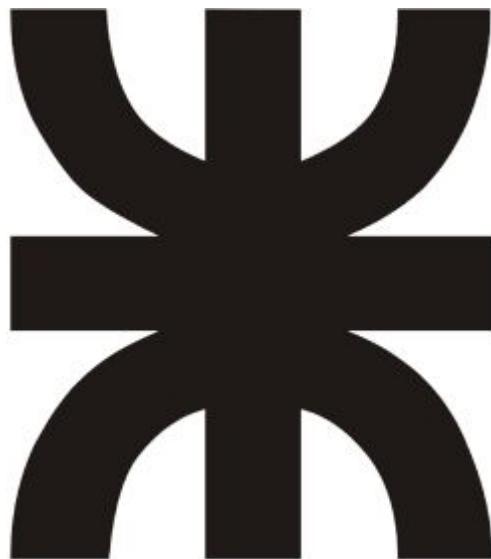


# UNIVERSIDAD TECNOLÓGICA NACIONAL

## FACULTAD REGIONAL RESISTENCIA



INGENIERÍA EN SISTEMAS DE INFORMACIÓN

## BASES DE DATOS

## TRABAJO PRÁCTICO INTEGRADOR

**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

2020

## **Consignas generales para la primer etapa**

### **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 los siguiente, relacionado al SGBD implementado:**

- **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 4GB.

- **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.

- **¿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 32/64 bit.

- **Indicar cuáles son las alternativas para la instalación.**

MySQL puede ser instalado en:

Operating System	Architecture	8.0	5.7	5.6
<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 20.04 LTS	x86_64	*		
Ubuntu 18.04 LTS	x86_32, x86_64	*	*	
Ubuntu 16.04 LTS	x86_32, x86_64	*	*	
<b>SUSE</b>				
SUSE Enterprise Linux 15 / OpenSUSE 15	x86_64	*		
SUSE Enterprise Linux 12 (12.4+)	x86_64	*	*	*
<b>Debian</b>				
Debian GNU/Linux 10	x86_64	*	*	
Debian GNU/Linux 9	x86_32, x86_64	*	*	*
<b>Microsoft Windows Server</b>				
Microsoft Windows 2019 Server	x86_64	*		
Microsoft Windows 2016 Server	x86_64	*	*	*
Microsoft Windows 2012 Server R2	x86_64	*	*	*
<b>Microsoft Windows</b>				
Microsoft Windows 10	x86_64	*	*	
<b>Apple</b>				
macOS 10.15	x86_64	*		
<b>FreeBSD</b>				
FreeBSD 12	x86_64	*		
<b>Various Linux</b>				
Generic Linux (tar format)	x86_32, x86_64, glibc 2.12, libstdc++ 4.4	*	*	*
Yum Repo		*	*	*
APT Repo		*	*	*
SUSE Repo		*	*	*

Además, podemos realizar la instalación de los sistemas operativos de tipo UNIX/Linux mediante los binarios.

Cuando instalamos en Linux, tenemos la posibilidad de descargar una versión específica o agregar el repositorio de MySQL a la lista de repositorios.

- ¿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 esto.

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.

- **Cuando la base de datos está vacía ¿Cuánto mide el espacio de tablas?**

```
root@srv-bbdd:/home/administrador# cd /var/lib/mysql
root@srv-bbdd:/var/lib/mysql# du -sh *
4,0K    auto.cnf
4,0K    BDA_TPI
4,0K    binlog.000001
4,0K    binlog.000002
4,0K    binlog.index
4,0K    ca-key.pem
4,0K    ca.pem
4,0K    client-cert.pem
4,0K    client-key.pem
192K   #ib_16384_0 dblwr
8,2M   #ib_16384_1 dblwr
4,0K    ib_buffer_pool
12M    ibdata1
48M    ib_logfile0
48M    ib_logfile1
12M    ibtmp1
164K   #innodb_temp
36K    mysql
25M    mysql.ibd
1,6M   performance_schema
4,0K   private_key.pem
4,0K   public_key.pem
4,0K   server-cert.pem
4,0K   server-key.pem
84K    sys
10M   undo_001
10M   undo_002
root@srv-bbdd:/var/lib/mysql#
```

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 BDA\_TPI que es el de la base de datos vacía:

Podemos ver que el directorio BDA\_TPI pesa 4KB.

- **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

- **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.

- **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.

- **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.

- **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.

- **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.
- 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

- 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.
- Indicar cuántos nodos 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.
- **¿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 vista 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

- **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

- Event
- Function
- Index
- Procedure
- Function
- Spatial Reference System
- Table
- Trigger
- View

- **Realice una síntesis sobre principales diferencias, ventajas y desventajas frente a otros SGBDs (elija 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
Tabla comparativa de MySQL vs MongoDB		
Desarrolladores	si	si
SO	MongoDB Inc.	Oracle Corporation
Lenguaje query	Multiplataforma	Multiplataforma
Mapa reducido	Javascript	SQL
Conversión de DB	si	no
Analisis 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 sincronizan 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 (0.33 sec)
```

```
mysql> CREATE USER 'OPERADOR'@'%' IDENTIFIED BY 'operadortpi';
Query OK, 0 rows affected (0.34 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 (0.33 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 BDA_TPI CHARSET=latin1 COLLATE=latin1_spanish_ci;
```

Action Output ▾				
#	Time	Action	Message	Duration / Fetch
1	13:19:11	CREATE DATABASE BDA_TPI CHARSET=latin1 COLLATE=latin1_spanish_ci...	1 row(s) affected	0,563 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 (0.11 sec)

mysql> REVOKE GRANT OPTION ON BDA_TPI.* FROM 'OPERADOR'@'%';
Query OK, 0 rows affected (0.02 sec)

mysql> REVOKE CREATE USER ON *.* FROM 'OPERADOR'@'%';
Query OK, 0 rows affected (0.02 sec)

mysql>
```

### 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 [Esquema Relacional Original- TPI.sql](#) o el archivo “Escenario Original-query’s 1”.

