# Práctica 1: Diseño e Implementación de un *Data Mart*

**Almacenes y Minería de Datos**

Pedro Allué Tamargo (758267) Cristina Oriol García (755922)
Alejandro Paricio García (761783)

2 de noviembre de 2020

# Índice

# 1. Data Mart para vuelos comerciales

## 1.1. Metodología de *Kimball* aplicada, diseño del *Data Mart*

Siguiendo la metodología de *Kimball* se va a diseñar un *Data Mart* para el análisis de la información de vuelos comerciales en Estados Unidos.
El primer paso en esta metodología consiste en la selección del proceso de negocio. En este caso el proceso de negocio se va a centrar en el estudio del tiempo total de los vuelos y sus retardos (si estos existiesen). Se estudiarán factores asociados a los vuelos como los aeropuertos de origen, destino y que días existe un peor rendimiento. Este *Data Mart* se centrará en los usuarios que tienen cabida en el proceso de toma de decisiones en el ámbito de la aviación comercial.

El siguiente paso es la elección del gránulo o nivel detalle de los datos. Se ha elegido un vuelo de un avión determinado en una fecha y horas de salida concretas. Utilizando e grano más fino posible conseguiremos que se centre en la información indivisible generada por el proceso de negocio para poder tener la flexibilidad necesaria para formular cualquier pregunta.

El tercer paso es la elección de las dimensiones primarias y secundarias. Las dimensiones primarias son las que son determinadas por el grano escogido para los hechos, por lo que, obviamente, van a ser Fecha (de salida), Hora (de salida), Avión. Un vuelo va a poderse identificar mediante su fecha y hora de salida y el avión encargado de realizar el vuelo en cuestión.
El resto de dimensiones van a ser secundarias y, por definición, tomarán un único valor por combinación de las dimensiones primarias. No aportan información directa acerca de la identificación del hecho "vuelo", y no formarán parte de la clave primaria de la tabla de hechos. Conforman este conjunto las dimensiones Aeropuerto (de origen y de destino), Operadora, Aerolínea, Fecha (de llegada), Hora (de llegada).

En el último de los pasos ha de refinarse qué se está midiendo e identificarse que hechos numéricos poblarán cada tupla de la tabla de hechos. Finalmente, se han incluido como hechos numéricos en la tabla de hechos "Vuelo" los siguientes:

- Tiempo real de Vuelo = hora de llegada real - hora de salida real. (minutos)
- Tiempo estimado de vuelo = hora de llegada estimada - hora salida estimada. (minutos)
- Tiempo de Retraso a la llegada = máximo entre la diferencia de la hora de llegada real y la hora de salida estimada y 0. (minutos)
- Tiempo total de vuelo = Hora de llegada real - hora de salida estimada. (minutos)
- Tiempo de retraso a la salida = Hora de salida real - hora de salida estimada. (minutos)
- Tiempo adelantado a la llegada = máximo entre la diferencia de la hora de llegada estimada y la hora de llegada real y 0. (minutos)

Entre los aspectos que dieron lugar a debate en el proceso anterior encontramos:

- El identificador del vuelo (*numeroVuelo*) es un número que se asigna en función de la aerolínea y la fecha [1]. Ello ha condicionado el no poder ser utilizada para determinar el avión de un vuelo determinado de manera unívoca, y por tanto no es una clave natural. Si que aparece como una clave natural el atributo matriculaAvion, que si es único para cada avión.
- La separación de las dimensiones Fecha y Hora es motivada por la posibilidad de que aparezca una explosión combinatoria. Por ejemplo, para 365 días y 24 horas al día, una única tabla podría llevar a un máximo de 365*24 tuplas, mientras que con dos, queda limitado a 365+24 tuplas.
- Finalmente, se ha considerado el *numeroVuelo* como una dimensión degenerada (*DD*), ya que correspondería a una dimensión secundaria adicional. Contiene información que puede llegar a ser relevante, pero no puede ser tratada como las demás ya que no contiene más atributos que el propio *númeroVuelo* y por lo tanto se corresponde con una dimensión degenerada (*DD*).

Por lo tanto, teniendo en cuenta todo lo anterior se ha creado el esquema en estrella de la Figura 2.

## 1.2. Alternativas consideradas

Una de las alternativas en el diseño del *data mart* que se consideraron atañe a las dimensiones con múltiples conexiones a la tabla de hechos en el esquema original. Motivados por aportar facilidad a los usuarios, que a la hora de llevar a cabo consultas que aprovechen ambas relaciones deberán renombrar la tabla de varias formas, se podría llevar a cabo la siguiente modificación, dividiendo conceptualmente la dimensión fecha en *FechaSalida* y *FechaLlegada* (cambio repetido no sólo para esa tabla, sino también para Hora y Aeropuerto).

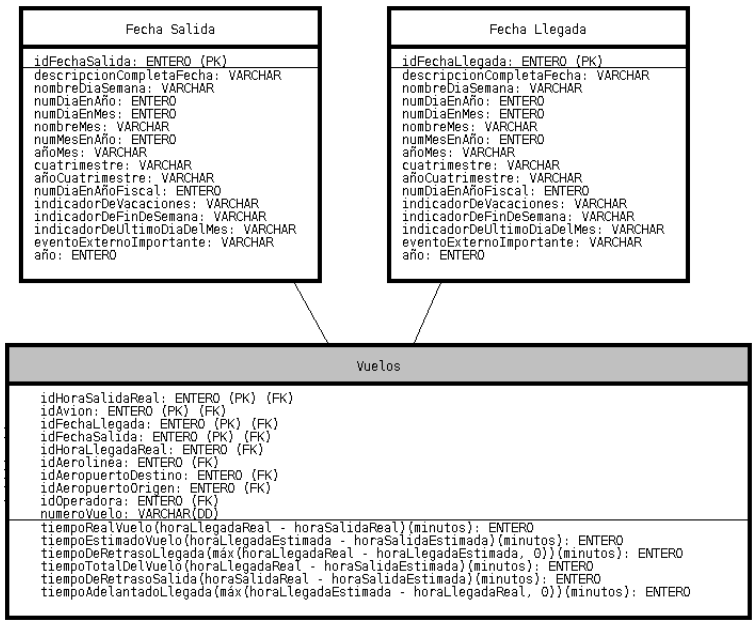


Figura 1: Modificaciones al esquema en estrella

A la hora de implementar el esquema, se llevaría a cabo como única tabla física sobre la que se definirían dos vistas, denominadas *FechaSalida* y *FechaLlegada*.
Como ejemplo ilustrativo del aporte que puede suponer a los usuarios, se despliega parte de una consulta que lo refleje:

- Implementación original: FROM Fecha AS FechaSalida, Fecha AS FechaOrigen.
- Implementación alternativa: FROM FechaSalida, FechaOrigen

Pese a ser una alternativa a valorar, se ha considerado que al haber un máximo de dos relaciones de una tabla de dimensión cualquiera a la tabla de hechos, no va a resultar problema alguno para el usuario el no disponer de las dos vistas, y que puede prescindirse de lo anterior.

