UNIVERSIDAD NACIONAL DE ENTRE RÍOS
FACULTAD DE INGENIERÍA

CARRERA: Tecnicatura Universitaria en Procesamiento y Explotación de Datos

MATERIA: Base de Datos

NOMBRE DE ACTIVIDAD: Trabajo Integrador Final

FECHA DE ENTREGA: 8/06/2023

PROFESORES:

Hadad Alejandro

Elías Walter

Fernandez Maximiliano

Cettour Hermes

ALUMNOS:

Ruiz Diaz, Enzo

Venturini, Angelo

Introducción

Para el desarrollo del trabajo, el grupo decidió utilizar la base de datos chinook proporcionada por la cátedra. Esta es una base de datos de una tienda de medios digitales que incluye tablas para artistas, álbumes, pistas de medios, facturas y clientes.
Esta base de datos posee 11 tablas y 15607 filas totales.
A la base de datos dada por la cátedra tuvimos que realizar ciertas modificaciones, ya que encontramos que Postgres nos obligaba a poner los nombres de las columnas y tablas entre comillas dobles, y esto fue debido a cómo Postgres interpreta las columnas y/o tablas con un nombre que es case sensitive. Por lo tanto, para simplificar la escritura de las queries, modificamos todos los nombres de las tablas y de las columnas a nombres que no contengan mayúsculas, y al tipo de escritura de camelcase lo modificamos a snake case.

Tabla 1

tabla exploratoria de la base de datos Chinook

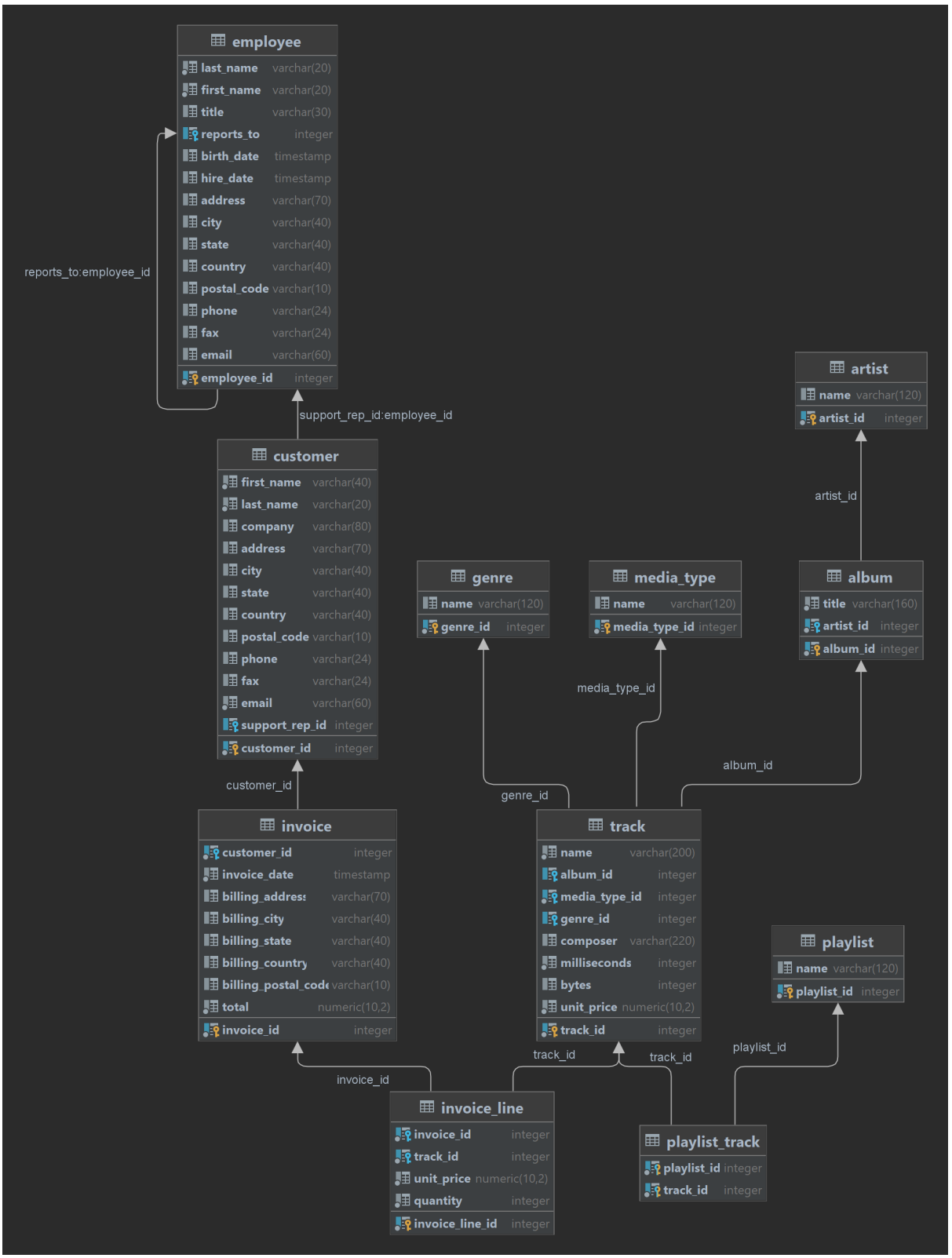
Nombre de la columna	Cantidad de filas	Cantidad de columnas
album	347	3
artist	275	2
customer	59	13
employee	8	15
genre	25	2
invoice	412	9
invoice_line	2240	5
media_type	5	2
playlist	18	2
playlist_track	8715	2
track	3503	9

