

UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL RESISTENCIA



INGENIERÍA EN SISTEMAS DE INFORMACIÓN

BASES DE DATOS

TRABAJO PRÁCTICO INTEGRADOR

SEGUNDA ENTREGA

Profesor de Teoría: Ing. Andrés Fantín
Jefe de T. P.: Ing. Juan Carlos Fernández
Auxiliar: Leandro Antonio Romero

Grupo Nro. 14 - Integrantes:

- ACEVEDO, Fernando Enrique
- GETZEL, Martín Exequiel
- LUCAS, Dania
- OJEDA, María José
- VALDÉS, Manuel Enrique

Programabilidad

Triggers

1. Incorporar, a partir de una fundamentación específica en cada caso, al menos tres disparadores en tablas de la base de datos de manera que cada uno cumpla con alguno de los siguientes requerimientos:

1. Control de la integridad, coherencia y/o consistencia de los datos.

El siguiente trigger controla que si la **excentricidad** es igual a cero entonces, obligatoriamente el atributo **circular** debe ser verdadero.

```
DELIMITER //
CREATE TRIGGER verificar_excent_orbita BEFORE INSERT on Orbitas for each row
begin
    IF (new.excentricidad = 0) then
        SET new.circular= true;
    ELSE
        SET new.circular= false;
    end IF;
end//
DELIMITER ;
```

2. Actualización automática de datos (puede ser necesario agregar algún campo o tabla para cumplir el requerimiento).

Agregamos un atributo **cant_lanzamientos** que representa la cantidad de lanzamientos de cada **Agencia** entonces, antes de que se inserte un nuevo lanzamiento, actualizamos el atributo cantidad de esa Agencia.

```
DELIMITER //
CREATE TRIGGER actualizar_cantlanz AFTER INSERT ON Lanza
FOR EACH ROW
begin
    UPDATE Agencias
    SET cant_lanzamientos=cant_lanzamientos+1
    WHERE nombre=new.nombre_agencia;
END//
DELIMITER;
```

3. Algún tipo de auditoría (puede ser necesario agregar algún campo o tabla para cumplir el requerimiento).

Para el punto 3 de la primera parte, añadimos una relación de agregación junto con la entidad **costos**, la cual posee como atributos los tipos de costos, más un atributo que es la suma de estos, que se carga con el siguiente trigger, para facilitar posteriores consultas.

```

DELIMITER //
CREATE TRIGGER total_costos BEFORE INSERT ON costo
FOR EACH ROW
BEGIN
    SET new.costo_total = new.costo_nave + new.costo_lanza +
    new.costo_agencia;
END //
DELIMITER ;

```

Programas almacenados

1. Escribir un procedimiento almacenado que reciba como parámetro algún valor compatible con un dato en la base de datos a partir del cual se emita un informe para ese dato en particular. El informe debe requerir al menos una consulta avanzada de las vistas en cátedra y debe hacer uso de al menos una variable definida en el procedimiento.

En base a una **Nave**, calcula la cantidad de órbitas que recorrió, cuántos tripulantes navegaron en la misma y cuánta basura generó.

```

DELIMITER //
CREATE PROCEDURE informeNave (unaMatricula int)
BEGIN
    DECLARE cant_orbitas,cant_trip,Basura_prod INT;

    SET cant_orbitas = (
        SELECT COUNT(*) FROM esta GROUP BY matricula HAVING
matricula=unaMatricula);

    SET cant_trip = (
        SELECT COUNT(*) FROM Tiene WHERE matricula = unaMatricula);

    SET Basura_prod = (
        SELECT COUNT(*) FROM Basuras WHERE matricula = unaMatricula);

    SELECT cant_orbitas,cant_trip,basura; /*mostramos variables*/
END //
DELIMITER ;

```

2. Escribir una función almacenada que reciba como parámetro algún valor compatible con un dato en la base de datos a partir del cual se calcule un valor resumido para ese dato en particular. En la función se debe realizar al menos una consulta avanzada de las vistas en cátedra y debe hacerse uso de al menos una variable definida localmente.

