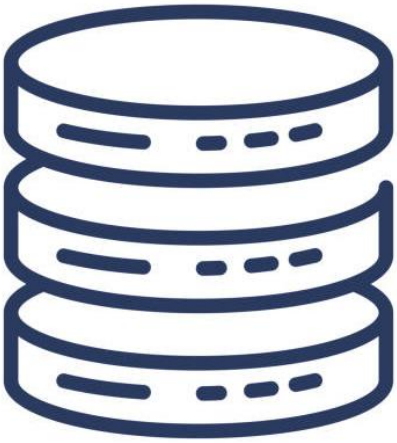
BASES DE DATOS

2023



PRÁCTICA 3

Fecha de entrega
30/06/2023

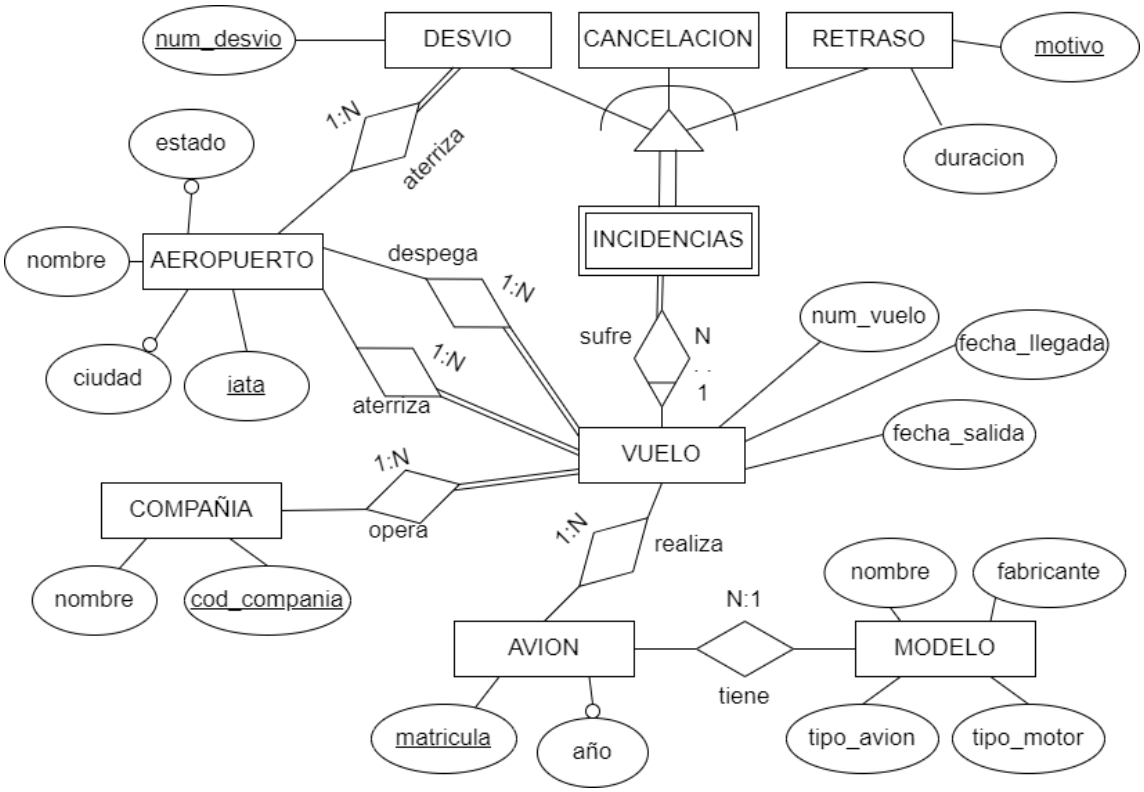
Grupo 11
Inés Román Gracia, 820731@unizar.es