Diagrama de tablas y DER

Para realizar el diagrama de tablas de la base de datos decidimos utilizar el software DataGrip de la empresa Jetbrains ya que este programa nos permite conectar directamente la base de datos en el explorador del mismo, y generarlas automáticamente, pero con la particularidad de que obtenemos un panorama detallado de cada tabla de forma sencilla simplemente posando el mouse sobre las mismas. Además de que el programa permite una gran personalización del diagrama en sí, moviendo las flechas donde creamos pertinente, así como también te explicita el nombre de las columnas de las claves foráneas y a qué columna apunta a un lado de la flecha.

Gráfico 1

Diagrama de tablas de la base de datos chinook



Nota: La imagen se encuentra adjunta en los archivos del trabajo práctico como diagrama_de_tablas_chinook.png

Diagrama de Entidad-Relación

Para realizar el diagrama de entidad-relación se utilizó la página Draw.io y se realizó en base al diagrama de tablas obtenido anteriormente. Notamos que tablas invoice_line y playlist_track tienen como clave primaria dos claves foráneas pertenecientes a dos entidades y esta tiene cardinalidad de muchos a muchos con esas tablas por lo que para el diagrama de entidad-relación decidimos que las tablas (invoice_line y playlist_track) esten representadas como una relación.

El diagrama quedó conformado por las siguientes entidades:

- Customer (Cliente): Almacena información sobre los clientes de la tienda de música, como su nombre, dirección, ciudad, país, etc.
- Employee (Empleado): Representa a los empleados de la tienda de música y almacena detalles como su nombre, título, fecha de contratación, etc.
- Invoice: Contiene información sobre las facturas generadas por las ventas de la tienda, incluyendo detalles como la fecha, el cliente asociado y el total.
- Track (Canción): Almacena información sobre las canciones disponibles en la tienda, como el nombre, el álbum al que pertenece, el género, la duración, etc.
- Album (Álbum): Contiene detalles sobre los álbumes de música, como su título, el artista asociado y el género.
- Artist (Artista): Representa a los artistas o bandas musicales que crearon los álbumes.
- Genre (Género): Almacena información sobre los diferentes géneros musicales.
- MediaType (Tipo de medio): Representa los diferentes formatos de medios disponibles, como MP3, AAC, etc

