Trabajo Práctico Final: TP Final

Felipe, Carrozzo[1] - Micaela Narváez[2]

[1] *felipe.carrozzo@ingenieria.uner.edu.ar* [2] *micaela.narvaez@ingenieria.uner.edu.ar*

***Resumen*—Se realiza la recuperación de una base de datos llamada "flights" en el contexto de la materia Bases de Datos de la Tecnicatura Universitaria en Procesamiento y Explotación de Datos; esta base de datos contiene información sobre vuelos internacionales durante un mes. Nos planteamos como objetivos realizar consultas complejas que involucren tres tablas y distintas operaciones como JOIN, EXIST, etc. Utilizamos "PgAdmin" para la interfaz de usuario y "PostGre" como Sistema de Gestión de Bases de Datos. En este trabajo se plasman los resultados de haber programado en sql y python, cumpliendo con los objetivos propuestos. La importancia de este tipo de trabajo radica en el aprendizaje para la realización de futuras consultas y visualizaciones de información, también es muy importante ver cómo podemos integrar conocimientos adquiridos en otras asignaturas.**

***Palabras clave*—sql, datos, vuelos, vista, gestión, python**

I. Introducción[1]

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada "base de datos", contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos relacional, de manera que sea tanto práctica como eficiente.
Este trabajo está enmarcado en la cátedra de Bases de datos, perteneciente a la Tecnicatura Universitaria en Procesamiento y Explotación de Datos. El objetivo general es demostrar los conocimientos adquiridos en el cursado de la materia mediante la restauración de una base de datos relacional elegida, realización de consultas e interfaz de acceso. En este trabajo, se utilizó "pgAdmin" como interfaz de acceso y "PostGreSQL" como SGBD. Además la interfaz fue realizada en el lenguaje Python.
La base de datos utilizada se denomina "flights" y contiene información sobre vuelos internacionales durante un mes. El archivo restaurado tiene extensión .tar, y la base de datos tiene 8 tablas: aircrafts_data, airports_data, boarding_passes, bookings, flights, seats, tickets_flights, tickets.

II. desarrollo

A. *Diagrama entidad-relación*

Un diagrama entidad-relación, también conocido como modelo entidad relación o ERD, es un tipo de diagrama de flujo que ilustra cómo las "entidades", como personas, objetos o conceptos, se relacionan entre sí dentro de un sistema. Fue introducido por Peter Chan en 1976. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Una entidad es algo que puede identificarse y que es importante para el sistema que se va a desarrollar. Son los objetos de datos más importantes de los que se va a obtener información. Por lo general representan una persona, lugar, cosa, evento o información de interés.

Las relaciones representan asociaciones del mundo real entre una o más entidades, no tiene otra existencia física o conceptual que no sea la de relacionar (asociar) dos entidades.

Los atributos son características de las entidades y de las relaciones que le agregan detalles descriptivos. [1]

En el apéndice se encuentra el Diagrama Entidad Relación para este caso en particular.

B. *Diagrama de tablas*

Las tablas son la estructura lógica de una BD Relacional. Ninguna TABLA de una BD relacional puede prescindir de

[1]felipe.carrozzo@ingenieria.uner.edu.ar Felipe,Carrozzo
micaela.narvaez@ingenieria.uner.edu.ar Micaela, Narváez

contar con una clave. En este caso el diagrama de tablas fue obtenido del esquema de la base de datos. En el apéndice se encuentra un enlace al esquema de éstas.

C. Consultas

Una consulta es una instrucción que solicita que se recupere información.

Para cumplir con los objetivos propuestos para este trabajo hemos planteado tres consultas:

Comenzamos recuperando información sobre la cantidad de tickets vendidos y la cantidad de tickets abordados (el código de esta consulta está en la Tabla I), ya que una persona que compra un ticket puede no abordar el avión, y ésta información podría ser de utilidad para introducir mejoras en este sentido.

