

**BASES DE DATOS APLICADAS**  
**TRABAJO PRÁCTICO INTEGRADOR**  
**Primera Entrega**

**GRUPO N° 14 (Grupo 4 unificado con grupo 14)**

**ALUMNOS:**

- ACEVEDO, FERNANDO
- GETZEL, MARTIN
- LUCAS, DANIA
- OJEDA, MARIA JOSE
- VALDES, MANUEL

**PROFESORES:**

- ING. FANTIN, ANDRES
- ING. FERNANDEZ, JUAN CARLOS

**CARRERA:** INGENIERÍA EN SISTEMAS DE INFORMACIÓN

**CURSO ACADÉMICO:** SEGUNDO CUATRIMESTRE – 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 a las siguientes consignas:**

#### **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:

		8.0	5.7	5.6
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 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	*	*	*
<a href="#">Yum Repo</a>		*	*	*
<a href="#">APT Repo</a>		*	*	*
<a href="#">SUSE Repo</a>		*	*	*

Además, podemos realizar la instalación 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.

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

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.**

- a. **Numéricos**

- i. TINYINT con signo: 127, sin signo: 255.
    - ii. SMALLINT con signo: 32767, sin signo: 65535
    - iii. MEDIUMINT con signo: 8388607, sin signo: 16777215
    - iv. INT con signo: 2147483647, sin signo: 4294967295
    - v. BIGINT con signo:  $2^{63}-1$ , sin signo:  $2^{64}-1$
    - vi. DECIMAL: 65 dígitos y 30 decimales.
    - vii. FLOAT: Los valores válidos van desde  $-3.402823466E+38$  a  $-1.175494351E-38$ , 0 y desde  $1.175494351E-38$  a  $3.402823466E+38$ .
    - viii. DOUBLE: Desde  $-1.7976931348623157E+308$  a  $-2.2250738585072014E-308$ , 0 y desde  $2.2250738585072014E-308$  a  $1.7976931348623157E+308$ .
    - ix. DATE: '9999-12-31'.
    - x. DATETIME: 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos.
    - xi. CHAR, VARCHAR, TinyText y TinyBlob: 255.
    - xii. BLOB Y TEXT: 65535.
    - xiii. MEDIUM BLOB, MEDIUM TEXT: 16.777.215.
    - xiv. LONGBLOB, LONGTEXT: 4.294.967.295.
    - xv. ENUM: 65535 valores distintos.
    - xvi. 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	si	si
Desarrolladores	MongoDB Inc.	Oracle Corporation
SO	Multiplataforma	Multiplataforma
Lenguaje query	Javascript	SQL
Mapa reducido	si	no
Convercion 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:**

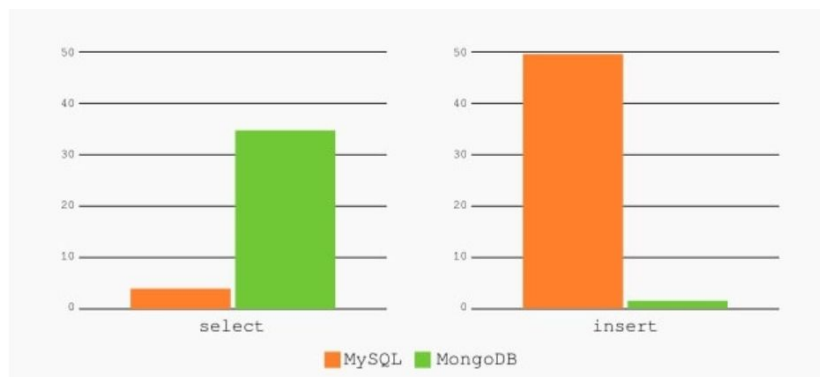
- Soporta transacciones atómicas
- Soporta el uso de JOIN
- Seguridad basada en privilegios con uso de contraseña
- Es una tecnología madura

#### **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 (0.33 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**

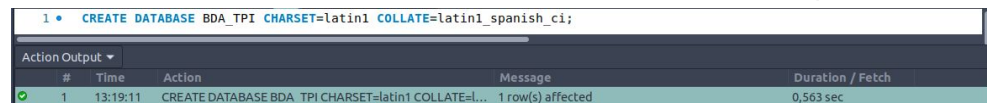
esta cuenta (no con *root*).

```
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;
```



1 • 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=l...	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 como [TPI\\_versionfinal.sql](#).