## 1.3. Aspectos de implementación del esquema en estrella *SQL* y del poblado

Para generar el código *SQL* de este diagrama en estrella se va a utilizar la funcionalidad de la herramienta *DBDap* que extrae el *SQL* a partir del esquema en estrella. Se ha observado que una vez generado este código no se reconoce el tipo *VARCHAR* y se genera como *CHAR(-1)*. También se ha tenido que añadir manualmente las restricciones de *NOT NULL* a los atributos. Dando como resultado al código al código 6.1. Conjuntamente al script de creación se ha creado el de borrado de tablas.

Para crear los datos de población del almacén de datos se ha creado un *script Python* (código 6.3) que crea-se las sentencias de inserción *SQL* a partir de un conjunto de datos de aeropuertos, aviones, aerolíneas y operadoras.

## 2. Modificación del *Data Mart* de vuelos comerciales

Para este apartado se ha modificado el esquema en estrella del apartado anterior (Figura 2) para reflejar la información de los asientos. Originalmente se planteó como idea modificar el gránulo, ya que este no coincidía con el nivel de detalle de los asientos. No obstante, al incluir redundancia innecesaria sin aportar grandes ventajas para el análisis (necesitarías tener el tiempo de vuelo para cada asiento, que coincidiría con el del resto de asientos del vuelo, y no aportaría nada al análisis de los retardos y rendimiento), terminó descartándose. Se debía buscar otra solución, y se optó por las dimensiones puente (*bridge dimensions*) [2].

Una primera aproximación que se pensó fue guardar la información del asiento con la de los pasajeros. No obstante, el asiento no es único para todos los vuelos que realiza ese avión. Por lo tanto, se decidió utilizar el pasajero (ya que es lo que se pide almacenar) y la información del asiento se guardará con la información del anterior.

Se ha creado el esquema en estrella de la Figura 3 en el cual se puede observar que la tabla de hechos sigue siendo "vuelo" y que tiene una conexión con la tabla de dimensiones *Pasajeros Vuelo Bridge*. Esto implica que cada vuelo está conectado con la dimensión puente que relaciona el vuelo con el pasajero que ocupa un asiento en el mismo. La dimensión *Pasajero* está conectada con esta dimensión puente ya que todos los pasajeros pertenecen a un vuelo.

Para la generación del esquema *SQL* se ha utilizado de nuevo las opciones proporcionadas por *DBDap* La diferencia con respecto al apartado anterior ha sido la generación de dos tablas extra correspondientes a las nuevas dimensiones.

Con respecto a la introducción de datos de este esquema se han añadido manualmente los datos de las tablas nuevas sobre los datos de población generados por el *script* (código 6.3) del apartado anterior. El código *SQL* de creación de tablas se puede observar en el código 6.2.

# 3. Enunciado propuesto: Alquiler de coches en Europa

Se desea implantar un *data mart* para el análisis de información de alquiler de vehículos de una empresa con sucursales por Europa. Se quiere almacenar los siguientes datos de cada alquiler: sucursal de recogida y de depósito del vehículo (*nombre*, *código* y *dirección*), país de origen y destino, ciudad de origen y destino, modelo del vehículo (*marca*, *nombre*, *matrícula*) y la fecha de la recogida y la devolución del vehículo. Además de los anteriores, se desea guardar la fianza exigida al cliente y el seguro adquirido, el nombre de la compañía aseguradora, así como el límite económico del que se hace responsable la empresa y el importe del mismo, además del precio y el tiempo del alquiler (se alquilará por días) y las características del cliente (*nombre*, *edad*, *estado civil*, *género*, *situación laboral*).

El análisis se va a centrar en el rendimiento económico de los alquileres y el beneficio o pérdida obtenidos. Se desea saber con qué coches se obtiene un mayor beneficio y que tipo de clientes lo proporcionan, así como cuáles son los seguros cuya contratación más dinero reporta.

El grano se va a definir como un alquiler en una fecha de recogida de un vehículo determinado por parte de un cliente en una sucursal concreta. Ello va a dar lugar al esquema de la Figura 4.

En el diagrama en estrella desarrollado se puede observar que las dimensiones primarias son: Sucursal (Origen), Vehículo, Cliente y Fecha (recogida del vehículo en la sucursal de origen), es decir, aquellas que definen el grano.

En la tabla de hechos *Alquiler* se puede observar que se han añadido los atributos correspondientes para el estudio del rendimiento económico de las sucursales de alquiler de vehículos. Los atributos que se van a estudiar son:

- Días de duración del alquiler (días).
- Importe cargado al cliente. Sin tener en cuenta el importe de la fianza y teniendo en cuenta el importe del seguro (euros).
- Gasto de la empresa. Gasto al que hace frente la empresa de alquiler por motivos como el mantenimiento del vehículo, limpieza... (euros)
- Beneficio sin descontar impuestos. Beneficio bruto obtenido del importe cargado al usuario y los gastos de la empresa (euros).
- Beneficio descontando impuestos. Beneficio de la empresa tras aplicar impuestos (euros).
- Fianza. Importe abonado por el cliente (euros).
- Importe de seguro abonado por el cliente (euros).

De nuevo, se podría considerar aplicar la misma alternativa propuesta para el primer apartado de la práctica, lo que podría variar ligeramente la implementación del modelo.