### **1. Consultas**

#### **1.1. Nombre de las naves que produjo al menos 10 basuras distintas.**

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

```
1 • use BDA_TPI;
2 • WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basuras AS B GROUP BY B.matricula )
3 SELECT C.nombre
4 FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo INNER JOIN Temp ON Temp.matricula = N.matricula
5 WHERE Temp.cant > 9;
6
```

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 especiales tripuladas
Naves especiales tripuladas
Naves especiales tripuladas

#### **1.2. 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 Basuras AS B1 , Basuras AS B2
WHERE B2.id_basura <> B1.id_basura AND B1.matricula = B2.matricula;
```

1.3. Listar los nombres de las agencias que no lanzaron ninguna nave que haya estado en orbita.

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

```

1 USE BDA_TPI;
2 • SELECT A2.nombre
3   FROM Agencias AS A2
4   WHERE A2.nombre NOT IN (SELECT A.nombre
5     FROM Agencias AS A INNER JOIN Lanza AS L ON L.nombre_agencia = A.nombre INNER JOIN Naves AS N ON L.matricula = N.matricula, O
6   WHERE O.sentido = L.sentido AND O.altura = L.altura AND L.excentricidad = O.excentricidad );

```

nombre
Amazon.com
Apple Inc.
Aeron Impex
Eridson
Global Print

#### 1.4. 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 Naves 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));

```

```

1 USE BDA_TPI;
2 • SELECT E.excentricidad, E.altura, E.sentido
3   FROM esta AS E
4   WHERE NOT EXISTS (SELECT N.matricula
5     FROM Naves AS N
6     WHERE NOT EXISTS (SELECT E2.excentricidad, E2.altura, E2.sentido
7       FROM esta AS E2
8       WHERE E2.excentricidad = E.excentricidad AND E.altura= E2.altura
9           AND E.sentido = E2.sentido AND E2.matricula = N.matricula));

```

excentricidad	altura	sentido

#### 1.5. 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 Privadas
                   GROUP BY nombre_publica)
SELECT P.nombre_e, P.nombre
  FROM Publicas AS P INNER JOIN supervisa AS S ON S.nombre_publica = P.nombre_e
 WHERE S.cant > 1;

```

Query 1

```

1 • USE BDA_TPI;
2 • WITH supervisa AS (SELECT nombre_publica, COUNT(*) AS cant
3   FROM Privadas
4   GROUP BY nombre_publica)
5   SELECT P.nombre_e, P.nombre
6   FROM Publicas AS P INNER JOIN supervisa AS S ON S.nombre_publica = P.nombre_e
7   WHERE S.cant > 1;
8
9

```

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

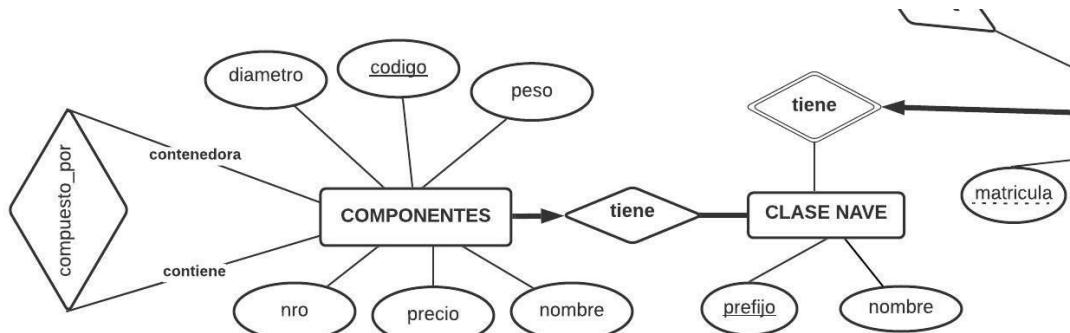
nombre_e	nombre
SpaceX	Metro Cash&Carry

## 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:** [Punto 2.sql](#) o en el archivo “Modificación-query’s 2.sql”

**DER MODIFICADO:**



#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

**2.1.Nombre de los componentes que forman parte de al menos dos 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)>1)
SELECT c.nombre
FROM compuesto as CC inner join Componentes c on c.nro = CC.cont;
```

Query 1

```
1 • USE BDA_TPI2;
2 • Ⓜ WITH compuesto AS (SELECT CC.componente_contenido as cont
3   FROM compuesto_por AS CC
4   GROUP BY CC.componente_contenido
5   HAVING COUNT(DISTINCT CC.componente_contenedor)>1)
6   SELECT c.nombre
7   FROM compuesto as CC inner join Componentes c on c.nro = CC.cont;
8
9
10
```

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

nombre
Conducto Dual Interno
Tornillo N45
Recidador Catalitico Interno
Giroscopio Coaxial Automatico
Inyector Holografico
Interruptor Subespacial De Emergencia
Giroscopio Dual Automatico
Comutador Holografico Plasmatico
Inyector Trifasico Rotatorio

2.2. 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 Componentes AS C1, Componentes 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))
```

The screenshot shows a MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, a query editor window displays the following SQL code:

```

1 • USE BDA_TPI2;
2 • SELECT C1.nombre, C1.nro , C2.nombre, C2.nro
3   FROM Componentes AS C1, Componentes AS C2
4   WHERE C1.nombre <> C2.nombre AND C1.nro IN (SELECT cx.componente_contenido
5     FROM compuesto_por AS cx
6     WHERE cx.componente_contenedor IN
7       (SELECT cc.componente_contenido
8        FROM compuesto_por AS cc
9        WHERE cc.componente_contenedor = C2.nro))
10
11
12

```

Below the code, the results are displayed in a grid:

nombre	nro	nombre	nro
Sanguchito de Miga Aeroespacial	16	Conducto Dual Interno	1
Injector Trifásico Rotatorio	15	Conducto Dual Interno	1
Commutador Holográfico Plasmático	14	Conducto Dual Interno	1
Giroscopio Dual Automático	13	Conducto Dual Interno	1
Recipitador Coaxial	12	Conducto Dual Interno	1
Interruptor Subespacial De Emergencia	11	Conducto Dual Interno	1
Injector Holográfico	10	Conducto Dual Interno	1
Junta Holográfica Principal	9	Conducto Dual Interno	1
Giroscopio Coaxial Automático	8	Conducto Dual Interno	1
Recipitador Catalítico Interno	7	Conducto Dual Interno	1
Tornillo N45	6	Conducto Dual Interno	1
Injector Subdual Plasmático	5	Conducto Dual Interno	1
Commutador de Flujo	4	Conducto Dual Interno	1
Actuador Catalítico	3	Conducto Dual Interno	1
Computadora aviónica	2	Conducto Dual Interno	1
Sanguchito de Miga Aeroespacial	16	Computadora aviónica	2
Injector Trifásico Rotatorio	15	Computadora aviónica	2
Commutador Holográfico Plasmático	14	Computadora aviónica	2
Giroscopio Dual Automático	13	Computadora aviónica	2
Recipitador Coaxial	12	Computadora aviónica	2

At the bottom of the results grid, it says "Result 90 x". Below the grid, there is a "Output:" section.

**2.3. 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 Componentes 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 Componentes 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 Componentes 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 ;

```

Query 1 >

```

1 • use BDA_TPI2;
2 • drop procedure todas_las_partes;
3 DELIMITER //
4 • CREATE PROCEDURE todas_las_partes()
5 BEGIN
6     IF (SELECT CC.componente_contenido
7         FROM compuesto_por AS CC
8     WHERE NOT EXISTS (SELECT C.nro
9         FROM Componentes AS C
10        WHERE C.nro <16 AND NOT EXISTS
11            (SELECT CC2.componente_contenido
12                FROM compuesto_por AS CC2
13               WHERE CC.componente_contenido = CC2.componente_contenido AND CC2.componente_contenido = C.nro))) IS NULL THEN
14         INSERT INTO compuesto_por(componente_contenedor, componente_contenido) SELECT nro, 1 FROM Componentes WHERE nro < 16
15         AND nro NOT IN (SELECT componente_contenedor FROM compuesto_por WHERE componente_contenido = 1);
16     else
17         SELECT CC.componente_contenido
18             FROM compuesto_por AS CC
19             WHERE NOT EXISTS (SELECT C.nro
20                 FROM Componentes AS C
21                 WHERE C.nro <16 AND NOT EXISTS (SELECT CC2.componente_contenido
22                     FROM compuesto_por AS CC2
23                     WHERE CC.componente_contenido = CC2.componente_contenido AND CC2.componente_contenido = C.nro));
24     END IF;
25 END //
26 DELIMITER ;
27 • call todas_las_partes;
28

```

Action Output

#	Time	Action	Message
15	21:59:59	use BDA_TPI2	0 row(s) affected
16	21:59:59	drop procedure todas_las_partes	0 row(s) affected
17	21:59:59	CREATE PROCEDURE todas_las_partes() BEGIN IF (SELECT CC.componente_contenido FROM compuesto_por AS CC WHERE NOT EXISTS (S...)	0 row(s) affected
18	21:59:59	call todas_las_partes	14 row(s) affected

## 2.4. 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 Componentes comp WHERE comp.nro=pp.componente_contenido) AS nombre_contenido
FROM partesde_parte pp, Componentes c
WHERE pp.componente_contenedor=c.nro

```

componente_contenedor	componente_contenido	nombre_contenedor	nombre_contenido
1	2	Conducto Dual Interno	Computadora avionica
1	3	Conducto Dual Interno	Actuador Catalitico
1	4	Conducto Dual Interno	Commutador de Flujo
1	16	Conducto Dual Interno	Sanguchito de Miga Aeroespacial
5	6	Inyector Subdual Plasmatico	Tornillo N45
5	7	Inyector Subdual Plasmatico	Recidador Catalitico Interno
6	8	Tornillo N45	Giroscopio Coaxial Automatico
7	15	Recidador Catalitico Interno	Inyector Trifasico Rotatorio
8	14	Giroscopio Coaxial Automatico	Comutador Holografico Plasmatico
9	10	Junta Holografica Principal	Inyector Holografico
9	11	Junta Holografica Principal	Interruptor Subespacial De Emerg...
12	11	Recidador Coaxial	Interruptor Subespacial De Emerg...
12	13	Recidador Coaxial	Giroscopio Dual Automatico
13	11	Giroscopio Dual Automatico	Interruptor Subespacial De Emerg...

## 2.5.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
//
```

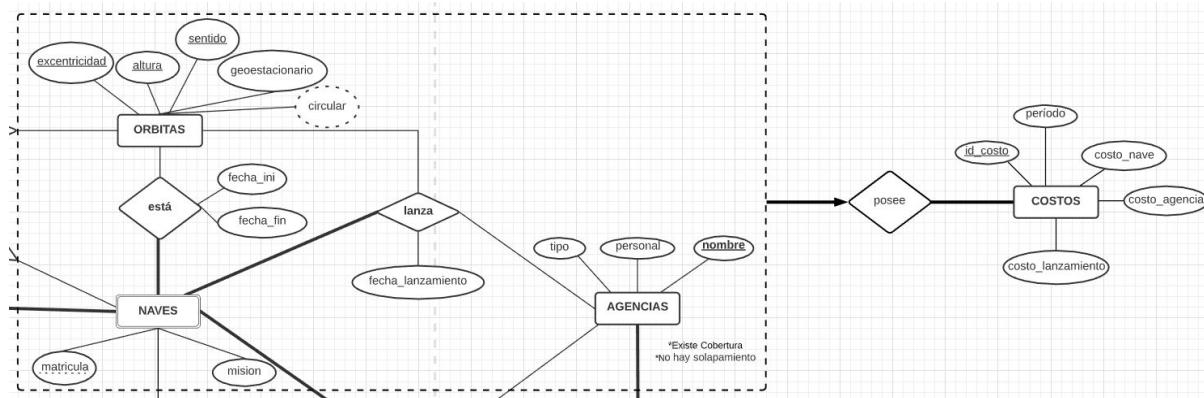
- Al realizar la consulta:

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

126 22:21:39 INSERT INTO compuesto\_por(componente\_contenedor, compon... Error Code: 1644. Error, el contenedor ingresado ya es contenido en ese componente

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

Modifique el modelo ER de manera de poder registrar el costo por lanzamiento, por nave y por agencia.