### **1. Consultas**

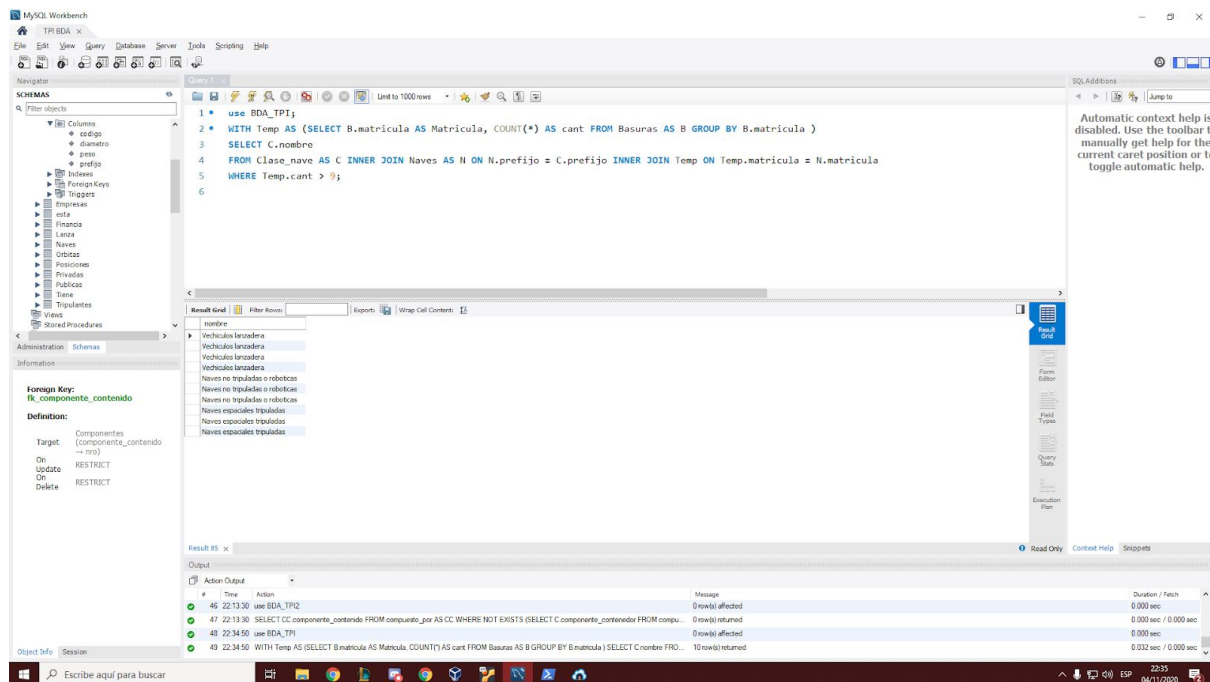
#### **1.1.Nombre de los naves que produjo al menos 10 basuras distintas.**

```
WITH Temp AS (SELECT B.matricula AS Matricula, COUNT(*) AS cant FROM Basuras AS B  
GROUP BY B.matricula )
```

```
SELECT C.nombre
```

