Grupo 9 – Entrega 4

Herramientas utilizadas para la persistencia

De acuerdo a lo establecido, persistimos los datos de interés en una base de datos MySQL.

La estrategia utilizada para persistir los mismos fue mediante el uso del ORM *Hibernate,* que nos permite integrar un proyecto Java con uno MySQL.

Dentro de la misma, podemos verificar los siguientes aspectos que resultan positivos al usar esta tecnología:

- Nos permite desarrollar mucho más rápido, ya que tenemos todo centralizado en una única interface.
- Permite trabajar con la base de datos por medio de entidades en vez de queries, dándonos una forma de trabajar con ella a más alto nivel, simplificando a su vez la cantidad de cosas que debemos controlar al mismo tiempo en caso de haber errores en tiempos de testeo.
- Nos ofrece un paradigma 100% orientado a objetos, permitiéndonos no tener que estar en contacto con el lenguaje MySQL en crudo.
- Elimina errores en tiempo de ejecución, ya que el mismo ORM mapea la base de datos y se encarga de realizar las conexiones correspondientes con la misma de forma autónoma.
- Mejora el mantenimiento del software, ya que los desarrolladores no tendremos que encargarnos de modificar la base de datos directamente sobre la misma en caso que debamos hacer cambios sobre el modelado en un futuro.

Decisiones relevantes tomadas durante el modelado de la Base de Datos

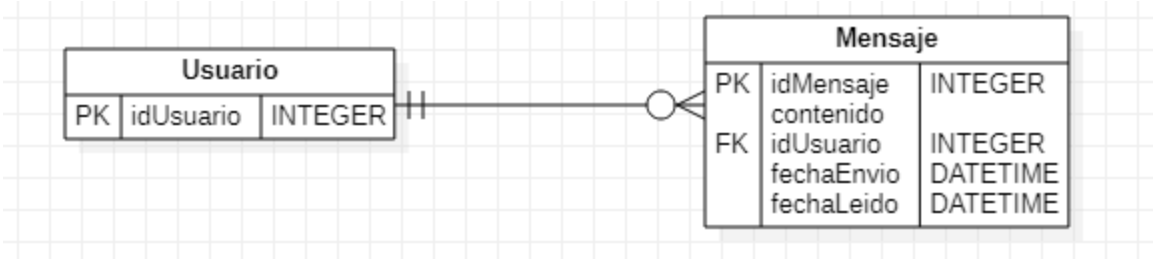
Viendo la Base de Datos, podemos dividirla en partes que nos han resultado de interés mencionar y especificar cómo las diseñamos, tanto por su complejidad como por su relación entre ellas:

1. La persistencia de los mensajes enviados a los usuarios.
2. La persistencia de las operaciones de egreso, tanto como de sus presupuestos relacionados.
3. La persistencia de las categorías y criterios para las operaciones de egreso que realizan las entidades jurídicas.
4. La persistencia de los sectores y categorías de los cuales una entidad jurídica forma parte.
5. La herencia ocasionada dentro de las entidades jurídicas: Empresa vs Organización Social.

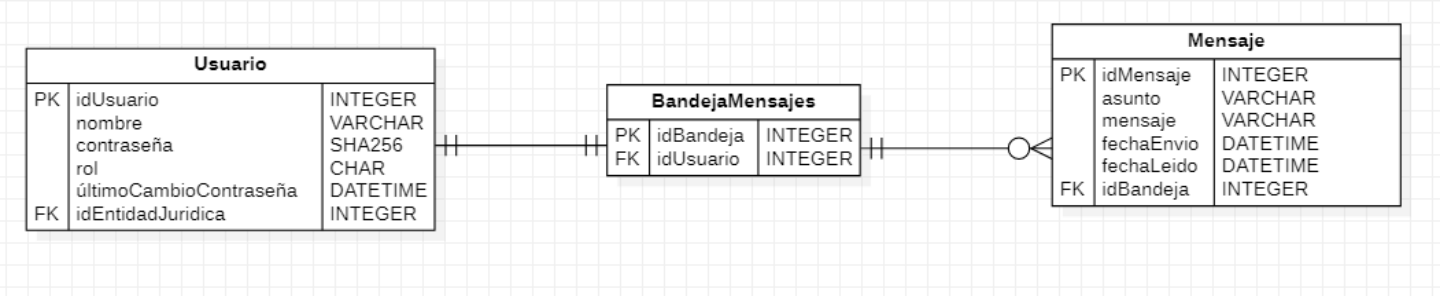
1. *La persistencia de los mensajes enviados a los usuarios.*

