

UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Resistencia



Base de Datos Aplicadas

Trabajo Práctico Integrador - Parques Naturales

Profesores:

- **Juan Carlos Fernández**
- **Andres Fantin**

Grupo N°4:

- **Acosta, Gastón**
- **Cornaló, Laura**
- **Gonzalez, Mariano**
- **Núñez, Lucas**
- **Ruiz, Franco**
- **Sosa, Valentina**

PRIMERA PARTE

Administración de un SGBD

1. Indicar cantidad de memoria RAM mínima y recomendada.

Cantidad mínima de RAM es 2GB y recomendable de RAM : 4 GB

2. Ídem para el espacio en disco:

Utilizaremos MySQL , el fabricante no dispone de requerimientos mínimos, reservaremos una cantidad minima de disco : 20GB.

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

Mysql es compatible y trabaja perfectamente con cualquier sistema operativo, limitado a una arquitectura de 32/64 bit.

4. Indicar cuáles son las alternativas para la instalación.

		8.0	5.7	5.6
Operating System	Architecture			
Oracle Linux / Red Hat / CentOS				
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	*	*	*
Oracle Solaris				
Solaris 11 (Update 4+)	SPARC_64	*	*	*
Canonical				
Ubuntu 20.04 LTS	x86_64	*		
Ubuntu 18.04 LTS	x86_32, x86_64	*	*	
Ubuntu 16.04 LTS	x86_32, x86_64	*	*	
SUSE				
SUSE Enterprise Linux 15 / OpenSUSE 15	x86_64	*		
SUSE Enterprise Linux 12 (12.4+)	x86_64	*	*	*
Debian				
Debian GNU/Linux 10	x86_64	*	*	
Debian GNU/Linux 9	x86_32, x86_64	*	*	*
Microsoft Windows Server				
Microsoft Windows 2019 Server	x86_64	*		
Microsoft Windows 2016 Server	x86_64	*	*	*
Microsoft Windows 2012 Server R2	x86_64	*	*	*
Microsoft Windows				
Microsoft Windows 10	x86_64	*	*	
Apple				
macOS 10.15	x86_64	*		
FreeBSD				
FreeBSD 12	x86_64	*		
Various Linux				
Generic Linux (tar format)	x86_32, x86_64, glibc 2.12, libstdc++ 4.4	*	*	*
Yum Repo		*	*	*
APT Repo		*	*	*
SUSE Repo		*	*	*

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

Mysql, tiene soporte para discos sin formato. Las particiones de disco sin formato se pueden utilizar como archivos de datos del espacio de tabla del sistema. Esta técnica habilita E / S sin búfer en Windows y algunos sistemas Linux y Unix sin sobrecarga del sistema de archivos.

Restricciones : cuando se utilice una partición de disco sin formato hay que asegurarse de que el ID de usuario que ejecuta el servidor MySQL tenga privilegios de lectura y escritura para esa partición.

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

Para ver cuánto mide el espacio de tablas, creamos una nueva base de datos, y accedemos al directorio donde estaría ubicada. Con `du -sh *` podemos ver cuál es el tamaño de los archivos y directorios, por lo que veremos cuánto pesa el directorio

Creamos la base de datos árbol, la cual está vacía.

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| arbol    |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0,02 sec)

mysql> use arbol;
Database changed
mysql> show tables;
Empty set (0,01 sec)
```

```
administrador@srv-bbdd:/var/lib/mysql$ du -sh arbol
4,0K    arbol
administrador@srv-bbdd:/var/lib/mysql$
```

La base de datos pesa 4,0 KB

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

```
mysql> SELECT table_schema AS "Base de datos",
-> ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Tamaño (MB)"
-> FROM information_schema.TABLES
-> GROUP BY table_schema;
+-----+-----+
| Base de datos | Tamaño (MB) |
+-----+-----+
| arbol        | 1.02        |
| information_schema | 0.00        |
| mysql        | 2.59        |
| performance_schema | 0.00        |
| sys          | 0.02        |
+-----+-----+
5 rows in set (0,08 sec)
```

Creamos 1 MB de datos.

```
administrador@srv-bbdd:/var/lib/mysql$ du -sh arbol
4,0M    arbol
```

Ahora podemos ver que el espacio de tabla de esta bases de datos es de 4 MB

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

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

Los ejecutables :

-mysqld:/usr/sbin/

-mysql,mysqldump,mysqlbinlog;/usr/bin

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

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

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

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

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

MySQL Workbench es una herramienta gráfica para trabajar con servidores y bases de datos MySQL. Soporta servidores MySQL de versiones 5.6 en adelante. No soporta versiones 4.x de servidores MySQL. Es una aplicación nativa, que se desarrolló para ser utilizada en Windows, Ubuntu, macOS, entre otros SO.

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

Una tabla es el lugar donde se almacenan los 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.

Los índices son un grupo de datos vinculado a una o varias columnas que almacena en una relacion entre el contenido y la fila en la que se encuentra. Con esto se agilizan las

busquedas en una tabla al evitar que MySQL tenga que correr toda la tabla para obtener datos solicitados.

Las vistas en MySQL son tablas virtuales, es decir, tablas que no guardan ningun dato dentro de ellas.Solo muestran los datos que están almacenados en otras tablas.

Los triggers o disparadores de MySQL son una serie de reglas predefinidas que están asociadas a una tabla . Estas reglas permiten la ejecucion de una serie de instrucciones cuando se producen ciertos eventos como pueden ser la insercion de un nuevo registro,la actualizacion o el borrado de los datos de una tabla.

12. Enumerar las capacidades de su SGBD:

1. Tamaño máximo de espacio de tablas:

En InnoDB tenemos las siguientes limitaciones:

Tamaño de página de InnoDB	Tamaño máximo del espacio de tabla
4 KB	16 TB
8 KB	32 TB
16 KB	64 TB
32 KB	128 TB
64 KB	256 TB

2. **Cantidad máxima de tablas, índices, stored procedures, vistas:** El número máximo de tablas soportado es de 4 mil millones, el de índices multicolumna se limita a 16 columnas y una longitud máxima de clave a 1000bytes
3. **Cantidad máxima de columnas por tabla:** El número máximo de columnas por tabla que se puede tener es de 4096.
4. **Longitud máxima de fila:** La representacion interna de una fila no puede ocupar mas de 65535 bytes.
5. **Tamaños máximos de los tipos de datos que soporta:**
 - 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.

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

El sistema operativo tiene incidencia sobre el tamaño de los archivos a utilizar:

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

14. Enumerar los procesos del motor ejecutándose en el sistema operativo, sus nombres de imágenes, tamaño en memoria y funciones de cada uno. Indicar a su parecer los dos más importantes.

Utilizamos este comando de Linux para ver los detalles de los procesos ejecutándose:

```
administrador@srv-bbdd:~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	1.5	0.6	103392	12916	?	Ss	18:21	0:09	/sbin/init maybe-ubiquity
root	2	0.0	0.0	0	0	?	S	18:21	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	18:21	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	18:21	0:00	[rcu_par_gp]
root	6	0.0	0.0	0	0	?	I<	18:21	0:00	[kworker/0:0H-kblockd]

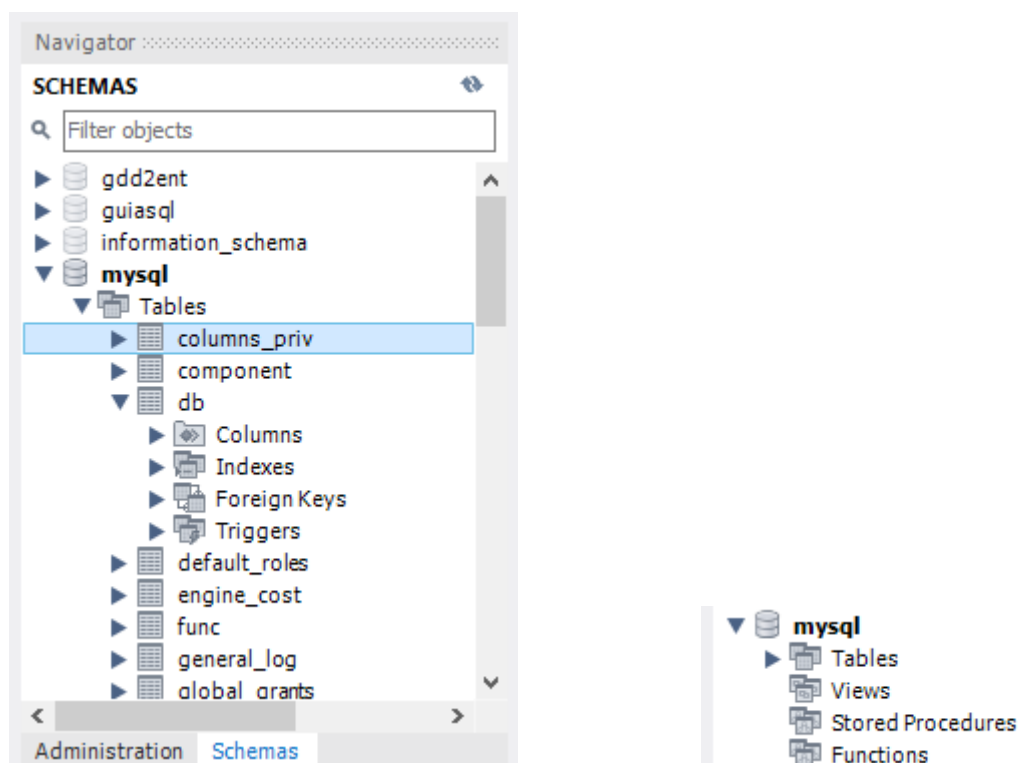
...

```
mysql      766  0.4 19.5 1291016 398364 ?      Ssl 18:21  0:02 /usr/sbin/mysqld
```

Tenemos ejecutando el daemon de mysql usando esa cantidad de recursos de cpu y porcentaje de memoria.

15. Indicar cuántos nodos (en la estructura de árbol en la que se encuentran los objetos de la instancia manejada) se muestran en el programa administrador del SGBD, y la función de cada uno. No incluye nodos relacionados con los datos de bases de datos de usuario, sólo los nodos que corresponden a funcionalidad predeterminada.

