Análisis del modelo proporcionado

Base de datos: **FleetLogix**

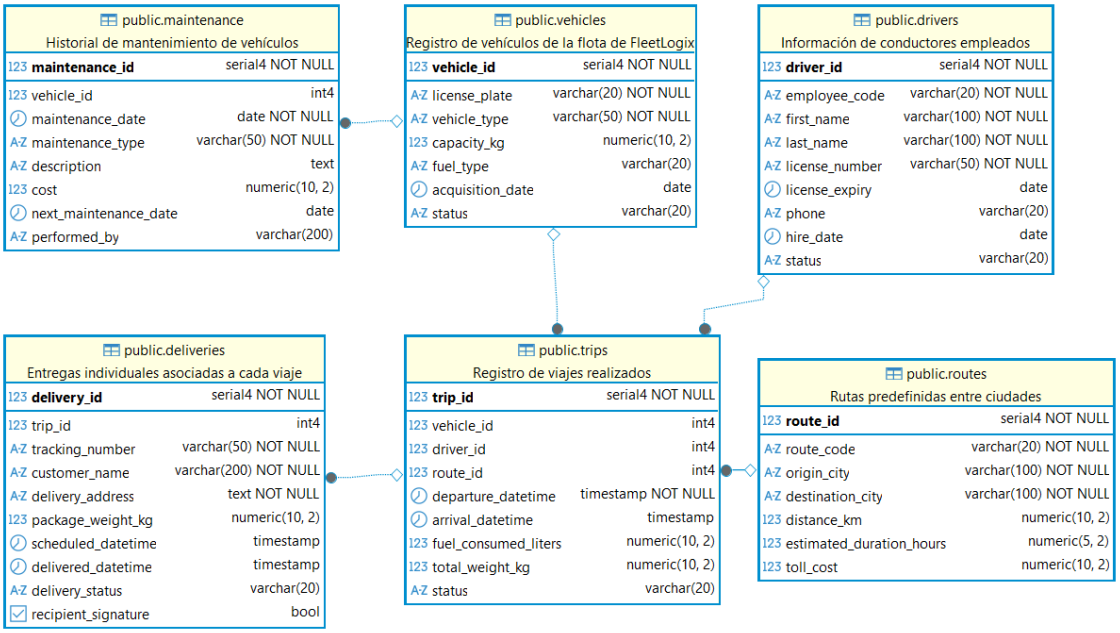
Descripción: representan la gestión de la flota de vehículos de la empresa

Cantidad de tablas del modelo: 6

Tablas Dim: vehicles, routes, drivers.

Tablas Fact: trips, deliveries, maintenance

Diagrama entidad relación (DER)



Relaciones entre las tablas

Relaciones de uno a muchos (1:N)

- **public.vehicles y public.maintenance:** Un vehículo puede tener muchos registros de mantenimiento, pero cada registro de mantenimiento

pertenece a un solo vehículo. La clave foránea es vehicle_id en public.maintenance.

- **public.vehicles y public.trips:** Un vehículo puede realizar muchos viajes, pero cada viaje es realizado por un solo vehículo. La clave foránea es vehicle_id en public.trips.
- **public.drivers y public.trips:** Un conductor puede realizar muchos viajes, pero cada viaje es realizado por un solo conductor. La clave foránea es driver_id en public.trips.
- **public.routes y public.trips:** Una ruta puede ser utilizada en muchos viajes, pero cada viaje se asocia a una sola ruta. La clave foránea es route_id en public.trips.

Restricciones (Constraints)

Restricciones NOT NULL

- public.maintenance: maintenance_date, maintenance_type, description.
- public.vehicles: license_plate, vehicle_type.
- public.drivers: employee_code, first_name, last_name, hire_date.
- public.trips: departure_datetime.
- public.deliveries: trip_id, tracking_number, customer_name, delivery_address.
- public.routes: route_code, origin_city, destination_city.

Claves primarias (Primary Keys)

- public.maintenance: maintenance_id
- public.vehicles: vehicle_id
- public.drivers: driver_id
- public.deliveries: trip_id (en este contexto, parece ser una clave primaria, aunque podría ser una clave foránea que referencia public.trips)
- public.trips: trip_id
- public.routes: route_id

Claves foráneas (Foreign Keys)

- public.maintenance: vehicle_id referencia a public.vehicles(vehicle_id).
- public.trips:
    - vehicle_id referencia a public.vehicles(vehicle_id).
    - driver_id referencia a public.drivers(driver_id).
    - route_id referencia a public.routes(route_id).
- public.deliveries: trip_id referencia a public.trips(trip_id).

Índices de la Base de Datos

- **Nombre del índice:** idx_trips_departure
    - **Tabla:** public.trips
    - **Columna:** departure_datetime
- **Nombre del índice:** idx_deliveries_status
    - **Tabla:** public.deliveries
    - **Columna:** delivery_status
- **Nombre del índice:** idx_vehicles_status
    - **Tabla:** public.vehicles
    - **Columna:** status

Justificación de tablas

Explique claramente el método generate_trips() y su auxiliar *get*_hourly_distribution

- **get_hourly_distribution()**

Esta función define **a qué hora es más probable que inicie un viaje.**

Devuelve un array de probabilidades para cada hora del día (0–23). Se usa para que cada hora tenga una probabilidad de salida realista de los viajes.

