Diseño

Se creó una pequeña base de datos donde se pueda tener usuarios de una plataforma (sistema, usuario y socio); con dos tipos de pago (tarjeta de crédito y paypal).

Se consideró que la empresa tiene algunos autos comprados y otros que se prestan por parte de los socios.

La empresa tiene unas cuantas sucursales donde se hace la renta y entrega de cada auto.

Por parte de los autos podemos saber algunas cosas básicas, como tipo de motor, capacidad del auto (número de personas), modelo, numero de registro del carro ( no se incluyó la placa ya que las placas cambian en determinado tiempo; sin embargo hubiera sido un buen aprovechamiento mostrando registros históricos con las distintas placas que pudo tener un auto).

Por parte del usuario se puede ver su email y celular con el cual se registran y además se puede ver relacionar con los datos de persona al que están relacionado ese usuario. En caso de usuarios de sistema se ligó directamente a una persona física para poder asignar una responsabilidad y no dejarlo ambiguo en la denominación “sistema”.

Por parte de los métodos de pago se tiene una “cartera” en la cual se puede asociar más de un método de pago a un usuario e indicar en una bandera cual es el principal.

Proceso

El proceso de carga va desde cargar algunos datos históricos y catálogos en una base operativa y transaccional; después cargar una capa dimensional con las versiones activas de cada registro y al final crear una tabla agregada con algunos cálculos.

Dentro del README.md del proyecto se indica las instrucciones para cargar la plataforma y la ejecución de la misma; sin embargo el proceso de carga es el siguiente:

Se corren dags (y en su caso un scrapper) para cargar los catálogos de la base de datos.

Se corren dags para cargar las tablas históricas y relacionales.

Se corren dags para cargar las tablas dimensionales y agregada.

Se verifica en el resultado de la tabla agregada

Arquitectura y tecnología

Se eligió Airflow por ser un etl libre y versátil, en el cual se podía montar soluciones de distintas maneras de cargas y no limitarse únicamente a usar scripts sql. También por el orquestamiento y el UI de la plataforma para entender más fácil el proceso.

La arquitectura consta de 3 servidores:

* Mysql: Base de datos transaccional del proyecto
* Airflow: servidor web con tecnología python para el orquestamiento y ejecución de los etls
* Posgresql: Metadata para la airflow

Base de Datos

* Tecnología:

Se eligió Mysql para dar un variedad en la arquitectura ya que Airflow tenía posgresql y se quería probar la capacidad de tener una arquitectura mixta conviviendo

* Estructura:

Existen 2 esquemas/bases de datos. cooler\_car y dim. Las tablas operativas están almacenadas en cooler\_car y las dimensionales en dim.

* Nombrado:

Se creó un prefijo en cada tabla para mantener un estándar y mejor ordenamiento de las tablas según el comportamiento de cada una de ellas.

CAT: Tablas catálogo; las cuales solo almacenan las llaves primarias para mejor acceso.

HIST: Tablas que guardan las distintas versiones de cada registro, heredando la llave primaria de su padre (tabla Cat) y creando una llave subrogada (hash\_key) para indicar que es una nueva versión del registro.

LU-TYPE: Catálogos de tipos de datos necesarios.

REL: Tablas para mantener la relación de dos o más catálogos. Son casi similares a las Hist en estructura pero el propósito es identificar que es la relación de muchos a muchos.

DIM: Tablas denormalizadas en la cata dimensional.

AGG: Tablas con cálculos agregados.

* Diagrama Entidad-Relación

el diagrama entidad relación de la solución está en la carpeta docs

Problemas al realizar el proyecto

* Host de base de datos: Al principio no podía realizarse la conexión entre airflow y mysql por el host de la base de datos, se probó usar una configuración creando un link en docker compose pero se mantenía la situación.

Solución: se creó una red bridge entre los contenedores

* Contenedor inestable: La imagen de mysql constantemente estaba cayendo después de cada insert en la base de datos, mandando el siguiente error: https://bugs.launchpad.net/percona-server/+bug/1712649

En ocasiones el contenedor no volvía a ser usable.

Se intentó cambiar de imágenes que aparentemente duraban más pero terminaban en el mismo escenario.

A demás del error anterior también ocasionalmente se obtuvo el error:

“Most likely, you have hit a bug, but this error can also be caused by malfunctioning hardware”.

Solución: Después de revisar posibles escenarios se detectó que la arquitectura estaba excediendo el poder del procesador.

Y se cambió la arquitectura de airflow executor celery por un executor local. Esto mantuvo la imagen estable; sin embargo se detectó que el procesador seguía estando al 90% por parte de los contenedores.