

Bases de Datos a Grande Escala

Entrega 3

ANDRÉS CAMPOS CUIÑA

FECHA DE ENTREGA: 10/12/2021

ÍNDICE

1	Ejercicio 1.....	1
2	Ejercicio 2.....	6

BDGE MaBD	Doc.: entrega3_CamposCuina.pdf
	Página 1 de 7

1 EJERCICIO 1

Probar el uso de la replicación. Para esto, borramos todas la tablas (en cascada) y las creamos de nuevo. Antes de crearlas modificar el script para eliminar las claves foráneas cláusulas *references*, ya que no se puede usar replicación en CITUS con claves foráneas. Antes de distribuir los datos, asignar el valor 2 a la variable de factor de replicación ("SET citus.shard_replication_factor = 2"). Cargar de nuevo los datos. Probar una de las consultas anteriores. Pausar una de las máquinas *worker*. Probar de nuevo la consulta. Intentar varias veces si es necesario, para que el coordinador se dé cuenta que uno de los *worker* no responde.

En primer lugar, debemos borrar todas las tablas (en cascada). Mediante el siguiente código de SQL podemos obtener todas las instrucciones para borrar las tablas:

```
select 'drop table if exists "' || tablename || '" cascade;'
from pg_tables
where schemaname = 'public';
```

Tras la ejecución del código SQL anterior obtenemos el siguiente código, que nos permitirá borrar todas las tablas:

```
drop table if exists "países" cascade;
drop table if exists "personas" cascade;
drop table if exists "colecciones" cascade;
drop table if exists "películas" cascade;
drop table if exists "idiomas" cascade;
drop table if exists "generos" cascade;
drop table if exists "pelicula_genero" cascade;
drop table if exists "pelicula_idioma_hablado" cascade;
drop table if exists "pelicula_pais" cascade;
drop table if exists "pelicula_personal" cascade;
drop table if exists "pelicula_productora" cascade;
drop table if exists "productoras" cascade;
drop table if exists "pelicula_reparto" cascade;
```

Una vez hemos eliminado todas las tabas, las volvemos a crear mediante el script que se nos proporciona en el enunciado de esta entrega, modificándolo para eliminar las claves foráneas (en la cláusula *references*), ya que no se puede usar replicación en CITUS con claves foráneas. Este script quedaría de la siguiente manera:

```
CREATE TABLE colecciones (
    id int4 not null,
    nombre text,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE generos (
    id int4 not null,
    nombre text,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE idiomas (
    id text NOT NULL,
    nombre text,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE paises (
    id text NOT NULL,
    nombre text,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE peliculas (
    id int4 NOT NULL,
    titulo text ,
    coleccion int4,
    para_adultos bool ,
    presupuesto int4 ,
    idioma_original text,
    titulo_original text ,
    sinopsis text ,
    popularidad float8 ,
    fecha_emision date ,
    ingresos int8 ,
    duracion float8 ,
    lema text ,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE personas (
    id int4 NOT NULL,
    nombre text ,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE productoras (
    id int4 NOT NULL,
    nombre text,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE pelicula_genero (
    pelicula int4 NOT NULL,
    genero int4 NOT NULL,
    PRIMARY KEY (pelicula, genero)
);
```

```
CREATE TABLE pelicula_idioma_hablado (
    pelicula int4 NOT NULL,
    idioma text NOT NULL,
    PRIMARY KEY (pelicula, idioma)
);
```

```
CREATE TABLE pelicula_pais (
    pelicula int4 NOT NULL,
    pais text NOT NULL,
    PRIMARY KEY (pelicula, pais)
);
```

```
CREATE TABLE pelicula_personal (
    pelicula int4 NOT NULL,
    persona int4 NOT NULL,
    departamento text,
    trabajo text NOT NULL,
    PRIMARY KEY (pelicula, persona, trabajo)
);
```

```
CREATE TABLE pelicula_productora (
    pelicula int4 NOT NULL,
    productora int4 NOT NULL,
    PRIMARY KEY (pelicula, productora)
);
```

```
CREATE TABLE pelicula_reparto (
    pelicula int4 NOT NULL,
    persona int4 NOT NULL,
    orden int4 NOT NULL,
    personaje text NOT NULL,
    PRIMARY KEY (pelicula, persona, personaje, orden)
);
```

