PROYECTO BASE DE DATOS

POKÉMON DATABASE

CREADO POR:
GERMAN ESCUDERO
FRANCESCO FEVOLI
DANIEL BASCOPE
MANUEL PARRA

Cuerpo de trabajo

Descripción

Esta base de datos está construida desde la idea de los juegos de Pokémon. En esta saga de videojuegos de varias entregas se han creado numerosos pokémon superando con la última entrega los mil, estos han de ser almacenados e identificables por eso decidimos desarrollar esta base de datos desde esa idea . En ella se tratan las características principales de los pokémon como su pertenencia a un grupo, su pertenencia a un tipo, su adquisición de habilidades, su aprendizaje de movimientos y su evolución a otro pokemon en caso de haberla.

Entidades:

pokemon, habilidad, movimiento, grupo_huevo, tipo

Atributos:

pokemon: **pokemonID(PK)**, nombre, altura, peso, fechaObtenido, EO, cintas, fechaLanzado, ataque, defensa, ataqueEspecial, defensaEspecial, putnosSalud, velocidad, IV, EV, experiencia, IDgrupo(FK)
habilidad: **habilidadID(PK)**, descripción, nombre
movimiento: **IDmovimiento(PK)**, nombre, IDtipo(FK), potencia, precisión, efecto
grupo_huevo: **IDgrupo(PK)**, nombre, crianza
tipo: **nombreTipo(PK)**, eficaz, pocoEficaz, débil
es(relación): (**pokemonID**(FK), **nombreTipo**(FK))**(PK)**, dobleTipo
tiene(relación): (**pokemonID**(FK), **habilidadID**(FK))**(PK)**, habilidadOculta
aprende(relación): (**pokemonID**(FK), **IDmovimiento**(FK))**(PK)**, nivel, MT/MO
evoluciona(relación): (**pokemonID**(FK))**(PK)**, nombre, método, nivel

Tipos de datos:

CHAR:
evoluciona: método

VARCHAR:
pokemon: nombre, EO
habilidad: descripción, nombre
movimiento: nombre, efecto
grupo_huevo: nombre
tipo: nombreTipo, eficaz, pocoEficaz, débil
evoluciona(relación): nombre, descripción

Int:
pokemon: pokemonID, cintas, ataque, defensa, ataqueEspecial, defensaEspecial, putnosSalud, velocidad, IV, EV, experiencia, altura, peso
habilidad: habilidadID
movimiento: IDmovimiento, potencia, precisión
grupo_huevo: IDgrupo
aprende(relación): nivel
evoluciona(relación): nivel

BOOLEAN:
grupo_huevo: crianza
es(relación): dobleTipo
tiene(relación): habilidadOculta
aprende(relación): MT/MO

Timestamp:
pokémon: fechaObtenido

Date:
pokémon: fechaIntroducido

Motivos de las claves:

pokemon:
pokemonID(PK) Todos y cada uno de los pokémon tendrán una ID propio e irremplazable que no podrá repetirse nunca aunque compartan especie.
IDgrupo(FK):
habilidad:
habilidadID(PK) Todas las habilidades tendrán un ID propio para ser facilmente diferenciables.
movimiento:
IDmovimiento(PK) Cada movimiento tendrá un ID permitiendo su diferenciación ya que existen movimientos con características iguales.
IDtipo(FK) Todos los movimientos son de un tipo por lo que se le importara la clave IDtipo como clave foránea.
grupo_huevo:
IDgrupo(PK) Todos los pokémon pertenecen a un grupo, este ID permitirá reconocer a que grupo pertenece cada pokémon ya que solo los pokémon que comparten grupo tendrán compatibilidad de crianza.
tipo:
IDTipo(PK) Todos los tipos tienen un ID que los reconoce y los diferencia de forma exclusiva.

es(relación):
(**pokemonID**(FK), **IDtipo**(FK))**(PK)** La relación “es” tiene cardinalidades (1, n)(1, n) por lo que las claves primarias de las entidades que relaciona son importadas como claves foráneas, y la fusión de estas genera la clave primaria de la relación.
tiene(relación):

(**pokemonID**(FK), **habilidadID**(FK))**(PK)** La relación “adquiere” tiene cardinalidades (1, n)(1, n) por lo que las claves primarias de las entidades que relaciona son importadas como claves foráneas, y la fusión de estas genera la clave primaria de la relación.
aprende(relación):
(**pokemonID**(FK), **IDmovimiento**(FK))**(PK)** La relación “aprende” tiene cardinalidades (1, n)(1, n) por lo que las claves primarias de las entidades que relaciona son importadas como claves foráneas, y la fusión de estas genera la clave primaria de la relación.
evoluciona(relación):
(**pokemonID**(FK))**(PK)** La relación de carácter recesivo “evoluciona” tiene cardinalidades (0, n)(0, 1) por lo que las clave primarias de las entidade que relaciona es importada como claves foránea y esta se convierte en la clave primaria de la relación.

