## Exercici 1:

```
--
-- Creació del keyspace per a l'aplicació de vídeo
--
CREATE KEYSPACE IF NOT EXISTS videossandra WITH REPLICATION = {'class':'SimpleStrategy',
'replication_factor':1};
--
USE videossandra;
-- Creació de taules
-- Creació de la taula d'usuaris
-- id_user és la clau de la partició
-- Com no hi ha cap query freqüent, es crea nua estructura genèrica
-- La query per email es farà per índex
CREATE TABLE user (
        id_user text PRIMARY KEY,
        email text,
        name text,
        surname text,
        register_date timeuuid
);
-- Creació de la taula de vídeos
-- Com a clau del cluster farem servir l'identificador de l'usuari
-- D'aquesta forma, es desarà al mateix cluster que l'usuari que l'ha pujat
-- La clau primària està formada per l'usuari i el vídeo
-- No cream clustering columns
CREATE TABLE video (
        id_user text,
        id_video text,
        description text,
        type text,
        URL text,
```

```
    upload_date timestamp,
    puntuation_avg float,
    genre_list set<text>,
    PRIMARY KEY (id_user, id_video)
);
-- Creació de la taula de vídeos per ordre invers d'inserció
-- Com a clau del cluster farem servir una cadena que indica el tipus de producte; això ens permetrà ordenar per dates
-- La clau primària està formada per aquest valor, el moment d'inserció, l'usuari, el vídeo (aquest dos darrers es poden evitar)
-- d'aquesta manera, conseguirem veure sempre el darrer vídeo publicat
-- Les clustering columns són upload_date, id_user i id_video
CREATE TABLE video_by_upload_date (
    producte text,
    id_user text,
    id_video text,
    description text,
    type text,
    URL text,
    upload_date timestamp,
    puntuation_avg float,
    genre_list set<text>,
    PRIMARY KEY ((producte), upload_date, id_user, id_video)
) WITH CLUSTERING ORDER BY (upload_date DESC);
-- Creació de la taula de comentaris; per a afavorir les cerques, tendrem dues taules (una per usuari i una altra per video)
-- id_user és la clau de la partició
-- comment_date és les clustering columns
-- L'ordenació es fa per comment_date per tal d'afavorir una de les cerques més comuns
CREATE TABLE comment_by_user (
    id_user text,
    id_video text,
```

```
    comment_date timestamp,
    comment text,
    PRIMARY KEY ((id_user), comment_date)
) WITH CLUSTERING ORDER BY (comment_date DESC);
-- Creació de la taula de comentaris; per a afavorir les cerques, tendrem dues taules (una per usuari i una altra per video)
-- id_user és la clau de la partició
-- comment_date és les clustering columns
-- L'ordenació es fa per comment_date per tal d'afavorir una de les cerques més comuns
CREATE TABLE comment_by_video (
    id_user text,
    id_video text,
    comment_date timestamp,
    comment text,
    PRIMARY KEY ((id_video), comment_date)
) WITH CLUSTERING ORDER BY (comment_date DESC);
-- Creació de la tauna de valoracions
-- El camp de puntuació és un float i no un enter perque entenc que es poden assignar decimals
-- S'ha afegit el camp de id_user per a assignar una partició (tanmateix, tenim la dada)
-- S'ha afegit el camp puntuation_date per a dur el registre de la data i crear una PK
CREATE TABLE puntuation (
    id_user text,
    id_video text,
    puntuation_date timestamp,
    puntuation float,
    PRIMARY KEY (id_user, id_video, puntuation_date)
);
--
-- Inserir registres
--
-- Usuaris
```

```
INSERT INTO user (id_user, email, name, surname, register_date) VALUES ('usu1', 'usu1@uoc.edu', 'Tommy', 'Ramone', now());

INSERT INTO user (id_user, email, name, surname, register_date) VALUES ('usu2', 'usu2@uoc.edu', 'Marky', 'Ramone', now());

INSERT INTO user (id_user, email, name, surname, register_date) VALUES ('usu3', 'usu3@uoc.edu', 'Johnny', 'Ramone', now());

INSERT INTO user (id_user, email, name, surname, register_date) VALUES ('usu4', 'usu4@uoc.edu', 'Joey', 'Ramone', now());

INSERT INTO user (id_user, email, name, surname, register_date) VALUES ('usu5', 'usu5@uoc.edu', 'Dee Dee', 'Ramone', now());

-- Vídeos

INSERT INTO video (id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('usu1', 'Pulp Fiction', 'Pelicula de Tarantino','Llargmetratge', 'www.pulfiction.com', toTimestamp(now()), 8, {'Comedia', 'Tiros'});