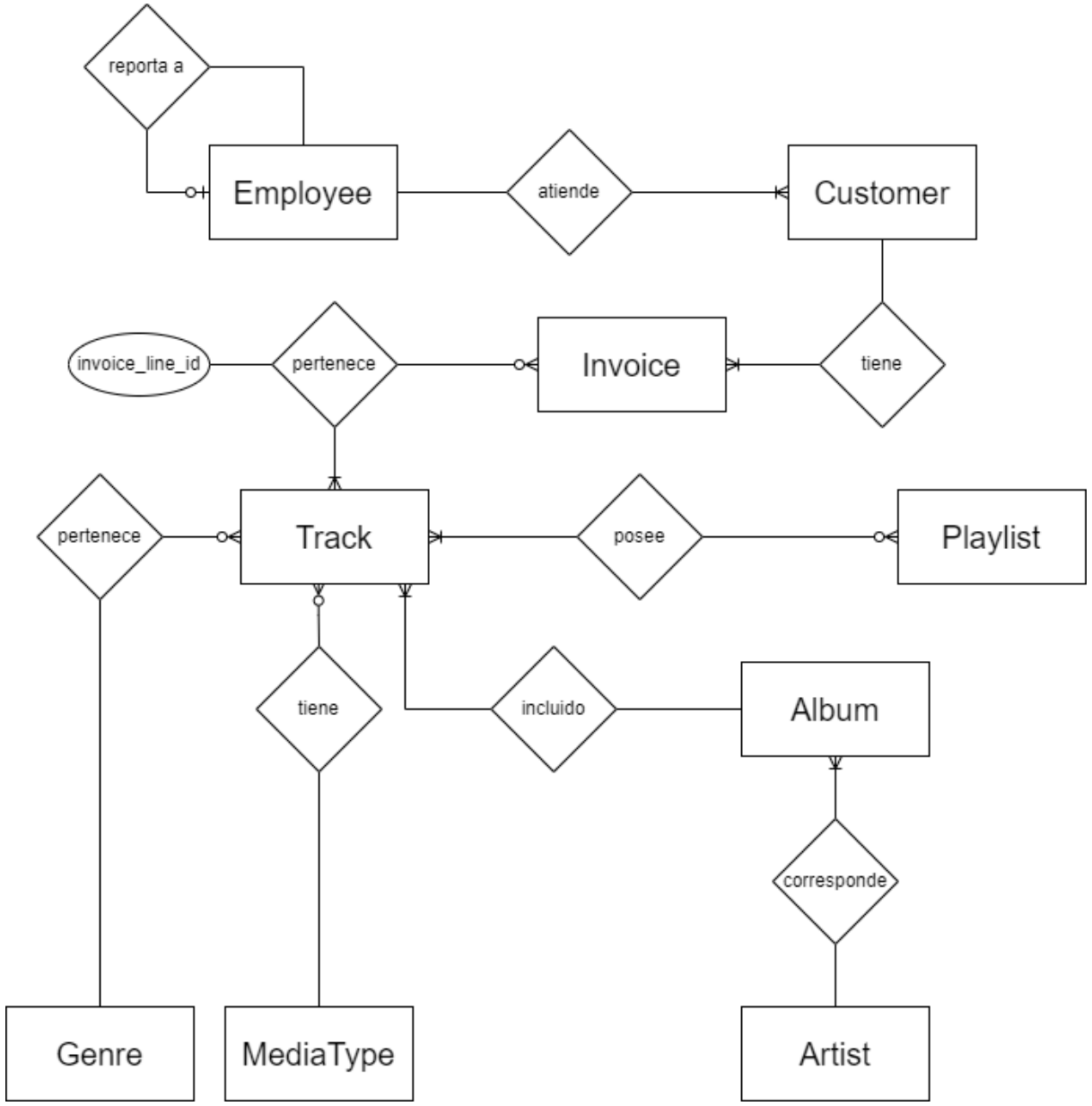
<u>Relaciones principales en el diagrama de entidad-relación de Chinook:</u>

- Customer y Invoice: Un cliente puede tener múltiples facturas, pero una factura pertenece a un solo cliente (relación uno a muchos).
- Employee y Customer: Un empleado puede atender a varios clientes, pero un cliente solo tiene un empleado asignado (relación uno a muchos).
- La entidad Invoice tiene una relación muchos a muchos con la entidad Track ya que una factura puede tener múltiples pistas y una pista puede estar asociada con varias facturas.
- La entidad Track tiene una relación muchos a uno con las entidades Album, Artist y Genre, ya que una canción pertenece a un álbum, es creada por un artista y se clasifica en un género específico.