ÍNDICE

Contenido

PARTE 1: CREACIÓN DE LA BASE DE DATOS

1.1. Modelo Entidad-Relación



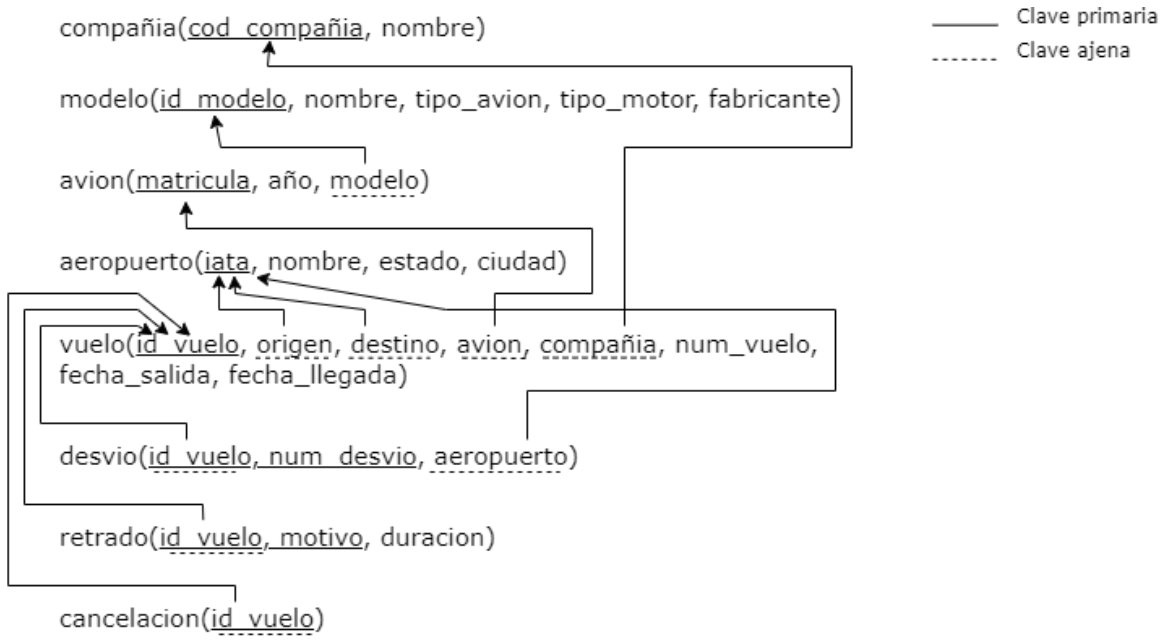
Restricciones

- Un avión no puede realizar vuelos de fecha anterior a su año de fabricación.
- La fecha de salida de un vuelo debe ser anterior a su fecha de llegada.
- Un avión que ha sido cancelado no puede ser desviado.
- Un avión no puede despegar y aterrizar en el mismo aeropuerto, y tampoco puede hacerlo en dos aeropuertos diferentes pero en la misma ciudad.
- Un avión no puede realizar más de un vuelo a la vez.

Soluciones alternativas

- Otra posible solución a la especialización de incidencias habría sido crear dos entidades para desvíos y retrasos, y tener un atributo en la entidad vuelo para las cancelaciones. Esta alternativa mejoraría el rendimiento al consultar los vuelos cancelados, pero a costa de hacer la tabla de vuelos más grande. En este caso, con las consultas planteadas, no supone ninguna mejora.
- Otra alternativa planteada fue hacer una relación entre aviones y compañías, en vez de la relación entre vuelos y compañías. Se descartó esta solución porque suponía una pérdida de datos, ya que hay vuelos de los que no se sabe el avión, pero sí la compañía.

1.2. Modelo Relacional



Decisiones de diseño en las especializaciones

Se ha optado por conservar las tablas de las especializaciones y no la tabla general ya que, además del atributo *id_vuelo* de la clave, no tienen más columnas en común. Por eso, no supone ninguna ventaja tener una tabla para las incidencias. En todo caso, resultaría un inconveniente ya que habría que hacer más intersecciones para acceder a los atributos de las especializaciones.

Normalización

El modelo se encuentra en 1ªFN ya que no hay atributos multivaluados.

El modelo se encuentra en 2ºFN ya que todos los atributos que forman parte de claves compuestas no dependen unos de otros.

El modelo se encuentra en 3ªFN y 3ªFNBC ya que todos los atributos dependes de su clave.