Nosotros sabemos que los usuarios reciben mensajes, los cuales son generados y enviados automáticamente por nuestro sistema, siendo objeto del resultado de la validación de una operación de egreso que realiza una entidad jurídica. Ahora bien, esto puede ser persistido de dos formas:

La primera es asociar directamente los mensajes a los usuarios a los cuales se envían los mismos. De esta forma, cada vez que queramos notificar a un conjunto de revisores sobre el resultado de una operación, haremos, por ejemplo, 20 instancias diferentes de un mensaje que contiene exactamente el mismo contenido. Ilustrando:



La segunda opción es mediante una bandeja de mensajes, tal que simplemente instanciemos un solo mensaje que luego será enviado al conjunto deseado de revisores, quedando de esta forma:

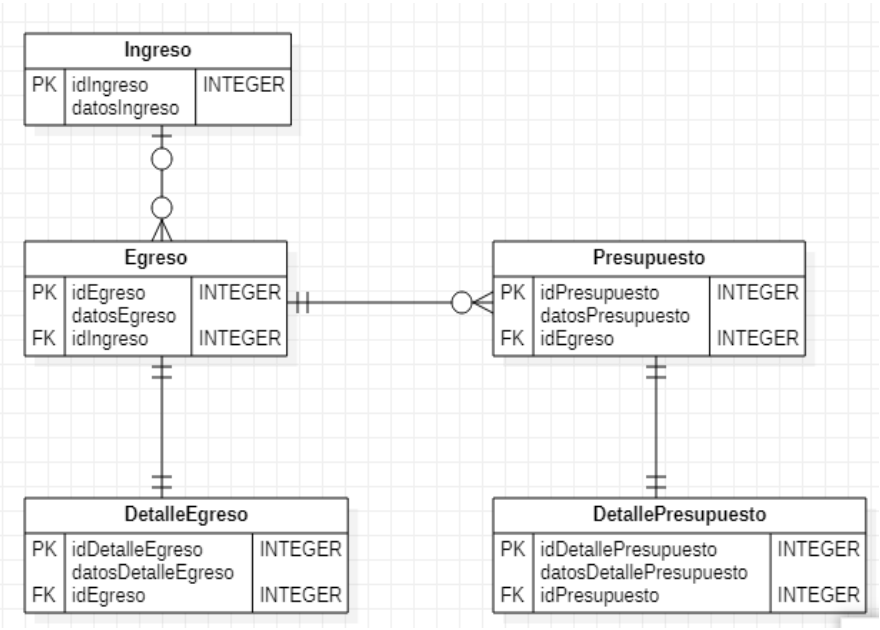


Finalmente nos decantamos por la segunda opción, ya que si bien podríamos acceder a los mensajes que tiene un usuario directamente a través de la tabla Mensaje. Esto nos llevaria a quitar la bandeja de mensajes y, por lo tanto, se pierde un grado de abstracción, generando mas responsabilidades a nuestra clase Usuarios. Es por esto, que nos parece más ordenado respetar nuestro modelo de objetos para seguir manteniendo ese grado de abstracción sobre los mensajes.