# 4. Conclusiones

En esta práctica se ha hecho una primera aproximación a los almacenes de datos. Se han creado esquemas en estrella siguiendo la metodología de *Kimball* a partir de una breve descripción del problema. Se ha determinado el proceso de negocio para la creación de este *Data Mart* y se ha seleccionado el gránulo adecuado. Además, se han elegido las dimensiones de manera consecuente a las decisiones tomadas en los pasos anteriores y se ha plasmado el esquema conceptual mediante el uso de la herramienta *DBDap*. Por último, se ha traducido ese esquema a una implementación correcta y coherente, siendo valoradas distintas alternativas y corrigiéndose los errores derivados de la automatización del proceso de generar código *SQL*. Todo el proceso ha sido llevado a cambo habiendo analizado y teniendo en cuenta las necesidades de los usuarios.

La metodología de *Kimball* es un proceso que sigue 4 etapas para el diseño de *Data Mart*. Siguiendo esta metodología y pensando en el usuario final que deberá utilizar este sistema se han diseñado de una forma eficiente los *Data Marts* de esta práctica. EL único gran problema observado con este *Data Mart* y el procedimiento utilizado para su construcción atañe a la relación de este con futuras prácticas. En lugar de diseñar el esquema global y, tras ello, centrarnos en cada uno de los esquemas individuales a cada proceso de negocio, se ha acudido directamente al segundo. Esto puede llegar a crear mucho conflictos si es necesario integrarlo con los creados en futuras prácticas ya que pueden existir problemas con la integración tipos de datos y con dimensiones que deban estar conformados.

## 4.1. Control de esfuerzos

El trabajo se ha realizado en su parte mayoritaria en la sesión de prácticas. Se han concretado varias reuniones entre los integrantes del grupo, una antes de la sesión de prácticas y un dos después de la misma para poner en común el trabajo realizado por cada uno y ofrecer alternativas e ideas. Las herramientas utilizadas para la práctica han sido *DBDap*, *dBeaver* como herramienta de conexión a bases de datos y, finalmente, *Python* para la generación de los *scripts* necesarios. Cada uno de los integrantes han realizado los siguientes trabajos:

- Pedro Allué Tamargo: creación del esquema en estrella de vuelos, creación del esquema en estrella de alquileres de coches, redacción de la memoria. Tiempo invertido: 10 horas.
- Cristina Oriol García: creación del esquema en estrella de vuelos, creación del esquema en estrella de alquileres de coches, creación del esquema en estrella de vuelos contemplando los asientos, poblar almacén de datos de vuelos con asientos. Tiempo invertido: 11 horas.
- Alejandro Paricio García: creación del esquema en estrella de vuelos, creación del esquema en estrella de alquileres de coches, poblar almacén de datos de vuelos. Tiempo invertido: 13 horas.

