Bases de Datos a Gran Escala

Abraham Trashorras Rivas

Práctica 5

Importación de los datos:

E1:

```
// Conexión a la base de datos PostgreSQL

WITH "jdbc:postgresql:bdge?user=alumnogreibd&password=greibd2021" AS url

CALL apoc.load.jdbc(url,

"SELECT id, titulo, presupuesto, fecha_emision, ingresos, duracion

FROM peliculas

ORDER BY ingresos DESC, id

LIMIT 2000") YIELD row

CREATE (p:Pelicula {

    id: row.id,

    titulo: row.titulo,

    presupuesto: row.presupuesto,

    fechaEmision: row.fecha_emision,

    ingresos: row.ingresos,

    duracion: row.duracion

});


// Crear índices
CREATE INDEX ON :Pelicula(id);
CREATE INDEX ON :Pelicula(titulo);
```

E2:

```
// Conexión a la base de datos PostgreSQL

WITH "jdbc:postgresql:bdge?user=alumnogreibd&password=greibd2021" AS url

CALL apoc.load.jdbc(url,

"SELECT DISTINCT per.id AS id, per.nombre AS nombre

 FROM (

     SELECT pp.persona AS id, per.nombre AS nombre

     FROM pelicula_personal AS pp

     JOIN personas AS per ON pp.persona = per.id

     WHERE pp.pelicula IN (SELECT id FROM peliculas ORDER BY ingresos DESC, id LIMIT
2000)
```

```
     UNION ALL
     SELECT pr.persona AS id, per.nombre AS nombre
     FROM pelicula_reparto AS pr
     JOIN personas AS per ON pr.persona = per.id
     WHERE pr.pelicula IN (SELECT id FROM peliculas ORDER BY ingresos DESC, id LIMIT
2000)
 ) AS per") YIELD row
CREATE (p:Persona {
     id: row.id,
     nombre: row.nombre
});


// Crear índices
CREATE INDEX ON :Persona(id);
CREATE INDEX ON :Persona(nombre);
```

E3:

```
// Conexión a la base de datos PostgreSQL
WITH "jdbc:postgresql:bdge?user=alumnogreibd&password=greibd2021" AS url
CALL apoc.load.jdbc(url,
"SELECT pr.persona AS persona, pr.personaje AS personaje, pr.pelicula AS pelicula
FROM pelicula_reparto AS pr
WHERE pr.pelicula IN (SELECT id FROM peliculas ORDER BY ingresos DESC, id LIMIT
2000)") YIELD row
MATCH (pelicula:Pelicula {id: row.pelicula})
MATCH (persona:Persona {id: row.persona})
CREATE (persona)-[r:ACTUO_EN {personaje: row.personaje}]->(pelicula);
```

E4:

```
// Conexión a la base de datos PostgreSQL
```

```
WITH "jdbc:postgresql:bdge?user=alumnogreibd&password=greibd2021" AS url

CALL apoc.load.jdbc(url,

"SELECT pp.persona AS persona, pp.trabajo AS trabajo, pp.pelicula AS pelicula,
pp.departamento AS departamento

FROM pelicula_personal AS pp

WHERE pp.pelicula IN (SELECT id FROM peliculas ORDER BY ingresos DESC, id LIMIT
2000)") YIELD row

MATCH (pelicula:Pelicula {id: row.pelicula})

MATCH (persona:Persona {id: row.persona})

MERGE (persona)-[r:TRABAJO_EN {trabajo: row.trabajo, departamento:
row.departamento}]->(pelicula);
```

Consultas:

E1:

Consulta:

```
MATCH (persona:Persona)-[r:TRABAJO_EN]->(pelicula:Pelicula)
            WHERE r.trabajo = "Director"
            CREATE (persona)-[:DIRIGE]->(pelicula);
```

E2:

Consulta:

```
MATCH (pelicula:Pelicula {titulo: "Star Wars"})<-[:ACTUO_EN]-(persona:Persona)
            WITH persona
            ORDER BY persona.orden
            RETURN persona.nombre AS ActorActriz, persona.id AS IdActorActriz,
                  COLLECT({personaje: persona.nombre, orden: persona.orden}) AS
Reparto;
```

E3:

Consulta:

```
MATCH (pelicula:Pelicula)
            WITH pelicula, pelicula.ingresos - pelicula.presupuesto AS beneficio
            ORDER BY beneficio DESC
            LIMIT 10
            RETURN pelicula.titulo AS TituloPelicula, beneficio AS Beneficio;
```