TABLA I

CONSULTA EN SQL SOBRE LA BASE DE DATOS FLIGHTS

Mostrar la cantidad de tickets vendidos y la cantidad de tickets que se abordaron
SELECT (SELECT COUNT(*) FROM bookings.tickets) AS num_tickets_vendidos, COUNT(DISTINCT t.ticket_no) AS num_tickets_abordados FROM bookings.tickets t JOIN bookings.boarding_passes bp ON t.ticket_no = bp.ticket_no;

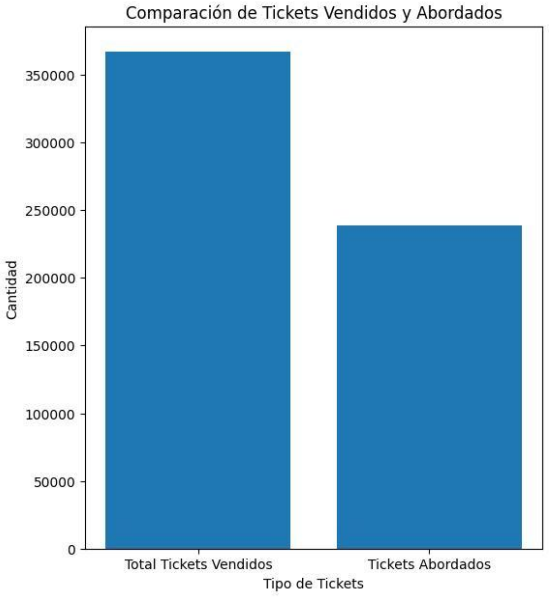

Fig. 1: Gráfica de proporción de la distribución de la cantidad de tickets comprados y tickets abordados. Fuente: Elaboración propia.

Si bien la consulta que realizamos anteriormente tiene una buena sintaxis y nos da el resultado que deseábamos, hay una mejor manera de expresar esto, como podemos ver en la tabla II:

TABLA II

CONSULTA EN SQL SOBRE LA BASE DE DATOS FLIGHTS

Mostrar la cantidad de tickets que fueron comprados pero no abordados
SELECT count(DISTINCT t.ticket_no) AS no_abordaron FROM bookings.tickets t LEFT JOIN bookings.boarding_passes bp ON t.ticket_no = bp.ticket_no WHERE bp.ticket_no IS NULL;

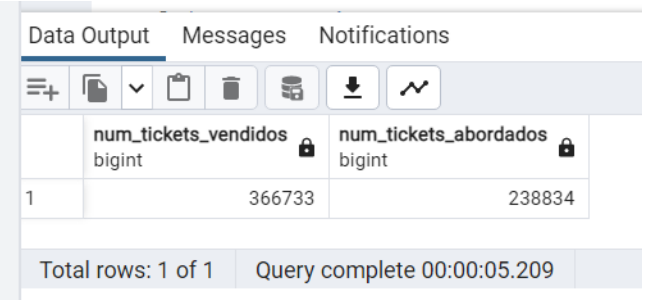

Fig. 2: Resultado y performance para la consulta de la Tabla I. Fuente: elaboración propia

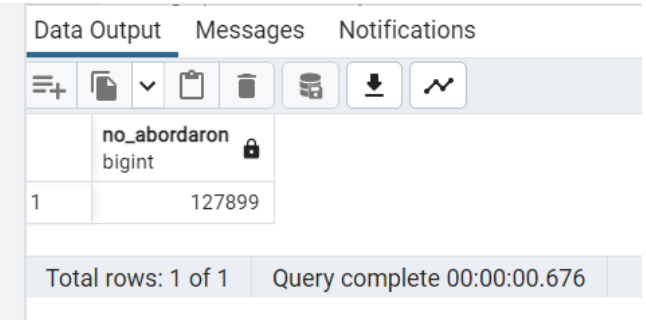

Fig. 3: Resultado y performance para la consulta de la Tabla II. Elaboración propia.

En las Figuras 2 y 3 podemos observar la diferencia de tiempos de ejecución entre una consulta y otra, esto se debe a que la primera utiliza funciones de agregación y la segunda operaciones de conjuntos.

En la Tabla III podemos ver el código de otra consulta que recupera los nombres de aquellos pasajeros que hayan tomado más de 10 vuelos, esta consulta sería pertinente ya que son vuelos internacionales para ver qué personas salen de su país y cuántas veces en un mes.