**MODIFICACIONES:** [Punto 3.sql](#) o en el archivo “Modificación- query’s 3.sql”

#### 3.1.Costo total de lanzamiento mensual por Empresa (Agencia Privada) de una nave en comparación con las del año anterior.

```

SELECT MONTH(C.periodo), SUM(C.costo_agencia)

FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN
Privadas as Priv

ON Priv.nombre = P.nombre_agencia INNER JOIN Financia as F ON F.nombre =
Priv.nombre

WHERE YEAR(C.periodo) LIKE YEAR(curdate())

GROUP BY MONTH(C.periodo);

```

```

2 • SELECT MONTH(C.periodo), SUM(C.costo_agencia)
3   FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN Privadas as Priv
4     ON Priv.nombre = P.nombre_agencia INNER JOIN Financia as F ON F.nombre = Priv.nombre
5   WHERE YEAR(C.periodo) LIKE YEAR(curdate())
6   GROUP BY MONTH(C.periodo);

```

#### 3.2.Crecimiento de Costo total de lanzamiento mensual por clase de nave, es decir, Costo total de lanzamiento totales por clase de nave en comparación con las del mes anterior.

```

SELECT MONTH(C.periodo) as mes, SUM(C.costo_agencia) as total, CN.nombre

```

```

FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN
Privadas as Priv ON Priv.nombre = P.nombre_agencia INNER JOIN Financia as F ON
F.nombre = Priv.nombre, Naves as N INNER JOIN Clase_nave as CN ON CN.prefijo=
N.prefijo

WHERE P.matricula = N.matricula

GROUP BY MONTH(C.periodo), CN.nombre

```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```

1 • use BDA_TPI;
2 • SELECT MONTH(C.periodo) as mes, SUM(C.costo_agencia) as total, CN.nombre
3 FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN Privadas as Priv
4 ON Priv.nombre = P.nombre_agencia INNER JOIN Financia as F ON F.nombre = Priv.nombre, Naves as N INNER JOIN Clase_nave as CN
5 ON CN.prefijo= N.prefijo
6 WHERE P.matricula = N.matricula
7 GROUP BY MONTH(C.periodo), CN.nombre;

```

The results grid shows the output of the query:

mes	total	nombre

### 3.3.Agencias que cubren el 50% del total del Costo total de lanzamiento de todas las naves.

```

WITH Totales as(SELECT P.nombre as Agencia, C.costo_lanza * 100 / T.tot AS
porcentaje

FROM Posee as P INNER JOIN Costo as C ON C.id_costo = P.id_costo CROSJOIN
(SELECT SUM(costo_lanza) AS tot FROM Costo) AS T)

SELECT Agencia, Porcentaje
FROM Totales
WHERE porcentaje > 50;

```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```

1 • use BDA_TPI;
2 • WITH Totales as(SELECT P.nombre_agencia as Agencia, C.costo_lanza * 100 / T.tot AS porcentaje
3 FROM posee as P INNER JOIN costo as C ON C.id_costo = P.id_costo CROSJOIN (SELECT SUM(costo_lanza) AS tot FROM costo) AS T)
4 SELECT Agencia, Porcentaje
5 FROM Totales
6 WHERE porcentaje > 50;

```

The results grid shows the output of the query:

Agencia	porcentaje

### 3.4.Costo total de lanzamiento acumuladas mensuales desde principio de cada año (YTD), por categoría de nave.

```

SELECT YEAR(C.periodo) as anio, SUM(C.costo_lanza) as costo, CN.nombre as categoria
FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN
Privadas as Priv ON Priv.nombre = P.nombre_agencia INNER JOIN
Financia as F ON F.nombre = Priv.nombre, Naves as N INNER JOIN Clase_nave AS CN
ON CN.prefijo = N.prefijo
WHERE N.matricula = P.matricula
GROUP BY YEAR(C.periodo), CN.nombre

```

The screenshot shows a MySQL query editor with the following details:

- Toolbar:** Includes icons for file operations, search, and export.
- Execution History:** Shows the steps of the query execution, numbered 1 through 7.
- Query:**

```

1 • use BDA_TPI;
2 • SELECT YEAR(C.periodo) as anio, SUM(C.costo_lanza) as costo, CN.nombre as categoria
3   FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN
4     Privadas as Priv ON Priv.nombre = P.nombre_agencia INNER JOIN
5     Financia as F ON F.nombre = Priv.nombre, Naves as N INNER JOIN Clase_nave AS CN ON CN.prefijo = N.prefijo
6   WHERE N.matricula = P.matricula
7   GROUP BY YEAR(C.periodo), CN.nombre

```
- Result Grid:** A table with columns 'anio', 'costo', and 'categoria'.