- La entidad Track tiene una relación muchos a muchos con la entidad Playlist ya que una lista de reproducción puede tener múltiples pistas, y una pista puede estar en múltiples listas de reproducción .

Gráfico 2

Diagrama de entidad-relación chinook



Nota: La imagen se encuentra adjunta en los archivos del trabajo práctico con el nombre de DER_chinook.png

Por último, el DDL resultante de estos diagramas se adjunta como archivo .sql con el nombre “DDL.sql”.

Consultas realizadas

En esta sección se presentan las consultas realizadas en la base de datos Chinook, con el objetivo de obtener información relevante sobre las ventas, los clientes y las canciones más populares, entre otros aspectos.

Gráfico 3

Consulta 1

```sql
-- El promedio de ventas de todos los empleados que hayan vendido productos.
SELECT em.first_name, em.last_name, AVG(iv.total) avg_ventas FROM employee AS em
    JOIN customer AS cust ON em.employee_id = cust.support_rep_id
    JOIN invoice AS iv ON iv.customer_id = cust.customer_id
GROUP BY em.first_name, em.last_name
HAVING AVG(iv.total) IS NOT NULL
ORDER BY avg_ventas DESC;
```

Como observamos en el gráfico 3, la consulta 1 nos permite conocer el promedio de ventas de todos los empleados. En primer lugar la instrucción SELECT selecciona las columnas first_name y last_name de la tabla employee, y con la función de agregación AVG calculamos los promedios de la columna total de la tabla invoice, que fue asignado al alias de avg_ventas. Con la cláusula FROM especificamos la tabla employee y se le asigna un alias, mediante la cláusula JOIN se une la tabla customer con employee utilizando la columna support_rep_id, y de la misma se une customer con la tabla invoice mediante la columna customer_id. La instrucción GROUP BY agrupa el conjunto de resultados por las columnas first_name y last_name. Seguidamente, la instrucción HAVING filtra cualquier grupo donde el promedio de la columna total sea nulo. Por último, la instrucción ORDER BY ordena al conjunto de resultados en orden descendente según la columna avg_ventas.

Gráfico 4

Consulta 2

```sql
-- Géneros que no hayan sido comprados por ningún cliente.
SELECT gn.name
FROM genre AS gn
WHERE NOT EXISTS (
  SELECT cust.customer_id
  FROM customer AS cust
  WHERE EXISTS (
    SELECT tr.genre_id
    FROM track AS tr
        JOIN invoice_line AS ivl ON tr.track_id = ivl.track_id
        JOIN invoice AS iv ON ivl.invoice_id = iv.invoice_id
    WHERE iv.customer_id = cust.customer_id
    AND tr.genre_id = gn.genre_id
  )
);
```

En la consulta 2 que vemos en el gráfico 4, obtenemos como resultado aquellos géneros que no fueron comprados por ningún cliente.

De la tabla genre, con el alias gn, seleccionamos la columna name, donde luego se realiza la instrucción WHERE NOT EXISTS para filtrar los géneros que no cumplen con la condición especificada en la subconsulta. Dentro de esta, se verifica si existe algún cliente que haya comprado un género específico, que utiliza una cláusula WHERE EXISTS donde verifica si existe alguna coincidencia entre el género, el cliente y las compras.

