# Bases de datos - Práctica 3

Curso 2022

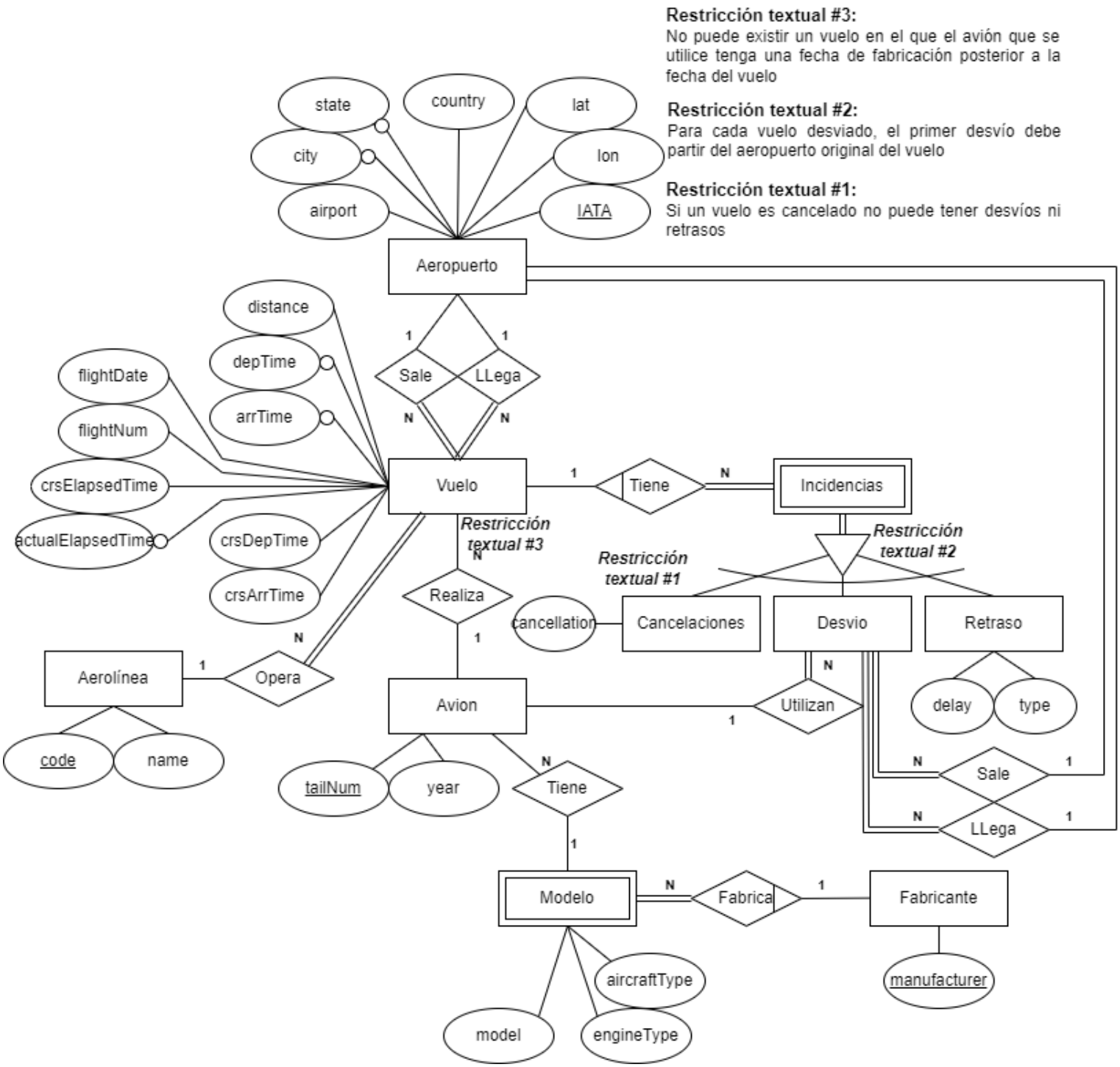
29/05/2022


## Participantes

Héctor Toral Pallás - 798095@unizar.es
Francisco Javier Pizarro Martinez - 821259@unizar.es
Pablo López Mosqueda - 779739@unizar.es

# Parte 1 - Creación de la base de datos

## 1.1 - Modelo Entidad/Relación



En está práctica se ha de realizar una base de datos para gestionar y almacenar información acerca de los vuelos que han tenido lugar en los aeropuertos de estados unidos, para ello, se ha planteado el siguiente modelo E/R:

Restricciones del modelo:

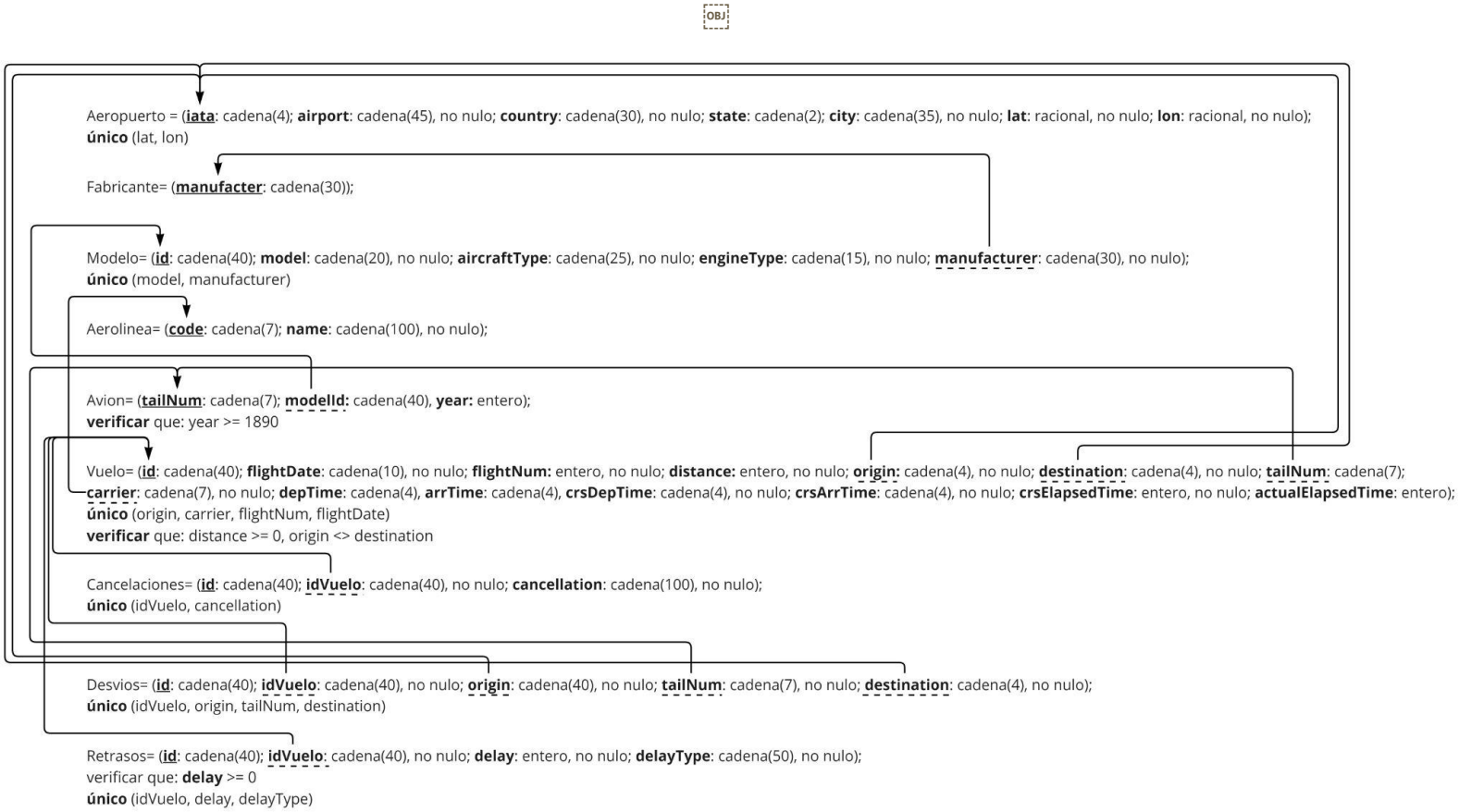
- Si un vuelo es cancelado no puede tener desvíos.
- En el primer desvío de un vuelo siempre debe ser siempre realizado por el avión que iba a realizar dicho vuelo
- La fecha del avión usado durante un vuelo debe ser posterior a la fabricación del avión utilizado
- La fecha de fabricación de un avión debe ser superior a la invención de los primeros aviones

Alternativas a nuestro modelo:

- En vez de tener una tablas incidencias débil de vuelo habíamos pensado tener 2 relaciones débiles y un atributo cancelado finalmente se descarto ya que finalmente se tradujo la especialización con 3 tablas (cancelaciones, desvíos, retrasos) y el atributo de cancelado no era necesario para ninguna de las consultas por lo que no supondría ninguna mejora de rendimiento según los requisitos actuales.

- Otra alternativa era tener la entidad de aerolíneas con una relación 1:N con la tabla de aviones pero esto no nos sacaría todos los datos correctos a la hora de realizar consultas, además de que perjudica el rendimiento a la hora de tener que hacer un join de más al sacarlo de la relación con avión, pudiendo hacerlo directamente desde la tabla de vuelo

## 1.2 - Modelo Relacional




El modelo se encuentra en la 1ªFN debido a que no tenemos atributos multivaluados
El modelo se encuentra en la 2ªFN debido a que no hay dependencias funcionales dentro de los atributos de claves compuestas puesto que no hay.
El modelo se encuentra en la 3ª FN y 3ªFNBC ya que todos los atributos dependen siempre de la clave.

