



**FACULTAD REGIONAL RESISTENCIA**

**INGENIERÍA EN SISTEMAS DE INFORMACIÓN**

**BASE DE DATOS APLICADAS**

**TRABAJO PRÁCTICO INTEGRADOR**

**Primera etapa “basura espacial”**

**Grupo Nro. 10**

**Integrantes:**

- Acevedo, Ariel A.
- Deppeler, Eric.
- Sosa Torres, Diego.

**AÑO 2021**

# Actividades a desarrollar en la Primer Etapa

## Administración del gestor de bases de datos (SGBD):

Realizar una instalación nueva del SGBD sobre una máquina virtual destinada específicamente para tal fin, diferente a la utilizada para los trabajos de clases (por ej. se puede comenzar con una mv desde cero, o a partir de una instantánea o una clonación previa al desarrollo de las guías de trabajos prácticos).

Responder las siguientes consignas:

Detallar los siguiente ítems, relacionados al SGBD implementado:

1. Indicar cantidad de memoria RAM mínima y recomendada.

El número de usuarios es uno de los factores que más afectan a la cantidad de memoria RAM necesaria. Si no disponemos de la suficiente memoria RAM en el servidor, el sistema empezaría a usar la memoria virtual en la unidad de almacenamiento, la cual es más lenta. Como cantidad mínima de RAM deberíamos tener 2GB y la cantidad recomendada es 4GB.

2. Ídem para el espacio en disco.

La base de datos será implementada en el motor MySQL, en este caso no existen requerimientos mínimos del sistema propuestos por el fabricante. Dado que no sabemos cuánto espacio de disco utilizaremos, reservaremos un espacio de unos 20 GB, que nos permitirá almacenar toda la información necesaria.

3. ¿Puede su SGBD instalarse en cualquier SO, sin limitación de arquitectura, del lenguaje o la localización del mismo?

El SGBD puede instalarse en cualquier SO, limitado a arquitecturas de x86\_32/64 bit o ARM 64.

4. Indicar cuáles son las alternativas para la instalación.

		8.0	5.7
Operating System	Architecture		
<b>Oracle Linux / Red Hat / CentOS</b>			
Oracle Linux 8 / Red Hat Enterprise Linux 8 / CentOS 8	x86_64, ARM 64	•	
Oracle Linux 7 / Red Hat Enterprise Linux 7 / CentOS 7	ARM 64	•	
Oracle Linux 7 / Red Hat Enterprise Linux 7 / CentOS 7	x86_64	•	•
Oracle Linux 6 / Red Hat Enterprise Linux 6 / CentOS 6	x86_32, x86_64	•	•
<b>Oracle Solaris</b>			
Solaris 11 (Update 4+)	SPARC_64	•	•
<b>Canonical</b>			
Ubuntu 21.04	x86_64	•	
Ubuntu 20.04 LTS	x86_64	•	
Ubuntu 18.04 LTS	x86_32, x86_64	•	•
<b>SUSE</b>			
SUSE Enterprise Linux 15 / OpenSUSE 15 (15.2)	x86_64	•	
SUSE Enterprise Linux 12 (12.5+)	x86_64	•	•
<b>Debian</b>			
Debian GNU/Linux 11	x86_64	•	
Debian GNU/Linux 10	x86_64	•	•

Microsoft Windows Server			
Microsoft Windows 2019 Server	x86_64	•	
Microsoft Windows 2016 Server	x86_64	•	•
Microsoft Windows 2012 Server R2	x86_64	•	•
Microsoft Windows			
Microsoft Windows 10	x86_64	•	•
Apple			
macOS 11	x86_64, ARM_64	•	
macOS 10.15	x86_64	•	
Various Linux			
Generic Linux (tar format)	x86_32, x86_64, glibc 2.12, libstdc++ 4.4	•	•
<a href="#">Yum Repo</a>		•	•
<a href="#">APT Repo</a>		•	•
<a href="#">SUSE Repo</a>		•	•

5. ¿Tiene soporte para discos sin formato (dispositivos en bruto o raw)? ¿da soporte parcial a esta característica? Enumerar las restricciones y ventajas expuestas por el fabricante si se da soporte a ésto.

En MySQL, se pueden usar particiones de dispositivos en bruto como ficheros de datos del espacio de tablas. Utilizando un dispositivo en bruto, se pueden llevar a cabo operaciones de E/S en Windows y algunas versiones de Unix sin que utilicen el búfer y sin la sobrecarga producida por el sistema de ficheros, lo cual incrementa el rendimiento.

6. Cuando la base de datos está vacía ¿Cuánto mide el espacio de tablas?

Para ver cuánto mide el espacio de tablas, creamos una nueva base de datos, y accedemos al directorio donde estaría ubicada.

Con `du -sh *` podemos ver cuál es el tamaño de los archivos y directorios, por lo que veremos cuánto pesa el directorio BDATPI que es el de la base de datos vacía:

```
administrador@srv-bbdd:~$ sudo su
[sudo] contraseña para administrador:
root@srv-bbdd:/home/administrador# cd /var/lib/mysql
root@srv-bbdd:/var/lib/mysql# du -sh *
du: no se puede acceder a '-': No existe el archivo o el directorio
du: no se puede acceder a 'sh': No existe el archivo o el directorio
4      auto.cnf
4      BDA2021
1780   BDA_Practica
4      BDATPI
4      binlog.000001
4      binlog.000002
4      binlog.000003
```

Podemos ver que el directorio BDATPI pesa 4KB.

7. Cuando la base de datos tiene datos equivalentes a 1MB ¿Cuánto mide el espacio de tablas? ¿Por qué?

NULL

8. Indicar la estructura de carpetas de instalación de los programas y ejecutables y archivos de configuración propios del SGBD.

Los archivos de configuración de MySQL se encuentran ubicados en `/etc/mysql/`.

Los ejecutables se ubican así:

- `mysqld`: `/usr/sbin/`
- `mysql`, `mysqldump`, `mysqlbinlog`: `/usr/bin`

9. Ídem anterior para los datos de tablas, y distintos registros de logs, ubicación predeterminada.

Si utilizamos un formato **InnoDB** los datos de las tablas se almacenarán por defecto en ficheros **ibdata** e **.ibd** de forma común para todas las bases de datos en el mismo directorio **/var/lib/mysql**.

Ficheros y directorios importantes:

**/var/lib/mysql/** Guarda las bases de datos del servidor. A cada base de datos corresponderá un directorio con el mismo nombre.

**/var/log/mysql/** Anotaciones y alertas del servidor. Por defecto, se crea un fichero de nombre **error.log** donde se registran los eventos que producen problemas en el servidor.

10. Describir brevemente la aplicación utilizada para administrar el SGBD (si está basada en Java, si es una aplicación web, nativa, etc.).

MySQL Workbench es una herramienta gráfica para trabajar con servidores y bases de datos MySQL. Soporta servidores MySQL de versiones 5.6 en adelante. Es compatible también con versiones más antiguas de servidores MySQL, excepto en ciertas situaciones (como mostrando la lista de procesos) debido al cambio del sistema de tablas. No soporta versiones 4.x de servidores MySQL.

Es una aplicación nativa, que se desarrolló para ser utilizada en Windows, Ubuntu, macOS, entre otros sistemas operativos.

11. Ingresar a la aplicación y describir las opciones que se observan. Investigar con la ayuda del motor: ¿Qué es una tabla?, ¿qué es un índice? ¿qué es una vista? ¿qué es un trigger?

- Una tabla es el lugar donde se almacenan datos, que tienen características similares y conforman la estructura de la base de datos. Los datos que pertenecen a un mismo elemento se organizan en columnas o campos.
- Los índices son un grupo de datos vinculado a una o varias columnas que almacena una relación entre el contenido y la fila en la que se encuentra. Con esto se agilizan las búsquedas en una tabla al evitar que MySQL tenga que recorrer toda la tabla para obtener los datos solicitados.
- Las vistas en MySQL (VIEWS) son tablas virtuales. Es decir, tablas que no guardan ningún dato propiamente dentro de ellas. Solo muestran los datos que están almacenados en otras tablas (que sí son reales).
- Los triggers o disparadores de MySQL son una serie de reglas predefinidas que están asociadas a una tabla. Estas reglas permiten la ejecución de una serie de instrucciones cuando se producen ciertos eventos como pueden ser la inserción de un nuevo registro, la actualización o el borrado de los datos de una tabla.

12. Enumerar las capacidades de su SGBD:

Las capacidades de MySQL son extremadamente amplias, ya que este servidor de bases de datos cuenta con un gran potencial de funcionamiento. El objetivo de este punto es el de mostrar el uso de MySQL para crear y usar una sencilla base de datos. MySQL

ofrece un programa interactivo que permite conectarnos a un servidor MySQL, ejecutar consultas y ver los resultados. Todas estas operaciones se pueden llevar a cabo tanto desde línea de comando en un shell, como desde un programa front-end gráfico que presente una interfaz gráfica de control.

- **Tamaño máximo de espacio de tablas.**

El tamaño máximo de espacio de tablas es de 64TB.

- **Cantidad máxima de tablas, índices, stored procedures, vistas.**

El número máximo de tablas soportado es 4 mil millones, el de índices se limita a 16 columnas y una longitud máxima de clave a 1000bytes.

Máximo de stored procedure

El número máximo de vistas

- **Cantidad máxima de columnas por tabla.**

El número máximo de columnas por tabla que se puede tener es de 4096.

- **Longitud máxima de fila.**

La representación interna de una fila no puede ocupar más de 65535 bytes.

- **Tamaños máximos de los tipos de datos que soporta.**

1. TINYINT con signo: 127, sin signo: 255.
2. SMALLINT con signo: 32767, sin signo: 65535
3. MEDIUMINT con signo: 8388607, sin signo: 16777215
4. INT con signo: 2147483647, sin signo: 4294967295
5. BIGINT con signo:  $2^{63}-1$ , sin signo:  $2^{64}-1$
6. DECIMAL: 65 dígitos y 30 decimales.
7. FLOAT: Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.
8. DOUBLE: Desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308.
9. DATE: '9999-12-31'.
10. DATETIME: 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos.
11. CHAR, VARCHAR, TinyText y TinyBlob: 255.
12. BLOB Y TEXT: 65535.
13. MEDIUM BLOB, MEDIUM TEXT: 16.777.215.
14. LONGBLOB, LONGTEXT: 4.294.967.295.
15. ENUM: 65535 valores distintos.
16. SET: 64 valores.

13. Explicar si el sistema operativo tiene incidencia sobre los tamaños máximos permitidos de espacios de tabla, catálogo u objetos grandes o sobre los nombres de los mismos.