Gráfica 5

Consulta 3

```sql
-- Las canciones agrupadas por géneros que hayan sido vendidos más de 40
-- veces pero menos de 1000 veces.
SELECT gn.name, SUM(ivl.quantity) AS veces_comprado FROM genre AS gn
    JOIN track AS tr ON gn.genre_id = tr.genre_id
    JOIN invoice_line AS ivl ON tr.track_id = ivl.track_id
GROUP BY gn.name
HAVING SUM(ivl.quantity) BETWEEN 40 AND 1000
ORDER BY veces_comprado DESC;
```

En la consulta 3 que vemos en el gráfico 5, obtenemos como resultado las canciones agrupadas por géneros que hayan sido vendidos más de 40 veces y menos de 1000 veces. Especificando la tabla genre con el alias gn, seleccionamos la columna name y utilizando la función de agregación SUM, obtenemos la suma total de la columna quantity dándole el alias veces_comprado. Uniendo las tablas utilizando la instrucción JOIN con las columnas pertinentes, agrupando los resultados por la columna name con la instrucción GROUP BY, y luego se filtran los grupos cuya columna veces_comprado se encuentre entre 40 y 1000 (estos números fueron elegidos arbitrariamente). Por último, con la cláusula ORDER BY…DESC se ordena el conjunto de forma descendente según la columna veces_comprado.

Consulta, árbol de ejecución y árbol optimizado

En esta sección presentamos una consulta realizada en la base de datos Chinook que utiliza cuatro tablas, contiene una condiciones de igual y una de rango junto con sus respectivos árbol de ejecución y árboles optimizados. El árbol de ejecución muestra cómo se ejecuta la consulta en la base de datos, mientras que los árboles optimizados muestran cómo se puede mejorar la consulta para que sea más eficiente. El objetivo de esta sección es mostrar cómo se puede mejorar el rendimiento de las consultas en la base de datos mediante la optimización del plan de ejecución.

Gráfico 6

Consulta utilizando cuatro tablas, una condición de igualdad y una de rango

```sql
-- Ejercicio 7
-- Clientes que han comprado una canción de más de 3 minutos y que son de Estados Unidos (USA)
--          A1        ,       A2      ,    A3  ,                 A4                    ,   A5
SELECT cust.first_name, cust.last_name, tr.name, (tr.milliseconds / 1000) duracion_en_s, gn.name
FROM customer AS cust
    JOIN invoice AS iv ON cust.customer_id = iv.customer_id -- C6
    JOIN invoice_line AS ivl ON iv.invoice_id = ivl.invoice_id -- C5
    JOIN track AS tr ON ivl.track_id = tr.track_id -- C4
    JOIN genre AS gn ON tr.genre_id = gn.genre_id -- C3
WHERE cust.country = 'USA' AND tr.milliseconds >= 180000; -- C1, C2
```