## 1.3 - Traducción a SQL

```
CREATE TABLE aeropuerto (
  iata    VARCHAR2(4),
  airport VARCHAR2(45)  NOT NULL,
  country VARCHAR2(30)  NOT NULL,
  state   VARCHAR2(2),
  city    VARCHAR2(35),
  lat     NUMBER        NOT NULL,
  lon     NUMBER        NOT NULL,
  CONSTRAINT pk_aeropuerto  PRIMARY KEY (iata),
  UNIQUE (lat, lon)
);

CREATE TABLE fabricante (
  manufacturer  VARCHAR2(30),
  CONSTRAINT pk_fabricante  PRIMARY KEY (manufacturer)
);

CREATE TABLE modelo (
  id            VARCHAR2(40),
  model         VARCHAR2(20) NOT NULL,
  aircraftType  VARCHAR2(25) NOT NULL,
  engineType    VARCHAR2(15) NOT NULL,
  manufacturer  VARCHAR2(30) NOT NULL,
  CONSTRAINT pk_modelo  PRIMARY KEY (id),
  CONSTRAINT fk_modelo_manufacturer FOREIGN KEY (manufacturer)  REFERENCES
fabricante(manufacturer),
  UNIQUE (model, manufacturer)
);

CREATE TABLE aerolinea (
  code  VARCHAR2(7),
  name  VARCHAR2(100) NOT NULL,
  CONSTRAINT pk_aerolinea  PRIMARY KEY (code)
);
CREATE TABLE avion (
  tailNum VARCHAR2(7),
  modelId VARCHAR2(40),
  year    NUMBER,
  CONSTRAINT pk_avion  PRIMARY KEY (tailNum),
  CONSTRAINT fk_avion_modelId  FOREIGN KEY (modelId) REFERENCES modelo(id),
  CONSTRAINT ck_avion_year CHECK (year >= 1890)
);
```

```
CREATE TABLE vuelo (
  id                VARCHAR2(40),
  flightDate        VARCHAR2(10)  NOT NULL,
  flightNum         NUMBER        NOT NULL,
  distance          NUMBER        NOT NULL,
  origin            VARCHAR2(4)   NOT NULL,
  destination       VARCHAR2(4)   NOT NULL,
  tailNum           VARCHAR2(7),
  carrier           VARCHAR2(7)   NOT NULL,
  depTime           VARCHAR2(4),
  arrTime           VARCHAR2(4),
  crsDepTime        VARCHAR2(4)   NOT NULL,
  crsArrTime        VARCHAR2(4)   NOT NULL,
  crsElapsedTime    NUMBER        NOT NULL,
  actualElapsedTime NUMBER,
  CONSTRAINT pk_vuelo             PRIMARY KEY (id),
  CONSTRAINT fk_vuelo_origin      FOREIGN KEY (origin)      REFERENCES
aeropuerto(iata),
  CONSTRAINT fk_vuelo_destination FOREIGN KEY (destination) REFERENCES
aeropuerto(iata),
  CONSTRAINT fk_vuelo_tailNum     FOREIGN KEY (tailNum)     REFERENCES
avion(tailNum),
  CONSTRAINT fk_vuelo_carrier     FOREIGN KEY (carrier)     REFERENCES
aerolinea(code),
  UNIQUE (origin, carrier, flightNum, flightDate),
  CONSTRAINT ck_vuelo_distance           CHECK (distance >= 0),
  CONSTRAINT ck_vuelo_origin_destination  CHECK (origin <> destination)
);

CREATE TABLE cancelaciones (
  id            VARCHAR2(40),
  idVuelo       VARCHAR2(40)  NOT NULL,
  cancellation  VARCHAR2(100) NOT NULL,
  CONSTRAINT pk_cancelaciones         PRIMARY KEY (id, idVuelo),
  CONSTRAINT fk_cancelaciones_idVuelo FOREIGN KEY (idVuelo)  REFERENCES
vuelo(id),
  UNIQUE (id, idVuelo)
);
```

```
CREATE TABLE desvios (
  id          VARCHAR2(40),
  idVuelo     VARCHAR2(40)  NOT NULL,
  origin      VARCHAR2(4)   NOT NULL,
  tailNum     VARCHAR2(7)   NOT NULL,
  destination VARCHAR2(4)   NOT NULL,
  CONSTRAINT pk_desvios               PRIMARY KEY (id, idVuelo),
  CONSTRAINT fk_desvios_idVuelo       FOREIGN KEY (idVuelo)     REFERENCES
vuelo(id),
  CONSTRAINT fk_desvios_origin        FOREIGN KEY (origin)      REFERENCES
aeropuerto(iata),
  CONSTRAINT fk_desvios_tailnum       FOREIGN KEY (tailNum)     REFERENCES
avion(tailNum),
  CONSTRAINT fk_desvios_destination   FOREIGN KEY (destination) REFERENCES
aeropuerto(iata),
  UNIQUE (id, idVuelo)
);

CREATE TABLE retrasos (
  id        VARCHAR2(40),
  idVuelo   VARCHAR2(40)  NOT NULL,
  delay     NUMBER        NOT NULL,
  delayType VARCHAR2(50)  NOT NULL,
  CONSTRAINT pk_retrasos          PRIMARY KEY (id, idVuelo),
  CONSTRAINT fk_retrasos_idvuelo  FOREIGN KEY (idVuelo)  REFERENCES vuelo(id),
  CONSTRAINT ck_retrasos_delay    CHECK (delay >= 0),
  UNIQUE (id, idVuelo)
);
```