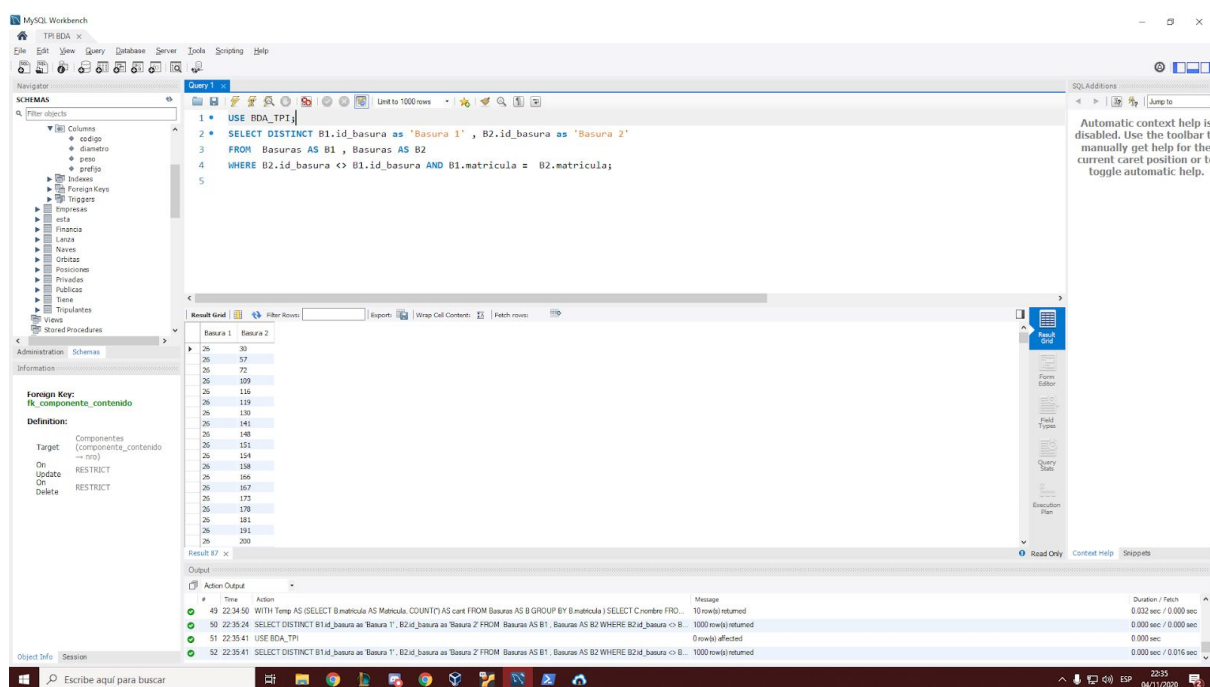
```
FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo INNER JOIN  
Temp ON Temp.matricula = N.matricula
```

```
WHERE Temp.cant > 9;
```



## 1.2. 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.

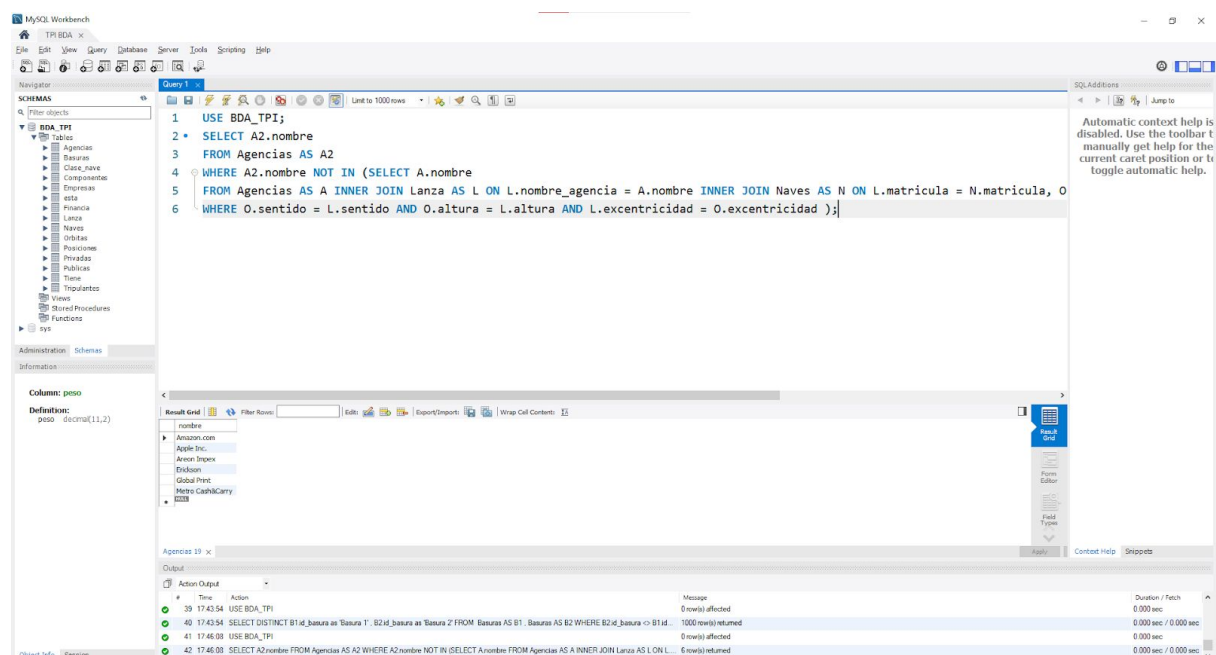
```
SELECT A2.nombre
```

```
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 );
```