Gráfico 7
Árbol canónico

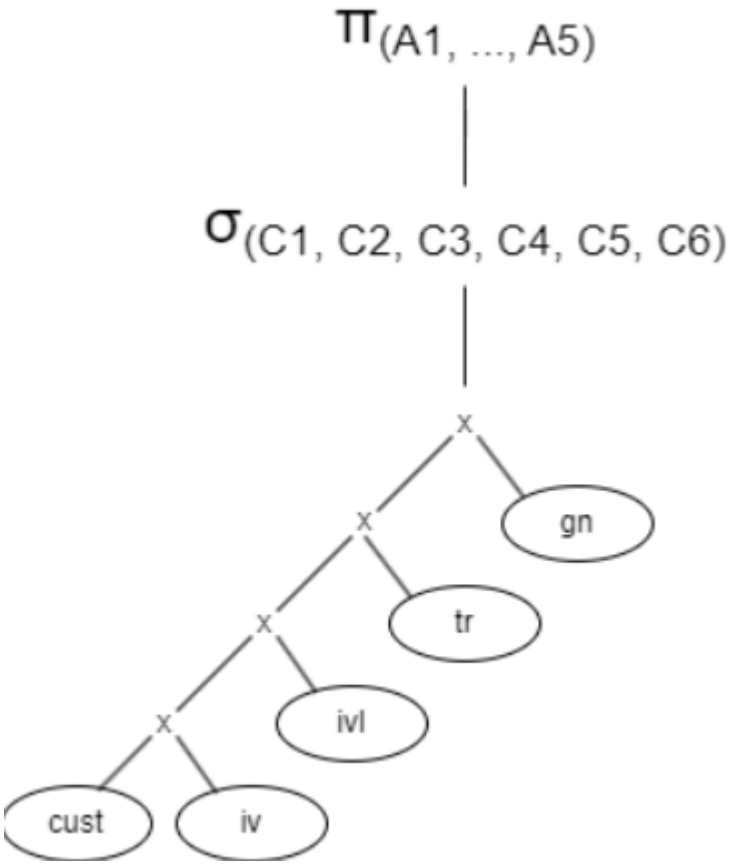
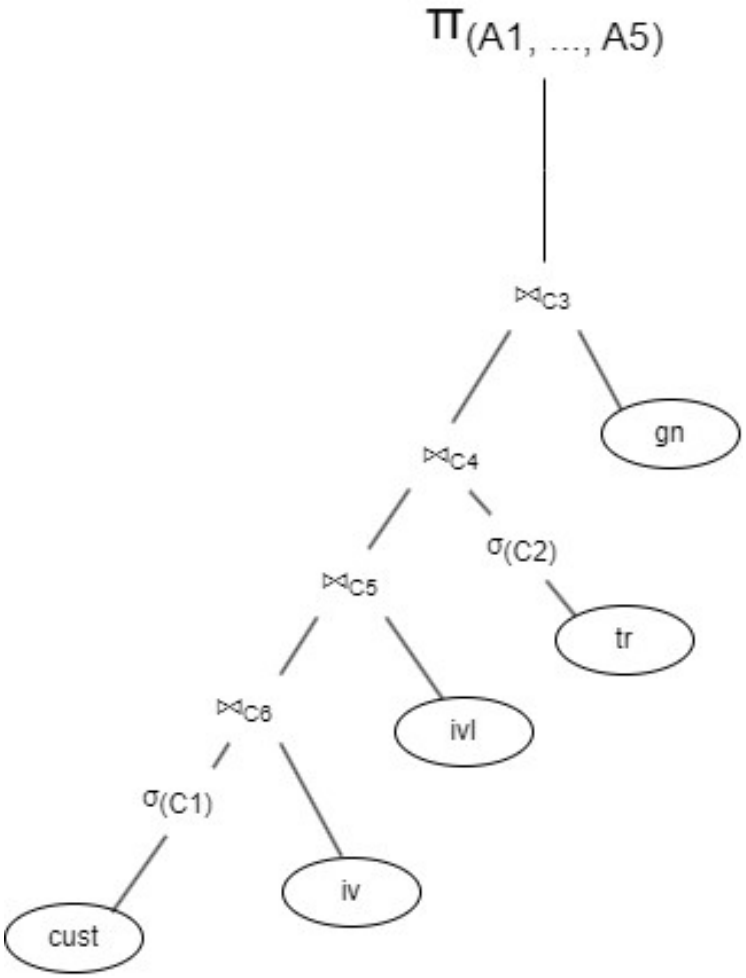