2. *La persistencia de las operaciones de egreso, tanto como de sus presupuestos relacionados.*

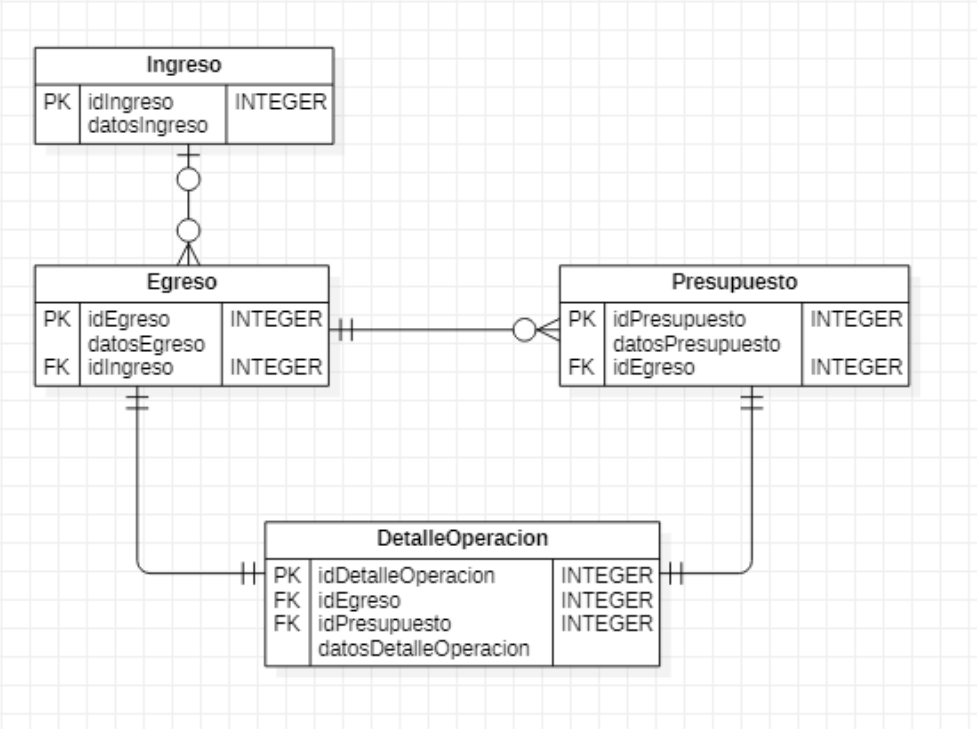
Comenzamos de algo simple: cada ingreso tiene asociado uno o más egresos, tal que en la tabla egreso tendremos siempre un campo *idIngreso* que los asocie.

Ahora bien, tenemos dos formas de representar a los presupuestos con los egresos, tal que tengan el mismo comportamiento y se puedan persistir de la misma forma: la primera es tratar a los egresos y a los presupuestos como dos cosas totalmente diferentes. Quedando así:



Cabe destacar que el modelo real está muy simplificado en esta ilustración. Es sólo con el objetivo de poder representar entre las diferentes opciones de modelado que teníamos.