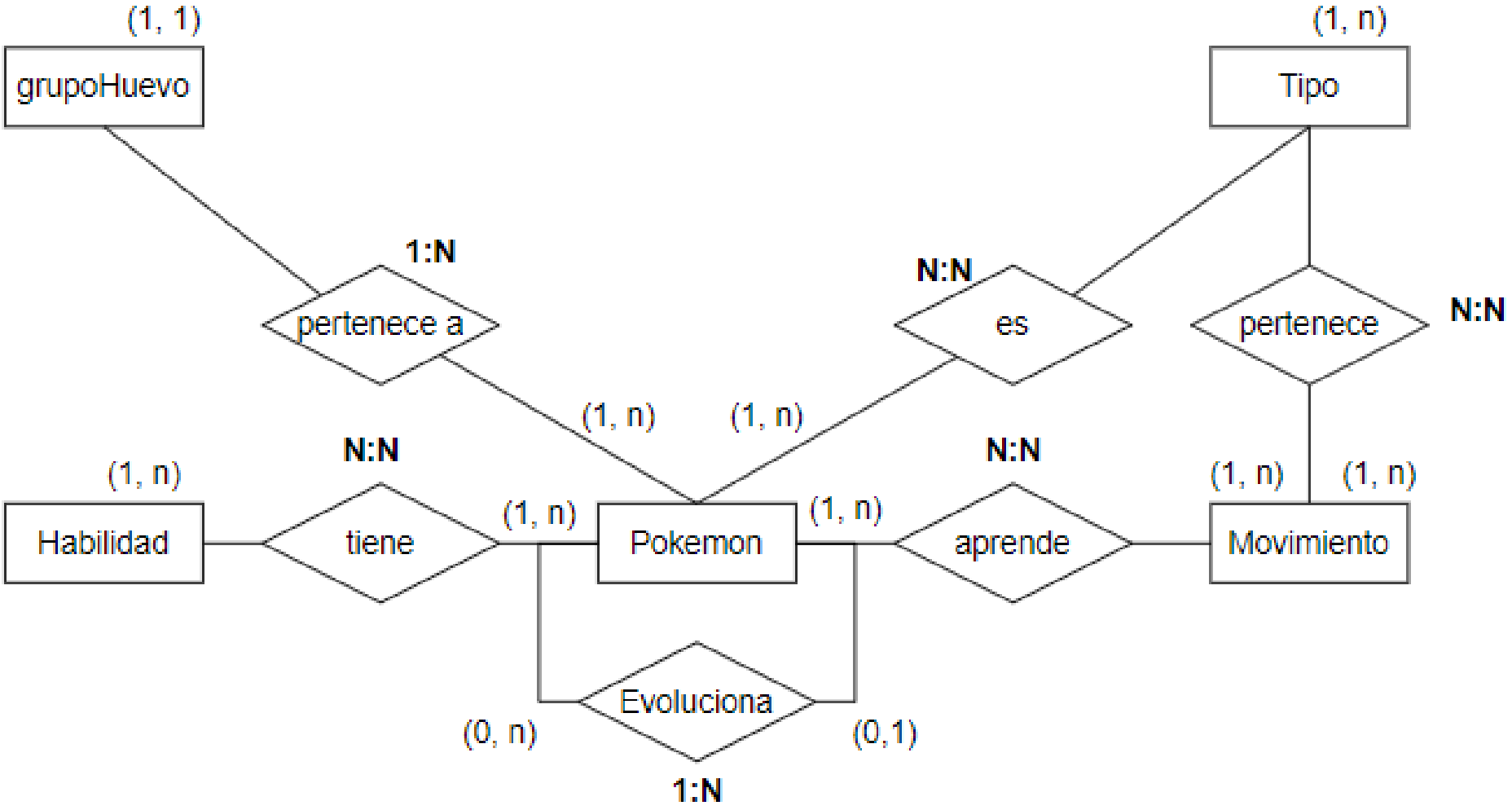
Relaciones:

pokémon **evoluciona** a pokémon
pokemon **tiene** habilidad
pokemon **pertenece a** un grupo huevo
pokemon **es** de tipo
pokémon **aprende** movimientos
movimientos **pertenecen** a un tipo