Gráfico 8

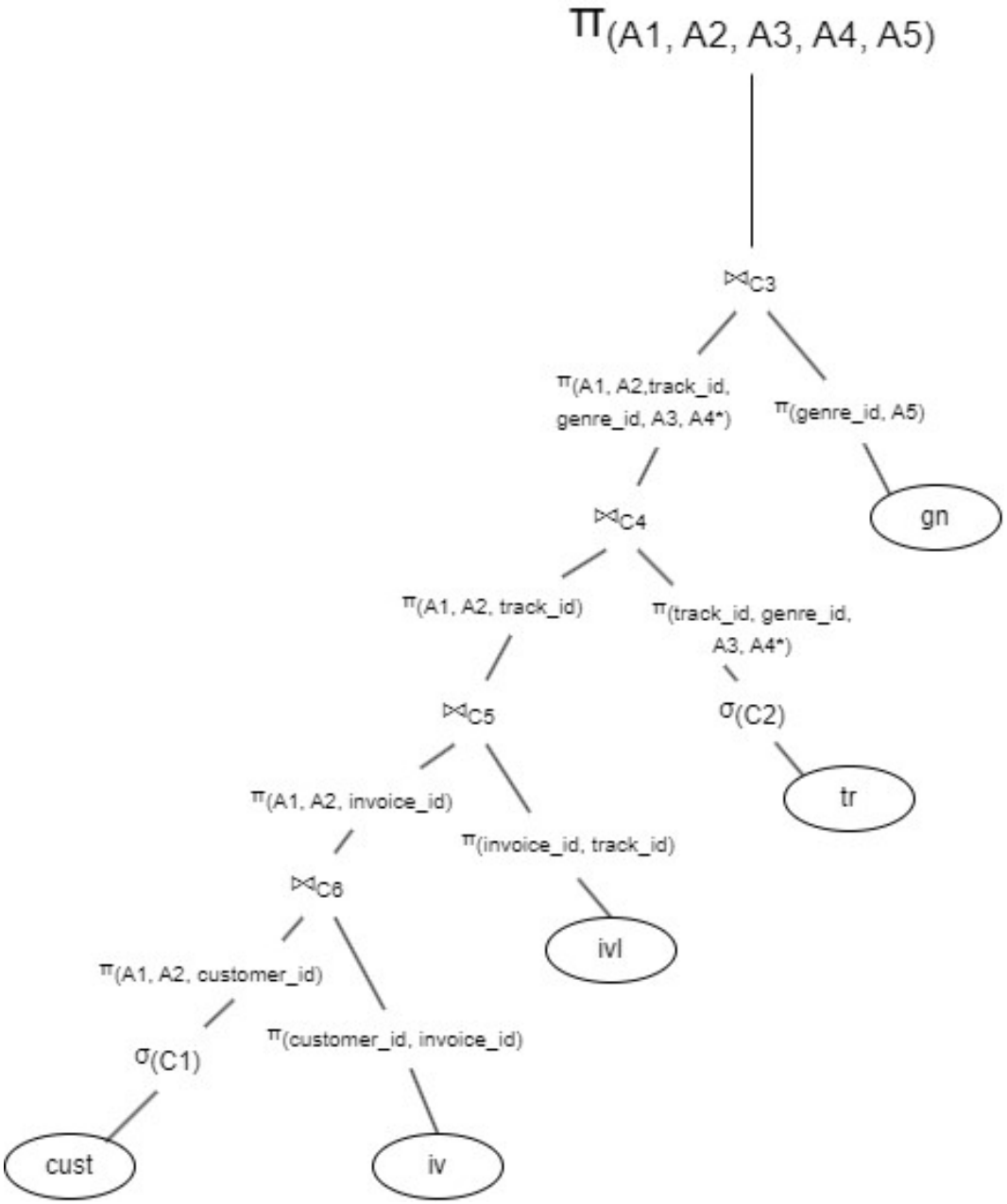
Árbol intermedio



En el gráfico 8 vemos el árbol intermedio, donde se busca optimizar la selección para así quedarnos con la menor cantidad de filas posibles.

Gráfico 9

Árbol de ejecución final



En el gráfico 9, vemos el árbol de ejecución optimizado, en donde se busca optimizar la proyección para quedarnos con la menor cantidad de columnas posibles.

Interfaz de acceso a datos

Para obtener un valor agregado a las consultas realizadas en la base de datos Chinook, hemos decidido utilizar una interfaz de acceso a datos en Python. En esta interfaz, hemos utilizado las bibliotecas pandas y matplotlib para generar gráficas que nos permitan visualizar de manera más clara y concisa los resultados obtenidos en las consultas. En particular, hemos utilizado gráficos de barras y de sectores para representar la información obtenida en algunas de las consultas realizadas. Con estas gráficas, podemos obtener información adicional sobre los datos, como por ejemplo, la distribución de clientes por países o los artistas más vendidos. El código de Python utilizado se encuentra con el nombre de “interface.py”.