1.3. Sentencias SQL de creación de tablas

```
CREATE TABLE aeropuerto
(
   iata                 VARCHAR(3) PRIMARY KEY,
   nombre               VARCHAR(40) NOT NULL,
   ciudad               VARCHAR(30),
   estado               VARCHAR(2)
);

CREATE TABLE compania
(
   cod_compania         VARCHAR(7) PRIMARY KEY,
   nombre               VARCHAR(100) NOT NULL
);

CREATE TABLE modelo
(
   id_modelo             NUMBER PRIMARY KEY,
   nombre               VARCHAR(20) NOT NULL,
   tipo_avion           VARCHAR(30) NOT NULL,
   tipo_motor           VARCHAR(20) NOT NULL,
   fabricante           VARCHAR(30) NOT NULL
);

CREATE TABLE avion
(
   matricula            VARCHAR(6) PRIMARY KEY,
   anyo                 NUMBER,
   id_modelo            NUMBER,
   FOREIGN KEY(id_modelo) REFERENCES modelo(id_modelo) ON DELETE CASCADE
);

CREATE TABLE vuelo
(
   id_vuelo             NUMBER PRIMARY KEY,
   origen               VARCHAR(3) NOT NULL,
   destino              VARCHAR(3) NOT NULL,
   avion                VARCHAR(6),
   num_vuelo            NUMBER,
   fecha_salida         DATE NOT NULL,
   fecha_llegada        DATE NOT NULL,
   cod_compania          VARCHAR(7) NOT NULL,
   FOREIGN KEY(origen) REFERENCES aeropuerto(iata) ON DELETE CASCADE,
   FOREIGN KEY(destino) REFERENCES aeropuerto(iata) ON DELETE CASCADE,
   FOREIGN KEY(avion) REFERENCES avion(matricula) ON DELETE CASCADE,
   FOREIGN KEY(cod_compania) REFERENCES compania(cod_compania) ON DELETE CASCADE
);
```

```
CREATE TABLE desvio
(
   id_vuelo              NUMBER,
   num_desvio            NUMBER,
   aeropuerto            VARCHAR(3) NOT NULL,
   PRIMARY KEY(id_vuelo, num_desvio),
   FOREIGN KEY(id_vuelo) REFERENCES vuelo(id_vuelo) ON DELETE CASCADE,
   FOREIGN KEY(aeropuerto) REFERENCES aeropuerto(iata) ON DELETE CASCADE
);

CREATE TABLE cancelacion
(
   id_vuelo              NUMBER PRIMARY KEY,
   FOREIGN KEY(id_vuelo) REFERENCES vuelo(id_vuelo) ON DELETE CASCADE
);

CREATE TABLE retraso
(
   id_vuelo              NUMBER,
   motivo                VARCHAR(20),
   duracion              NUMBER,
   PRIMARY KEY(id_vuelo, motivo),
   FOREIGN KEY(id_vuelo) REFERENCES vuelo(id_vuelo) ON DELETE CASCADE
);
```

PARTE 2: INTRODUCCIÓN DE DATOS Y EJECUCIÓN DE CONSULTAS

2.1. Población de la base de datos

Para poblar esta base de datos se ha usado la tabla *datosdb.datosvuelos* que se encuentra en el servidor de Oracle realizando inserciones con consultas sobre esta tabla. El código con el que se pobló la base de datos se encuentra en el fichero *poblar.sql*.

En primer lugar, se han poblado aquellas tablas que no tienen ninguna clave ajena y que, por tanto, no dependen de ninguna otra tabla: *aeropuerto, compañía* y *modelo.*

Después se ha creado la tabla de la entidad *avión* con una clave ajena apuntando a la clave primaria de la tabla *modelo*. Esta tabla se ha poblado en dos partes, primero aquellos aviones de los que se sabe el modelo, haciendo las intersecciones necesarias entre la tabla *modelo* y *datosdb.datosvuelos*, y después aquellos de los que se desconoce el modelo.

A continuación se ha poblado la tabla de vuelos, haciendo las intersecciones correspondientes entre las tablas de las que tiene claves ajenas y la tabla *datosdb.datosvuelos*.

Por último, se han poblado las tablas de las especializaciones de incidencia, que tienen una clave ajena apuntado a la tabla de vuelos.

Algunos de los puntos sobre la población que merece la pena destacar son:

- Para la tabla de retrasos se ha realizado una unión con todos los tipos de retrasos y se han sumado aquellos que eran del mismo tipo y eran del mismo vuelo para que los atributos *motivo* e *id_vuelo* puedan formar la clave de esta tabla.
- Para la tabla de desvíos se ha realizado una unión entre las columnas del primer desvío y las del segundo de la tabla *datosdb.*datosvuelos, dándoles valores “1” y “2” respectivamente en el atributo *num_desvio*.
- Al poblar los datos de las fechas de la tabla de vuelos surge el inconveniente de que un vuelo puede llegar al día siguiente. Por ello, se ha poblado de forma que, si la hora de llegada de un vuelo es menor que su hora de salida, eso significa que el vuelo llega un día más tarde y se le suma un día a su fecha de llegada.