Operating System	File-size Limit
Win32 w/ FAT/FAT32	2GB/4GB
Win32 w/ NTFS	2TB (possibly larger)
Linux 2.2-Intel 32-bit	2GB (LFS: 4GB)
Linux 2.4+	(using ext3 file system) 4TB
Solaris 9/10	16TB
MacOS X w/ HFS+	2TB
NetWare w/NSS file system	8TB

14. Enumerar los procesos del motor ejecutándose en el sistema operativo, sus nombres de imágenes, tamaño en memoria y funciones de cada uno. Indicar a su parecer los dos más importantes.
15. Indicar cuántos nodos (en la estructura de árbol en la que se encuentran los objetos de la instancia manejada) se muestran en el programa administrador del SGBD, y la función de cada uno. No incluya nodos relacionados con los datos de bases de datos de usuario, sólo los nodos que corresponden a funcionalidad predeterminada.
16. ¿Cuántas bases de datos de sistema tiene su SGBD? Enumerar y explicar sucintamente cuál es la función de cada una.

**Information schema:** es la base de datos de información que almacena información acerca de todas las otras bases de datos que mantiene el servidor MySQL. Dentro de ellas se encuentran vistas de las cuales no hay ningún fichero asociado a ellas.

**MySql:** se corresponde con el esquema del sistema mysql, el cual contiene información requerida por el servidor cada vez que se lo corre, esta base de datos contiene las tablas de diccionario de datos y las del sistema.

**Performance schema:** se almacenan las tablas de esquemas de rendimiento, donde puede consultar estas tablas para ver información en tiempo real sobre las características de rendimiento del servidor y las aplicaciones que está ejecutando.

**Sys:** conjunto de objetos que ayuda a interpretar datos recopilados en el esquema de rendimiento. Incluyen viste de resumen los datos del esquema de rendimiento, procedimientos almacenados que realizan operaciones como la configuración del esquema de rendimiento, y funciones almacenadas que consultan la configuración del esquema de rendimiento.

17. Indicar cuántos tipos de objetos distintos puede contener una base de datos típica en su gestor SGBD (tabla, índice, etc.).

La base de datos puede contener 9 tipos distintos de objetos

- 1) Event.
  - 2) Function.
  - 3) Index.
  - 4) Procedure.
  - 5) Function.
  - 6) Spatial Reference System.
  - 7) Table.
  - 8) Trigger.
  - 9) View.
18. Realice una síntesis sobre principales diferencias, ventajas y desventajas frente a otros SGBDs (elijá al menos dos, puede tomar como referencia <https://db-engines.com/en/>).

### **MYSQL VS MONGODB**

Tabla comparativa, en esta tabla se destacan algunas de las diferencias entre ambos

Característica	MongoDB	MySQL
Cloud, SaaS, Web	si	si
Desarrolladores	MongoDB Inc.	Oracle Corporation
SO	Multiplataforma	Multiplataforma
Lenguaje query	Javascript	SQL
Mapa reducido	si	no
Conversion de DB	si	no
Análisis de performance	si	no
Virtualización	si	no
Modelo de integridad	BASE	ACID
Atomicidad	condicional	si
Aislamiento	no	si
Transacciones	no	si
Integridad referencial	no	si
CAP	CP	CA
Escalabilidad horizontal	si	condicional
Modo de replicación	Maestro-Esclavo	Maestro - Maestro/Esclavo

### **MONGODB:**

MongoDB es una base de datos documental, lo que significa que almacena datos en forma de documentos tipo JSON. Utiliza un **lenguaje de consultas no estructurado** por lo que realizamos las consultas especificando el nombre del documento con las propiedades que queremos filtrar. Este tipo de consultas permite una amplia variedad de operadores.

Como vemos en la tabla comparativa MongoDB en el teorema CAP se inclina a **CP (Consistencia y Tolerancia a particiones)** esto significa que todos los clientes acceden a una vista consistente de la base de datos. Lo cual implica que los usuarios de un nodo

deben esperar a que los otros nodos se sincronicen para poder ser visibles y editables, en este caso la disponibilidad queda en segundo plano frente a la consistencia. En el ámbito de la seguridad MongoDB utiliza un **control de acceso basado en roles con privilegios flexibles**, sus características de seguridad incluyen **autenticación, auditoría y autorización**. También **permite el uso de TLS/SSL** con el propósito de encriptar los datos y que solo sean accesibles para el cliente.

#### **Ventajas:**

- Validación de documentos.
- Motores de almacenamiento integrado.
- Menor tiempo de recuperación ante fallas.

#### **Desventajas:**

- No es una solución adecuada para aplicaciones con transacciones complejas.
- No tiene un reemplazo para las soluciones de herencia.
- Aún es una tecnología joven.

#### **MYSQL:**

MySQL Database Service es un servicio de base de datos totalmente administrado que permite a las organizaciones implementar aplicaciones nativas de la nube utilizando la base de datos de código abierto más popular del mundo.

En MySQL **las consultas se manejan con un lenguaje estructurado como lo es SQL**, por lo general suele ser bastante simple una vez le agarramos la mano, el mismo implica 2 partes un lenguaje de definición de datos (DDL) y un lenguaje de manipulación de datos (DML).

En la tabla comparativa vemos que MYSQL bajo el teorema CAP se inclina por **CA (Consistencia y disponibilidad)**, esto significa que **la información será consistente entre todos los nodos mientras los mismos estén disponibles**. Esto nos permite leer/escribir desde cualquier nodo y estar seguros de que los datos serán consistentes. Si se realiza una partición entre nodos los datos no estarán sincronizados y no se resolverá hasta que se resuelva la partición.

MySQL utiliza un **modelo de seguridad basado en privilegios**, a cada usuario se le asigna privilegios sobre la base de datos de tipo CREATE, SELECT, INSERT, UPDATE entre otros. MySQL también utiliza **encriptación para la conexión entre el server y el cliente del tipo SSL**.

#### **Ventajas:**

- Soporta transacciones atómicas.
- Soporta el uso de JOIN.
- Seguridad basada en privilegios con uso de contraseña.
- Es una tecnología madura.

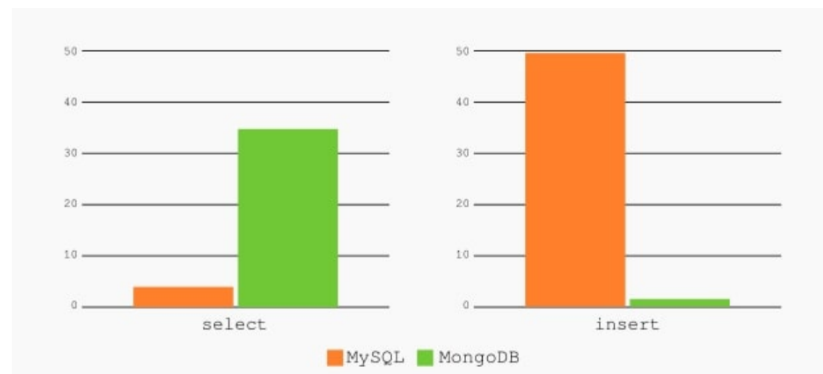


### Desventajas:

- Difícil de escalar.
- Problemas de estabilidad.
- No es un desarrollo impulsado por la comunidad.

### Performance y Velocidad

Los datos de la gráfica fueron tomados en base a 1.000.000 registros con MySQL 5.7.9 y MongoDB 3.2.0 utilizando las configuraciones por defecto en un servidor Ubuntu con 8 CPUs virtuales y 32 GB de ram en entornos separados.



MongoDB es más rápido que MySQL gracias a su capacidad para manejar grandes cantidades de datos no estructurados, permitiendo realizar consultas de manera sensible al workload (carga de trabajo). Mientras que MySQL suele ser más lento al momento de manejar grandes bases de datos.

Realizar las siguientes tareas de administración:

1. Agregar los usuarios "ADMIN" y "OPERADOR".

```
CREATE USER 'ADMIN'@'%' IDENTIFIED BY 'admintpi';  
CREATE USER 'OPERADOR'@'%' IDENTIFIED BY 'operadortpi';
```

```
mysql> CREATE USER 'ADMIN'@'%' IDENTIFIED BY 'admintpi';  
Query OK, 0 rows affected (0,00 sec)  
  
mysql> CREATE USER 'OPERADOR'@'%' IDENTIFIED BY 'operadortpi';  
Query OK, 0 rows affected (0,01 sec)
```

2. Permitir que ADMIN sea administrador de sistema. A partir de éste momento todas las tareas administrativas deberán realizarse con esta cuenta (no con root).

```
GRANT ALL ON *.* TO 'ADMIN'@'%' WITH GRANT OPTION;
```

```
mysql> GRANT ALL ON *.* TO 'ADMIN'@'%' WITH GRANT OPTION;  
Query OK, 0 rows affected (0,00 sec)
```

3. Crear una base de datos para desarrollar el trabajo práctico integrador. Como único

requisito debe llamarse BDA-TPI. Escoger el resto de los parámetros usando su criterio, documentar cada paso con una copia de pantalla del asistente de creación o el comando utilizado.

```
CREATE DATABASE IF NOT EXISTS BDA-TPI DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_spanish_ci;
```

```
mysql> CREATE DATABASE IF NOT EXISTS BDA_TPI DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_spanish_ci;  
Query OK, 1 row affected (0,00 sec)
```

4. Asignar los privilegios necesarios al usuario OPERADOR para que tenga acceso a todas las tareas de administración sobre la base de datos *BDA-TPI* excepto la gestión de usuarios y permisos.

```
GRANT ALL PRIVILEGES ON BDA_TPI.* TO 'OPERADOR'@'%';  
REVOKE GRANT OPTION ON BDA_TPI.* FROM 'OPERADOR'@'%';  
REVOKE CREATE USER ON *.* FROM 'OPERADOR'@'%';
```

```
mysql> GRANT ALL PRIVILEGES ON BDA_TPI.* TO 'OPERADOR'@'%';  
Query OK, 0 rows affected (0,00 sec)  
  
mysql> REVOKE GRANT OPTION ON BDA_TPI.* FROM 'OPERADOR'@'%';  
Query OK, 0 rows affected (0,00 sec)  
  
mysql> REVOKE CREATE USER ON *.* FROM 'OPERADOR'@'%';  
Query OK, 0 rows affected (0,01 sec)
```

#### Análisis del escenario y modelado:

El escenario sobre el que cada grupo deberá trabajar está basado en una organización cuya descripción se presenta como ANEXO II de esta guía. A efectos de simplificar y unificar criterios respecto al diseño de la base de datos resultante, se presenta como ANEXO III una propuesta de solución para el modelado con un diagrama entidad-relación (DER). Se adjunta también un archivo con la "Notación usada en los DERs de las soluciones propuestas". Para el desarrollo del trabajo práctico deberán ejecutarse las siguientes consignas:

- Leer el escenario en general en forma individual.
- Analizar el escenario en grupo, y el DER resultante. Deberán registrarse los ítems que no resulten claros o que se considere que no fueron capturados por el DER con respecto a los requerimientos planteados en el escenario y que no se hayan contemplado en supuestos adicionales.

