BASES DE DATOS

Base de datos de aviones

Grupo de tardes nº 13

CURSO 22/23

15 de mayo de 2023

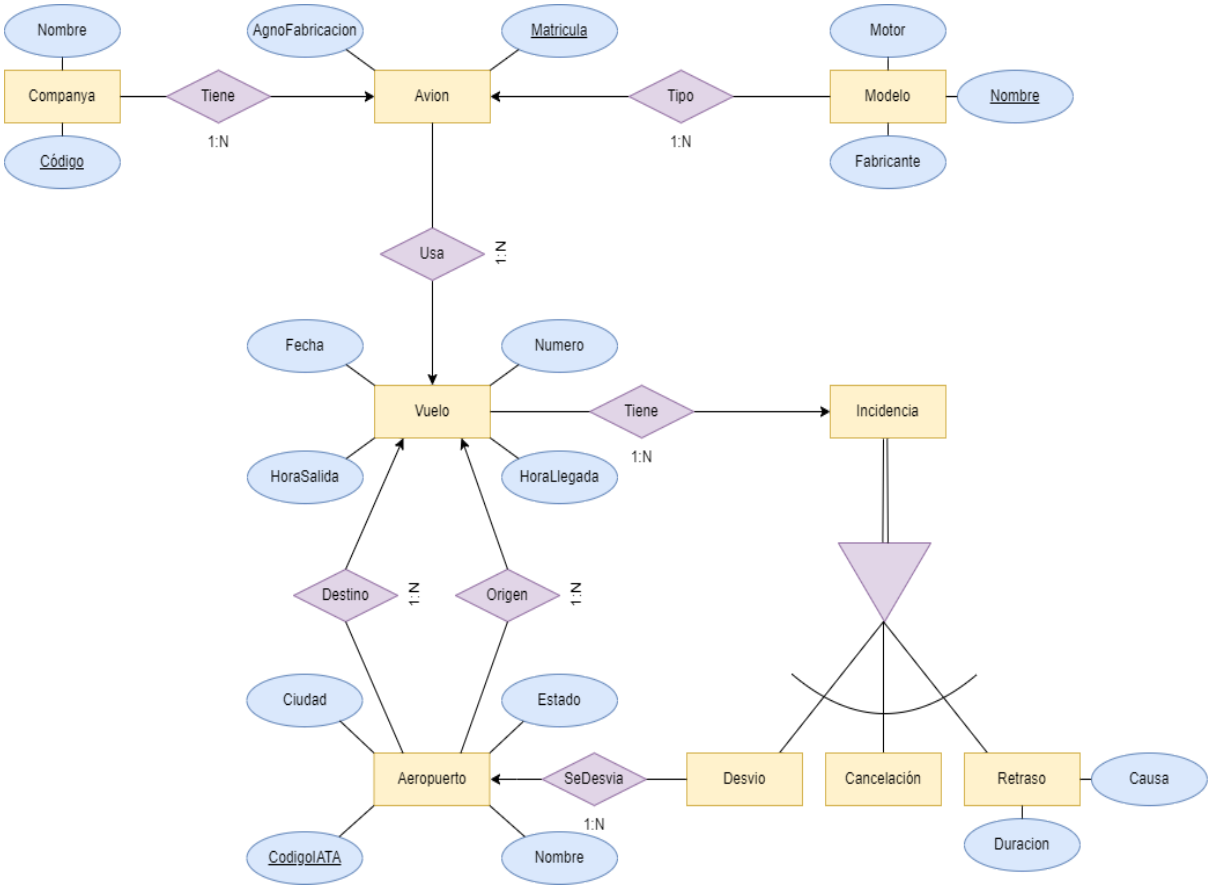
Hecho por LIZER BERNAD FERRANDO, 779035@unizar.es
GUILLERMO BAJO LABORDA, 842748@unizar.es
AXEL ISAAC PAZMIÑO ORTEGA, 817627@unizar.es

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Parte I
Creación de una base de datos

Esquema E/R y restricciones:



En el esquema E/R aparecen un total de 9 **entidades** distintas: *Companya, Avion, Modelo, Vuelo, Aeropuerto* e *Incidencia*(que tiene cobertura total con *Retraso, Desvio* y *Cancelacion*), cada una con sus respectivos **atributos**. Todos los atributos son normales, a excepción de *CodigoIATA* (de *Aeropuerto*), *Nombre* (de *Modelo*) y *Codigo* (de *Companya*), que son únicos, es decir, que los valores de dichos atributos son únicos en cada tabla. Para definir las incidencias se ha empleado una **especialización de cobertura total**, una subdivisión de una clase principal en subclases, donde cada subclase representa una entidad diferente. Por otra parte, se han utilizado **relaciones** para poner en relación las diversas entidades. Son un total de 7 relaciones, todas ellas 1:N

A continuación se muestran alguna de las principales **restricciones** encontradas:

Avion:

- El año de fabricación de un avión debe estar entre 1903 (año de fabricación del primer avión) y 2023.
- Un avión no puede realizar dos vuelos simultáneamente.

Vuelo:

- La fecha de un vuelo no puede ser anterior al año de fabricación del avión.
- Las horas de salida y de llegada deben estar entre 00:00 y 23:59
- La fecha de salida de un vuelo ha de ser anterior a la fecha de llegada.

Aeropuerto:

- Puede haber dos ciudades con el mismo nombre en distintos estados.

Desvios:

- Un vuelo no puede ser desviado hacia el aeropuerto destino.
- Un vuelo no puede ser desviado dos veces hacia el mismo aeropuerto.

Retraso:

- La duración de un retraso debe ser superior a 0 minutos.

Caben destacar algunas de las **alternativas** que fueron planteadas durante el diseño del esquema E/R:

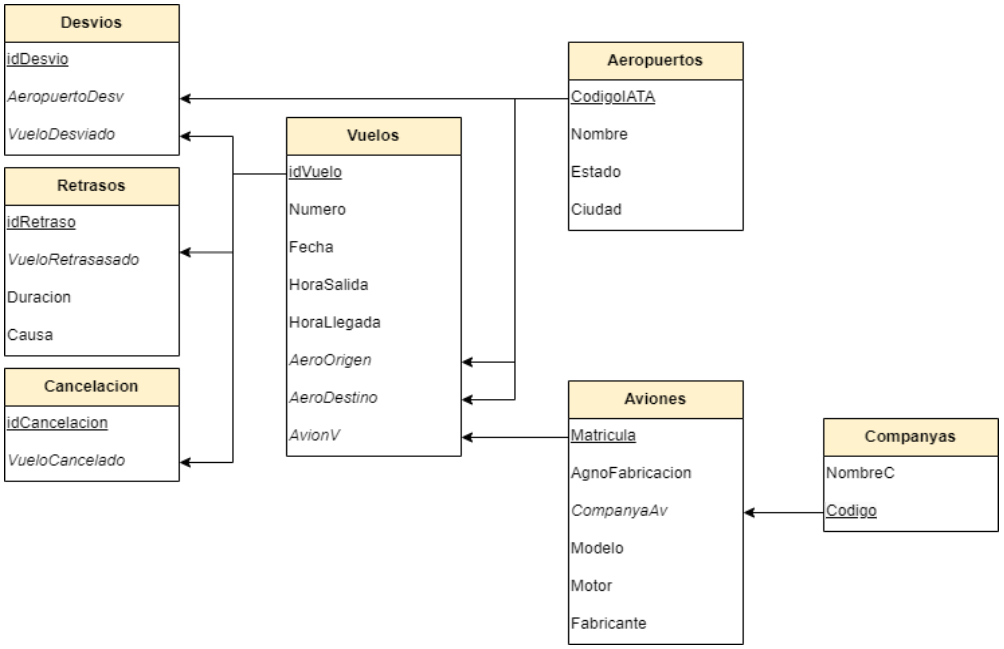
- Relación N:M entre *Companya* y *Vuelo*
- Forma alternativa para las incidencias