# Parte 2 - Introducción de datos y ejecución de consultas

## 2.1 - Población de la Base de Datos

Para empezar con la parte de población se empleó la aplicación MySQL Workbench para extraer la información de la base de datos proporcionada, con la información extraída de las distintas consultas realizadas, se exportaba directamente a formato sql insert.

Después simplemente debíamos modificar los nombres de las tablas y de los valores insertados desde Visual Studio Code empleando el atajo de teclado “ctrl + h”.

Adicionalmente a diferencia de la base de datos anterior dadas las diferencias entre el modelo proporcionado y el diseñado hemos tenido que usar scripts auxiliares creados en python para poder generar los inserts de algunas tablas.

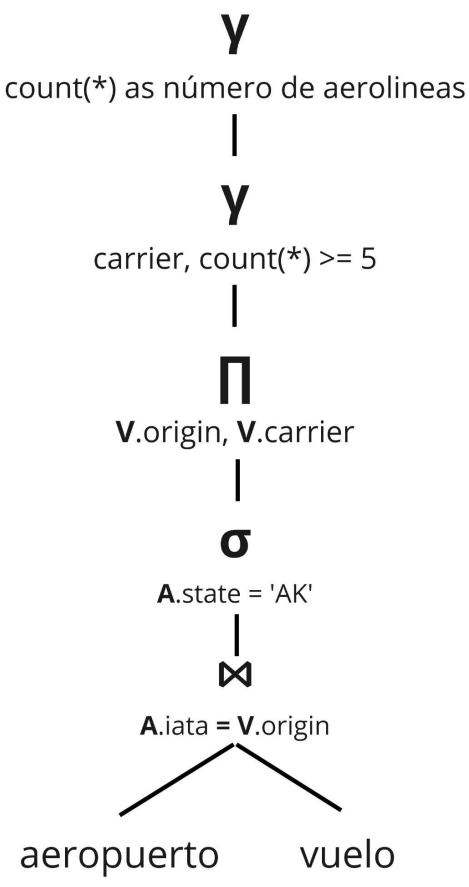
## 2.2 - Consultas de la Base de Datos

1. Calcular el número de compañías que operan en al menos cinco aeropuertos de Alaska.

```
select count(*) as "Número de aerolineas"
from (
    select carrier
    from (
        select distinct V.origin, carrier
        from aeropuerto A, vuelo V
        where A.iata = V.origin and A.state = 'AK'
    ) A
    group by carrier
    having count(*) >= 5
) N;
```

RESULTADOS:

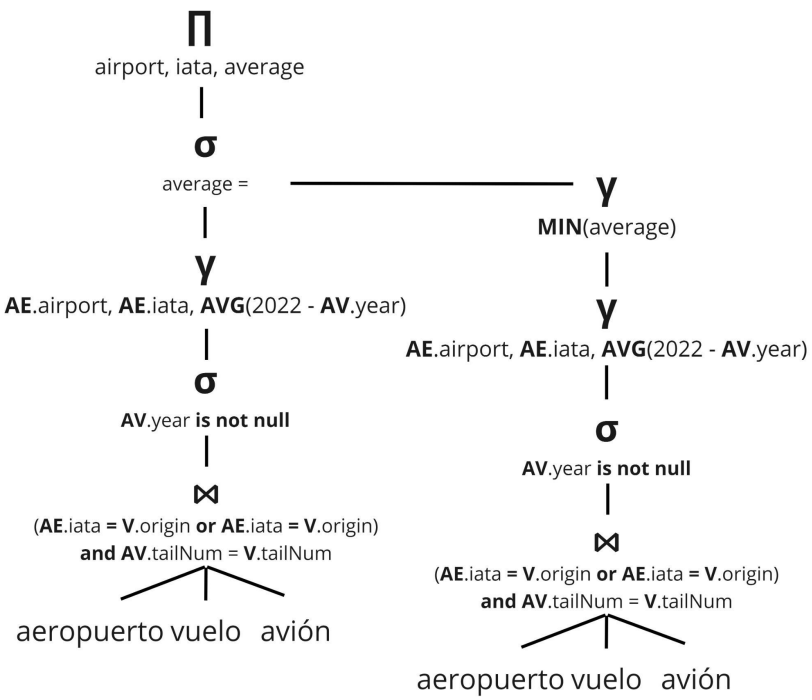
El número de compañías que cumplen esas condiciones es de **1**



