Avance #3

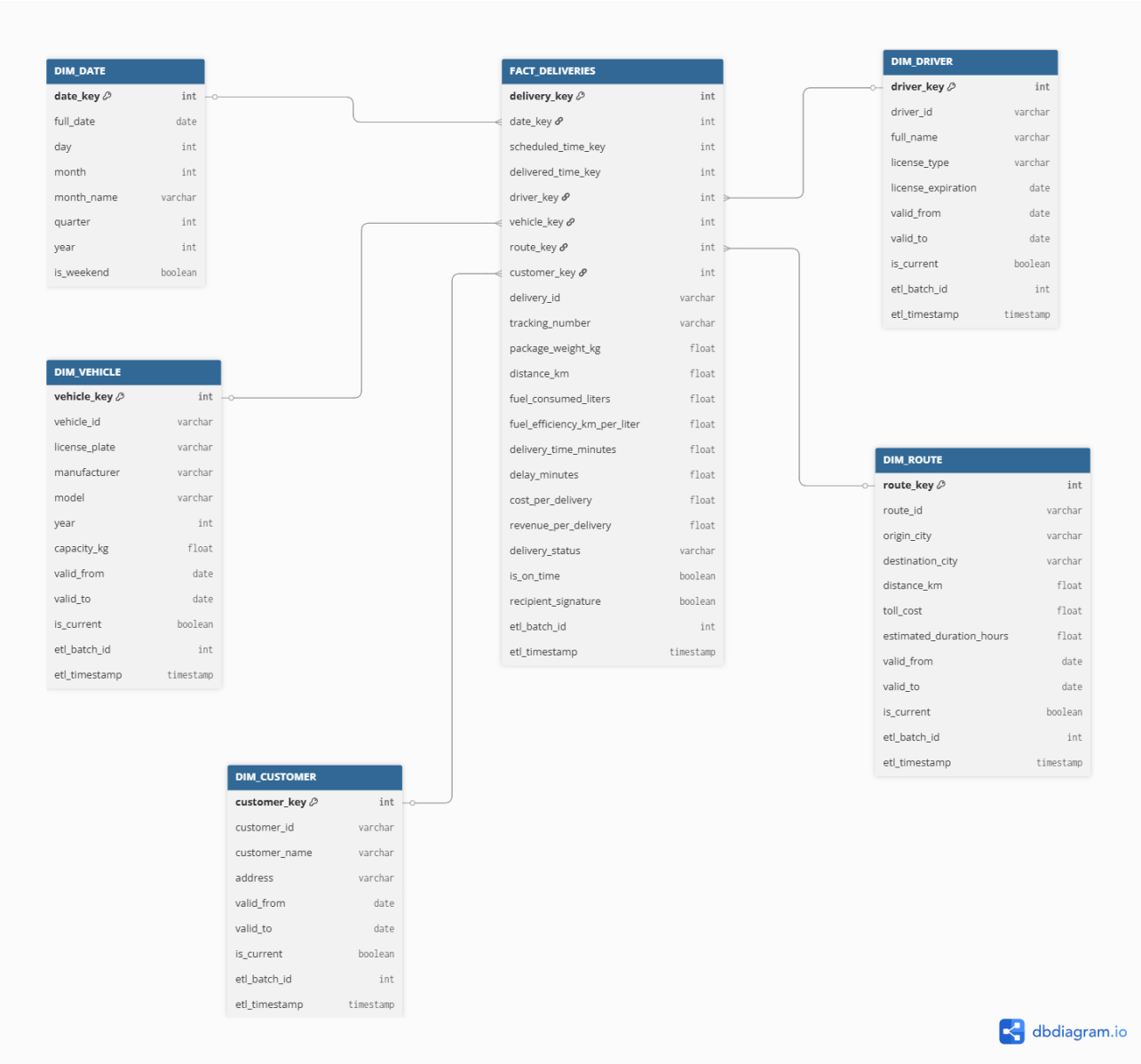
Este avance se divide en dos grandes partes. La primera que es crear un esquema de estrella en SnowFlake y la segunda parte que usar un ETL que se conecta a la base de datos local y extrae datos del día anterior y los sube a SnowFlake mediante una transformación de los datos convirtiéndolos en una tabla de hechos y tablas de dimensiones. Esto con el fin de migrar la base de datos a un esquema tipo estrella. Así que vamos por partes.