En la siguiente figura podemos encontrar la estructura de arbol de nodos de MySQL Workbench:



En la misma podemos encontrar en primer lugar a los esquemas, que serían las bases de datos en sí mismas, de cada una de ellas se desprenden los nodos para las tablas, vistas, procedimientos almacenados y funciones. De cada una de las tablas se desprenden las columnas, índices, claves foráneas y triggers asociados a las mismas. Con esto sumamos una cantidad de 9 nodos organizados en estructura de árbol que corresponden a una funcionalidad predeterminada.

16. ¿Cuántas bases de datos de sistema tiene su SGBD? Enumerar y explicar sucintamente cuál es la función de cada una.

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

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

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

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

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

La base de datos puede contener 9 tipos distintos de objetos:

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

- Trigger
- View

18. 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
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 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 'contradmi';
CREATE USER 'OPERADOR'@'%' IDENTIFIED BY 'contraope';
SELECT User from mysql.user;
```

4 • **SELECT User from mysql.user;**
5

<

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

User
ADMIN
OPERADOR
trabajador
Jorge
mysql.infoschema
mysql.session
mysql.sys
root

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;
show grants for ADMIN;
```

7 • show grants for ADMIN;

Result Grid	Filter Rows:	Exports	Wrap Cell Contents
Grants for ADMIN@%			
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CR...			
GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ADMIN,BACKUP_ADMIN,BINLOG_ADMIN,BINLOG_ENCRYPTION_ADMIN,CLONE_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,FLUSH_OPTIMIZER_COSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOURCES,GROUP_REPLICA...			

3. Crear una base de datos para desarrollar el trabajo práctico integrador. Como único requisito debe llamarse BDA-TPI. Escoger el resto de los parámetros usando su criterio, documentar cada paso con una copia de pantalla del asistente de creación o el comando utilizado.

```
CREATE DATABASE IF NOT EXISTS BDA_TPI
CHARACTER SET=latin1
COLLATE=latin1_spanish_ci;
```

Query 1 SQL File 3*

1		CREATE DATABASE IF NOT EXISTS BDA_TPI	
2		CHARACTER SET=latin1	
3		COLLATE=latin1_spanish_ci;	

Output

#	Time	Action	Message
1	15:28:25	CREATE DATABASE IF NOT EXISTS BDA_TPI CHARACTER SET=latin1 COLLATE=latin1_spanish_ci	1 row(s) affected

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.

Damos al usuario OPERADOS todos los permisos con GRANT ALL, pero no incluimos la gestión de permisos, luego quitamos la creación de usuarios.

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

```

13 • GRANT ALL ON *.* TO 'OPERADOR'@'%';
14 • REVOKE CREATE USER ON *.* FROM 'OPERADOR'@'%';

```

Output		
Action Output		
#	Time	Action
✓ 1	15:33:39	REVOKE CREATE USER ON *.* FROM 'OPERADOR'@'%'

CONSULTAS ANEXO IV

1. Consultas

-- 1.1 Nombre de los parques en donde residen al menos 10 especies distintas.

```

SELECT nombreParque
FROM residen
GROUP BY nombreParque
HAVING count(distinct nomCientifico)>=10;

```

-- 1.2 Listar los pares (animal 1, animal 2) tales que animal 1 se alimenta por un animal o vegetal que también alimenta al animal 2.

```

SELECT a.nomDepredador AS Depredador1, b.nomDepredador AS Depredador2
FROM come a, come b
WHERE a.nomPresa = b.nomPresa AND a.nomDepredador != b.nomDepredador AND
a.nomDepredador < b.nomDepredador
UNION
SELECT a.nomDepredador AS Depredador1, b.nomDepredador AS Depredador2
FROM se_alimenta_de1 a, se_alimenta_de1 b
WHERE a.nomPresa = b.nomPresa AND a.nomDepredador != b.nomDepredador AND
a.nomDepredador < b.nomDepredador;

```

-- 1.3 Listar los nombres de las especies que no fueron visitadas por ningún visitante que haya tenido alojamiento.

```

WITH visi_registrados AS(
SELECT * FROM registra NATURAL JOIN entrada
)
SELECT distinct(nomCientifico) FROM residen NATURAL JOIN
visi_registrados vr
WHERE vr.dni_visitante IN (SELECT distinct(dni_visitante) FROM aloja)

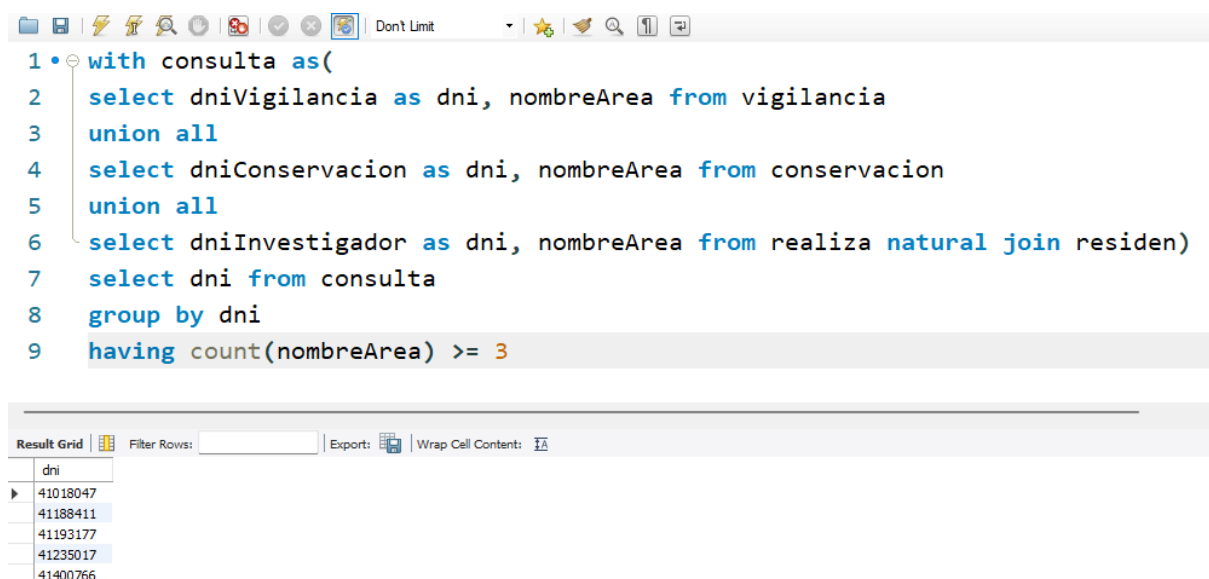
```

-- 1.4 Listar las especies que fueron visitadas por todos los visitantes.

```
SELECT nomCientifico FROM registra NATURAL JOIN entrada NATURAL JOIN residen
GROUP BY nomCientifico
HAVING COUNT(dni_visitante) =
(SELECT COUNT(dni_visitante) FROM visitantes)
```

-- 1.5 Liste el nombre de todo el personal que trabaja en al menos tres áreas distintas.

```
with consulta as(
select dniVigilancia as dni, nombreArea from vigilancia
union all
select dniConservacion as dni, nombreArea from conservacion
union all
select dniInvestigador as dni, nombreArea from realiza natural join residen)
select dni from consulta
group by dni
having count(nombreArea) >= 3
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search. The query editor contains the following SQL code:

```
1 • with consulta as(
2   select dniVigilancia as dni, nombreArea from vigilancia
3   union all
4   select dniConservacion as dni, nombreArea from conservacion
5   union all
6   select dniInvestigador as dni, nombreArea from realiza natural join residen)
7   select dni from consulta
8   group by dni
9   having count(nombreArea) >= 3
```

Below the query editor, the 'Result Grid' is visible. It has a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results table has a single column 'dni' with the following values:

dni
41018047
41188411
41193177
41235017
41400766

2) insertamos valores necesarios para las consultas:

```
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('1','perro');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('5','gato');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('6','lobo');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('7','oso');
```

```

INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('8','pitufo');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('9','ballena');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('12','calamar');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('13','ornitorrinco');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('2','lechuga');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('3','rosas');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('4','tomate');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('10','mangorolo');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('11','aloevera');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('14','limon');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('15','bananas');
INSERT INTO especies (nomCientifico, nomVulgar)
VALUES ('16','mandioca');

```

```

INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('2',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('3',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('4',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('10',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('11',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('14',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('15',

```



```

'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`vegetales` (`nomCientifico`,
`fechaInicioFloracion`, `fechaFinFloracion`, `floracion`) VALUES ('16',
'2010-12-12', '2010-12-23', '1');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('1', 'Carnivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('5', 'Carnivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('6', 'Carnivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('7', 'Carnivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('8', 'Hervivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('9', 'Hervivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('12', 'Hervivoro',
'2020-05-5', '2021-05-5');
INSERT INTO `pn-tpi`.`animales` (`nomCientifico`, `tipoAlimentacion`,
`fechaInicioCelo`, `fechaFinCelo`) VALUES ('13', 'Hervivoro',
'2020-05-5', '2021-05-5');

```

#Animal come vegetal

```

INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('1', '2');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('1', '3');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('1', '4');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('1', '16');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('7', '15');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('8', '14');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('9', '10');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES

```

```

('9', '11');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('13', '11');
INSERT INTO `pn-tpi`.`se_alimenta` (`nomDepredador`, `nomPresas`) VALUES
('12', '11');
#Animal come animal
INSERT INTO `pn-tpi`.`come` (`nomDepredador`, `nomPresas`) VALUES ('5',
'6');
INSERT INTO `pn-tpi`.`come` (`nomDepredador`, `nomPresas`) VALUES ('5',
'7');
INSERT INTO `pn-tpi`.`come` (`nomDepredador`, `nomPresas`) VALUES ('6',
'8');
INSERT INTO `pn-tpi`.`come` (`nomDepredador`, `nomPresas`) VALUES ('12',
'13');