Inicialmente, se propuso una relación *Opera* N:M entre las entidades *Companya* y *Vuelo*, puesto que se consideró que de esta forma los datos estarían más accesibles y se evitarían hacer algunos joins en determinadas consultas sobre las compañías aéreas. En cambio, finalmente se decidió descartar esta idea puesto que se priorizó el **no almacenar información redundante** en la base de datos frente a evitar usar joins de más en ciertas consultas.

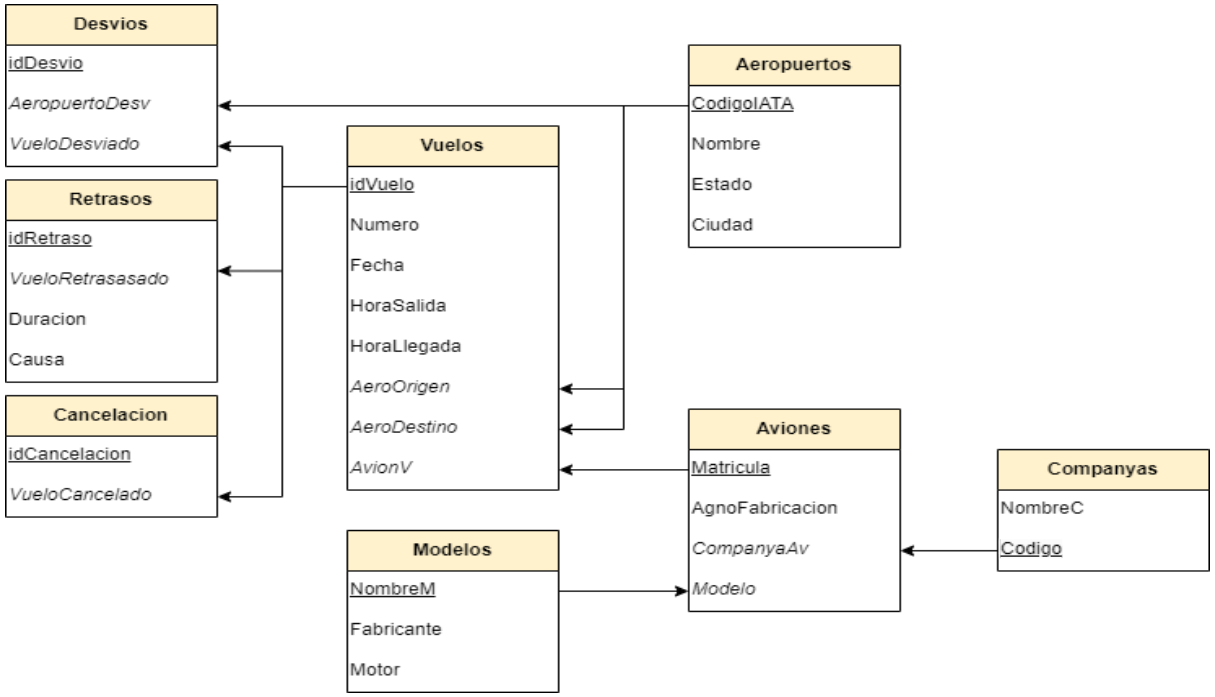
El otro aspecto que causó al grupo más dudas fue la forma de especificar las incidencias. En el primer diseño del esquema entidad relación se especificaron los desvíos con una relación *tieneDesvio* N:M entre *Vuelo* y *Aeropuerto*. Para los retrasos se propuso una entidad *Retraso*, relacionada con *Vuelo* mediante una relación N:M *tieneRetraso*. Las cancelaciones en cambio se gestionaban mediante un atributo de vuelo (*Cancelado*, que tomaba los valores “Si” o “No”). Sin embargo, se plantearon los pros y contras de esa versión y de la versión con la especialización, y se eligió la segunda, por varios motivos. Para empezar, se consideró que no era lo mejor tener un atributo *Cancelado*, ya que en el caso de que, por ejemplo, hubiese cien mil vuelos y solo cien cancelados, se estarían almacenando en la columna *Cancelado* de la tabla Vuelo decenas de miles de datos que solamente sirven para saber que el vuelo no ha sido cancelado. En cambio, con una tabla *Cancelados*, se guardan solamente tuplas para aquellos vuelos cancelados. Por otra parte, se consideró que cancelación, desvío y retraso eran **especializaciones** de incidencia, por lo que lo más adecuado sería ponerlas **al mismo nivel**, como entidades, y no una como entidad, otra como relación y otra como atributo.

Esquema relacional y normalización:

Una vez el diseño E/R ha sido considerado satisfactorio, se ha llevado a cabo el **modelo relacional** y la normalización del mismo. En el proceso de **Normalización** no ha sido necesario realizar apenas cambios pues el modelo prácticamente ya cumplía con las propiedades de las formas normales exigidas por el enunciado. El **modelo relacional inicial** es el siguiente:



Este modelo cumple tanto la 1FN como la 2FN, pero **viola la 3FN** ya que existen atributos no clases que dependen funcionalmente de atributos no clave. En este caso, tanto motor como fabricante dependen funcionalmente de modelo. Para solucionar esto, se ha creado la tabla *Modelo*, cuyos atributos son *Motor*, *Fabricante* y *NombreM* (clave primaria, que será extranjera en la tabla de *Aviones*. Una vez realizada esta modificación, el modelo cumple la 3FN y la Forma Normal de Boyce-Codd, por tanto, el modelo queda normalizado.