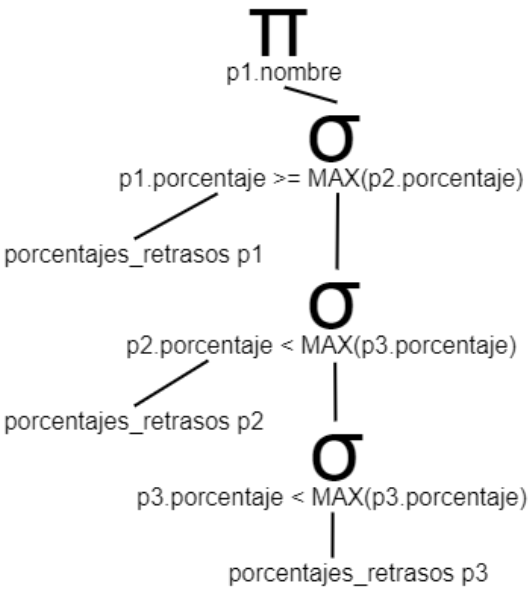
2.2. Consultas SQL

Consulta 1

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

```
CREATE VIEW porcentajes_retrasos AS
SELECT c.nombre, COUNT(DISTINCT r.id_vuelo) / COUNT(DISTINCT v.id_vuelo) * 100 AS
porcentaje
FROM compania c
JOIN vuelo v ON c.cod_compania = v.cod_compania
LEFT JOIN retraso r ON v.id_vuelo = r.id_vuelo
GROUP BY c.nombre;

SELECT nombre
FROM porcentajes_retrasos
WHERE porcentaje >= (
  SELECT MAX(porcentaje)
  FROM porcentajes_retrasos
  WHERE porcentaje < (
    SELECT MAX(porcentaje)
    FROM porcentajes_retrasos
    WHERE porcentaje < (
      SELECT MAX(porcentaje)
      FROM porcentajes_retrasos)))
ORDER BY porcentaje DESC;
```



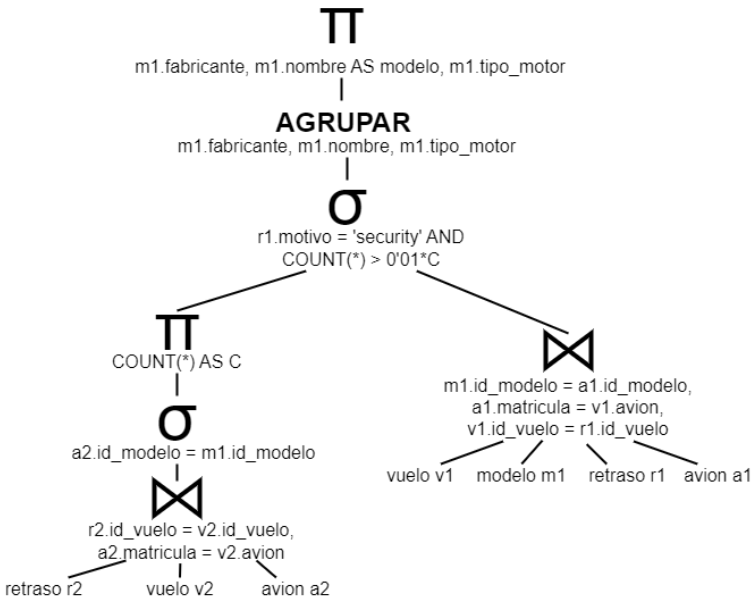
Resultado obtenido

```
NOMBRE
--------------------------------------------------------------------------------
Mesa Airlines Inc.
American Airlines Inc.
American Eagle Airlines Inc.
```

Consulta 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.

```
SELECT m1.fabricante, m1.nombre AS modelo, m1.tipo_motor
FROM modelo m1
JOIN avion a1 ON m1.id_modelo = a1.id_modelo
JOIN vuelo v1 ON a1.matricula = v1.avion
JOIN retraso r1 ON v1.id_vuelo = r1.id_vuelo
WHERE r1.motivo = 'security'
GROUP BY m1.fabricante, m1.nombre, m1.tipo_motor, m1.id_modelo
HAVING COUNT(r1.id_vuelo) > 0.01 * (
   SELECT COUNT(*)
   FROM retraso r2
   JOIN vuelo v2 ON v2.id_vuelo = r2.id_vuelo
   JOIN avion a2 ON a2.matricula = v2.avion
   WHERE a2.id_modelo = m1.id_modelo
);
```