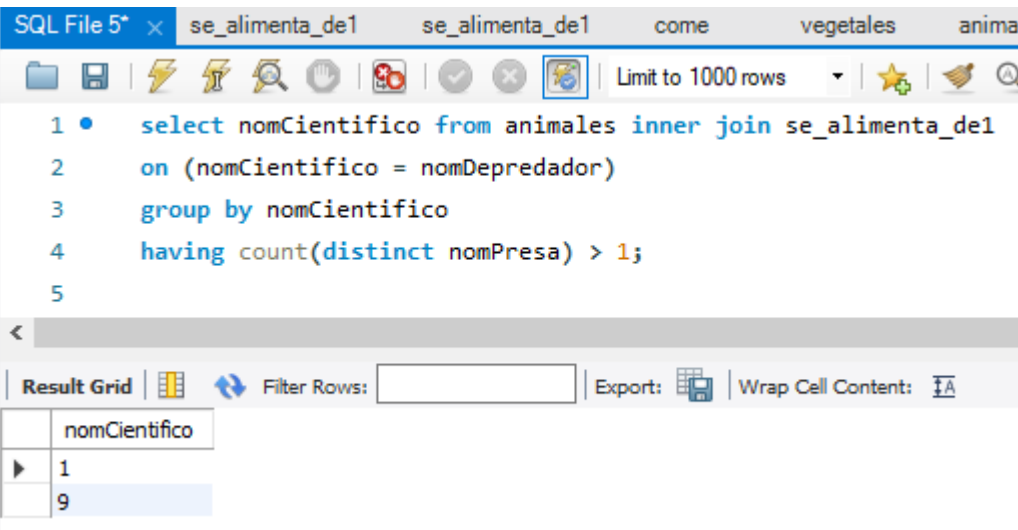
```

-- 2.1.Nombre de los animales que se alimentan de al menos dos vegetales distintos.

```

SELECT nomCientifico FROM animales INNER JOIN se_alimenta
ON (nomCientifico = nomDepredador)
GROUP BY nomCientifico
HAVING COUNT(DISTINCT nomPresas) > 1;

```



The screenshot shows a SQL client window with multiple tabs. The active tab is 'se_alimenta_de1'. The query editor displays the following SQL query:

```

1 select nomCientifico from animales inner join se_alimenta_de1
2 on (nomCientifico = nomDepredador)
3 group by nomCientifico
4 having count(distinct nomPresas) > 1;
5

```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The results are as follows:

nomCientifico
1
9

-- 2.2 Listar los pares (animal1, animal2), donde animal1 y animal2 son tales que animal1 se alimenta de otro que a su vez se alimenta de animal2.

```

SELECT a.nomDepredador, b.nomDepredador
FROM se_alimenta a, se_alimenta b
WHERE a.nomDepredador != b.nomDepredador
AND a.nomPresas = b.nomPresas
AND a.nomDepredador >= b.nomDepredador

```

```

1 • SELECT * FROM parques_naturales.come;
2 #animales a animales

```

	nomDepredador	nomPresa
	5	12
	12	13
	12	6
	5	6
	12	7
	5	7
	6	7
	6	8
	NULL	NULL

SQL File 5*

se_alimenta_de1

se_alimenta_de1

come

vegetales

animales

Limit to 1000 rows

```

1  -- 2.1.Nombre de los animales que se alimentan de al menos dos vegetales distintos.
2  • select distinct come1.nomDepredador, come2.nomPresa
3    from come as come1, come as come2
4    where come1.nomPresa = come2.nomDepredador
5

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	nomDepredador	nomPresa
▶	5	13
	5	6
	5	7
	12	7
	12	8
	5	8

#Una especie puede comerse una misma especie

-- 2.3. Listar los animales que se alimentan de todos los demás (en forma directa). Luego si la consulta está vacía inserte los registros que sean necesarios para que la respuesta no esté vacía.

```

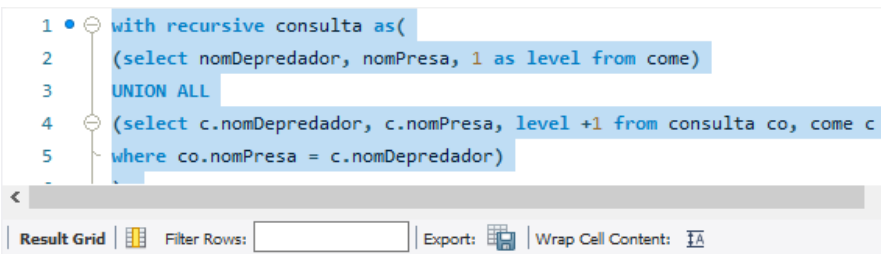
SELECT nomDepredador
FROM come
GROUP BY nomDepredador
HAVING COUNT(nomPresa) = (SELECT count(*) FROM animales)

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
	nomDepredador		
	5		

-- 2.4. Listar, para cada nombre vulgar de animal, todos los nombres vulgares de los animales de los cuales se alimenta en forma recursiva.

```
with recursive consulta as(
  (select nomDepredador, nomPresas, 1 as level from come)
  UNION ALL
  (select c.nomDepredador, c.nomPresas, level +1 from consulta co, come c
   where co.nomPresas = c.nomDepredador)
)
select distinct e1.nomCientifico, e1.nomVulgar, e2.nomCientifico,
e2.nomVulgar, level from consulta c
inner join especies e1 on e1.nomCientifico = c.nomDepredador
inner join especies e2 on e2.nomCientifico = c.nomPresas
order by e1.nomVulgar, level
```



The screenshot shows a SQL query editor with the following code:

```
1 with recursive consulta as(
2   (select nomDepredador, nomPresas, 1 as level from come)
3   UNION ALL
4   (select c.nomDepredador, c.nomPresas, level +1 from consulta co, come c
5    where co.nomPresas = c.nomDepredador)
6 )
```

Below the editor is a screenshot of the 'Result Grid' showing the output of the query. The grid has 6 columns: nomCientifico, nomVulgar, nomCientifico, nomVulgar, and level. The data is as follows:

nomCientifico	nomVulgar	nomCientifico	nomVulgar	level
6	Gato	7	Tiburon	1
6	Gato	8	Cocodrilo	1
12	Lobo	1	perro	1
12	Lobo	13	Tigre	1
12	Lobo	5	Mono	1
12	Lobo	6	Gato	1
12	Lobo	7	Tiburon	1
12	Lobo	8	Cocodrilo	1
12	Lobo	9	Yacare	1
5	Mono	6	Gato	1
5	Mono	7	Tiburon	1
6	Gato	7	Tiburon	2
6	Gato	8	Cocodrilo	2
5	Mono	6	Gato	2
5	Mono	7	Tiburon	2
6	Gato	7	Tiburon	3
6	Gato	8	Cocodrilo	3

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

-- Poniendo un límite para el nivel de profundidad de la cadena alimenticia.

```
with recursive consulta as(
  (select nomDepredador, nomPresas, 1 as level from come)
  UNION ALL
  (select c.nomDepredador, c.nomPresas, level +1 from consulta co, come c
   where co.nomPresas = c.nomDepredador and level < 10)
)
select distinct e1.nomCientifico, e1.nomVulgar, e2.nomCientifico,
```

```
e2.nomVulgar, level from consulta c
inner join especies e1 on e1.nomCientifico = c.nomDepredador
inner join especies e2 on e2.nomCientifico = c.nomPresas
order by e1.nomVulgar, level
```

```
1 • with recursive consulta as(
2   (select nomDepredador, nomPresas, 1 as level from come)
3   UNION ALL
4   (select c.nomDepredador, c.nomPresas, level +1 from consulta co, come c
5    where co.nomPresas = c.nomDepredador and level < 10)
6  )
7  select distinct e1.nomCientifico, e1.nomVulgar, e2.nomCientifico,
8  e2.nomVulgar, level from consulta c
9  inner join especies e1 on e1.nomCientifico = c.nomDepredador
10 inner join especies e2 on e2.nomCientifico = c.nomPresas
11 order by e1.nomVulgar, level
```

nomCientifico	nomVulgar	nomCientifico	nomVulgar	level
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	1
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	2
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	3
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	4
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	5
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	6
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	7
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	8
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	9
Tachyglossus aculeatus	Australian spiny anteater	Acrantophis madagascariensis	Eastern boa constrictor	10
Trachynotus vaillanti	Barbet, crested	Acrantophis madagascariensis	Eastern boa constrictor	1

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

En caso de ser necesario modifique el modelo ER de manera de responder las siguientes consultas.

3.1. Agregue una columna que muestre el promedio de entradas vendidas por mes y año por Parque y Area.

Agregamos a la tabla de entrada el campo nombreArea

```
CREATE TABLE `entrada` (
  `numeroEntrada` int NOT NULL,
  `nombreParque` varchar(55) NOT NULL,
  `nombreArea` varchar(55) NOT NULL,
  PRIMARY KEY (`numeroEntrada`),
  FOREIGN KEY (`nombreParque`) REFERENCES `areas`
  (`nombreParque`),
  FOREIGN KEY (`nombreArea`) REFERENCES `areas` (`nombreArea`)
)
```

```

WITH RegistraConFechaDesglosada AS (
SELECT *, DAY(fechaRegistro) AS dia, MONTH(fechaRegistro) AS
mes , YEAR(fechaRegistro) AS año FROM registra)
SELECT nombreParque, count(*) OVER (PARTITION BY nombreParque) AS
entradasParque,
nombreArea, count(*) OVER (PARTITION BY nombreArea) AS
entradasArea,
año, count(*) OVER (PARTITION BY año) AS entradasAño,
mes, count(*) OVER (PARTITION BY mes) AS entradasMes
FROM RegistraConFechaDesglosada r NATURAL JOIN entrada e
ORDER BY nombreParque, nombreArea, año, mes

```

```

1 • WITH RegistraConFechaDesglosada AS (
2   SELECT *, DAY(fechaRegistro) AS dia, MONTH(fechaRegistro) AS
3   mes , YEAR(fechaRegistro) AS año FROM registra)
4   SELECT distinct nombreParque, count(*) OVER (PARTITION BY nombreParque) AS entradasParque,
5   nombreArea, count(*) OVER (PARTITION BY nombreArea) AS entradasArea,
6   año, count(*) OVER (PARTITION BY año) AS entradasAño,
7   mes, count(*) OVER (PARTITION BY mes) AS entradasMes
8   FROM RegistraConFechaDesglosada r NATURAL JOIN entrada e
9   ORDER BY nombreParque, nombreArea, año, mes

```

nombreParque	entradasParque	nombreArea	entradasArea	año	entradasAño	mes	entradasMes
Gastonto Este	10329	Norte	20484	2010	9278	1	8336
Gastonto Este	10329	Norte	20484	2010	9278	2	8406
Gastonto Este	10329	Norte	20484	2010	9278	3	8239
Gastonto Este	10329	Norte	20484	2010	9278	4	8374
Gastonto Este	10329	Norte	20484	2010	9278	5	8363

3.2. A la consulta anterior agregue otra columna que muestre el ranking de parques de acuerdo al mismo criterio.