Tras ejecutar este script ya tendríamos las tablas creadas de nuevo. A continuación, establecemos el factor de replicación al valor 2 mediante la siguiente instrucción:

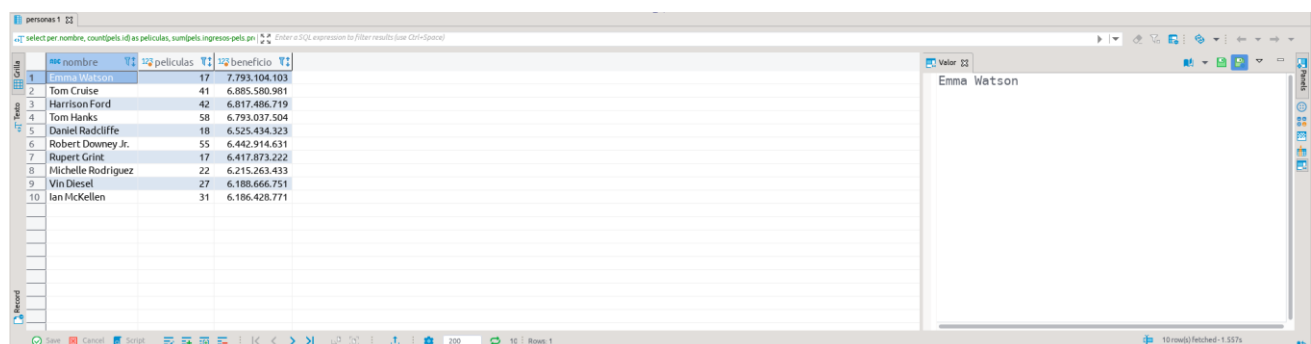
```
SET citus.shard_replication_factor = 2;
```

Después de ejecutar esta instrucción, debemos distribuir las tablas y ejecutar las instrucciones de inserción de datos tal y como se muestra en el enunciado de esta entrega.

Ahora probamos la ejecución de la consulta:

```
select
  per.nombre,
  count(pels.id) as peliculas,
  sum(pels.ingresos - pels.presupuesto) as beneficio
from
  peliculas as pels,
  personas as per,
  pelicula_reparto as pr
where
  pels.id = pr.pelicula
  and pr.persona = per.id
  and pr.orden < 5
group by per.nombre
order by beneficio desc
limit 10
```

Esta consulta nos da el siguiente resultado en un tiempo de 1,557s :



#	nombre	peliculas	beneficio
1	Emma Watson	17	7.793.104.103
2	Tom Cruise	41	6.885.580.981
3	Harrison Ford	42	6.817.486.719
4	Tom Hanks	58	6.793.037.504
5	Daniel Radcliffe	16	6.525.434.333
6	Robert Downey Jr.	55	6.442.914.631
7	Rupert Grint	17	6.417.873.222
8	Michelle Rodriguez	22	6.215.263.433
9	Vin Diesel	27	6.188.666.751
10	Ian McKellen	31	6.186.428.771

Ahora pausamos la máquina **cluster2** y volvemos a probar la ejecución de la misma consulta.