Resultado obtenido

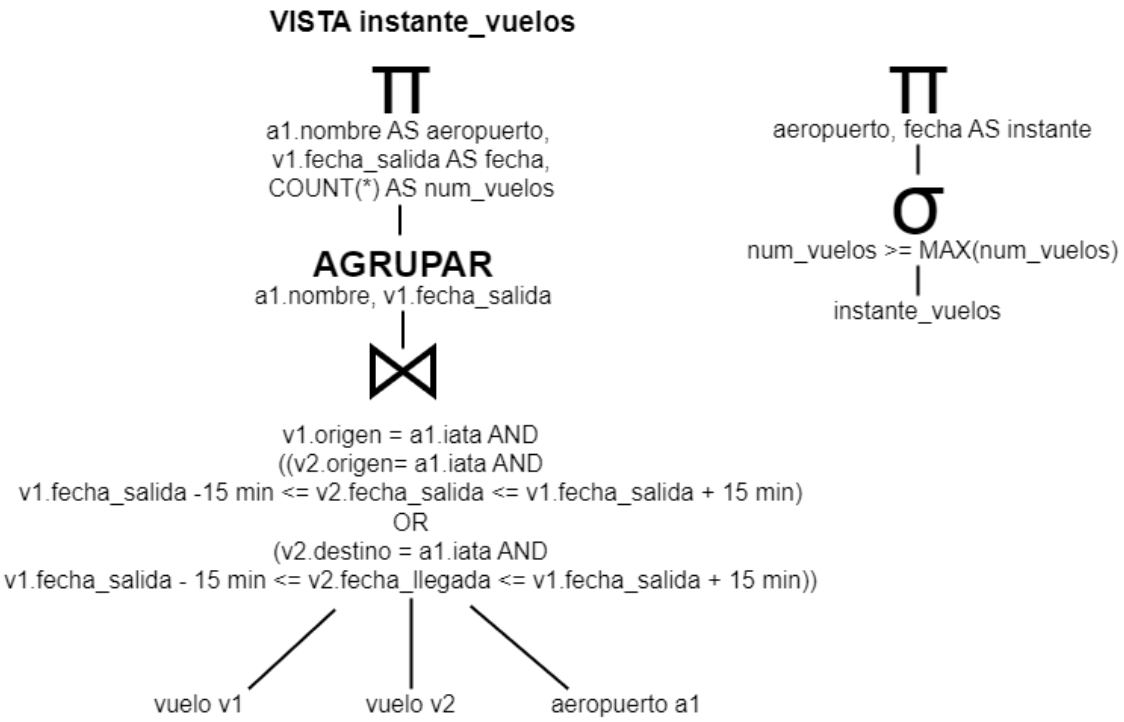
FABRICANTE	MODELO	TIPO_MOTOR
BOMBARDIER INC	CL600-2D24	Turbo-Fan
BOEING	737-3G7	Turbo-Jet
BOEING	737-3TO	Turbo-Jet
BOEING	767-33A	Turbo-Fan
BOEING	737-5H4	Turbo-Jet
AIRBUS	A321-211	Turbo-Jet
AIRBUS	A319-132	Turbo-Jet
BOEING	737-3H4	Turbo-Fan
BOEING	737-524	Turbo-Fan
BOEING	757-33N	Turbo-Jet
BOEING	737-924ER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

Consulta 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 (según el horario estimado).

```
CREATE VIEW instante_vuelos AS
SELECT a1.nombre AS aeropuerto, v1.fecha_salida as fecha, COUNT(*) AS num_vuelos
FROM vuelo v1
JOIN aeropuerto a1 ON v1.origen = a1.iata
JOIN vuelo v2 ON ((v2.origen = a1.iata AND v2.fecha_salida BETWEEN (v1.fecha_salida -
INTERVAL '15' MINUTE) AND (v1.fecha_salida + INTERVAL '15' MINUTE))
            OR (v2.destino = a1.iata AND v2.fecha_llegada BETWEEN (v1.fecha_salida - INTERVAL
'15' MINUTE) AND (v1.fecha_salida + INTERVAL '15' MINUTE)))
GROUP BY a1.nombre, v1.fecha_salida;

SELECT aeropuerto, TO_CHAR(fecha, 'YYYY-MM-DD HH24:MI') AS instante
FROM instante_vuelos
WHERE num_vuelos = (
   SELECT MAX(num_vuelos)
   FROM instante_vuelos
);
```



Resultado obtenido

AEROPUERTO	INSTANTE
William B Hartsfield-Atlanta Intl	2008-10-27 08:35

PARTE 3: DISEÑO FÍSICO

3.1. Rendimiento de las consultas

Consulta 1

El coste de la consulta es el coste de recorrer la intersección de la tabla compañía con las tablas de vuelos y retrasos cuatro veces. Uno de los mayores problemas de rendimiento de esta consulta es el gran tamaño de la tabla vuelo. Resultaría conveniente realizar una partición vertical de esta para mejorar las consultas. Las mejoras se encuentran en el fichero *consulta1_mejorada.sql*.

