

# Optimización y rendimiento en MySQL

19



IVANA SÁNCHEZ PÉREZ-2ºASIR

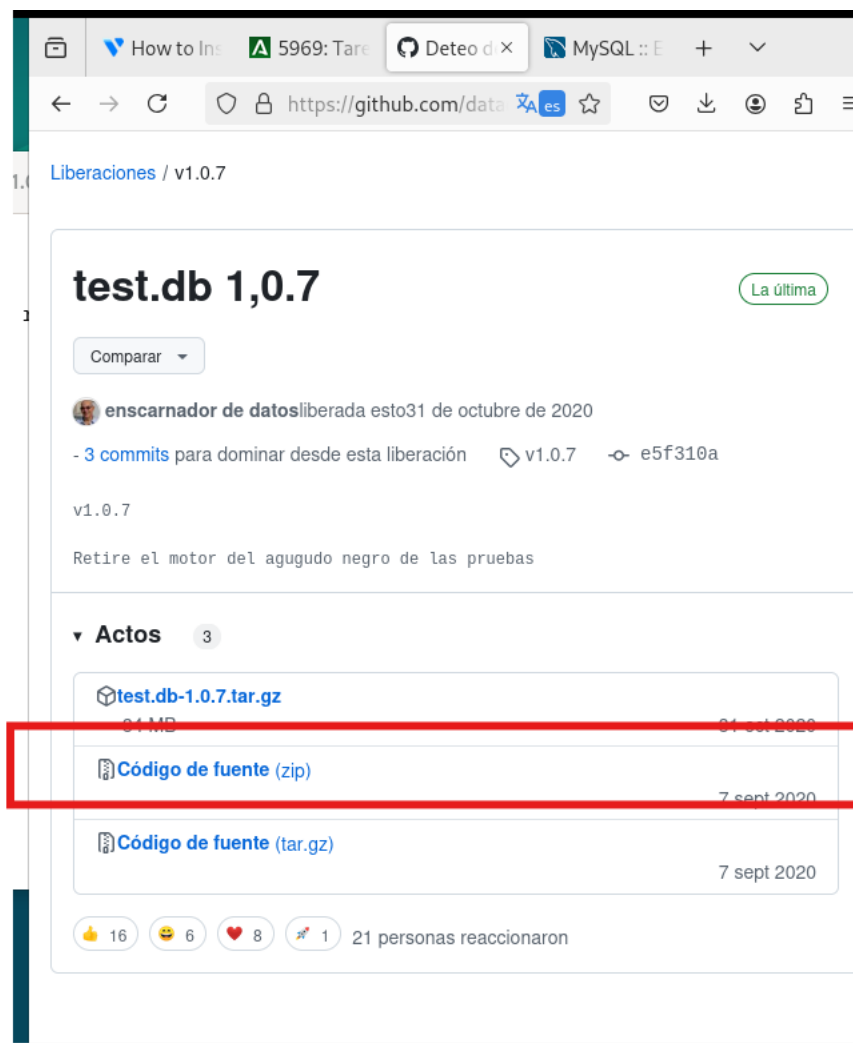


# 1.- Introducción

En esta tarea tiene como objetivo ejemplificar de forma práctica algunos de los aspectos más relevantes para que una base de dato se mantenga en marcha en las mejores condiciones de rendimiento.

## 2.- Apartado 1

Voy a utilizar una máquina virtual Debian12 para realizar la tarea. Para ello, lo primero que voy a hacer es descargar MySQL a través de su terminal, con sus correspondientes archivos debug, y el MySQL Workbench, para trabajar de una forma más cómoda con una interfaz gráfica.