Sentencias SQL de creación de tablas:

La creación de las tablas de la base de datos se ha llevado a cabo mediante el fichero *crear_bd.sql*. En este fichero se crean un total de **8 tablas**, las mismas que aparecen en el modelo relacional normalizado mostrado. También se han añadido **checks** que comprueban varias de las restricciones mencionadas. Así mismo, se dispone de un fichero *borrar_bd.sql* para borrar las tablas creadas en orden inverso al de creación.

```
CREATE TABLE Companyas(
    Codigo       VARCHAR(100) PRIMARY KEY,
    NombreC      VARCHAR(100)
);

CREATE TABLE Aeropuertos(
    codigoIATA  VARCHAR(5) PRIMARY KEY,
    Nombre      VARCHAR(100),
    Estado      VARCHAR(100),
    Ciudad      VARCHAR(100)
);

CREATE TABLE Modelos(
    NombreM         VARCHAR(100) PRIMARY KEY,
    Fabricante      VARCHAR(100),
    Motor           VARCHAR(100)
);

CREATE TABLE Aviones(
    Matricula       VARCHAR(10) PRIMARY KEY,
    AgnoFabricacion NUMBER(4),
    Modelo          VARCHAR(100),
    CompanyaAv      VARCHAR(100),
    FOREIGN KEY (Modelo) REFERENCES Modelos(NombreM),
    FOREIGN KEY (CompanyaAv) REFERENCES Companyas(Codigo),
    CONSTRAINT ck_Fabricacion CHECK(AgnoFabricacion > 1903 AND
    AgnoFabricacion < 2024)
);


CREATE TABLE Vuelos(
    idVuelo          NUMBER(5) PRIMARY KEY,
    Numero           NUMBER(5),
    Fecha            VARCHAR(10),
    HoraSalida       NUMBER(5),
    HoraLlegada      NUMBER(5),
```

```
    AeroOrigen       VARCHAR(5),
    AeroDestino      VARCHAR(5),
    AvionV          VARCHAR(10),
    FOREIGN KEY (AeroOrigen) REFERENCES Aeropuertos(codigoIATA),
    FOREIGN KEY (AeroDestino) REFERENCES Aeropuertos(codigoIATA),
    FOREIGN KEY (AvionV) REFERENCES Aviones(Matricula),
    CONSTRAINT ck_HoraSal CHECK (HoraSalida>=0 AND HoraSalida<=2359),
    CONSTRAINT ck_Lleg CHECK (HoraLlegada>=0 AND HoraLlegada<=2359
);


CREATE TABLE Retrasos(
    idRetraso       NUMBER(5) PRIMARY KEY,
    VueloRetrasado  NUMBER(5),
    Duracion        NUMBER(5),
    Causa           VARCHAR(100),
    CONSTRAINT ck_Duracion CHECK (Duracion > 0),
    FOREIGN KEY (VueloRetrasado) REFERENCES Vuelos(idVuelo)
);


CREATE TABLE Desvios(
    idDesvio         NUMBER(3) PRIMARY KEY,
    AeropuertoDesv   VARCHAR(5),
    VueloDesviado    NUMBER(5),
    FOREIGN KEY (AeropuertoDesv) REFERENCES Aeropuertos(codigoIATA),
    FOREIGN KEY (VueloDesviado) REFERENCES Vuelos(idVuelo)
);


CREATE TABLE Cancelaciones(
    idCancelacion     NUMBER(4) PRIMARY KEY,
    VueloCancelado    NUMBER(5),
    FOREIGN KEY (VueloCancelado) REFERENCES Vuelos(idVuelo)
);
```

Parte II
Introducción de datos y ejecución de consultas

Población de la base:

Tras haber realizado este proceso para las dos bases anteriores, el proceso de población ha resultado bastante sencillo y apenas han surgido complicaciones relacionadas con la población en sí. Para cargar los datos en las tablas se ha llevado a cabo el siguiente procedimiento:

Inicialmente se ha partido del recurso csv *DatosVuelo* proporcionado en un recurso de la plataforma Moodle. Para cada una de las tablas, se han **seleccionado los campos deseados** y se han borrado el resto, se han **borrado los duplicados** y se han introducido los datos en la tabla correspondiente mediante la creación de un sencillo fichero ctl que introduce los datos en la tabla seleccionada. Este proceso se ha repetido un total de 8 veces, una por tabla, lo que da lugar a un total de 8 ficheros csv y 8 ficheros ctl que permiten la población de las tablas.

La mayoría de csvs se han creado rápidamente. En cambio, alguno ha supuesto más tiempo de hacer, como por ejemplo *Retrasos.csv,* en donde primero se han filtrado los vuelos dejando solo aquellos que han sido retrasados. Una vez seleccionados todos los vuelos retrasados, se han modificado la disposición de los datos de forma que una columna sea la duración del retraso, otra la causa y otra el identificador del vuelo asociado a ese retraso. Finalmente, se le ha asociado un identificador de retraso a cada una de las tuplas resultantes.

Lo que más tiempo ha llevado ha sido poblar las tablas de nuevo tras cambiar el E/R y el modelo relacional (por el tema de las incidencias comentado en la parte I). Otra de las decisiones de esta parte que más tiempo llevó al grupo tomar fue cómo poblar las fechas y horas. Inicialmente, se planteó cargarlas en la base como tipos de datos DATE. Sin embargo, esto habría supuesto la modificación de los datos originales para convertirlos a un formato que no era indispensable. A su vez, para esto supondría utilizar marcas de tiempo como TIMESTAMP y otras operaciones no dadas en clase, por lo que finalmente se decidió poblar la base con los datos suministrados sin realizar ningún tipo de modificación sobre estos.

Una vez creados los archivos csv y ctl, para hacer más cómoda la población de la base se ha diseñado un **script** *poblar.sh* que solicitará al usuario su contraseña de Oracle y poblará automáticamente todas las tablas. Para ello es necesario que los csv y ctl se encuentren en el mismo directorio del script. Para ejecutarlo basta con darle permisos de ejecución (chmod u+x poblar.sh) y ejecutarlo (./poblar.sh). Esto evitará tener que poblar las tablas una a una e ir introduciendo la contraseña de Oracle para poblar cada tabla.

Consultas:

1. **Lista de las tres compañías aéreas menos puntuales (calculado en base al porcentaje de vuelos retrasados).**