### 1.4. Listar las órbitas en las cuales estuvieron todas las naves.

```
SELECT E.excentricidad, E.altura, E.sentido
```

```
FROM esta AS E
```

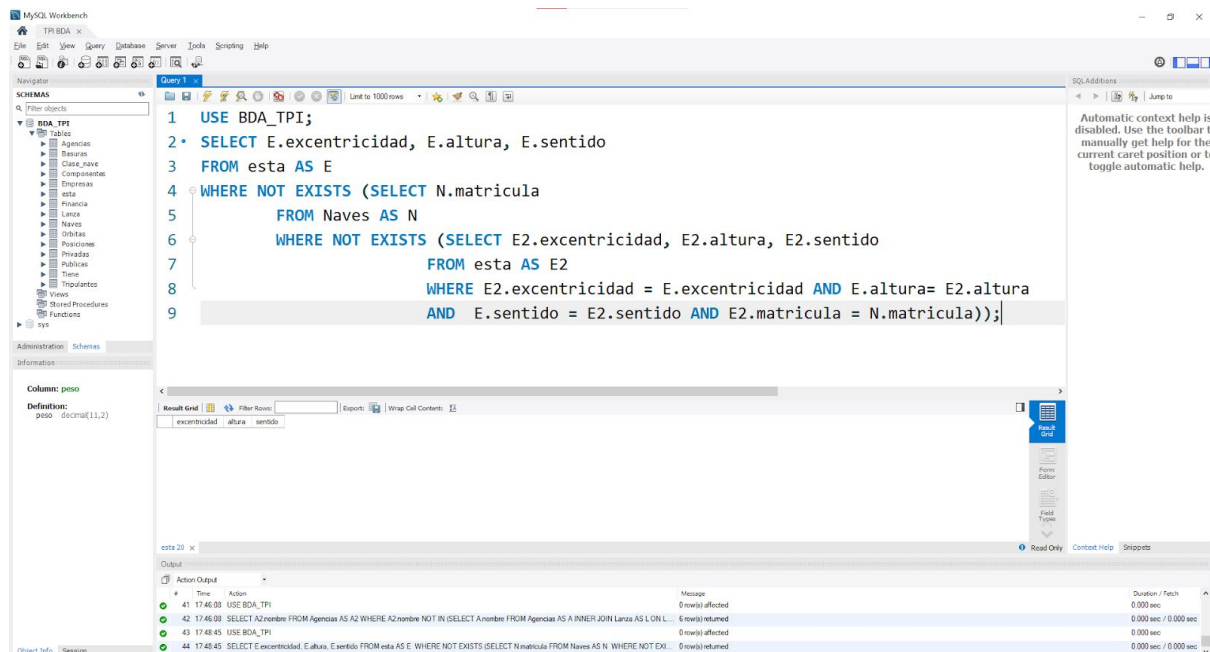
```
WHERE NOT EXISTS (SELECT N.matricula
```

```
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));
```



**1.5. Liste el nombre de todas las empresas publicas y su agencia que supervisan al menos dos empresas privadas distintas.**