2. Obtener el aeropuerto en el que operan los aviones más modernos (es decir, con menor media de edad). Obtener tanto el nombre y el código del aeropuerto como la media de edad de los aviones que operan en él.

```
select S.airport, S.iata, FLOOR(S.average)
from edadMediaAeropuertos S
where S.average = (
    select MIN(average) as minimo
    from edadMediaAeropuertos A
);

create view edadMediaAeropuertos as
select AE.airport, AE.iata, AVG(2022 - AV.year) as average
from aeropuerto AE, vuelo V, avion AV
where (AE.iata = V.origin or AE.iata = V.origin) and AV.tailNum = V.tailNum and
    AV.year is not null
group by AE.airport, AE.iata;
```




.

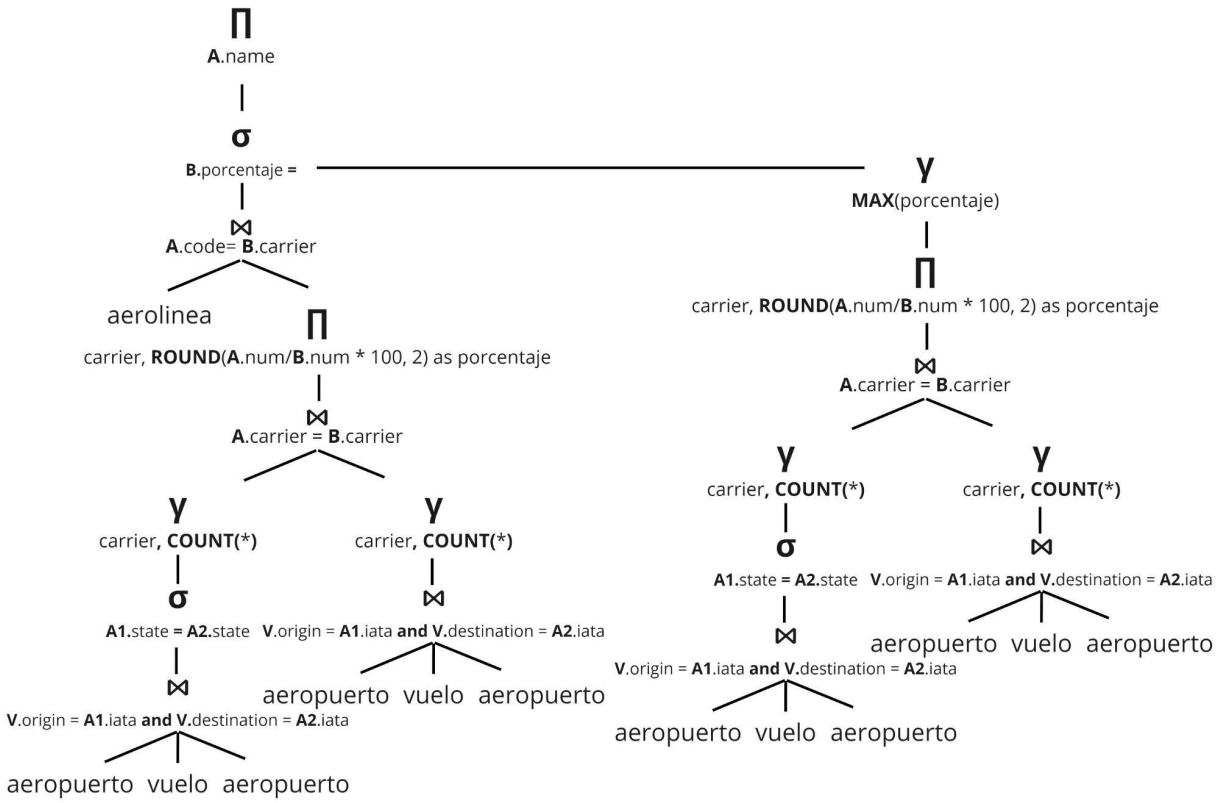
RESULTADOS

El aeropuerto es **Yampa Valley**, su código de aeropuerto es **HDN** y la media de edad de los aviones que operan en él es de **17** aproximadamente ya que se realizó un redondeo.