Después de que el coordinador se da cuenta de que uno de los nodos *worker* no responde, la consulta se puede ejecutar de forma correcta en el nodo que sigue activo, aunque el tiempo de ejecución es mayor (de 4,413s) debido a que no se puede paralelizar la consulta entre los dos nodos *worker*:

personas 1

Salida

select per.nombre, count(pels.id) as peliculas, sum(pels.ingresos-pels.presq) Enter a SQL expression to filter results (use Ctrl+Space)

	nombre	peliculas	beneficio
1	Emma Watson	17	7.793.104.103
2	Tom Cruise	41	6.885.580.981
3	Harrison Ford	42	6.817.486.719
4	Tom Hanks	58	6.793.037.504
5	Daniel Radcliffe	18	6.525.434.323
6	Robert Downey Jr.	55	6.442.914.631
7	Rupert Grint	17	6.417.873.222
8	Michelle Rodriguez	22	6.215.263.433
9	Vin Diesel	27	6.188.666.751
10	Ian McKellen	31	6.186.428.771

Valor

Emma Watson

10 row(s) Fetched - 4.413s

BDGE MaBD	Doc.: entrega3_CamposCuina.pdf
	Página 6 de 7

2 EJERCICIO 2

Buscar información sobre el formato de almacenamiento columnar en Citus (https://docs.citusdata.com/en/v10.1/admin_guide/table_management.html#columnar-storage). Probar a crear la tabla "películas" en formato columnar y comprobar si hay ganancia en tiempo de ejecución en las consultas anteriores.

Ejecutamos las dos consultas anteriores con los dos nodos *worker* activados y con un factor de replicación igual a 2. Para la consulta siguiente se obtiene un tiempo de ejecución de 2,441s:

```
select
  pels.titulo,
  pels.fecha_emision
from
  peliculas as pels,
  pelicula_personal as pp ,
  personas as per
where
  pels.id = pp.pelicula
  and pp.persona = per.id
  and per.nombre = 'Ridley Scott'
  and pp.trabajo = 'Director'
order by fecha_emision desc
```

Mientras que para la consulta siguiente se obtiene un tiempo de ejecución de 2,913s:

```
select
  per.nombre,
  count(pels.id) as peliculas,
  sum(pels.ingresos - pels.presupuesto) as beneficio
from
  peliculas as pels,
  personas as per,
  pelicula_reparto as pr
where
  pels.id = pr.pelicula
  and pr.persona = per.id
  and pr.orden < 5
group by per.nombre
order by beneficio desc
limit 10
```

A continuación, borramos las tablas como se hizo en el ejercicio anterior y las volvemos a crear con la tabla "películas" en formato columnar, mediante el siguiente código SQL:

```
CREATE TABLE peliculas (
  id int4 NOT NULL,
  titulo text ,
  coleccion int4,
  para_adultos bool ,
  presupuesto int4 ,
  idioma_original text,
  titulo_original text ,
  sinopsis text ,
  popularidad float8 ,
  fecha_emision date ,
  ingresos int8 ,
  duracion float8 ,
  lema text
) USING columnar;
```

La creación del resto de tablas es igual que antes. Después, seguimos los mismos pasos que en el ejercicio anterior para distribuir las tablas y para llevar a cabo la inserción de los datos. Una vez tenemos esto hecho ya podemos ejecutar las dos consultas anteriores y comparar los tiempos de ejecución.

Con la tabla “peliculas” en formato columnar la primera de las dos consultas anteriores se ejecuta en un tiempo de 761ms mientras que la segunda de las consultas anteriores se ejecutó en un tiempo de 1,625s.

Como se puede ver en la siguiente tabla:

	No Columnar	Columnar	Mejora
Consulta 1	2,441s	0,761s	x3,2
Consulta 2	2,913s	1,625s	X1,8

En ambas consultas, se ejecutó de forma más rápida la consulta que se corrió sobre la tabla “películas” en formato columnar. Esto se debe a que las consultas sobre esta tabla columnar son más rápidas y más eficientes que sobre una tabla que no lo sea. Esto se debe a que a que estas consultas sólo acceden a algunas de estas columnas, pero a todas las filas (debido a las agregaciones) por lo que al no tener que obtener las filas de diferentes nodos *worker* las consultas son más eficientes.