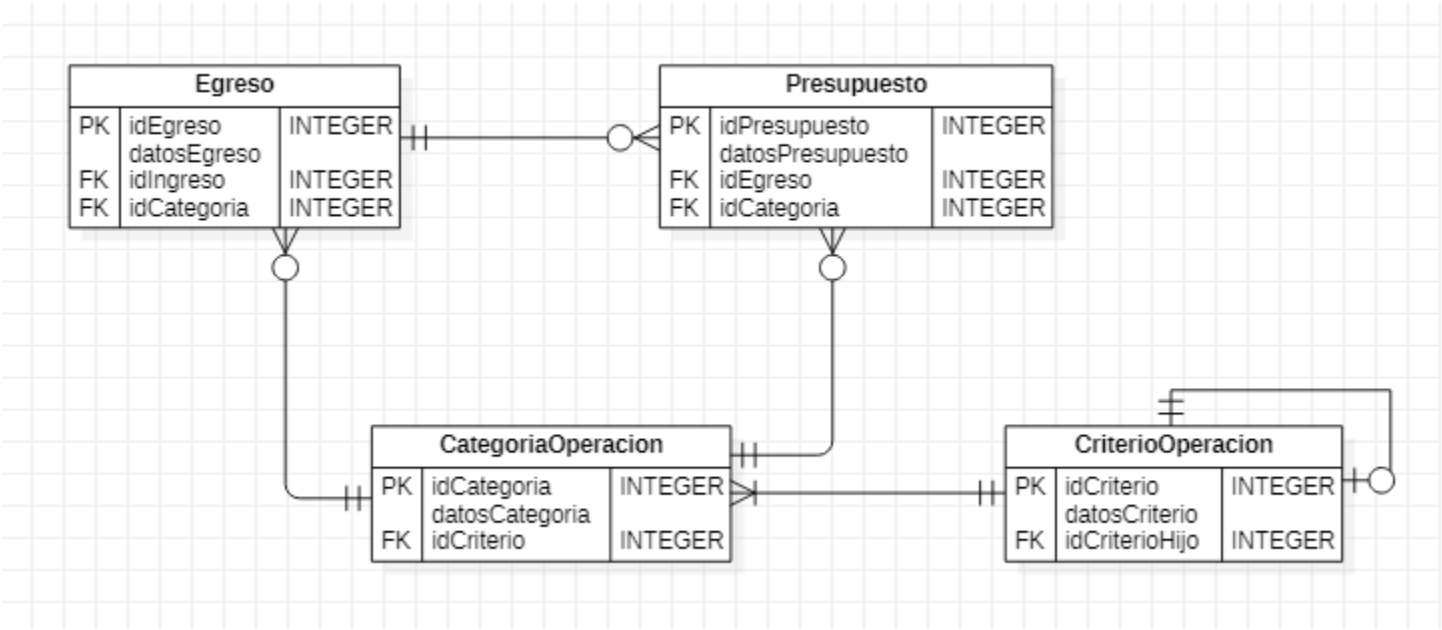
Retomando, tenemos una segunda forma de modelar esta relación, y parte a partir de: qué tanta relación existe entre los egresos y los presupuestos. Parándonos en nuestro dominio, sabemos que los presupuestos serán cargados con exactamente los mismos atributos que los egresos, por lo cual nos decidimos por implementar un modelado como el siguiente:



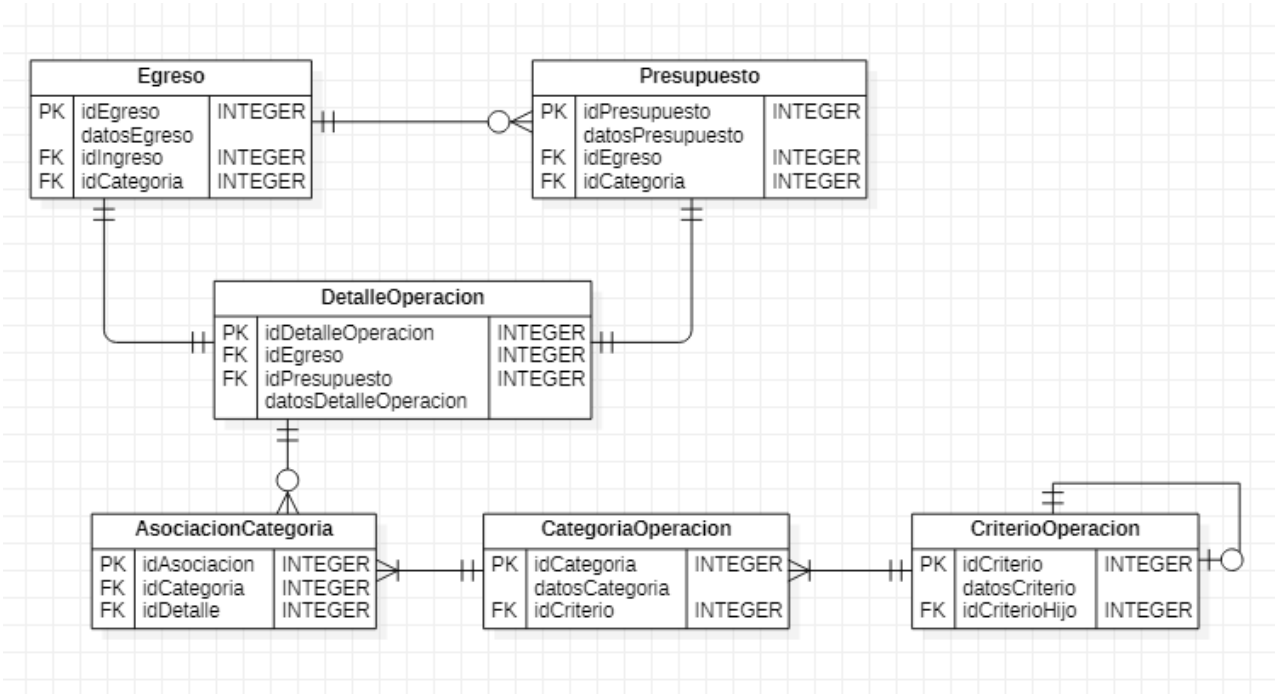
De esta forma, podremos tratar como nulas a las claves foráneas de acuerdo al tipo de operación que se desee cargar en el sistema.