### 3.5. Importe del costo total de lanzamiento realizado por una nave en comparación con el monto total de lanzamiento realizadas por otra que estuvo en la misma órbita el mismo año.

```

WITH costos as(SELECT YEAR(C.periodo) as anio, SUM(C.costo_lanza) as costo,
N.matricula AS Nave
FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN
Privadas as Priv ON Priv.nombre = P.nombre_agencia INNER JOIN Financia as F ON
F.nombre = Priv.nombre , Naves as N
GROUP BY YEAR(C.periodo), N.matricula)

SELECT C1.anio, C1.costo as Costo_Nave_1, C2.costo as Costo_Nave_2
FROM costos as C1 INNER JOIN esta as E1 ON C1.Nave = E1.matricula, Orbitas as
O1, costos as C2 INNER JOIN esta AS E2 ON E2.matricula = C2.Nave, Orbitas as O2
WHERE C1.Nave <> C2.Nave AND O1.excentricidad = O2.excentricidad AND O2.altura =
O1.altura AND O2.sentido = O1.sentido AND C1.anio = C2.anio AND O1.excentricidad =
E1.excentricidad AND O1.altura = E1.altura AND E1.sentido = O1.sentido;

```

```

Query 1
use BDA_TPI;
WITH costos AS(SELECT YEAR(C.periodo) as anio, SUM(C.costo_lanza) as costo, N.matricula AS Nave
FROM costo as C INNER JOIN posee as P ON P.id_costo = C.id_costo INNER JOIN Privadas as Priv ON Priv.nombre = P.nombre_agencia
INNER JOIN Financia as F ON F.nombre = Priv.nombre , Naves as N
GROUP BY YEAR(C.periodo), N.matricula)
SELECT C1.anio, C1.costo as Costo_Nave_1, C2.costo as Costo_Nave_2
FROM costos as C1 INNER JOIN esta as E1 ON C1.Nave = E1.matricula, Orbitas as O1, costos as C2 INNER JOIN
esta AS E2 ON E2.matricula = C2.Nave, Orbitas as O2
WHERE C1.Nave <> C2.Nave AND O1.excentricidad = O2.excentricidad AND O2.altura = O1.altura AND
O2.sentido = O1.sentido AND C1.anio = C2.anio AND O1.excentricidad = E1.excentricidad AND O1.altura = E1.altura AND
E1.sentido = O1.sentido;
12
13

```

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

anio	Costo_Nave_1	Costo_Nave_2

## 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*

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

**1.1.Nombre de las naves que produjo al menos 10 basuras distintas.**

```

WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basuras AS
B GROUP BY B.matricula )
SELECT C.nombre
FROM Clase_nave AS C INNER JOIN Naves 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 Basuras AS B GROUP BY B.matricula )
      -> SELECT C.nombre
      -> FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo INNER JOIN Temp ON Temp.matricula
      -> = N.matricula
      -> WHERE Temp.cant > 9
      ->
      -> ;
+-----+
| nombre
+-----+
| Vechiculos lanzadera
| Vechiculos lanzadera
| Vechiculos lanzadera
| Vechiculos 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 (0.05 sec)

```

```
tiempo → 0.05 sec
create index basura_index ON Basuras(id_basura);
create index nave_index ON Naves(matricula);
```

**ejecutar consulta otra vez**

```
tiempo → 0,03 sec
```

```
mysql> create index nave_index ON Nave(matricula);
ERROR 1146 (42S02): Table 'BDA_TPI.Nave' doesn't exist
mysql> create index basura_index ON Basuras(id_basura);
ERROR 1061 (42000): Duplicate key name 'basura_index'
mysql> create index nave_index ON Naves(matricula);
Query OK, 0 rows affected, 1 warning (0.14 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basuras AS B GROUP BY B.matricula )
      -> SELECT C.nombre
      -> FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo INNER JOIN Temp 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 (0.03 sec)

mysql>
```

**4.2.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 Naves AS N ON N.prefijo = C.prefijo, (SELECT
    B.matricula AS Matricula, COUNT(*) AS cant FROM Basuras AS B GROUP BY
    B.matricula ) AS Temp
 WHERE Temp.matricula = N.matricula AND Temp.cant > 9;
```

```
10 rows in set (0.02 sec)

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, COUNT(*) AS cant FROM Basuras 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 | 47 | BDA_TPI.C.prefijo | 3 | 100.00 | Using index
| 1 | PRIMARY | <derived2> | NULL | ref | <auto_key0> | <auto_key0> | 13 | BDA_TPI.N.matricula | 999 | 33.33 | Using where; Using index
| 2 | DERIVED | B | NULL | index | fk_basuras | fk_basuras | 13 | NULL | 99875 | 100.00 | Using index
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set, 1 warning (0.02 sec)

mysql>
```

```

1 • USE BDA_TP12;
2 • EXPLAIN SELECT C.nombre
3   FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo, (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basuras AS B GROUP BY B
4 WHERE Temp.matricula = N.matricula AND Temp.cant > 9;
5
6
7
8
9

```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	C	HULL	ALL	PRIMARY	HULL	HULL	HULL	3	100.00	HULL
1	PRIMARY	N	HULL	ref	PRIMARY, idx_matricula, nave_index	PRIMARY	47	BDA_TP12.C.prefijo	3	100.00	Using index
1	PRIMARY	<derived2>	HULL	ref	<auto_key0>	<auto_key0>	13	BDA_TP12.N.matricula	998	100.00	HULL
2	DERIVED	B	HULL	index	fk_basuras	fk_basuras	13	HULL	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:

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= 1;
    ELSE
        SET new.circular= 0;
    end IF;
end// 
DELIMITER ;

```

Para comprobar el correcto funcionamiento, lo realizamos las siguientes consultas:

```

INSERT INTO Orbitas(excentricidad, altura, sentido, geostacionario) VALUES
(0,156821.23,'positivo',1);
SELECT * FROM Orbitas WHERE excentricidad=0 AND altura=156821.23 AND
sentido='positivo';