Calcular cuántas naves diferentes ingresaron a una nueva órbita en un intervalo definido por una **fecha inicial** y una cantidad **n** de días.

```

DELIMITER //
CREATE FUNCTION contarOrbitas(fecha_entrada date, n int) RETURNS int

```

```

BEGIN
    DECLARE fecha_final date;
    DECLARE cantidad int;
    SET fecha_final = (SELECT date_add(fecha_entrada, interval n day));
    SET cantidad = (SELECT COUNT(DISTINCT matricula) FROM esta WHERE
fecha_ini BETWEEN fecha_entrada and fecha_final);
    RETURN cantidad;
END//

DELIMITER ;

```

Seguridad

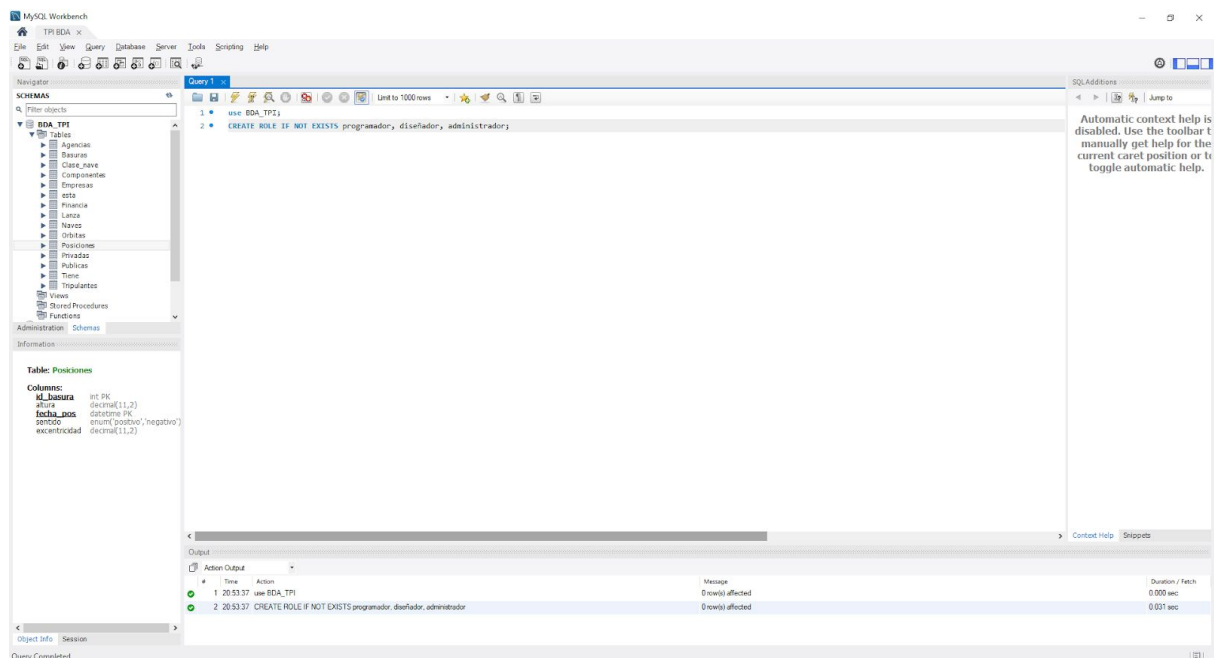
Este tema requiere la creación de roles y cuentas de usuario, con asignación de políticas de seguridad y uso de recursos, y la asignación de privilegios. Deberán debatir en el grupo y fundamentar cada una de las tareas realizadas.

Se pide:

1. Crear tres roles para que en la base de datos se pueda agrupar a los usuarios según los siguientes perfiles:

- 1. Programador**
- 2. Diseñador**
- 3. Administrador**

```
CREATE ROLE IF NOT EXISTS programador, diseñador, administrador;
```

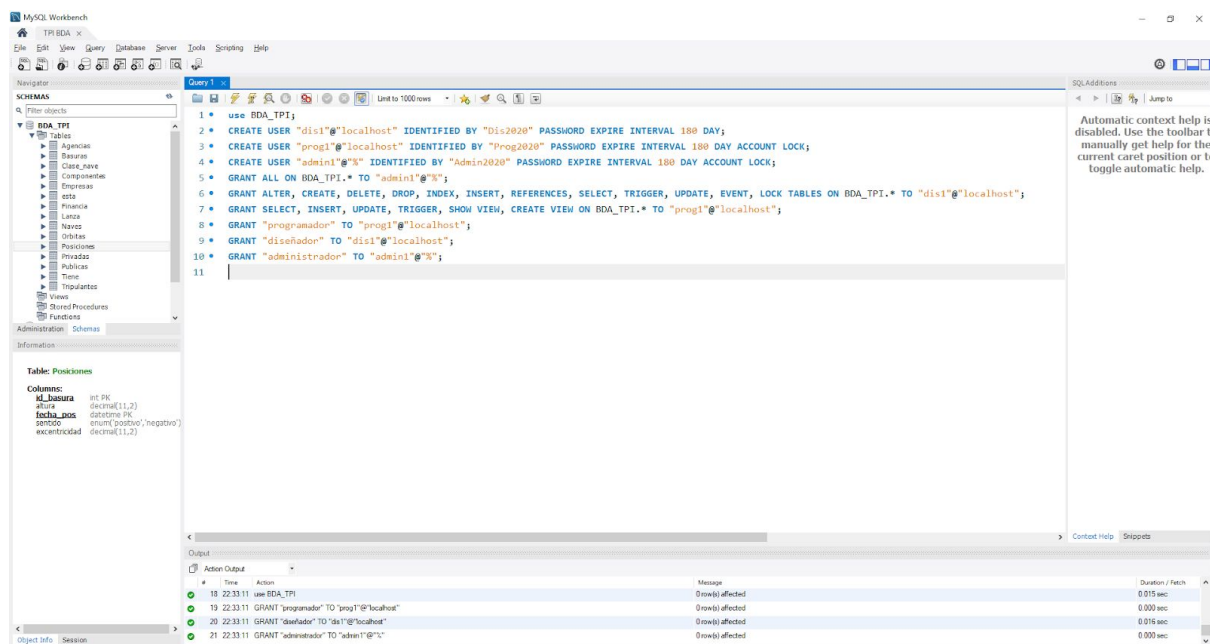


2. Debatir y fundamentar sobre los permisos necesarios para cada uno de los roles creados y realizar la asignación de los mismos.

Nos parece que el **administrador** debería tener privilegios del contexto “*server administration*”, es más podríamos entonces darle todos los privilegios . Mientras que el **diseñador** suponemos se encargaría del modelado de la base de datos pudiendo así modificar el esquema de la misma, por lo tanto necesitaría permisos del contexto “*Databases, tables, columns, indexes*”. Por último el **programador** sería capaz de realizar consultas, procedimientos almacenados, funciones, vistas.

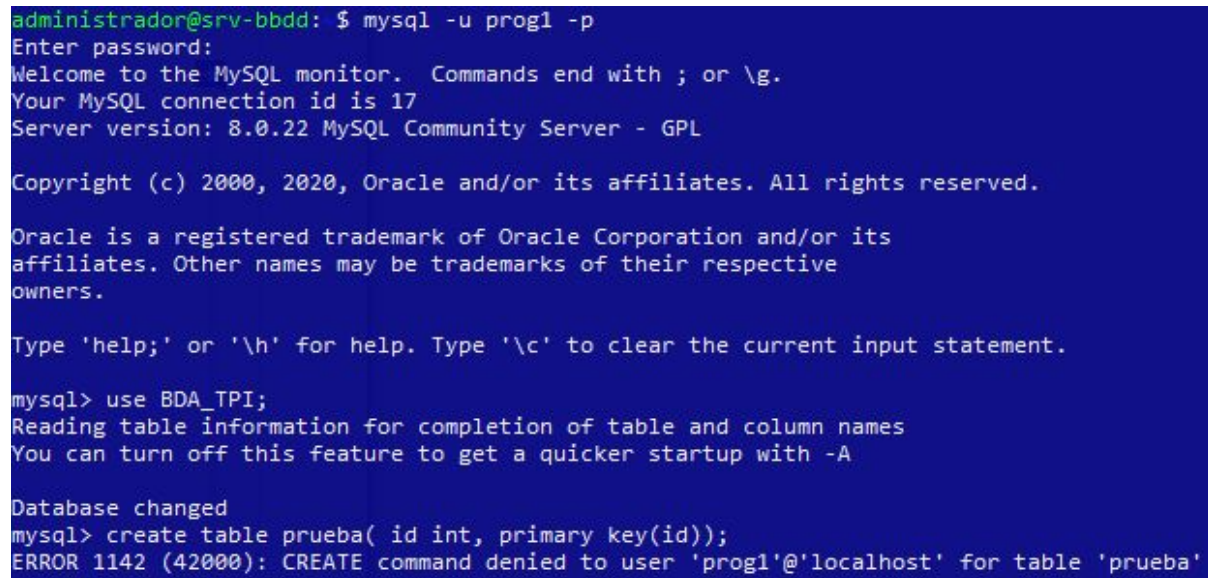
3. Crear al menos una cuenta para cada rol, estableciendo y fundamentando las políticas utilizadas para nombres de los usuarios, host desde los que se pueden conectar, políticas de contraseñas, utilización de recursos.

```
CREATE USER "dis1"@"localhost" IDENTIFIED BY "Dis2020" PASSWORD EXPIRE
INTERVAL 180 DAY ;
CREATE USER "prog1"@"localhost" IDENTIFIED BY "Prog2020" PASSWORD EXPIRE
INTERVAL 180 DAY;
CREATE USER "admin1"@"%" IDENTIFIED BY "Admin2020" PASSWORD EXPIRE INTERVAL
180 DAY;
GRANT ALL ON BDA_TPI.* TO "admin1"@"%";
GRANT ALTER, CREATE, DELETE, DROP, INDEX, INSERT, REFERENCES, SELECT,
TRIGGER, UPDATE, EVENT, LOCK TABLES ON BDA_TPI.* TO "dis1"@"localhost";
GRANT SELECT, INSERT, UPDATE, TRIGGER, SHOW VIEW, CREATE VIEW ON BDA_TPI.*
TO "prog1"@"localhost";
GRANT "programador" TO "prog1"@"localhost";
GRANT "diseñador" TO "dis1"@"localhost";
GRANT "administrador" TO "admin1"@"%";
```



4. Probar mediante distintas acciones que las cuentas de usuarios se comportan de acuerdo a lo planificado. Hacer capturas de pantalla de las pruebas realizadas.
ingresando con programador → crear una tabla

```
mysql -u prog1 -p
create table prueba(
id int,
primary key(id));
-- no debería permitir esta operación
```



```
administrador@srv-bbdd: $ mysql -u prog1 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

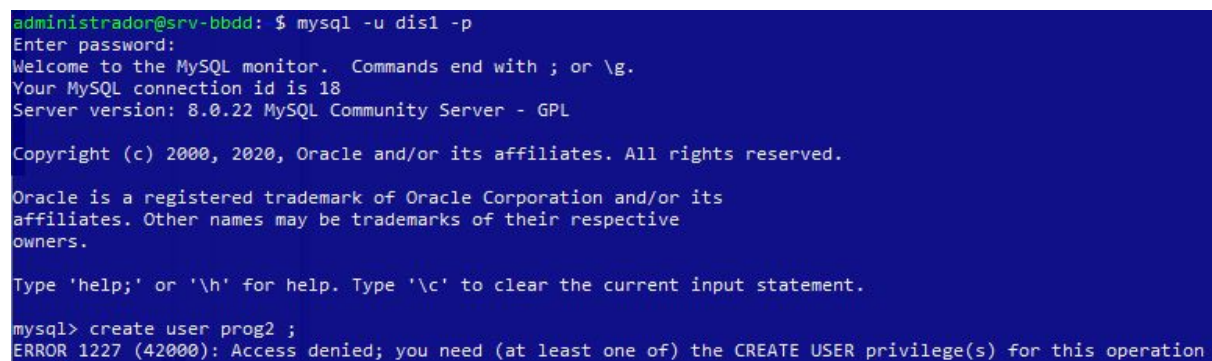
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use BDA_TPI;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table prueba( id int, primary key(id));
ERROR 1142 (42000): CREATE command denied to user 'prog1'@'localhost' for table 'prueba'
```

ingresando con diseñador → crear un usuario

```
mysql -u dis1 -p
create user 'prog2' ;
```



```
administrador@srv-bbdd: $ mysql -u dis1 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create user prog2 ;
ERROR 1227 (42000): Access denied; you need (at least one of) the CREATE USER privilege(s) for this operation
```

mientras que el administrador podría hacer cualquier acción

```
-mysql -u admin1 -p

create table prueba (id int,
primary key(id));
insert into prueba(id) values(1);
select * from prueba;
```



```

administrador@srv-bbdd: $ mysql -u admin1 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use BDA_TPI;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table prueba (id int,
-> primary key(id));
insert into prueba(id) values(1);
select * from prueba;
Query OK, 0 rows affected (0.01 sec)

mysql> insert into prueba(id) values(1);
Query OK, 1 row affected (0.01 sec)

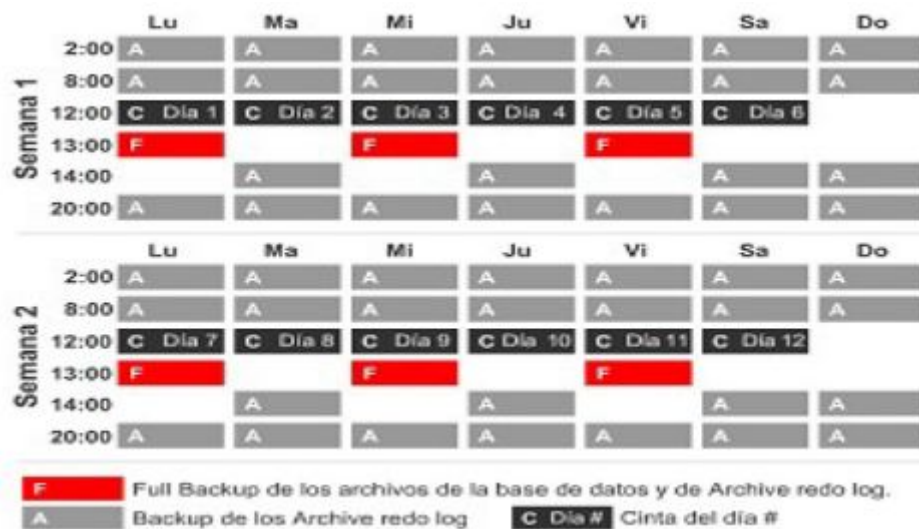
mysql> select * from prueba;
+----+
| id |
+----+
|  1 |
+----+
1 row in set (0.00 sec)

```

Respaldo y recuperación

Para un volumen de transacciones diarias (120.000 inserciones aprox.) y una disponibilidad requerida de la base de datos 24 horas al día, se debe implementar una política de backup soportada sobre un mecanismo de respaldo físico en caliente, con el fin de garantizar la menor pérdida de datos y su mayor disponibilidad, de acuerdo al tiempo que la aplicación debe estar en producción. Analizando la carga de la base de datos y las horas pico de consulta y transaccionalidad, se fijó como hora cero para backup, las 2:00 PM, ya que hacia las horas de la tarde la base de datos se encuentra con un nivel de carga bajo. Los intervalos de tiempo, en los cuales se hace el respaldo de los datos, son fijados de acuerdo al crecimiento en volumen de datos y el nivel de dinamismo que presenta la base de datos. Cuando se fija como política de respaldo el backup físico en línea, se corre el riesgo de provocar una caída en la base de datos, si no se garantiza espacio suficiente para el copiado de los archivos necesarios. Teniendo en cuenta lo anterior, se definió un esquema de respaldo con cintas diarias, realizando Full Backup de la base de datos los días lunes, miércoles y viernes, y backup de archivos de Log cada seis horas, eliminando estos archivos después de realizado el backup automáticamente, garantizando disponibilidad de espacio en disco para estos tipos de archivos. Se programó la toma del backup a través de tareas automáticas del sistema operativo, necesitando sólo la intervención del usuario, para el cambio de cinta y la validación de los backups. Se debe manejar un pool (conjunto) de doce (12) cintas, las cuales se deben intercambiar diariamente, rotándolas cada dos (2)

semanas, es decir que se contará con el backup de las dos últimas semanas. Por fines prácticos y para evitar inconvenientes en el proceso de intercambio de rotación de cintas, se fijó las doce del mediodía (12:00 PM), como la hora en que se debe realizar el intercambio de la cinta en el servidor, siguiendo la secuencia determinada. Este proceso se debe realizar todos los días de lunes a sábado y debe hacerlo la persona responsable de los backups. Por mantenimiento, confiabilidad y seguridad se recomienda cambiar el pool de cintas por unas nuevas cada seis meses. La Figura muestra el diagrama resultante de la esquematización de una estrategia de backup según el escenario planteado.



* Aclaración: Tomaremos el Backup de Archivos "redo log" como backups tipo incremental. Si bien no es lo mismo, lo hacemos para no complicar el escenario.

Tareas:

- a) **Investigue ¿Qué es un redo log?. ¿Su base de datos lo soporta o es algo exclusivo de un motor en particular? En caso afirmativo ¿cómo lo llama?**

El redo-log (registro de rehacer) es una estructura de datos basada en disco que se utiliza durante la recuperación de fallos para corregir los datos escritos por transacciones incompletas. Durante las operaciones normales, el redo-log codifica las solicitudes para cambiar los datos de la tabla que resultan de declaraciones SQL o llamadas a API de bajo nivel. Las modificaciones que no terminaron de actualizar los archivos de datos antes de un apagado inesperado se reproducen automáticamente durante la inicialización y antes de que se acepten las conexiones.

Cada instancia de una base de datos Oracle tiene un Redo Log asociado para proteger la base de datos en caso de falla de la instancia.

- b) **Deberá buscar y comparar soluciones de hardware para dar soporte a esta planificación. Al menos deberán ser tres opciones como mínimo con sus respectivos análisis de costos y de las tres proponer la mejor opción.**

Almacenamiento en la nube: 5usd en one drive

HDD 1TB : \$5380

SSD 1TB: \$10600

Cinta: NO ENCONTRAMOS PRECIOS

La mejor opción por lejos es la del almacenamiento en Cintas, debido a que nos brindan **seguridad** (permiten cifrar los datos y guardarlos de forma permanente en una cinta particular utilizando WORM (Write Once, Read Many) para evitar tanto el acceso no autorizado como la sobreescritura accidental de los datos), **duración** (puede durar 30 años si se conserva en un ambiente estable en cuanto a temperatura y humedad), relación **costo-beneficio** (tiene un coste muy bajo por gigabyte en comparación con sus competidores, incluyendo los discos y el almacenamiento en la nube. Además, las cintas individuales pueden contener varios terabytes de datos.), entre otras ventajas.

c) **Describa el procedimiento de restauración que utilizaría si la base de datos se dañase el primer Jueves de la 2da semana a las 11 de la mañana. ¿Tiene un plan "B" por si falla alguna restauración?**

d) **¿Cómo debería modificar la planificación si en vez de dos semanas se necesitarán conservar al menos 3 meses? Esto implicaría inversión adicional en equipamiento / software? Justifique.**

Si necesitamos conservar al menos 3 meses deberemos adaptar la planificación, realizando una inversión en equipamiento de almacenamiento, ya que el presupuesto inicial está enfocado a realizar copias de solamente los primeros 2 meses. Con respecto al software, no se deberán realizar cambios adicionales a lo ya previsto.

Datos semiestructurados

Dadas las modificaciones al escenario inicial, que implican el cumplimiento de nuevos requerimientos, deberán:

1. Analizar los nuevos requerimientos de registro de datos y obtener en base a la estructura del documento propuesta un esquema JSON a partir de alguna de las herramientas mostradas por la Cátedra.
2. Hacer una carga masiva de documentos utilizando alguna herramienta de generación de datos.
3. Resolver las consultas planteadas a continuación del escenario.

Escenario

La Agencia Nacional Aero Espacial decidió implementar una mejora en los datos de las Empresas privadas que financias los viajes espaciales, por lo que comenzó a recolectar información a través de formularios abiertos. Esa información se volcará en formato JSON a la tabla de Empresas de la base de datos desarrollada, de acuerdo a la siguiente estructura:

```
{"Principal Accionista" : {  
  "dni" : 40404040,  
  "nombre" : "juan paredéz",  
  "edad" : 24,  
  "ciudad" : "Madrid",  
  "nivel estudio" : "universitario",  
  "email" : "juanpa24@correos.com",
```

```

"redes sociales" : [
  {"instagram" : "juanpa24"},
  {"twitter" : "juanpa24"}],
"accionistas secundarios " :[
  {"accionista" :
    { "dni" : 40404040,
      "nombre" : "dfdfdsfds",
      "edad" : 87,
      "ciudad" : "xxxxxx",
      "nivel estudio" : "xxxxxxxxxxx",
      "email" : "xxxxx",
      "redes sociales" : [
        {"instagram" : "juanpa24"},
        {"twitter" : "juanpa24" }]
      } },
  {"accionista" :
    { "dni" : 40404040,
      "nombre" : "dfdfdsfds",
      "edad" : 87,
      "ciudad" : "xxxxxx",
      "nivel estudio" : "xxxxxxxxxxx",
      "email" : "xxxxx",
      "redes sociales" : [
        {"instagram" : "juanpa24"},
        {"twitter" : "juanpa24" }]
      } }],
"viajes" : [
  {"mundos" : "Martes", "tipo" : "Cientifico"}],
"aspiraciones" : [
  {"mundos" : "Jupiter", "tipo" : "Militar"},
  {"mundos" : "Martes", "tipo" : "Espía"}]
}
}

```

Agregamos una columna “*Accionista*” del tipo JSON que tendra el esquema anterior.

```
ALTER TABLE Empresas ADD COLUMN Accionista JSON;
```

Generamos un JsonSchema a partir del ejemplo brindado por la consigna

Consultas:

Mediante el soporte JSON del SGBD, realice las siguientes consultas:

1. Obtener el promedio de edad de los principales accionistas con nivel de estudio universitario.

```

select avg(accionista -> '$."Principal Accionista".edad')
from Empresas
where (accionista -> '$."Principal Accionista"."nivel estudio"')
LIKE '%universitario%'

```

2. Listar los nombres de los mundos con mayor cantidad de interesados. //mayor numero de interesados????

```
SELECT e.accionista -> '$.viajes.mundos'
FROM empresas as e,
      (select accionista -> '$.viajes.mundos' as mundo ,
        count(*) as cant from empresas group by (accionista ->
        '$.viajes.mundos') order by desc) as temp
WHERE (e.accionista -> '$.viajes.mundos') LIKE temp.mundo
AND temp.cant = (Select max(temp2.cant)
                 from (select accionista -> '$.viajes.mundos' as
                    mundo , count(*) as cant from empresas group by
                    (accionista -> '$.viajes.mundos') order by desc) as
                    temp2)
```

3. Listar los nombres, edades, ciudad de residencia y nombre de los mundos de interés del tipo de misión Científico.

```
SELECT accionista -> '$."Principal Accionista".nombre' , accionista
-> '$."Principal Accionista".edades', accionista -> '$."Principal
Accionista"."ciudad de residencia"',
accionista -> '$.viajes.mundos'
FROM Empresas
WHERE accionista -> '$.viajes.tipo' LIKE '%Cientifico%'
```

4. Listar los nombres, edades y ciudad de residencia de quienes visitan mundos en misiones del tipo Espía solamente (No Científico, No militar).

5. Listar los nombres y redes sociales de quienes hicieron alguna visita a Venus y les interesa conocer Saturno.