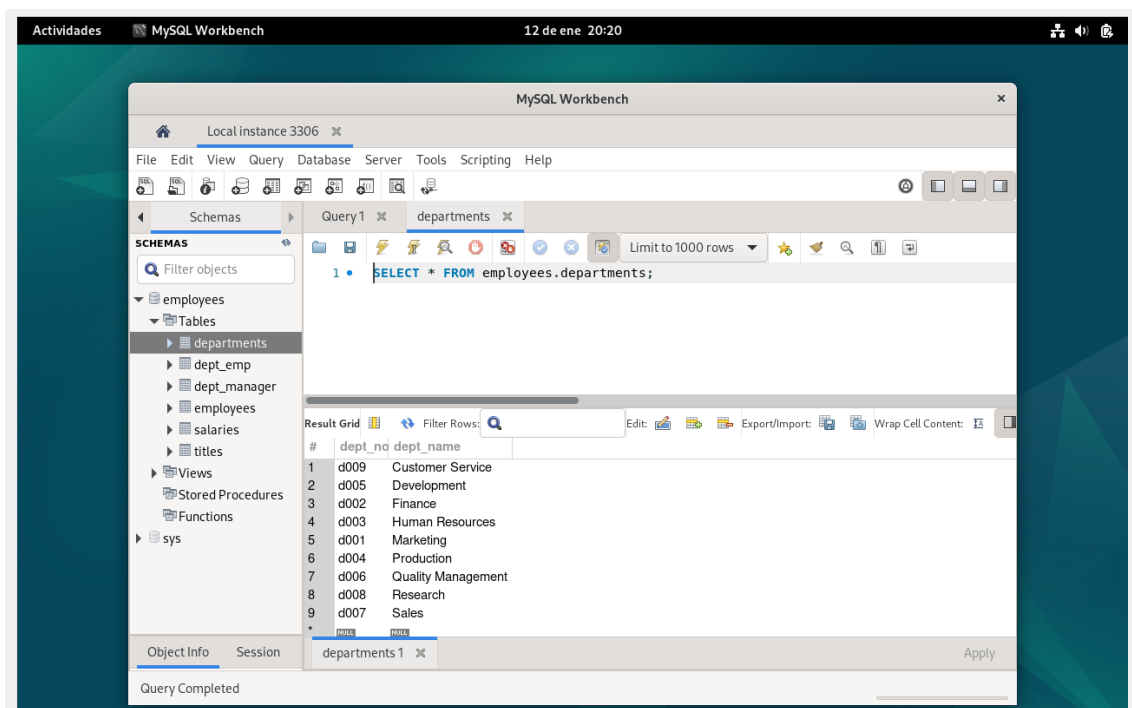
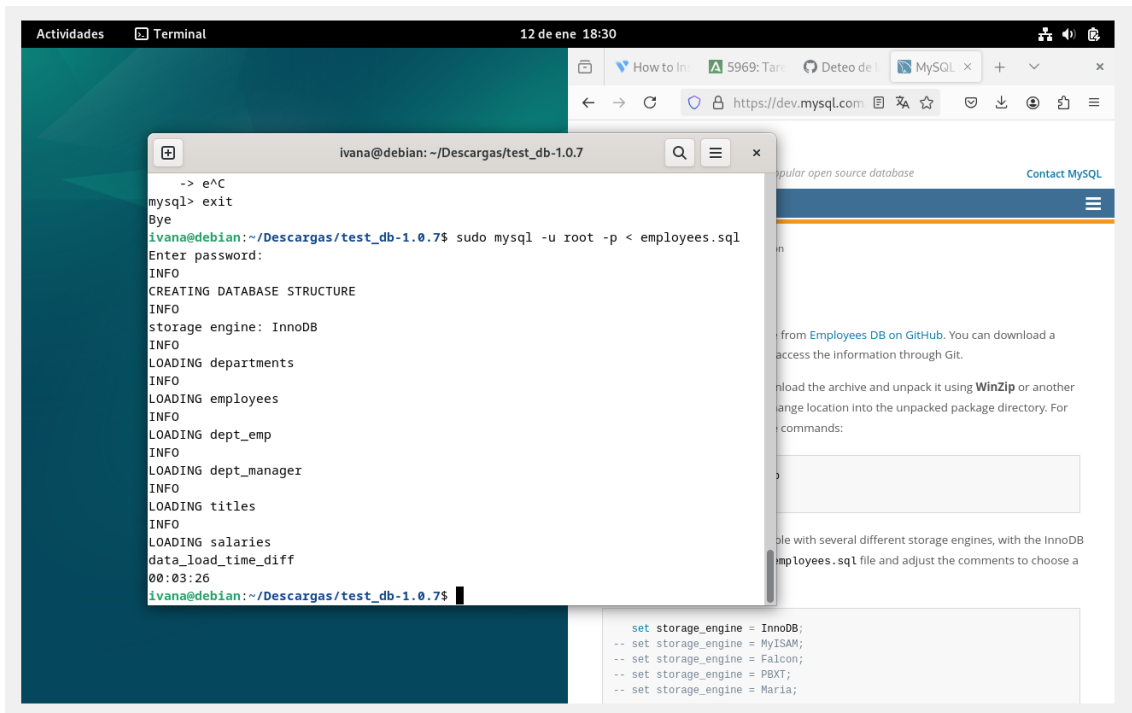


```
ivana@debian: ~/Descargas
ivana@debian:~$ sudo apt install snapd
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  squashfs-tools

Se instalarán los siguientes paquetes NUEVOS:
  snapd squashfs-tools
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 15,1 MB de archivos.
Se utilizarán 61,9 MB de espacio de disco adicional después de esta operación.
Des:1 http://deb.debian.org/debian bookworm/main amd64 squashfs-tools amd64 1:4.5.1-1 [183 kB]
Des:2 http://deb.debian.org/debian bookworm/main amd64 snapd amd64 2.57.6-1+b5 [14,9 MB]
Descargados 15,1 MB en 2s (6.720 kB/s)
Seleccionando el paquete squashfs-tools previamente no seleccionado.
(Leyendo la base de datos ... 155054 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../squashfs-tools_1%3a4.5.1-1_amd64.deb ...
Desempaquetando squashfs-tools (1:4.5.1-1) ...
Seleccionando el paquete snapd previamente no seleccionado.
Preparando para desempaquetar .../snapd_2.57.6-1+b5_amd64.deb ...
```

```
ivana@debian: ~/Descargas