Coste en CPU inicial: 2509

Primera mejora

La primera mejora consiste en la partición de la tabla *vuelo.* Para esta consulta se crea la tabla *vuelo2* donde se guardan únicamente las columnas del identificador de vuelo y el código de la compañía y se crea la misma vista pero usando la tabla *vuelo2*.

Coste en CPU tras la primera mejora: 888

```
CREATE TABLE vuelo2
(
  id_vuelo              NUMBER PRIMARY KEY,
  cod_compania          VARCHAR(7) NOT NULL,
  FOREIGN KEY(cod_compania) REFERENCES compania(cod_compania) ON DELETE CASCADE
);

INSERT INTO vuelo2(id_vuelo, cod_compania)
SELECT id_vuelo, cod_compania
FROM vuelo;

CREATE VIEW porcentajes_retrasos AS
SELECT c.nombre, COUNT(DISTINCT r.id_vuelo) / COUNT(DISTINCT v.id_vuelo) * 100 AS porcentaje
FROM compania c
JOIN vuelo2 v ON c.cod_compania = v.cod_compania
LEFT JOIN retraso r ON v.id_vuelo = r.id_vuelo
GROUP BY c.nombre;
```

Segunda mejora

Se ha materializado la vista que se usa en la consulta. La mejora es muy significativa, pero habría que considerar si resulta conveniente esta decisión pues mantener esta vista materializada también tiene un coste.

Coste en CPU tras la segunda mejora: 50

```
CREATE MATERIALIZED VIEW porcentajes_retrasos AS
SELECT c.nombre, COUNT(DISTINCT r.id_vuelo) / COUNT(DISTINCT v.id_vuelo) * 100 AS
porcentaje
FROM compania c
JOIN vuelo2 v ON c.cod_compania = v.cod_compania
LEFT JOIN retraso r ON v.id_vuelo = r.id_vuelo
GROUP BY c.nombre;
```

Consulta 2

El coste de esta consulta es el recorrido de la intersección de las tablas avión, vuelo, retraso y modelo agrupándolos por modelos y, para cada modelo, se hace un subrecorrido de la intersección de las tablas avión, vuelo y retraso. Los problemas de rendimiento de esta consulta es el gran número de intersecciones y el gran tamaño de la tabla vuelo. Las mejoras de esta consulta se encuentran en el fichero *consulta2_mejorada.sql*.

Coste en CPU inicial: 666

Primera mejora

Se han encontrado tres índices que consiguen mejorar el rendimiento de la consulta, aunque la mejora es poco significativa.

Coste en CPU tras la primera mejora: 639

```
CREATE INDEX idx_avion_modelo ON avion(id_modelo);
CREATE INDEX idx_retraso_vuelo  ON retraso(id_vuelo);
CREATE BITMAP INDEX idx_retraso_motivo ON retraso(motivo);
```

Segunda mejora

Se ha creado la tabla *vuelo3*, que es una partición de la tabla vuelo donde se guardan únicamente el identificador del vuelo y la matrícula del avión que lo realiza.

Coste en CPU tras la segunda mejora: 363

```
CREATE TABLE vuelo3
(
   id_vuelo              NUMBER PRIMARY KEY,
   avion                 VARCHAR(6),
   FOREIGN KEY(avion) REFERENCES avion(matricula) ON DELETE CASCADE
);

INSERT INTO vuelo3(id_vuelo, avion)
SELECT id_vuelo, avion
FROM vuelo;

SELECT m1.fabricante, m1.nombre AS modelo, m1.tipo_motor
FROM modelo m1
JOIN avion a1 ON m1.id_modelo = a1.id_modelo
JOIN vuelo3 v1 ON a1.matricula = v1.avion
JOIN retraso r1 ON v1.id_vuelo = r1.id_vuelo
WHERE r1.motivo = 'security'
GROUP BY m1.fabricante, m1.nombre, m1.tipo_motor, m1.id_modelo
HAVING COUNT(r1.id_vuelo) > 0.01 * (
   SELECT COUNT(*)
   FROM retraso r2
   JOIN vuelo3 v2 ON v2.id_vuelo = r2.id_vuelo
   JOIN avion a2 ON a2.matricula = v2.avion
   WHERE a2.id_modelo = m1.id_modelo
);
```

Consulta 3

Esta consulta tiene un gran coste y, en parte, también se debe al gran tamaño de la tabla vuelo. Las mejoras de la consulta se encuentran en el fichero *consulta3_mejorada.sql*.
Coste en CPU inicial: 4614