   Lo más interesante de esta consulta es la vista CompanyasPorPuntualidad que calcula mediante 2 subconsultas primeramente la cantidad de vuelos de cada compañía y después la cantidad de vuelos con retraso de cada compañía. Con esa información calcula el porcentaje de retrasos de cada compañía. Después se utiliza esta visa para extraer de la misma las tres primeras filas (Ya que está ordenada de mayor porcentaje de retrasos a menor) y finalmente se muestran los resultados requeridos por el enunciado de la consulta.

```
// Compañias ordenadas por puntualidad por nombre
CREATE OR REPLACE VIEW CompanyasPorPuntualidad AS
SELECT NombreC, Puntualidad
FROM Companyas
    JOIN(
        SELECT VuelosCompanyas.Codigo, (TotalRetrasos * 100 / TotalVuelos) AS
Puntualidad
        FROM(
                SELECT Codigo, COUNT(Codigo) AS TotalVuelos
                FROM Vuelos
                        JOIN Aviones ON
                                AvionV = Matricula
                        JOIN Companyas ON
                                CompanyaAv = Codigo
                GROUP BY Codigo
        ) VuelosCompanyas
                JOIN(
                        SELECT Codigo, COUNT(Codigo) AS TotalRetrasos
                        FROM(
                                SELECT DISTINCT idVuelo, AvionV
                                FROM Vuelos
                                        JOIN Retrasos ON
                                                idVuelo = VueloRetrasado
                        ) VuelosRetrasados
                                JOIN Aviones ON
                                        AvionV = Matricula
                                JOIN Companyas ON
                                        CompanyaAv = Codigo
                        GROUP BY Codigo
                ) VuelosRetrasados ON
                        VuelosCompanyas.Codigo = VuelosRetrasados.Codigo
        ORDER BY Puntualidad DESC
    ) PuntualidadCompanyas ON
        Companyas.Codigo = PuntualidadCompanyas.Codigo;

CREATE OR REPLACE VIEW PrimeraCompanyaPorPuntualidad AS
SELECT NombreC, Puntualidad
FROM CompanyasPorPuntualidad
WHERE
    Puntualidad = (
        SELECT MAX(Puntualidad)
        FROM CompanyasPorPuntualidad
    );

CREATE OR REPLACE VIEW SegundaCompanyaPorPuntualidad AS
```

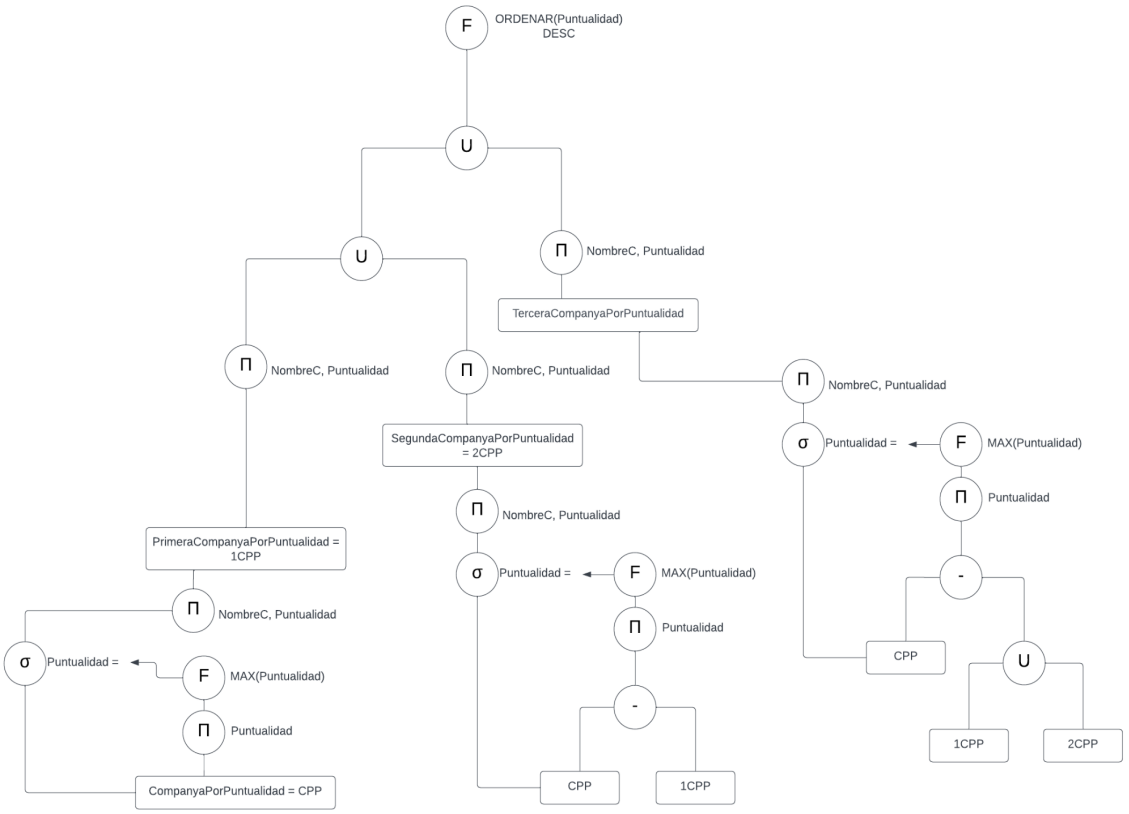
```
SELECT NombreC, Puntualidad
FROM CompanyasPorPuntualidad
WHERE
    Puntualidad = (
        SELECT MAX(Puntualidad)
        FROM(
                SELECT NombreC, Puntualidad
                FROM CompanyasPorPuntualidad
                WHERE
                        NombreC NOT IN(
                                SELECT NombreC
                                FROM PrimeraCompanyaPorPuntualidad
                        )
        )
    );

CREATE OR REPLACE VIEW TerceraCompanyaPorPuntualidad AS
SELECT NombreC, Puntualidad
FROM CompanyasPorPuntualidad
WHERE
    Puntualidad = (
        SELECT MAX(Puntualidad)
        FROM(
                SELECT NombreC, Puntualidad
                FROM CompanyasPorPuntualidad
                WHERE
                        NombreC NOT IN(
                                SELECT NombreC
                                FROM PrimeraCompanyaPorPuntualidad
                                UNION
                                SELECT NombreC
                                FROM SegundaCompanyaPorPuntualidad
                        )
        )
    );

SELECT NombreC
FROM(
    SELECT NombreC, Puntualidad
    FROM PrimeraCompanyaPorPuntualidad
    UNION
    SELECT NombreC, Puntualidad
    FROM SegundaCompanyaPorPuntualidad
    UNION
    SELECT NombreC, Puntualidad
    FROM TerceraCompanyaPorPuntualidad
    ORDER BY Puntualidad DESC
);
```

Salida:

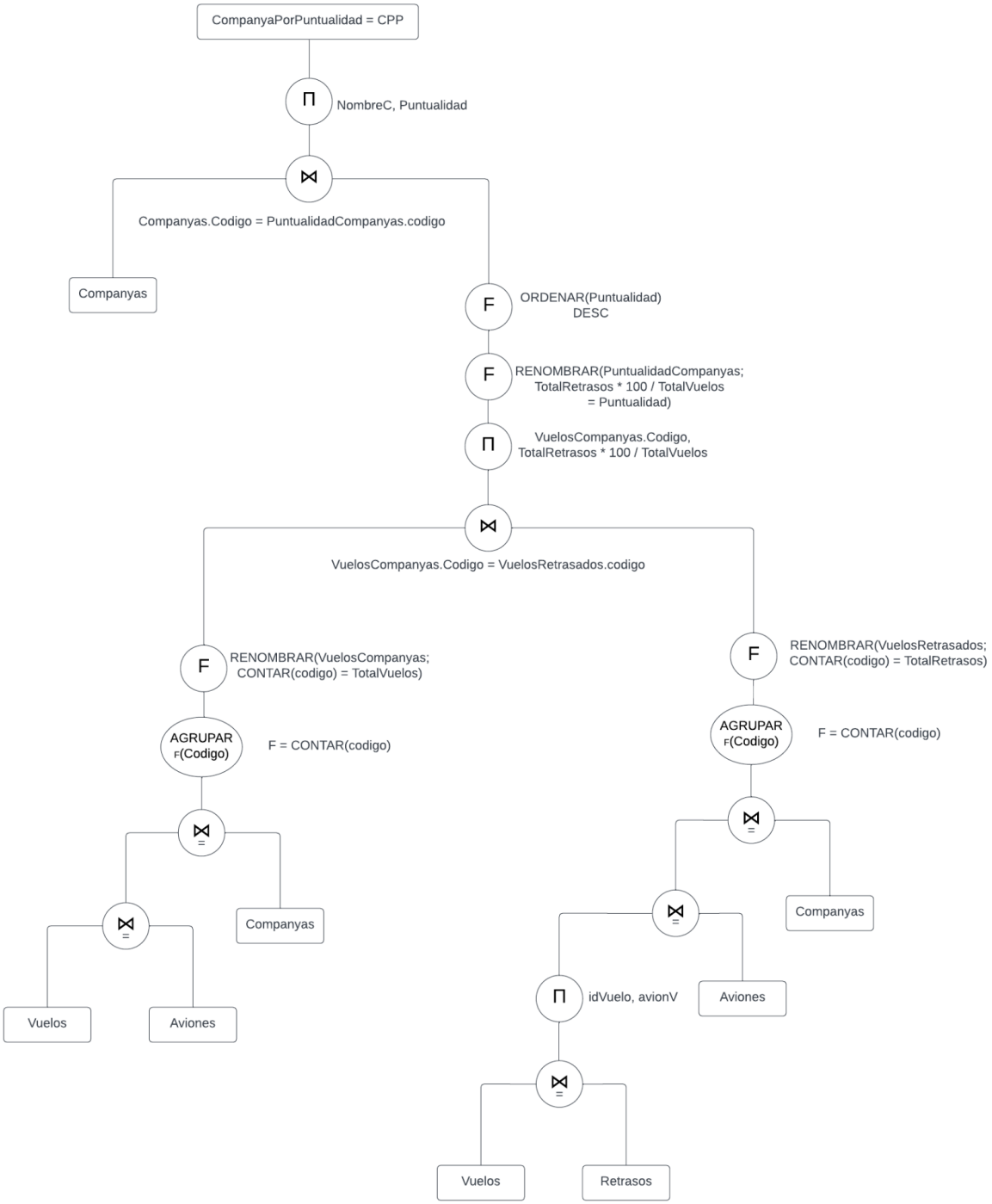
NombreC

American Eagle Airlines Inc.
Mesa Airlines Inc.
American Airlines Inc.

Árbol Sintáctico:



Debido al tamaño del árbol sintáctico se ha optado por dividirlo en 2 partes, a continuación el árbol sintáctico de la vista *CompanyaPorPuntualidad*:


CompanyaPorPuntualidad = CPP
Π
NombreC, Puntualidad
⋈
Companyas.Codigo = PuntualidadCompanyas.codigo
Companyas
F
ORDENAR(Puntualidad)
DESC
F
RENOMBRAR(PuntualidadCompanyas;
TotalRetrasos * 100 / TotalVuelos
= Puntualidad)
Π
VuelosCompanyas.Codigo,
TotalRetrasos * 100 / TotalVuelos
⋈
VuelosCompanyas.Codigo = VuelosRetrasados.codigo
F
RENOMBRAR(VuelosCompanyas;
CONTAR(codigo) = TotalVuelos)
AGRUPAR
F(Codigo)
F = CONTAR(codigo)
⋈=
⋈=
Companyas
Vuelos
Aviones
F
RENOMBRAR(VuelosRetrasados;
CONTAR(codigo) = TotalRetrasos)
AGRUPAR
F(Codigo)
F = CONTAR(codigo)
⋈=
⋈=
Companyas
Π
idVuelo, avionV
Aviones
⋈=
Vuelos
Retrasos

2. **Lista los fabricantes, modelos y motores en los que el número de vuelos retrasados por motivos de seguridad sea mayor del 1 por ciento de su total de retrasos.**

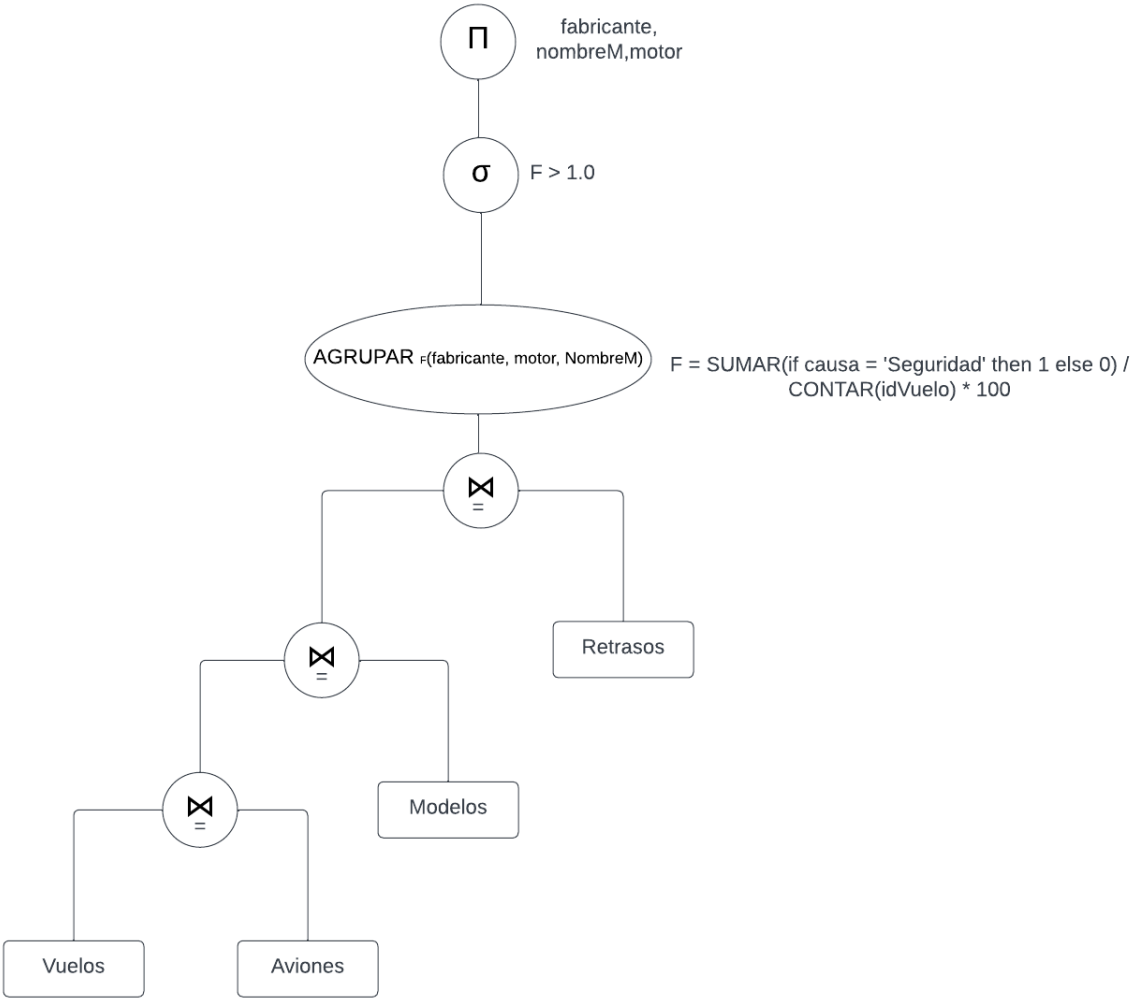
Para esta consulta el **enfoque** se ha planteado de manera que observamos el porcentaje de retrasos por motivos de seguridad respecto al total de retrasos, agrupando por el fabricante, modelo y motor del avión que realiza cada vuelo.

```
SELECT M1.fabricante, M1.nombreM, M1.motor
FROM aviones A1, vuelos V1, modelos M1, retrasos R1
-- hacemos un join de las diferentes tablas para tener de cada vuelo (V1) retrasado (R1)
-- información de su modelo (M1) de avión (A1). descartamos los vuelos de aviones sin
-- información relativa a su modelo.
-- Hay tuplas de vuelos duplicadas si tienen más de una causa de retraso.
WHERE A1.modelo IS NOT NULL AND V1.avionv = A1.Matricula AND M1.nombreM = A1.modelo AND
V1.idVuelo = R1.vueloRetrasado
-- agrupamos los vuelos por el fabricante, modelo y motor del avión que lo realizó
GROUP BY fabricante, motor, NombreM
-- distinct idvuelo nos quita las tuplas duplicadas de los vuelos con más de un tipo
-- de retraso y nos quedamos con los que el porcentaje de retrasos por Seguridad respecto al
-- total de retrasos es mayor del 1%
HAVING SUM(CASE causa WHEN 'Seguridad' THEN 1 ELSE 0 END)/COUNT(DISTINCT idvuelo)*100 > 1.0;
```