INSERT INTO video (id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('usu2', 'Citizen Kane', 'Pelicula de Welles','Llargmetratge', 'www.citizen.com', toTimestamp(now()), 9.5, {'Drama', 'Biopic'});

INSERT INTO video (id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('usu1', 'Casablanca', 'Pelicula de Curtis','Llargmetratge', 'www.casablanca.com', toTimestamp(now()), 9, {'Drama'});

INSERT INTO video (id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('usu1', 'Cinema Paradiso', 'Pelicula de Tornatore','Llargmetratge', 'www.cinema.com', toTimestamp(now()), 8, {'Drama'});

INSERT INTO video (id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('usu2', 'El verdugo', 'Pelicula de Berlanga','Llargmetratge', 'www.verdugo.com', toTimestamp(now()), 8, {'Comedia negra'});

--

INSERT INTO video_by_upload_date (producte, id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('video', 'usu1', 'Pulp Fiction', 'Pelicula de Tarantino','Llargmetratge', 'www.pulfiction.com', toTimestamp(now()), 8, {'Comedia', 'Tiros'});

INSERT INTO video_by_upload_date (producte, id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('video', 'usu2', 'Citizen Kane', 'Pelicula de Welles','Llargmetratge', 'www.citizen.com', toTimestamp(now()), 9.5, {'Drama', 'Biopic'});

INSERT INTO video_by_upload_date (producte, id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('video', 'usu1', 'Casablanca', 'Pelicula de Curtis','Llargmetratge', 'www.casablanca.com', toTimestamp(now()), 9, {'Drama'});

INSERT INTO video_by_upload_date (producte, id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('video', 'usu1', 'Cinema Paradiso', 'Pelicula de Tornatore','Llargmetratge', 'www.cinema.com', toTimestamp(now()), 8, {'Drama'});
```

```
INSERT INTO video_by_upload_date (producte, id_user, id_video, description, type, URL, upload_date, puntuation_avg, genre_list) VALUES ('video', 'usu2', 'El verdugo', 'Pelicula de Berlanga','Llargmetratge', 'www.verdugo.com', toTimestamp(now()), 8, {'Comedia negra'});

-- Comentaris

INSERT INTO comment_by_user (id_user, id_video, comment_date, comment) VALUES ('usu1', 'Pulp Fiction', toTimestamp(now()), 'molt bona peli');

INSERT INTO comment_by_video (id_user, id_video, comment_date, comment) VALUES ('usu1', 'Pulp Fiction', toTimestamp(now()), 'molt bona peli');

INSERT INTO comment_by_user (id_user, id_video, comment_date, comment) VALUES ('usu2', 'Pulp Fiction', toTimestamp(now()), 'no me va agradar');

INSERT INTO comment_by_video (id_user, id_video, comment_date, comment) VALUES ('usu2', 'Pulp Fiction', toTimestamp(now()), 'no me va agradar');

INSERT INTO comment_by_user (id_user, id_video, comment_date, comment) VALUES ('usu1', 'Casablanca', toTimestamp(now()), 'està molt bé');

INSERT INTO comment_by_video (id_user, id_video, comment_date, comment) VALUES ('usu1', 'Casablanca', toTimestamp(now()), 'està molt bé');

-- Puntuacions

INSERT INTO puntuation(id_user, id_video, puntuation_date, puntuation) values ('usu1', 'Pulp Fiction', toTimestamp(now()), 10);

INSERT INTO puntuation(id_user, id_video, puntuation_date, puntuation) values ('usu2', 'Pulp Fiction', toTimestamp(now()), 5);

--

-- Creació d'indexos per a afavorir les cerques

--

-- Videos by gènere

CREATE INDEX video_by_genre_idx ON video (genre_list);

-- Usuaris per email especificat

CREATE INDEX users_by_email_idx ON user (email);

-- Queries

-- Vídeos per gèneres

SELECT * FROM video WHERE genre_list CONTAINS 'Drama';

-- Comentaris per usuari ordenats per data de registre

SELECT * FROM comment_by_user WHERE id_user = 'usu1';

-- Comentaris per vídeo ordenats per data de registre
```

```
SELECT * FROM comment_by_video WHERE id_video = 'Pulp Fiction';

-- Trobar un usuari per email especificat

SELECT * FROM user WHERE email = 'usu1@uoc.edu';

-- Trobar l'últim vídeo pujat

SELECT * FROM video_by_upload_date LIMIT 1;

-- Trobar els comentaris sobre un vídeo concret

SELECT * FROM comment_by_video WHERE id_video = 'Pulp Fiction';

-- Vídeos per gènere

SELECT * FROM video WHERE genre_list CONTAINS 'Drama';
```