# 5. Anexo 1: Figuras

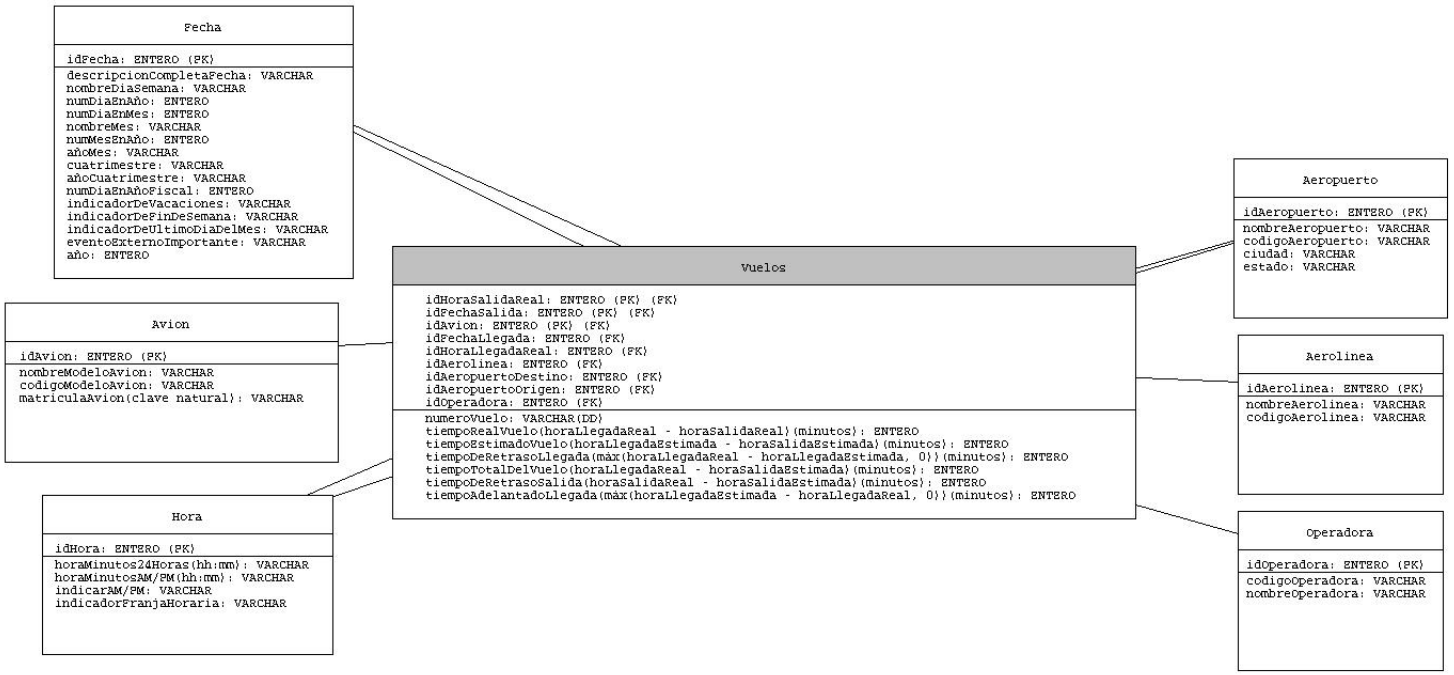


Figura 2: Esquema en estrella del *Data Mart* de vuelos comerciales

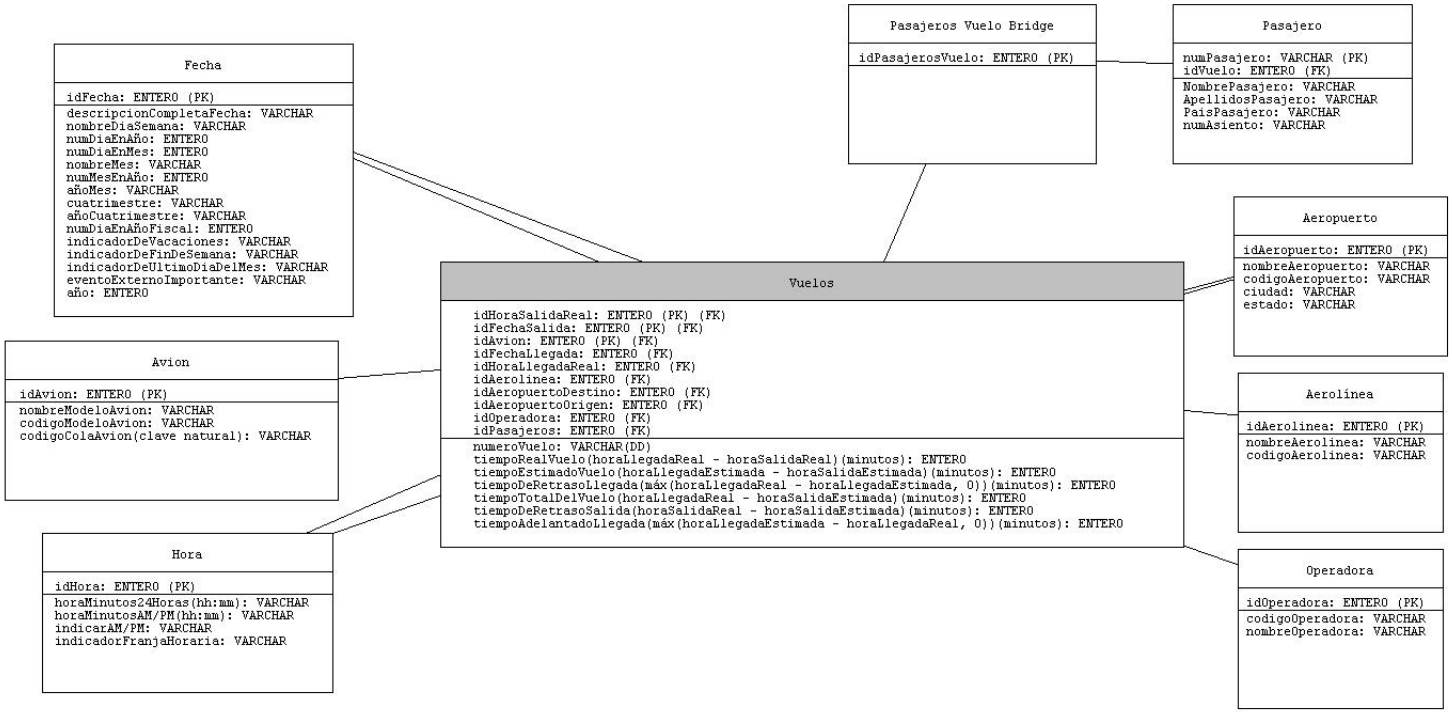


Figura 3: Esquema en estrella del *Data Mart* de vuelos comerciales teniendo en cuenta el asiento