TABLA III

CONSULTA EN SQL SOBRE LA BASE DE DATOS FLIGHTS

Mostrar el nombre de los pasajeros que tomaron 10 vuelos o más
SELECT passenger_name FROM bookings.flights f JOIN bookings.boarding_passes bp ON f.flight_id = bp.flight_id JOIN bookings.tickets t ON bp.ticket_no = t.ticket_no GROUP BY passenger_name HAVING COUNT(*) > 10;

TABLA IV

CONSULTA EN SQL SOBRE LA BASE DE DATOS FLIGHTS

Mostrar el identificador de vuelo en el que hayan abordado entre 100 y 120 personas
SELECT f.flight_id, COUNT (DISTINCT bp.boarding_no) as cantidad_pasajeros FROM bookings.flights f JOIN bookings.boarding_passes bp ON f.flight_id = bp.flight_id GROUP BY f.flight_id HAVING COUNT (DISTINCT bp.boarding_no) BETWEEN 100 AND 120 ;

En la tabla IV se presenta una consulta para saber qué vuelos llevaron entre 100 y 120 pasajeros, esto podría permitir saber qué vuelos son más rentables que otros en función de la ocupación del avión.

TABLA V

CONSULTA EN SQL SOBRE LA BASE DE DATOS FLIGHTS

Mostrar el nombre de los pasajeros que abordaron algún avión entre dos fechas específicas desde un aeropuerto específico
SELECT t.passenger_name FROM bookings.flights f JOIN bookings.boarding_passes bp ON f.flight_id = bp.flight_id JOIN bookings.tickets t ON bp.ticket_no = t.ticket_no WHERE f.actual_departure BETWEEN '2017-07-16 03:44:00-03' AND '2017-07-17 09:40:00-03' AND f.departure_airport = 'DME'

Sería útil por motivos varios (pérdida de equipaje por ejemplo) saber qué pasajeros abordaron un vuelo en específico para recuperar esta información necesitamos saber la fecha, aproximada y el aeropuerto del que salió el avión, el código de esta consulta se puede ver en la Tabla V.

III. OPTIMIZACIÓN

La optimización de consultas es el proceso de selección del plan de evaluación de las consultas más eficiente de entre las muchas estrategias generalmente disponibles para el procesamiento de una consulta dada, especialmente si la consulta es compleja. No se espera que los usuarios escriban las consultas de modo que puedan procesarse de manera eficiente. Por el contrario, se espera que el sistema cree un plan de evaluación que minimice el coste de la evaluación de las consultas. Ahí es donde entra en acción la optimización de consultas.

Un aspecto de la optimización de las consultas tiene lugar en el nivel del álgebra relacional, donde el sistema intenta hallar una expresión que sea equivalente a la expresión dada, pero de ejecución más eficiente. Otro aspecto es la elección de una estrategia detallada para el procesamiento de la consulta, como puede ser la selección del algoritmo que se usará para ejecutar una operación, la selección de los índices concretos que se van a emplear, etc.

La diferencia en coste (en términos de tiempo de evaluación) entre una estrategia buena y una mala suele ser sustancial, y puede resultar de varios órdenes de magnitud [2], como pudimos ver en las Figuras 2 y 3.

Para hallar el plan de evaluación de consultas menos costoso el optimizador necesita generar planes alternativos que produzcan el mismo resultado que la expresión dada y escoger el de menor coste.

En pos de poder ver esto con mayor claridad, en este trabajo decidimos realizar la representación 3 de las etapas del optimizador.

La primera representación hace referencia a la consulta tal y como está escrita.

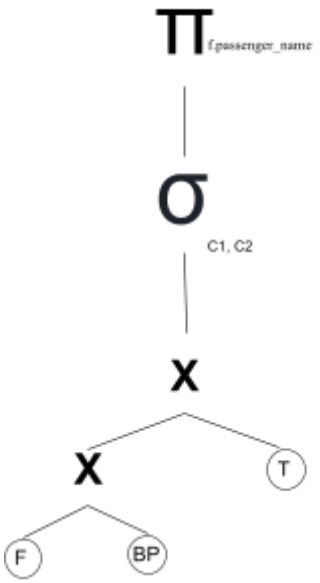

Fig. 4: Árbol canónico.

En la segunda representación podemos apreciar la optimización en cuanto al número de filas.