Primera mejora

Se ha creado la tabla *vuelo4* guardando los atributos de vuelo que se utilizan en la consulta.
Coste en CPU tras la primera mejora: 3114

```
CREATE TABLE vuelo4
(
   id_vuelo              NUMBER PRIMARY KEY,
   origen                VARCHAR(3) NOT NULL,
   destino               VARCHAR(3) NOT NULL,
   fecha_salida          DATE NOT NULL,
   fecha_llegada         DATE NOT NULL,
   FOREIGN KEY(origen) REFERENCES aeropuerto(iata) ON DELETE CASCADE,
   FOREIGN KEY(destino) REFERENCES aeropuerto(iata) ON DELETE CASCADE
);

INSERT INTO vuelo4(id_vuelo, origen, destino, fecha_salida, fecha_llegada)
SELECT id_vuelo, origen, destino, fecha_salida, fecha_llegada
FROM vuelo;

CREATE VIEW instante_vuelos AS
SELECT a1.nombre AS aeropuerto, v1.fecha_salida as fecha, COUNT(*) AS num_vuelos
FROM vuelo4 v1
JOIN aeropuerto a1 ON v1.origen = a1.iata
JOIN vuelo4 v2 ON ((v2.origen = a1.iata AND v2.fecha_salida BETWEEN (v1.fecha_salida -
INTERVAL '15' MINUTE) AND (v1.fecha_salida + INTERVAL '15' MINUTE))
          OR (v2.destino = a1.iata AND v2.fecha_llegada BETWEEN (v1.fecha_salida - INTERVAL
'15' MINUTE) AND (v1.fecha_salida + INTERVAL '15' MINUTE)))
GROUP BY a1.nombre, v1.fecha_salida;
```

Segunda mejora

Se ha materializado la vista empleada en la consulta. La vista materializada supone una gran mejora, pero habría que calcular los costes de mantener esta vista y compararlos con los tiempos y recursos ahorrados al realizar consultas utilizando la vista.
Coste en CPU tras la segunda mejora: 212

```
CREATE MATERIALIZED VIEW instante_vuelos AS
SELECT a1.nombre AS aeropuerto, v1.fecha_salida as fecha, COUNT(*) AS num_vuelos
FROM vuelo4 v1
JOIN aeropuerto a1 ON v1.origen = a1.iata
JOIN vuelo4 v2 ON ((v2.origen = a1.iata AND v2.fecha_salida BETWEEN (v1.fecha_salida -
INTERVAL '15' MINUTE) AND (v1.fecha_salida + INTERVAL '15' MINUTE))
          OR (v2.destino = a1.iata AND v2.fecha_llegada BETWEEN (v1.fecha_salida - INTERVAL
'15' MINUTE) AND (v1.fecha_salida + INTERVAL '15' MINUTE)))
GROUP BY a1.nombre, v1.fecha_salida;
```

3.2. Creacción de triggers

Se han creado tres triggers que aseguran que se cumplen algunas de las restricciones del modelo descritas en el apartado del diseño del modelo entidad-relación.

Trigger 1

Este trigger asegura que se cumplen la restricción impuesta sobre que los vuelos cancelados no pueden ser desvíados.

```
CREATE OR REPLACE TRIGGER desvios_no_cancelados
BEFORE INSERT ON desvio
FOR EACH ROW
DECLARE
   cancelado NUMBER;
BEGIN
   SELECT COUNT(*) INTO cancelado
   FROM cancelacion
   WHERE id_vuelo = :NEW.id_vuelo;

   IF cancelado >= 1 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Un vuelo que ha sido cancelado no puede ser
retrasado');
   END IF;
END;
/
```

Para comprobar el correcto funcionamiento del trigger se ha probado la siguiente inserción de un vuelo cancelado a la tabla de desvíos:

```
INSERT INTO desvio(id_vuelo, num_desvio, aeropuerto) VALUES(303, 1, 'YUM');
```

Trigger 2

Este trigger asegura que se cumplen las restricciones de que un avión no puede volar antes de su año de fabricación y que la fecha de llegada de un vuelo no puede ser anterior a su fecha de salida.

```
CREATE OR REPLACE TRIGGER fechas_vuelo
BEFORE INSERT OR UPDATE ON vuelo
FOR EACH ROW
WHEN (NEW.avion IS NOT NULL)
DECLARE
  anyo_avion NUMBER;
BEGIN
  SELECT anyo
  INTO anyo_avion
  FROM avion
  WHERE matricula = :NEW.avion;

  IF EXTRACT(YEAR FROM :NEW.fecha_salida) < anyo_avion THEN
    RAISE_APPLICATION_ERROR(-20002, 'La fecha de salida del vuelo
    no puede ser anterior al anyo de fabricacion del avion.');
  END IF;

  IF :NEW.fecha_salida > :NEW.fecha_llegada THEN
    RAISE_APPLICATION_ERROR(-20003, 'La fecha de llegada de un vuelo no puede
    ser anterior a su fecha de salida.');
  END IF;
END;
/
```