- Hay **más probabilidad entre las 6:00 y las 20:00** (franja laboral).
- Horas de madrugada tienen casi cero probabilidad. 8:00 y 9:00 tienen la probabilidad máxima

- Se normaliza para que la suma sea exactamente 1 (no es necesario pero es una buena práctica para distribuciones de probabilidad).
- Ejemplo de uso en *generate_trips:*

```
hourly_probs = get_hourly_distribution()

chosen_hours = np.random.choice(np.arange(24), size=n, p=hourly_probs)
```

generate_trips(): este método genera viajes coherentes, con fechas realistas, retrasos aleatorios, consumos, pesos y estado; y los inserta en la base de forma eficiente

Genera e inserta **N viajes (por defecto 100.000)** en la tabla trips, de forma **vectorizada** con NumPy y Pandas para ser eficiente.

Flujo paso a paso:

1. **Trae recursos activos**
    - Vehículos activos (vehicles)
    - Conductores activos (drivers)
    - Rutas (routes, con duración estimada)

Si alguno falta, se corta la función.

---

2. **Prepara el rango de fechas**
    - Los viajes se simulan en los **últimos 2 años** (730 días)
    - Se elige aleatoriamente un day_offset (0–730 días).

---

3. **Genera atributos para cada viaje (vectorizado con NumPy):**
    - Vehículo, Conductor, Ruta (con duración estimada), basado en la tablas ingestadas.
    - Día : aleatorio de últimos dos años a la fecha.
    - Hora: con probabilidad de selección diferencial según ger_hourly_distribution()

      Minutos aleatorios

---

4. **Calcula la hora de salida (departure_datetime)**
   Sumando día + hora + minuto al start_date (aleatorio entre fecha actual y 2 años atrás).

---

5. **Calcula la hora de llegada (arrival_datetime)**
   - Usa la duración de la ruta (durations), la cual fue ingestada en forma realista según los km aproximados de cada ruta.
   - Suma un retraso aleatorio de hasta **15 min por cada hora estimada**. Ejemplo: si la ruta dura 4 horas → puede tener entre 0 y 60 min extra.
   - Con ambas se calcula el tiempo de viaje, el cual se suma a la hora de salida para **asegurar consistencia temporal.**

---

6. **Genera atributos adicionales con perspectiva realista**
   - Consumo de combustible (fuel_consumed_liters, entre 50 y 500 litros), calculado en función de la distancia recorrida, realista para cada ruta
   - Peso transportado (total_weight_kg, entre 100 y 20.000 kg). Calculado aleatoriamente pero según la capacidad del vehículo (moto, camión grande, etc).
   - Estado (status con probabilidad: 85% *completed*, 5% *cancelled*, 10% *in_progress*).

---

7. **Crea un DataFrame Pandas** con todas las columnas de los viajes.

---

8. **Inserta en la base por lotes (batch_size=5000)**
   - Convierte cada chunk en lista de tuplas.
   - Ejecuta executemany → mucho más rápido que un loop individual.

---

9. **Loguea el resultado** en la tabla load_logs.

Posteriormente a la ingesta se comprueba que en ningún viaje la llegada es anterior a la partida:

select * **from** trips *t* **where** *t*.arrival_datetime < *t*.departure_datetime;

Mejoras implementadas en la generación de datos sintéticos con perspectiva realista

1. Rutas (generate_routes)

- Se extendió la lista de ciudades a **9 ciudades**, para evitar repetición de rutas al generar muchos trips.
- Se consideraron todas las combinaciones posibles de **ida y vuelta** para las rutas, asegurando variedad y cobertura de 50 rutas distintas.
- La duración de la ruta se calcula usando una **velocidad promedio aleatoria** (70-100 km/h).
- Se asigna un costo de peaje aproximado por ruta.

2. Trips (generate_trips)

- Uso de **Pandas y NumPy** para generar 100k viajes de forma eficiente.
- Se respetan **vehículos y conductores activos**, y rutas existentes.
- Fecha de salida (departure_datetime) se genera con:
    - Offset aleatorio de días dentro de los últimos 2 años.
    - Horas seleccionadas según **distribución horaria personalizada** (picos y valles de actividad).
    - Minutos aleatorios.
- Fecha de llegada (arrival_datetime) calculada como:
    - Suma de departure_datetime + duración estimada de ruta.
    - Se agregan minutos extra aleatorios: **hasta 15 min por cada hora estimada** para simular variaciones reales.

3. Deliveries (generate_deliveries)

- Cada trip puede tener **2 a 6 entregas**, ponderadas para que 4 entregas sean las más frecuentes.

- En lugar de que cada cliente sea único, se configuraron con peso diferente para que haya clientes con más o menos pedidos.
- **Scheduled datetime**:
    - Se asegura que quede **dentro del rango de tiempo estimado del viaje** (departure → arrival).
- **Delivered datetime**:
    - Se genera solo si el estado es "delivered".
    - Se añade un **pequeño retraso** (0-30 min) sobre scheduled_datetime.
    - Se mantiene None si la entrega falla o está pendiente.
- Se vectorizó la creación de entregas usando **Pandas** para eficiencia y luego se insertan en batch.
- Se manejan correctamente valores nulos (NaT → None) para compatibilidad con PostgreSQL.

4. General

- Registro de logs de inserción (log_load) para **vehículos, conductores, rutas, trips y deliveries**.
- Uso de **batch insert** para optimizar performance en tablas grandes.
- Coherencia temporal entre departure, arrival, scheduled_datetime y delivered_datetime.
- Estado de las entregas ponderado para reflejar la realidad: mayoría completadas, algunas pendientes o fallidas.