Bases de Datos

Memoria de prácticas - Gestión de una base de datos de aviones

Grupo: Miércoles B
Hora: 17:00 - 19:00
Turno: Tardes, en todos los integrantes

Integrantes: Jorge Aznar López, Ángel Cañal Muniesa y Abel Chils Trabanco

NIPS, en orden relativo: 721556, 716205, 718997

ÍNDICE

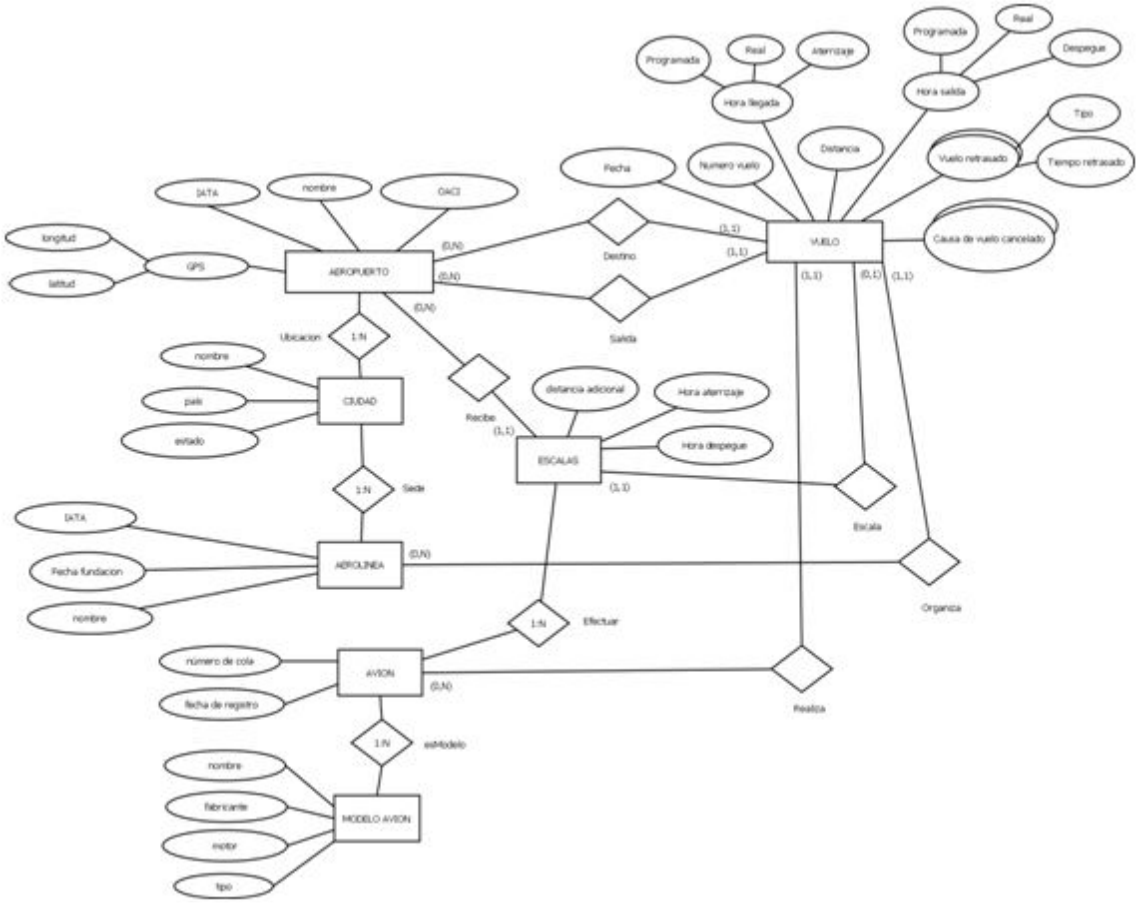
PARTE 1: CREACIÓN DE LA BASE DE DATOS

PARTE 2: POBLACIÓN DE LA BASE Y CONSULTAS REALIZADAS

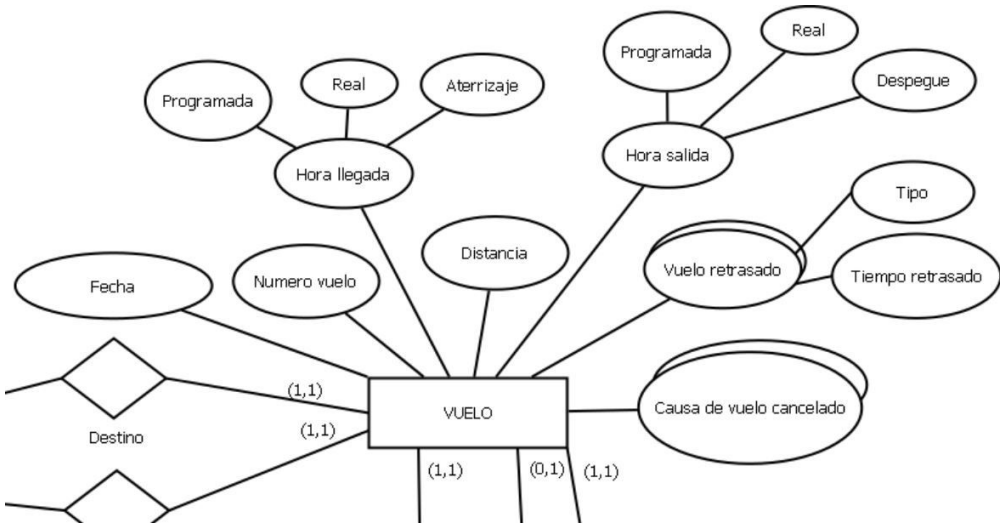
PARTE 3: OPTIMIZACIÓN, DISEÑO FÍSICO Y TRIGGERS

ANEXOS

ESQUEMA ENTIDAD-RELACIÓN



A la hora de diseñar el esquema entidad-relación que hará alusión a la base de datos se decidió en primer lugar crear la entidad vuelo que haría referencia a un "vuelo físico", es decir, a la ejecución del propio vuelo, para así poder disponer en cada vuelo de todos los datos relevantes de los que se pudiera disponer (hora de salida, de llegada, de aterrizaje, etcétera).



También se determinó que la manera correcta de representar el origen y el destino de un vuelo sería mediante dos relaciones binarias entre la entidad vuelo y la entidad aeropuerto .

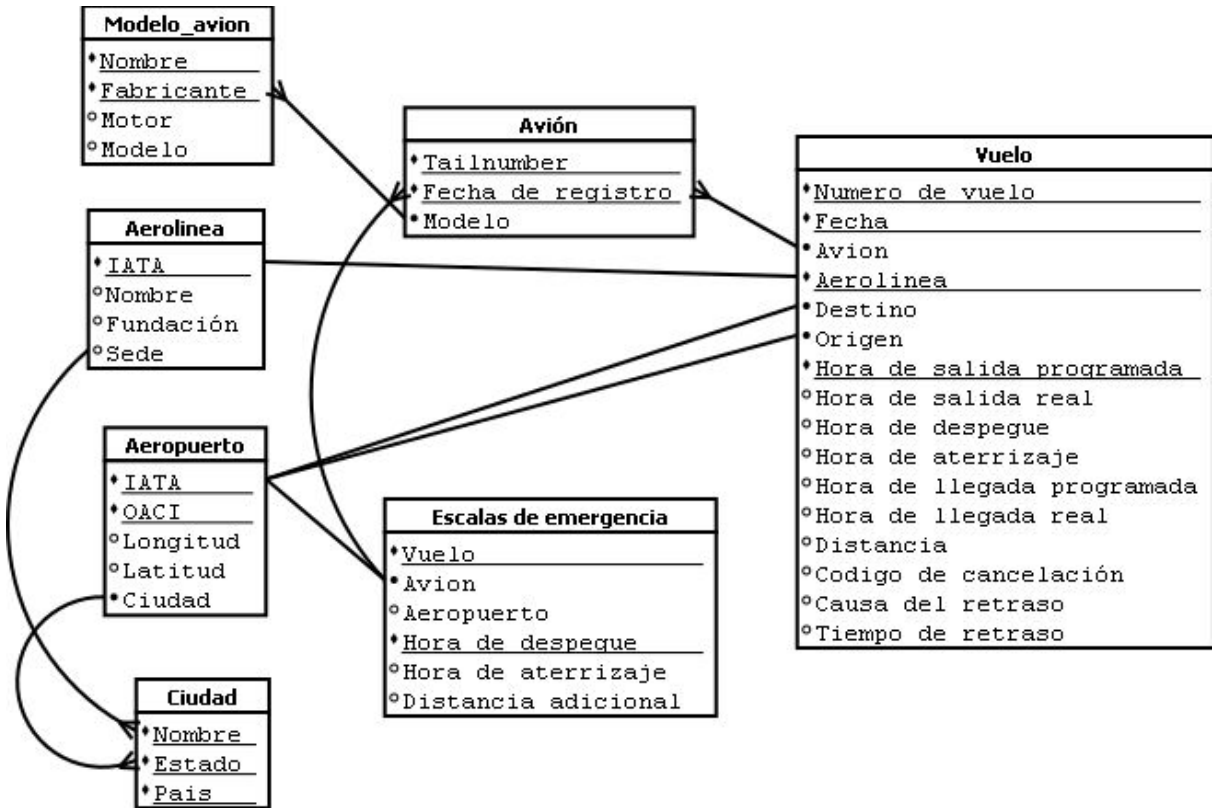
DECISIONES DE DISEÑO

Se tomo la decisión de que tan solo las escalas de emergencia que llevan a cabo los vuelos se representarían con una entidad, pues contienen diversos atributos y requieren relacionarse también con un aeropuerto, sin embargo si se concluyó que los tanto los vuelos retrasados, como los cancelados podían representarse como atributos multievaluados de la entidad vuelo, pues no contienen la información suficiente como para considerarlos entidades.

Ambas representaciones fueron debatidas, pero de optó por lo dicho dado que las dos tienen una traducción idéntica al modelo relacional pero representarlos como atributos y no como entidades refleja mejor la descripción dada acerca de un vuelo.
También se decidió separar a los aviones de su modelo, pues no bastaría con un atributo para contener toda la información necesaria acerca del modelo de un avión (fabricante, motor, etcétera).

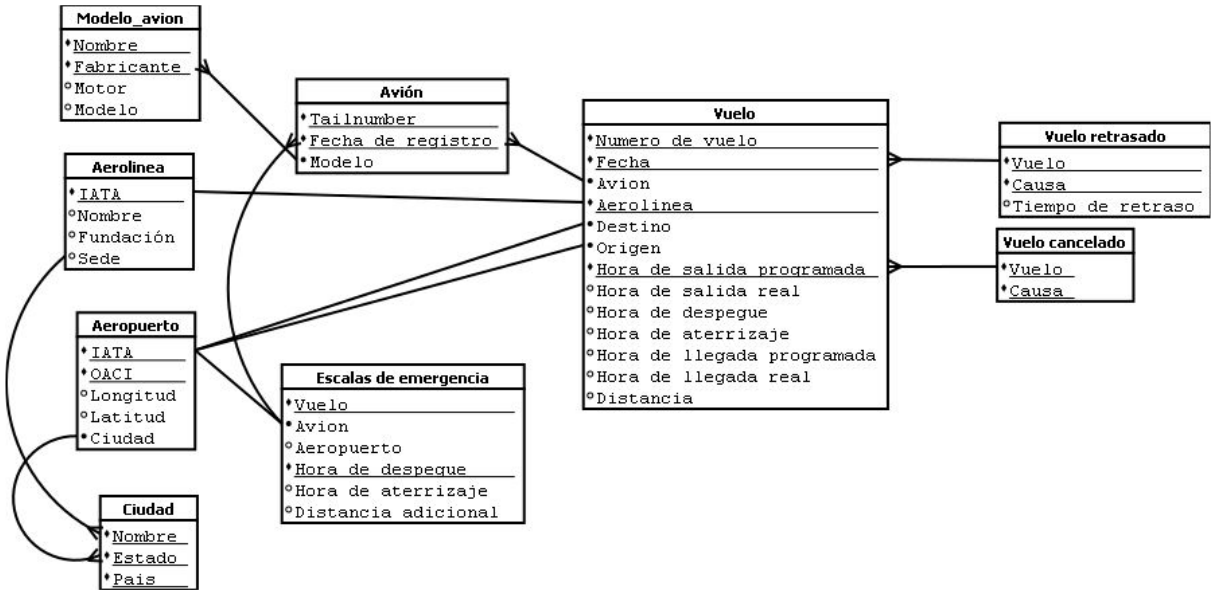
MODELO RELACIONAL Y PROCESO DE NORMALIZACIÓN

Modelo pre-normalizado



1ª Forma normal: Debido a la existencia de los atributos multi-evaluados "Causa_retraso", "Tiempo_retraso" y "Codigo de cancelación" en la entidad "Vuelo", el modelo no cumplía la primera forma normal, por lo que se crearon las tablas Vuelo_retrasado y Vuelo_cancelado con clave primaria compuesta por el vuelo, que referencia a la tabla Vuelo, y Tipo_retraso/Codigo_cancelación que son propias de las relaciones. Y en la tabla Vuelo_retrasado se añadió el atributo Tiempo_retraso.

Modelo en primera forma normal



2ª Forma normal: El modelo relacional se hallaba en segunda forma normal cuando fue analizado, dado que para todo atributo clave A no existía dependencia funcional con cualquier otro atributo también clave B, es decir, ningún atributo A clave depende funcionalmente de parte de la clave. Heurística: No es necesario analizar las relaciones con clave única, sólo aquellas con clave compuesta (varias claves candidata).

3ª Forma normal: El modelo relacional se hallaba en tercera forma normal cuando fue analizado, dado que para todo atributo clave A no existía dependencia funcional con cualquier otro atributo, o conjunto de atributos, no clave B, es decir, ningún atributo A clave depende funcionalmente de otro atributo de la relación o conjunto de atributos no clave. Heurística: No es necesario analizar las relaciones con clave única, sólo aquellas con uno o varios atributos no clave.

4ª Formal normal: El modelo relacional se hallaba en cuarta forma normal cuando fue analizado, dado que no existen dependencias multi-evaluadas entre cualquier conjunto de atributos A y otro conjunto de atributos B, es decir, los valores de B sí que dependen de los valores que tome A en un caso determinado.

RESTRICCIONES

No hemos adoptado ninguna restricción especialmente importante a la hora de diseñar nuestra base de datos.

TRADUCCIÓN DEL MODELO RELACIONAL NORMALIZADO A SQL

A la hora de crear las tablas en formato SQL se ha tenido que crear varias claves artificiales debido a que no se puede referenciar una clave compuesta desde otra tabla. Para evitar problema de repetición de la verdadera clave principal se ha usado la palabra clave UNIQUE, a esto se le ha unido la directiva not null en los atributos correspondientes para evitar que estos atributos pudiesen no tener valor.
Por otro lado se ha usado también la directiva on delete cascade sobre algunas referencias, la cual al eliminar el dato referenciado, elimina también la tupla que lo referencia, esto tiene sentido usarlo en tablas en los que una clave extranjera forma parte de la clave primaria.
También se ha usado la directiva on delete set null, la cual en vez de eliminar la tupla al eliminar el dato referenciado, elimina el valor del atributo en cuestión. Esto tiene sentido para casos en los que la clave extranjera no sea relevante en el significado total de la tupla.
Por último se ha usado también la directiva check, la cual comprueba al realizar una inserción o actualización que el atributo(s) en cuestión cumple(n) la condición "CHECK". Esto permite con la directiva check realizar las comprobaciones que de otro modo se tendrían que hacer mediante un trigger.

Las tablas se encuentran el la carpeta CREATE TABLES con el nombre CREATE_TABLES.sql.

POBLADO DE LA BASE DE DATOS

Para realizar el poblado de la base de datos primero se obtuvo mediante consultas los datos de la base de datos proporcionada en un formato similar al que tienen las tablas de nuestra base de datos. Una vez obtenidos estos datos en formato TSV hubo que modificarlos para adaptarlo al formato de nuestra base de datos debido entre otros factores al uso de claves artificiales. Por un lado, algunos de estos ficheros se trabajaron mediante shell scripts para conseguir transformarlos al formato necesario (un ejemplo de script utilizado se encuentra en la carpeta poblar base con el nombre script.sh).

Por otro lado, otro conjunto de datos se exportaron a una base de datos mysql y se trabajaron desde ahí.

Todos los datos insertados se encuentran tanto en forma de scripts sql como en formato tsv dentro de la carpeta poblar base.

En la base se insertaron toda la información que se encontraba en la base de datos proporcionada, por lo que en total se almacena toda la información referente a 48982 vuelos.

Los datos estadísticos son los siguientes:

Nombre de la Tabla	Número de tuplas	Peso en MB
VUELOS_CANCELADOS	403	0.0390625
VUELOS_RETRASADOS	12614	0.492734
ESCALAS_EMERGENCIAS	127	0.0390625
CIUDAD	3194	0.1406250
AVION	5333	0.1562500
MODELO_DE_AVION	187	0.0390625
AEROPUERTO	3376	0.1718750
AEROLINEA	1491	0.0781250
VUELO	48982	4.8671875

DISEÑO Y DESCRIPCIÓN DE LAS CONSULTAS SQL

Consultas obligatorias

Primera consulta obligatoria

Descripción: Número de vuelos desviados

Script SQL: La consulta en lenguaje SQL se encuentra dentro de la carpeta QUERIES con el nombre QUERY_1.sql

Plan de ejecución:

Explicación: Para resolver la consulta, se obtienen todos los estados que son fuente y destino de vuelos. Esto se hace uniendo las tablas que contienen datos de vuelos, vuelos retrasados, aeropuertos y ciudades. Al hacer la unión propuesta se obtiene para cada vuelo el estado del que parte y el estado al que se dirige. Finalmente, contando los resultados agrupados se obtienen los datos pedidos.

Álgebra relacional: El plan de ejecución de la consulta se encuentra dentro de la carpeta Stats con el nombre query1.txt

$$e = \Pi_{e.id}(\sigma_{e.vuelo=v.id}(escalas\_emergencias\ e))$$
$$a1 = \Pi_{estado}(\sigma_{e}(\bowtie_{v.id=r.vuelo \wedge a1.iata=v.origen \wedge a1.ciudad=c1.id}(vuelos\_retrasados\ r,\ vuelo\ v,\ aeropuerto\ a1,\ ciudad\ c1)))$$
$$a2 = \Pi_{estado}(\sigma_{e}(\bowtie_{v.id=r.vuelo \wedge a2.iata=v.destino \wedge a2.ciudad=c2.id}(vuelos\_retrasados\ r,\ vuelo\ v,\ aeropuerto\ a2,\ ciudad\ c2)))$$
$$\Pi_{num,\ estado}((estado)\zeta_{COUNT(*)\ num}\ (\cup_{ALL}\ (a1,\ a2))$$

Resultados (cuenta, estado):

8 AK	10 FL	1 MA	1 NH	14 PA
1 AL	8 GA	2 MD	2 NJ	3 TN
8 AZ	6 HI	4 MI	1 NM	11 TX
46 CA	2 ID	6 MN	4 NV	1 UT
5 CO	11 IL	1 MO	38 NY	1 VA
5 CT	3 KY	2 NC	3 OH	2 WA

Segunda consulta obligatoria

Descripción: Aeropuerto con los aviones más antiguos (mayor media de edad de los aviones que operan en el aeropuerto).
Se presupone que los aviones que operan en cada aeropuerto son aquellos que realizan trayectos con destino o con origen el aeropuerto concreto. No se tomarán como tal aquellos que solamente hayan hecho escala por emergencia en ese aeropuerto.

Script SQL: La consulta en lenguaje SQL se encuentra dentro de la carpeta QUERIES con el nombre QUERY_2.sql

Plan de ejecución: El plan de ejecución de la consulta se encuentra dentro de la carpeta Stats con el nombre querie2.txt

Explicación: Para resolver la consulta primero se obtuvieron los aviones que operaban en cada aeropuerto, a partir de esto se puede obtener la media de edad de los aviones que operaban en cada aeropuerto, por último se obtiene la media de edad más alta. Teniendo esta sólo falta ver a qué aeropuerto corresponde y proyectar ambos valores.

Álgebra relacional:

$(\pi_{\text{aeropuerto.nombre, media\_edad}}) - (\bowtie_{\text{aeropuerto = aeropuerto.iata AND maxima\_edad = media\_edad}}) -$
$(_{\text{(aeropuerto)}}\mathbf{f}_{\text{EXTRACT(YEAR FROM sysdate) - avg(fecha\_de\_registro) AS media\_edad}} - (\bowtie_{\text{avion = avion.id}}) -$
$((\mathbf{U} - ((\pi_{\text{avión, destino as aerolínea}}) - (\text{VUELO})), ((\pi_{\text{avión, origen as aerolínea}}) - (\text{VUELO}))), \text{AVION})),$
$\text{AEROPUERTO}, ((_{\text{(aeropuerto)}}\mathbf{f}_{\text{max (EXTRACT(YEAR FROM sysdate)) - avg(fecha\_de\_registro) as media\_edad}} - (\bowtie_{\text{avion = avion.id}} - ((\mathbf{U} - ((\pi_{\text{avión, destino as aerolínea}}) - (\text{VUELO})), ((\pi_{\text{avión, origen as aerolínea}}) -$
$(\text{VUELO}))), \text{AVION}))))$

Resultados:

NOMBRE	MEDIA_EDAD
Henry E. Rohlsen	28.5

Tercera consulta obligatoria:

Descripción: Por cada compañía aérea, lista aquellas otras compañías que cubren al menos el 30% de sus trayectos (aunque no coincidan en fechas y horarios), ordenándolo de mayor a menor porcentaje de cobertura.

Script SQL: La consulta en lenguaje SQL se encuentra dentro de la carpeta QUERIES con el nombre QUERY_3.sql

Plan de ejecución: El plan de ejecución de la consulta se encuentra dentro de la carpeta Stats con el nombre query3.txt

Explicación: Para llevar a cabo la consulta primero se han obtenido los distintos viajes que lleva a cabo cada aerolínea de un aeropuerto a otro, se ha realizado un conteo de estos y posteriormente se ha realizado otro conteo con las demás aerolíneas en aquellos trayectos que coincidan, obteniendo así el porcentaje dividiendo el número de coincidencias entre el total para cada aerolínea con las demás, proyectando los nombres de ambas si su coincidencia supera el 30%.

Álgebra relacional:

VIEW VUELOS_AEROLINEA

π (aerolinea, destino, origen - vuelo)

CONSULTA

($\pi_{A.nombre}$ AS aerolinea_1, $_{B.nombre}$ AS aerolinea_2, $_{coindicendias/numero\_vuelos * 100}$ AS porcent

- ($\bowtie_{aerolinea\_contar = aerolinea\_ppal}$ AND $_{((coincidencias/numero\_vuelos)>=0.3)}$

- ($\pi_{aerolinea}$ AS aerolinea_contar, f $_{COUNT(*)}$ AS numero_vuelos - vuelos_aerolinea))

- ($\bowtie_{aerolinea\_ppal=A.IATA}$ - aerolinea A) - ($\bowtie_{aerolinea\_ppal=B.IATA}$ - aerolinea B) -

($\pi_{a.aerolinea}$ AS aerolinea_ppal, $_{b.aerolinea}$ AS aerolinea_sec, f $_{COUNT(*)}$ AS coincidencias

- ($\bowtie_{a.destino=b.destino\ AND\ a.origen=b.origen\ AND\ a.aerolinea!=b.aerolinea}$ - vuelos_aerolinea B) - vuelos_aerolinea a))

Resultados:

AEROLINEA_1	AEROLINEA_2	PORCENT
Frontier Airlines Inc.	United Air Lines Inc.	80.2631579
Frontier Airlines Inc.	Southwest Airlines Co.	63.1578947
AirTran Airways Corporation	Delta Air Lines Inc.	40.6926407
Continental Air Lines Inc.	Expressjet Airlines Inc.	39.8058252
US Airways Inc	Southwest Airlines Co.	31.8681319

Consultas personales

Primera consulta personal

Descripción: Mayor potencia que cada tipo de motor ha conseguido. Por potencia se supondrá una mayor velocidad en mi/hora.

Script SQL: La consulta en lenguaje SQL se encuentra dentro de la carpeta QUERIES con el nombre QUERY_P_1.sql.

Plan de ejecución: El plan de ejecución de la consulta se encuentra dentro de la carpeta Stats con el nombre querie_p1.txt.

Explicación: Para obtener los datos de la consulta, se obtienen todos los vuelos y qué velocidad media han logrado (por medio de dos funciones), después se juntan esos datos con los datos de aviones y los modelos de los mismos. Finalmente se selecciona la máxima velocidad agrupadas por el tipo de motor.

Álgebra relacional:

$$v = \Pi_{avion,\ velocity(hora\_despegue,\ hora\_aterrizaje,\ distancia)}(vuelo)$$

$$j = \bowtie_{a.id=v.avion\ \wedge\ m.id=a.modelo}(avion\ a,\ modelo\_de\_avion\ m,\ v)$$

$$\Pi_{v,\ tipo\_motor}(\zeta_{ORDER\ BY\ v\ DESC}((tipo\_motor)\ \zeta_{MAX(vel)\ v}\ (j\ )))$$

Resultados:

V	TIPO_MOTOR
15.6	Turbo-Fan
14.5333333	Turbo-Jet
4.82222222	Reciprocating
2.05833333	Turbo-Prop
0.976851852	Turbo-Shaft
0.815942029	4 Cycle

Segunda consulta personal

Descripción: Devuelve el modelo de avión que más distancia ha recorrido así como esta distancia

Script SQL: La consulta en lenguaje SQL se encuentra dentro de la carpeta QUERIES con el nombre QUERY_P_2.sql.

Plan de ejecución: El plan de ejecución de la consulta se encuentra dentro de la carpeta Stats con el nombre querie_p2.txt.

Explicación: Para resolver esta consulta primero se obtuvo la distancia recorrida por cada modelo de avión sumando las distancias recorridas en vuelos de emergencia y las distancias recorridas en vuelos convencionales. Una vez obtenida esta distancia se obtuvo la mayor distancia recorrida por algún modelo y se obtuvo el modelo de avión asociado. Por último se proyectó el nombre y fabricante de este modelo.

Álgebra relacional:

$(\pi_{\text{modelo\_de\_avion.nombre, modelo\_de\_avion.fabricante, distancia}}) - ((\bowtie_{\text{modelo = modelo\_de\_avion .id AND distancia = maxima\_distancia}}) - (\text{MODELO\_DE\_AVION}) ,((_{\text{(avion)}}\mathbf{f}_{\text{SUM(distancia) AS distancia}}) - (\bowtie_{\text{avion = avion.id}}) - (\text{AVION}), (\mathbf{U} - ((_{\text{(avion)}}\mathbf{f}_{\text{SUM(distancia}})) - (\text{VUELO})),((_{\text{(avion)}}\mathbf{f}_{\text{SUM(distancia\_adicional) as distancia}}) - (\text{ESCALAS\_EMERGENCIAS})))) , ((_{()}\mathbf{f}_{\text{MAX(distancia) AS maxima\_distancia}})-((_{\text{(avion)}}\mathbf{f}_{\text{SUM(distancia) AS distancia}}) - (\bowtie_{\text{avion = avion.id}}) - (\text{AVION}), (\mathbf{U} - ((_{\text{(avion)}}\mathbf{f}_{\text{SUM(distancia)}}) - (\text{VUELO})),((_{\text{(avion)}}\mathbf{f}_{\text{SUM(distancia\_adicional) AS distancia }}) - (\text{ESCALAS\_EMERGENCIAS})))))$

Resultados:

NOMBRE	FABRICANTE	DISTANCIA
CL-600-2B19	BOMBARDIER INC	1937514

Tercera consulta personal

Consulta personal nº3 :

Descripción: Obtiene la(s) ciudad(es) que más escalas de emergencia reciben así como el total de la(s) distancia(s) adicional(es) que ha(n) causado la(s) escala(s), contando una escala de emergencia como aquella en la que es necesario recorrer una dada distancia adicional para llegar a la nueva ciudad.

Script SQL: La consulta en lenguaje SQL se encuentra dentro de la carpeta QUERIES con el nombre QUERY_P_3.sql.

Plan de ejecución: El plan de ejecución de la consulta se encuentra dentro de la carpeta Stats con el nombre querie_p3.txt.

Explicación: Primero se obtienen todas las ciudades en las cuales se ha realizado alguna escala así como la cuenta total del número del escalas realizadas sobre dicha ciudad, se separan aquellas que no sean escalas con una distancia adicional, y de las restantes se obtiene la que tiene un mayor número de escalas y se obtiene la distancia total recorrida por los aviones a esa ciudad en las escalas realizadas.

Álgebra relacional:

VIEW CIUDADES_ESCALAS

$(\pi_{\text{ciudad, distancia\_adiccional}} - (\bowtie_{\text{aeropuerto.iata = escalas\_emergencias.aeropuerto\_escala AND distancia\_adiccional > 0}} - \text{escalas\_emergencias}) - \text{aeropuerto})$

VIEW NUMERO_ESCALAS

$(\pi_{\text{ciudad, fCOUNT(*)}}\ \text{AS escalas,}\ _{\text{SUM(distancia\_adiccional)}}\ \text{AS distancia- ciudades\_escalas})$

VIEW MAX_ESCALAS

$(\ \pi_{\text{ciudad, escalas,distancia}}\ ) - (\ \sigma_{\text{escalas}} = (\ \pi_{\text{fMAX(escalas)}} - \text{numero\_escalas}\ ) - (\ \text{numero\_escalas})$

CONSULTA

$(\pi_{\text{nombre, escalas,distancia}} - (\bowtie_{\text{ciudad.id = max\_escalas.ciudad}} - \text{max\_escalas}) - \text{ciudad})$

Resultados:

NOMBRE	ESCALAS	DISTANCIA
Pittsburgh	2	602
Boston	2	182

DISEÑO FÍSICO Y TRIGGERS

Diseño general

Debido a las posibilidades que ofrece oracle al permitir añadir diversos atributos en las tablas como puede ser la opción check y que los problemas de inconsistencia de los datos que podía presentarse en el esquema original se podían resolver con ello, decidimos aprovecharlo, evitando crear triggers que hicieran algo que una condición de check podía realizar como puede ser a forma de ejemplo evitar introducir una hora de llegada de un vuelo superior a las 23:59 o inferior a 00:00.

También aprovechamos las posibilidades que nos ofrece para introducir cláusulas como on delete cascade y on delete null, las cuales te permiten de una forma sencilla mantener la coherencia en los datos.

Optimización de la consulta propuesta nº1

Tras realizar la consulta propuesta primera, se notó que tenía un tiempo de ejecución excesivo, de una media de más de 7 segundos (aún con índices), por lo que se propone realizar un trigger para reducirlo.

Del plan de ejecución se extrae que se realizan muchos bucles y dos accesos completos a la tabla vuelo (ya que se busca por separado los aeropuertos origen y destino). Estos dos últimos accesos están determinando el tiempo de ejecución de la consulta, ya que se accede a la tabla más grande, de más de 40000 registros.

Se detectó 4 puntos de actuación que necesitaba el trigger propuesto para optimizar la consulta, a saber:

1. Al registrar una incidencia en la tabla "*escalas_emergencias*".
2. Al actualizar una incidencia de la tabla "*escalas_emergencias*".
3. Al eliminar una incidencia de la tabla "*escalas_emergencias*".
4. Al actualizar un vuelo que ha tenido alguna incidencia.

Aparte, para reducir el código de los triggers se ha decidido crear funciones y procedimientos que simplificasen y abstrayesen la complejidad del problema, estos se usan para añadir un vuelo a un estado, eliminarlo y obtener el estado al que pertenece un aeropuerto (dado por el código IATA).

El plan de ejecución tras todas las optimizaciones se reduce a un acceso total a la tabla que actualiza el trigger, y que elimina 4 joins ya que los precalcula los triggers.
Finalmente, el tiempo de ejecución se reduce a medio segundo.

Conclusión
Nótese que el conjunto de triggers reducen notablemente el tiempo de ejecución a costa de incrementar ligeramente el tiempo al gestionar una incidencia o actualizar un vuelo.
El espacio en disco no va a suponer un problema, ya que sólo se guarda el estado y el número de vuelos desviados que tenían origen o destino en dicho estado, por lo que sólo supondrá 52 filas como máximo en el caso de EE.UU., y unos pocos KB de datos.

Por lo tanto, se considera que sí merece la pena el trigger.

Optimización de la consulta propuesta nº2

Esta consulta originalmente tenía un tiempo de ejecución de 5.8 segundos sin realizarse ninguna optimización sobre ella.

Primeramente se analizó el plan de ejecución y se optó por reestructurar el código estratégicamente para reducir el tiempo de ejecución, logrando reducirlo a 5.53 segundos, siendo el resultado de esta reestructuración el script sql que se encuentra en la carpeta QUERIES con el nombre QUERY_2.sql y el plan de ejecución asociado se encuentra en la carpeta Stats con el nombre querie2.txt.
Pero esta era una mejora poco sustancial por lo que se decidió explorar otras sendas para mejorarla.

La siguiente alternativa fué crear índices en la tabla vuelos, tanto en el atributo origen como en el atributo destino para intentar reducir el costo de una unión y el posterior group by que se produce al obtener los aviones que operaban en cada aeropuerto.

También se valoró y se probaron a crear índices en todos los elementos que se usasen en algún group by para así reducir su costo. Pero ninguna de las soluciones anteriores supuso una mejora en el rendimiento, en algunos casos el gestor ni siquiera usaba los índices, y en otros el tiempo consumido por la consulta o era el mismo, o en el mejor de los casos se reducía en una décima.
Por lo que la solución de mejorar el rendimiento de esta consulta sólo utilizando métodos que no afectasen al diseño de la base de datos inicial se descartó.

En ese momento se empezó a valorar la creación de tablas adicionales unidas a sus correspondientes trigger como posible solución, y viendo la traza de ejecución parecía indicar que cuando más recursos se consumían era al obtener los aviones que operaban en cada aeropuerto debido a que se producían varios TABLE ACCESS (FULL) OF 'VUELO' además de un SORT (UNIQUE).

Por ello se decidió comprobar el tiempo que tarda en ejecutarse la sentencia sql que obtenían los aviones que operaban en cada aeropuerto, obteniendo como resultado 1.7 segundos.

Por lo que se decidió disponer de una estructura en la cual se almacenaran los aviones que operaban en cada aeropuerto. Para llevarlo a cabo se podría optar por dos vías, la primera es crear una vista materializada y la segunda consiste en crear una tabla para almacenar estos valores y el correspondiente trigger que la mantenga actualizada. Debido a que no se disponen de los permisos suficientes como para

operar con vistas materializadas nos vimos forzados a optar por la segunda solución.

Una vez creada esta tabla se volvió a ejecutar la consulta (la nueva consulta que utiliza esa tabla se encuentra en QUERIES con el nombre QUERY_2_opt.sql) para ver el cambio en el rendimiento, y esta había pasado de costar 5.53 segundos en ejecutarse a costar 1.12 segundos, por lo que era aproximadamente 5 veces más rápida que la consulta inicial (el plan de ejecución se encuentra el la carpeta Stats con el nombre querie2_opt.txt).

Aunque la utilización de esta nueva tabla supone un aumento de espacio en disco (0.625 MB)y un ligero aumento en el tiempo de modificación de un dato en la tabla VUELOS, el aumento de rendimiento que esta mejora produce justificaría estos puntos.
Además el aumento en el tiempo de ejecución al modificar un dato era inapreciable (los planes de ejecución se encuentran el la carpeta Stats con los nombres antes_trigger_2.txt y despues_trigger_2.txt).

Una vez realizada esta tabla se volvió a intentar crear índices en los dos campos que tiene para intentar mejorar su rendimiento al realizar “group by”, pero de nuevo, estos no produjeron ninguna mejora de rendimiento.
Por ello se decidió atacar la sección de la consulta en la que se accedía a la tabla avión ya que esto produce un TABLE ACCESS (FULL) OF 'AVION'.
Primero se creó un índice en fecha el cual el gestor no usaba, por lo que se descartó.

Siendo que sólo se necesita acceder a dos campos de la tabla 'AVION' para esta consulta, se pensó en crear una partición vertical de estos campos para obtener una mejora de rendimiento, pero no disponemos de los permisos suficientes para realizarla, por lo que se decidió emular esa partición creando una tabla aparte y un trigger que la mantuviese coherente.

Una vez creada y ejecutada la consulta utilizando esta nueva tabla (el plan de ejecución se encuentra el la carpeta Stats con el nombre querie2_part_vertical_avion.txt) se vio que daba un aumento de rendimiento de 1 centésima, pasando a costar 1.11 segundos ejecutar la consulta. Debido a este escaso aumento de rendimiento se decidió que no estaba justificada la creación de esta nueva tabla con el trigger correspondiente así que se descartó.
El precálculo de joins no se tomó como una alternativa en ningún caso debido a que esto produciría un aumento significativo del espacio ocupado en disco.

Optimización de la consulta propuesta nº3:

Esta consulta fue realizada con un algoritmo inicial bastante ineficiente basado en la creación de diversas vistas que tenía un tiempo de ejecución de aproximadamente 10 segundos sin realizarse ninguna optimización sobre ella.

Se analizó el plan de ejecución y se opto por rediseñar el algoritmo de la consulta a una sola vista, lo cual redujo el tiempo de ejecución a 1'89 segundos, sin todavía realizar ningún tipo de diseño físico.

Tras la reestructuración del código se re-analizó el nuevo plan de ejecución y se determinó que el acceso a tres atributos de la tabla vuelo, para luego eliminar los repetidos generaba demasiadas lecturas innecesarias a la memoria (1400 aproximadamente), así como también varias lecturas en disco que son mucho más ineficientes.

Por lo tanto, se optó por realizar una partición física de la tabla "vuelo" para obtener en otra tabla "viaje" tan sólo los atributos necesarios para la ejecución de la consulta, pero en Oracle no disponemos de los permisos necesarios, por lo cual se decidió realizar un trigger que realizará una función similar a la de crear la tabla "viaje" con la aerolínea y sus distintos viajes (distinto origen y destino) así como la cuenta de viajes que existe en la tabla vuelo para poder mantenerla actualizada correctamente.

Esta tabla supuso un aumento del rendimiento de la consulta, reduciendo su tiempo de ejecución a 0,79 segundos, menos de la mitad del tiempo anterior, y aunque la utilización de esta nueva tabla suponga un aumento de espacio en disco y un aumento en el tiempo de inserción de un dato en la tabla vuelo, el aumento en el rendimiento justifica la "partición" realizada.

INFORME SOBRE REUNIONES, DIVISIÓN DEL TRABAJO Y PROBLEMAS RELATIVOS A LA COORDINACIÓN DEL GRUPO

Jorge

PARTE 1

Día	Horas Invertidas	Tarea
11/05	1	Desarollo de nueva versión del esquema E/R
11/05	1	Creación fichero SQL tablas
13/05	2	Normalización y rediseño esquema E/R a v.final
28/05	2	Desarrollo memoria parte 1

PARTE 2

Día	Horas Invertidas	Tarea
12/05	1	Reparto consultas y análisis
17/05	2	Desarollo consulta personal
18/05	2	Desarrolo consulta obligatoria
28/05	3	Desarrollo memoria parte 2

PARTE 3

Día	Horas Invertidas	Tarea
27/05	4	Desarrollo trigger para optimizar 3 consulta
28/05	4	Desarrollo memoria parte 3
Total horas	23	Parte 1: 6 Parte 2: 9 Parte 3: 8

Abel

PARTE 1

Día	Horas Invertidas	Tarea
--/--	4	Desarrollo de nueva versión del esquema E/R
--/--	20 minutos	Corrección y mejoras del fichero SQL de tablas
--/--	2	Normalización y rediseño esquema E/R a versión final
--/--	2	Desarrollo memoria parte 1

PARTE 2

Día	Horas Invertidas	Tarea
--/--	4	Poblado de la base de datos
--/--	1	Reparto consultas y análisis
--/--	30 minutos	Desarrollo consulta personal
--/--	1	Desarrollo consulta obligatoria

--/--	3	Desarrollo memoria parte 2

PARTE 3

Día	Horas Invertidas	Tarea
--/--	4	Análisis y testeo de posibles mejoras sobre la consulta nº2
--/--	30 minutos	Desarrollo trigger para optimizar consulta nº2
--/--	2	Desarrollo memoria parte 3
Total horas	24.4	Parte 1: 8,4 Parte 2: 9,5 Parte 3: 6,5

Ángel

PARTE 1

Día	Horas Invertidas	Tarea
--/--	2	Desarrollo de nueva versión del esquema E/R
--/--	1	Normalización y rediseño esquema E/R a versión final
--/--	2	Desarrollo memoria parte 1

PARTE 2

Día	Horas Invertidas	Tarea
--/--	3	Poblado de la base de datos
--/--	1	Reparto consultas y análisis
--/--	1	Desarrollo consulta personal nº1
--/--	1	Desarrollo consulta obligatoria nº1
--/--	2	Desarrollo memoria parte 2

PARTE 3

Día	Horas Invertidas	Tarea
--/--	2	Desarrollo trigger para optimizar consulta nº1
--/--	3	Desarrollo memoria parte 3
Total horas	18	Parte 1: 5 Parte 2: 8 Parte 3: 5