Salida:

Fabricante	NombreM	Motor
BOEING	737-5H4	Turbo-Jet
BOEING	737-524	Turbo-Fan
AIRBUS	A319-132	Turbo-Jet
BOEING	757-33N	Turbo-Jet
BOEING	767-33A	Turbo-Fan
BOEING	737-3H4	Turbo-Fan
BOMBARDIER INC	CL600-2D24	Turbo-Fan
BOEING	737-924ER	Turbo-Fan
BOEING	737-7H4	Turbo-Fan
BOEING	737-3G7	Turbo-Jet
BOEING	737-3TO	Turbo-Jet
AIRBUS	A321-211	Turbo-Jet

Árbol sintáctico:



3. **Aeropuertos que han tenido el momento con mayor actividad, y momento en el que ha ocurrido. La actividad de un aeropuerto se calcula en el horario estimado de salida de cada vuelo y consiste en el número de aviones a menos de 15 minutos de despegar o aterrizar.**

   En esta consulta se obtiene una subconsulta con toda la actividad de cada aeropuerto formada por la actividad generada por salidas de aviones unida a la actividad generada por llegadas de aviones. Después se recorre el resultado de la subconsulta de la actividad de salidas al representar esto los "momentos candidatos" y cada momento de estos se comprueba con la totalidad de los momentos para ver si está 15 min antes o menos en el tiempo, si esto se cumple se incrementa la actividad de este momento.

   La consulta finaliza extrayendo el momento con mayor actividad tras realizar la comprobación anterior con todos los momentos y se obtiene el nombre del aeropuerto al que pertenece ese momento.

```
CREATE OR REPLACE VIEW ActividadAeropuertosSalidas AS
SELECT idVuelo, AeroOrigen AS Aeropuerto, Fecha, HoraSalida AS Hora
FROM Vuelos;

CREATE OR REPLACE VIEW ActividadAeropuertos AS
SELECT idVuelo, AeroDestino AS Aeropuerto, Fecha, HoraLlegada AS Hora
FROM Vuelos
UNION
SELECT *
FROM ActividadAeropuertosSalidas;

CREATE OR REPLACE VIEW ActividadMomentos AS
SELECT a1.Aeropuerto, a1.Fecha, a1.Hora, count(*) AS Actividad
FROM ActividadAeropuertosSalidas a1
JOIN ActividadAeropuertos a2 ON
        a1.Aeropuerto = a2.Aeropuerto AND ((
                        a1.Fecha = a2.Fecha AND
                        a2.Hora - a1.Hora <= 15 AND
                        a2.Hora - a1.Hora >= 0
                ) OR (
                        to_date(a2.Fecha) = to_date(a1.Fecha) + 1 AND
                        a2.Hora - (a1.Hora - 2400) <= 15 AND
                        a2.Hora - (a1.Hora - 2400) >= 0
                )
        )
GROUP BY a1.Aeropuerto, a1.Fecha, a1.Hora
ORDER BY Actividad DESC;

SELECT Nombre, Fecha, Hora
FROM (
        SELECT Aeropuerto, Fecha, Hora, Actividad
        FROM ActividadMomentos
        WHERE Actividad = (
                        SELECT MAX(Actividad)
                        FROM ActividadMomentos
                )
) JOIN Aeropuertos ON
        Aeropuerto = CodigoIATA;
```

Salida:

Nombre	Fecha	Hora
William B Hartsfield-Atlanta Intl	26/10/2008	830

Parte III
Diseño físico

Análisis de rendimiento de las consultas:

Ante la utilización de vistas para realizar las consultas, y estas tener un tamaño considerable, para optimizar las consultas 1 y 3 se decidió utilizar vistas materializadas en vez de vistas normales en: *ActividadMomentos* y *CompanyasPorPuntualidad.*

Al analizar los resultados de los pasos de planificación que se realizan en la consulta 1, ni siquiera cabe completo todo el plan con más de 200 pasos a realizar, prácticamente con todas con un coste de más del 1% del uso del CPU y con unos accesos a filas de más de 1K de bytes e incluso megas. Comparado con la vista materializada podemos apreciar notablemente el cambio tanto en el número de pasos menor de 40, y aunque es cierto que el coste de CPU al principio es costoso por las operaciones que tiene que realizar, el resto de pasos con casi despreciables pudiendo apreciar un gran cambio sobre todo en el tiempo de respuesta a la consulta pasando de varios segundos, minutos, a prácticamente de inmediato o pocos segundos.