ce - /lib/systemd/system/snapd.apparmor.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.recovery-chooser-trigger.service -> /lib/systemd/system/snapd.recovery-chooser-trigger.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.seeded.service -> /lib/systemd/system/snapd.seeded.service.
Created symlink /etc/systemd/system/cloud-final.service.wants/snapd.seeded.service -> /lib/systemd/system/snapd.seeded.service.
Unit /lib/systemd/system/snapd.seeded.service is added as a dependency to a non-existent unit cloud-final.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.service -> /lib/systemd/system/snapd.service.
Created symlink /etc/systemd/system/sockets.target.wants/snapd.socket -> /lib/systemd/system/snapd.socket.
Procesando disparadores para gnome-menus (3.36.0-1.1) ...
Procesando disparadores para man-db (2.11.2-2) ...
Procesando disparadores para dbus (1.14.10-1~deb12u1) ...
Procesando disparadores para mailcap (3.70+nmul) ...
Procesando disparadores para desktop-file-utils (0.26-1) ...
ivana@debian:~$ cd Descargas/
bash: cd: Descargas/: No existe el fichero o el directorio
ivana@debian:~/Descargas$ sudo snap install mysql-workbench-community
2025-01-12T18:56:35+01:00 INFO Waiting for automatic snapd restart...
```