WITH supervisa AS (SELECT nombre\_publica, COUNT(\*) AS cant

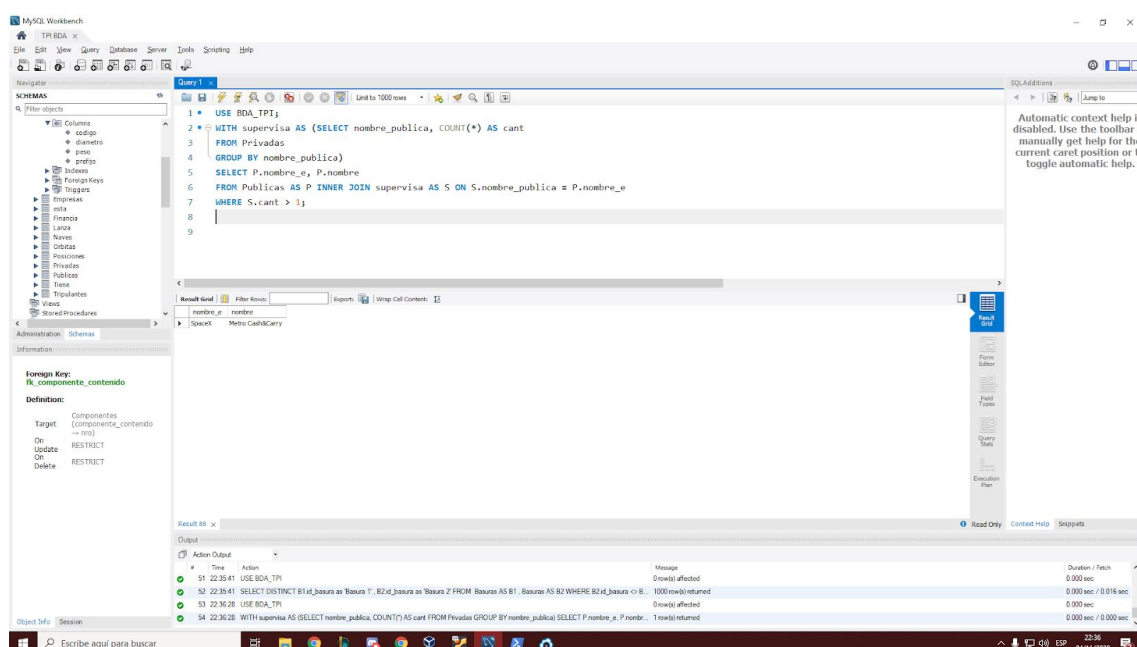
FROM Privadas

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;





## 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 Componentes
```

```
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);
```

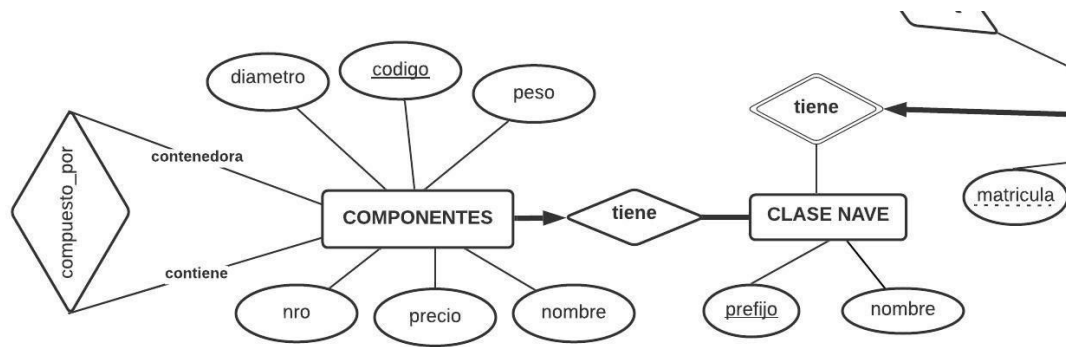
```
create index componente_contenido_index ON compuesto_por(componente_contenido);
```

```
ALTER TABLE compuesto_por
```

```
add CONSTRAINT fk_componente_contenedor FOREIGN KEY(componente_contenedor) REFERENCES  
Componentes(nro),
```

```
add CONSTRAINT fk_componente_contenido FOREIGN KEY(componente_contenido) REFERENCES  
Componentes(nro);
```

### DER MODIFICADO:



**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 Componentes SET nro = 1, nombre = 'Conducto Dual Interno' WHERE codigo = 1;
```

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

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

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

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

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

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

```
UPDATE Componentes SET nro = 8, nombre = 'Giroscopio Coaxial Automatico' WHERE codigo = 8;
```

```
UPDATE Componentes SET nro = 9, nombre = 'Junta Holografica Principal' WHERE codigo = 9;
```

```
UPDATE Componentes SET nro = 10, nombre = 'Inyector Holografico' WHERE codigo = 10;
```

```
UPDATE Componentes SET nro = 11, nombre = 'Interruptor Subespacial De Emergencia' WHERE codigo = 11;
```

```
UPDATE Componentes SET nro = 12, nombre = 'Reciclador Coaxial' WHERE codigo = 12;
```

```
UPDATE Componentes SET nro = 13, nombre = 'Giroscopio Dual Automatico' WHERE codigo = 13;
```

```
UPDATE Componentes SET nro = 14, nombre = 'Conmutador Holografico Plasmatico' WHERE codigo = 14;
```

```
UPDATE Componentes SET nro = 15, nombre = 'Inyector Trifasico Rotatorio' WHERE codigo = 15;
```

```
UPDATE Componentes 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);
```

---

## **2.1.Nombre de los componentes que forman parte de al menos dos componentes distintos.**

```
WITH compuesto AS (SELECT CC.componente_contenido as cont
```