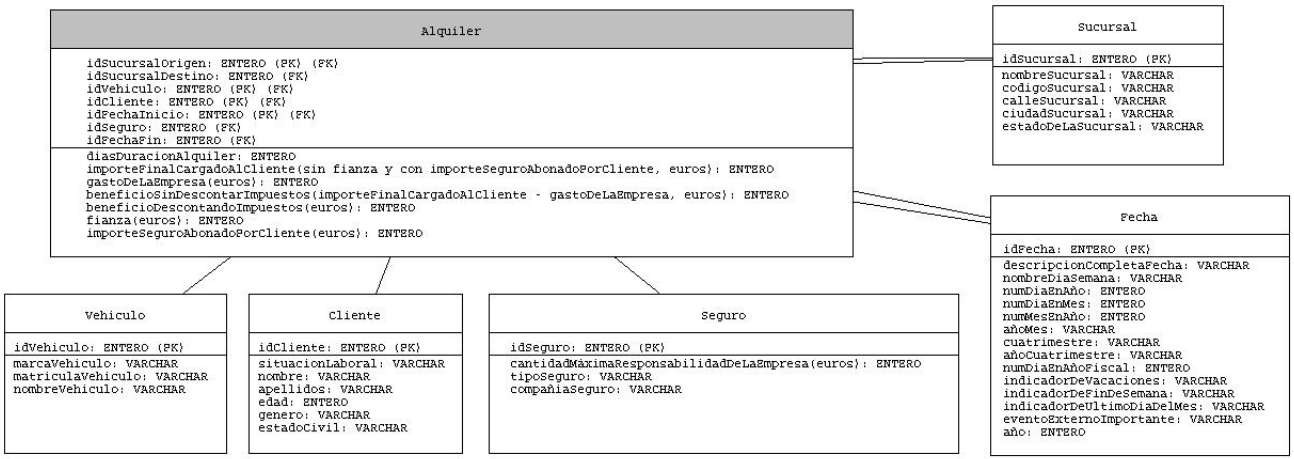


Figura 4: Esquema en estrella del enunciado propuesto

# 6. Anexo 2: Códigos

## 6.1. Script *SQL* para la creación de las tablas del apartado 1

```
CREATE TABLE Fecha
(
        idFecha                                 INT,
        descripcionCompletaFecha                VARCHAR(100) NOT NULL,
        nombreDiaSemana                         VARCHAR(100) NOT NULL,
        numDiaEnAo                              INT NOT NULL,
        numDiaEnMes                             INT NOT NULL,
        nombreMes                               VARCHAR(100) NOT NULL,
        numMesEnAo                              INT NOT NULL,
        aoMes                                   VARCHAR(100) NOT NULL,
        cuatrimestre                            VARCHAR(100) NOT NULL,
        aoCuatrimestre                                  VARCHAR(100) NOT NULL,
        numDiaEnAoFiscal                        INT NOT NULL,
        indicadorDeVacaciones                   VARCHAR(100) NOT NULL,
        indicadorDeFinDeSeamana                 VARCHAR(100) NOT NULL,
        indicadorDeUltimoDIaDelMes              VARCHAR(100) NOT NULL,
        eventoExternoImportante                 VARCHAR(100) NOT NULL,
        ao                                      INT NOT NULL,
        PRIMARY KEY(idFecha));


CREATE TABLE Avion
(
        idAvion                                 INT,
        nombreModeloAvion                       VARCHAR(100) NOT NULL,
        codigoModeloAvion                       VARCHAR(100) NOT NULL,
        matriculaAvion                          VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAvion));


CREATE TABLE Aeropuerto
(
        idAeropuerto                            INT,
        nombreAeropuerto                        VARCHAR(100) NOT NULL,
        codigoAeropuerto                        VARCHAR(100) NOT NULL,
        ciudad                                  VARCHAR(100) NOT NULL,
        estado                                  VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAeropuerto));


CREATE TABLE Aerolinea
(
        idAerolinea                             INT,
        nombreAerolinea                         VARCHAR(100) NOT NULL,
        codigoAerolinea                         VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAerolinea));


CREATE TABLE Operadora
(
        idOperadora                             INT,
        codigoOperadora                         VARCHAR(100) NOT NULL,
```

```
        nombreOperadora                         VARCHAR(100) NOT NULL,
        PRIMARY KEY(idOperadora));


CREATE TABLE Hora
(
        idHora                                  INT,
        horaMinutos24Horas                      VARCHAR(100) NOT NULL,
        horaMinutosAMPM                         VARCHAR(100) NOT NULL,
        indicarAMPM                             VARCHAR(100) NOT NULL,
        indicadorFranjaHoraria                  VARCHAR(100),
        PRIMARY KEY(idHora));


CREATE TABLE Vuelos
(
        idHoraSalidaReal                        INT,
        idFechaSalida                           INT,
        idAvion                                 INT,
        idFechaLlegada                          INT NOT NULL,
        idHoraLlegadaReal                       INT NOT NULL,
        idAerolinea                             INT NOT NULL,
        idAeropuertoDestino                     INT NOT NULL,
        idAeropuertoOrigen                      INT NOT NULL,
        idOperadora                             INT NOT NULL,
        numeroVuelo                             VARCHAR(100) NOT NULL,
        tiempoRealVuelo                         INT NOT NULL,
        tiempoEstimadoVuelo                     INT NOT NULL,
        tiempoDeRetrasoLlegada                  INT NOT NULL,
        tiempoTotalVuelo                        INT NOT NULL,
        tiempoDeRetrasoSalida                   INT NOT NULL,
        tiempoAdelantadoLlegada                 INT NOT NULL,
        PRIMARY KEY(idHoraSalidaReal, idFechaSalida, idAvion),
        FOREIGN KEY(idHoraSalidaReal) REFERENCES Hora (idHora),
        FOREIGN KEY(idFechaSalida) REFERENCES Fecha (idFecha),
        FOREIGN KEY(idAvion) REFERENCES Avion (idAvion),
        FOREIGN KEY(idFechaLlegada) REFERENCES Fecha (idFecha),
        FOREIGN KEY(idHoraLlegadaReal) REFERENCES Hora (idHora),
        FOREIGN KEY(idAerolinea) REFERENCES Aerolinea (idAerolinea),
        FOREIGN KEY(idAeropuertoDestino) REFERENCES Aeropuerto (idAeropuerto),
        FOREIGN KEY(idAeropuertoOrigen) REFERENCES Aeropuerto (idAeropuerto),
        FOREIGN KEY(idOperadora) REFERENCES Operadora (idOperadora));
```