Creación del DataWarehouse en SnowFlake:

Para crear un DataWarehouse en SnowFlake primero debemos crearnos una cuenta en SnowFlake, para esto entramos en la página oficial:

1. Crear cuenta gratuita en https://www.snowflake.com/en/ (30 días trial)
2. No descargas nada - es 100% en la nube
3. Obtienes credenciales (account, user, password) para usar en el script Python
4. Ejecuta el dimensional_model.sql desde la interfaz web de SnowFlake. Los pasos son:
    a. **Iniciar sesión** en SnowFlake web
    b. Ir a **Worksheets** (hojas de trabajo)
    c. Crear un SQL Worksheets
    d. Copiar y pegar el contenido de 04_dimensional_model.sql. Para su ejecución, seguir los pasos que se muestran abajo en **Ejecución**
5. El script creará:
    a. El DataWarehouse (FLEETLOGIX_WH)
    b. La base de datos (FLEETLOGIX_DW)
    c. El esquema (ANALYTICS)
    d. Todas las tablas dimensionales y de hechos

El modelo de estrella queda a continuación



Luego de haber creado la tabla de hechos y las tablas de dimensiones solo nos queda habilitar el time travel que es una función poderosa que permite a los usuarios acceder, consultar, clonar y restaurar versiones históricas de datos (tablas, esquemas, bases de datos) dentro de un período de retención definido.

Habiendo terminado la creación del DataWarehouse tenemos que subir los datos desde nuestro motor PostgreSQL a SnowFlake. Para esto usaremos el scrip etl_pipeline.py

ETL (Extract, Transform and Load) desde PostgreSQL a SnowFlake:

El objetivo de este pipeline es crear un ETL para extraer los datos de la

a. **Herramientas Utilizadas**

- **Python 3.9+**
- **PostgreSQL 13+**
- **SnowFlake**
- **Librerías de Python:**
    - **psycopg2**
    - **snowflake.connector**
    - **NumPy**
    - **pandas**
    - **Datetime**
    - **logging**
    - **schedule**
    - **json**
    - **typing**
    - **os**
    - **dotenv**

Para usar la librería dotenv es necesario las credenciales, puedes crear un archivo .env con tus credenciales y llamarlo credenciales.env puedes guardar tanto las de PostgreSQL como las de SnowFlake, las de SnowFlake son mas delicadas ya que este esta en la nube en cambio el motor de PostgreSQL esta de manera local la única manera de que la contraseña les sirva de algo es con acceso a tu computador.

El pipelines tiene diferentes funciones cada una con una función especifica que se unen en la clase FleetLogicETL vamos a verla una por una:

- extract_daily_data()
  Esta función es la encargada de extraer los datos diarios. Lo correcto seria en la consulta SQL usar current_day o current_timestamp ya que se quiere subir los datos diarios hayan o no hayan datos. Para este caso puntual use la ultima fecha que registra la base de datos para poder subir los datos a SnowFlake.
- Transform_data()
  Aqui se hace la segunda parte de un ETL la transformación de los datos pasar de una base de datos transaccional a un modelo estrella en snowflake con tablas de hechos y dimensiones. Es decir que tenemos que convertir las columnas en dimensiones y hechos para ello tenemos que calcular las métricas como delivery_time_minutes, delay_minutes, etc.

- load_dimensions()
  El tercer paso de un ETL es el Load es decir cargar los datos en la nube en este caso en SnowFlake, para ellos usamos un MERGE para las tablas que necesitamos ingresar datos en snowflake.
- load_fact()
  Luego de todo eso simplemente cargamos las tablas de hechos deliveries
- run_etl()
  Se ejecuta todo el pipeline

Se esperan un archivo que es el .log donde se guarda el proceso de ejecución del ETL.