# Exercici 2: MongoDB

-- 1, Trobar tots els usuaris que hagin bloquejat a usuaris que procedeixen de

-- Pamplona i tenen entre 20 i 40 anys. Mostra el nom i cognoms. (Pista: 2

-- registres)

-- Registrs:

```
mongodb.Usuarios_bloqueadores.find({'Bloqueos.Usuario_bloqueado.Ciudad' : 'Pamplona', 'Bloqueos.Usuario_bloqueado.Edad' : {$gt : 19}, 'Bloqueos.Usuario_bloqueado.Edad' : {$lt : 41}}, {_id : 0, "Nombre" : 1, "Apellidos" : 1, "Bloqueos.Usuario_bloqueado.Ciudad" : 1, "Bloqueos.Usuario_bloqueado.Edad" : 1}).pretty()
```

-- Nombre de registres:

```
db.Usuarios_bloqueadores.find({'Bloqueos.Usuario_bloqueado.Ciudad' : 'Pamplona', 'Bloqueos.Usuario_bloqueado.Edad' : {$gt : 19}, 'Bloqueos.Usuario_bloqueado.Edad' : {$lt : 41}}).count()
```

-- 2. Trobar quants usuaris són membres alhora dels grups: "Intercanvi de videojocs" i "Jugadors d'escacs"

```
db.Usuarios_grupos.find({'Grupos_miembro.Nombre_Grupo' : {$all : ['Intercambio de videojuegos', 'Jugadores de ajedrez']}}, {_id:0, "Nombre":1, "Apellidos":1})
```

-- 3. Trobar tots els grups que tinguin membres de Valladolid o que no en tinguin cap de Madrid. Mostra el nom de grup i identificador del grup.

```
db.Grupos_usuarios.find({$or : [{'Miembros.Ciudad' : {$nin : ['Madrid']}}, {'Miembros.Ciudad' : 'Valladolid'}]}, {_id : 0, Identificador : 1})
```

-- 4. Crear la col·lecció "ciutats_usuaris" que contingui per cada ciutat un llistat d'usuaris amb les dades: nom, cognoms,