Se han probado el siguiente código para probar el funcionamiento del trigger:

```
---------- ERROR FECHA ANTERIOR A AVION ----------
INSERT INTO vuelo(id_vuelo, origen, destino, avion, num_vuelo, fecha_salida,
  fecha_llegada, cod_compania)
VALUES (111111, 'SBN', 'TUS', 'N200WN', 123,
  TO_DATE('2004-01-01 10:00', 'YYYY-MM-DD HH24:MI'),
  TO_DATE('2004-01-01 09:00', 'YYYY-MM-DD HH24:MI'), 'WEQ');
-------------------------------------------------------------------------------------------


---------- ERROR FECHAS NO VALIDAS ----------
INSERT INTO vuelo(id_vuelo, origen, destino, avion, num_vuelo, fecha_salida,
  fecha_llegada, cod_compania)
VALUES (111111, 'SBN', 'TUS', 'N200WN', 123,
  TO_DATE('2005-01-01 09:00', 'YYYY-MM-DD HH24:MI'),
  TO_DATE('2005-01-01 08:00', 'YYYY-MM-DD HH24:MI'), 'WEQ');
-------------------------------------------------------------------------------------------
```

Trigger 3

Este trigger comprueba que se cumple la restricción acerca de los aeropuertos de origen y destino de los vuelos.

```
CREATE OR REPLACE TRIGGER aeropuertos_diferentes
BEFORE INSERT OR UPDATE OF origen, destino ON vuelo
FOR EACH ROW
DECLARE
   ciudad_origen VARCHAR(30);
   ciudad_destino VARCHAR(30);
BEGIN
   IF :NEW.origen = :NEW.destino THEN
      RAISE_APPLICATION_ERROR(-20004, 'Los aeropuertos de origen y destino no pueden ser
iguales');
   END IF;

   SELECT ciudad INTO ciudad_origen
   FROM aeropuerto
   WHERE iata = :NEW.origen;

   SELECT ciudad INTO ciudad_destino
   FROM aeropuerto
   WHERE iata = :NEW.destino;

   IF ciudad_destino = ciudad_origen THEN
      RAISE_APPLICATION_ERROR(-20005, 'Un avion no puede viajar de un aeropuerto
      a otro en la misma ciudad');
   END IF;
END;
/
```

Para comprobar el funcionamiento del trigger se ha ejecutado el siguiente código:

```
---------------------- ERROR AEROPUERTOS IGUALES ----------------------------------
INSERT INTO vuelo(id_vuelo, origen, destino, avion, num_vuelo, fecha_salida,
   fecha_llegada, cod_compania)
VALUES (111111, 'SBN', 'SBN', 'N200WN', 123,
   TO_DATE('2004-01-01 10:00', 'YYYY-MM-DD HH24:MI'),
   TO_DATE('2004-01-01 09:00', 'YYYY-MM-DD HH24:MI'), 'WEQ');
-----------------------------------------------------------------------------------


------------------------ ERROR MISMA CIUDAD ---------------------------------------
INSERT INTO vuelo(id_vuelo, origen, destino, avion, num_vuelo, fecha_salida,
   fecha_llegada, cod_compania)
VALUES (111111, 'LGA', 'JFK', 'N200WN', 123,
   TO_DATE('2008-01-01 01:00', 'YYYY-MM-DD HH24:MI'),
   TO_DATE('2008-01-01 09:00', 'YYYY-MM-DD HH24:MI'), 'WEQ');
-----------------------------------------------------------------------------------
```

ANEXO

Recuento de horas

Para la realización de este trabajo me he basado en gran medida en el modelo entidad-relación y el relacional que hice con mis compañeros para la primera entrega. Los ficheros de creación y población de tablas de la entrega anterior también me han permitido ir más rápido, así como haber hecho las consultas anteriormente. Por ello, para el recuento de horas dedicadas he contado también mis horas en las partes 1 y 2 de la entrega anterior que son 10 horas y 16 horas respectivamente.

Parte 1	Parte 2	Parte 3	Memoria	Total
5h + 10h	13h + 16h	6h	9h	59h

Sesiones

Solo incluyo las de la segunda entrega:

09/06/2023 – Correcciones al modelo entidad-relación y relacional 5h

10/06/2023 – Poblar 5h

11/06/2023 – Consultas 8h

12/06/2023 – Diseño físico y triggers 6h

13/06/2023 – Memoria 9h