### 6.2. Script *SQL* para la creación de las tablas del apartado 2

```
CREATE TABLE Fecha
(
        idFecha                         INT,
        descripcionCompletaFecha        VARCHAR(100) NOT NULL,
        nombreDiaSemana                 VARCHAR(100) NOT NULL,
        numDiaEnAño                     INT NOT NULL,
        numDiaEnMes                     INT NOT NULL,
        nombreMes                       VARCHAR(100) NOT NULL,
        numMesEnAño                     INT NOT NULL,
        añoMes                          VARCHAR(100) NOT NULL,
        cuatrimestre                    VARCHAR(100) NOT NULL,
        añoCuatrimestre                 VARCHAR(100) NOT NULL,
        numDiaEnAñoFiscal               INT NOT NULL,
        indicadorDeVacaciones           VARCHAR(100) NOT NULL,
        indicadorDeFinDeSeamana         VARCHAR(100) NOT NULL,
        indicadorDeUltimoDIaDelMes      VARCHAR(100) NOT NULL,
        eventoExternoImportante         VARCHAR(100) NOT NULL,
        año                             INT NOT NULL,
        PRIMARY KEY(idFecha));


CREATE TABLE Avion
(
        idAvion                         INT,
        nombreModeloAvion               VARCHAR(100) NOT NULL,
        codigoModeloAvion               VARCHAR(100) NOT NULL,
        matriculaAvion                  VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAvion));


CREATE TABLE Aeropuerto
(
        idAeropuerto                    INT,
        nombreAeropuerto                VARCHAR(100) NOT NULL,
        codigoAeropuerto                VARCHAR(100) NOT NULL,
        ciudad                          VARCHAR(100) NOT NULL,
        estado                          VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAeropuerto));


CREATE TABLE Aerolinea
(
        idAerolinea                     INT,
        nombreAerolinea                 VARCHAR(100) NOT NULL,
        codigoAerolinea                 VARCHAR(100) NOT NULL,
        PRIMARY KEY(idAerolinea));


CREATE TABLE Operadora
(
        idOperadora                     INT,
        codigoOperadora                 VARCHAR(100) NOT NULL,
        nombreOperadora                 VARCHAR(100) NOT NULL,
        PRIMARY KEY(idOperadora));
```

```
CREATE TABLE Hora
(
        idHora                                  INT,
        horaMinutos24Horas                      VARCHAR(100) NOT NULL,
        horaMinutosAMPM                         VARCHAR(100) NOT NULL,
        indicarAMPM                             VARCHAR(100) NOT NULL,
        indicadorFranjaHoraria                  VARCHAR(100),
        PRIMARY KEY(idHora));


CREATE TABLE Pasajeros_Vuelo_Bridge
(
        idPasajerosVuelo                        INT,
        PRIMARY KEY(idPasajerosVuelo));


CREATE TABLE Pasajero
(
        numPasajero                     varchar(100) ,
        idVuelo                 INT,
        NombrePasajero                  varchar(100)            NOT NULL,
        ApellidosPasajero                       varchar(100)            NOT NULL,
        PaisPasajero                    varchar(100)            NOT NULL,
        numAsiento                      varchar(100)            NOT NULL,
        PRIMARY KEY(numPasajero),
        FOREIGN KEY(idVuelo) REFERENCES Pasajeros_Vuelo_Bridge (idPasajerosVuelo));
CREATE TABLE Vuelos
(
        idHoraSalidaReal                        INT,
        idFechaSalida                           INT,
        idAvion                                 INT,
        idFechaLlegada                          INT NOT NULL,
        idHoraLlegadaReal                       INT NOT NULL,
        idAerolinea                             INT NOT NULL,
        idAeropuertoDestino                     INT NOT NULL,
        idAeropuertoOrigen                      INT NOT NULL,
        idOperadora                             INT NOT NULL,
        idPasajeros                             INT NOT NULL,
        numeroVuelo                             VARCHAR(100) NOT NULL,
        tiempoRealVuelo                         INT NOT NULL,
        tiempoEstimadoVuelo                     INT NOT NULL,
        tiempoDeRetrasoLlegada                  INT NOT NULL,
        tiempoTotalVuelo                        INT NOT NULL,
        tiempoDeRetrasoSalida                   INT NOT NULL,
        tiempoAdelantadoLlegada                 INT NOT NULL,
        PRIMARY KEY(idHoraSalidaReal, idFechaSalida, idAvion),
        FOREIGN KEY(idHoraSalidaReal) REFERENCES Hora (idHora),
        FOREIGN KEY(idFechaSalida) REFERENCES Fecha (idFecha),
        FOREIGN KEY(idAvion) REFERENCES Avion (idAvion),
        FOREIGN KEY(idFechaLlegada) REFERENCES Fecha (idFecha),
        FOREIGN KEY(idHoraLlegadaReal) REFERENCES Hora (idHora),
        FOREIGN KEY(idAerolinea) REFERENCES Aerolinea (idAerolinea),
        FOREIGN KEY(idAeropuertoDestino) REFERENCES Aeropuerto (idAeropuerto),
        FOREIGN KEY(idAeropuertoOrigen) REFERENCES Aeropuerto (idAeropuerto),
```

```
    FOREIGN KEY(idOperadora) REFERENCES Operadora (idOperadora),
    FOREIGN KEY(idPasajeros) REFERENCES Pasajeros_Vuelo_Bridge (
        idPasajerosVuelo));
```