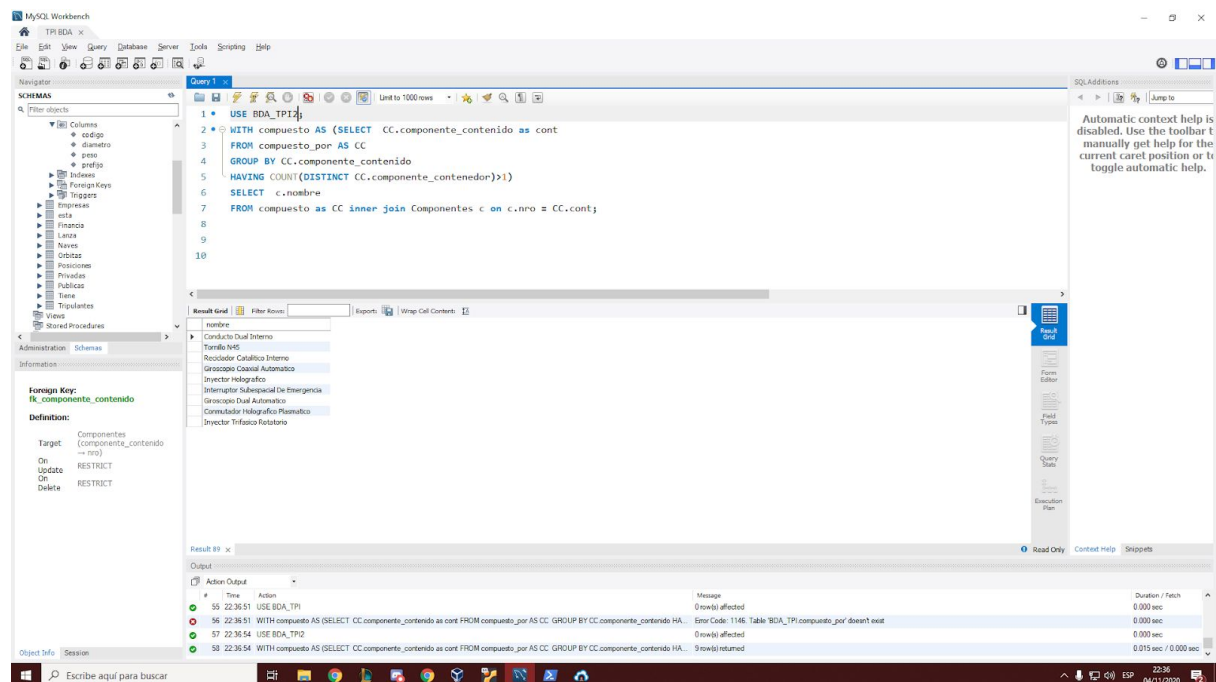
```
FROM compuesto_por AS CC
```

```
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;



**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.**

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

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

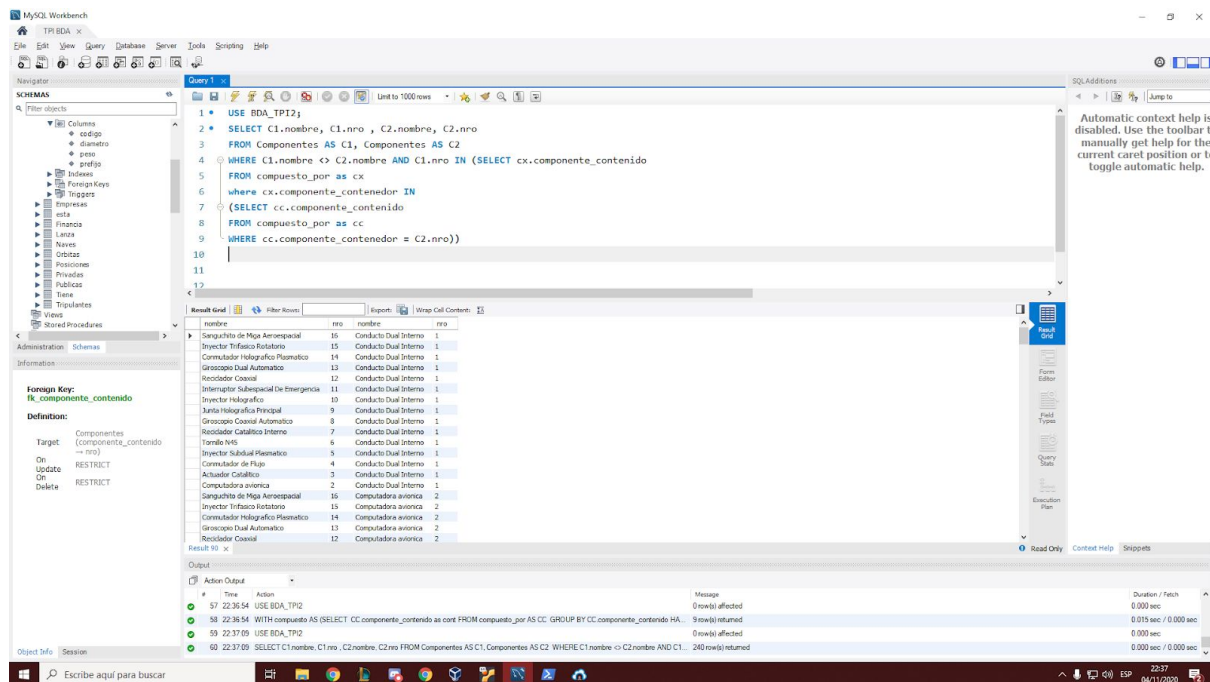
FROM compuesto\_por as cx

where cx.componente\_contenedor IN

(SELECT cc.componente\_contenido

FROM compuesto\_por as cc

WHERE cc.componente\_contenedor = C2.nro))



**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.**

supongo que se refiere a mostrar aquellos sub\_componentes (articulos) que estan en TODOS los componentes padres.

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 ;