-- email i nombre de grups als quals pertany

-- Primer cream una variable que contendrà

```
var groupdata = db.Contactos_usuarios.aggregate(
  {
    $group : {
      _id : '$Ciudad', usuarios : {
        $addToSet : {
```

```
            Id : 'Identificador',
            Nombre : '$Nombre',
            Apellidos : '$Apellidos',
            Email : '$Email'
          }
        }
      }
    }
)
-- Cream la col·lecció amb les dades (fa)
db.ciutats_usuaris.insert(groupdata.toArray())
-- Cream una variable amb els usuaris i el nombre de grups per usuaris
var resumen_grupo = db.Usuarios_grupos.aggregate (
  [
    {
      $project : {_id : 1, Id : "$Identificador", Numero_grupos:{}$size:"$Grupos_miembro"}
    }
  ]
)
db.resumen_grups_usuaris.insert(resumen_grupo.toArray())

-- 5. Indicar quins índexs crearies per millorar el rendiment de la consulta.
-- Jo crearia un índex que inclogués els camps dels filtres
-- i els camps a mostrar per tal d'evitar accedir a la col·lecció original

db.Usuario_desbloqueado.createIndex(
  {
    "Ciudad":1,
    "Edad":1,
    "_id":1,
    "Nombre":1,
```

```
        "Apellidos":1,
    }
)
```

# Exercici 3: Neo4j

1. Considerar l'usuari rellevant amb un nombre de seguidors, comptabilitzant els seguidors de primer i segon nivell de forma conjunta, és de 1436. Mostra el nom i el nombre de seguidors.

```
MATCH (user:TwitterUser) <- [:FOLLOWS * 1..2] - (follower:TwitterUser)
WITH user, count(follower) as followers
WHERE followers = 1436
RETURN user.userName, followers
```

2. Considerar el tweet que ha tingut més retweets i respostes. Mostra la suma total de retweets i respostes, i els 20 primers caràcters del tweet.

```
MATCH (reply:Tweet) - [:IS_A_REPLY_OF] -> (originalTweet:Tweet),
   (retweet:Tweet) - [:IS_A_RETWEET_OF] -> (originalTweet:Tweet)
WITH originalTweet, count(DISTINCT  reply) as numberReplies, count(DISTINCT  retweet) as numberRetweet
ORDER BY (numberReplies + numberRetweet) DESC
RETURN substring (originalTweet.text, 0, 20), numberReplies, numberRetweet
LIMIT 1
```

3. Obtenir el percentatge de tweets que estan geolocalitzats a la latitud 42.0355

```
-- Nota: Multipliquem per 100.0 i 1.0 perque si no, ho tracta com a sencer i el resultat és 0
-- Estic forçant un producte cartesià per a obtenir tots els tweets; el resultat és el mateix sense el distinct de count(DISTINCT  glt)
--  això és perque només hi ha 1 (si hi ha més, ja no se compleix)

MATCH (glt:GeoLocatedTweet {lat: 42.0355}),
   (tweet:Tweet)
WITH count(DISTINCT  glt) as number_glt, count(DISTINCT  tweet) as number_tweet
RETURN (number_glt * 100.0) / (number_tweet * 1.0)
```

4. Considerar el sisè usuari que ha enviat més tweets geolocalitzats. Mostra el seu nom i el nombre de tweets.

```
MATCH (glt:GeoLocatedTweet) <- [:HAS_WRITEN] - (user:TwitterUser)
WITH count(glt) as number_glt, user
ORDER BY number_glt DESC
RETURN user.userName, number_glt
SKIP 5
LIMIT 1
```

5. Considerar els tweets geolocalitzats que són rèpliques a altres tweets geolocalitzats. Mostra el nombre de tweets que són rèpliques on la diferència en valor absolut entre les latituds que caracteritzen la rèplica i al tweet replicat és més gran que 20.

```
MATCH (reply:GeoLocatedTweet) - [:IS_A_REPLY_OF] -> (originalTweet:GeoLocatedTweet)
WHERE (reply.lat - originalTweet.lat < -20 OR reply.lat - originalTweet.lat > 20)
RETURN count(reply) as repliesNumber
```

# Exercici 4: Neo4j

Es vol organitzar la guia de carrers d'Espanya a nivell de ciutats. Per a això cal tenir en compte que els noms dels carrers es poden repetir entre les diferents unitats de l'estructura territorial excepte a nivell d'un mateix municipi. En un mateix municipi no poden existir dos carrers diferents amb el mateix nom. Si es realitza una simplificació, l'estructura territorial seria la següent: Comunitats Autònomes, Províncies, i municipis. S'ha decidit utilitzar una base de dades en graf per emmagatzemar tota la informació.