*Esquema relacional:* Tomando como entrada el DER y las consideraciones hechas al mismo, se deberá realizar el esquema de la base de datos en el modelo relacional. Este esquema puede ser gráfico o bien un esquema relacional textual.

- Convertir las entidades y relaciones del DER a esquemas de relación, según la metodología practicada en clases.
- Tener en cuenta:
  - Nombres dados a las tablas/relaciones.
  - Dominio de los atributos.
  - Atributos de claves primarias y claves foráneas.
  - Los distintos tipos de restricciones (dominio, integridad, participación, etc.).
- El resultado de esta actividad es un archivo que será parte de la documentación del TPI.

*Esquema físico:* A partir de la actividad anterior, obtener el script sql para implementar las tablas y relaciones resultantes del modelo relacional en el SGBD instalado.

- El resultado de esta etapa debe ser un script sql que deberá ejecutarse en el SGBD.

#### *Recomendaciones sobre los esquemas*

- Podrá utilizarse en estas instancias algún software o herramienta CASE para facilitar el modelado (p.e. MySQL Workbench o ER-Studio).
- En el esquema físico tener en cuenta los nombres de tablas y campos, por ejemplo, el domicilio de un cliente no conviene llamarlo “Domicilio Cliente” sino “DomCli” (o similar). También prestar especial atención a los tipos de datos y sus características, justificando adecuadamente cada elección.

*Base de Datos:* Una vez creada la base de datos con todos los objetos planteados en el esquema físico, realizar las siguientes consignas:

- Cargar datos en las tablas para que todas tengan al menos una fila. Al menos dos tablas deben tener más de 10 filas.
- Utilizando alguna herramienta para generación de datos, importar masivamente filas a las tablas indicadas. Tener en cuenta que se intenta trabajar con volúmenes importantes similares a un sistema real, por lo que deberá asegurarse que al menos una tabla cuente con más de 100000 registros.
- Una vez cargadas las tablas con datos, realizar, a criterio del grupo, 5 (cinco) consultas sql distintas para borrado de filas, y otras 5 para modificación de datos. Las consultas pueden ser ejecutadas sobre una misma tabla o distintas.

*Consultas SQL:* Deberán resolverse los requerimientos sobre el escenario, entregados como ANEXO IV, mediante consultas efectuadas en el lenguaje SQL.

- Previo a la ejecución de las consultas deberá cargarse la base de datos con cierta cantidad de datos que aseguren que éstas tengan un resultado visible.

#### Esquema Relacional:

El esquema relacional se desarrolló en código mediante un script en SQL que se adjunta en la entrega cómo [full\\_BDA\\_TPI-GRUPO10.sql](#)

DER: [DER TPI GRUPO 10](#)

## 1. Consultas

- a. Nombre de las naves que produjeron al menos 10 basuras distintas.

```
WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basura AS B GROUP BY B.matricula )
SELECT C.nombre
FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo
INNER JOIN Temp ON Temp.matricula = N.matricula
WHERE Temp.cant > 9;
```

```
mysql> WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basura AS B GROUP BY B.matricula )
-> SELECT C.nombre
-> FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo INNER JOIN Temp ON Temp.matricula = N.matricula
-> WHERE Temp.cant > 9;
+-----+
| nombre |
+-----+
| Vehículos lanzadera |
| Vehículos lanzadera |
| Vehículos lanzadera |
| Vehículos lanzadera |
| Naves no tripuladas o roboticas |
| Naves no tripuladas o roboticas |
| Naves no tripuladas o roboticas |
| Naves espaciales tripuladas |
| Naves espaciales tripuladas |
| Naves espaciales tripuladas |
+-----+
10 rows in set (0,03 sec)
```

- b. Listar los pares (basura 1,basura 2) tales que la basura 1 fue producida por una nave que también produjo basura 2.

```
SELECT DISTINCT B1.id_basura as 'Basura 1' , B2.id_basura as 'Basura 2'
FROM Basura AS B1 , Basura AS B2
WHERE B2.id_basura <> B1.id_basura AND B1.matricula = B2.matricula;
```

```
mysql> SELECT DISTINCT B1.id_basura as 'Basura 1' , B2.id_basura as 'Basura 2'
-> FROM Basura AS B1 , Basura AS B2
-> WHERE B2.id_basura <> B1.id_basura AND B1.matricula = B2.matricula
-> LIMIT 30;
+-----+-----+
| Basura 1 | Basura 2 |
+-----+-----+
| 26 | 30 |
| 26 | 57 |
| 26 | 72 |
| 26 | 109 |
| 26 | 116 |
| 26 | 119 |
| 26 | 130 |
```

- c. Listar los nombres de las agencias que no lanzaron ninguna nave que haya estado en órbita.

```
SELECT A2.nombre
FROM Agencia AS A2
WHERE A2.nombre NOT IN (SELECT A.nombre
FROM Agencia AS A INNER JOIN Lanza AS L ON L.nombre_agencia = A.nombre
INNER JOIN Nave AS N ON L.matricula = N.matricula, Orbita AS O
WHERE O.sentido = L.sentido AND O.altura = L.altura AND L.excentricidad =
O.excentricidad );
```

```
mysql> SELECT A2.nombre
FROM      -> FROM Agencia AS A2
      -> WHERE A2.nombre NOT IN (SELECT A.nombre
      -> FROM Agencia AS A INNER JOIN Lanza AS L ON L.nombre_agencia = A.nombre INNER JO
IN Nave AS N ON L.matricula = N.matricula, Orbita AS O
      -> WHERE O.sentido = L.sentido AND O.altura = L.altura AND L.excentricidad = O.exc
entricidad );
+-----+
| nombre |
+-----+
| Amazon.com |
| Apple Inc. |
| Areon Impex |
| Boeing |
| Demaco |
| DynCorp |
| Erickson |
| Global Print |
| Metro Cash&Carry |
| Telekom |
+-----+
10 rows in set (0,00 sec)
```

- d. Listar las órbitas en las cuales estuvieron todas las naves.

```
SELECT E.excentricidad, E.altura, E.sentido
FROM esta AS E
WHERE NOT EXISTS (SELECT N.matricula
FROM Nave AS N
WHERE NOT EXISTS (SELECT E2.excentricidad, E2.altura,
E2.sentido
FROM esta AS E2
WHERE E2.excentricidad = E.excentricidad AND E.altura= E2.altura AND
E.sentido = E2.sentido AND E2.matricula = N.matricula));
```

```
mysql> SELECT E.excentricidad, E.altura, E.sentido
-> FROM esta AS E
-> WHERE NOT EXISTS (SELECT N.matricula
-> FROM Nave AS N
-> WHERE NOT EXISTS (SELECT E2.excentricidad, E2.altura, E2.sentido
-> FROM esta AS E2
-> WHERE E2.excentricidad = E.excentricidad AND E.altura= E2.altura AND E.sentido
= E2.sentido AND E2.matricula = N.matricula));
Empty set (0,00 sec)
```

- e. Liste el nombre de todas las empresas públicas y su agencia que supervisan al menos dos empresas privadas distintas.

```
WITH supervisa AS (SELECT nombre_publica, COUNT(*) AS cant
FROM Privada
GROUP BY nombre_publica)
SELECT P.nombre_e, P.nombre
FROM Publica AS P INNER JOIN supervisa AS S ON S.nombre_publica =
P.nombre_e
WHERE S.cant > 1;
```

```
mysql> WITH supervisa AS (SELECT nombre_publica, COUNT(*) AS cant
-> FROM Privada
-> GROUP BY nombre_publica)
-> SELECT P.nombre_e, P.nombre
-> FROM Publica AS P INNER JOIN supervisa AS S ON S.nombre_publica = P.nombre_e
-> WHERE S.cant > 1;
+-----+-----+
| nombre_e | nombre          |
+-----+-----+
| SpaceX   | Metro Cash&Carry |
+-----+-----+
1 row in set (0,00 sec)
```

## 2. Consultas con modificación de escenario

Modifique el ER de tal manera que, además del tipo de componente por clase de nave se registre los componentes (nro, nombre y precio) de la misma reflejando que una parte puede ser a su vez parte de otra y que una parte puede ser también parte de varias al mismo tiempo. Inserte datos para reflejar la situación de la figura y luego resuelva.

### MODIFICACIONES:

- *Agregar atributos a tabla componentes:*
  - *nro integer.*
  - *precio decimal(9,2).*
  - *nombre varchar(45).*

```
ALTER TABLE tipo_Componente
ADD nro integer DEFAULT NULL,
ADD precio decimal(9,2) DEFAULT NULL,
ADD nombre varchar(45) DEFAULT NULL;
create index componente_index ON Componentes(nro);
```

```
CREATE TABLE compuesto_por(
componente_contenedor int,
componente_contenido int,
CONSTRAINT pk_compuesto_por PRIMARY KEY(componente_contenedor,
componente_contenido));
create index componente_contenedor_index ON
```

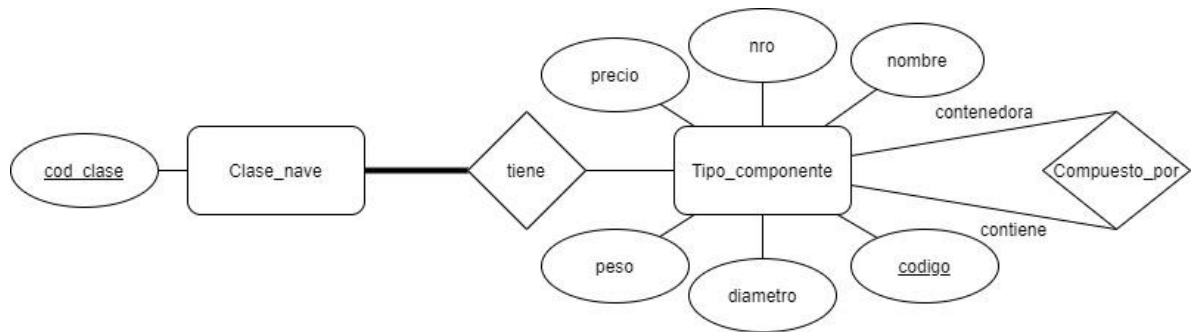
```

compuesto_por(componente_contenedor);
create index componente_contenido_index ON
compuesto_por(componente_contenido);

ALTER TABLE compuesto_por
add CONSTRAINT fk_componente_contenedor FOREIGN
KEY(componente_contenedor) REFERENCES Componente(nro),
add CONSTRAINT fk_componente_contenido FOREIGN KEY(componente_contenido)
REFERENCES Componente(nro);

```

Parte del DER modificado:



Modificaciones: [TPI\\_Punto2.sql](#)

COMPONENTE 2 → COMPUESTO POR → (COMPONENTE X (COMPONENTE 1 ))

