UNIVERSIDAD ADOLFO IBAÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Tarea 2

TICS564: Tópicos en Data Management y Data Analytics

Profesor: Miguel Romero Ayudante: Isabel Donoso

Fecha de Entrega: Viernes 24 de Junio, 23:59 hrs.

Indicaciones:

1. Se aceptarán tareas enviadas después de la fecha de entrega. Se descontará 1.0 punto sobre la nota final por cada día de retraso.

2. La tarea se puede hacer en grupos de a lo más 3 personas. Debe respetar los grupos inscritos en la planilla de Webcursos.

3. **IMPORTANTE :** Debe subir a Webcursos un informe con sus respuestas a los problemas de la Parte 1 y Parte 2. En cada caso debe mostrar un **screenshot** de su cliente (pgAdmin u otro), en donde se vea la consulta y el resultado de ella (al menos las primeras tuplas). En caso que se pidan planes de ejecución o tiempos de ejecución **también** debe mostrar un screenshot. En caso que tenga que ejecutar la misma consulta varias veces para estimar tiempo de ejecución, basta que muestre el screenshot de una de esas ejecuciones.

   Adicionalmente, debe subir un .sql con todas las consultas utilizadas. Agregue comentarios para dejar claro a qué parte e item corresponde cada una.

Trabajaremos con una base de datos sobre el mismo esquema de la Tarea 1:

titles(id, title, release_year, rating, votes)
names(id, name, birth_year, death_year)
categories(title_id, category)
directed(title_id, name_id)
acted(title_id, name_id, role)

La base de datos también fue generada a partir de los datos del *Internet Movie Database (IMDb)* (https://www.imdb.com/interfaces/). Ahora se filtraron los datos considerando sólo títulos (es decir, películas) con al menos 5.000 votos a la hora de calcular el rating (la vez anterior eran al menos 100.000 votos, por ende la base de datos es más grande ahora). Para cargar esta base de datos se provee un archivo `dump_imdb2.sql` con los comandos para su creación.

Al igual que antes, debe crear una instancia en ElephantSQL (https://www.elephantsql.com/). Para conectarse al servidor, se recomienda utilizar pgAdmin (https://www.pgadmin.org/). Puede utilizar otro cliente si lo desea.

Observación: En las dos partes, la distribución de puntaje entre los items es uniforme.

Parte 1 (50%)

1. Escriba una consulta que entregue los *co-stars*, es decir, las parejas `(name1, name2)` de nombres de actores que han actuado en una misma película. Cada pareja debe consistir de dos nombres *distintos*, y en la salida *no* pueden haber duplicados. Observe que la relación es simétrica, es decir, si aparece `(name1, name2)` también aparece `(name2, name1)`.

2. Escriba una consulta recursiva que entregue todos los nombres de los actores que son *co-stars indirectos* de `'Kevin Bacon'`, es decir, que son alcanzables desde `'Kevin Bacon'` por una cadena de co-stars (de cualquier largo). Su respuesta *no* debe tener duplicados. Es probable que en su respuesta aparezca `'Kevin Bacon'` mismo, lo cual no es un problema.

   Debe escribir una consulta de la siguiente forma:

   ```
   WITH RECURSIVE co_star (name1, name2) AS (
   ......definicion de co_star...
   ),
   indirect_co_star_bacon (name) AS (
   ......definicion recursiva de indirect_co_star_bacon...
   )
   Consulta Final;
   ```

   Esta es la forma de concatenar varios `WITH`'s cuando alguno de ellos es recursivo, es decir, ponemos al principio `WITH RECURSIVE` *independiente* que algunas de las tablas que estamos creando no sea recursiva (por ejemplo `co_star`).

3. Haga lo mismo que en la pregunta anterior pero partiendo de la actriz `'Lyudmila Saveleva'`. Cambie el nombre de la tabla `indirect_co_star_bacon` por `indirect_co_star_saveleva`.

4. Implemente la misma consulta del punto (2) pero creando una tabla `indirect_co_star` que almacena las parejas `(name1, name2)` conectadas por una cadena de co-star. Es decir, la consulta tiene la estructura:

   ```
   WITH RECURSIVE co_star (name1, name2) AS (
   ......definicion de co_star...
   ),
   indirect_co_star (name1, name2) AS (
   ......definicion recursiva de indirect_co_star...
   )
   Consulta Final;
   ```

   ¿Qué puede observar cuando ejecuta su consulta? Discuta brevemente por qué sucede lo observado.

IMPORTANTE : Respete la estructura de las consultas recursivas. En particular la tabla `co_star` **debe** ser definida como tabla temporal con un `WITH` y no como una tabla nueva (`create table`) en la base de datos, ya que la base de datos podría exceder el tamaño permitido en ElephantSQL.

Parte 2 (50%)

1. Considere la siguiente consulta: Entregue los nombres (sin duplicados) de todos los actores que han hecho el rol de `'Batman'`.

   Escriba esta consulta en SQL de dos formas distintas: utilizando el operador `JOIN` y sin usar el operador `JOIN` (es decir, haciendo el cruce de tablas en el `FROM`). Asegúrese que las dos consultas entreguen los mismos resultados. Utilize `EXPLAIN ANALYZE` para visualizar

el plan de ejecución de cada consulta. ¿Qué puede observar de los planes de ejecución de ambas consultas (sólo los planes, no los tiempos)?

2. Tome la consulta anterior escrita con el operador `JOIN`. Utilice nuevamente `EXPLAIN ANALYZE` y entregue el tiempo de ejecución de la consulta. Esto aparece en el campo *Execution Time* (en milisegundos), y corresponde al tiempo que el motor toma en ejecutar la consulta y entregar los resultados (OJO: esto no incluye el tiempo de planificación, y es distinto al tiempo que pgAdmin tarda en ejecutar y desplegar la respuesta de la consulta). Como cada ejecución de la consulta puede dar distintos tiempos de ejecución, ejecute varias veces la consulta y entregue un estimado (un rango).

   Ahora agregue un índice *btree* en el atributo `role` de la table `acted`. Mida nuevamente el tiempo de ejecución de la consulta. Discuta sus resultados.

3. Escriba una consulta que entregue todos los atributos de los títulos con año de estreno 2022. Compare los tiempos de ejecución (usando `EXPLAIN ANALYZE`) de 3 casos: sin indices extras, un indice *btree* en el atributo `release_year` de la tabla `titles`, un indice *hash* en el atributo `release_year` de la tabla `titles`. Discuta sus resultados.

4. Escriba una consulta que entregue todos los atributos de los títulos con año de estreno entre 2018 y 2020 (utilice los operadores `and`, `>=` y `<=`; no use `between`). Compare los tiempos de ejecución (usando `EXPLAIN ANALYZE`) de 3 casos: sin indices extras, un indice *btree* en el atributo `release_year` de la tabla `titles`, un indice *hash* en el atributo `release_year` de la tabla `titles`. Discuta sus resultados.

IMPORTANTE : En la pregunta (3) y (4) asegurese de no tener los dos índices (btree y hash) simultaneamente. Recuerde que puede borrar índices con `DROP INDEX`. Además puede ver los índices disponibles con la siguiente consulta:

```
SELECT tablename, indexname, indexdef
FROM pg_indexes
WHERE schemaname = 'public'
ORDER BY tablename, indexname;
```

Si copia y pega esta consulta en el cliente, probablemente deberá arreglar el guión bajo (pg_indexes) y las comillas.