### 6.3. Script de creación de elementos a insertar *Data Mart* de vuelos comerciales

```
import string
import random


DiccAvion = {}
# Nombre, c digo modelo, matr cula
for i in range(30):
        DiccAvion[i*10 + 0] = {(i*10 + 0, 'Boeing 787-9', '789', 'EC-A' + str(
            random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 1] = {(i*10 + 1, 'Boeing 787-8', '78P', 'EC-A' + str(
            random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 2] = {(i*10 + 2, 'Boeing 777-300', '773', 'EC-A' + str(
            random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 3] = {(i*10 + 3, 'Boeing 777-200', '772', 'EC-A' + str(
            random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 4] = {(i*10 + 4, 'Boeing 767-300', '76P', 'EC-A' + str(
            random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 5] = {(i*10 + 5, 'Boeing 737-800', '73L', 'EC-A' + str(
            random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 6] = {(i*10 + 6, 'Airbus A321', '321', 'EC-A' + str(random
            .randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 7] = {(i*10 + 7, 'Airbus A320', '32G', 'EC-A' + str(random
            .randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 8] = {(i*10 + 8, 'Bombardier DHC8-Q400', 'Q4A', 'EC-A' +
            str(random.randint(1,9999)) + str(random.randint(1,9999)))}
        DiccAvion[i*10 + 9] = {(i*10 + 9, 'Bombardier DHC8-Q200', 'Q82', 'EC-A' +
            str(random.randint(1,9999)) + str(random.randint(1,9999)))}

for i in range(300):
        print("INSERT INTO Avion VALUES " + ','.join(str(j) for j in DiccAvion[i])
            + ";")

DiccAeropuerto = {}
# Nombre aeropuerto, c digo aeropuerto, ciudad, estado
DiccAeropuerto[0] = {(0,'Aeropuerto Internacional Hartsfield-Jackson', 'ATL', '
    Atlanta', 'GA')}
DiccAeropuerto[1] = {(1,'Aeropuerto Internacional de Los  ngeles ', 'LAX', 'Los
    Angeles', 'CA')}
DiccAeropuerto[2] = {(2,'Aeropuerto Internacional OHare', 'ORD', 'Chicago', 'IL')}
DiccAeropuerto[3] = {(3,'Aeropuerto Internacional de Dallas-Fort Worth', 'DFW', '
    Dallas', 'TX')}
DiccAeropuerto[4] = {(4,'Aeropuerto Internacional de Denver', 'DEN', 'Denver', 'CO
    ')}
DiccAeropuerto[5] = {(5,'Aeropuerto Internacional John F. Kennedy', 'JFK', 'Nueva
    York', 'NY')}
DiccAeropuerto[6] = {(6,'Aeropuerto Internacional de San Francisco', 'SFO', 'San
    Francisco', 'CA')}
DiccAeropuerto[7] = {(7,'Aeropuerto Internacional de Seattle-Tacoma', 'SEA', '
    Seattle', 'WA')}
DiccAeropuerto[8] = {(8,'Aeropuerto Internacional McCarran', 'LAS', 'Las Vegas', '
    NV')}
DiccAeropuerto[9] = {(9,'Aeropuerto Internacional de Orlando', 'MCO', 'Orlando', '
    FL')}
```

```
DiccAeropuerto[10] = {(10,'Aeropuerto Internacional Libertad de Newark', 'EWR', '
    Newark', 'NJ')}
DiccAeropuerto[11] = {(11,'Aeropuerto Internacional de Charlotte-Douglas', 'CLT', '
    Charlotte', 'NC')}
DiccAeropuerto[12] = {(12,'Aeropuerto Internacional de Phoenix-Sky Harbor', 'PHX',
    'Phoenix', 'AZ')}
DiccAeropuerto[13] = {(13,'Aeropuerto Intercontinental George Bush', 'IAH', '
    Houston', 'TX')}
DiccAeropuerto[14] = {(14,'Aeropuerto Internacional de Miami', 'MIA', 'Miami', 'FL
    ')}
DiccAeropuerto[15] = {(15,'Aeropuerto Internacional Logan', 'BOS', 'Boston', 'MA')}
DiccAeropuerto[16] = {(16,'Aeropuerto Internacional de Mine polis-Saint Paul', '
    MSP', 'Mine polis', 'MN')}
DiccAeropuerto[17] = {(17,'Aeropuerto Internacional de Fort Lauderdale-Hollywood',
    'FLL', 'Fort Lauderdale', 'FL')}
DiccAeropuerto[18] = {(18,'Aeropuerto Internacional de Detroit', 'DTW', 'Detroit',
    'MI')}
DiccAeropuerto[19] = {(19,'Aeropuerto Internacional de Filadelfia', 'PHL', '
    Filadelfia', 'PA')}
DiccAeropuerto[20] = {(20,'Aeropuerto LaGuardia', 'LGA', 'Nueva York', 'NY')}
DiccAeropuerto[21] = {(21,'Aeropuerto Internacional de Baltimore-Washington', 'BWI
    ', 'Baltimore', 'MD')}
DiccAeropuerto[22] = {(22,'Aeropuerto Internacional de Salt Lake City', 'SLC', '
    Salt Lake City', 'UT')}
DiccAeropuerto[23] = {(23,'Aeropuerto Internacional de San Diego', 'SAN', 'San
    Diego', 'CA')}
DiccAeropuerto[24] = {(24,'Aeropuerto Internacional de Washington-Dulles', 'IAD', '
    Washington D.C.', 'VA')}
DiccAeropuerto[25] = {(25,'Aeropuerto Nacional Ronald Reagan de Washington', 'DCA',
     'Washington D.C.', 'VA')}
DiccAeropuerto[26] = {(26,'Aeropuerto Internacional Midway', 'MDW', 'Chicago', 'IL
    ')}
DiccAeropuerto[27] = {(27,'Aeropuerto Internacional de Tampa', 'TPA', 'Tampa', 'FL
    ')}
DiccAeropuerto[28] = {(28,'Aeropuerto Internacional de Portland', 'PDX', 'Portland
    ', 'OR')}
DiccAeropuerto[29] = {(29,'Aeropuerto Internacional Daniel K. Inouye', 'HNL', '
    Honolulu', 'HI')}

for i in range(30):
        print("INSERT INTO Aeropuerto VALUES " + ','.join(str(j) for j in
            DiccAeropuerto[i]) + ";")

DiccAerolinea = {}
# Nombre aerolinea, c digo aerolinea
DiccAerolinea[0] = {(0, 'Allegiant Air', 'G4')}
DiccAerolinea[1] = {(1, 'United Airlines', 'UA')}
DiccAerolinea[2] = {(2,'Delta Air Lines', 'DL')}
DiccAerolinea[3] = {(3, 'American Airlines', 'AA')}
DiccAerolinea[4] = {(4, 'Spirit Airlines', 'NK')}
DiccAerolinea[5] = {(5, 'PAKLOOK AIR INC.', 'K2')}
DiccAerolinea[6] = {(6, 'JetBlue Airways', 'B6')}
DiccAerolinea[7] = {(7, 'Frontier Airlines', 'F9')}
DiccAerolinea[8] = {(8, 'Alaska Airlines', 'AS')}
DiccAerolinea[9] = {(9, 'Southwest Airlines', 'WN')}
```

```
DiccOperadora = {}
# Nombre operadora, c digo operadora
DiccOperadora[0] = {(0, 'Allegiant Air', 'G4')}
DiccOperadora[1] = {(1, 'United Airlines', 'UA')}
DiccOperadora[2] = {(2,'Delta Air Lines', 'DL')}
DiccOperadora[3] = {(3, 'American Airlines', 'AA')}
DiccOperadora[4] = {(4, 'Spirit Airlines', 'NK')}
DiccOperadora[5] = {(5, 'PAKLOOK AIR INC.', 'K2')}
DiccOperadora[6] = {(6, 'JetBlue Airways', 'B6')}
DiccOperadora[7] = {(7, 'Frontier Airlines', 'F9')}
DiccOperadora[8] = {(8, 'Alaska Airlines', 'AS')}
DiccOperadora[9] = {(9, 'Southwest Airlines', 'WN')}

for i in range(10):
        print("INSERT INTO Aerolinea VALUES " + ','.join(str(j) for j in
            DiccAerolinea[i]) + ";")
        print("INSERT INTO Operadora VALUES " + ','.join(str(j) for j in