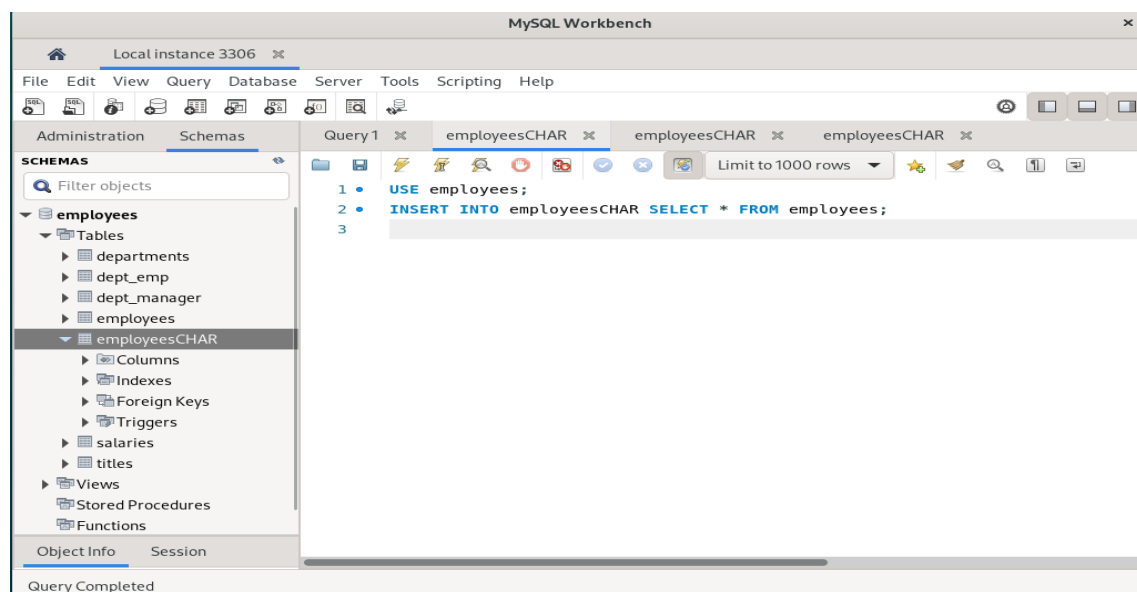
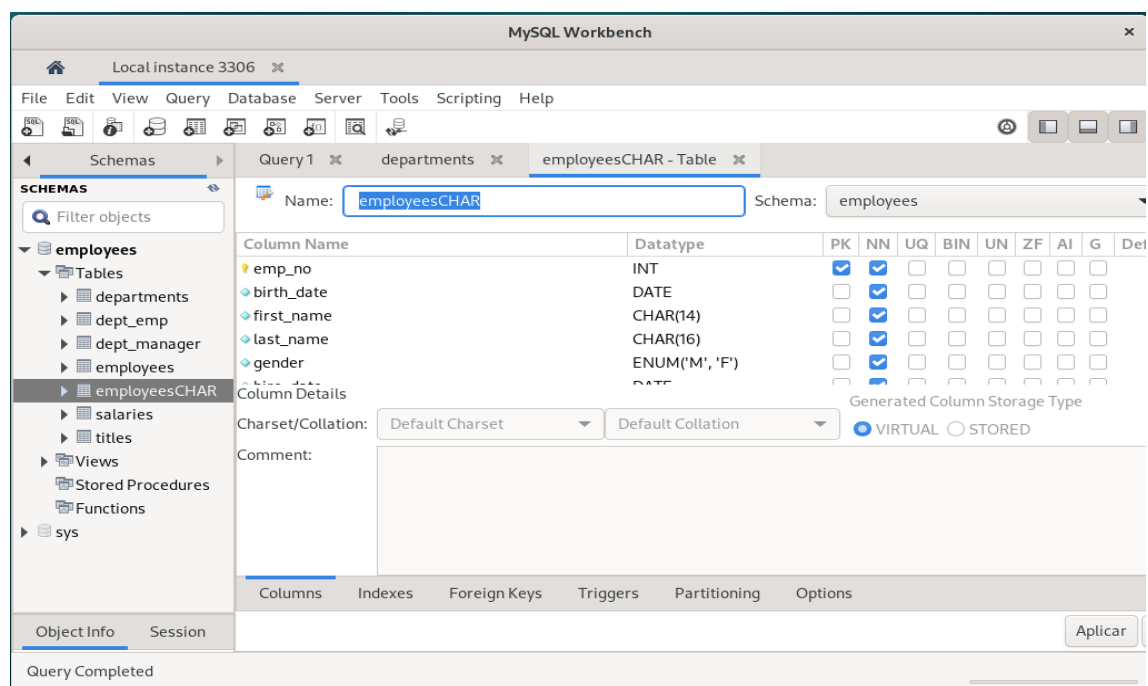
Una vez instalados, procedemos a descargarnos la base de datos de prueba que MySQL ha puesto a disposición en su web, llamada employees.sql, y que descargaremos en el terminal y comprobaremos que se ha creado a la misma vez en Workbench.