```

select nombreParque, count(*)
as entradasVendidas from registra natural join entrada
group by nombreParque
order by entradasVendidas desc;

```

nombreParque	entradasVendidas
Gastonto Norte	16214
Gastonto Este	10329
Valentina Sur	10271
Gastonto Oeste	10221
Gastonto Sur	8228
Valentina Norte	8124
Lucas Oeste	8093
Mariano Sur	8082

3.3. Liste los parques que cubren que representan el 50% del total de las ventas entradas ordenados de mayor a menor

```
SELECT e.nombreParque, COUNT(*) AS cantidadDeEntradas
FROM registra r NATURAL JOIN entrada e
GROUP BY e.nombreParque
HAVING COUNT(*) >= (SELECT (COUNT(*) DIV 2) FROM registra);
```

```
7 • SELECT e.nombreParque, COUNT(*) AS cantidadDeEntradas
8 FROM registra r NATURAL JOIN entrada e
9 GROUP BY e.nombreParque
10 HAVING COUNT(*) >= (SELECT (COUNT(*) DIV 2) FROM registra);
11
```

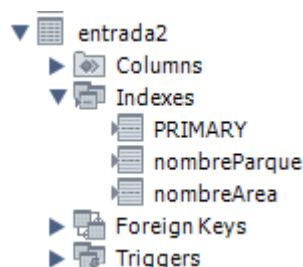
Result Grid	Filter Rows:	Export:	Wrap Cell Content:
nombreParque	cantidadDeEntradas		

4. Índices

Para realizar los siguientes ejercicios inserte más de 100k de registros en la tabla de entrada. Puede Utilizar un Excel con valores al azar e importarlos masivamente con el comando LOAD DATA

Insertamos 100k registros en la tabla registra en lugar de la tabla entrada ya que esta última guarda información de las entradas físicas de un parque, en cambio la tabla registra guarda el ingreso de los visitantes a determinado parque en determinada fecha, lo cual consideramos que es el verdadero ticket (entrada).

4.1. Tome el tiempo de la consulta 3.1. Luego cree un índice sobre id área y id parque en la tabla Entradas y vuelve a ejecutar la consulta 3.1. Ver si hay diferencia en los tiempos de ejecución con respecto a la primera.



Los índices mencionados se crearon automáticamente a la hora de crear la tabla,(por ser FOREIGN KEY) pero podemos eliminarlos y volver a crearlos para ver la diferencia entre tenerlos y no tenerlos.

Primero creamos una tabla idéntica a la tabla registra (que contiene 100 mil registros) pero sin índices, y probamos una consulta que la implica viendo su tiempo de ejecución

Creamos la tabla de prueba:

```
1 CREATE TABLE `registraTest` (  
2   `numeroEntrada` int NOT NULL,  
3   `dni_visitante` int NOT NULL,  
4   `fechaRegistro` date NOT NULL,  
5   PRIMARY KEY (`numeroEntrada`,`dni_visitante`,`fechaRegistro`),  
6   KEY `dni_visitante` (`dni_visitante`),  
7   KEY `fechaRegistro` (`fechaRegistro`)  
8 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
9  
10  
11 • INSERT INTO registraTest (SELECT * FROM registra)  
12
```

Output			
Action Output			
#	Time	Action	Message
✓ 1	22:02:35	INSERT INTO registraTest (SELECT * FROM registra)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

Ejecutamos la consulta anteponiendo el comando explain analyze:

The screenshot shows a SQL query editor with the following query:

```
1 EXPLAIN ANALYZE WITH RegistraConFechaDesglosada AS (  
2   SELECT *, day(fechaRegistro) AS dia, month(fechaRegistro) AS mes, year(fechaRegistro) AS año  
3   FROM registraTest  
4 )  
5  
6   select nombreParque, count(*) over (partition by nombreParque order by año) as countParque  
7   nombreArea, count(*) over (partition by nombreArea order by año) as countArea  
8   año, count(*) over (partition by año order by nombreParque) as countAño
```

The 'Edit Data for EXPLAIN (VARCHAR)' window shows the execution plan:

- time=0.001..7.125 rows=100000 loops=1
- time=297.406..311.261 rows=100000 loops=1
-> Temporary table (cost=2.50..2.50 rows=0) (actual time=0.001..0.001 rows=0) (loops=1)
- time=144.347..283.480 rows=100000 loops=1
-> Window aggregate with buffering: count(0) OVER (PARTITION BY e.nombreParque) (actual time=144.347..283.480 rows=100000 loops=1)
- time=132.731..139.658 rows=100000 loops=1
-> Sort: e.nombreParque (actual time=132.731..139.658 rows=100000 loops=1)
- time=0.070..91.862 rows=100000 loops=1
-> Stream results (cost=44400.20 rows=98471) (actual time=0.070..91.862 rows=100000 loops=1)
- time=0.064..58.442 rows=100000 loops=1
-> Nested loop inner join (cost=44400.20 rows=98471) (actual time=0.064..58.442 rows=100000 loops=1)
- time=0.052..28.653 rows=100000 loops=1
-> Index scan on registraTest using PRIMARY (cost=9935.35 rows=98471) (actual time=0.052..28.653 rows=100000 loops=1)
- time=0.000..0.000 rows=1 loops=1
-> Single-row index lookup on e using PRIMARY (numeroEntrada=registraTest.numeroEntrada) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=1)

The 'Result Grid' shows the output of the query:

#	Time	Action	Message	Duration / Fetch
✓ 1	22:21:37	WITH RegistraConFechaDesglosada AS (SELECT *, day(fechaRegistro) AS dia, month(fechaRegistro) AS mes, year(fechaRegistro) AS año FROM registraTest)	100000 row(s) returned	1.016 sec / 0.078 sec
✓ 2	22:21:55	EXPLAIN ANALYZE WITH RegistraConFechaDesglosada AS (SELECT *, day(fechaRegistro) AS dia, month(fechaRegistro) AS mes, year(fechaRegistro) AS año FROM registraTest)	1 row(s) returned	1.125 sec / 0.000 sec

Vemos que tarda 1.125 segundos, usando solo el índice de la primary key de la tabla. Ahora hacemos la misma consulta pero con una tabla con todos los demás índices:

The screenshot displays the SQL Server Enterprise Manager interface. On the left, a query window shows the following SQL code:

```

4 from registra
5 )
6 select nombreParque, count(*) over (part
7 nombreArea, count(*) over (partition by
8 año, count(*) over (partition by año) as
9 mes, count(*) over (partition by mes) as
10 from RegistraConFechaDesglosada r natura
11 order by nombreParque, nombreArea, año,
12

```

Below the query window, the 'Result Grid' shows the EXPLAIN output. The output is as follows:

```

14 --> Table scan on <temporary> (cost=2.50..2.50 rows=0) (actual
15 time=0.001..8.839 rows=100000 loops=1)
16 --> Temporary table (cost=2.50..2.50 rows=0) (actual
17 time=364.927..380.436 rows=100000 loops=1)
18 --> Window aggregate with buffering: count(0) OVER (PARTITION BY
19 e.nombreParque ) (actual time=208.825..350.904 rows=100000 loops=1)
20 --> Sort: e.nombreParque (actual time=196.810..204.003
21 rows=100000 loops=1)
22 --> Stream results (cost=41975.18 rows=92553) (actual
23 time=0.061..141.647 rows=100000 loops=1)
24 --> Nested loop inner join (cost=41975.18 rows=92553) (actual
25 time=0.049..105.688 rows=100000 loops=1)
26 --> Index scan on registra using fechaRegistro
27 (cost=9581.63 rows=92553) (actual time=0.040..26.227 rows=100000 loops=1)
28 --> Single-row index lookup on e using PRIMARY
29 (numeroEntrada=registra.numeroEntrada) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=100000)

```

The 'Result Grid' also shows the duration of the query execution:

#	Time	Action	Message	Duration / Fetch
1	22:27:48	EXPLAIN ANALYZE WITH RegistraConFechaDesglosada AS (SELECT *, day(fechaRegistro) AS dia, m...	1 row(s) returned	1.188 sec / 0.000 sec
2	22:28:49	EXPLAIN ANALYZE WITH RegistraConFechaDesglosada AS (SELECT *, day(fechaRegistro) AS dia, m...	1 row(s) returned	1.172 sec / 0.000 sec

Y vemos que aquí la consulta llevó un poco más de tiempo, la misma utilizó el índice fechaRegistro, en lugar del índice PRIMARY.

4.2. Ejecutar el comando explain sobre la consulta anterior. Verificar que ahora se define un index scan. Es decir, se usa el índice.

Ya se utilizó el comando EXPLAIN ANALYZE antes del select, para poder explicar las consultas mencionadas en el punto anterior.

4.3. Proponer e implementar índices para mejorar las restantes consultas del punto 1, 2 o 3 (al menos 2).

Podemos proponer los siguientes índices que mejoran la velocidad de respuesta de las consultas realizadas.

```

CREATE INDEX nombreArea ON areas (nombreArea(15));
CREATE INDEX nombreArea ON residen (nombreArea(15));
CREATE INDEX nomDepredador ON se_alimenta_de1 (nomDepredador(10));
CREATE INDEX fechaRegistro ON registra (fechaRegistro);

```

Todas las consultas de los puntos anteriores se realizan usando primary keys, las cuales por defecto son índices combinados, en los índices propuestos arriba simplemente individualizamos esos índices para cada campo clave de determinadas tablas.

SEGUNDA PARTE

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

Este trigger controla que la fecha de inicio de celo no sea igual a la fecha de fin.