Vamos a comparar los análisis del uso de los recursos de las consultas sin optimizar y optimizadas de la consulta 3:

Sin optimizar:

```
-----------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                 | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |   922 | 85746 |       |  2022   (3)| 00:00:01 |
|*  1 |  HASH JOIN                   |                      |   922 | 85746 |       |  2022   (3)| 00:00:01 |
|   2 |   TABLE ACCESS FULL          | AEROPUERTOS          |   272 | 15232 |       |     3   (0)| 00:00:01 |
|   3 |   VIEW                       | ACTIVIDADMOMENTOS    |   922 | 34114 |       |  2019   (3)| 00:00:01 |
|   4 |    SORT ORDER BY             |                      |   922 | 44256 |       |  2019   (3)| 00:00:01 |
|*  5 |     FILTER                   |                      |       |       |       |            |          |
|   6 |      HASH GROUP BY           |                      |   922 | 44256 |       |  2019   (3)| 00:00:01 |
|*  7 |       HASH JOIN              |                      |   922 | 44256 |       |  2017   (3)| 00:00:01 |
|   8 |        TABLE ACCESS FULL     | VUELOS               | 42844 |  1004K|       |   102   (0)| 00:00:01 |
|   9 |        VIEW                  | ACTIVIDADAEROPUERTOS | 85688 |  2008K|       |  1868   (1)| 00:00:01 |
|  10 |         SORT UNIQUE          |                      | 85688 |  3096K|  4376K|  1868   (1)| 00:00:01 |
|  11 |          UNION-ALL           |                      |       |       |       |            |          |
|  12 |           TABLE ACCESS FULL  | VUELOS               | 42844 |  1548K|       |   102   (0)| 00:00:01 |
|  13 |           TABLE ACCESS FULL  | VUELOS               | 42844 |  1548K|       |   102   (0)| 00:00:01 |
|  14 |      SORT AGGREGATE          |                      |     1 |    13 |       |            |          |
|  15 |       VIEW                   | ACTIVIDADMOMENTOS    |   922 | 11986 |       |  2018   (3)| 00:00:01 |
|  16 |        SORT GROUP BY         |                      |   922 | 44256 |       |  2018   (3)| 00:00:01 |
|* 17 |         HASH JOIN            |                      |   922 | 44256 |       |  2017   (3)| 00:00:01 |
|  18 |          TABLE ACCESS FULL   | VUELOS               | 42844 |  1004K|       |   102   (0)| 00:00:01 |
|  19 |          VIEW                | ACTIVIDADAEROPUERTOS | 85688 |  2008K|       |  1868   (1)| 00:00:01 |
|  20 |           SORT UNIQUE        |                      | 85688 |  3096K|  4376K|  1868   (1)| 00:00:01 |
|  21 |            UNION-ALL         |                      |       |       |       |            |          |
|  22 |             TABLE ACCESS FULL| VUELOS               | 42844 |  1548K|       |   102   (0)| 00:00:01 |
|  23 |             TABLE ACCESS FULL| VUELOS               | 42844 |  1548K|       |   102   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

   1 - access("AEROPUERTO"="CODIGOIATA")
   5 - filter(COUNT(*)= (SELECT MAX("ACTIVIDAD") FROM  (SELECT "AEROORIGEN" "AER
OPUERTO","FECHA"

              "FECHA","HORASALIDA" "HORA",COUNT(*) "ACTIVIDAD" FROM  ( (SELECT "IDVUELO" "IDVUELO","AERODESTINO"

              "AEROPUERTO","FECHA" "FECHA","HORALLEGADA" "HORA" FROM "A817627"."VUELOS" "VUELOS")UNION (SELECT

              "IDVUELO" "IDVUELO","AEROORIGEN" "AEROPUERTO","FECHA" "FECHA","HORASALIDA" "HORA" FROM

              "A817627"."VUELOS" "VUELOS")) "A2","A817627"."VUELOS" "VUELOS" WHERE ("FECHA"="A2"."FECHA" AND

              "A2"."HORA"-"HORASALIDA"<=15 AND "A2"."HORA"-"HORASALIDA">=0 OR
              TO_DATE("A2"."FECHA")=TO_DATE("FECHA")+1 AND "A2"."HORA"-("HORASALIDA"-2400)<=15 AND

              "A2"."HORA"-("HORASALIDA"-2400)>=0) AND "AEROORIGEN"="A2"."AEROPUERTO" GROUP BY

              "AEROORIGEN","FECHA","HORASALIDA") "ACTIVIDADMOMENTOS"))
   7 - access("AEROORIGEN"="A2"."AEROPUERTO")
       filter("FECHA"="A2"."FECHA" AND "A2"."HORA"-"HORASALIDA"<=15 AND "A2"."HORA"-"HORASALIDA">=0

              OR TO_DATE("A2"."FECHA")=TO_DATE("FECHA")+1 AND "A2"."HORA"-("HORASALIDA"-2400)<=15 AND
              "A2"."HORA"-("HORASALIDA"-2400)>=0)
  17 - access("AEROORIGEN"="A2"."AEROPUERTO")
       filter("FECHA"="A2"."FECHA" AND "A2"."HORA"-"HORASALIDA"<=15 AND "A2"."HORA"-"HORASALIDA">=0

              OR TO_DATE("A2"."FECHA")=TO_DATE("FECHA")+1 AND "A2"."HORA"-("HORASALIDA"-2400)<=15 AND

              "A2"."HORA"-("HORASALIDA"-2400)>=0)
```

Optimizada:

```
-------------------------------------------------------------------------------------------------------

| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |

-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |     1 |    93 |    73 (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |                   |     1 |    93 |    37 (0)| 00:00:01 |
|   2 |   NESTED LOOPS               |                   |     1 |    93 |    37 (0)| 00:00:01 |
|*  3 |    MAT_VIEW ACCESS FULL      | ACTIVIDADMOMENTOS |     1 |    37 |    36 (0)| 00:00:01 |
|   4 |     SORT AGGREGATE           |                   |     1 |    13 |          |          |
|   5 |      MAT_VIEW ACCESS FULL    | ACTIVIDADMOMENTOS | 30173 |   383K|    36 (0)| 00:00:01 |
|*  6 |    INDEX UNIQUE SCAN         | SYS_C00476333     |     1 |       |     0 (0)| 00:00:01 |
|   7 |   TABLE ACCESS BY INDEX ROWID| AEROPUERTOS       |     1 |    56 |     1 (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("ACTIVIDAD"= (SELECT MAX("ACTIVIDAD") FROM "ACTIVIDADMOMENTOS"
              "ACTIVIDADMOMENTOS"))

   6 - access("AEROPUERTO"="CODIGOIATA")
```

Puede observarse a simple vista la cantidad de recursos que se ahorran (de % de CPU y sobre todo de memoria) al ver el número de pasos que realiza la consulta, y las filas a las que accede: Pasamos de tener que usar más del 1% del CPU en operaciones costosas como joins, ordenaciones y ahora prácticamente el despreciable el consumo de la consulta.