#1 esta formado por 2,3,4 y 16

#5 esta formado por 6 y 7

#6 esta formado por 8

#7 esta formado por 15

#8 esta formado por 14

#9 esta formado por 10 y 11

#12 esta formado por 11 y 13

#13 esta formado por 11

```

UPDATE Tipo_componente SET nro = 1, nombre = 'Conducto Dual Interno'
WHERE codigo = 1;

```

```

UPDATE Tipo_componente SET nro = 2, nombre = 'Computadora avionica' WHERE
codigo = 2;

```

```

UPDATE Tipo_componente SET nro = 3, nombre = 'Actuador Catalitico' WHERE
codigo = 3;

```

```

UPDATE Tipo_componente SET nro = 4, nombre = 'Conmutador de Flujo' WHERE
codigo = 4;

```

```

UPDATE Tipo_componente SET nro = 5, nombre = 'Inyector Subdual
Plasmatico' WHERE codigo = 5;

```

```

UPDATE Tipo_componente SET nro = 6, nombre = 'Tornillo N45' WHERE codigo
= 6;

```

```

UPDATE Tipo_componente SET nro = 7, nombre = 'Reciclador Catalitico
Interno' WHERE codigo = 7;

```

```

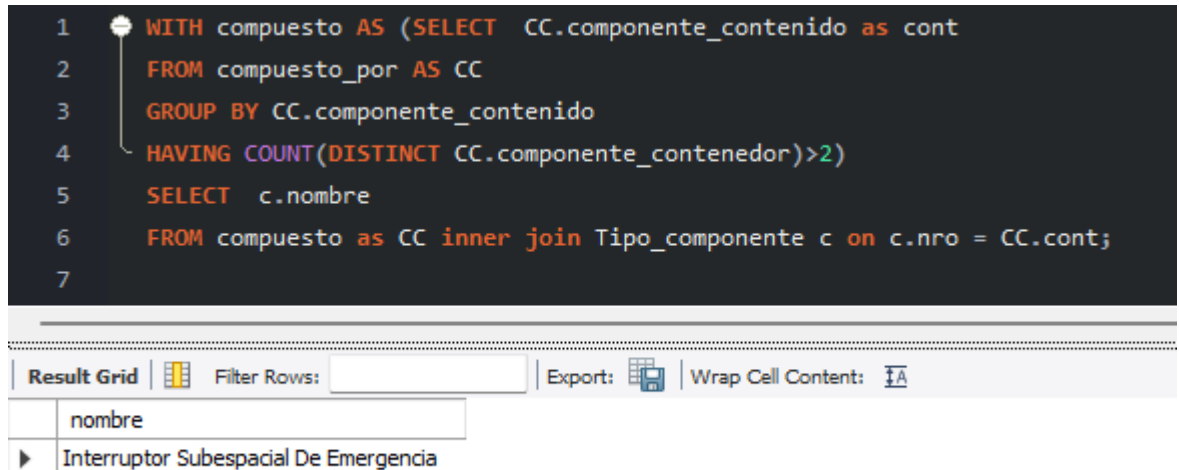
UPDATE Tipo_componente SET nro = 8, nombre = 'Giroscopio Coaxial
Automatico' WHERE codigo = 8;
UPDATE Tipo_componente SET nro = 9, nombre = 'Junta Holografica
Principal' WHERE codigo = 9;
UPDATE Tipo_componente SET nro = 10, nombre = 'Inyector Holografico'
WHERE codigo = 10;
UPDATE Tipo_componente SET nro = 11, nombre = 'Interruptor Subespacial De
Emergencia' WHERE codigo = 11;
UPDATE Tipo_componente SET nro = 12, nombre = 'Reciclador Coaxial' WHERE
codigo = 12;
UPDATE Tipo_componente SET nro = 13, nombre = 'Giroscopio Dual
Automatico' WHERE codigo = 13;
UPDATE Tipo_componente SET nro = 14, nombre = 'Conmutador Holografico
Plasmatico' WHERE codigo = 14;
UPDATE Tipo_componente SET nro = 15, nombre = 'Inyector Trifasico
Rotatorio' WHERE codigo = 15;
UPDATE Tipo_componente SET nro = 16, nombre = 'Sanguchito de Miga
Aeroespacial' WHERE codigo = 16;
insert into compuesto_por(componente_contenedor, componente_contenido)
values(1,2);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(1,3);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(1,4);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(1,16);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(5,6);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(5,7);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(6,8);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(7,15);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(8,14);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(9,10);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(9,11);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(12,11);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(12,13);
insert into compuesto_por(componente_contenedor, componente_contenido)
values(13,11);

```



- a. Nombre de los componentes que forman parte de al menos tres componentes distintos.

```
WITH compuesto AS (SELECT CC.componente_contenido as cont
FROM compuesto_por AS CC
GROUP BY CC.componente_contenido
HAVING COUNT(DISTINCT CC.componente_contenedor)>2)
SELECT c.nombre
FROM compuesto as CC inner join Tipo_componente c on c.nro = CC.cont;
```



The screenshot shows a SQL query editor with a dark background. The query is as follows:

```
1 WITH compuesto AS (SELECT CC.componente_contenido as cont
2 FROM compuesto_por AS CC
3 GROUP BY CC.componente_contenido
4 HAVING COUNT(DISTINCT CC.componente_contenedor)>2)
5 SELECT c.nombre
6 FROM compuesto as CC inner join Tipo_componente c on c.nro = CC.cont;
7
```

Below the query editor, there is a toolbar with buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. The 'Result Grid' button is active. Below the toolbar, a table is displayed with the following data:

nombre
Interruptor Subespacial De Emergencia

- b. Listar los pares (componente1,componente2), donde componente1 y componente2 son tales que componente1 forma parte de otro que a su vez depende de componente2.

```
SELECT C1.nombre, C1.nro , C2.nombre, C2.nro
FROM Tipo_componente AS C1, Tipo_componente AS C2
WHERE C1.nombre <> C2.nombre AND C1.nro IN (SELECT
cx.componente_contenido
FROM compuesto_por as cx
where cx.componente_contenedor IN
(SELECT cc.componente_contenido
FROM compuesto_por as cc
WHERE cc.componente_contenedor = C2.nro));
```

```
mysql> SELECT C1.nombre, C1.nro , C2.nombre, C2.nro
-> FROM Tipo_componente AS C1, Tipo_componente AS C2
-> WHERE C1.nombre <> C2.nombre AND C1.nro IN (SELECT cx.componente_contenido
-> FROM compuesto_por as cx
-> where cx.componente_contenedor IN
-> (SELECT cc.componente_contenido
-> FROM compuesto_por as cc
-> WHERE cc.componente_contenedor = C2.nro));
```

nombre	nro	nombre	nro
Inyector Trifasico Rotatorio	15	Inyector Subdual Plasmatico	5
Giroscopio Coaxial Automatico	8	Inyector Subdual Plasmatico	5
Conmutador Holografico Plasmatico	14	Tornillo N45	6
Interruptor Subespacial De Emergencia	11	Reciclador Coaxial	12

```
4 rows in set (0,01 sec)
```

- c. Listar los artículos que forman parte de todas las partes (en forma directa). Luego si la consulta es vacía inserte los registros que sean necesarios para que la respuesta no sea vacía.

```
DELIMITER //
CREATE PROCEDURE todas_las_partes()
BEGIN
IF (SELECT CC.componente_contenido
FROM compuesto_por AS CC
WHERE NOT EXISTS (SELECT C.nro
FROM Tipo_componente AS C
WHERE C.nro < 17 AND NOT EXISTS
(SELECT CC2.componente_contenido
FROM compuesto_por AS CC2
WHERE CC.componente_contenido = CC2.componente_contenido AND
CC2.componente_contenido = C.nro))) IS NULL THEN
INSERT INTO compuesto_por(componente_contenedor,
componente_contenido) SELECT nro, 1 FROM Tipo_componente WHERE nro < 17
AND nro NOT IN (SELECT componente_contenedor FROM compuesto_por WHERE
componente_contenido = 1);
else
SELECT CC.componente_contenido
FROM compuesto_por AS CC
WHERE NOT EXISTS (
SELECT C.nro
FROM Tipo_componente AS C
WHERE C.nro <17 AND NOT EXISTS (
SELECT CC2.componente_contenido
FROM compuesto_por AS CC2
WHERE CC.componente_contenido =
CC2.componente_contenido AND CC2.componente_contenido =
C.nro));
```

```

        END IF;
        END //
DELIMITER ;

```

```
CALL todas_las_partes();
```

```

mysql> DELIMITER //
mysql> CREATE PROCEDURE todas_las_partes()
-> BEGIN
-> IF (SELECT CC.componente_contenido
-> FROM compuesto_por AS CC
-> WHERE NOT EXISTS (SELECT C.nro
-> FROM Tipo_componente AS C
-> WHERE C.nro < 17 AND NOT EXISTS
-> (SELECT CC2.componente_contenido
-> FROM compuesto_por AS CC2
-> WHERE CC.componente_contenido = CC2.componente_contenido AND CC2.componente_con
tenido = C.nro))) IS NULL THEN
-> INSERT INTO compuesto_por(componente_contenedor, componente_contenido)SELECT nr
o, 1 FROM Componente WHERE nro < 17
-> AND nro NOT IN (SELECT componente_contenedor FROM compuesto_por WHERE component
e_contenido = 1);
-> else
-> SELECT CC.componente_contenido
-> FROM compuesto_por AS CC
-> WHERE NOT EXISTS (SELECT C.nro
-> FROM Tipo_componente AS C
-> WHERE C.nro <17 AND NOT EXISTS (SELECT CC2.componente_contenido
-> FROM compuesto_por AS CC2
-> WHERE CC.componente_contenido = CC2.componente_contenido AND CC2.componente_con
tenido = C.nro));
-> END IF;
-> END //
Query OK, 0 rows affected (0,01 sec)

mysql> DELIMITER ;
mysql> |

```

```

mysql> CALL todas_las_partes();
Query OK, 16 rows affected (0,00 sec)

```

- d. Listar, para cada nombre de parte, todos los nombres de las subpartes que la componen.

```

WITH RECURSIVE partesde_parte AS ( --query inicial
(SELECT DISTINCT C.componente_contenedor,C.componente_contenido, 0 lvl
FROM compuesto_por C
WHERE C.componente_contenedor NOT IN ( SELECT c.componente_contenido
FROM compuesto_por c
/*si no está entre los componentes contenidos, es el nodo raíz*/)
)
UNION ALL
(
--query recursiva #termina cuando la row que devuelve es null
SELECT c.componente_contenedor,c.componente_contenido, lvl+1

```

```

FROM compuesto_por c INNER JOIN partesde_parte p ON
p.componente_contenido=c.componente_contenedor)
)
SELECT componente_contenedor,componente_contenido,c.nombre AS
nombre_contenedor,(SELECT nombre FROM Tipo_componente comp WHERE
comp.nro=pp.componente_contenido) AS nombre_contenido
FROM partesde_parte pp, Tipo_componente c
WHERE pp.componente_contenedor=c.nro;

```

```

mysql> WITH RECURSIVE partesde_parte AS (
-> (SELECT DISTINCT C.componente_contenedor,C.componente_contenido, 0 lvl
-> FROM compuesto_por C
-> WHERE C.componente_contenedor NOT IN ( SELECT c.componente_contenido
-> FROM compuesto_por c)
-> )
-> UNION ALL
-> (
-> SELECT c.componente_contenedor,c.componente_contenido, lvl+1
-> FROM compuesto_por c INNER JOIN partesde_parte p ON p.componente_contenido=c.c
componente_contenedor)
-> )
-> SELECT componente_contenedor,componente_contenido,c.nombre AS nombre_contenedor
,&#x28;SELECT nombre FROM Tipo_componente comp WHERE comp.nro=pp.componente_contenido) AS n
ombre_contenido
-> FROM partesde_parte pp, Tipo_componente c
-> WHERE pp.componente_contenedor=c.nro;
blucle :c^C^C -- query aborted

```

- e. Insertar la tupla ParteDe(11,12) ¿Cómo evitaría el ciclo infinito que se produce al ejecutar la consulta recursiva anterior?

Solución (evitar ciclo infinito):

```
DELIMITER //
```

```
CREATE TRIGGER control_recursivo BEFORE INSERT ON compuesto_por FOR
EACH ROW
BEGIN
```

```
IF new.componente_contenedor IN (SELECT componente_contenido FROM
compuesto_por WHERE new.componente_contenedor= componente_contenido AND
new.componente_contenido= componente_contenedor) THEN
```

```
-- Me fijo si el componente que ahora cuenta como contenedor, ya no es
que forma parte del componente que cuenta como contenido en el insert
```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Error, el contenedor ingresado
ya es contenido en ese componente';
```

```
END IF;
```

```
END //
```

```
DELIMITER ;
```

```

Database changed
mysql> DELIMITER //
mysql> CREATE TRIGGER control_recursivo BEFORE INSERT ON compuesto_por FOR EACH ROW
-> BEGIN
-> IF new.componente_contenedor IN (
-> SELECT componente_contenido
-> FROM compuesto_por
-> WHERE new.componente_contenedor= componente_contenido AND new.componente_conten
ido= componente_contenedor)
HEN SIGN    -> THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Error, el contenedor ing
resado ya es contenido en ese componente';
-> END IF;
-> END //
Query OK, 0 rows affected (0,01 sec)

mysql> DELIMITER ;

```

- Al realizar la consulta:

```

INSERT INTO compuesto_por(componente_contenedor, componente_contenido)
VALUES(11,12);

```

```

mysql> INSERT INTO compuesto_por(componente_contenedor, componente_contenido) VALUES(1
1,12);
ERROR 1644 (45000): Error, el contenedor ingresado ya es contenido en ese componente
mysql> |

```

### 3. Para usar agregación, funciones de ventana y CTE (modificación de ER)

En caso de ser necesario modifique el modelo ER de manera de poder responder las consultas de más abajo.

3.1. Dada la consulta de naves y orbitas que estuvo añadida una columna adicional que muestre el tiempo promedio que estuvo en cada orbita.

```

SELECT *, AVG(DATEDIFF(fecha_fin, fecha_ini)) OVER (PARTITION BY matricula, sentido, altura, excentricidad ORDER BY matricula) as tiempoPromedioPorOrbita FROM esta;

```

```

mysql> SELECT *, AVG(DATEDIFF(fecha_fin, fecha_ini)) OVER (PARTITION BY matricula, sentido, altura, excentricidad ORDER BY matricula) as tiempoPromedioPorOrbita FROM esta;
+-----+-----+-----+-----+-----+-----+-----+
| fecha_ini | fecha_fin | excentricidad | altura | sentido | matricula | tiempoPromedioPorOrbita |
+-----+-----+-----+-----+-----+-----+-----+
| 2019-07-14 00:00:00 | 2019-08-09 00:00:00 | 47174354.87 | 2151718.84 | positivo | 1118075597 | 26.0000 |
| 2019-11-21 00:00:00 | 2020-11-19 00:00:00 | 67685037.07 | 67211107.75 | negativo | 1165517337 | 364.0000 |
| 2018-11-23 00:00:00 | 2019-11-21 00:00:00 | 8485447.70 | 93302528.76 | positivo | 1271811096 | 363.0000 |
| 2005-09-04 00:00:00 | 2005-11-02 00:00:00 | 64848104.23 | 83651558.27 | negativo | 1316787792 | 59.0000 |
| 2009-06-04 00:00:00 | 2009-05-04 00:00:00 | 98022544.37 | 13018454.41 | positivo | 1613643118 | -31.0000 |
| 2012-07-04 00:00:00 | 2012-09-02 00:00:00 | 29250851.46 | 22123335.67 | negativo | 1633627658 | 60.0000 |
| 2013-07-18 00:00:00 | 2013-09-08 00:00:00 | 64142024.52 | 98990996.24 | positivo | 1775411958 | 52.0000 |
| 2013-09-18 00:00:00 | 2013-10-16 00:00:00 | 67423270.09 | 33645686.05 | negativo | 1888519394 | 28.0000 |
| 2005-04-15 00:00:00 | 2005-12-13 00:00:00 | 83047706.22 | 73291281.60 | positivo | 1928584992 | 242.0000 |
| 2002-11-06 00:00:00 | 2003-03-03 00:00:00 | 35045385.83 | 13810323.01 | positivo | 2130478654 | 117.0000 |
+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0,00 sec)

```

3.2 . Liste el nombre de las agencias cuyas naves producen más del 50% del promedio de cantidad de basura en un año.

```

SELECT a.nombre FROM Agencia a JOIN Nave n ON
n.nombre_agencia_prop=a.nombre JOIN Produce p USING(matricula) JOIN
Basura b USING(id_basura)
HAVING COUNT(a.nombre)>=
(0.5*(SELECT AVG(count) FROM
(SELECT count(*) as count, matricula FROM Basura JOIN Produce
USING(id_basura) WHERE fecha=YEAR(now()) GROUP BY (matricula)
) AS temp));

```

3.3 Agregue una columna con el ranking las naves de acuerdo a la cantidad promedio de basura que produce por mes y año.

```

SELECT *, AVG(basuraPorNave) OVER (PARTITION BY matricula, MONTH(fecha))
as basuraNaveMes, AVG(basuraPorNave) OVER (PARTITION BY matricula,
YEAR(fecha)) as basuraNaveAño
FROM (SELECT *, count(*) as basuraPorNave
FROM Produce GROUP BY matricula) AS temp;

```

## 4. Índices

Para realizar los siguientes ejercicios inserte más de 100k de registros en las tablas basura. Puede utilizar un Excel con valores al azar e importarlos masivamente con el comando LOAD DATA

- a. Tome el tiempo de la consulta 1.1. Luego cree un índice sobre id en basura en la tabla Nave y volver a ejecutar la consulta 1.1. Ver si hay diferencia en los tiempos de ejecución con respecto a la primera.

```

WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM
Basura AS B GROUP BY B.matricula )
SELECT C.nombre
FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo
INNER JOIN Temp ON Temp.matricula = N.matricula
WHERE Temp.cant > 9;

```

```
mysql> WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basura AS
B GROUP BY B.matricula )
-> SELECT C.nombre
-> FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo INNER JOIN T
emp ON Temp.matricula = N.matricula
-> WHERE Temp.cant > 9;
+-----+
| nombre |
+-----+
| Vehiculos lanzadera |
| Vehiculos lanzadera |
| Vehiculos lanzadera |
| Vehiculos lanzadera |
| Naves no tripuladas o roboticas |
| Naves no tripuladas o roboticas |
| Naves no tripuladas o roboticas |
| Naves espaciales tripuladas |
| Naves espaciales tripuladas |
| Naves espaciales tripuladas |
+-----+
10 rows in set (0,07 sec)

mysql> |
```

```
+-----+
| tiempo → 0.07 sec |
+-----+
CREATE INDEX basura_index ON Basura(id_basura);
CREATE INDEX nave_index ON Nave(matricula);
```

Ejecuto nuevamente la consulta

```
mysql> WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basura AS
B GROUP BY B.matricula )
-> SELECT C.nombre
-> FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo INNER JOIN T
emp ON Temp.matricula = N.matricula
-> WHERE Temp.cant > 9;
+-----+
| nombre |
+-----+
| Vehiculos lanzadera |
| Vehiculos lanzadera |
| Vehiculos lanzadera |
| Vehiculos lanzadera |
| Naves no tripuladas o roboticas |
| Naves no tripuladas o roboticas |
| Naves no tripuladas o roboticas |
| Naves espaciales tripuladas |
| Naves espaciales tripuladas |
| Naves espaciales tripuladas |
+-----+
10 rows in set (0,04 sec)
```

```
+-----+
```

```
| tiempo → 0.04 sec |
+-----+
```

- b. Ejecutar el comando explain sobre la consulta anterior. Verificar que ahora se define un index scan. Es decir, se usa el índice. Ejecutar la consulta y proponer e implementar índices para mejorar las restantes consultas del punto 1 (al menos 2).

```
EXPLAIN SELECT C.nombre
FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo,
(SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basura AS B
GROUP BY B.matricula ) AS Temp
WHERE Temp.matricula = N.matricula AND Temp.cant > 9;
```

```
mysql> EXPLAIN SELECT C.nombre
-> FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo, (SELECT B.matricula AS Matricula, COU
NT(*) AS cant FROM Basuras AS B GROUP BY B.matricula ) AS Temp
-> WHERE Temp.matricula = N.matricula AND Temp.cant > 9;
ERROR 1146 (42S02): Table 'BDA_TPI.Naves' doesn't exist
mysql> EXPLAIN SELECT C.nombre
-> FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo, (SELECT B.matricula AS Matricula, COUN
T(*) AS cant FROM Basura AS B GROUP BY B.matricula ) AS Temp
-> WHERE Temp.matricula = N.matricula AND Temp.cant > 9;
+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+
| 1 | PRIMARY | C | NULL | ALL | PRIMARY | NULL | NULL | NULL | 3 | 100.00 | NULL |
| 1 | PRIMARY | N | NULL | ref | PRIMARY,idx_matricula,nave_index | PRIMARY | 182 | BDA_TPI.C.prefijo | 3 | 100.00 | Using index |
| 1 | PRIMARY | <derived2> | NULL | ref | <auto_key0> | <auto_key0> | 43 | BDA_TPI.N.matricula | 998 | 100.00 | NULL |
| 2 | DERIVED | B | NULL | index | fk_basura | fk_basura | 43 | NULL | 99875 | 100.00 | Using index |
+-----+
4 rows in set, 1 warning (0.00 sec)
```

Como resulta complicado entender el resultado por terminal lo muestro desde la herramienta Workbench:

```
76 • EXPLAIN SELECT C.nombre
77 FROM Clase_nave AS C INNER JOIN Nave AS N ON N.prefijo = C.prefijo, (
78 SELECT B.matricula AS Matricula, COUNT(*) AS cant
79 FROM Basura AS B
80 GROUP BY B.matricula) AS Temp
81 WHERE Temp.matricula = N.matricula AND Temp.cant > 9;
82
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	C		ALL	PRIMARY				3	100.00	
	1	PRIMARY	N		ref	PRIMARY,idx_matricula,nave_index	PRIMARY	182	BDA_TPI.C.prefijo	3	100.00	Using index
	1	PRIMARY	<derived2>		ref	<auto_key0>	<auto_key0>	43	BDA_TPI.N.matricula	998	100.00	
	2	DERIVED	B		index	fk_basura	fk_basura	43		99875	100.00	Using index



# 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:
  - a. Control de la integridad, coherencia y/o consistencia de los datos.