```

The screenshot shows a SQL IDE interface. The top window, titled 'Query 1', contains the following SQL code:

```

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

```

The bottom window, titled 'Output', shows the execution results of the stored procedure. It has a table with columns: #, Time, Action, and Message.

#	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.

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

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

**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 q 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'
```

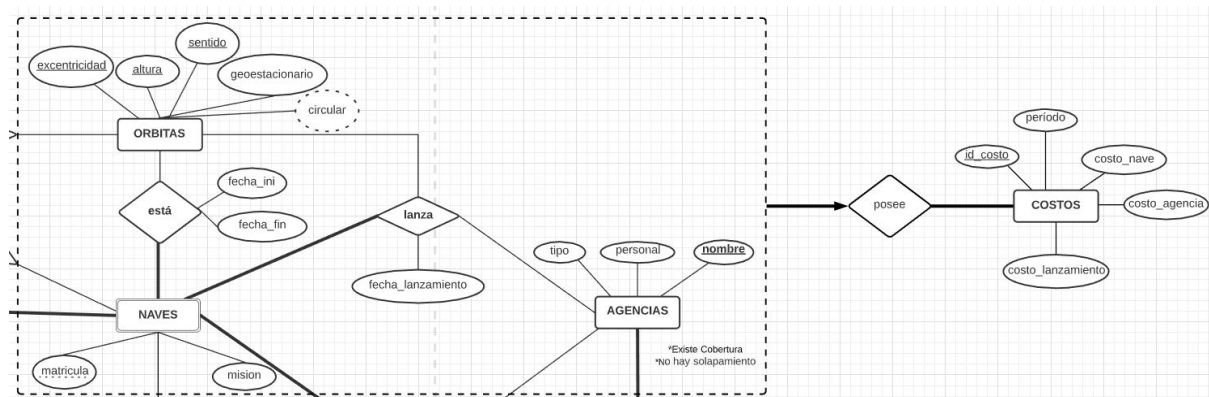
```
    end IF;
```

```
end
```

```
//
```

### 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.*



*agreguemos <costo> atributos : costo\_nave , costo\_agencia, costo\_lanza, periodo,*

*agregar a lanza -- osea encerra todo lo de lanza en un cuadrado*

*// vinculo*

```
CREATE TABLE posee(  
    id_costo int Auto_increment,  
    fecha_lanzamiento DATETIME NOT NULL,  
    nombre_agencia varchar(45),  
    excentricidad decimal(9,2),  
    altura decimal(9,2),  
    sentido enum('positivo', 'negativo'),  
    matricula varchar(10),  
  
    CONSTRAINT fk_lanza_nombre FOREIGN KEY(nombre_agencia) REFERENCES  
    Agencias(nombre),  
  