```

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the following SQL code:

```

3  DELIMITER //
4  • CREATE TRIGGER verificar_excent_orbita BEFORE INSERT ON Orbitas FOR EACH ROW
5  begin
6      IF (new.excentricidad = 0) then
7          SET new.circular=1;
8      ELSE
9          SET new.circular=0;
10     end IF;
11  end//;
12  DELIMITER ;
13
14 • INSERT INTO Orbitas(excentricidad, altura, sentido, geostacionario) VALUES (0,156821.23,'positivo',1);
15 • SELECT * FROM Orbitas WHERE excentricidad=0 AND altura=156821.23 AND sentido='positivo';
16

```

Below the code editor is a table named 'Orbitas' with the following data:

#	excentricidad	altura	sentido	geostacionario	circular
0.00	156821.23	positivo	1		1

At the bottom, the 'Action Output' tab displays the following log entries:

#	Time	Action	Message	Duration / Fetch
1	10:39:22	DROP TRIGGER IF EXISTS verificar_excent_orbita	0 row(s) affected	0,418 sec
2	10:39:22	CREATE TRIGGER verificar_excent_orbita BEFORE INSERT ON Orbitas FOR EACH ROW	0 row(s) affected	0,409 sec
3	10:39:23	INSERT INTO Orbitas(excentricidad,altura,sentido,geo...	1 row(s) affected	0,484 sec
4	10:39:23	SELECT * FROM Orbitas WHERE excentricidad=0 AND alt...	1 row(s) returned	0,0011 sec / 0,0000...

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.

```
ALTER TABLE Agencias ADD COLUMN (cant_lanzamientos INT DEFAULT 0);
```

```

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 ;

Para comprobar el correcto funcionamiento, lo probamos con las siguientes consultas:

```

INSERT INTO Lanza VALUES (curdate(),'Apple Inc.', 0.00, 156821.23, 'positivo',
'1613643118');
SELECT * FROM Agencias WHERE nombre='Apple Inc.';
```

*Dado que inicialmente todas las cantidades de lanzamientos están en 0, cuando agreguemos uno nuevo solo mostrará que la agencia realizó un lanzamiento.*

```

2   DELIMITER //
3 •  CREATE TRIGGER actualizar_cantlanz AFTER INSERT ON Lanza
4   FOR EACH ROW
5   begin
6       UPDATE Agencias
7       SET cant_lanzamientos=cant_lanzamientos+1
8       WHERE nombre=new.nombre_agencia;
9   END// 
10  DELIMITER ;
11
12 •  INSERT INTO Lanza VALUES (curdate(),'Apple Inc.',0.00,156821.23,'positivo','1613643118');
13 •  SELECT * FROM Agencias WHERE nombre='Apple Inc.';
14
15

```

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the trigger creation script. Below it is a results grid showing a single row inserted into the 'Lanza' table. At the bottom, there is a log window titled 'Agencias 18' showing the history of actions taken on the database.

#	nombre	personal	tipo	cant_lanzamiento
	Apple Inc.	900956	Consumer Staples	1

Agencias 18

Action Output	#	Time	Action	Message	Duration / Fetch
✓	17	10:50:13	SELECT * FROM Orbitas LIMIT 0, 1000	1000 row(s) returned	0,0013 sec / 0,0009...
✓	18	10:50:13	SELECT * FROM Naves LIMIT 0, 1000	10 row(s) returned	0,0021 sec / 0,0000...
✓	19	10:51:37	DROP TRIGGER IF EXISTS actualizar_cantlanz	0 row(s) affected	0,870 sec
✓	20	10:51:37	CREATE TRIGGER actualizar_cantlanz AFTER INSERT ON ...	0 row(s) affected	0,794 sec
✓	21	10:51:38	INSERT INTO Lanza VALUES (curdate(),'Apple Inc.',0.00,1...)	1 row(s) affected	0,317 sec
✓	22	10:51:39	SELECT * FROM Agencias WHERE nombre='Apple Inc.' Li...	1 row(s) returned	0,00069 sec / 0,000...

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

Para el punto 3 de la primera etapa, 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 ;

```

Para comprobar el correcto funcionamiento, lo probamos con las siguientes consultas:

```

INSERT INTO costo(costo_nave, costo_agencia, costo_lanza) VALUES
(15898.23,8585.26,9685725.61);
SELECT * FROM costo;

```

```

2   DELIMITER //
3 •  CREATE TRIGGER total_costos BEFORE INSERT ON costo
4 FOR EACH ROW
5 BEGIN
6     SET new.costo_total = new.costo_nave + new.costo_lanza + new.costo_agencia;
7 END //
8 DELIMITER ;
9
10 • INSERT INTO costo(costo_nave, costo_agencia, costo_lanza) VALUES (15898.23,8585.26,9685725.61);
11 • SELECT * FROM costo;

```

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the SQL code for creating a trigger. Below it is a results grid showing a single row of data from the 'costo' table. At the bottom, there is an 'Action Output' window displaying the history of actions taken, including the creation of the trigger, its subsequent drop, and the execution of the insert statement.

#	id_costo	costo_nave	costo_agencia	costo_lanza	costo_total	periodo
1	HULL	15898.23	8585.26	9685725.61	9710209.10	HULL

Action Output:

#	Time	Action	Message	Duration / Fetch
25	10:57:33	DESCRIBE costo	6 row(s) returned	0,0037 sec / 0,0000...
26	10:59:20	DROP TRIGGER IF EXISTS total_costos	0 row(s) affected, 1 warning(s); 1360 Trigger does not exist	0,122 sec
27	10:59:20	CREATE TRIGGER total_costos BEFORE INSERT ON costo ...	0 row(s) affected	0,456 sec
28	10:59:21	INSERT INTO costo(costo_nave, costo_agencia, costo_la...)	1 row(s) affected	0,342 sec
29	10:59:21	SELECT * FROM costo LIMIT 0, 1000	1 row(s) returned	0,00063 sec / 0,000...