El siguiente Trigger controla la *excentricidad*, la cual si es igual a cero, automáticamente el atributo *circular* debe ser verdadero

```
DELIMITER //
CREATE TRIGGER verif_excentricidad 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 ;
```

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

Utilizamos un atributo nuevo que podría llamar *cant\_lanz* que hace referencia a la cantidad de lanzamientos de cada *Agencia*, por lo cual, antes que inserte un nuevo lanzamiento, se actualiza este atributo.

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

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

Añadimos una relacion de agregacion junto con la entidad *costos*, la cual posee como atributos los tipos de costos, mas un atributo que es la suma de estos, el cual 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.

Al introducir una *Nave*, se calcula la cantidad de órbitas que recorrió, cuantos tripulantes navegaron en la misma y cuantas basuras género la misma.

```

DELIMITER //
CREATE PROCEDURE infoNave (unaMatricula varchar(10))
BEGIN
    DECLARE cant_orbitas, cant_trip, basura_prod INT;
    SET cant_orbitas = (SELECT COUNT(*) FROM esta WHERE
matricula=unaMatricula GROUP BY matricula);
    SET cant_trip = (SELECT COUNT(*) FROM tiene WHERE
matricula=unaMatricula);
    SET basura_prod = (SELECT COUNT(*) FROM produce WHERE
matricula=unaMatricula);
    SELECT cant_orbitas, cant_trip, basura_prod; /*MOSTRAMOS
RESULTADOS*/
END//
DELIMITER ;

```

```

1  DELIMITER //
2  * CREATE PROCEDURE infoNave (unaMatricula varchar(10))
3  BEGIN
4      DECLARE cant_orbitas, cant_trip, basura_prod INT;
5      SET cant_orbitas = (SELECT COUNT(*) FROM esta WHERE matricula=unaMatricula GROUP BY matricula);
6      SET cant_trip = (SELECT COUNT(*) FROM tiene WHERE matricula=unaMatricula);
7      SET basura_prod = (SELECT COUNT(*) FROM produce WHERE matricula=unaMatricula);
8      SELECT cant_orbitas, cant_trip, basura_prod; /*MOSTRAMOS RESULTADOS*/
9  END//
10 DELIMITER ;
11

```

Output

#	Time	Action	Message
68	16:36:11	SELECT * FROM BDA_TPI.Empresa	20 row(s) returned
69	16:42:35	CREATE PROCEDURE infoNave (unaMatricula varchar(10)) BEGIN DECL...	0 row(s) affected

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

```

DELIMITER //
CREATE FUNCTION contarNaves_ingresaronOrbitas (fecha_inicio date, n
int) RETURNS int
DETERMINISTIC
BEGIN
    DECLARE fecha_final date;
    DECLARE cantidad int;
    SET fecha_final = (SELECT DATE_ADD(fecha_inicio, INTERVAL n
DAY));
    SET cantidad = (SELECT COUNT(DISTINCT matricula) FROM esta WHERE
fecha_ini BETWEEN fecha_inicio and fecha_final);
    RETURN cantidad;
END//
DELIMITER ;

```

```

1  DELIMITER //
2  CREATE FUNCTION contarNaves_ingresaronOrbitas (fecha_inicio date, n int) RETURNS int
3  DETERMINISTIC
4  BEGIN
5      DECLARE fecha_final date;
6      DECLARE cantidad int;
7      SET fecha_final = (SELECT DATE_ADD(fecha_inicio, INTERVAL n DAY));
8      SET cantidad = (SELECT COUNT(DISTINCT matricula) FROM esta WHERE fecha_ini BETWEEN fecha_inicio and fecha_final);
9      RETURN cantidad;
10 END//
11 DELIMITER ;
12

```

Output

#	Time	Action	Message	Durat
71	16:45:38	CREATE FUNCTION contarNaves_ingresaronOrbitas (fecha_inicio date, n int...	Error Code: 1418. This function has none of DETERMINISTIC, NO SQL, or R...	0.000
72	16:48:01	CREATE FUNCTION contarNaves_ingresaronOrbitas (fecha_inicio date, n int...	0 row(s) affected	0.016

## 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:
  - a. Programador
  - b. Diseñador
  - c. Administrador

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

```

mysql> CREATE ROLE IF NOT EXISTS programador, diseñador, administrador;
Query OK, 0 rows affected (0,01 sec)

mysql> |

```

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 "dis"@"localhost" IDENTIFIED BY "Di$2021" PASSWORD EXPIRE
INTERVAL 200 DAY;
CREATE USER "prog"@"localhost" IDENTIFIED BY "Progr@m2021" PASSWORD
EXPIRE INTERVAL 200 DAY;
CREATE USER "admin"@"%" IDENTIFIED BY "@dmin2021" PASSWORD EXPIRE
INTERVAL 200 DAY;
GRANT ALL ON BDA_TPI.* TO "admin"@"%";
GRANT ALTER, CREATE, DELETE, DROP, INDEX, INSERT, REFERENCES, SELECT,
TRIGGER, UPDATE, EVENT, LOCK TABLES ON BDA_TPI.* TO "dis"@"localhost";
GRANT SELECT, INSERT, UPDATE, TRIGGER, SHOW VIEW, CREATE VIEW ON
BDA_TPI.* TO "prog"@"localhost";
GRANT "programador" TO "prog"@"localhost";
GRANT "diseñador" TO "dis"@"localhost";
GRANT "administrador" TO "admin"@"%";
FLUSH PRIVILEGES;

```

```

mysql> CREATE USER "dis"@"localhost" IDENTIFIED BY "Di$2021" PASSWORD EXPIRE INTERVAL 200 DAY;
Query OK, 0 rows affected (0,00 sec)

mysql> CREATE USER "prog"@"localhost" IDENTIFIED BY "Progr@m2021" PASSWORD EXPIRE INTERVAL 200 DAY;
Query OK, 0 rows affected (0,01 sec)

mysql> CREATE USER "admin"@"%" IDENTIFIED BY "@dmin2021" PASSWORD EXPIRE INTERVAL 200 DAY;
Query OK, 0 rows affected (0,00 sec)

mysql> GRANT ALL ON BDA_TPI.* TO "admin"@"%";
Query OK, 0 rows affected (0,01 sec)

mysql> GRANT ALTER, CREATE, DELETE, DROP, INDEX, INSERT, REFERENCES, SELECT, TRIGGER, UPDATE, EVENT, LOCK TABL
ES ON BDA_TPI.* TO "dis"@"localhost";
Query OK, 0 rows affected (0,00 sec)

mysql> GRANT SELECT, INSERT, UPDATE, TRIGGER, SHOW VIEW, CREATE VIEW ON BDA_TPI.* TO "prog"@"localhost";
Query OK, 0 rows affected (0,01 sec)

mysql> GRANT "programador" TO "prog"@"localhost";
Query OK, 0 rows affected (0,01 sec)

mysql> GRANT "diseñador" TO "dis"@"localhost";
Query OK, 0 rows affected (0,00 sec)

mysql> GRANT "administrador" TO "admin"@"%";
Query OK, 0 rows affected (0,00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,01 sec)

```

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.

Iniciando con programador → crear una tabla

```
mysql -u prog -p
```

```

create table prueba(
    id int,
    primary key(id));

```

-- Esto no se debería permitir porque el programador no tiene permiso para crear tabla

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

Iniciando con diseñador → crear un usuario

```
mysql -u dis -p
```

```
create user 'prog2'@'localhost'
```

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

Iniciando con administrador → puede realizar cualquier acción, en este caso, crearemos el usuario anterior y le daremos los privilegios de programador.

```
mysql -u admin -p
```

```
create user 'prog2'@'localhost' IDENTIFIED BY 'N€wProg2021' PASSWORD
EXPIRE INTERVAL 200 DAY;
GRANT SELECT, INSERT, UPDATE, TRIGGER, SHOW VIEW, CREATE VIEW ON
BDA_TPI.* TO "prog2"@"localhost";
GRANT "programador" TO "prog2"@"localhost";
FLUSH PRIVILEGES;
```

```
mysql> create user 'prog2'@'localhost' IDENTIFIED BY 'N€wProg2021' PASSWORD EXPIRE INTERVAL 200
DAY;
Query OK, 0 rows affected (0,01 sec)

mysql> GRANT SELECT, INSERT, UPDATE, TRIGGER, SHOW VIEW, CREATE VIEW ON BDA_TPI.* TO "prog2"@"
localhost";
Query OK, 0 rows affected (0,00 sec)

mysql> GRANT "programador" TO "prog2"@"localhost";
Query OK, 0 rows affected (0,00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,00 sec)

mysql> |
```

## Respaldo y recuperación

Tomando como referencia los ejercicios de “Respaldo y restauración” de la guía del Trabajo Práctico N.º 5:

1. Planteen un escenario de volumen de transacciones que deberá soportar la base de datos.

Para un volumen de transacciones diarias (100.000 inserciones aprox.) y una disponibilidad requerida de la base de datos 24 horas al día, se requiere 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.

2. Elaboren, detallando y fundamentando cada ítem, un plan de respaldos y recuperación ante contingencias.

Analizando la carga de la base de datos y las horas pico de consulta y transaccionalidad, se fijó como hora cero para backup, las 3:00 PM, ya que en estas horas la base de datos se encuentra con un bajo nivel de carga.

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 define 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, rotandolas 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.

Semana 1:

	L	M	X	J	V	S	D
2:00 AM	A	A	A	A	A	A	A
8:00 AM	A	A	A	A	A	A	A
12:00 PM	C D1	C D2	C D3	C D4	C D5	C D6	
2:00 PM	F		F		F		
3:00 PM		A		A		A	A
8:00 PM	A	A	A	A	A	A	A

Semana 2:

	L	M	X	J	V	S	D
2:00 AM	A	A	A	A	A	A	A
8:00 AM	A	A	A	A	A	A	A
12:00 PM	C D7	C D8	C D9	C D10	C D11	C D12	
2:00 PM	F		F		F		
3:00 PM		A		A		A	A
8:00 PM	A	A	A	A	A	A	A

Donde:

A	Backup de los archivos log.
C D#	Cinta del día.
F	Full backup de los archivos de la base de datos y de archivos log.

- Implementen y documenten (comandos ejecutados y capturas de pantallas) en el servidor todo lo que se pueda implementar de acuerdo al plan elaborado para resguardar los datos.

Habilitamos los binary logs.



```
root@srv-bbdd: /etc/mysql
GNU nano 4.8 mysql.cnf
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License, version 2.0, for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
#
# The MySQL Server configuration file.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
#
# * IMPORTANT: Additional settings that can override those from this file!
#   The files must end with '.cnf', otherwise they'll be ignored.
#
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mysql.conf.d/

[mysqld]
log_bin = /var/log/mysql/mysql-bin.log
binlog_expire_logs_seconds = 259200
max_binlog_size = 100M

[ Cancelado ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Texto ^J Justificar ^C Posición M-U Deshacer M-A Marcar text
^X Salir ^R Leer fich. ^N Reemplazar ^U Pegar ^T Ortografía ^_ Ir a línea M-E Rehacer M-G Copiar
```

Ejecutamos el comando para realizar el full backup:

```
sudo mysqldump --user=ADMIN --password=admintpi --flush-logs
--delete-source-logs --single-transaction --all-databases | gzip >
/var/backups/mysql/full_$(date +%d-%m-%Y_%H-%M)-inc.gz
```

```
administrador@srv-bbdd:~$ sudo su
root@srv-bbdd:/home/administrador# sudo mysqldump --user=ADMIN --password=admintpi --flush-logs
--delete-source-logs --single-transaction BDA_TPI | gzip > /var/backups/mysql/$(date +%d-%m-%Y
_%H-%M)-inc.gz
mysqldump: [Warning] Using a password on the command line interface can be insecure.
root@srv-bbdd:/home/administrador# exit
exit
administrador@srv-bbdd:~$ |
```

Al ejecutar este comando obtenemos un archivo comprimido con el siguiente formato de nombre 'dd-mm-aaaa\_hh-mm-inc'

```
administrador@srv-bbdd:~$ cd /var/backups/mysql/
administrador@srv-bbdd:/var/backups/mysql$ ls
01-12-2021_17-11-inc.gz
administrador@srv-bbdd:/var/backups/mysql$ ls -al
total 4368
drwxr-xr-x 2 root root 4096 dic 1 17:11 .
drwxr-xr-x 3 root root 4096 dic 1 17:07 ..
-rw-r--r-- 1 root root 4463787 dic 1 17:11 01-12-2021_17-11-inc.gz
administrador@srv-bbdd:/var/backups/mysql$ |
```

Para implementar los full backups, agregamos este comando al crontab

```
0 0 */15 * 0 sudo mysqldump --user=ADMIN --password=admintpi
--flush-logs --delete-source-logs --single-transaction --all-databases
```

```
| gzip > /var/backups/mysql/full_$(date +%d-%m-%Y_%H-%M)-inc.gz
0 0 */3* * 1,5 sudo bash ~/scripts/mysql_inc_backup.sh
```

```
administrador@srv-bbdd:/var/backups/mysql$ crontab -e
no crontab for administrador - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano          <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed

Choose 1-4 [1]: 1
crontab: installing new crontab
administrador@srv-bbdd:/var/backups/mysql$ |
```

```
GNU nano 4.8 /tmp/crontab.Yqpg07/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
0 0 */15 * 0 sudo mysqldump --user=ADMIN --password=admintpi --flush-logs --delete-source->
0 0 */3* * 1,5 sudo bash ~/scripts/mysql_inc_backup.sh

^G Ver ayuda  ^O Guardar    ^W Buscar     ^K Cortar Texto ^J Justificar  ^C Posición
^X Salir      ^R Leer fich. ^\ Reemplazar  ^U Pegar       ^T Ortografía  ^_ Ir a línea
```

Nota: en un principio consideramos realizar un script que realice el resguardo pero al ejecutarlo la salida del script era inutilizable

```
USER=ADMIN
PASSWORD=admintpi
DATABASE=TPI_BDA
mysqldump --user=$USER --password=$PASSWORD --single-transaction
```

```
--flush-logs BDA_TPI > "$DATABASE$(date +%Y%m%d-%H%M)".sql
gzip "$DATABASE$(date +%Y%m%d-%H%M)".sql
```

```
administrador@srv-bbdd:~$ ls
01-12-2021_17-03-inc.gz  'backup.sql'$'\r'$.gz'  -inc.sql
archivos                backup-TPI2.sh          TPI_2daEntrega_FULL.sql
backup-BDA_TPI.sh       backup-TPI.sh.save      'TPI_BDA'$'\r'$(date +%Y%m%d-%H).sql'$'\r'$.gz'
backup.sh               BDA_TPI                 'TPI_BDA'$'\r'$.sql'$'\r'$.gz'
administrador@srv-bbdd:~$ |
```

4. Elaboren 2 escenarios de posibles contingencias, uno con pérdida total de los datos y otro en el que se detecte que se hicieron inserciones/ modificaciones/ borrados erróneos o maliciosos que dejaron datos inconsistentes.
  - a. Utilizando la funcionalidad de crear “instantáneas” de la máquina virtual para recuperar posteriormente un estado consistente, simulen dos situaciones en las que se producen contingencias a partir de las cuales se debe recurrir al plan elaborado para recuperar los datos.
  - b. Realicen el plan de restauración para cada situación, describiendo y documentando detalladamente el procedimiento utilizado en cada caso y el estado en que quedaría la base de datos, incluso si hubo pérdida de algunos datos. ¿Tienen un plan "B" por si falla alguna restauración?.

Primer caso de pérdida :

Pérdida total de la base de datos, se hace un drop de la base de datos y se debe restaurar. En el caso de que se pierda la BD el día Martes antes del resguardo incremental, se debería tomar el último resguardo full si se hizo el día anterior, si no se hizo un full el día anterior se debería tomar el último full y el último diferencial .

Segundo caso de pérdida :

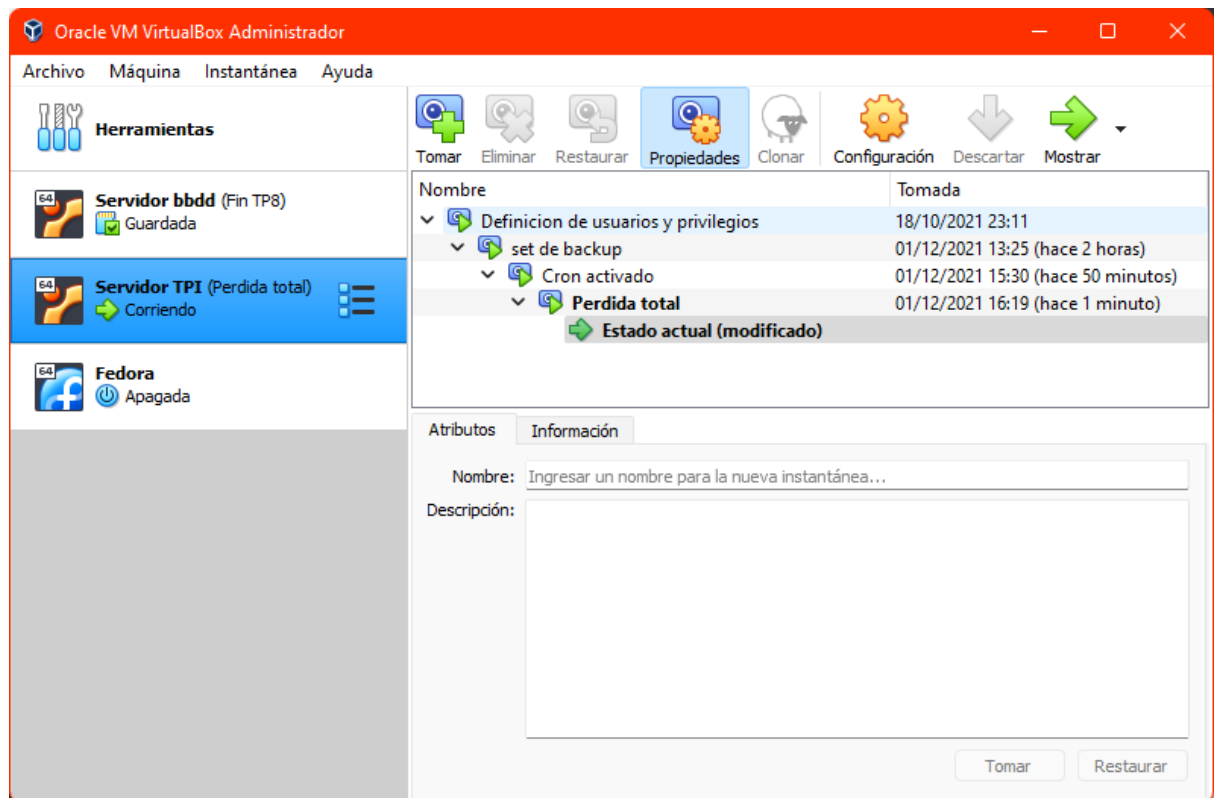
Datos inconsistentes : algún fallo de foreign key o algo similar .

Si no se llegó a hacer el resguardo incremental de los días Jueves y no se hizo un full el día Miércoles, se debe tomar el último diferencial.

Si se llegó a hacer el incremento del Jueves , tomar el incremental.

Si ocurre entre el Martes y el Jueves se debería tomar el último diferencial.

Pérdida total:



Como podemos ver no se encuentra la BD BDA\_TPI

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| menagerie |
| mysql |
| performance_schema |
| sakila |
| sys |
+-----+
6 rows in set (0,00 sec)

mysql> |
```

Ahora restauramos la base de datos con el siguiente comando:

```
sudo gunzip < full_01-12-2021_19-32-inc.gz | sudo mysql -u ADMIN -p
administrador@srv-bbdd:~$ sudo gunzip < full_01-12-2021_19-32-inc.gz | sudo mysql -u ADMIN -p
Enter password:
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| BDA_TPI  |
| information_schema |
| menagerie |
| mysql    |
| performance_schema |
| sakila   |
| sys      |
+-----+
7 rows in set (0,00 sec)
```

## 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 de 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 paredez",
    "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" : "xxxxxxxxxx",
          "email" : "xxxxx",
          "redes sociales" : [
```

```

        {"instagram": "juanpa24"},
        {"twitter": "juanpa24" }}
    }},
    { "accionista":
      { "dni": 40404040,
        "nombre": "dfdfdsfds",
        "edad": 44,
        "ciudad": "xxxxxx",
        "nivel estudio": "xxxxxxxxxxx",
        "email": "xxxxx",
        "redes sociales": [
          {"instagram": "juanpa24"},
          {"twitter": "juanpa24" }}
        ]
      }
    ]
  }
}

```

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

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

Generamos un JsonSchema a partir del ejemplo brindado por la consigna

*Nota: En el siguiente link se encuentra la BD con con los datos cargados*  
[full\\_BDA\\_TPI-GRUPO10.sql](#)

### 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 -> '$."Principal Accionista".edad') FROM Empresa
WHERE (accionista -> '$."Principal Accionista"."nivel estudio"' LIKE
'%universitario%');
```

```

28 • SELECT avg(accionista -> '$."Principal Accionista".edad') as "EDAD PROMEDIO"
29 FROM empresa
30 WHERE (accionista -> '$."Principal Accionista"."nivel estudio"' LIKE '%universitario%';

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
EDAD PROMEDIO				
▶ 38.4				

2. Listar los nombres de los mundos con mayor cantidad de interesados.

```
DELIMITER //
```

```
CREATE PROCEDURE mundo_mas_aspirado ()
```

```
BEGIN
```

```
DECLARE i, maximo INT DEFAULT 0;
```

```
DROP TABLE IF EXISTS temp;
```

```
CREATE TABLE temp (mundos VARCHAR(100));
```

```
WHILE i<(SELECT MAX(json_length(Accionista -> '$."Principal
```

```
Accionista".aspiraciones')) FROM Empresa) DO
```

```
INSERT INTO temp SELECT
```

```
JSON_UNQUOTE(JSON_EXTRACT(Accionista,CONCAT('$."Principal
```

```
Accionista".aspiraciones[',i,'].mundos')))) as mundos FROM Empresa;
```

```
SET i=i+1;
```

```
END WHILE;
```

```
SET maximo = (SELECT MAX(a.cuenta) FROM (SELECT COUNT(*) cuenta FROM
```

```
temp
```

```
WHERE mundos IS NOT NULL GROUP BY mundos) as a);
```

```
SELECT mundos FROM temp WHERE mundos IS NOT NULL GROUP BY mundos HAVING
```

```
COUNT(*)=maximo;
```

```
DROP TABLE IF EXISTS temp;
```

```
END //
```

```
DELIMITER ;
```

```
CALL mundo_mas_aspirado();
```

```

6 CREATE TABLE temp (mundos VARCHAR(100));
7 WHILE i<(SELECT MAX(json_length(Accionista -> '$."Principal Accionista".aspiraciones')) FROM Empresa) DO
8 INSERT INTO temp SELECT
9 JSON_UNQUOTE(JSON_EXTRACT(Accionista,CONCAT('$."Principal Accionista".aspiraciones['',i,']'.mundos'))) as mundos FROM Empresa;
10 SET i=i+1;
11 END WHILE;
12 SET maximo = (SELECT MAX(a.cuenta) FROM (SELECT COUNT(*) cuenta FROM temp
13 WHERE mundos IS NOT NULL GROUP BY mundos) as a);
14 SELECT mundos FROM temp WHERE mundos IS NOT NULL GROUP BY mundos HAVING
15 COUNT(*)=maximo;
16 DROP TABLE IF EXISTS temp;
17 END //
18 DELIMITER ;
19 CALL mundo_mas_aspirado();

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

mundos
--------

Result 1 x			
Output			
Action Output			
#	Time	Action	Message
27	02:07:13	DROP procedure mundo_mas_aspirado	Error Code: 1305. PROCED
28	02:07:35	DROP procedure mundo_mas_aspirado	Error Code: 1305. PROCED
29	02:07:35	DROP procedure mundo_mas_aspirado	Error Code: 1305. PROCED
30	02:07:44	DROP procedure mundo_mas_aspirado	Error Code: 1305. PROCED
31	02:08:42	CREATE PROCEDURE mundo_mas_aspirado () BEGIN DECLARE i, maximo INT DEFAULT 0; DROP TABLE IF EXISTS temp; CREATE TABLE temp ...	0 row(s) affected
32	02:08:48	CALL mundo_mas_aspirado()	0 row(s) returned

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

```

SELECT Accionista -> '$."Principal Accionista".nombre' as Nombre,
Accionista -> '$."Principal Accionista".edad' as Edad,
Accionista -> '$."Principal Accionista".ciudad' as Ciudad
FROM empresa
WHERE Accionista -> '$."Principal Accionista".viajes[*].tipo' LIKE
'%Cientifico%';

```



```

12 • SELECT Accionista -> '$."Principal Accionista".nombre' as Nombre,
13 Accionista -> '$."Principal Accionista".edad' as Edad,
14 Accionista -> '$."Principal Accionista".ciudad' as Ciudad
15 FROM empresa
16 WHERE Accionista -> '$."Principal Accionista".viajes[*].tipo' LIKE '%Cientifico%';

```

	Nombre	Edad	Ciudad
►	"Braná Cornelissen"	50	"Walakeri"
	"Minnaminnie Rushworth"	26	"Pawitan"
	"Jacquie Roads"	31	"Mino"
	"Robbi Amer"	38	"Kassala"
	"Tommie Champe"	44	"Hacqabul"
	"Ginger Mityakov"	49	"Balesari"
	"Sydelle Stanex"	32	"Krásná Lípa"
	"Jacenta Tinwell"	57	"Bornu Yassu"
	"Holly Wickett"	52	"Kými"
	"Lesley Covotti"	30	"Guanyinsi"
	"Christyna Adolthine"	51	"Goba"
	"Shepard Clementet"	36	"Peterhof"
	"Emiline Orford"	37	"Oktyabr'skiy"
	"Kristos Birdfield"	41	"Hammām Da..."

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

```

SELECT Accionista -> '$."Principal Accionista".nombre' as Nombre,
Accionista ->
'$."Principal Accionista".edad' as Edad, Accionista -> '$."Principal
Accionista".ciudad' as Ciudad, Accionista -> '$."Principal
Accionista".viajes[*].tipo' as Tipo
FROM Empresa
WHERE Accionista -> '$."Principal Accionista".viajes[*].tipo' NOT LIKE
'[%Cientifico%]' AND Accionista -> '$."Principal
Accionista".viajes[*].tipo'
NOT LIKE '[%Militar%]';

```

```

1 • SELECT Accionista -> '$."Principal Accionista".nombre' as Nombre,
2 Accionista -> '$."Principal Accionista".edad' as Edad,
3 Accionista -> '$."Principal Accionista".ciudad' as
4 Ciudad, Accionista -> '$."Principal Accionista".viajes[*].tipo' as Tipo
5 FROM Empresa
6 WHERE Accionista -> '$."Principal Accionista".viajes[*].tipo' NOT LIKE
7 '["Cientifico"]' AND Accionista -> '$."Principal Accionista".viajes[*].tipo'
8 NOT LIKE '["Militar"]';

```

Result Grid				
Filter Rows:		Export:		
Wrap Cell Content:				
	Nombre	Edad	Ciudad	Tipo
▶	"Rosabella Hawkswell"	35	"Nginokrajan"	["Espia"]
	"Shannon O Sirin"	48	"Siepraw"	["Espia"]
	"Taffy Christauffour"	43	"Villanova"	["Espia"]

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

```

SELECT Accionista -> '$."Principal Accionista".nombre' as Nombre,
Accionista ->
'$."Principal Accionista"."redes sociales"' as redes_sociales
FROM Empresa
WHERE Accionista -> '$."Principal Accionista".viajes[*].mundos' LIKE
'%Venus%'
AND Accionista -> '$."Principal Accionista".aspiraciones[*].mundos'
LIKE
'%Saturno%';

```

```

1 • SELECT Accionista -> '$."Principal Accionista".nombre' as Nombre,
2 Accionista -> '$."Principal Accionista".redes sociales' as redes_sociales
3 FROM Empresa
4 WHERE Accionista -> '$."Principal Accionista".viajes[*].mundos' LIKE '%Venus%'
5 AND Accionista -> '$."Principal Accionista".aspiraciones[*].mundos' LIKE '%Saturno%';

```

< Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Nombre	redes_sociales
--------	----------------

Debido a que la consulta inicialmente no retorna ningún valor, ya que no existe ninguna tupla que coincida con la condición, realizaremos una modificación a los valores, de forma tal que exista al menos uno.

```

UPDATE Empresa SET Accionista=( '{"Principal
Accionista":{"dni":37992248,"nombre":"Shepard
Clementet","edad":36,"ciudad":"Peterhof","nivel
estudio":"terciario","email":"sclementetd@hc360.com","redes
sociales":[{"instagram":"hmcgilben0","twitter":"rwipfler0","facebook":"
rthomtson
0"}]},{"accionistas
secundarios":[{"accionista":{"dni":25849413,"nombre":"Juliana
Feldberg","edad":40,"ciudad":"Zhuxi Chengguanzhen","nivel
estudio":"terciario","email":"jfeldberg0@topsy.com","redes
sociales":[{"instagram":"nromanini0","twitter":"fandraud0","facebook":"
gbuzza0"}
]}]},{"accionista":{"dni":35354314,"nombre":"Tracie
Orcott","edad":44,"ciudad":"Oratorio","nivel
estudio":"primario","email":"torcott1@cmu.edu","redes
sociales":[{"instagram":"sclericoates0","twitter":"rmeddick0","facebook
":"racome
0"}]}}]},{"accionista":{"dni":36536920,"nombre":"Beverie
Affuso","edad":28,"ciudad":"Wāling","nivel
estudio":"universitario","email":"baffuso2@i2i.jp","redes
sociales":[{"instagram":"gwhitlow0","twitter":"zmacquaker0","facebook":"
hsaylor0
"}]}}]},{"viajes":[{"mundos":"Kepler-186f","tipo":"Cientifico"}, {"mundos"
:"Venus",
"tipo":"Espia"}]},{"aspiraciones":[{"mundos":"Venus","tipo":"Espia"}, {"mu

```

```

ndos": "HI
P 13044b", "tipo": "Espia"}, {"mundos": "Saturno", "tipo": "Militar"}]}}')
WHERE
CIF="R2352527B";





```

Si ejecutamos la consulta anterior, luego de haber realizado esta modificación, obtenemos el siguiente resultado:

```

1 • SELECT Accionista -> '$."Principal Accionista", nombre' as Nombre,
2 Accionista -> '$."Principal Accionista", redes sociales"' as redes_sociales
3 FROM Empresa
4 WHERE Accionista -> '$."Principal Accionista".viajes[*].mundos' LIKE '%Venus%'
5 AND Accionista -> '$."Principal Accionista".aspiraciones[*].mundos' LIKE '%Saturno%';
6
7 • UPDATE Empresa
8 SET Accionista=('{ "Principal Accionista": {"dni": 37992248, "nombre": "Shepard Clementet", "edad": 36, "ciudad": "Peterhof", "nive:

```

<   Filter Rows:  | Export:  | Wrap Cell Content: 

Nombre	redes_sociales
Shepard Clementet	[{"twitter": "rwipfler0", "facebook": "rthomson ...