```
DELIMITER //
CREATE TRIGGER control_fechas_celos
BEFORE INSERT ON animales
FOR EACH ROW
BEGIN
IF DATEDIFF(new.fechaFinCelo, new.fechaInicioCelo)<1 THEN
signal sqlstate "02000" SET MESSAGE_TEXT = "La fecha de inicio de celo
no puede ser igual o menor a la fecha de fin";
END IF;
END//
DELIMITER ;
```

--Este trigger elimina los registros de "disfruta" (que relaciona una excursión con un visitante y los registros de "organiza" (que relaciona el parque con una excursión y un alojamiento) antes de eliminar una excursión:

```
DELIMITER //
CREATE TRIGGER eliminar_excursiones
BEFORE DELETE ON excursiones
FOR EACH ROW
BEGIN

DELETE FROM `pn-tpi`.`disfruta` WHERE id_excursion = old.id_excursion;
DELETE FROM `pn-tpi`.`organiza` WHERE id_excursion = old.id_excursion;

END
// DELIMITER ;
```

Este trigger resguarda los nuevos proyectos costosos donde su presupuesto supera los \$20.000 pesos o dólares.

```
Create table auditoria_tpi(  
    id_pdi int,  
    monto int NOT NULL,  
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    PRIMARY KEY(id_pdi, monto)  
);  
  
DELIMITER //  
CREATE TRIGGER auditoria_proyectos_de_investigacion  
after insert on proyectodeinves  
for each row  
begin  
If new.presupuesto > 20000 then  
insert into auditoria_tpi (SELECT new.id_proyecto, new.presupuesto FROM  
proyectodeinves);  
end if;  
end //  
DELIMITER ;
```

Programas almacenados

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

Este procedimiento recibe un tipo de mineral y devuelve un informe donde se muestra cada parque con su cantidad de minerales de dicho tipo siempre y cuando la cantidad sea mayor a 100.

```
DELIMITER //  
CREATE PROCEDURE individuos_prom_por_parque(in tipoMineral VARCHAR(55))  
BEGIN  
DECLARE minimo INT;  
SET minimo = 50;  
SELECT r.nombreParque, SUM(r.cantIndividuos) as CantidadDeMineral  
FROM residen r NATURAL JOIN minerales m  
WHERE r.nomCientifico = tipoMineral  
GROUP BY r.nombreParque  
HAVING SUM(r.cantIndividuos) > minimo;  
END //  
DELIMITER ;
```

```

2  CREATE PROCEDURE individuos_prom_por_parque(in tipoMineral VARCHAR(55))
3  BEGIN
4  DECLARE minimo INT;
5  SET minimo = 50;
6  SELECT r.nombreParque, SUM(r.cantIndividuos) as CantidadDeMineral
7  FROM residen r NATURAL JOIN minerales m
8  WHERE r.nomCientifico = tipoMineral
9  GROUP BY r.nombreParque
10  HAVING SUM(r.cantIndividuos) > minimo;
11  END //
12  DELIMITER ;
13
14 • call individuos_prom_por_parque('Geochelone elegans')

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
nombreParque	CantidadDeMineral			
Gastonto Sur	84			
Lucas Sur	200			

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.

Esta función recibe el D.N.I. de un investigador y devuelve la sumatoria de los presupuestos de los proyectos que realiza si supera los 100000

```

DELIMITER //
CREATE FUNCTION get_sum_presupuesto(dniInve int)
RETURNS double DETERMINISTIC
BEGIN
DECLARE sum_presupuesto double;
DECLARE supera_monto int;
SET supera_monto = 5000;
SELECT SUM(pi.presupuesto) INTO sum_presupuesto
FROM investigador i INNER JOIN realiza r USING(dniInvestigador)
INNER JOIN proyectodeinves pi USING(id_proyecto)
WHERE r.dniInvestigador = dniInve
GROUP BY i.dniInvestigador
HAVING SUM(pi.presupuesto) > supera_monto;
RETURN sum_presupuesto;
END//
DELIMITER ;
SELECT get_sum_presupuesto(41193177)

```

```
1 DELIMITER //
2 • CREATE FUNCTION get_sum_presupuesto(dniInvE int)
3 RETURNS double DETERMINISTIC
4 BEGIN
5 DECLARE sum_presupuesto double;
6 DECLARE supera_monto int;
7 SET supera_monto = 5000;
8 SELECT SUM(pi.presupuesto) INTO sum_presupuesto
9 FROM investigador i INNER JOIN realiza r USING(dniInvestigador)
10 INNER JOIN proyectodeinves pi USING(id_proyecto)
11 WHERE r.dniInvestigador = dniInve
12 GROUP BY i.dniInvestigador
13 HAVING SUM(pi.presupuesto) > supera_monto;
14 RETURN sum_presupuesto;
15 END//
16 DELIMITER ;
17
18 SELECT get_sum_presupuesto(41193177)
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
get_sum_presupuesto(41193177)				
▶		34206.7		

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:
 - Programador
 - Diseñador
 - Administrador

Los roles nos permiten crear agrupaciones para facilitar la administración de la bdd. Se utilizará el siguiente comando para la creación de roles no existentes actualmente en la bdd.

```
CREATE ROLE IF NOT EXISTS 'Programador', 'Diseñador',  
'Administrador';
```

The screenshot shows the MySQL Workbench interface. The top panel displays the 'Schemas' tree on the left, with 'BDA_TPI' selected. The main editor shows a SQL query: `1 • CREATE ROLE IF NOT EXISTS Programador, Diseñador, Administrador;`. The bottom panel shows the 'Output' tab with a table of actions and their results.

#	Time	Action	Message	Duration / Fetch
25	12:39:55	Create table Alojaj [id_alojamiento integer, dni_visitante integer, fechaSalida date, h...	0 row(s) affected	0.047 sec
26	12:39:55	Create table Dishuta [id_excursion integer, dni_visitante integer, PRIMARY KEY (id_excursion, dni_visit...	0 row(s) affected	0.047 sec
27	12:39:55	Create table Organiza [id_alojamiento integer, dni_visitante integer, PRIMARY KEY (id_alojamiento, dni...	0 row(s) affected	0.140 sec
28	12:39:55	create table registro [numeroEntrada integer, dni_visitante integer, fechaRegistro date, nombreParque ...	0 row(s) affected	0.078 sec
29	16:29:32	CREATE ROLE IF NOT EXISTS Programador, Diseñador, Administrador	0 row(s) affected	0.047 sec

Below the output, a SQL query is shown in the 'SQL File 7*' window:

```
1 • SELECT DISTINCT User 'Role Name', if(from_user is NULL,0, 1) Active
2 FROM mysql.user LEFT JOIN role_edges ON from_user=user
3 WHERE account_locked='Y' AND password_expired='Y' AND authentication_string='';
```

The 'Result Grid' at the bottom shows the results of the query:

Role Name	Active
Administrador	0
Diseñador	0
Programador	0

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

Los roles nos permiten crear agrupaciones con los mismos derechos para facilitar la administración de la bdd y no tener que asignarlos de forma individual a cada usuario así cada uno de ellos podrá realizar sus tareas correspondientes con cierto nivel de responsabilidad.

Rol administrador: debe permitir a los usuarios pertenecientes a este rol, administrar el funcionamiento del servidor MySQL. Estos privilegios no son específicos ya que dependen de la BDD determinada, por lo tanto son privilegios globales. Nos parece que el administrador de la BDD debería tener todos los privilegios. El comando GRANT nos permitirá asignarle una gran cantidad de privilegios (select, files, shutdowns etc)

Rol diseñador: El diseñador estaría a cargo del modelado de la base de datos definiendo su estructura lógica determinando entidades, relaciones, almacenamiento, acceso de datos, etc. entonces decidimos darle permisos de databases, tables, columns, indexes.

Rol programador: Nosotros decidimos que el programador sea capaz de hacer consultas, procedimientos almacenados(procedures: create, alter, drop, etc), funciones y 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.**

Para administrador

Asignamos los permisos al rol.

```
GRANT ALL ON BDA_TPI.* TO 'Administrador';
```

Luego creamos un usuario administrador con contraseña compleja y que expire luego de un determinado tiempo además permitimos que pueda acceder fuera del local host.

```
CREATE USER 'administrador1'@'%' IDENTIFIED BY 'Contraseña123!'
PASSWORD EXPIRE INTERVAL 60 DAY;
```

Por último asignamos roles a los usuarios.

```
GRANT Administrador TO 'administrador1'@'%';
```

Para diseñador

Asignamos los permisos al rol.

```
GRANT
    ALTER, CREATE, DELETE, DROP, INDEX, INSERT, REFERENCES,
    SELECT, TRIGGER, UPDATE, EVENT, LOCK TABLES
ON BDA_TPI.* TO 'Diseñador';
```

Luego creamos un usuario diseñador con contraseña compleja y que expire luego de un determinado tiempo y solo pueda acceder desde el localhost.

```
CREATE USER 'diseñador1'@'localhost' IDENTIFIED BY
'Contraseña123!' PASSWORD EXPIRE INTERVAL 60 DAY;
```

Por último asignamos roles a los usuarios.

```
GRANT Diseñador TO 'diseñador1'@'localhost';
```

Para programador

Asignamos los permisos al rol.

```
GRANT
    SELECT, INSERT, UPDATE, TRIGGER, SHOW VIEW, CREATE VIEW
ON BDA_TPI.* TO 'Programador';
```

Luego creamos un usuario programador con contraseña compleja y que expire luego de un determinado tiempo y solo pueda acceder desde el localhost.

```
CREATE USER 'programador1'@'localhost' IDENTIFIED BY
'Contraseña123!' PASSWORD EXPIRE INTERVAL 60 DAY;
```

Por último asignamos roles a los usuarios.