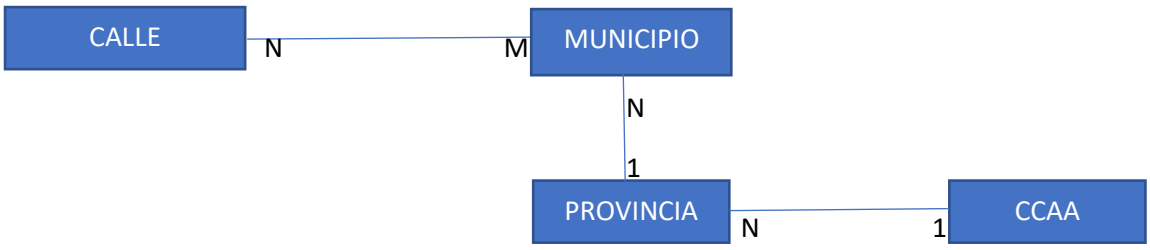
Es demana :

a) Argumentar quina és la forma més eficient d'organitzar la informació en una base de dades orientada a grafs si es volen recuperar els noms dels carrers.

   Considerant:
   - Els noms dels carrers no es poden repetir a un municipi.
   - Del carrer no tenim més dada que el nom (ni localització ni cap altra dada)

   La forma més òptima i funcional de emmagatzemar la informació és creant un node per nom de carrer i no per carrer físic. D'aquesta forma, creem una estructura en forma d'estrella en la que un carrer pot pertànyer a més d'un municipi. Si volem llegir tots els noms dels carrers, no és necessari filtrar ni eliminar duplicats.

   Assegurem que un municipi no pot tenir més d'un carrer amb el mateix nom creant una constraint. Crearem una constraint a la resta d'entitats per la mateixa raó.




b) A partir de les següents dades crea, usant Cypher en Neo4J, una base de dades d'acord amb la solució plantejada en a). Has de mostrar les consultes en Cypher així com captures dels grafs creats.

-- Creació de carrers

```
CREATE (n1:Calle { nombre: 'Ramón y Cajal'})

CREATE (n2:Calle { nombre: 'Cervantes'})
```

-- Creació de municipis

```
CREATE (n1:Municipio { nombre: 'Torrejón de Ardoz'})

CREATE (n2:Municipio { nombre: 'Segovia'})

CREATE (n3:Municipio { nombre: 'Albolote'})

CREATE (n4:Municipio { nombre: 'Onda'})

CREATE (n5:Municipio { nombre: 'Tarrasa'})

CREATE (n6:Municipio { nombre: 'Alcalá de Henares'})

CREATE (n7:Municipio { nombre: 'Peñaranda de Bracamonte'})

CREATE (n8:Municipio { nombre: 'Armilla'})

CREATE (n9:Municipio { nombre: 'Villajoyosa'})

CREATE (n10:Municipio { nombre: 'Badalona'})


-- Creació de províncies

CREATE (n1:Provincia { nombre: 'Madrid'})

CREATE (n2:Provincia { nombre: 'Segovia'})

CREATE (n3:Provincia { nombre: 'Granada'})

CREATE (n4:Provincia { nombre: 'Castellón de la Plana'})

CREATE (n5:Provincia { nombre: 'Barcelona'})

CREATE (n6:Provincia { nombre: 'Salamanca'})

CREATE (n7:Provincia { nombre: 'Alicante'})


-- Creació de CCAA

CREATE (n1:Ccaa { nombre: 'Madrid'})

CREATE (n2:Ccaa { nombre: 'Castilla y León'})

CREATE (n3:Ccaa { nombre: 'Andalucía'})

CREATE (n4:Ccaa { nombre: 'Valencia'})

CREATE (n5:Ccaa { nombre: 'Cataluña'})


-- Cream rel·lacions de províncies amb CCAA

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Madrid' AND b.nombre = 'Madrid' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Segovia' AND b.nombre = 'Castilla y León' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)
```