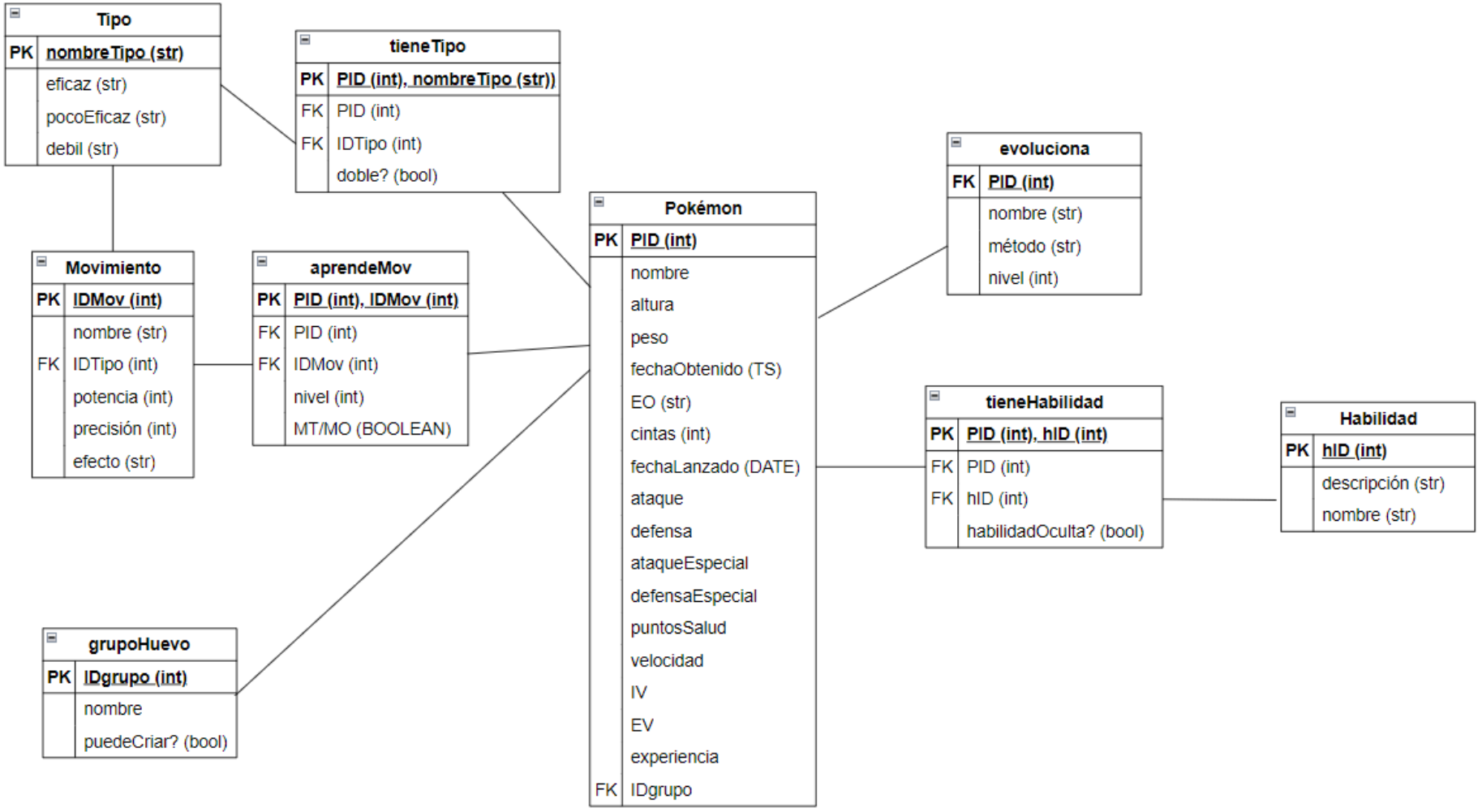
Cardinalidad:

evoluciona (0, n) **N:1** (0, 1): Los pokemon como norma general suelen poder evolucionar pero algunos de ellos carecen de esta capacidad, de los pokémon que pueden evolucionar algunos pueden hacerlo varias veces o pueden evolucionar a diferentes versiones partiendo del mismo pokemon.
tiene (1, n) **N:N** (1, n): Los pokémon tienen como mínimo una habilidad aunque pueden tener más de una, estas pueden ser habilidades normales o habilidades ocultas. Cabe recalcar que hay habilidades que son exclusivas de un pokémon en concreto mientras que hay otras que pueden repetirse en más de uno.
pertenece_a (1, n) **N:1** (1, 1): Todos los pokémon pertenecen a un único grupo huevo, que determina la posibilidad de criar con otros pokémon, que deben ser pertenecientes a este mismo grupo..
es (1, n) **N:N** (1, n): Un pokémon puede tener uno o dos tipos elementales, que les dan ventaja o desventaja frente a otros tipos y combinaciones. Estos tipos no son exclusivos y pueden darse en otros pokemon.
aprende (1, n) **N:N** (1, n): Los pokémon aprenden movimientos a determinados niveles, todos los pokémon aprenden como mínimo un movimiento aunque pueden aprender más de uno. También se pueden aprender movimientos mediantes MT/MO que no requieren un nivel determinado.
pertenece (1, n) **N:N** (1, n): Todos los movimientos pertenecen a un único tipo, y todos los tipos tienen como mínimo un movimiento aunque pueden tener más de uno.