E4:

Consulta:

```
MATCH (persona:Persona {nombre: "Quentin Tarantino"})--(pelicula:Pelicula)
WITH persona, pelicula,
```

```
    COLLECT(DISTINCT CASE
        WHEN (persona)-[:DIRIGE]->(pelicula) THEN "Director"
        WHEN (persona)-[:ACTUO_EN]->(pelicula) THEN "Actor"
        ELSE "Otro"
    END) AS participaciones
RETURN pelicula.titulo AS TituloPelicula, pelicula.fechaEmision AS FechaEmision,
       pelicula.presupuesto AS Presupuesto, pelicula.ingresos AS Ingresos,
       participaciones
```

ORDER BY pelicula.fechaEmision;

E5:

Consulta:

```
MATCH (pelicula:Pelicula {titulo: "The Godfather"})<-[r:TRABAJO_EN]-(persona:Persona)
WITH DISTINCT persona, r
ORDER BY r.departamento
RETURN r.departamento AS Departamento,
       COUNT(DISTINCT persona) AS NumeroPersonas,
       COLLECT({nombre: persona.nombre, trabajo: r.trabajo}) AS Personas;
```

E6:

Consulta:

```
// Identificar la película con mayores ingresos en la que trabajó Spielberg
MATCH (spielberg:Persona {nombre: "Steven Spielberg"})-->(pelicula:Pelicula)
WITH pelicula
ORDER BY pelicula.ingresos DESC LIMIT 1

// Contar el número de actores en la película
MATCH (pelicula)<-[r:ACTUO_EN]-(actor:Persona)
WITH pelicula, COLLECT(DISTINCT actor) AS actores

// Contar otros trabajadores, excluyendo a los actores
MATCH (pelicula)<-[r:TRABAJO_EN]-(trabajador:Persona)
WHERE NOT trabajador IN actores
WITH pelicula, SIZE(actores) AS NumeroActores, COUNT(DISTINCT trabajador) AS
NumeroTrabajadores

// Obtener detalles finales de la película
RETURN pelicula.titulo AS Titulo,
```

```
       pelicula.presupuesto AS Presupuesto,
       pelicula.ingresos AS Ingresos,
       NumeroActores AS NumeroActores,
       NumeroTrabajadores AS NumeroTrabajadores;
```

E7:

Consulta:

```
MATCH (brando:Persona {nombre: "Marlon Brando"})-[:ACTUO_EN]->(p:Pelicula)<-
[:DIRIGE]-(director:Persona)
            WITH DISTINCT director

            // Encontrar películas dirigidas por esos directores
            MATCH (director)-[:DIRIGE]->(pelicula:Pelicula)

            // Encontrar actores y actrices de esas películas
            MATCH (pelicula)<-[:ACTUO_EN]-(actor:Persona)
            RETURN DISTINCT actor.nombre AS NombreActor
            ORDER BY NombreActor;
```

E8:

Consulta:

```
MATCH (director:Persona)-[:DIRIGE]->(pelicula:Pelicula)
WITH pelicula, COLLECT(director.nombre) AS directores
WHERE SIZE(directores) > 1
RETURN pelicula.titulo AS Titulo, directores, SIZE(directores) AS NumeroDeDirectores
ORDER BY NumeroDeDirectores DESC;
```

E9:

```
MATCH (persona:Persona)-[rol:TRABAJO_EN]->(pelicula:Pelicula)
WITH persona, pelicula, COLLECT(rol.trabajo) AS roles
ORDER BY SIZE(roles) DESC
LIMIT 10
RETURN persona.id AS IdPersona, persona.nombre AS Nombre, pelicula.titulo AS
TituloPelicula, SIZE(roles) AS NumeroRoles, roles AS Roles;
```

E10:

```
MATCH (tarantino:Persona {nombre: "Quentin Tarantino"})
            MATCH (actor:Persona)
            WHERE (actor)-[:ACTUO_EN]->(:Pelicula)<-[:ACTUO_EN]-(:Persona)-
[:DIRIGE]->(:Pelicula {director: "Quentin Tarantino"})
            OR (actor)-[:ACTUO_EN]->(:Pelicula)<-[:ACTUO_EN]-(:Persona {nombre:
"Quentin Tarantino"})
```

```
RETURN DISTINCT actor.nombre AS NombreActor
ORDER BY NombreActor;
```