3. Obtener la compañía (o compañías, en caso de empate) con el mayor porcentaje de vuelos que despegan y aterrizan en un mismo estado.

```
create view porcentajeSalidasLLegadasMismoEstado as
select A.carrier, ROUND((A.num / B.num * 100), 2) as porcentaje
from (
    select S.carrier, COUNT(*) as num
    from (
        select V.carrier, A1.state as origin, A2.state as destination
        from vuelo V, aeropuerto A1, aeropuerto A2
        where V.origin = A1.iata and V.destination = A2.iata and A1.state =
A2.state
    ) S
    group by S.carrier
) A, (
    select T.carrier, COUNT(*) as num
    from (
        select V.carrier, A1.state as origin, A2.state as destination
        from vuelo V, aeropuerto A1, aeropuerto A2
        where V.origin = A1.iata and V.destination = A2.iata
    ) T
    group by T.carrier
) B
where A.carrier = B.carrier

select A.name, B.porcentaje
from aerolinea A, porcentajeSalidasLLegadasMismoEstado B
where A.code = B.carrier and B.porcentaje = (
    select MAX(porcentaje)
    from porcentajeSalidasLLegadasMismoEstado A
);
```




## RESULTADOS

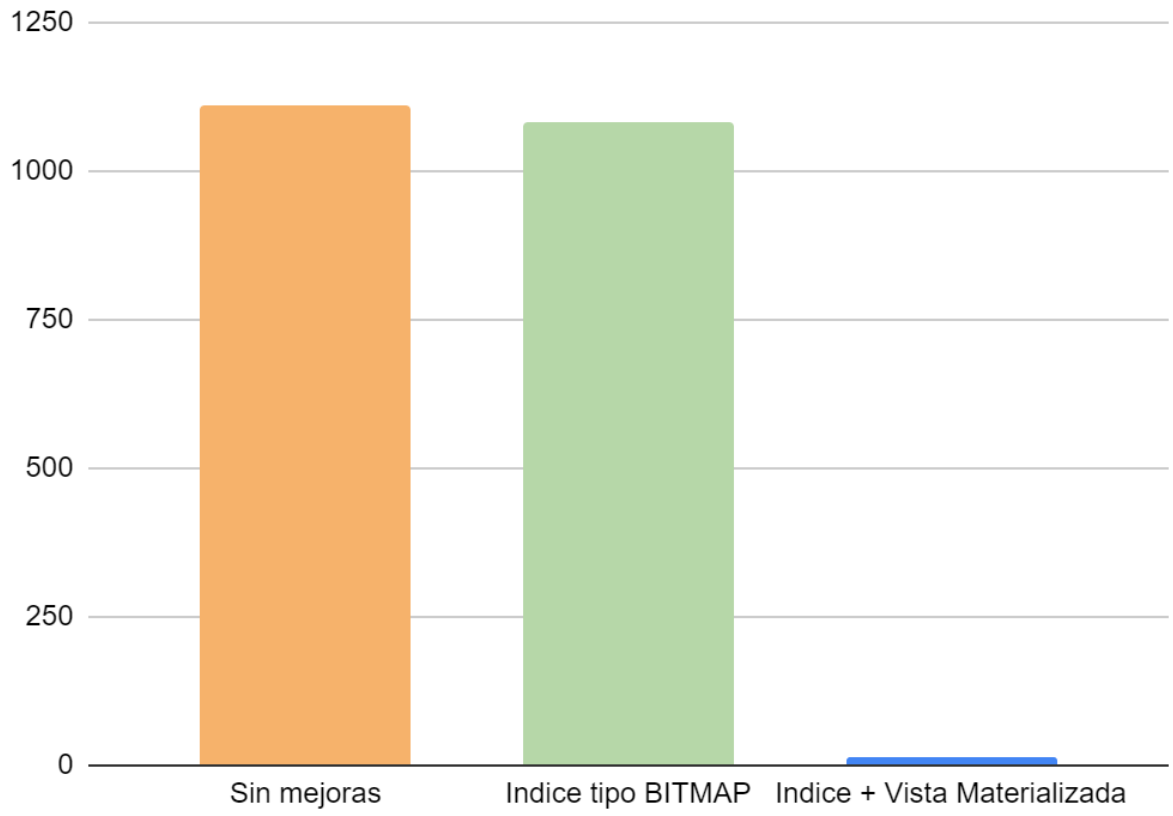
La aerolínea que cumple la query del enunciado es **Hawaiian Airlanes Inc.**

