UAM EPS

Memoria Practica1

Bases de Datos

Luis Fernández Fernández y Nicolás Ruiz Martínez

10-25-2023

Índice

Claves primarias y extranjeras de cada tabla:

aircrafts_data (**aircraft_code**, model)

airports_data (**airport_code**, airport_name, city, coordinates, timezone)

boarding_passes (**ticket_no** → ticket_flights. ticket_no, **flight_id** →ticket_flights.flight_id, boarding_no, seat_no)

bookings (**book_ref**, book_date, total_amount)

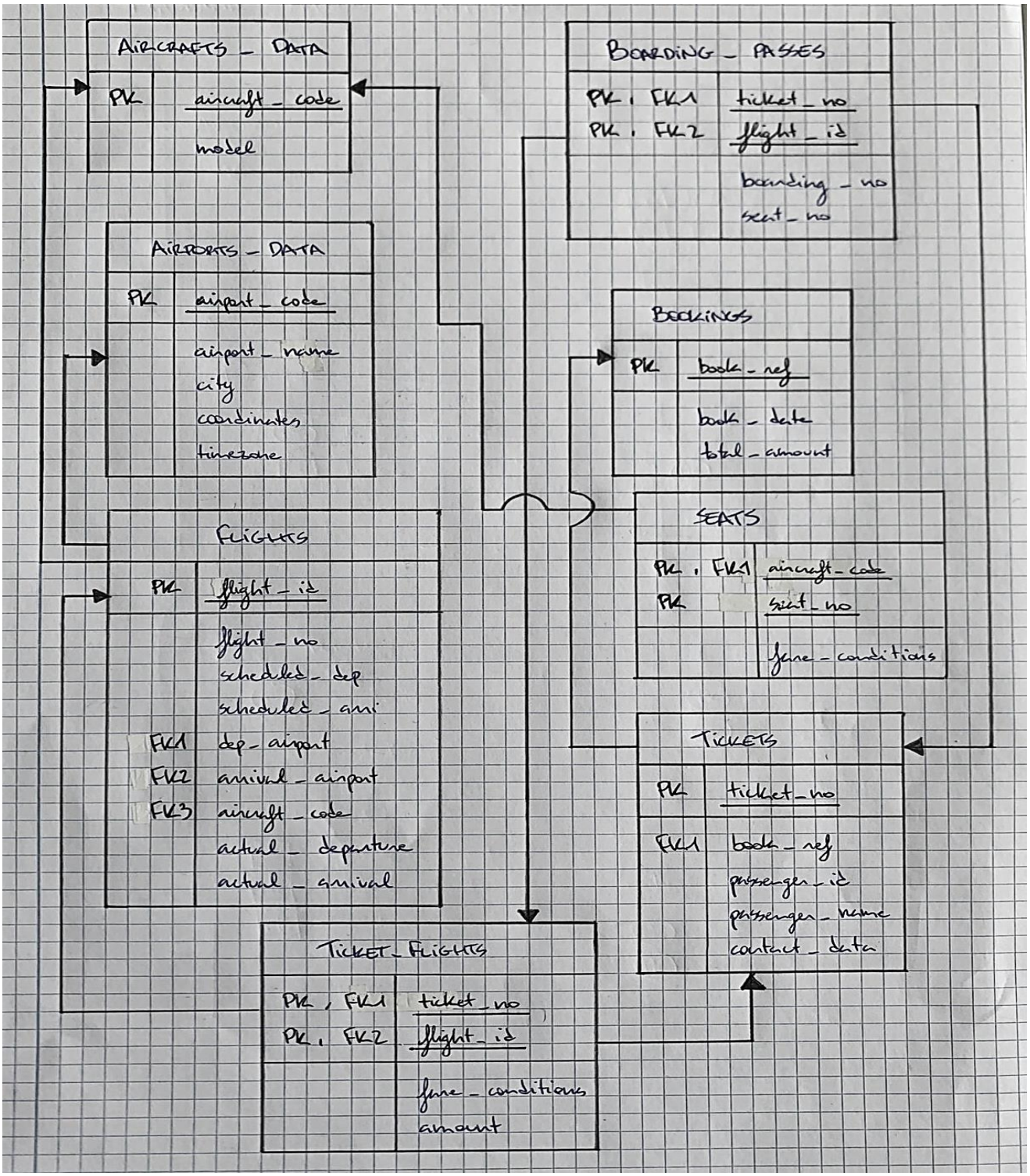
flights (**flight_id**, flight_no, scheduled_departure, scheduled_arrival, departure_airport → airports_data.airport_code, arrival_airport → airports_data.airport_code, aircraft_code → aircrafts_data.aircraft_code, actual_departure, actual_arrival)

seats (**seat_no**, **aircraft_code** →aircrafts_data.aircraft_code, fare_conditions)

ticket_flights (**ticket_no** →tickets.ticket_no, **flight_id** →flights.flight_id, fare_conditions, amount)

tickets (**ticket_no**, book_ref →bookings.book_ref, passenger_id, passenger_name, contact_data)

Diagrama del modelo relacional



Todas las querys se han obtenido en formato html (y texto) para una mejor visualización de la información usando el comando \H en psql para cambiar la salida a dicho formato.
El makefile realiza la query de manera normal (cargandola en formato texto en el log) y después la vuelve a realizar pero la guarda en formato html y no la muestra en la terminal.

Query 1:

Lista el número de reservas (bookings) que contengan un billete de ida y vuelta agrupadas por aeropuerto. Un billete es de ida y vuelta si el aeropuerto de origen del primer vuelo es idéntico al aeropuerto de destino del último vuelo. El resultado mostrará dos columnas con el código del aeropuerto de salida, así como el número de reservas que contienen billetes de ida y vuelta para ese aeropuerto. Igualmente, el resultado se ordenará de forma ascendente usando el atributo (departure airport).

No se ha hecho.

Query 2:

El precio de una reserva (booking.total amount) puede ser calculado a partir del precio de cada vuelo (ticket flights.amount). Crea una consulta que para cada reserva muestre el precio de la misma calculado a partir del valor almacenado en ticket flights.amount. La salida debe mostrar tres columnas conteniendo los atributos booking.book ref, booking.total amount y el valor calculado por vuestra consulta. Ordena la consulta de forma ascendiente usando el atributo bookigs.book ref.

```
SELECT bookings.book_ref,
       bookings.total_amount,
       Sum(amounts.amount) AS "Coste calculado"
FROM   (SELECT tickets.book_ref,
               ticket_flights.amount
        FROM   tickets
               JOIN ticket_flights
                 ON tickets.ticket_no = ticket_flights.ticket_no) AS amounts
       JOIN bookings
         ON bookings.book_ref = amounts.book_ref
GROUP  BY bookings.book_ref
ORDER  BY bookings.book_ref ASC;
```

Se ha implementado de manera anidada, el select del from permite obtener los atributos book_ref y el coste de cada ticket de la unión de las tablas tickets y ticket_flights. Esta tabla se une con bookings, donde se encuentra el coste total de la reserva. Todo esto se agrupa y ordena de forma ascendente por book_ref. En el select principal se obtiene el book_ref, el coste por reserva y la suma de los costes de cada ticket agrupados por reserva(book_ref).

```
query-2: Muestra el book_ref de la reserva, su precio y el precio calculado en base a la suma los
precios de los tickets en la reserva en orden ascendente.

 book_ref | total_amount | Coste calculado
----------+--------------+-----------------
 00000F   |    265700.00 |       265700.00
 000012   |     37900.00 |        37900.00
 000068   |     18100.00 |        18100.00
 000181   |    131800.00 |       131800.00
 0002D8   |     23600.00 |        23600.00
 0002DB   |    101500.00 |       101500.00
 0002E0   |     89600.00 |        89600.00
 0002F3   |     69600.00 |        69600.00
 00034E   |     73300.00 |        73300.00
 000352   |    109500.00 |       109500.00
 000374   |    136200.00 |       136200.00
 00044D   |      6000.00 |         6000.00
```

Query 3:

Crea una consulta que muestre por pantalla el código de aeropuerto y el número de pasajeros recibidos, y ordena la misma de forma ascendiente por el número de pasajeros recibidos. Se considera que un pasajero ha volado al aeropuerto si se ha emitido una tarjeta de embarque. Una tarjeta de embarque para un vuelo del aeropuerto X al Y incrementa en 1 el número de pasajeros recibidos en el aeropuerto Y .

```sql
SELECT f.arrival_airport AS "Código del Aeropuerto",
       Count(bp.seat_no) AS "Número de Pasajeros Recibidos"
FROM   flights AS f
       JOIN boarding_passes AS bp
         ON f.flight_id = bp.flight_id
GROUP  BY f.arrival_airport
ORDER  BY Count(bp.seat_no) ASC;
```

Se ha implementado haciendo una unión de flights y boarding_passes, agrupándolo por el aeropuerto de llegada, y haciendo un count en cada grupo del número de asientos(seat_no).
Esto se ha ordenado por la cuenta de los asientos de forma ascendente.
Al final muestra el código del aeropuerto de llegada y el número de asientos(llenos) de los aviones que llegan a dicho aeropuerto, lo cual equivale a decir el número de pasajeros recibidos.

```
e459630@6B-11-18-11:~/bases_de_datos-main$ make query3
query-3: Obtiene el número de pasajeros recibidos totales por aeropuerto, lo muestra co
n el código del aeropuerto al que llegan y el número de pasajeros calculado en orden as
cendente

 Código del Aeropuerto | Número de Pasajeros Recibidos
-----------------------+-------------------------------
 SWT                   |                             8
 RGK                   |                            51
 CEE                   |                            80
 NYA                   |                            82
 USK                   |                           131
 KXK                   |                           158
 KLF                   |                           168
 UCT                   |                           169
 GDX                   |                           253
 KGP                   |                           257
 NFG                   |                           279
 DYR                   |                           283
 CNN                   |                           340
 TBW                   |                           341
 PEZ                   |                           356
 GRV                   |                           364
```

Query 4:

Vuelo con más asientos vacíos. Muestra en la salida el atributo flight id y el número de asientos vacíos. En caso de empate muestra todos los vuelos que hayan empatado.

```
SELECT f.flight_id AS "  vuelo  ",
       Count(*)    AS "número de asientos vacíos"
FROM   flights AS f
       JOIN seats AS s
         ON f.aircraft_code = s.aircraft_code
       LEFT JOIN boarding_passes AS bp
              ON f.flight_id = bp.flight_id
                 AND s.seat_no = bp.seat_no
GROUP  BY f.flight_id
HAVING Count(s.seat_no) - Count(bp.seat_no) = (SELECT Max(cs - cbp)
                                               FROM
              (SELECT Count(s.seat_no)  AS cs,
                      Count(bp.seat_no) AS cbp
               FROM   flights AS f
                      JOIN seats AS s
                        ON f.aircraft_code =
                           s.aircraft_code
                      LEFT JOIN boarding_passes
                                AS bp
                        ON f.flight_id =
                           bp.flight_id
                           AND
                      s.seat_no = bp.seat_no
               GROUP  BY f.flight_id) AS sitios_vacios);
```

Se ha implementado haciendo un left join de manera que la tabla boarding_passes no tenga la prioridad y por tanto en los casos que el seat_no de esta tabla no pueda añadirse al join se pondrá null y al contarla contribuirá como 0 asientos. Al unir flights y seats se busca obtener los asientos que se pueden llenar de cada avión, al restarle a esto el número de asientos no nulos de boarding_passes se están contando los asientos que no han sido ocupados en cada avión (ya que se agrupa por flight_id).
En la parte inferior al Having se da la condición de que el número de asientos no ocupados de cada avión(Count(s.seat_no)-Count(bp.seat_no)) sea igual al máximo número de asientos vacíos contado(SELECT Max(cs-cbp) FROM) en el select de abajo(que hace exactamente lo mismo que el FROM y GROUP BY de arriba).
Se muestra con el select principal el flight_id de cada vuelo y la cuenta de asientos vacíos de este. Mostrando solo los que tengan el máximo número de asientos vacíos por la condición del GROUP BY HAVING.

```
query-4: Muestra el número de asientos vacíos para cada vuelo representado por su flight_id
   vuelo   | número de asientos vacíos
-----------+--------------------------
      9924 |                      402
      5291 |                      402
      5257 |                      402
      9822 |                      402
      7722 |                      402
      7743 |                      402
     30587 |                      402
       256 |                      402
```

Query 5:

Muestra reservas para los cuales no se haya emitido alguna de las tarjetas de embarque. La salida debe mostrar los atributos book ref y flight id. Ordenar el resultado de forma ascendente por la pareja (book ref, flight id).

```
SELECT ti.book_ref, tf.flight_id
FROM tickets ti

JOIN ticket_flights tf ON ti.ticket_no = tf.ticket_no
LEFT JOIN boarding_passes bp ON ti.ticket_no = bp.ticket_no
WHERE bp.ticket_no IS NULL

ORDER BY ti.book_ref, tf.flight_id;
```

Se ha implementado haciendo uso de 1 join y 1 left join. El primero join combina los datos de las tablas tickets y ticket_flights, a través de la condición ticket_no, que ha de ser igual en ambas tablas . De esta manera conseguimos establecer una relación entre los tickets y los vuelos asociados a esos tickets. Después, con el left join, incluimos todos los registros de la tabla tickets con los registros que coincidan en la tabla boarding_passes, estableciendo así la relación entre ambas tablas. Otra vez, la condición es que ese ticket_no sea igual en ambas tablas. Ahora viene la condición WHERE bp.ticket_no IS null, la cual es clave para conseguir el resultado deseado por el enunciado. Al coger únicamente los registros donde no hay una coincidencia en la tabla boarding_passes, de esta manera sabemos que NO se han emitido tarjetas de embarque para dichos vuelos. Después, mediante el order by, ordenamos la salida justo como se nos indica.

```
query-5: Muestra reservas para los cuales no se haya emitido alguna de las tarjetas de
embarque y el resultado de forma ascendiente por la pareja (book ref, flight id).

 book_ref | flight_id
----------+-----------
 000068   |      1039
 000068   |     17082
 000181   |      1039
 000181   |      1039
 000181   |     11238
 000181   |     11238
 000181   |     17082
 000181   |     17082
 000181   |     20881
 000181   |     20881
 0002D8   |      5522
 0002D8   |     13462
 0002DB   |      4153
 0002DB   |      4153
```

Query 6:

Por definición dos vuelos (flights) realizan el mismo trayecto si su número de vuelo (flight no) es el mismo. Se solicita el trayecto donde la media de los retrasos acumulados sea mayor. Se define retraso como la diferencia entre los valores actual arrival - scheduled arrival. Mostrar en la línea de salida flight no y el retraso medio. En caso de empate deben aparecer todos los trayectos con mayor retraso.

```
SELECT flight_no,
       Avg(Extract(epoch FROM ( actual_arrival - scheduled_arrival ))) AS
       "retraso medio"
FROM   flights
GROUP  BY flight_no
HAVING Avg(Extract(epoch FROM ( actual_arrival - scheduled_arrival ))) >= ALL (
       SELECT
       Avg(Extract(epoch FROM ( actual_arrival - scheduled_arrival )))
       FROM
              flights
       GROUP
              BY flight_no);
```

Se ha implementado haciendo uso de Extract epoch FROM con lo que se ha obtenido la diferencia actual arrival - scheduled arrival en segundos; a eso se le ha realizado un Avg para obtener la media acumulada.
Se han agrupado los valores de las medias en función de flight_no y se muestran aquellas tuplas, de flight_on y la media de sus retrasos, que tengan una media mayor o igual a los demás trayectos(flight_no).
De esta manera se obtienen las que mayor media de retrasos tienen.

```
e459630@6B-11-18-11:~/bases_de_datos-main$ make query6
query-6: Calcula y muestra el o los trayectos con la mayor media de retrasos acumulados
, el trayecta lo muestra por su fllight_no y muestra a su derecha el valor de la media
acumulada del trayecto

 flight_no |     retraso medio
-----------+-----------------------
 PG0450    | 6435.0000000000000000
(1 row)

query-6: The query-6 result has been successfully saved inside folder solutions in html
 format and in the main folder as log
```