            DiccOperadora[i]) + ";")

DiccHora = {}
# horaMinutos24Horas(hh:mm), horaMinutosAM/PM(hh:mm), indicarAM/PM,
    indicadorFranjaHorario
for i in range(7000):
        hora = random.randint(12,23)
        minuto = random.randint(0,59)
        DiccHora[i] = (i, str("{0:0=2d}".format(hora) + ":" + "{0:0=2d}".format(
            minuto)), str("{0:0=2d}".format(hora-12) + ":" + "{0:0=2d}".format(
            minuto)), 'PM', 'CEST')
        hora = [j for j in DiccHora[i]]
        for j in range(1, len(hora)):
                hora[j] = "'" + hora[j] + "'"
        print("INSERT INTO Hora VALUES (" + ','.join(str(j) for j in hora) + ");")

# Se utiliza   nicamente   una fecha para concentrar los vuelos
DiccFecha = {}
Meses = ["nada", "enero", "febrero", "marzo", "abril", "mayo", "junio", "julio", "
    agosto", "septiembre", "octubre", "noviembre", "diciembre"]
diasMeses = ["nada", 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
Semana = ["nada", "lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "
    domingo"]
diasMes = 31
for i in range(7000):

        mesNumero = random.randint(1,12)
        mes = Meses[mesNumero]
        a o = random.randint(2005, 2019)
        diaEnMes = random.randint(1, diasMeses[mesNumero])
        diaSemanaNumero = random.randint(1,7)
        diaSemana = Semana[diaSemanaNumero]
        diaEnA o = (int)(diasMes*(mesNumero-1)) + diaEnMes

        esFinde = "'NO'"
        if(diaSemanaNumero == 6 or diaSemanaNumero == 7):
                esFinde = "'SI'"

        cuatrimestre = "Q4"
```

```
        if(mesNumero <= 3):
                cuatrimestre = "Q1"
        elif(mesNumero <= 6):
                cuatrimestre = "Q2"
        elif(mesNumero <= 9):
                cuatrimestre = "Q3"

        DiccFecha[i] = (i, "'" + str(diaEnMes) + " de " + mes + " de " + str(a o)
            + "'", "'" + diaSemana + "'", diaEnA o, diaEnMes, "'" + mes + "'",
            mesNumero, "'" + mes + " de " + str(a o) + "'", "'" + cuatrimestre +
            "'", "'" + cuatrimestre + " de " + str(a o) + "'", diaEnA o, "'NO'",
            esFinde, "'NO'", "'NO'", str(a o))

        print("INSERT INTO Fecha VALUES (" + ','.join(str(j) for j in DiccFecha[i])
            + ");")

# Devuelve cierto si y solo si pri es anterior a sec
def horaAnterior(pri, sec):
        priHora = pri[1].split(":")
        secHora = sec[1].split(":")
        if(int(priHora[0]) < int(secHora[0])):
                return True
        elif(int(priHora[0]) == int(secHora[0]) and int(priHora[1]) < int(secHora
            [1])):
                return True
        return False


for i in range(900):

        # Todos los vuelos son el d a 1 entre las 00 y las 12 pm, la resta es
            directa
        horaInit = random.randint(1,7000) - 1
        horaFin = random.randint(1,7000) - 1
        while(not horaAnterior(DiccHora[horaInit], DiccHora[horaFin])):
                horaInit = random.randint(1,7000) - 1
                horaFin = random.randint(1,7000) - 1

        horaInicialDefinitiva = DiccHora[horaInit]
        horaFinalDefinitiva = DiccHora[horaFin]
        minsHoraInitReal = horaInicialDefinitiva[1].split(":")
        minsHoraInitReal = int(minsHoraInitReal[0])*60 + int(minsHoraInitReal[1])
        minsHoraFinReal = horaFinalDefinitiva[1].split(":")
        minsHoraFinReal = int(minsHoraFinReal[0])*60 + int(minsHoraFinReal[1])
        tiempoRealVuelo = minsHoraFinReal - minsHoraInitReal

        retrasoSalida = 0
        retrasoLlegada = 0
        minsHoraInitEstimada = minsHoraInitReal
        minsHoraFinEstimada = minsHoraFinReal

        if(random.randint(1,9) %9 == 0):
                retrasoLlegada = random.randint(0,30)
                minsHoraFinEstimada = minsHoraFinReal - retrasoLlegada
                if(minsHoraFinEstimada - minsHoraInitEstimada <= 0):
                        minsHoraFinEstimada = minsHoraFinReal
```

```
                retrasoLlegada = 0

        if(random.randint(1,9) %9 == 0):
                retrasoSalida = random.randint(0,30)
                minsHoraInitEstimada = minsHoraInitReal - retrasoSalida
                if(minsHoraInitEstimada - minsHoraInitReal <= 0):
                        minsHoraInitEstimada = minsHoraInitReal
                        retrasoSalida = 0

        tiempoEstimadoVuelo = minsHoraFinEstimada - minsHoraInitEstimada
        tiempoTotalDelVuelo = minsHoraFinReal - minsHoraInitEstimada
        tiempoAdelantadoLlegada = max([minsHoraFinEstimada - minsHoraFinReal, 0])

        """print(horaInicialDefinitiva)
        print(horaFinalDefinitiva)
        print("Minutos hora inicial real: " + str(minsHoraInitReal))
        print("Minutos hora inicial estimada: " + str(minsHoraInitEstimada))
        print("Minutos hora final real: " + str(minsHoraFinReal))
        print("Minutos hora final estimada: " + str(minsHoraFinEstimada))
        print("Minutos retraso salida: " + str(retrasoSalida))
        print("Minutos retraso llegada: " + str(retrasoLlegada))
        print("Minutos total vuelo: " + str(tiempoTotalDelVuelo))
        print("Minutos estimado vuelo: " + str(tiempoEstimadoVuelo))
        print("Minutos adelantado llegada: " + str(tiempoAdelantadoLlegada))
        print("Minutos real vuelo: " + str(tiempoRealVuelo))"""


        print("INSERT INTO Vuelos VALUES (" + str(horaInit) + ",0," + str(random.
            randint(0,299)) + ",0," +  str(horaFin) + "," + str(random.randint(0,9))
             + "," + str(random.randint(0,29)) + "," + str(random.randint(0,29)) +
            "," + str(random.randint(0,9)) + "," + "'" + str(random.randint(100000,
            999999)) + "'" + "," + str(tiempoRealVuelo) + "," + str(
            tiempoEstimadoVuelo) + "," + str(retrasoLlegada) + "," + str(
            tiempoTotalDelVuelo) + "," + str(retrasoSalida) + "," + str(
            tiempoAdelantadoLlegada) + ");")
```

# Referencias

[1] Wikipedia. Número de vuelo. https://es.wikipedia.org/wiki/N%C3%BAmero_de_vuelo, Marzo 2020. Consultado el 19/10/2020.

[2] JIM MCHUGH. Data warehouse design techniques – bridge tables. https://www.nuwavesolutions.com/bridge-tables/, Marzo 2017. Consultado el 28/10/2020.