Restricciones y creación de triggers:

Se han encontrado diversidad de **restricciones** a tener en cuenta. A continuación se van a listar aquellas que **Oracle no puede verificar** con la estructura de tablas definidas ni con las restricciones de integridad de los CREATE TABLE:

- Un vuelo no puede ser desviado hacia el aeropuerto de destino.
- La fecha de un vuelo no puede ser anterior al año de fabricación del avión.
- Un avión no puede realizar dos vuelos simultáneamente.
- La fecha de salida de un vuelo ha de ser anterior a la fecha de llegada
- Un mismo vuelo no puede ser desviado dos veces al mismo aeropuerto.

De esta forma, se han decidido implementar los siguientes triggers:

DesvioDistintoDestino: Es el trigger correspondiente a la primera restricción mencionada. Previene la inserción de tuplas en la tabla *Desvios* en las que el aeropuerto del desvío sea el mismo que el aeropuerto destino del vuelo que sufre el desvío. A continuación, se muestra el código del trigger con comentarios que explican su diseño:

```
-- Trigger que comprueba que un desvío no tenga como destino el
aeropuerto al que originalmente
-- se dirigía el avión, ya que en ese caso no es un desvío.
CREATE OR REPLACE TRIGGER DesvioDistinoDestino
-- El trigger comprueba inserciones de tuplas en la tabla de desvios
BEFORE INSERT OR UPDATE ON Desvios
FOR EACH ROW
DECLARE
    -- Declaramos una variable para almacenar el valor del destino
       original del vuelo desviado
    destinoT VARCHAR(5);
BEGIN
    -- Guardamos en destinoT el aeropuerto destino original
    SELECT AeroDestino INTO destinoT FROM Vuelos
    WHERE (idVuelo =:NEW.VueloDesviado);
    -- Si el aeropuerto de desvio es igual al aeropuerto de destino,
       salta un error
    IF (destinoT =:new.AeropuertoDesv) THEN
          RAISE_APPLICATION_ERROR(-20000,'Error al insertar en Desvios:
un vuelo no puede ser desviado al aeropuerto de destino.');
    END IF;
END DesvioDistinoDestino;
/
```

FabricacionAnteriorAVuelo: Este trigger previene la inserción de tuplas en la tabla *Vuelos* en las que la fecha de fabricación del avión que realiza el vuelo es posterior a la fecha del vuelo. Esto, obviamente, se debe a que un vuelo no puede ser realizado por un avión que en el momento del vuelo ni siquiera ha sido fabricado. A continuación, se muestra el código del trigger con comentarios que explican su diseño:

```
-- Trigger que comprueba que la fecha de un vuelo sea posterior a la de
fabricación del avión
-- que realiza el vuelo
CREATE OR REPLACE TRIGGER FabricacionAnteriorAVuelo
-- El trigger comprueba inserciones de tuplas en la tabla de vuelos
BEFORE INSERT OR UPDATE ON Vuelos
FOR EACH ROW
-- El trigger solo tendrá en cuenta aquellos vuelos que tengan avión,
dado que puede darse el caso de que se recopile información de vuelos
de los que no conste información del avión
WHEN (NEW.AvionV IS NOT NULL)
DECLARE
     -- Declaramos una variable para almacenar el año de fabricacion del
avión que realiza el vuelo a insertar
    fabricacion NUMBER(4);
BEGIN
      -- Guardamos en fabricación el año de fabricacion del vuelo a
insertar
    SELECT AgnoFabricacion INTO fabricacion FROM Aviones
    WHERE (Matricula =:NEW.AvionV);
      -- Si la fabricación es mas reciente que la fecha del vuelo, se
dispara el trigger
    IF (fabricacion > EXTRACT(year FROM TO_DATE(:NEW.Fecha))) THEN
           RAISE_APPLICATION_ERROR(-20000,'Error al insertar en Vuelos:
la fecha del vuelo no puede ser anterior a la de fabricacion del
avion.');
    END IF;
END FabricacionAnteriorAVuelo;
/
```

DobleDesvio: Este trigger previene la inserción de tuplas en la tabla *Desvios* en las que el vuelo desviado ya aparece como desviado hacia el mismo aeropuerto. Esto se debe a que un vuelo no se puede desviar dos veces al mismo aeropuerto. A continuación, se muestra el código del trigger con comentarios que explican su diseño:

```
-- Trigger que comprueba que la un vuelo no sea desviado dos veces al
mismo aeropuerto
CREATE OR REPLACE TRIGGER DobleDesvio
-- El trigger comprueba inserciones de tuplas en la tabla de desvios
BEFORE INSERT ON Desvios
FOR EACH ROW
DECLARE
     -- Declaramos un contador en el que se almacenara el numero de
desvios del mismo vuelo
    -- al mismo aeropuerto en caso de que los haya
    contador INTEGER :=0;
BEGIN
      -- Calculamos la cuenta de desvios del mismo vuelo al mismo
aeropuerto que la tupla que se desea insertar
     SELECT COUNT(*) INTO contador FROM Desvios WHERE (AeropuertoDesv =
:NEW.AeropuertoDesv AND VueloDesviado = :NEW.VueloDesviado);
    -- Si hay algunom se dispara el trigger
    IF contador > 0 THEN
         RAISE_APPLICATION_ERROR(-20000, 'Error al insertar en Desvios:
Este vuelo ya ha sido desviado a este aeropuerto.');
    END IF;
END tr_evitar_desvios_repetidos;
/
```

Coordinación del grupo

De cara a la realización de esta tercera base de datos, han sido necesarias entorno a unas 6 horas de trabajo conjunto del grupo, principalmente en clases prácticas, y alrededor de unas 15h de trabajo individual por cada miembro del equipo. El reparto del trabajo ha sido bastante equitativo y se ha procurado evitar hacer distinciones del mismo, con el fin de que todos los miembros del grupo participaran en todos los apartados por igual y así poder tener distintos puntos de vista en las diversas decisiones tomadas a lo largo del desarrollo de la tercera base de datos.

En cambio, una vez más, ciertos apartados han sido llevados a cabo en mayor medida por un integrante u otro de forma voluntaria más que por asignación de trabajo. En definitiva, se podría decir que todos los integrantes del equipo han participado de forma bastante equitativa en la elaboración del proyecto. Por tanto no ha habido problemas de organización y se ha podido desarrollar la base de datos con relativa fluidez.

El trabajo en sí ha resultado más costoso que el de la segunda base de datos, debido a que pese a conocer las herramientas disponibles, ha habido partes donde ha habido especiales complicaciones que han llevado tiempo para subsanar. El diseño conceptual y lógico se compleataron en la primera sesión de prácticas y las siguientes se enfocaron más en partes como la población o para empezar algunas consultas. Sin embargo, gran parte del trabajo se ha realizado desde casa debido a las modificaciones mencionadas anteriormente en el esquema entidad relación.