3. *La persistencia de las categorías y criterios para las operaciones de egreso que realizan las entidades jurídicas.*

Expandiendo un poco el caso previo que acabamos de mostrar, se nos presentó una situación similar relacionada a la persistencia de las categorías y criterios de los cuales forman parte cada una de las operaciones egreso o presupuestos. Cada una podría ser tratada por separado, quedando de la siguiente forma:

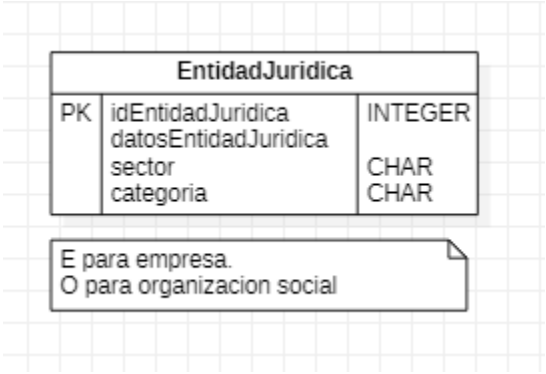


Pero aquí podemos aplicarnos un poco más en nuestro dominio, y entender que cada una de las relaciones que se realizan sobre el egreso, el presupuesto y las categorías forman también parte del detalle de cada una de estas operaciones (que son independientes de los ítems que tienen asociados. Esto puede verse en el DER completo), quedando entonces de la siguiente forma:

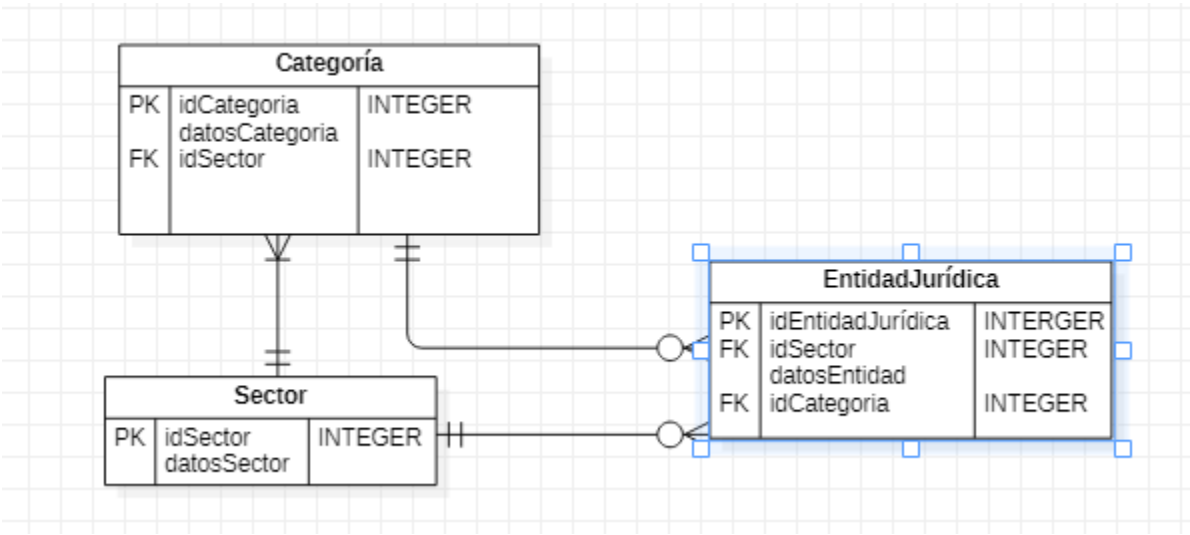


4. *La persistencia de los sectores y categorías de los cuales una entidad jurídica forma parte.*

En primer lugar, tuvimos que consultar sobre la importancia de tener persistidas estas entidades en la base de datos. La duda surgió a partir de que en realidad se podrían tener los campos *sector* y *categoría* dentro de la entidad jurídica y cambiarlos cada vez que sea necesario, de acuerdo al cambio de los campos relacionados a la categorización de cada entidad. Quedando de esta forma:



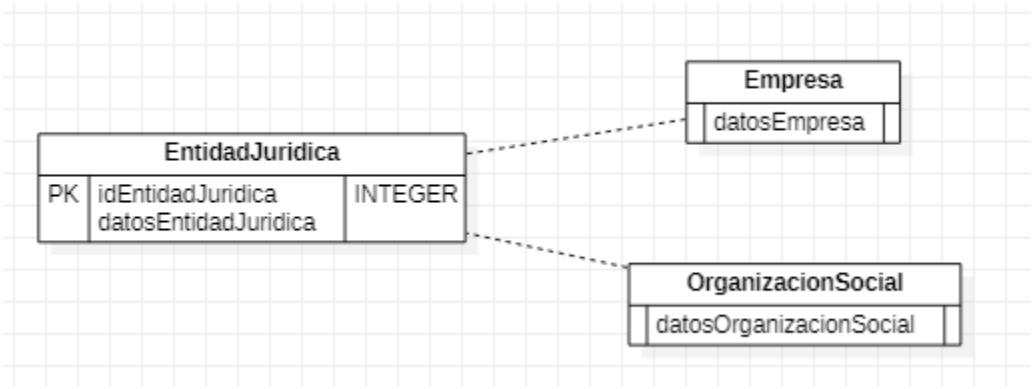
Pero nos dimos cuenta que en realidad sí nos interesa mantener persistidos cada sector y cada categoría de las cuales una entidad jurídica forma parte, ya que en el caso de querer mantenerlos en memoria nos estaríamos arriesgando a perder cada uno de los objetos que serían cada uno de ellos entonces en el caso de que el servidor se caiga. Finalmente queda de esta forma:



Cabe destacar que si bien cada sector conoce cada una de sus categorías, es necesario que la entidad jurídica conozca la categoría a la que pertenece directamente, ya que existe comportamiento detrás que facilita su persistencia de esta forma.

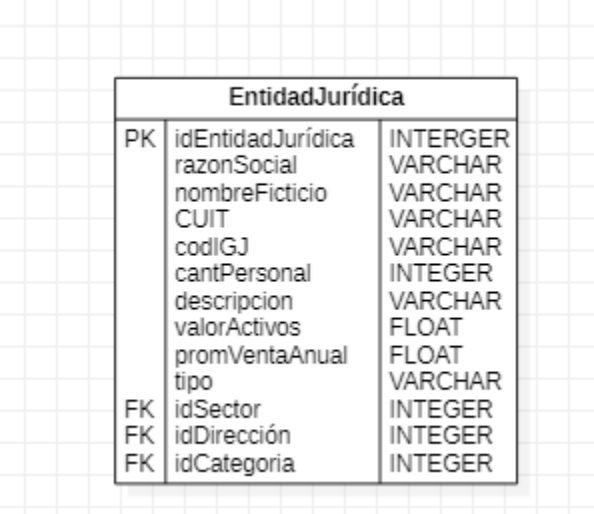
5. La herencia ocasionada dentro de las entidades jurídicas: Empresa vs Organización Social.