Gráfico 10

Los 10 artistas más vendidos

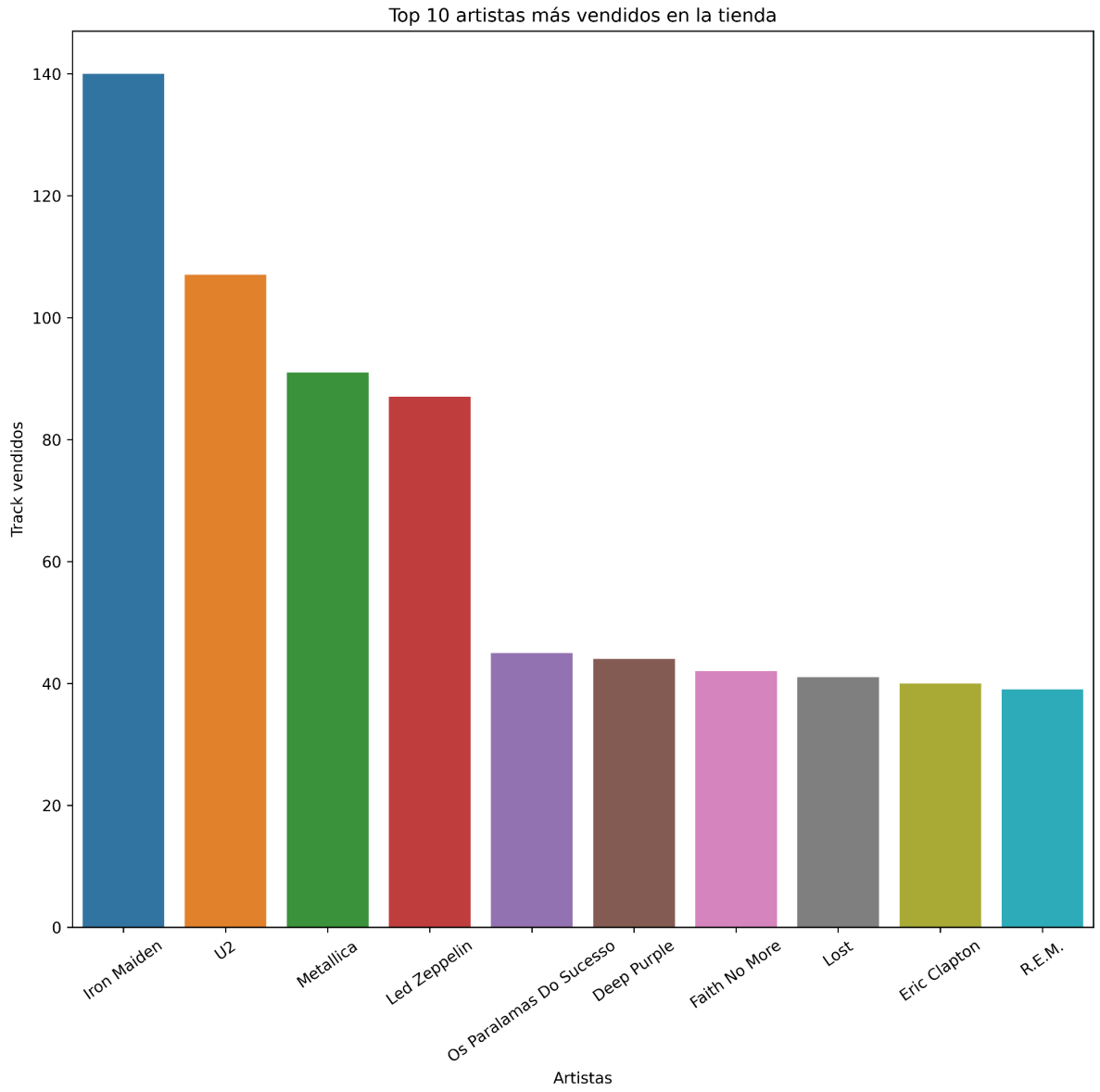


Gráfico 11

Distribución de clientes segmentado por países