### Programas almacenados

- 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_prod; /*mostramos variables*/
END //

```

```
DELIMITER ;
```

The screenshot shows the MySQL Workbench interface with a code editor and a results grid. The code editor contains the following SQL script:

```
3 • drop procedure informeNave;
4 CREATE PROCEDURE informeNave (unaMatricula int)
5
6 BEGIN
7     DECLARE cant_orbitas,cant_trip,Basura_prod INT;
8
9     SET cant_orbitas = (
10        SELECT COUNT(*) FROM esta GROUP BY matricula HAVING matricula=unaMatricula);
11
12     SET cant_trip = (
13        SELECT COUNT(*) FROM Tiene WHERE matricula = unaMatricula);
14
15     SET Basura_prod = (
16        SELECT COUNT(*) FROM Basuras WHERE matricula = unaMatricula);
17
18     SELECT cant_orbitas,cant_trip,Basura_prod; /*mostreamos variables*/
19
20 END //
21 DELIMITER ;
22
23 • call informeNave('1118075597')
```

The results grid shows the output of the stored procedure call:

	cant_orbitas	cant_trip	Basura_prod
1	1	3	9937

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
DETERMINISTIC
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 ;
```

#	Time	Action	Message	Duration / Fetch
5	02:05:25	DROP FUNCTION IF EXISTS contarOrbitas	0 row(s) affected	0,535 sec
6	02:05:25	CREATE FUNCTION contarOrbitas(fecha_entrada date, n...	Error Code: 1418. This function has none of DETE...	0,00093 sec
7	02:06:14	DROP FUNCTION IF EXISTS contarOrbitas	0 row(s) affected, 1 warning(s): 1305 FUNCTION BDA_TPI.contarOrbitas does no...	0,108 sec
8	02:06:15	CREATE FUNCTION contarOrbitas(fecha_entrada date, n...	0 row(s) affected	0,643 sec
9	02:06:50	SELECT contarOrbitas('2012-10-10',35689) LIMIT 0, 1000	1 row(s) returned	0,0043 sec / 0,0000...

## 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;
```

#	Time	Action	Message
1	20:53:37	use BDA_TPI;	0 row(s) affected
2	20:53:37	CREATE ROLE IF NOT EXISTS programador, diseñador, administrador;	0 row(s) affected

**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"@"%";
```

The screenshot shows the MySQL Workbench interface. The top window is the 'Query' editor with the following content:

```
use BDA_TPI;
CREATE USER "dis1"@"localhost" IDENTIFIED BY "Dis2020" PASSWORD EXPIRE INTERVAL 180 DAY;
CREATE USER "prog1"@"localhost" IDENTIFIED BY "Prog2020" PASSWORD EXPIRE INTERVAL 180 DAY ACCOUNT LOCK;
CREATE USER "admin1"@"%" IDENTIFIED BY "Admin2020" PASSWORD EXPIRE INTERVAL 180 DAY ACCOUNT LOCK;
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"@"%";
```

The bottom window is the 'Output' pane, which displays the execution log:

Action	Time	Action	Message
18	22:33:11	use BDA_TPI	0 row(s) affected
19	22:33:11	GRANT "programador" TO "prog1"@"localhost"	0 row(s) affected
20	22:33:11	GRANT "diseñador" TO "dis1"@"localhost"	0 row(s) affected
21	22:33:11	GRANT "administrador" TO "admin1"@"%"	0 row(s) affected

**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.

	Lu	Ma	Mi	Ju	Vi	Sa	Do
2:00	A	A	A	A	A	A	A
8:00	A	A	A	A	A	A	A
12:00	C Día 1	C Día 2	C Día 3	C Día 4	C Día 5	C Día 6	
13:00	F	F	F		F		
14:00	A		A		A	A	A
20:00	A	A	A	A	A	A	A

	Lu	Ma	Mi	Ju	Vi	Sa	Do
2:00	A	A	A	A	A	A	A
8:00	A	A	A	A	A	A	A
12:00	C Día 7	C Día 8	C Día 9	C Día 10	C Día 11	C Día 12	
13:00	F	F	F		F		
14:00	A		A		A	A	A
20:00	A	A	A	A	A	A	A

F Full Backup de los archivos de la base de datos y de Archive redo log.  
A Backup de los Archive redo log      C Día # Cinta del día #

\* 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:

- 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.

- 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: USD \$73.97 mensual en Google Cloud SQL para una máquina standard de 3.75GB de RAM.

HDD 1TB : \$5380

SSD 1TB: \$10600

Cinta LTO-7 (15TB): \$11000 - Aprox. \$730/TB

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?**

En el caso de fallar a las 11 AM, el último backup incremental fue realizado a las 8 AM.

En esta situación tenemos la posibilidad de utilizar un backup diferencial con los datos que encontramos en: el full backup del miércoles + backups incrementales del miércoles y jueves existentes hasta el momento de la falla.

El plan posterior en caso de que fallara alguna de estas restauraciones, sería cargar los datos de la cinta del día 9, sin embargo esto provocaría perder las transacciones realizadas en las últimas 23 hs.

- 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 financian 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" : 87,
              "ciudad" : "xxxxxx",
              "nivel estudio" : "xxxxxxxxxx",
              "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 tendrá el esquema anterior.

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

Generamos un JsonSchema a partir del ejemplo brindado por la consigna.