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Granada' AND b.nombre = 'Andalucía' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Castellón de la Plana' AND b.nombre = 'Valencia' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Barcelona' AND b.nombre = 'Cataluña' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Salamanca' AND b.nombre = 'Castilla y León' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)

MATCH (a:Provincia),(b:Ccaa) WHERE a.nombre = 'Alicante' AND b.nombre = 'Valencia' CREATE (a)-[r:ES_PROVINCIA_DE]->(b) RETURN type(r)

-- Cream rel·lacions de municipis amb províncies

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Torrejón de Ardoz' AND b.nombre = 'Madrid' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Segovia' AND b.nombre = 'Segovia' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Albolote' AND b.nombre = 'Granada' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Onda' AND b.nombre = 'Castellón de la Plana' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Tarrasa' AND b.nombre = 'Barcelona' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Alcalá de Henares' AND b.nombre = 'Madrid' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Peñaranda de Bracamonte' AND b.nombre = 'Salamanca' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Armilla' AND b.nombre = 'Granada' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Villajoyosa' AND b.nombre = 'Alicante' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

MATCH (a:Municipio),(b:Provincia) WHERE a.nombre = 'Badalona' AND b.nombre = 'Barcelona' CREATE (a)-[r:ES_MUNICIPIO_DE]->(b) RETURN type(r)

-- Cream rel·lacions de carrers

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Ramón y Cajal' AND b.nombre = 'Torrejón de Ardoz' CREATE (a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

```
MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Ramón y Cajal' AND b.nombre = 'Segovia'
CREATE (a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Ramón y Cajal' AND b.nombre = 'Albolote'
CREATE (a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Ramón y Cajal' AND b.nombre = 'Onda' CREATE
(a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Ramón y Cajal' AND b.nombre = 'Tarrasa' CREATE
(a)-[r:ES_CALLE_DE]->(b) RETURN type(r)


MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Cervantes' AND b.nombre = 'Alcalá de Henares'
CREATE (a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Cervantes' AND b.nombre = 'Peñaranda de
Bracamonte' CREATE (a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Cervantes' AND b.nombre = 'Armilla' CREATE (a)-
[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Cervantes' AND b.nombre = 'Villajoyosa' CREATE
(a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

MATCH (a:Calle),(b:Municipio) WHERE a.nombre = 'Cervantes' AND b.nombre = 'Badalona' CREATE
(a)-[r:ES_CALLE_DE]->(b) RETURN type(r)

-- Cream constraints per a assegurar que cap valor es repeteix

CREATE CONSTRAINT ON (calle:Calle) ASSERT calle.nombre IS UNIQUE

CREATE CONSTRAINT ON (municipio:Municipio) ASSERT municipio.nombre IS UNIQUE

CREATE CONSTRAINT ON (provincia:Provincia) ASSERT provincia.nombre IS UNIQUE

CREATE CONSTRAINT ON (ccaa:Ccaa) ASSERT ccaa.nombre IS UNIQUE

-- Graf creat:

MATCH p = () - [M:ES_CALLE_DE] -> () - [r:ES_MUNICIPIO_DE]-> () - [l:ES_PROVINCIA_DE] -> ()
RETURN p
```

```
$ MATCH p = () - [M:ES_CALLE_DE] -> () - [r:ES_MUNICIPIO_DE]-> () - [l:ES_PROVINCIA_DE] -> () RETURN p
```

*(23) Calle(2) Ccaa(5) Municipio(9) Provincia(7)

*(25) ES_CALLE_DE(9) ES_MUNICIPIO_DE(9) ES_PROVINCIA_DE(7)