MODELO ENTIDAD-RELACIÓN (MER) :



MODELO RELACIONAL (MR):



Restricciones:

Restricciones tipo 1:

Restricción de tipo not null

Restricciones **not null** de pokemon: nombre, altura, peso, fechaObtenido, EO, fechaLanzado, ataque, defensa, ataqueEspecial, defensaEspecial, putnosSalud, velocidad, IV, EV
Restricciones **not null** de habilidad: descripción, nombre
Restricciones **not null** de movimiento: nombre, precisión, efecto
Restricciones **not null** de grupoHuevo: nombre, crianza
Restricciones **not null** de tipo: eficaz, pocoEficaz, débil
Restricciones **not null** de es(relación): dobleTipo
Restricciones **not null** de tiene(relación): habilidadOculta
Restricciones **not null** de aprende(relación): nivel, MT/MO
Restricciones **not null** de evoluciona(relación): nombre, método, nivel

Restricción de tipo Primary Key

pokemon: **pokemonID(PK)**
habilidad: **habilidadID(PK)**
movimiento: **IDmovimiento(PK)**
grupoHuevo: **IDgrupo(PK)**
tipo: **IDTipo(PK)**

Restricciones tipo 2:

Esta restricción exige que los IV tengan que ser un número igual o mayor que 0 y menor o igual 168 y que los EV tengan que ser un número igual o mayor que 0 y menor o igual a 255.

```
CONSTRAINT CHK_values CHECK (IV>= 0 and IV <= 168 and EV >= 0 and EV <= 255)
```

La restricción impide que los puntos de salud del pokémon sean menores que 0 incluyendo el 0.

```
CONSTRAINT CHK_salud CHECK (pSalud > 0)
```

Esta restricción obliga que la precisión de los movimientos sea mayor a 0 ya que esta significaría que los movimientos no podrían acertar, también obliga que la precisión sea menor o como máximo igual a 100 ya que este supone el límite haciendo que el movimiento no falle nunca.

```
CONSTRAINT CHK_precision CHECK (precisionMov>0 and precisiónMov<=100)
```

La restricción de tipo 2 **CONSTRAINT primary key** es usada para crear claves primarias compuestas en las tablas de relación.

```
constraint pk_tieneTipo primary key (PID, IDtipo)
constraint pk_aprendeMov primary key (PID, IDmov)
constraint pk_adquiereHabilidad primary key (PID, hID)
```

Por último, la restricción de la tabla evoluciona se establece para permitir identificar rápidamente cómo evoluciona cada criatura de una forma genérica, y explicar posteriormente el método completo.

```
CONSTRAINT CHK_metodo  CHECK (método in ('NIVELES', 'PIEDRAS', 'AMISTAD', 'CAMBIOS', 'OBJETOS'))
```

La restricción foreign key presente en varias tablas se usa para establecer claves foráneas en las tablas que referencian a las claves primarias de otras.

```
foreign key (PID) references pokemon(PID) ON DELETE cascade
```

Normas de integridad referencial:
Para las FK de todas las tablas exceptuando pokemon, se aplica **`ON DELETE CASCADE`**, ya que es importante la integridad de todas las tablas y que se borren al borrarse una de las columnas principales referenciadas.

Para la tabla pokemon, se aplica **`ON DELETE SET NULL`**, ya que se quiere permitir que se introduzca un Pokémon sin tener en cuenta su grupo huevo, por lo que puede hacerse poniendo nula la columna de identificación del grupo de huevo de cada criatura.

CONCLUSIÓN

Durante la realización del trabajo, encontramos problemas a la hora de la elección de un tema adecuado para la base de datos, sin embargo, logramos encontrar un tema que nos apasionarse como lo es Pokémon, y volcamos todo nuestro conocimiento en la saga para crear una base de datos fiel a las estadísticas e información disponibles en los juegos.

A la hora de elaborar el script SQL, ciertos errores debieron ser resueltos para la correcta ejecución del mismo, tales como el uso de la base de datos o la creación de las tablas. Cambios de última hora del mismo modo dificultaron la inserción correcta de los datos, que tuvieron que ser arreglados para ajustarse a los nuevos requerimientos.

BIBLIOGRAFÍA

https://www.w3schools.com/sql/sql_or.asp

https://www.wikidex.net

https://www.pokexperto.net/

https://www.mysqltutorial.org/mysql-basics/mysql-check-constraint/

https://www.percona.com/blog/how-to-use-check-constraint-in-mysql-8/

https://www.w3schools.com/mysql/mysql_check.asp

https://www.dongee.com/tutoriales/error-1064-mysql/