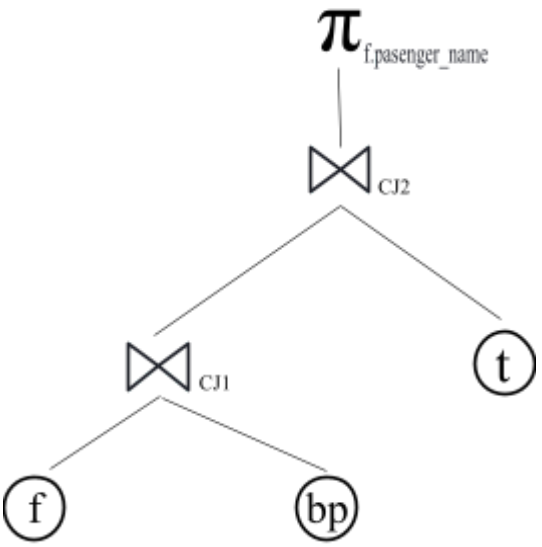

Fig. 5: Representación de la optimización de filas. CJ1 y CJ2 son las condiciones utilizadas en la consulta

Y por último la representación de la ejecución del optimizador la vemos en la Figura 6.

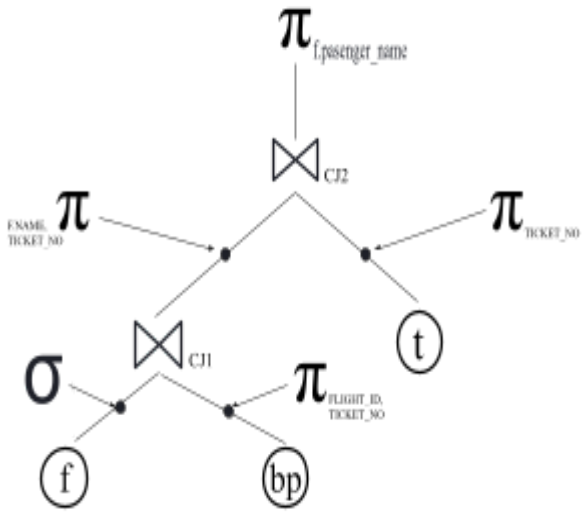

Fig. 6: Representación de la optimización final. CJ1 y CJ2 son las condiciones utilizadas en la consulta

IV. CONCLUSIONES

Durante el desarrollo de este trabajo pudimos cumplir con los objetivos planteados: pensar y escribir consultas eficientes, de las cuales se puedan obtener resultados y entender de qué forma trabaja el optimizador de una consulta. También logramos trabajar con la base de datos y con Python, con el cual generamos una gráfica para visualizar mejor los datos, integrando así conocimientos adquiridos en otras materias de la carrera.

Este trabajo nos deja como aprendizaje que, si bien la optimización de las consultas la realiza el sistema, la forma de expresar la consulta puede influir también en el tiempo de ejecución de la misma por lo que nos pone a pensar en nuestro rol.

En futuras investigaciones se podría tener una entrevista con personas que trabajen con vuelos diariamente, y necesiten recuperar información, para conocer exactamente las necesidades y evaluar si las consultas que hicimos tienen valor real.

APÉNDICES

A. Diagrama entidad relación

Adjuntamos un link en el que se puede apreciar con mejor calidad y tamaño el Diagrama Entidad Relación para el caso de este trabajo.

DIAGRAMA ENTIDAD RELACIÓN

B. Tablas

Dejamos en este apartado una imagen con las tablas y sus relaciones.

DIAGRAMA DE TABLAS

C. Algoritmos

Dejamos aquí el código de creación de las tablas y consultas

SCRIPT POSTGRE

En este enlace pueden encontrar el código de la Figura 1 en “Python”

INTERFAZ EN PYTHON

REFERENCIAS

[1] Recursos en el campus virtual cátedra “Bases de datos”, FIUNER.
[2] Elmasri, Ramez, Shamkant B. Navathe, y José Manuel Díaz. Fundamentos de Sistemas de Bases de Datos, Quinta Edición. Madrid: Addison-Wesley, 2007.
[3] Abraham Silberschatz, Henry F. Korth, S. Sudarshan. Fundamentos de bases de datos