# Parte 3 - Evaluación de rendimiento y triggers

## 3.1 - Evaluación de rendimiento

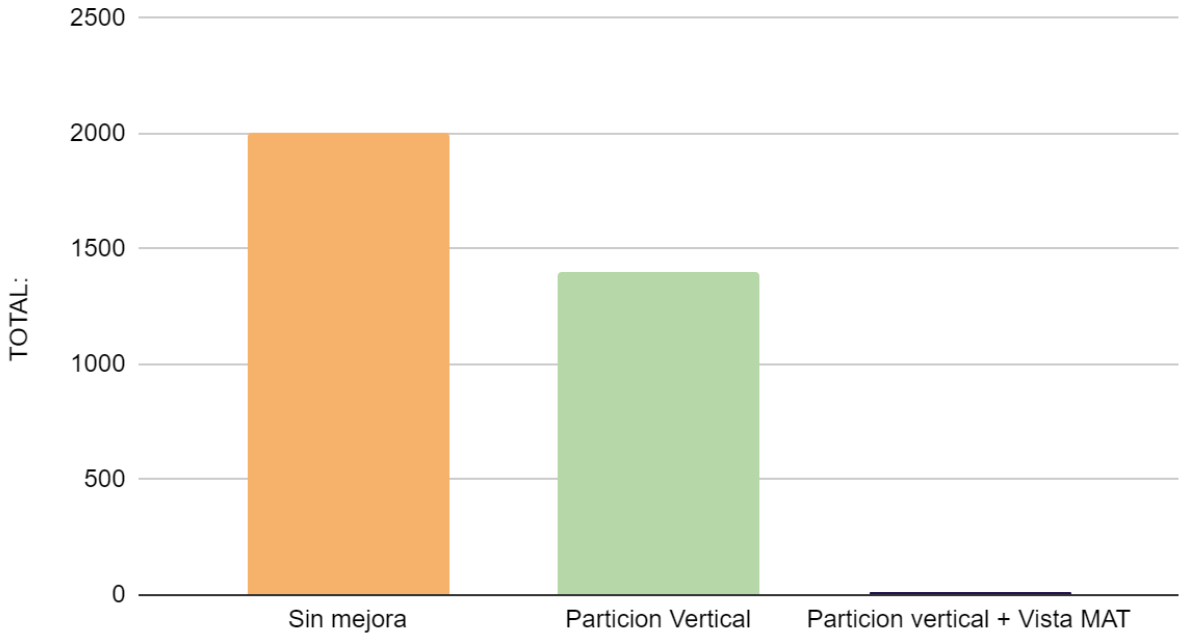
En la primera consulta el coste fue de **1113**

Para intentar mejorar el resultado obtenido se ha decidido, primeramente, añadir un índice de tipo BITMAP el cual no mejoró mucho los resultados, bajando el coste hasta **1081**. Después, recurrimos a una vista materializada que impulsó la mejora reduciendo el coste hasta **15**

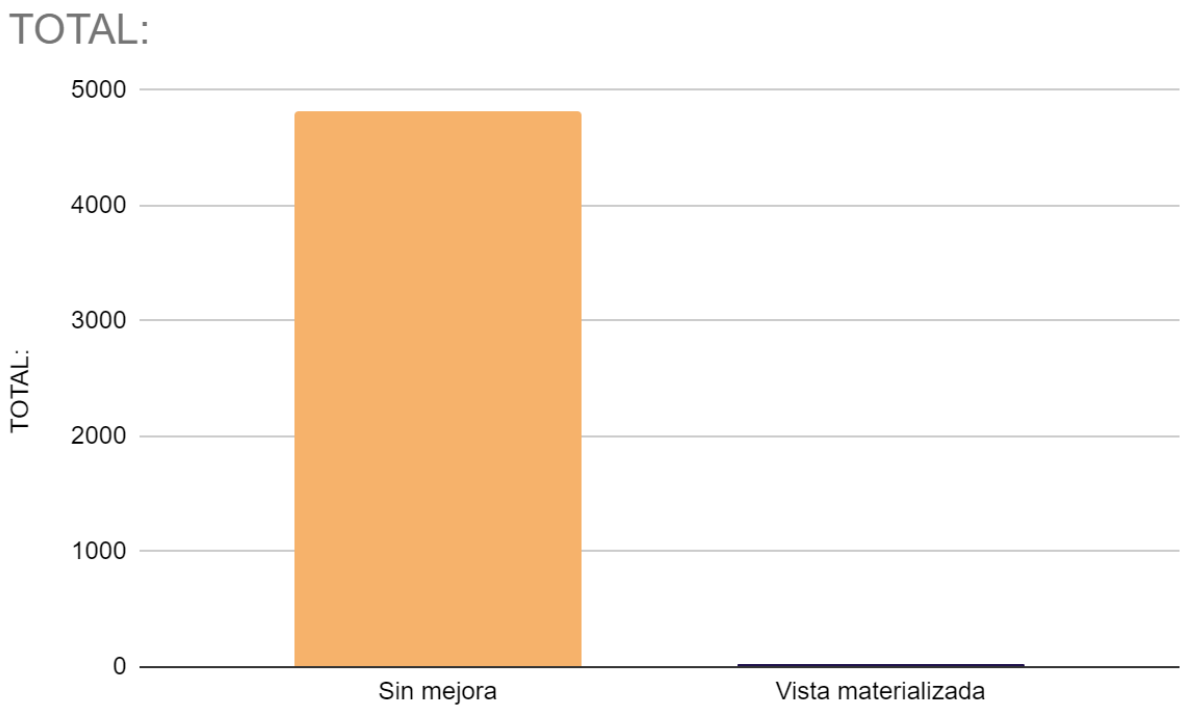


En la segunda consulta el coste era de **2008**, para intentar rebajar su coste se realizó primero un particionado vertical sobre la tabla de vuelos, mejorando el coste a **1399** adicionalmente se realizó después una vista materializada, dejando el coste final en **12**