Aquí se da un caso de herencia dentro de la entidad jurídica: existen por un lado las empresas y por otro las organizaciones sociales. Esto en diagrama queda representado de la siguiente forma:



Luego, debemos realizar algún tipo de estrategia para mapear la base de datos.

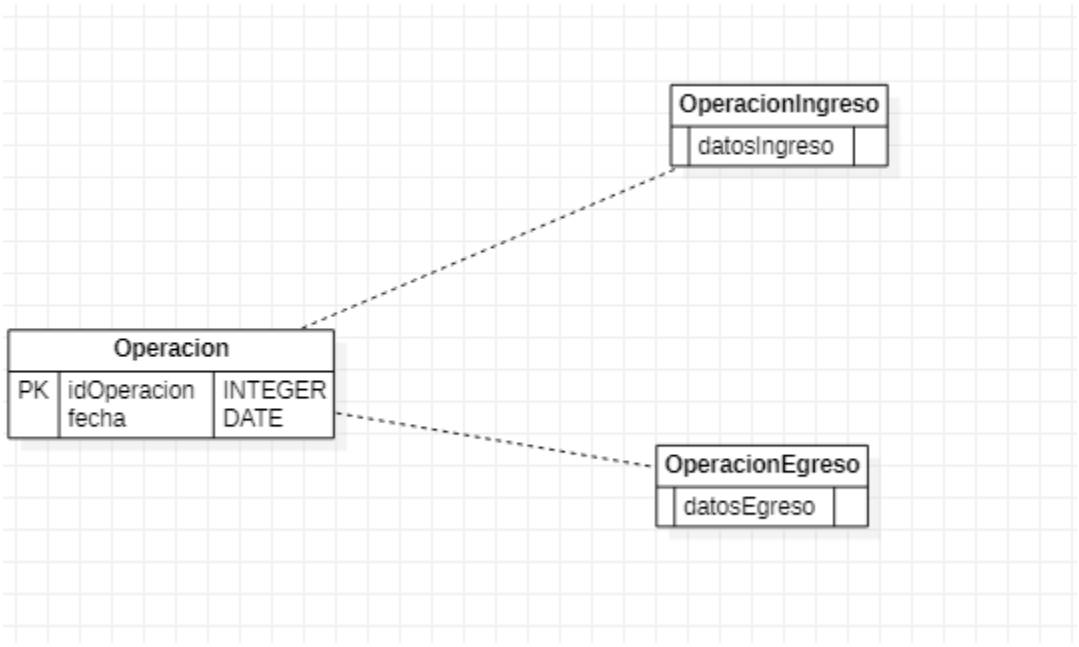
Dentro de nuestras posibilidades y aplicando nuestro dominio, nos damos cuenta que ambas entidades no difieren de muchos atributos, por lo cual decidimos optar por una estrategia *single class* cuyo campo discriminador será “tipo”. Quedando así:



Cabe destacar que, la estrategia *single class* no brinda una gran flexibilidad en caso de cambios pero tiene mejor performance que las otras estrategias. Asumimos que los posibles cambios no tendrán gran impacto en nuestro modelo y preferimos perder esa pizca de flexibilidad en performance.

6. La herencia ocasionada dentro de las operaciones: Operación Egreso vs Operación Ingreso.

Aquí se da un caso idéntico al anterior, donde existe una herencia entre las operaciones. Esto en diagrama queda representado de la siguiente forma:



En este caso, ambas comparten atributos. Esto nos llevó a pensar en utilizar la estrategia *single class,* pero como explicamos en el caso anterior, nos cierta rigidez a cambios. Para una futura iteración existen cambios en los datos y procesamiento de las Operaciones de Ingresos. Por lo que inclinarse por esa estrategia nos llevaría a ganar en performance pero perderíamos en modificabilidad y extensibilidad. Es por esto, que decidimos mantenerlas por separado y utilizar una estrategia *joined,* tal como presentamos ahora: