

# Práctica 05. Índices y optimización de las bases de datos

Administración y Diseño de Bases de Datos

Cheuk Kelly Ng Pante (alu0101364544@ull.edu.es)

29 de noviembre de 2023

# Índice general

<b>1. Restauracion de la base de datos <i>postgres-air</i></b>	<b>1</b>
<b>2. Incluir sentencias SQL para la creación de los índices</b>	<b>1</b>
<b>3. Identifique las tablas principales y sus principales elementos</b>	<b>2</b>
<b>4. Diagrama Entidad-Relación</b>	<b>5</b>
<b>5. Realizar la siguiente consulta</b>	<b>6</b>
5.1. Utilizar <i>EXPLAIN</i> para obtener el plan de consulta . . . . .	6
5.2. Obtener información de la consulta <i>EXPLAIN</i> . . . . .	6
5.3. Repetir la consulta con <i>EXPLAIN</i> con un limite de 15 registros . . . . .	7
5.4. Obtener información de la consulta <i>EXPLAIN</i> con un limite de 15 registros . . . . .	7
<b>6. Realizar la siguiente consulta similar a la anterior</b>	<b>8</b>
6.1. Utilizar <i>EXPLAIN</i> para obtener el plan de consulta . . . . .	8
6.2. Comparar resultados con la consulta anterior . . . . .	8
6.3. Diferencia de rendimiento . . . . .	8
<b>7. Comparación de consultas</b>	<b>9</b>
7.1. Construir índices . . . . .	9
7.2. Evaluación del rendimiento . . . . .	9

## 1. Restauracion de la base de datos *postgres\_air*

Para la restauración de la base de datos se ha optado por usar la base de datos *postgres\_air.backup*. Antes de restaurar la base de datos, hay que crear la base de datos *postgres\_air*, primero entramos en la consola de postgres y luego creamos la base de datos con la siguiente sentencia:

```
CREATE DATABASE postgres_air;
```

Una vez creada la base de datos, la restauramos con el siguiente comando:

```
pg_restore -x --no-owner -U postgres -d postgres_air ./postgres_air.backup
```

## 2. Incluir sentencias SQL para la creación de los índices

Tenemos las siguientes sentencias SQL:

```
SET search_path TO postgres_air;  
CREATE INDEX flight_departure_airport ON  
flight(departure_airport);  
CREATE INDEX flight_scheduled_departure ON postgres_air.flight  
(scheduled_departure);  
CREATE INDEX flight_update_ts ON postgres_air.flight (update_ts);  
CREATE INDEX booking_leg_booking_id ON postgres_air.booking_leg  
(booking_id);  
CREATE INDEX booking_leg_update_ts ON postgres_air.booking_leg  
(update_ts);  
CREATE INDEX account_last_name  
ON account (last_name);
```

Figura 2.1: Sentencias SQL

Lo que hacen estas sentencias es crear índices en las tablas y atributos más consultados. De esta manera el rendimiento de la base de datos mejora sustancialmente.

Aquí una captura de pantalla de la ejecución de las sentencias SQL:

```
postgres_air=# \i create_index.sql  
SET  
CREATE INDEX  
CREATE INDEX  
CREATE INDEX  
CREATE INDEX  
CREATE INDEX  
CREATE INDEX
```

Figura 2.2: Ejecución de las sentencias SQL

### 3. Identifique las tablas principales y sus principales elementos

Las tablas principales son las siguientes:

- **accounts:** Contiene información de las cuentas de los usuarios.
  - account\_id:** Identificador de la cuenta, de tipo *INTEGER*.
  - login:** Nombre de usuario, de tipo *TEXT*.
  - first\_name** Nombre del usuario, de tipo *TEXT*.
  - last\_name** Apellido del usuario, de tipo *TEXT*.
  - frequent\_flyer\_id** Identificador del viajero frecuente, de tipo *INTEGER*.
  - update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.
- **aircraft:** Contiene información de los aviones.
  - model** Modelo del avión, de tipo *TEXT*.
  - range** Rango del avión, de tipo *NUMERIC*.
  - class** Clase del avión, de tipo *INTEGER*.
  - velocity** Velocidad del avión, de tipo *NUMERIC*.
  - code** Código del avión, de tipo *TEXT*.
- **airport:** Contiene información de los aeropuertos.
  - airport\_code** Código del aeropuerto, de tipo *CHARACTER(3)*.
  - airport\_name** Nombre del aeropuerto, de tipo *TEXT*.
  - city** Ciudad del aeropuerto, de tipo *TEXT*.
  - airport\_tz** Zona horaria del aeropuerto, de tipo *TEXT*.
  - continent** Continente del aeropuerto, de tipo *TEXT*.
  - iso\_country** Código del país del aeropuerto, de tipo *TEXT*.
  - iso\_region** Código de la región del aeropuerto, de tipo *TEXT*.
  - intl** Si el aeropuerto es internacional o no, de tipo *BOOLEAN*.
  - update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.
- **boarding\_pass:** Contiene información de las tarjetas de embarque.
  - pass\_id** Identificador de la tarjeta de embarque, de tipo *INTEGER*.
  - passenger\_id** Identificador del pasajero, de tipo *BIGINT*.
  - booking\_leg\_id** Identificador de la reserva del vuelo, de tipo *BIGINT*.
  - seat** Asiento del pasajero, de tipo *TEXT*.

- boarding\_time** Fecha de embarque, de tipo *TIMESTAMP WITH TIME ZONE*.
- precheck** Si el pasajero ha hecho el check-in o no, de tipo *BOOLEAN*.
- update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.
- **booking:** Contiene información de las reservas.
 

**booking\_id** Identificador de la reserva, de tipo *BIGINT*.

**booking\_ref** Referencia de la reserva, de tipo *TEXT*.

**booking\_name** Nombre de la reserva, de tipo *TEXT*.

**account\_id** Identificador de la cuenta, de tipo *INTEGER*.

**email** Correo electrónico, de tipo *TEXT*.

**phone** Teléfono, de tipo *TEXT*.

**update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.

**price** Precio de la reserva, de tipo *NUMERIC*.
  - **booking\_leg:** Contiene información de las reservas de los vuelos.
 

**booking\_leg\_id** Identificador de la reserva del vuelo, de tipo *INTEGER*.

**booking\_id** Identificador de la reserva, de tipo *INTEGER*.

**flight\_id** Identificador del vuelo, de tipo *INTEGER*.

**leg\_num** Número de la reserva del vuelo, de tipo *INTEGER*.

**is\_returning** Si el vuelo es de vuelta o no, de tipo *BOOLEAN*.

**update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.
  - **flight:** Contiene información de los vuelos.
 

**flight\_id** Identificador del vuelo, de tipo *INTEGER*.

**flight\_no** Número del vuelo, de tipo *TEXT*.

**scheduled\_departure** Fecha de salida programada, tipo *TIMESTAMP WITH TIME ZONE*.

**scheduled\_arrival** Fecha de llegada programada, de tipo *TIMESTAMP WITH TIME ZONE*.

**departure\_airport** Código del aeropuerto de salida, de tipo *CHARACTER(3)*.

**arrival\_airport** Código del aeropuerto de llegada, de tipo *CHARACTER(3)*.

**status** Estado del vuelo, de tipo *TEXT*.

**aircraft\_code** Código del avión, de tipo *CHARACTER(3)*.

**actual\_departure** Fecha de salida actual, de tipo *TIMESTAMP WITH TIME ZONE*.

**actual\_arrival** Fecha de llegada actual, de tipo *TIMESTAMP WITH TIME ZONE*.

**update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.

- **frequent\_flyer:** Contiene información de los viajeros frecuentes.
  - frequent\_flyer\_id** Identificador del viajero frecuente, de tipo *INTEGER*.
  - first\_name** Nombre del viajero frecuente, de tipo *TEXT*.
  - last\_name** Apellido del viajero frecuente, de tipo *TEXT*.
  - title** Título del viajero frecuente, de tipo *TEXT*.
  - card\_num** Número de la tarjeta del viajero frecuente, de tipo *TEXT*.
  - level** Nivel del viajero frecuente, de tipo *INTEGER*.
  - award\_points** Puntos del viajero frecuente, de tipo *INTEGER*.
  - email** Correo electrónico del viajero frecuente, de tipo *TEXT*.
  - phone** Teléfono del viajero frecuente, de tipo *TEXT*.
  - update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.
- **passenger:** Contiene información de los pasajeros.
  - passenger\_id** Identificador del pasajero, de tipo *INTEGER*.
  - booking\_id** Identificador de la reserva, de tipo *INTEGER*.
  - booking\_ref** Referencia de la reserva, de tipo *TEXT*.
  - passenger\_no** Número del pasajero, de tipo *INTEGER*.
  - first\_name** Nombre del pasajero, de tipo *TEXT*.
  - last\_name** Apellido del pasajero, de tipo *TEXT*.
  - account\_id** Identificador de la cuenta, de tipo *INTEGER*.
  - update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.
  - age** Edad del pasajero, de tipo *INTEGER*.
- **phone:** Contiene información de los teléfonos.
  - phone\_id** Identificador del teléfono, de tipo *INTEGER*.
  - account\_id** Identificador de la cuenta, de tipo *INTEGER*.
  - phone** Teléfono, de tipo *TEXT*.
  - phone\_type** Tipo de teléfono, de tipo *TEXT*.
  - primary\_phone** Si es el teléfono principal o no, de tipo *BOOLEAN*.
  - update\_ts** Fecha de actualización, de tipo *TIMESTAMP WITH TIME ZONE*.

## 4. Diagrama Entidad-Relación

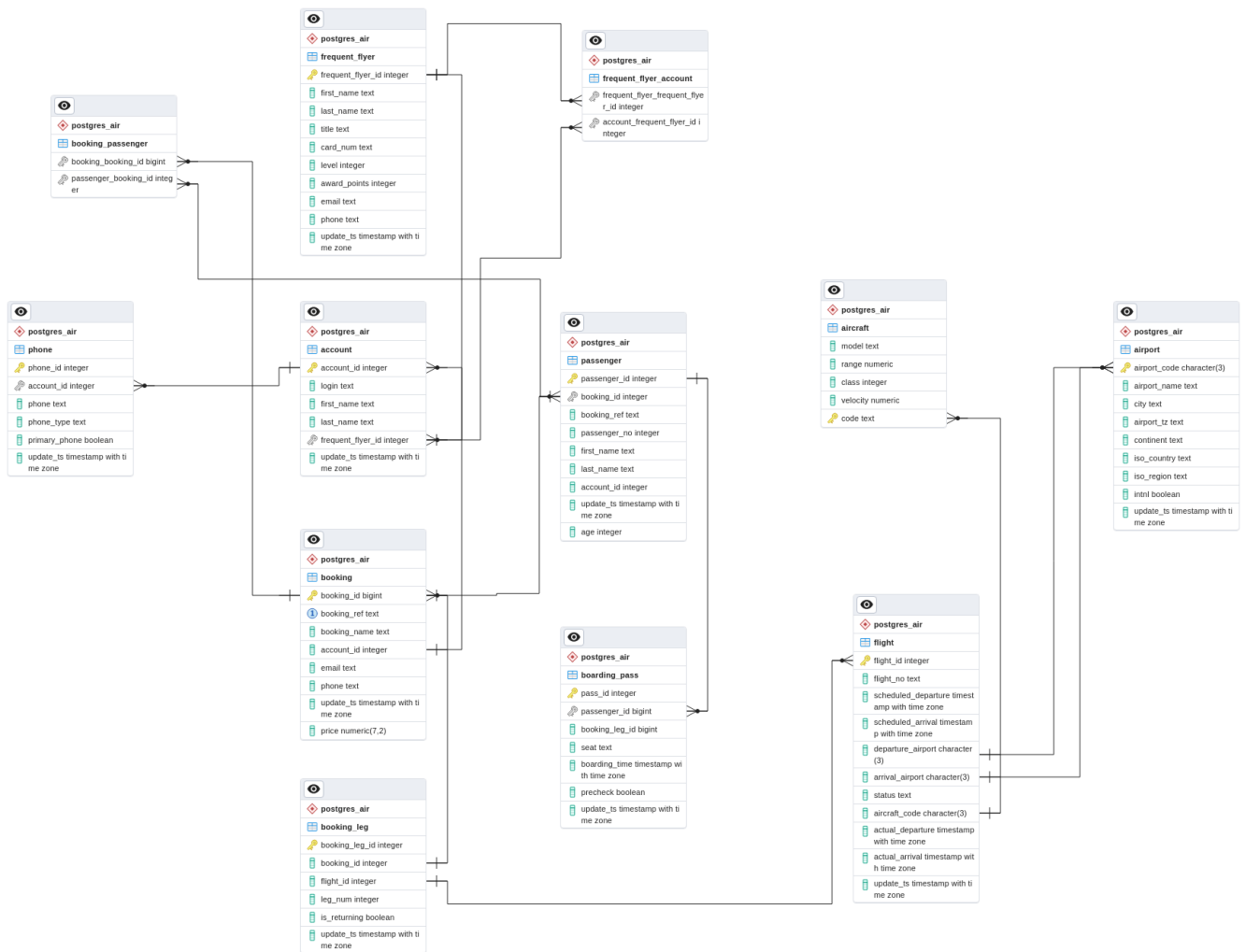


Figura 4.1: Diagrama Entidad-Relación

## 5. Realizar la siguiente consulta

```
SELECT flight_id, scheduled_departure
FROM flight f
JOIN airport a
ON departure_airport=airport_code
AND iso_country='US'
```

### 5.1. Utilizar *EXPLAIN* para obtener el plan de consulta

```
postgres_air=# EXPLAIN SELECT flight_id, scheduled_departure
FROM flight f
JOIN airport a
ON departure_airport=airport_code
AND iso_country='US';
               QUERY PLAN
-----
Hash Join  (cost=20.09..17223.60 rows=144636 width=12)
  Hash Cond: (f.departure_airport = a.airport_code)
    -> Seq Scan on flight f  (cost=0.00..15398.76 rows=683176 width=16)
    -> Hash  (cost=18.33..18.33 rows=141 width=4)
          -> Seq Scan on airport a  (cost=0.00..18.33 rows=141 width=4)
              Filter: (iso_country = 'US'::text)
(6 rows)
```

Figura 5.1: Consulta con *EXPLAIN*

### 5.2. Obtener información de la consulta *EXPLAIN*

- Costo total de la consulta: 17223.60 unidades.
- Costo de configuración: 20.09 unidades.
- Cantidad de filas que se devolverán: 144636 filas.
- Cantidad de filas estimadas: 683176 filas.



### 5.3. Repetir la consulta con *EXPLAIN* con un limite de 15 registros

```
postgres_air=# EXPLAIN SELECT flight_id, scheduled_departure
FROM flight f
JOIN airport a
ON departure_airport=airport_code
AND iso_country='US'
LIMIT 15;

               QUERY PLAN
-----
Limit  (cost=0.29..3.67 rows=15 width=12)
->  Nested Loop  (cost=0.29..32658.60 rows=144636 width=12)
->   Seq Scan on flight f  (cost=0.00..15398.76 rows=683176 width=16)
->   Memoize  (cost=0.29..0.31 rows=1 width=4)
      Cache Key: f.departure_airport
      Cache Mode: logical
->    Index Scan using airport_pkey on airport a  (cost=0.28..0.30 rows=1 width=4)
      Index Cond: (airport_code = f.departure_airport)
      Filter: (iso_country = 'US'::text)

(9 rows)
```

Figura 5.2: Consulta con *EXPLAIN* y límite de 15 registros

### 5.4. Obtener información de la consulta *EXPLAIN* con un limite de 15 registros

Hay una reducción del costo abismal. Hay en total 2 pasos principales, que son dos los dos Merges. El segundo de esto se subdivide en otros dos en donde consulta los índices previamente creados.

- **Costo del paso limitante:** 3.67 unidades.
- **Costo de configuración:** 0.29 unidades.
- **Cantidad de filas que se devolverán:** 15 filas.
- **Cantidad de filas estimadas:** 144636 filas.

## 6. Realizar la siguiente consulta similar a la anterior

```
SELECT flight_id, scheduled_departure
FROM flight f
JOIN airport a
ON departure_airport=airport_code
AND iso_country='CZ'
```

Figura 6.1: Consulta con *EXPLAIN*

### 6.1. Utilizar *EXPLAIN* para obtener el plan de consulta

```
postgres_air=# EXPLAIN SELECT flight_id, scheduled_departure
FROM flight f
JOIN airport a
ON departure_airport=airport_code
AND iso_country='CZ';
                                QUERY PLAN
-----
Nested Loop  (cost=12.42..2964.72 rows=1026 width=12)
-> Seq Scan on airport a  (cost=0.00..18.33 rows=1 width=4)
    Filter: (iso_country = 'CZ'::text)
-> Bitmap Heap Scan on flight f  (cost=12.42..2936.08 rows=1032 width=16)
    Recheck Cond: (departure_airport = a.airport_code)
-> Bitmap Index Scan on flight_departure_airport  (cost=0.00..12.16 rows=1032 width=0)
    Index Cond: (departure_airport = a.airport_code)
(7 rows)
```

Figura 6.2: Consulta con *EXPLAIN*

### 6.2. Comparar resultados con la consulta anterior

Esta consulta tiene un bajo costo, para no haber limitado la consulta de ninguna manera.

### 6.3. Diferencia de rendimiento

El principal motivo de que esta consulta sea más ligera se debe precisamente a que sus operaciones son más sencillas de usar, pues no realiza grandes joins como antes, y que además devuelve muchas menos filas.

## 7. Comparación de consultas

<pre>SELECT flight_id       ,departure_airport       ,arrival_airport FROM flight WHERE scheduled_arrival BETWEEN '2020-10-14' AND '2020-10-15';</pre>	<pre>SELECT flight_id       ,departure_airport       ,arrival_airport FROM flight WHERE scheduled_arrival:: date='2020- 10-14';</pre>
A	B

### 7.1. Construir índices

Para construir los índices, se han usado las siguientes sentencias SQL:

- Consulta A:

```
CREATE INDEX flight_idx_brin ON flight USING BRIN(flight_id);
```

- Consulta B:

```
CREATE INDEX flight_idx_hash ON flight USING hash(flight_id);
```

### 7.2. Evaluación del rendimiento

Al realizar la consulta sin índices de la consulta *A* y *B* obtenemos lo siguiente:

```
postgres_air=# SELECT flight_id
,departure_airport
,arrival_airport
FROM flight
WHERE scheduled_arrival BETWEEN
'2020-10-14' AND '2020-10-15';
Time: 358,191 ms
```

(a) Consulta A sin índice.

```
postgres_air=# SELECT flight_id
,departure_airport
,arrival_airport
FROM flight
WHERE scheduled_arrival:: date='2020-10-14';
Time: 68,007 ms
postgres_air=#
```

(b) Consulta B sin índice.

Figura 7.1: Consultas de ambas consultas sin índices

Ahora, al realizar la consulta con índices de la consulta *A* y *B* obtenemos lo siguiente:

```
postgres_air=# CREATE INDEX flight_idx_brin ON flight USING BRIN(flight_id);
CREATE INDEX
Time: 132,746 ms
postgres_air=# SELECT flight_id
,departure_airport
,arrival_airport
FROM flight
WHERE scheduled_arrival BETWEEN
'2020-10-14' AND '2020-10-15';
Time: 49,429 ms
```

(a) Consulta A con índice.

```
postgres_air=# CREATE INDEX flight_idx_hash ON flight USING hash(flight_id);
CREATE INDEX
postgres_air=# \timing on
Timing is on.
postgres_air=# SELECT flight_id
,departure_airport
,arrival_airport
FROM flight
WHERE scheduled_arrival:: date='2020-10-14';
Time: 65,865 ms
```

(b) Consulta B con índice.

Figura 7.2: Consultas de ambas consultas con índices

	<b>Consulta A</b>	<b>Consulta B</b>
Sin índice	358,191ms	68,007
Con índice	49,429ms	65,865

Cuadro 7.1: Tabla de resultados

Con los resultados obtenidos, con se observa en la tabla 7.1, podemos ver que al realizar un índice en la consulta *A* el rendimiento mejora considerablemente, mientras que en la consulta *B* no mejora tanto ya que la consulta de por sí era bastante rápida.

Para el caso de la consulta *A*, se usa un índice del tipo *BRIN* ya que es un índice que se usa para tablas muy grandes, y en este caso la tabla *flight* tiene 144636 filas. Para el caso de la consulta *B*, se usa un índice del tipo *hash* por tratarse de una comparacion de igualdad simple.