TOTAL:



En la tercera consulta el coste resultó ser de **4814**, lo único que realizamos fue una vista materializada que redujo el coste hasta **27**. El motivo de la vista fue tener guardado el resultado de una subconsulta que se repite 2 veces dentro de la consulta principal, dicha subconsulta permitía obtener los datos para calcular el porcentaje con un acceso completo a la tabla.



## 3.2 - Triggers

El primer trigger aporta  la restricción de que si un vuelo es cancelado no puede tener desvíos.

- Trigger 1:

```
-- TRIGGER 1:
CREATE OR REPLACE TRIGGER EXC_DESVIOS
BEFORE INSERT ON desvios
FOR EACH ROW
DECLARE
    flag NUMBER;
BEGIN
    SELECT COUNT(*) INTO flag
    FROM cancelaciones
    WHERE idVuelo = :NEW.idVuelo;

    IF flag >= 1 THEN
        RAISE_APPLICATION_ERROR (-20000, 'No puede haber desvío puesto que
el vuelo ha sido cancelado');
    END IF;
END;
/
```

El 2º trigger aporta la restricción de que en el primer desvío de un vuelo siempre debe ser realizado con el mismo aeropuerto de origen que el vuelo original

- Trigger 2:

```
CREATE OR REPLACE TRIGGER EXT_PRIMER_DESVIO
BEFORE INSERT ON desvios
FOR EACH ROW
DECLARE
    flag NUMBER;
    origin VARCHAR2(4);
BEGIN
    SELECT origin INTO origin
    FROM vuelo
    WHERE id = :NEW.idVuelo;

    SELECT COUNT(*) INTO flag
    FROM desvios
    WHERE idVuelo = :NEW.idVuelo;

    IF origin <> :NEW.origin and flag = 0 THEN   -- No es el mismo avión que el
del vuelo original vuelo
        RAISE_APPLICATION_ERROR (-20002, 'No existe el primer desvio del vuelo:
' || :NEW.idVuelo || ', con matricula: ' || origin);
    END IF;
END;
/
```

El tercer trigger aporta la restricción de que un vuelo no puede tener una fecha de salida anterior a la creación del avión que lo realiza.

- Trigger 3:

```
CREATE OR REPLACE TRIGGER FECHAS
BEFORE INSERT ON vuelo
FOR EACH ROW
DECLARE
    flag NUMBER;
BEGIN
    SELECT COUNT(*) INTO flag
    FROM avion
    WHERE year > SUBSTR(:NEW.flightDate,1,4);
    IF flag >= 1 THEN
        RAISE_APPLICATION_ERROR (-20000, 'Error al añadir el vuelo: la fecha
del vuelo no puede ser anterior a la de creación del avion');
    END IF;
END;
/
```

# Gestión del grupo

## Reuniones

8/05 Reunión inicial, para organizarnos.

10/05 Modelos E/R

16/05 Probar a poblar la base de datos entera. Comenzar la documentación.

18/05 Consultas 1 y 2

23/05 Consulta 3 y plantear triggers

26/05 Implementación de los triggers

30/05 Diseño físico y documentación.

## División del trabajo

El modelo E/R fue realizado principalmente por Héctor, aportando los otros dos miembros alguna idea.

El modelo relacional lo realizó Pablo y Héctor corrigió alguna parte donde fallaba.

La población del modelo fue realizada por Héctor.

Las consultas fueron planteadas de forma básica por Javier e implementadas por Javier y Pablo, adicionalmente fueron revisadas por Héctor.

Los árboles de las consultas fueron realizados por Héctor y Pablo

Cada uno realizó un trigger y luego fueron puestos en común.

Con la documentación ya avanzada y todo funcionando entre los 3 hicimos el diseño físico y finalizamos la práctica.

## Problemas de coordinación

Dado que a partir de la primera práctica éramos conscientes de la carga de trabajo de la práctica, esta vez no hemos tenido apenas problemas, siendo el único destacable el escaso tiempo para elaborar la práctica dadas todas las entregas finales que hemos tenido en el mismo plazo.El único gran problema con el que nos hemos encontrado en esta práctica ha sido el hecho de que el servidor de la base de datos se cayese durante más de 72 horas justo antes de la entrega lo que nos dificulta revisar el perfecto funcionamiento de todo, así como implementar el diseño físico.