Memoria Práctica 2
Bases de Datos

Curso 24/25

Carlos Alejos Fumanal - 872342@unizar.es
Mario Caudevilla Ruiz - 870421@unizar.es
Rodrigo Dávila Duarte - 872715@unizar.es

Grupo T15

Ingeniería Informática

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

4 de Mayo, 2025

Contenidos

1 | Creación de la base de datos

1.1 Esquema E/R

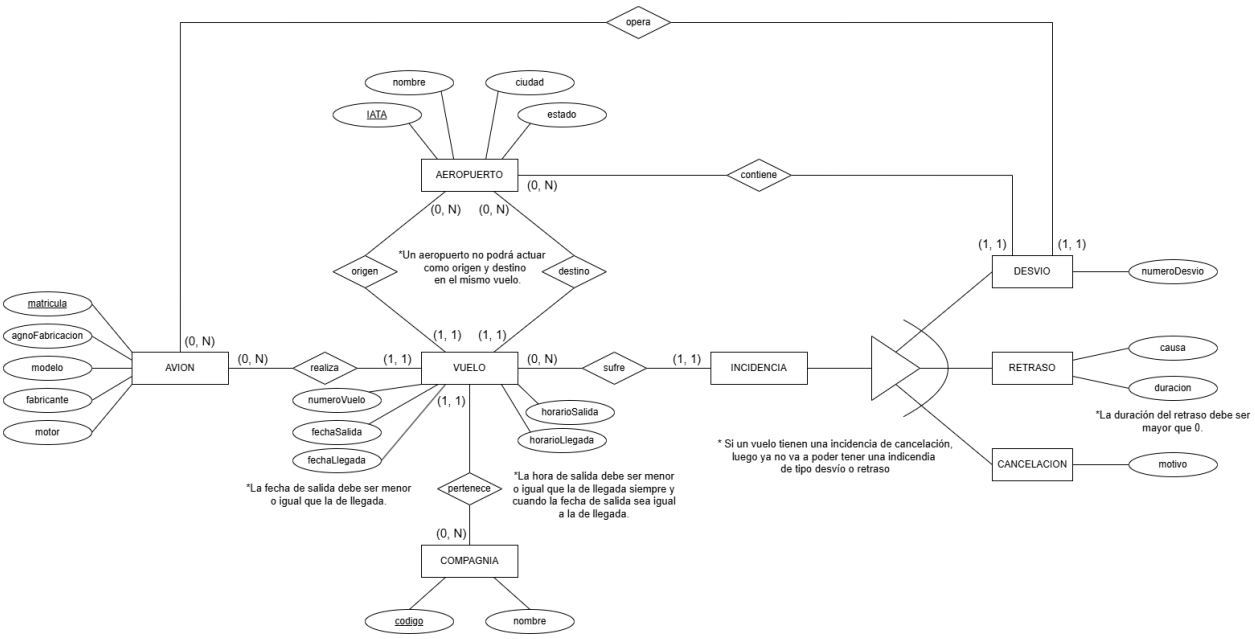

Figura 1: Esquema Entidad-Relación

Se han aplicado las siguientes restricciones para evitar conflictos que se podrían dar con el esquema E/R creado:

- Un aeropuerto no podrá actuar como origen y destino en el mismo vuelo.
- En un vuelo la fecha de salida debe ser menor o igual que la de llegada. Además, la hora de salida debe ser menor o igual que la de llegada siempre y cuando la fecha de salida sea igual a la de llegada.
- Un vuelo cancelado no puede tener posteriormente incidencias de tipo desvío o retraso.
- La duración de una incidencia de tipo retraso siempre será mayor que 0.

El esquema permite que se den ciertas circunstancias que hemos asumido como posibles:

- Un avión puede no tener ningún vuelo asociado.
- Cada vuelo deberá ser llevado a cabo por un avión.
- Cada vuelo tendrá una compañía asociada.
- Un vuelo puede no tener ninguna incidencia asociada.
- Un aeropuerto puede no tener ningún vuelo asociado.
- Un aeropuerto puede ser origen y destino de múltiples vuelos y destino de varios desvíos.
- Una incidencia no podrá ser al mismo tiempo retraso, cancelación y desvío debido a la especialización disjunta.
- Todo desvío tendrá un aeropuerto de destino y un avión para llevarlo a cabo.

Se ha añadido el atributo numeroDesvio en la entidad Desvío para indicar el orden en que ocurrieron, incluso si la tabla no está ordenada.

1.2 Esquema Relacional

1.2.1 Esquema relacional no normalizado

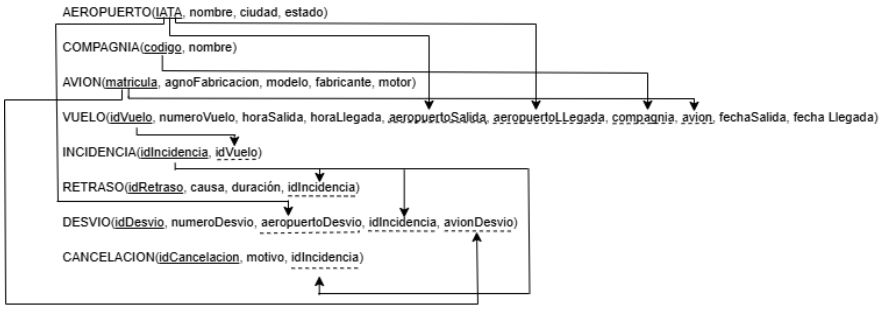

Figura 2: Esquema Relacional sin normalizar

1.2.2 Esquema relacional normalizado

Primero de todo, se ha comprobado si estaba en primera forma normal (1FN) y para ello se ha verificado que todos los atributos tuviesen sólo un valor posible. Dado que ningún atributo es multivaluado, se cumple que el esquema está en 1FN.
Respecto a la segunda forma normal (2FN), se cumple que todas las tablas están en 1FN y no tienen dependencias parciales ya que no hay claves compuestas.
En cuanto a la tercera forma normal (3FN), se cumple que todas las tablas están en 2FN y se debe comprobar que no tienen dependencias funcionales transitivas, es decir, que ningún atributo no clave dependa de otro atributo no clave. Como se puede observar en la tabla AVIÓN, con el atributo modelo se puede sacar el fabricante y el motor, por lo que aquí hay una dependencia funcional transitiva. Para solucionar esto, se ha creado otra tabla llamada MODELO en la cual la clave primaria es modelo.
Por último, se ha comprobado que el esquema esté en FNBC ya que no existen dependencias funcionales que no partan de la clave, tal y como se ha comprobado anteriormente en la 3FN.

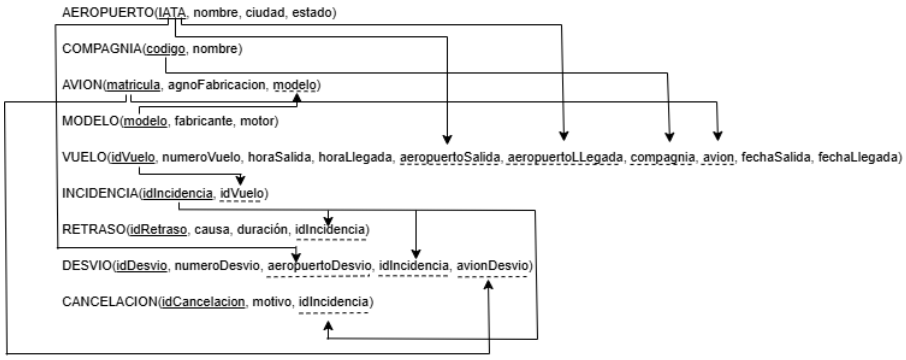

Figura 3: Esquema Relacional normalizado

Se ha decidido representar la especialización disjunta mediante cuatro tablas: una para la entidad Incidencia y una para cada tipo específico. Aunque este enfoque requiere más joins, permite una mejor organización y claridad de los datos. Se descartó la opción de una única tabla con todos los atributos, ya que implicaría muchos valores nulos y dificultaría la incorporación de nuevos tipos de incidencia, a pesar de que a la hora de consultar los datos pueda ser más eficiente.

1.3 Sentencias SQL para la creación de tablas

```sql
CREATE TABLE AEROPUERTO (
    IATA VARCHAR(4) PRIMARY KEY,
    nombre VARCHAR(64) NOT NULL,
    ciudad VARCHAR(64) NOT NULL,
    estado VARCHAR(8) NOT NULL
);
CREATE TABLE COMPAGNIA (
    codigo VARCHAR(8) PRIMARY KEY,
    nombre VARCHAR(128) NOT NULL
);
CREATE TABLE MODELO (
    nombre VARCHAR(32) PRIMARY KEY,
    fabricante VARCHAR(32) NOT NULL,
    motor VARCHAR(32) NOT NULL
);
CREATE TABLE AVION (
    matricula VARCHAR(8) PRIMARY KEY,
    agnoFabricacion NUMBER(4),
    modelo VARCHAR(32),
    FOREIGN KEY (modelo) REFERENCES MODELO(nombre)
);
CREATE TABLE VUELO (
    idVuelo NUMBER(8) PRIMARY KEY,
    numeroVuelo NUMBER(8) NOT NULL,
    fechaSalida VARCHAR(10) NOT NULL,
    fechaLlegada VARCHAR(10) NOT NULL,
    horaSalida VARCHAR(4) NOT NULL,
    horaLlegada VARCHAR(4) NOT NULL,
    aeropuertoSalida VARCHAR(4) NOT NULL,
    aeropuertoLlegada VARCHAR(4) NOT NULL,
    compagnia VARCHAR(8) NOT NULL,
    avion VARCHAR(8) NULL,
    FOREIGN KEY (aeropuertoSalida) REFERENCES AEROPUERTO(IATA),
    FOREIGN KEY (aeropuertoLlegada) REFERENCES AEROPUERTO(IATA),
    FOREIGN KEY (compagnia) REFERENCES COMPAGNIA(codigo),
    FOREIGN KEY (avion) REFERENCES AVION(matricula),
    CONSTRAINT chk_vuelo_unico UNIQUE (fechaSalida, fechaLlegada, horaSalida, horaLlegada, avion),
    CONSTRAINT chk_aeropuertos_distintos CHECK ( aeropuertoSalida <> aeropuertoLlegada ),
    CONSTRAINT chk_formato_fechas CHECK (
        LENGTH(fechaSalida) = 10 AND TO_DATE(fechaSalida, 'DD/MM/YYYY') IS NOT NULL
        AND
        LENGTH(fechaLlegada) = 10 AND TO_DATE(fechaLlegada, 'DD/MM/YYYY') IS NOT NULL
    ),
    CONSTRAINT chk_coherencia_fechas CHECK (
        (TO_DATE(fechaSalida, 'DD/MM/YYYY') < TO_DATE(fechaLlegada, 'DD/MM/YYYY'))
        OR
        (TO_DATE(fechaSalida, 'DD/MM/YYYY') = TO_DATE(fechaLlegada, 'DD/MM/YYYY')
         AND TO_NUMBER(horaSalida) <= TO_NUMBER(horaLlegada))
    )
);
CREATE TABLE INCIDENCIA (
    idIncidencia NUMBER(8) PRIMARY KEY,
    idVuelo NUMBER(8) NOT NULL,
    FOREIGN KEY (idVuelo) REFERENCES VUELO(idVuelo)
);
CREATE TABLE RETRASO (
    idRetraso NUMBER(8) PRIMARY KEY,
    causa VARCHAR(32) NOT NULL,
    duracion NUMBER(8) NOT NULL,
    idIncidencia NUMBER(8) NOT NULL,
    FOREIGN KEY (idIncidencia) REFERENCES INCIDENCIA(idIncidencia),
    CONSTRAINT chk_idIncidencia_retraso UNIQUE ( idIncidencia ),
    CONSTRAINT chk_duracion CHECK ( duracion > 0 )
);
CREATE TABLE DESVIO (
    idDesvio NUMBER(8) PRIMARY KEY,
```

```
    numeroDesvio NUMBER(2) NOT NULL,
    aeropuertoDesvio VARCHAR(4) NOT NULL,
    avionDesvio VARCHAR(8) NULL,
    idIncidencia NUMBER(8) NOT NULL,
    FOREIGN KEY (idIncidencia) REFERENCES INCIDENCIA(idIncidencia),
    FOREIGN KEY (aeropuertoDesvio) REFERENCES AEROPUERTO(IATA),
    FOREIGN KEY (avionDesvio) REFERENCES AVION(matricula),
    CONSTRAINT chk_idIncidencia_desvio UNIQUE ( idIncidencia )
);
CREATE TABLE CANCELACION (
    idCancelacion NUMBER(8) PRIMARY KEY,
    motivo VARCHAR(32) NOT NULL,
    idIncidencia NUMBER(8) NOT NULL,
    FOREIGN KEY (idIncidencia) REFERENCES INCIDENCIA(idIncidencia),
    CONSTRAINT chk_idIncidencia_cancelacion UNIQUE ( idIncidencia )
);
CREATE SEQUENCE secVuelo
    START WITH 1
    INCREMENT BY 1;
CREATE OR REPLACE TRIGGER trg_vuelo_id
BEFORE INSERT ON VUELO
FOR EACH ROW
BEGIN
    :NEW.idVuelo := secVuelo.NEXTVAL;
END;
/
CREATE SEQUENCE secIncidencia
    START WITH 1
    INCREMENT BY 1;
CREATE OR REPLACE TRIGGER trg_incidencia_id
BEFORE INSERT ON INCIDENCIA
FOR EACH ROW
BEGIN
    :NEW.idIncidencia := secIncidencia.NEXTVAL;
END;
/
CREATE SEQUENCE secRetraso
    START WITH 1
    INCREMENT BY 1;
CREATE OR REPLACE TRIGGER trg_retraso_id
BEFORE INSERT ON RETRASO
FOR EACH ROW
BEGIN
    :NEW.idRetraso := secRetraso.NEXTVAL;
END;
/
CREATE SEQUENCE secDesvio
    START WITH 1
    INCREMENT BY 1;
CREATE OR REPLACE TRIGGER trg_desvio_id
BEFORE INSERT ON DESVIO
FOR EACH ROW
BEGIN
    :NEW.idDesvio := secDesvio.NEXTVAL;
END;
/
CREATE SEQUENCE secCancelacion
    START WITH 1
    INCREMENT BY 1;
CREATE OR REPLACE TRIGGER trg_cancelacion_id
BEFORE INSERT ON CANCELACION
FOR EACH ROW
BEGIN
    :NEW.idCancelacion := secCancelacion.NEXTVAL;
END;
/
```

Código 1: Creación de tablas en SQL

2 | Introducción de datos y ejecución de consultas

2.1 Pasos para poblar la base de datos

Para poblar la base de datos, se ha optado por usar archivos .csv, que luego han sido añadidos a la base de datos a través de archivos .ctl específicos para cada uno de ellos. Lo primero que se hizo fue la extracción de la información correspondiente a cada tabla en su respectivo .csv. Para extraer los datos se ha empleado Excel, comandos de Linux (awk, sort, uniq, ...) y también un archivo de Python para hacer que cada fila de INCIDENCIA solo tenga una única incidencia, por ejemplo, si un vuelo tiene dos tipos de retraso, el archivo lo que hace es separar la fila en dos, haciendo que una fila tenga uno de los retrasos y el otro a 0 y viceversa. Asimismo, también se desarrolló otro archivo de Python para calcular e insertar los datos de la fechas de llegada de los vuelos. Este calcula la duración prevista del vuelo a partir de la hora de salida y la hora de llegada. En base a esa duración y la fecha de salida proporcionada, determina la fecha estimada de llegada. Estos archivos se encuentran en el directorio Ficheros_Apoyo de la entrega.

Cada archivo .csv contiene una columna por cada atributo (separados por ";") de la tabla correspondiente, excluyendo los identificadores propios (id), ya que estos se generan automáticamente mediante secuencias. Para introducir en las tablas las claves extranjeras (id de otras tablas), se han generado temporalmente estos identificadores en Excel, asociándolos manualmente a las filas correspondientes en la tabla que los utiliza como clave foránea.
Es importante también tener en cuenta que, para que no se produzcan fallos a la hora de asociar los id's generados por Oracle (secuencias) con los id's que hemos creado artificialmente nosotros en Excel, hay que vigilar que nunca se poble una tabla con una secuencia que no esté borrada, ya que si la secuencia ya se encuentra inicializada, cuando se poble la tabla, la secuencia continuará poblando el campo del id con el valor que se encontraba último sin asignar, por lo que los id's generados por nosotros en Excel no cuadrarían con los generados por Oracle con la secuencia. Para evitar este problema, hay que asegurarse de que cada vez que se vaya a poblar una tabla, se reinicie la secuencia que genera los id's de dicha tabla (*DROP SEQUENCE secTabla*), es decir, borrarla y volverla a crear justo antes de poblar para que se generen los id's correctos.

Después de extraer todos los datos del archivo inicial DatosVuelo.csv y distribuirlos en los correspondientes archivos .csv, el siguiente paso consistió en crear los archivos .ctl (Control Files). Estos archivos son procesados por SQL*Loader. Cada archivo .ctl establece la estructura del archivo .csv correspondiente y especifica la manera en que los datos deben ser introducidos en la base de datos. A continuación se presenta la estructura de los archivos de control utilizados para cargar la información de los .csv:

```
load data
infile './<nombreArchivo>.ctl'
into table <tablaCorrespondiente>
fields terminated by ';'
(<atributos separados por comas>)
```

Finalmente, tras la creación de los archivos .ctl, se puso en marcha SQL*Loader para introducir los datos en la base de datos Oracle mediante el comando: sqlldr a<*NIP*>@barret.danae04.unizar control=<*nombreArchivo.ctl*>. Este comando se ejecutó para cada tabla de la BD, siguiendo un orden concreto para no crear tablas que guardan claves extranjeras antes que las tablas a las que hacen referencia. Dado que resultaba muy incómodo ejecutar manualmente el comando cada vez que se quería poblar una tabla, se decidió crear un script que ejecuta dicho comando junto con la contraseña requerida por Oracle para poblar todas las tablas de forma automática en un solo paso. Este script se encuentra en el directorio Ficheros_Apoyo de la entrega.

Cabe destacar que, debido a la incorporación del atributo numeroDesvio, el cual permite contabilizar el orden en que ocurren los desvíos de cada vuelo, se ha desarrollado una consulta que se ejecuta posteriormente a las inserciones en la tabla Desvío. Esta consulta detecta si en alguno de los registros insertados el valor del campo numeroDesvio es 0, y en tal caso, lo actualiza automáticamente con el número correspondiente en función del orden de los desvíos del vuelo asociado. Esta consulta se encuentra en el directorio Ficheros_SQL de la entrega.
Por lo tanto, si un usuario desea insertar un nuevo desvío pero desconoce qué número de desvío le corresponde, puede establecer provisionalmente el valor 0 en el campo numeroDesvio. Posteriormente, deberá ejecutar la consulta mencionada para que el sistema actualice dicho número de forma adecuada.

2.2 Consultas SQL

2.2.1 Consulta 1

Lista todas las compañías aéreas, de más a menos puntual (según su media de retrasos, que también debe listarse), siempre que hayan operado al menos 1000 vuelos cada día.

```sql
WITH VuelosPorCompaniaPorDia AS (   -- Contar vuelos de cada compagnia por dia
    SELECT v.compagnia, v.fechaSalida, COUNT(*) AS vuelos_por_dia
    FROM VUELO v
    GROUP BY v.compagnia, v.fechaSalida
),
DiasDistintos AS (    -- Obtener el numero total de dias distintos
    SELECT COUNT(DISTINCT fechaSalida) AS total_dias FROM VUELO
),
CompaniasCalificadas AS (   -- Seleccionar compagnias que tuvieron 1000+ vuelos en TODOS los dias
    SELECT vpcd.compagnia
    FROM VuelosPorCompaniaPorDia vpcd
    WHERE vpcd.vuelos_por_dia >= 1000
    GROUP BY vpcd.compagnia
    HAVING COUNT(*) = (SELECT total_dias FROM DiasDistintos)
),
RetrasosPorVuelo AS (   -- Obtener la duracion del retraso para cada vuelo (solo si existe retraso)
    SELECT v.idVuelo, v.compagnia, r.duracion
    FROM VUELO v
    JOIN INCIDENCIA i ON v.idVuelo = i.idVuelo
    JOIN RETRASO r ON i.idIncidencia = r.idIncidencia
    WHERE v.compagnia IN (SELECT compagnia FROM CompaniasCalificadas)
),
TotalRetrasosPorCompania AS (   -- Sumar las duraciones de los retrasos por compagnia
    SELECT compagnia, SUM(duracion) AS total_minutos_retraso
    FROM RetrasosPorVuelo
    GROUP BY compagnia
),
TotalVuelosPorCompania AS (    -- Contar el total de vuelos por compangia
    SELECT compagnia, COUNT(*) AS total_vuelos
    FROM VUELO
    WHERE compagnia IN (SELECT compagnia FROM CompaniasCalificadas)
    GROUP BY compagnia
),
RetrasoCompletoCompania AS (    -- Combinar totales con retraso y sin retraso
    SELECT tvpc.compagnia, tvpc.total_vuelos, trpc.total_minutos_retraso
    FROM TotalVuelosPorCompania tvpc
    LEFT JOIN TotalRetrasosPorCompania trpc ON tvpc.compagnia = trpc.compagnia
),
PromediosConRetraso AS (    -- Calcular promedio para compagnias con retrasos
    SELECT compagnia, total_minutos_retraso / total_vuelos AS retraso_promedio
    FROM RetrasoCompletoCompania
    WHERE total_minutos_retraso IS NOT NULL
),
PromediosSinRetraso AS (    -- Asignar 0 como promedio para compagnias sin retrasos
    SELECT compagnia, 0 AS retraso_promedio
    FROM RetrasoCompletoCompania
    WHERE total_minutos_retraso IS NULL
),
PromediosCombinados AS (    -- Combinar ambos conjuntos
    SELECT * FROM PromediosConRetraso
    UNION ALL
    SELECT * FROM PromediosSinRetraso
)
SELECT c.nombre AS nombre_compania, pc.retraso_promedio AS retraso_promedio_por_vuelo_total
FROM PromediosCombinados pc
JOIN COMPAGNIA c ON pc.compagnia = c.codigo
ORDER BY retraso_promedio_por_vuelo_total ASC;
```

Código 2: Código de la Consulta 1

Nombre Compañía	Retraso Promedio
Southwest Airlines Co.	5.6
American Airlines Inc.	9.5
American Eagle Airlines Inc.	12.2
Skywest Airlines Inc.	12.4

Tabla 1: Resultados de la Consulta 1

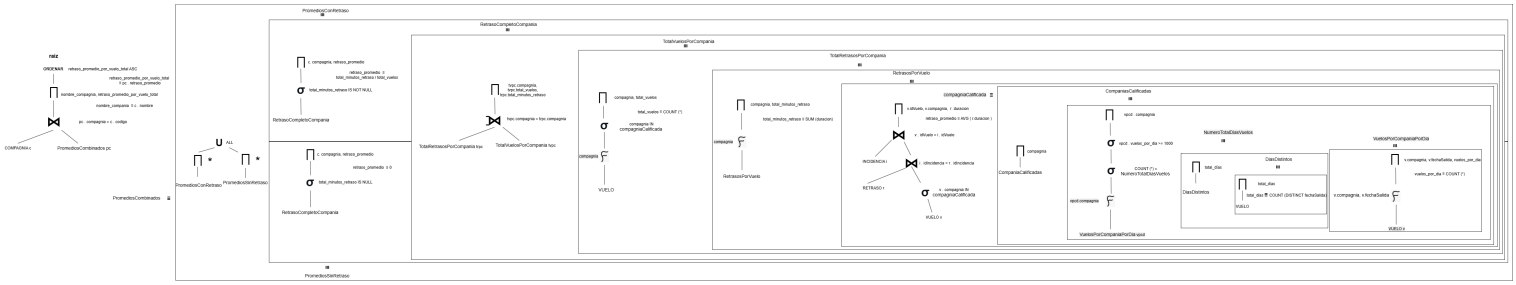

Figura 4: Árbol sintáctico de la consulta 1

2.2.2 Consulta 2

Listado de aeropuertos en Alaska o California en los que no opera la compañía con más aviones.

```sql
WITH CompagniaConMasAviones AS (
    -- Obtener las compagnias junto con el numero de aviones que operan y ordenarlas de mayor a menor
    SELECT v.compagnia, COUNT(DISTINCT v.avion) AS num_aviones
    FROM VUELO v
    GROUP BY v.compagnia
    ORDER BY num_aviones DESC
),
CompagniaPrincipal AS (
    -- Seleccionar la o las compagnias con mas aviones
    SELECT c.compagnia
    FROM CompagniaConMasAviones c
    WHERE c.num_aviones = (SELECT MAX(num_aviones) FROM CompagniaConMasAviones)
),
AeropuertosOperadosPorCompagniaPrincipal AS (
    -- Obtener los aeropuertos operados por la o las compagnias principales
    SELECT DISTINCT v.aeropuertoSalida AS codigo_aeropuerto
    FROM VUELO v
    JOIN CompagniaPrincipal cp ON v.compagnia = cp.compagnia
    UNION
    SELECT DISTINCT v.aeropuertoLlegada AS codigo_aeropuerto
    FROM VUELO v
    JOIN CompagniaPrincipal cp ON v.compagnia = cp.compagnia
)
SELECT a.IATA, a.nombre, a.ciudad, a.estado
FROM AEROPUERTO a
WHERE (a.estado = 'AK' OR a.estado = 'CA')
   AND a.IATA NOT IN (SELECT codigo_aeropuerto FROM AeropuertosOperadosPorCompagniaPrincipal
)
ORDER BY a.estado, a.ciudad, a.nombre;
```

Código 3: Código de la Consulta 2

Somos conscientes de que en el caso de que hubiese varias compañías empatadas con el número máximo de aviones, nuestra consulta mostraría el listado de aeropuertos en los que no opera ninguna de las compañías empatadas a más aviones. Esto ha sido probado de forma manual añadiendo tantos aviones como los que le faltan a la segunda compañía con más aviones para que se iguale a la primera y así ver los resultados.

IATA	Nombre	Ciudad	Estado
ADK	Adak	Adak	AK
ANC	Ted Stevens Anchorage International	Anchorage	AK
BRW	Wiley Post Will Rogers Memorial	Barrow	AK
BET	Bethel	Bethel	AK
CDV	Merle K (Mudhole) Smith	Cordova	AK
SCC	Deadhorse	Deadhorse	AK
FAI	Fairbanks International	Fairbanks	AK
JNU	Juneau International	Juneau	AK
KTN	Ketchikan International	Ketchikan	AK
ADQ	Kodiak	Kodiak	AK
OTZ	Ralph Wien Memorial	Kotzebue	AK
OME	Nome	Nome	AK
PSG	James C. Johnson Petersburg	Petersburg	AK
SIT	Sitka	Sitka	AK
WRG	Wrangell	Wrangell	AK
YAK	Yakutat	Yakutat	AK
ACV	Arcata	Arcata/Eureka	CA
BFL	Meadows	Bakersfield	CA
CIC	Chico Municipal	Chico	CA
CEC	Jack McNamara	Crescent City	CA
IPL	Imperial County	Imperial	CA
IYK	Inyokern	Inyokern	CA
LGB	Long Beach (Daugherty)	Long Beach	CA
MOD	Modesto City-County-Harry Sham	Modesto	CA
MRY	Monterey Peninsula	Monterey	CA
OAK	Metropolitan Oakland International	Oakland	CA
OXR	Oxnard	Oxnard	CA
PMD	Palmdale Production Flight	Palmdale	CA
RDD	Redding Municipal	Redding	CA
SBP	San Luis Obispo Co-McChesney	San Luis Obispo	CA
SBA	Santa Barbara Municipal	Santa Barbara	CA
SMX	Santa Maria Pub/Capt G Allan Hancock	Santa Maria	CA

Tabla 2: Resultados de la Consulta 2

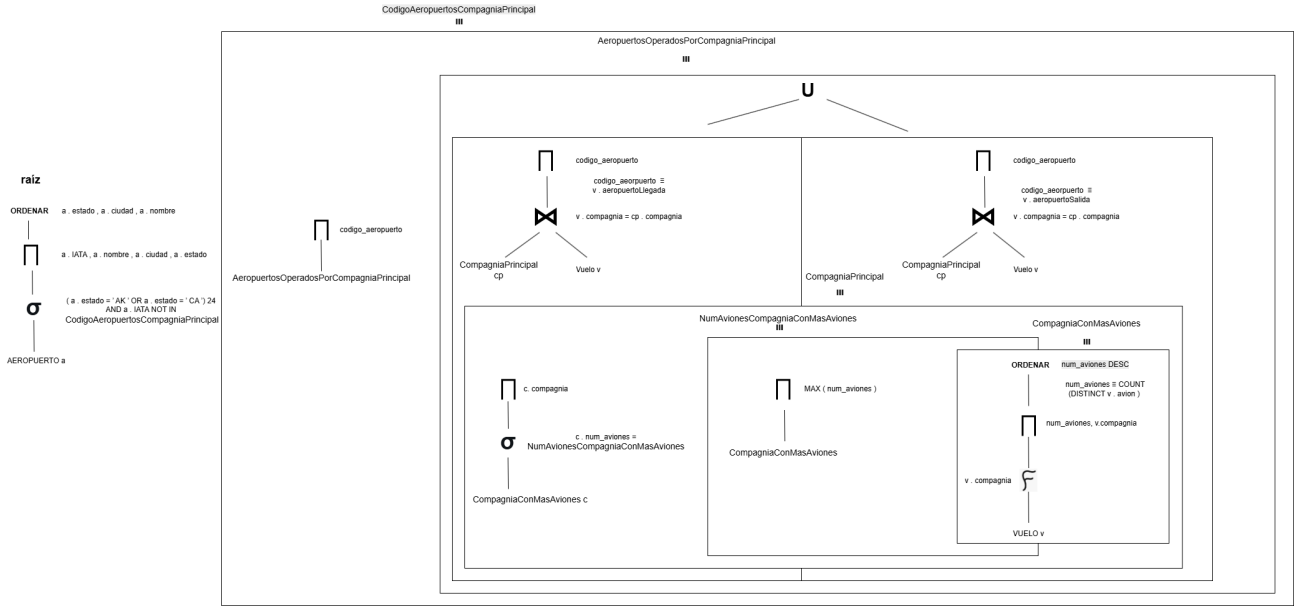

Figura 5: Árbol sintáctico de la consulta 2

2.2.3 Consulta 3

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

```sql
WITH AvionesEnAeropuertos AS (
    -- Recopila todos los aviones que han estado en cada aeropuerto (tanto salidas como llegadas)
    SELECT v.aeropuertoSalida AS codigo_aeropuerto, v.avion, a.agnoFabricacion
    FROM VUELO v
    JOIN AVION a ON v.avion = a.matricula
    UNION
    SELECT v.aeropuertoLlegada AS codigo_aeropuerto, v.avion, a.agnoFabricacion
    FROM VUELO v
    JOIN AVION a ON v.avion = a.matricula
),
EdadPromedioPorAeropuerto AS (
    -- Calcula la edad promedio de los aviones para cada aeropuerto
    SELECT codigo_aeropuerto, AVG(EXTRACT(YEAR FROM SYSDATE) - agnoFabricacion) AS edad_promedio
    FROM AvionesEnAeropuertos
    GROUP BY codigo_aeropuerto
),
AeropuertoConAvionesJovenes AS (
    -- Filtra solo los aeropuertos con la menor edad promedio (los mas nuevos)
    SELECT codigo_aeropuerto, edad_promedio
    FROM EdadPromedioPorAeropuerto
    WHERE edad_promedio = (SELECT MIN(edad_promedio) FROM EdadPromedioPorAeropuerto)
)
SELECT a.IATA, a.nombre, aaj.edad_promedio AS media_edad_aviones
FROM AeropuertoConAvionesJovenes aaj
JOIN AEROPUERTO a ON aaj.codigo_aeropuerto = a.IATA;
```

Código 4: Código de la Consulta 3

Somos conscientes de que EXTRACT (YEAR FROM SYSDATE) no está en la cheetsheet pero lo necesitamos para poder poner el año actual sin que sea un valor fijo marcado por la consulta, es decir que en lugar de poner: 2025 se pone EXTRACT (YEAR FROM SYSDATE). En caso de que no se quiera usar se podría poner simplemente: AVG(2025 - agnoFabricacion) AS edad_promedio.

IATA	Nombre	Edad Media Aviones
HDN	Yampa Valley	20.89

Tabla 3: Resultados de la Consulta 3

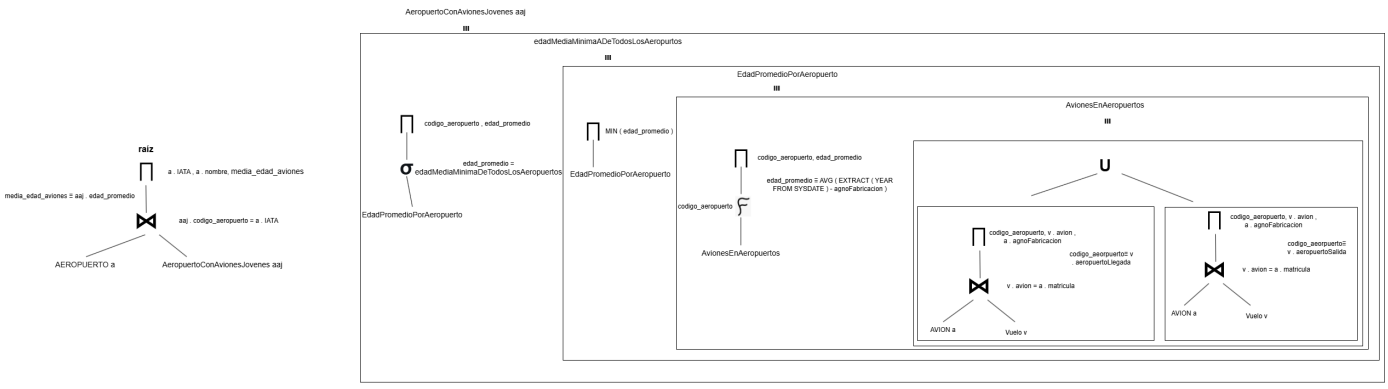

Figura 6: Árbol sintáctico de la consulta 3

3 | Diseño Físico

3.1 Problemas, acciones y mejoras de rendimiento

3.1.1 Consulta 1

(*Este es el Explain Plan de la Consulta 1 antes de ser optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |  1055  (33) | 00:00:01 |
```

Aunque el tiempo de ejecución real de la consulta es muy bajo, ya que los resultados se obtienen de forma casi instantánea, se ha detectado un elevado consumo de CPU. Este consumo se debe a que la consulta realiza dos exploraciones completas (TABLE ACCESS FULL) sobre la tabla VUELO, que contiene un volumen considerable de datos. Con el fin de optimizar el rendimiento y reducir la carga sobre el sistema, se ha decidido crear dos vistas materializadas que almacenen los resultados de dichas subconsultas.

```sql
-- Vista materializada para los vuelos por compagnia por dia
CREATE MATERIALIZED VIEW MV_VUELOS_POR_COMPAGNIA_DIA
REFRESH ON DEMAND
AS
SELECT v.compagnia, v.fechaSalida, COUNT(*) AS vuelos_por_dia
FROM VUELO v
GROUP BY v.compagnia, v.fechaSalida;

-- Vista materializada para retrasos
CREATE MATERIALIZED VIEW MV_RETRASOS_VUELO
REFRESH ON DEMAND
AS
SELECT v.idVuelo, v.compagnia, r.duracion
FROM VUELO v
JOIN INCIDENCIA i ON v.idVuelo = i.idVuelo
JOIN RETRASO r ON i.idIncidencia = r.idIncidencia;
```

Código 5: Vistas materializadas para la consulta 1

De esta forma, cada vez que se ejecuta la consulta principal, no se deberá recorre la tabla VUELO entera para encontrar los vuelos por día o los retrasos de los vuelos, sino que accederá a la vista para coger los resultados.

(*Este es el Explain Plan de la Consulta 1 optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |   168   (4) | 00:00:01 |
```

3.1.2 Consulta 2

(*Este es el Explain Plan de la consulta 2 antes de ser optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |    529   (3) | 00:00:01 |
```

En esta consulta ocurre una situación similar a la anterior: aunque los resultados se obtienen de forma inmediata al ejecutarla, el coste de CPU es elevado.
Al analizar el plan de ejecución, se observa nuevamente un acceso completo a la tabla VUELO (TABLE ACCESS FULL), lo que impacta negativamente en el rendimiento debido al gran volumen de datos. Para mitigar este problema, se ha implementado una vista materializada que evita el recorrido completo de la tabla, optimizando así el uso de recursos.

```sql
CREATE MATERIALIZED VIEW MV_COMPAGNIA_MAX_AVIONES
BUILD IMMEDIATE
REFRESH ON DEMAND AS
SELECT v.compagnia, COUNT(DISTINCT v.avion) as num_aviones
FROM VUELO v
GROUP BY v.compagnia
ORDER BY num_aviones DESC;
```

Código 6: Vista materializada para la consulta 2

Tal y como se observa en los resultados, con la vista materializada el coste de CPU disminuye, que era el objetivo principal de la mejora.

(*Este es el Explain Plan de la Consulta 2 optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |    290   (2) | 00:00:01 |
```

3.1.3 Consulta 3

(*Este es el Explain Plan de la Consulta 3 antes de ser optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |   2472  (1) | 00:00:01 |
```

En esta consulta, se observan múltiples accesos completos a tablas (TABLE ACCESS FULL) en VUELO y AVIÓN, que son operaciones costosas. Por otro lado, el uso de UNION y HASH JOIN, además de varias operaciones de ordenación y agrupación, elevan el coste total.
Para solucionar estos problemas, se ha creado una vista materializada que precalcula la parte de vuelos.

```sql
CREATE MATERIALIZED VIEW MV_AEROPUERTOS_AVIONES_JOVENES
REFRESH COMPLETE ON DEMAND
AS
SELECT v.aeropuertoSalida AS codigo_aeropuerto, v.avion, a.agnoFabricacion
FROM VUELO v
JOIN AVION a ON v.avion = a.matricula
UNION
SELECT v.aeropuertoLlegada AS codigo_aeropuerto, v.avion, a.agnoFabricacion
FROM VUELO v
JOIN AVION a ON v.avion = a.matricula;
```

Código 7: Vista materializada para la consulta 3

Con esta vista materializada se precalcula y almacena físicamente los resultados de la subconsulta que identifica todos los aviones de los aeropuertos, mejorando considerablemente el coste de CPU.

(*Este es el Explain Plan de la Consulta 3 optimizada.*)

```
| Id  | Operation            | Cost (%CPU) | Time     |
|-----|----------------------|-------------|----------|
|   0 | SELECT STATEMENT     |     76  (3) | 00:00:01 |
```

Cabe destacar que, en las tres consultas analizadas, se implementaron diversas mejoras como índices, índices bitmap y particiones. Sin embargo, ninguna de estas optimizaciones logró una reducción significativa en el uso de CPU, que era el objetivo principal, ya que el tiempo de ejecución de las consultas ya era muy reducido desde el inicio, incluso sin aplicar dichas mejoras.

3.2 Triggers

Para realizar los triggers, se ha revisado el script de creación de tablas, identificando restricciones no gestionables con CHECK. Tras este análisis, se han seleccionado los siguientes triggers.

3.2.1 Trigger 1

Si un vuelo tiene una incidencia de cancelación, no se puede insertar una nueva incidencia sobre ese vuelo.

```
CREATE OR REPLACE TRIGGER trg_desvio_coherencia
BEFORE INSERT ON DESVIO
FOR EACH ROW
DECLARE
    v_existe_cancelacion NUMBER := 0;
BEGIN
    -- Verificar si existe una cancelacion para el vuelo asociado a esta incidencia
    SELECT COUNT(*)
    INTO v_existe_cancelacion
    FROM CANCELACION c
    JOIN INCIDENCIA i ON c.idIncidencia = i.idIncidencia
    WHERE i.idVuelo = (SELECT idVuelo FROM INCIDENCIA WHERE idIncidencia = :NEW.idIncidencia);

    -- Si existe cancelacion, no permitir la insercion de desvio
    IF v_existe_cancelacion > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'No se puede registrar un desvio en un vuelo cancelado');
    END IF;
END;
/
```

Código 8: Trigger 1 (Desvío)

```
CREATE OR REPLACE TRIGGER trg_retraso_coherencia
BEFORE INSERT ON RETRASO
FOR EACH ROW
DECLARE
    v_existe_cancelacion NUMBER := 0;
BEGIN
    -- Verificar si existe una cancelacion para el vuelo asociado a esta incidencia
    SELECT COUNT(*)
    INTO v_existe_cancelacion
    FROM CANCELACION c
    JOIN INCIDENCIA i ON c.idIncidencia = i.idIncidencia
    WHERE i.idVuelo = (SELECT idVuelo FROM INCIDENCIA WHERE idIncidencia = :NEW.idIncidencia);

    -- Si existe cancelacion, no permitir la insercion de retraso
    IF v_existe_cancelacion > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'No se puede registrar un retraso en un vuelo cancelado');
    END IF;
END;
/
```

Código 9: Trigger 1 (Retraso)

3.2.2 Trigger 2

Comprueba que la nueva incidencia a insertar en cualquiera de las tablas hijas de INCIDENCIA no tenga un idIncidencia que ya exista en una fila de dichas tablas, ya que cada idIncidencia es único entre todas las subtablas, es decir, solo aparece en la tabla INCIDENCIA y en una de sus tablas hijas.

```
CREATE OR REPLACE TRIGGER trg_incidencia_exclusividad_cancelacion
BEFORE INSERT ON CANCELACION
FOR EACH ROW
DECLARE
    v_count_retraso NUMBER := 0;
    v_count_desvio NUMBER := 0;
BEGIN
    -- Verificar si la incidencia ya existe en RETRASO o en DESVIO o en CANCELACION
    SELECT COUNT(*) INTO v_count_retraso FROM RETRASO r WHERE r.idIncidencia = :NEW.idIncidencia;
    SELECT COUNT(*) INTO v_count_desvio FROM DESVIO d WHERE d.idIncidencia = :NEW.idIncidencia;

    -- Si ya existe en otra tabla, rechazar la insercion
    IF v_count_retraso > 0 THEN
        RAISE_APPLICATION_ERROR(-20107, 'La incidencia ' || :NEW.idIncidencia ||
                                ' ya esta registrada como retraso. Una incidencia solo puede ser de un
                                    tipo.');
    END IF;
    IF v_count_desvio > 0 THEN
        RAISE_APPLICATION_ERROR(-20108, 'La incidencia ' || :NEW.idIncidencia ||
                                ' ya esta registrada como desvio. Una incidencia solo puede ser de un tipo
                                    .');
    END IF;
END;
/
```

Código 10: Trigger 2 (Cancelación)

```
CREATE OR REPLACE TRIGGER trg_incidencia_exclusividad_desvio
BEFORE INSERT ON DESVIO
FOR EACH ROW
DECLARE
    v_count_retraso NUMBER := 0;
    v_count_cancelacion NUMBER := 0;
BEGIN
    -- Verificar si la incidencia ya existe en RETRASO o en CANCELACION o en DESVIO
    SELECT COUNT(*) INTO v_count_retraso FROM RETRASO r WHERE r.idIncidencia = :NEW.idIncidencia;
    SELECT COUNT(*) INTO v_count_cancelacion FROM CANCELACION c WHERE c.idIncidencia = :NEW.idIncidencia;

    -- Si ya existe en otra tabla, rechazar la insercion
    IF v_count_retraso > 0 THEN
        RAISE_APPLICATION_ERROR(-20104, 'La incidencia ' || :NEW.idIncidencia ||
                                ' ya esta registrada como retraso. Una incidencia solo puede ser de un
                                    tipo.');
    END IF;
    IF v_count_cancelacion > 0 THEN
        RAISE_APPLICATION_ERROR(-20105, 'La incidencia ' || :NEW.idIncidencia ||
                                ' ya esta registrada como cancelacion. Una incidencia solo puede ser de un
                                     tipo.');
    END IF;
END;
/
```

Código 11: Trigger 2 (Desvío)

```
CREATE OR REPLACE TRIGGER trg_incidencia_exclusividad_retraso
BEFORE INSERT ON RETRASO
FOR EACH ROW
DECLARE
    v_count_desvio NUMBER := 0;
    v_count_cancelacion NUMBER := 0;
BEGIN
    -- Verificar si la incidencia ya existe en DESVIO o en CANCELACION o en RETRASO
    SELECT COUNT(*) INTO v_count_desvio FROM DESVIO d WHERE d.idIncidencia = :NEW.idIncidencia;
    SELECT COUNT(*) INTO v_count_cancelacion FROM CANCELACION c WHERE c.idIncidencia = :NEW.idIncidencia;

    -- Si ya existe en otra tabla, rechazar la insercion
    IF v_count_desvio > 0 THEN
        RAISE_APPLICATION_ERROR(-20101, 'La incidencia ' || :NEW.idIncidencia ||
                                ' ya esta registrada como desvio. Una incidencia solo puede ser de un tipo
                                    .');
    END IF;
    IF v_count_cancelacion > 0 THEN
        RAISE_APPLICATION_ERROR(-20102, 'La incidencia ' || :NEW.idIncidencia ||
                                ' ya esta registrada como cancelacion. Una incidencia solo puede ser de un
                                     tipo.');
    END IF;
END;
/
```

Código 12: Trigger 2 (Retraso)

En estos triggers no se ha incluido también BEFORE UPDATE ya que se ha probado que si el trigger intentaba consultar la misma tabla que estaba siendo modificada durante la ejecución de este, Oracle lanzaba el error de tabla mutante.

3.2.3 Trigger 3

Comprueba que ningún aeropuerto de desvío es igual al aeropuerto de llegada original del vuelo.

```
CREATE OR REPLACE TRIGGER trg_aeropuerto_desvio_distinto_destino
BEFORE INSERT OR UPDATE ON DESVIO
FOR EACH ROW
DECLARE
    v_aeropuerto_llegada VARCHAR(4);
BEGIN
    -- Obtener el aeropuerto de llegada original del vuelo asociado a esta incidencia
    SELECT v.aeropuertoLlegada
    INTO v_aeropuerto_llegada
    FROM VUELO v
    JOIN INCIDENCIA i ON v.idVuelo = i.idVuelo
    WHERE i.idIncidencia = :NEW.idIncidencia;

    -- Comprobar si el aeropuerto de desvio es el mismo que el de llegada original
    IF :NEW.aeropuertoDesvio = v_aeropuerto_llegada THEN
        RAISE_APPLICATION_ERROR(-20201, 'El aeropuerto de desvio (' || :NEW.aeropuertoDesvio ||
                                    ') no puede ser el mismo que el aeropuerto de llegada original
                                        del vuelo.');
    END IF;
END;
/
```

Código 13: Trigger 3

4 | Anexo: Coordinación del grupo

A la hora de coordinarnos para hacer la BD no se ha tenido ningún problema significativo, esto puede deberse a la experiencia previa del equipo trabajando juntos y, por lo tanto, se conoce qué cosas se le dan mejor a cada uno, consiguiendo así una distribución más rápida y eficiente de las tareas. Además, hemos aprendido de todas las cosas que pudimos haber hecho mejor en la primera entrega y las hemos corregido.
Cuando ha sido necesario reunirse para discutir sobre decisiones a tomar, o para tareas que necesitaban de los 3 integrantes del grupo, el equipo se ha reunido tanto de forma presencial como de forma en línea mediante Discord.
Para asegurarse de que todos los integrantes del equipo se han enterado de todo, incluso de las partes en las que menos se ha trabajado, se han explicado mutuamente los avances realizados, forzando así una comprensión completa del desarrollo de la base de datos.

Tareas	**Carlos**	**Rodrigo**	**Mario**
Esquema E/R	8.5	7	8
Esquema Relacional	1.5	2	2,5
Creación de tablas	3.5	2.5	3
Población BD	15	7	9
Consulta 1	2.5	2.5	2.5
Consulta 2	2	1.5	1.5
Consulta 3	1	2	1
Rendimiento consultas	4.5	2,5	4,5
Trigger 1	2	2	2
Trigger 2	1.5	2	2.5
Trigger 3	1.5	1.5	1.5
Total	**43.5**	**32.5**	**38**

Tabla 4: Tabla de tareas asignadas a Carlos, Rodrigo, y Mario