*Los datos que debemos ingresar están en un script .sql dentro del siguiente enlace: [PuntoJSON.sql](#) o en el archivo “PuntoJSON.sql” dentro de la carpeta entregada.*

## 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 -> '$."Principal Accionista".edad') as "Edad Promedio"
from Empresas
where (accionista -> '$."Principal Accionista"."nivel estudio"') LIKE
'%universitario%';
mysql> SELECT avg(accionista -> '$."Principal Accionista".edad') as "Edad Promedio"
-> from Empresas
-> where (accionista -> '$."Principal Accionista"."nivel estudio"') LIKE '%universitario%';
+-----+
| Edad Promedio |
+-----+
|      38.4 |
+-----+
1 row in set (0.00 sec)

mysql> █
```

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 -> '$."Principal
Accionista".aspiraciones')) FROM Empresas) DO
        INSERT INTO temp SELECT
        JSON_UNQUOTE(JSON_EXTRACT(Accionista,CONCAT('$.Principal
Accionista".aspiraciones[',i,'].mundos'))) as mundos FROM Empresas;
        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();
```

```
# mundos
HIP 13044b

Result 57 x
Action Output ▾
# Time Action Message
1 01:56:38 DROP PROCEDURE IF EXISTS mundo_mas_aspirado 0 row(s) affected
2 01:56:38 CREATE PROCEDURE mundo_mas_aspirado () #RETURN... 0 row(s) affected
3 01:56:39 CALL mundo_mas_aspirado() 1 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 -> '$."Principal
Accionista".ciudad' as Ciudad
FROM Empresas
WHERE Accionista -> '$."Principal Accionista".viajes[*].tipo' LIKE
'%Cientifico%';
```

#	Nombre	Edad	Ciudad
1	"Branu Cornelissen"	50	"Walakeri"
2	"Minnaminie Rushworth"	26	"Pawitan"
3	"Jacquie Roads"	31	"Mino"
4	"Robbi Amer"	38	"Kassala"
5	"Tommie Champe"	44	"Haciqabul"
6	"Ginger Mityakov"	49	"Balesari"
7	"Sydelle Stanex"	32	"Krásná Lipa"
8	"Jacenta Tinwell"	57	"Bornu Yassu"
9	"Holly Wickett"	52	"Kými"
10	"Lesley Covotti"	30	"Guanyinsi"
11	"Christyna Adolthine"	51	"Goba"
12	"Shepard Clementel"	36	"Peterhol"
13	"Emiline Orford"	37	"Oktyabr'skiy"
14	"Kristos Birdfield"	41	"Hammâm ..."

Result 15 x

Action Output ▾

#	Time	Action	Message	Duration / Fetch
1	09:55:12	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	14 row(s) returned	0,0013 sec / 0,0000...

4. Listar los nombres, edades y ciudad de residencia de quienes visitan 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 Empresas
WHERE Accionista -> '$."Principal Accionista".viajes[*].tipo' NOT LIKE
'[%"Cientifico%"]' AND Accionista -> '$."Principal Accionista".viajes[*].tipo'
NOT LIKE '[%"Militar%"]';
```

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

#	Nombre	Edad	Ciudad	Tipo
	"Rosabella Hawks"	35	"Nginokrajan"	["Espia"]
	"Shannon O Sirin"	48	"Siepraw"	["Espia"]
	"Taffy Christauffour"	43	"Villanova"	["Espia"]

Action Output ▾

#	Time	Action	Message	Duration / Fetch
4	10:09:18	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	0 row(s) returned	0,00074 sec / 0,000...
5	10:09:24	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	0 row(s) returned	0,00082 sec / 0,000...
6	10:09:34	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	3 row(s) returned	0,00097 sec / 0,000...
7	10:09:48	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	0 row(s) returned	0,00081 sec / 0,000...
8	10:09:50	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	3 row(s) returned	0,0014 sec / 0,000...
9	10:10:33	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	3 row(s) returned	0,0013 sec / 0,000...

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 Empresas
WHERE Accionista -> '$."Principal Accionista".viajes[*].mundos' LIKE '%Venus%'
AND Accionista -> '$."Principal Accionista".aspiraciones[*].mundos' LIKE
'%Saturno%';
```

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

#	Nombre	redes_sociales
---	--------	----------------

Action Output ▾

#	Time	Action	Message	Duration / Fetch
1	10:13:32	SELECT Accionista ->'\$."Principal Accionista".nombre' a...	0 row(s) returned	0,0010 sec / 0,000...

Debido a que la consulta inicialmente no retorna ningnú 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 Empresas SET Accionista='{"Principal
Accionista":{"dni":37992248,"nombre":"Shepard
Clementet","edad":36,"ciudad":"Peterhof","nivel
estudio":"terciario","email":"sclementetd@hc360.com","redes
sociales":[{"instagram":"hmccgilben0","twitter":"rwipfler0","facebook":"rthomtson
0"}],"accionistas secundarios":[{"accionista":{"dni":25849413,"nombre":"Juliana
Feldberg","edad":40,"ciudad":"Zhuxi Chengguanzen","nivel
estudio":"terciario","email":"jfheldberg0@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":"sclericooates0","twitter":"rmeddick0","facebook":"racome
0"}]}}, {"accionista":{"dni":36536920,"nombre":"Beverie
Affuso","edad":28,"ciudad":"Wäling","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"}, {"mundos": "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:

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with 'Result Grid', 'Filter Rows' (with a search icon), and 'Export' options. Below the toolbar is a results grid titled 'redes\_sociales'. The grid has two columns: '# Nombre' and 'redes\_sociales'. A single row is displayed: "# Nombre" is 'Shepard Clementet' and 'redes\_sociales' is a JSON array containing [{"twitter": "rwipfler0", "facebook": "rthomtson0", "instagram": "hmcgilben0"}].

Below the results grid is a 'Result 28' tab. Underneath it is a 'Action Output' section with a table. The table has columns: '#', 'Time', 'Action', 'Message', and 'Duration / Fetch'. It contains four rows:

#	Time	Action	Message	Duration / Fetch
1	10:13:32	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	0 row(s) returned	0,0010 sec / 0,0000...
2	10:14:25	SELECT * FROM Empresas LIMIT 0, 1000	20 row(s) returned	0,0015 sec / 0,0002...
3	10:17:04	UPDATE Empresas SET Accionista=(\$."Principal Accionista".nomb...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0,348 sec
4	10:17:04	SELECT Accionista -> '\$."Principal Accionista".nombre' a...	1 row(s) returned	0,0011 sec / 0,0000...