    CONSTRAINT fk_lanza_excentricidad FOREIGN KEY(excentricidad) REFERENCES  
    Orbitas(excentricidad),  
  
    CONSTRAINT fk_lanza_altura FOREIGN KEY(altura) REFERENCES Orbitas(altura),  
  
    CONSTRAINT fk_lanza_sentido FOREIGN KEY(sentido) REFERENCES Orbitas(sentido),  
  
    CONSTRAINT fk_lanza_matricula FOREIGN KEY(matricula) REFERENCES Naves(matricula));
```

*//tabla*

```
CREATE TABLE costo(  
    id_costo int Auto_increment,  
    costo_nave decimal(9,2) NOT NULL,  
    costo_agencia decimal(9,2) NOT NULL,  
    costo_lanza decimal(9,2) NOT NULL,  
    costo_total decimal(12,2) DEFAULT NULL,  
    periodo DATE,  
  
    CONSTRAINT pk_costo PRIMARY KEY(id_costo)  
);
```



*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 ;*

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

**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.**

WITH totales AS (SELECT C.prefijo as prefijo, Co.periodo as periodo,  
Co.costo\_total as costo\_total

FROM Clase\_nave AS C, Costo as Co, posee as P

WHERE C.matricula = P.matricula AND P.id\_costo = Co.id\_costo

GROUP BY C.prefijo, Co.periodo

ORDER BY DATE(Co.periodo) DESC)

SELECT T.prefijo, YEAR(T.periodo) AS anio, MONTH(T.periodo) AS Mes, T.costo\_total

FROM totales AS T

GROUP BY YEAR(T.periodo), MONTH(T.periodo)

HAVING

ORDER BY

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

WITH todas\_naves AS(

SELECT L.nombre\_agencia

FROM Lanza AS L INNER JOIN

WHERE)

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

**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 orbita el mismo año.**

#### 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 volver a ejecutar la consulta 1.1. Ver si hay diferencia en los tiempos de ejecución con respecto a la primera.**

**1.1.Nombre de los 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 |
+-----+
| 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 (0.05 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 (42502): 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
a )
-> 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).

<https://dev.mysql.com/doc/refman/8.0/en/using-explain.html>

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>
```

MySQL Workbench

TP1 BDA x

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

Schemas

- Columns
  - cedepi
  - diametro
  - peso
  - prefijo
- Indexes
- Foreign Keys
- Triggers
- Empresas
- esta
- financia
- Lanza
- Naves
- Orbitas
- Posicones
- Privadas
- Publicas
- Tarea
- Triplantes
- Views
- Stored Procedures

Administration Schemas

Information

Foreign Key:  
**fk\_componente\_contenido**

Definition:

Target: Componentes (componente\_contenido → mro)

On Update: RESTRICT

On Delete: RESTRICT

Query 1

```
1 USE BDA_TP12;  
2  
3 EXPLAIN SELECT C.noembre  
4 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 B  
5 WHERE Temp.matricula = N.matricula AND Temp.cant > 9;  
6  
7  
8  
9
```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	C		ALL	PRIMARY				3	100.00	Using index
1	PRIMARY	N		ref	PRIMARY_idx_matricula_nave_index	PRIMARY	47	BDA_TP12.C.prefijo	3	100.00	Using index
1	PRIMARY	clases2		ref	<auto key>	<auto key>	13	BDA_TP12.N.matricula	998	100.00	Using index
2	DERIVED	B		index	fk_basuras	fk_basuras	13		99879	100.00	Using index

Result 91

Output

Action Output

#	Time	Action	Message	Duration / Fetch
59	22:37:09	USE BDA_TP12	0 rows(s) affected	0.000 sec
60	22:37:09	SELECT C1.noembre, C1.no_ , C2.noembre, C2.no FROM Componentes AS C1, Componentes AS C2 WHERE C1.noembre <> C2.noembre AND C1...	240 rows(s) returned	0.000 sec / 0.000 sec
61	22:37:50	USE BDA_TP12	0 rows(s) affected	0.000 sec
62	22:37:50	EXPLAIN SELECT C.noembre FROM Clase_nave AS C INNER JOIN Naves AS N ON N.prefijo = C.prefijo, (SELECT B.matricula AS Matricula, C...	4 rows(s) returned	0.000 sec / 0.000 sec

Object Info Session

Write here to search

22:37 04/11/2020

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.