```
GRANT Programador TO 'programador1'@'localhost';
```

Verificación

Para comprobar los permisos otorgados:

```
SHOW GRANTS FOR 'administrador1'@'%';
```

SQL File 7* x visitantes

Limit to 400 rows

```

1 • GRANT Diseñador TO 'diseñador1'@'localhost';
2
3 • show grants for 'diseñador1'@'localhost';

```

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

Grants for diseñador1@localhost	
▶	GRANT USAGE ON *.* TO 'diseñador1'@'localhost'
	GRANT 'Diseñador'@'%' TO 'diseñador1'@'localhost'

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.

Probamos realizar acciones desde diseñador:

Query 1 x

Limit to 400 rows

```

1 use parques_naturales;
2 • select current_role()

```

Result Grid | Filter Rows: | E

current_role()
NONE

Nos indica que aún no tenemos el rol seteado entonces, usamos set role para poder realizar por ejemplo las consultas select en la base de datos requerida.

SQL File 7* x

Limit to 400 rows

```

1 • set role 'Diseñador'@'%';
2 • use parques_naturales;
3 • select * from visitantes;

```

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

	dni_visitante	domicilio	profesion	nombre	informacion
▶	1	French 122	QA	Sosa Valentina	{"age": 37, "dni": 41541781, "name": "George ...
	2	Chacabuco 133	Arquitecto de Software	Gonzalez Mariano	{"age": 29, "dni": 43834723, "name": "Wilda Cr...
	3	Santa Rosa 144	BDA	Acosta Gaston	{"age": 40, "dni": 43069739, "name": "Albert Bri...
	4	España 222	Ingeniero en Sistemas	Ruiz Franco	{"age": 28, "dni": 38811370, "name": "Harringt...
	5	4 Dryden Trail	Registered Nurse	Romain Ugo	{"age": 20, "dni": 42228007, "name": "Sheryl B...
	6	83464 Anderson Place	Web Developer III	Wakefield Mauchline	{"age": 31, "dni": 44304922, "name": "Barron R...
	7	1 Duke Park	Graphic Designer	Addy Schlagtmans	{"age": 17, "dni": 41034980, "name": "Colon Du...
	8	4 Montana Junction	Nurse	Maddie Sonley	{"age": 18, "dni": 39360472, "name": "Matthew...
	9	07095 Fordem Hill	Sales Representative	Leon Rolingson	{"age": 37, "dni": 39227545, "name": "Rogers ...

Respaldo y recuperación

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

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

Para un volumen de transacciones diarias (210000 inserciones aprox.) y una disponibilidad requerida de la base de datos de 8 horas para la carga de datos desde los clientes y 10 horas extras para otras transacciones internas.

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

Analizando la carga de la base de datos por parte de los clientes y la transaccionalidad se fijó como hora cero para el backup las 00:00 AM ya que no hay carga de datos o transaccionalidad en ese horario, por lo que se realizará “en frío”.

Se definió un esquema de respaldo con cintas. Los días lunes se hace un respaldo incremental, los días jueves un diferencial y los domingos cada un mes un respaldo full.

Se programó la toma del backup a través de tareas automáticas del sistema operativo, necesitando solo la intervención del usuario para el cambio de cinta y la validación de los backups.

Se debe manejar un conjunto de 5 (cinco) cintas dos para respaldo full, una para respaldo incremental y dos para diferencial, las cuales se deben intercambiar cada una semana y rotándolas cada 2 (dos) semanas, es decir se contará con el backup de las últimas dos semanas.

Por fines prácticos y teniendo en cuenta cualquier inconveniente que pueda surgir antes de realizar el respaldo se fijó las 8 de la noche (20:00hs) como la hora en que se debe realizar el intercambio de cinta en el servidor, siguiendo la secuencia determinada. Este proceso se realizará los días domingos, lunes y jueves y debe hacerlo la persona responsable de los backups.

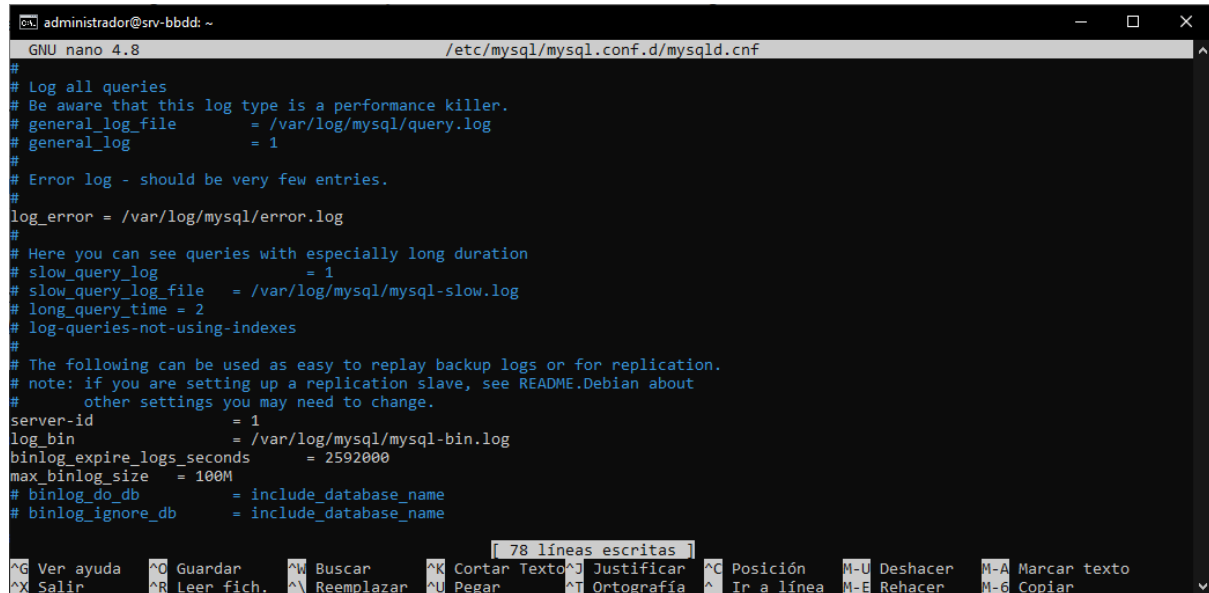
Por mantenimiento, confiabilidad y seguridad se recomienda cambiar el conjunto de cintas por unas nuevas cada seis meses.

La figura muestra el diagrama resultante de la esquematización de la estrategia de backup:

		lunes	martes	miercoles	jueves	viernes	sabado	domingo
semana 1	10:00							
	14:00							
	18:00							
	20:00	C Incremental			C Diferencial dia 1			C Full dia 1
	00:00	R.Incremental			R.Diferencial			R Full
		lunes	martes	miercoles	jueves	viernes	sabado	domingo
semana 2	10:00							
	14:00							
	18:00							
	20:00	C Incremental			C Diferencial dia 2			
	00:00	R.Incremental			R.Diferencial			
		lunes	martes	miercoles	jueves	viernes	sabado	domingo
semana 3	10:00							
	14:00							
	18:00							
	20:00	C Incremental			C Diferencial dia 1			
	00:00	R.Incremental			R.Diferencial			
		lunes	martes	miercoles	jueves	viernes	sabado	domingo
semana 4	10:00							
	14:00							
	18:00							
	20:00	C Incremental			C Diferencial dia 2			C Full dia 2
	00:00	R.Incremental			R.Diferencial			R Full
		C Incremental : cambio de cinta incremental				R Incremental : respaldo incremental		
		C Diferencial Dia # : cambio de cinta diferencial				R Diferencial : respaldo diferencial		
		C Full dia # : cambio de cinta full				R Full : respaldo full		

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

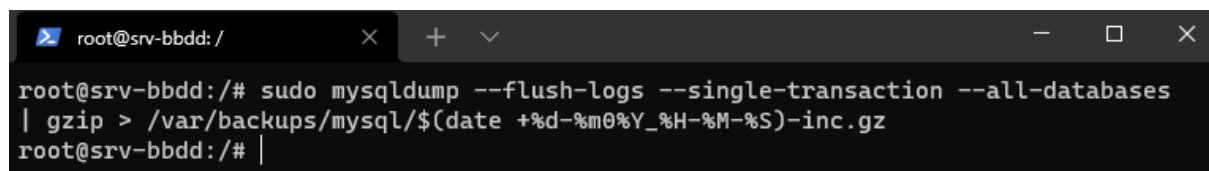
Habilitamos los binary logs.



```
administrador@srv-bbdd: ~  
GNU nano 4.8 /etc/mysql/mysql.conf.d/mysqld.cnf  
#  
# Log all queries  
# Be aware that this log type is a performance killer.  
# general_log_file = /var/log/mysql/query.log  
# general_log = 1  
#  
# Error log - should be very few entries.  
#  
log_error = /var/log/mysql/error.log  
#  
# Here you can see queries with especially long duration  
# slow_query_log = 1  
# slow_query_log_file = /var/log/mysql/mysql-slow.log  
# long_query_time = 2  
# log-queries-not-using-indexes  
#  
# The following can be used as easy to replay backup logs or for replication.  
# note: if you are setting up a replication slave, see README.Debian about  
# other settings you may need to change.  
server-id = 1  
log_bin = /var/log/mysql/mysql-bin.log  
binlog_expire_logs_seconds = 2592000  
max_binlog_size = 100M  
# binlog_do_db = include_database_name  
# binlog_ignore_db = include_database_name  
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Texto ^J Justificar ^C Posición M-U Deshacer M-A Marcar texto  
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^T Ortografía ^_ Ir a línea M-E Rehacer M-6 Copiar
```

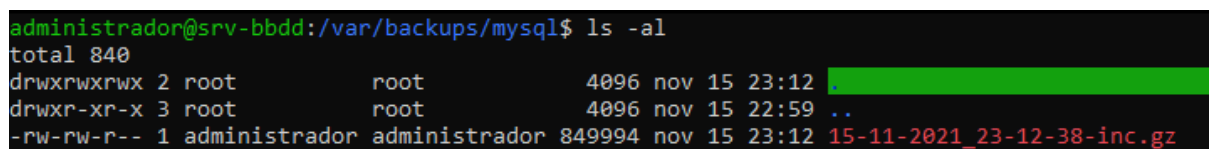
Ejecutamos el siguiente comando para crear un backup full:

```
sudo mysqldump --flush-logs --single-transaction --all-databases | gzip >  
/var/backups/mysql/$(date +%d-%m%Y_%H-%M-%S)-inc.gz
```



```
root@srv-bbdd: /  
root@srv-bbdd:/# sudo mysqldump --flush-logs --single-transaction --all-databases  
| gzip > /var/backups/mysql/$(date +%d-%m%Y_%H-%M-%S)-inc.gz  
root@srv-bbdd:/#
```

Dicho comando al ejecutarse creará un archivo comprimido con el siguiente formato en su nombre “dd-mm-yyy_HH_MM_SS-inc”:



```
administrador@srv-bbdd:/var/backups/mysql$ ls -al  
total 840  
drwxrwxrwx 2 root root 4096 nov 15 23:12 .  
drwxr-xr-x 3 root root 4096 nov 15 22:59 ..  
-rw-rw-r-- 1 administrador administrador 849994 nov 15 23:12 15-11-2021_23-12-38-inc.gz
```

Para implementar los backups full e incrementales, programamos el siguiente script en crontab:

```
0 0 */15 * 0 sudo mysqldump --flush-logs --delete-master-logs --single-transaction  
--al-databases | gzip > /var/backups/mysql/full_$(date  
+%d-%m%Y_%H-%M-%S)-inc.gz
```

```
0 0 */3 * 1,5 sudo bash ~/scripts/mysql_inc_backup.sh
```

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

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

Choose 1-4 [1]: 1
crontab: installing new crontab
```

```
GNU nano 4.8 /tmp/crontab.KHhMAT/crontab
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 0 */15 * 0 sudo mysqldump --flush-logs --delete-master-logs --single-transaction --al
0 0 */3 * 1,5 sudo bash ~/scripts/mysql_inc_backup.sh
```

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

Primer caso de pérdida :

Pérdida total de la base de datos, se hace un drop de la base de datos y se debe restaurar.

En el caso de que se pierda la BD el día lunes antes del resguardo incremental, se debería tomar el último resguardo full si se hizo el día anterior, si no se hizo un full el día anterior se debería tomar el último full y el último diferencial .

Segundo caso de pérdida :

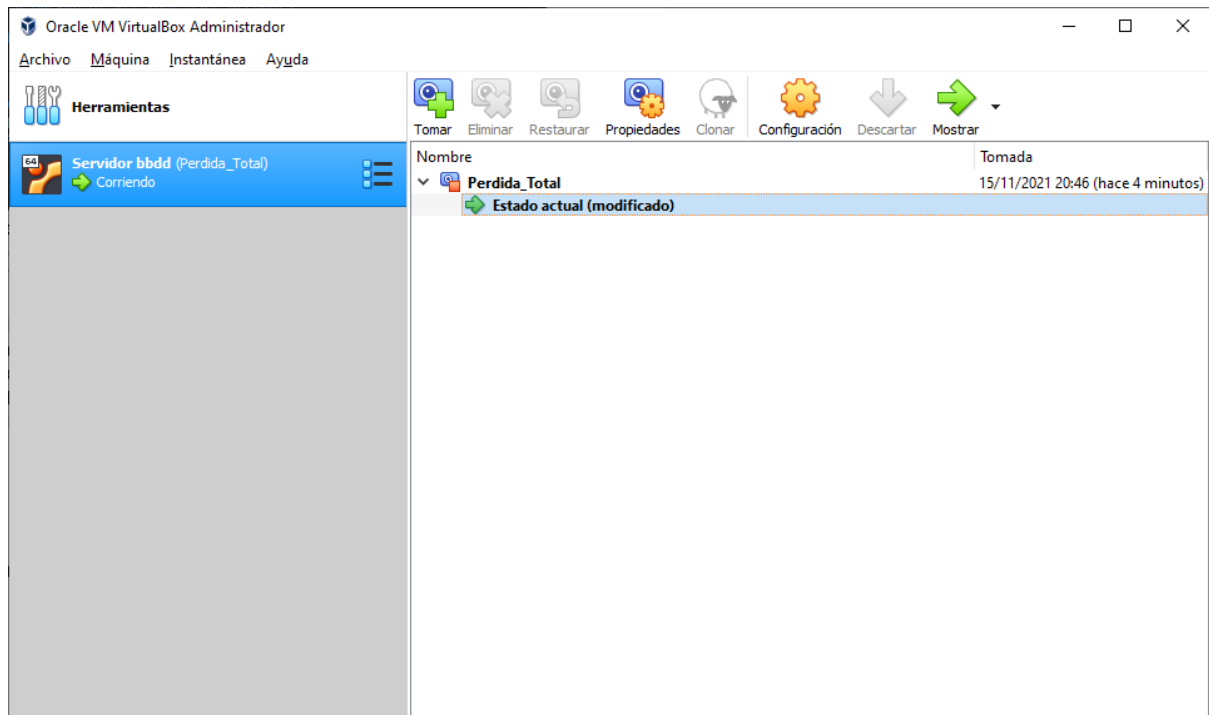
Datos inconsistentes : algún fallo de foreign key o borrado de datos erróneo.

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

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

Si ocurrió entre el jueves y el lunes se debería tomar el último diferencial.

Pérdida total:



```
mysql> drop database tpi;
Query OK, 31 rows affected (0,94 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0,00 sec)

mysql>
```

Ahora restauramos la base de datos con el siguiente comando:

```
sudo gunzip < nombredelbackup.gz | sudo mysql -u root
```



```
sudo gunzip < 15-11-2021_23-12-38-inc.gz | sudo mysql -u root
```

```
mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| tpi |
+-----+
5 rows in set (0,00 sec)
```

Datos semiestructurados

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

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

Escenario

El ministerio de Medio Ambiente junto al de Educación decidieron implementar un sitio web para visitas virtuales a los parques nacionales y a la vez un programa internacional de becas de intercambio para estudiantes interesados en la temática, por lo que comenzó a recolectar información tanto en el sitio web como en los alojamientos de los parques. Esa información se volcará en formato JSON a la tabla de visitantes de la base de datos desarrollada, de acuerdo a la siguiente estructura:

```

{
  "visitante": {
    "dni" : 40404040,
    "nombre" : "juan perez",
    "edad" : 24,
    "ciudad" : "Madrid",
    "nivel estudio" : "universitario",
    "email" : "juanpa24@correos.com",
    "redes sociales" : [
      {"instagram" : "juanpa24"},
      {"twitter" : "juanpa24"}
    ],
    "parques visitados" : [
      {
        "parque" : {
          "nombre" : "Timanfaya",
          "pais" : "España",
          "tipo visita" : "presencial"
        }
      },
      {
        "parque" : {
          "nombre" : "Monte perdido",
          "pais" : "España",
          "tipo visita" : "virtual"
        }
      }
    ],
    "parques de interés" : [
      {"pais" : "Argentina"},
      {"parque" : "Nacional Chaco"},
      {"pais" : "Argentina"},
      {"parque" : "Mburucuyá"}
    ],
    "especies de interes" : [
      {
        "animales" :
          ["zorro gris", "perro blanco"]
      },
      {
        "vegetales" :
          ["sauce llorón"]
      },
      {
        "minerales" :
          ["carbón blanco", "grafito"]
      }
    ]
  }
}

```

Consultas:

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

1. Obtener el promedio de edad de los visitantes con nivel de estudio universitario.
2. Listar los nombres de las especies vegetales con mayor cantidad de interesados.
3. Listar los nombres, edades, ciudad de residencia y nombre de los parques de interés de quienes tienen interés en visitar parques en Argentina.
4. Listar los nombres, edades y ciudad de residencia de quienes solamente hicieron visitas virtuales (ninguna presencial).
5. Listar los nombres y redes sociales de quienes hicieron alguna visita a parques en Argentina y les interesa la serpiente Yará.

Primero agregamos la columna información a la tabla visitantes:

```

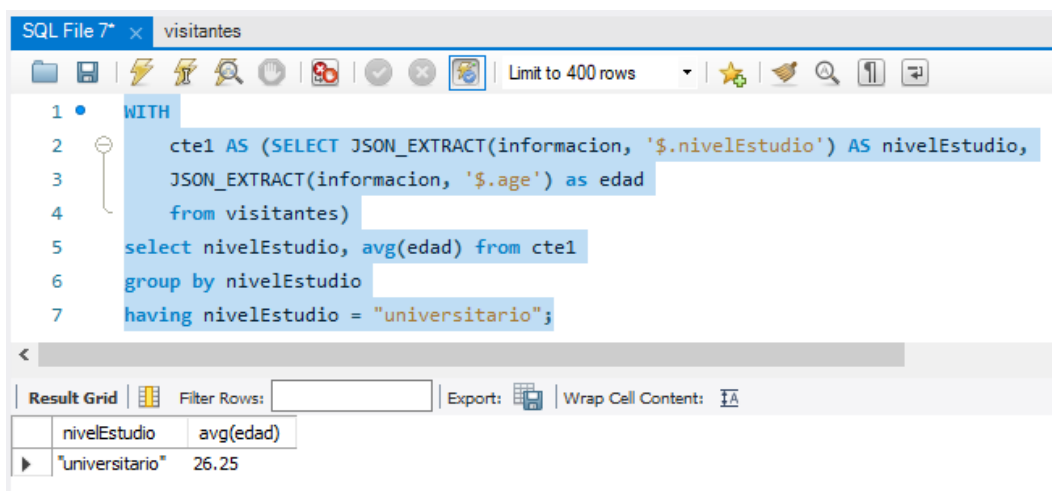
ALTER TABLE visitantes
ADD COLUMN informacion JSON;

```

Luego creamos masivamente los UPDATE's para incorporar la información JSON a cada uno de los registros de visitantes. Estos updates fueron generados mediante <https://json-generator.com/> y los updates para agregarlos a los registros existentes los creamos mediante un programa hecho por el grupo en Python.

