra solo el nombre del movimiento y nombre de la especie que lo tiene. ¿Qué ocurre con los movimientos que aún no lo tienen ninguna especie?
- Para el movimiento Salto salvaje, mostrar las especies que lo tienen.
- Para la especie Dragón, mostrar todos los movimientos que tiene almacenados.