Bases de Datos a Gran Escala

Abraham Trashorras Rivas

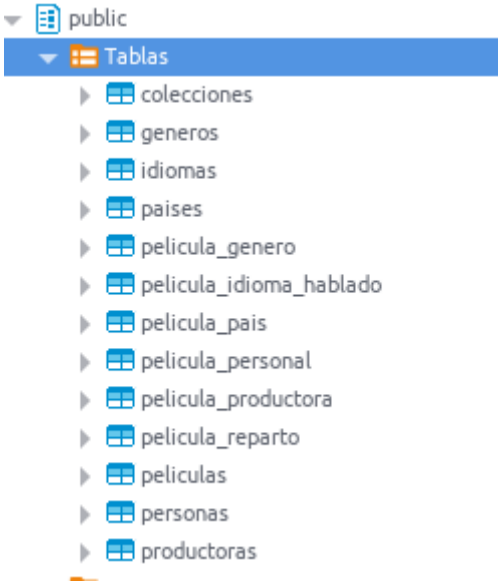
Práctica 3

Ejercicio 1:

Probar el uso de la replicación. Para esto, borramos todas las 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 de cuenta que uno de los worker no responde.

Borrar todas las tablas en cascada:

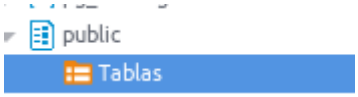
Tablas antes de borrar:



Ejecutamos:

```
DROP TABLE IF EXISTS paises CASCADE;
DROP TABLE IF EXISTS personas CASCADE;
DROP TABLE IF EXISTS colecciones CASCADE;
DROP TABLE IF EXISTS peliculas 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;
```

Tablas despues de borrar:



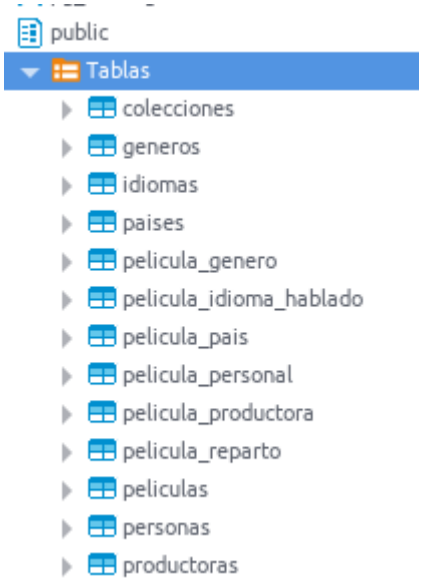
Crearlas sin claves foráneas en las cláusulas references:

Basado en el código de PeliculasSchema.sql

```
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)
);
```

Después de ejecutar los CREATE:



Asignamos factor de replicación:

SET citus.shard_replication_factor = 2;

Cargar de nuevo los datos:

Ejecutamos los comandos:

```
chmod 664 /home/alumnogreibd/BDGE/datos/*.csv
sudo -i -u postgres psql -c "\copy colecciones from
/home/alumnogreibd/BDGE/datos/colecciones.csv csv"
sudo -i -u postgres psql -c "\copy generos from
/home/alumnogreibd/BDGE/datos/generos.csv csv"
sudo -i -u postgres psql -c "\copy idiomas from
/home/alumnogreibd/BDGE/datos/idiomas.csv csv"
sudo -i -u postgres psql -c "\copy paises from
/home/alumnogreibd/BDGE/datos/paises.csv csv"
sudo -i -u postgres psql -c "\copy personas from
/home/alumnogreibd/BDGE/datos/personas.csv csv"
sudo -i -u postgres psql -c "\copy productoras from
/home/alumnogreibd/BDGE/datos/productoras.csv csv"
sudo -i -u postgres psql -c "\copy peliculas from
/home/alumnogreibd/BDGE/datos/peliculas.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_genero from
/home/alumnogreibd/BDGE/datos/pelicula_genero.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_idioma_hablado from
/home/alumnogreibd/BDGE/datos/pelicula_idioma_hablado.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_pais from
/home/alumnogreibd/BDGE/datos/pelicula_pais.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_personal from
/home/alumnogreibd/BDGE/datos/pelicula_personal.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_productora from
/home/alumnogreibd/BDGE/datos/pelicula_productora.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_reparto from
/home/alumnogreibd/BDGE/datos/pelicula_reparto.csv csv"
```