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

```
WITH
    cte1 AS (SELECT JSON_EXTRACT(informacion, '$.nivelEstudio')
AS nivelEstudio,
    JSON_EXTRACT(informacion, '$.age') as edad
    from visitantes)
select nivelEstudio, avg(edad) from cte1
group by nivelEstudio
having nivelEstudio = "universitario";
```



2. Listar los nombres de las especies vegetales con mayor cantidad de interesados.

```
WITH consulta_cte AS(
SELECT Mine, count(*) as cant
FROM(
    SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[0]')
as Mine FROM visitantes
    UNION ALL
    SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[1]')
as Mine FROM visitantes
    UNION ALL
    SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[2]')
as Mine FROM visitantes
    UNION ALL
    SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[3]')
as Mine FROM visitantes
) subcons
```



```

WHERE Mine IS NOT NULL
GROUP BY Mine)
SELECT ct.Mine, ct.cant FROM consulta_cte ct
GROUP BY ct.Mine
HAVING ct.cant = (SELECT MAX(ctt.cant) FROM consulta_cte ctt)

```

```

12 • WITH consulta_cte AS(
13     SELECT Mine, count(*) as cant
14     FROM(
15         SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[0]') as Mine FROM visitantes
16         UNION ALL
17         SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[1]') as Mine FROM visitantes
18         UNION ALL
19         SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[2]') as Mine FROM visitantes
20         UNION ALL
21         SELECT JSON_EXTRACT(informacion, '$.especiesDeInteres[*].vegetales[3]') as Mine FROM visitantes
22     ) subcons
23     WHERE Mine IS NOT NULL
24     GROUP BY Mine)
25 SELECT ct.Mine, ct.cant FROM consulta_cte ct
26 GROUP BY ct.Mine
27 HAVING ct.cant = (SELECT MAX(ctt.cant) FROM consulta_cte ctt)

```

Mine	cant
["Abeto"]	5104

3. Listar los nombres, edades, ciudad de residencia y nombre de los parques de interés de quienes tienen interés en visitar parques en Argentina.

```

select JSON_EXTRACT(informacion, '$.name') AS nombre,
JSON_EXTRACT(informacion, '$.age') AS edad,
JSON_EXTRACT(informacion, '$.ciudad') AS ciudad
from visitantes
where JSON_EXTRACT(informacion, '$.parquesDeInteres[*]') like
'%Argentina%';

```

```

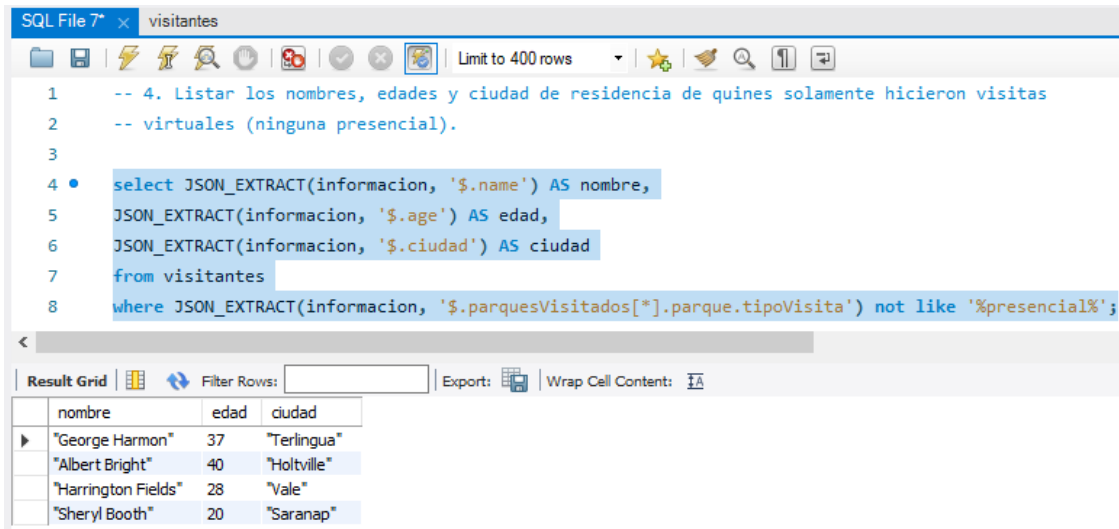
1 • select JSON_EXTRACT(informacion, '$.name') AS nombre,
2     JSON_EXTRACT(informacion, '$.age') AS edad,
3     JSON_EXTRACT(informacion, '$.ciudad') AS ciudad
4     from visitantes
5     where JSON_EXTRACT(informacion, '$.parquesDeInteres[*]') like '%Argentina%';

```

nombre	edad	ciudad
"Woodward Leonard"	31	"Tyhee"
"Alyssa Nguyen"	22	"Canoochee"
"Julianne Santana"	30	"Avoca"
"Hurley Roberson"	27	"Odessa"

4. Listar los nombres, edades y ciudad de residencia de quienes solamente hicieron visitas virtuales (ninguna presencial).

```
select JSON_EXTRACT(informacion, '$.name') AS nombre,  
JSON_EXTRACT(informacion, '$.age') AS edad,  
JSON_EXTRACT(informacion, '$.ciudad') AS ciudad  
from visitantes  
where JSON_EXTRACT(informacion,  
'$.parquesVisitados[*].parque.tipoVisita') not like '%presencial%';
```



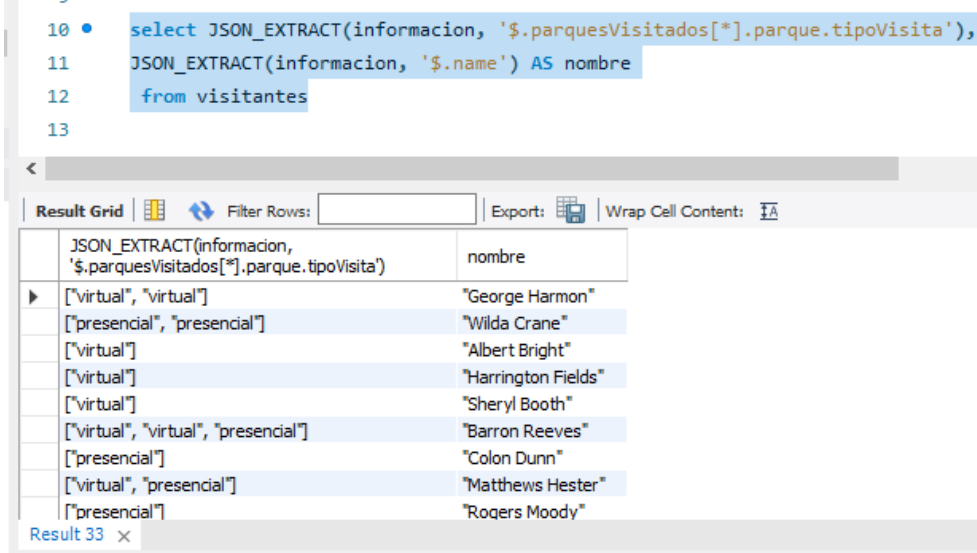
The screenshot shows a SQL IDE window titled "visitantes". The query editor contains the following SQL code:

```
-- 4. Listar los nombres, edades y ciudad de residencia de quienes solamente hicieron visitas  
-- virtuales (ninguna presencial).  
  
select JSON_EXTRACT(informacion, '$.name') AS nombre,  
JSON_EXTRACT(informacion, '$.age') AS edad,  
JSON_EXTRACT(informacion, '$.ciudad') AS ciudad  
from visitantes  
where JSON_EXTRACT(informacion, '$.parquesVisitados[*].parque.tipoVisita') not like '%presencial%';
```

The "Result Grid" shows the following data:

nombre	edad	ciudad
"George Harmon"	37	"Terlingua"
"Albert Bright"	40	"Holtville"
"Harrington Fields"	28	"Vale"
"Sheryl Booth"	20	"Saranap"

Comprobamos:



The screenshot shows a SQL IDE window with the following SQL code:

```
select JSON_EXTRACT(informacion, '$.parquesVisitados[*].parque.tipoVisita'),  
JSON_EXTRACT(informacion, '$.name') AS nombre  
from visitantes
```

The "Result Grid" shows the following data:

JSON_EXTRACT(informacion, '\$.parquesVisitados[*].parque.tipoVisita')	nombre
["virtual", "virtual"]	"George Harmon"
["presencial", "presencial"]	"Wilda Crane"
["virtual"]	"Albert Bright"
["virtual"]	"Harrington Fields"
["virtual"]	"Sheryl Booth"
["virtual", "virtual", "presencial"]	"Barron Reeves"
["presencial"]	"Colon Dunn"
["virtual", "presencial"]	"Matthews Hester"
["presencial"]	"Rogers Moody"

5. Listar los nombres y redes sociales de quienes hicieron alguna visita a parques en Argentina y les interesa la serpiente Yará.

```
select JSON_EXTRACT(informacion, '$.name') AS nombre,  
JSON_EXTRACT(informacion, '$.redesSociales[0].instagram') AS instagram,  
JSON_EXTRACT(informacion, '$.redesSociales[0].twitter') AS twitter
```

```

from visitantes
where JSON_EXTRACT(informacion, '$.especiesDeInteres[*].animales') like
'%serpiente Yarara%'
and JSON_EXTRACT(informacion, '$.parquesVisitados[*].parque.pais') like
'%Argentina%' ;

```

Cambiamos los valores de búsqueda para que la consulta nos muestre algún resultado:

```

4 • select JSON_EXTRACT(informacion, '$.name') AS nombre,
5   JSON_EXTRACT(informacion, '$.redesSociales[0].instagram') AS instagram,
6   JSON_EXTRACT(informacion, '$.redesSociales[0].twitter') AS twitter
7 from visitantes
8 where JSON_EXTRACT(informacion, '$.especiesDeInteres[*].animales') like '%Oso%'
9 and JSON_EXTRACT(informacion, '$.parquesVisitados[*].parque.pais') like '%Italy%' ;
10

```

Result Grid			
Filter Rows:			
Export:			
Wrap Cell Content:			
	nombre	instagram	twitter
▶	"Albert Bright"	"Cynthia32"	"Hanson37"

FIN.