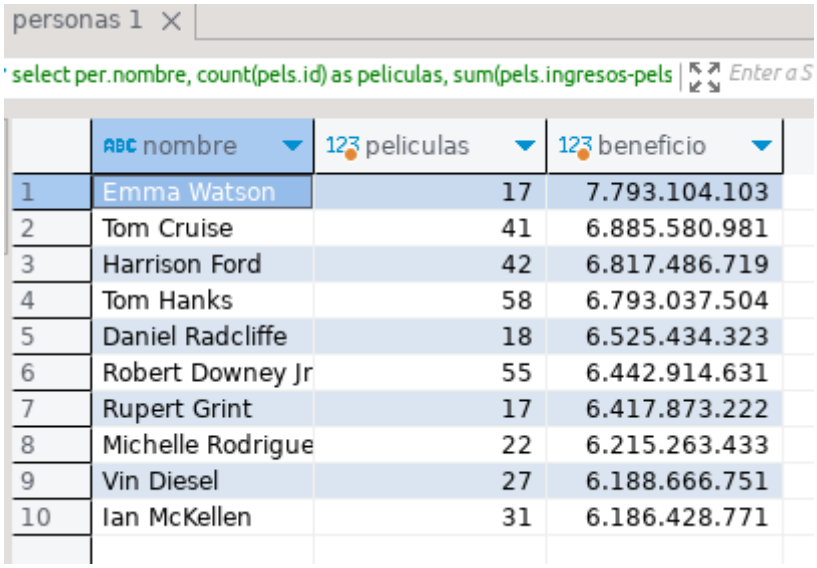
El resultado es:

Probamos consulta:

Primero ejecutamos:

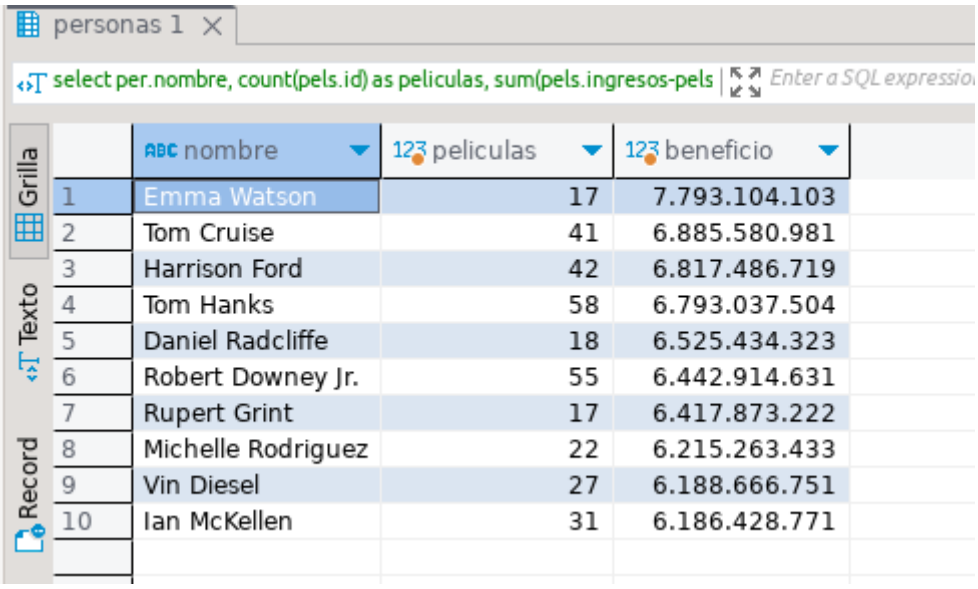
```
select per.nombre, count(pels.id) as peliculas, sum(pels.ingresos-
pels.presupuesto) as beneficio
from peliculas pels, pelicula_reparto pr, personas per
where pels.id=pr.pelicula and pr.persona=per.id
  and pr.orden<5
group by per.nombre
order by beneficio desc
limit 10
```

Y nos da:



	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 Rodrigue	22	6.215.263.433
9	Vin Diesel	27	6.188.666.751
10	Ian McKellen	31	6.186.428.771

Ahora pausamos citius2 y ejecutamos el mismo código. La consulta sigue devolviendo resultado, pero se ve que el tiempo de ejecución es mayor, casi de medio segundo:



	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

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 "peliculas" en formato columnar y comprobar si hay ganancia en tiempo de ejecución en las consultas anteriores.

Medimos el tiempo de las dos consultas del guion con EXPLAIN ANALIZE:

Consulta 1: 77.6ms
Consulta 2: 290.4 ms

Preparamos el sistema para columnar:

Realizamos los pasos del ejercicio anterior, haciendo drop a todas las tablas, volviendo a crearlas e insertarles los datos. Esta vez, modificamos la tabla de películas de esta forma:

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

Insertamos los datos como en el ejercicio anterior.

Medimos el tiempo de las dos consultas del guion con EXPLAIN ANALIZE:

Consulta 1: 68.6ms
Consulta 2: 236.4 ms

He visto una ligera mejora con columnar.