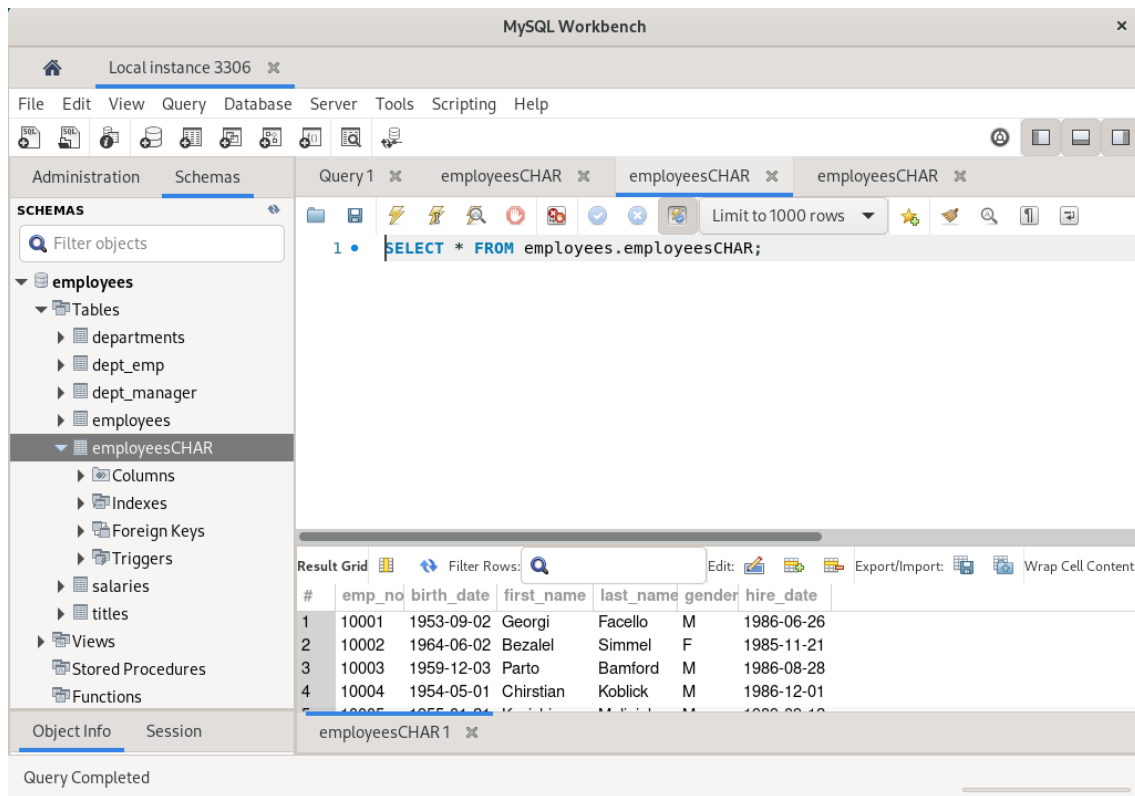


### 3.- Apartado 2

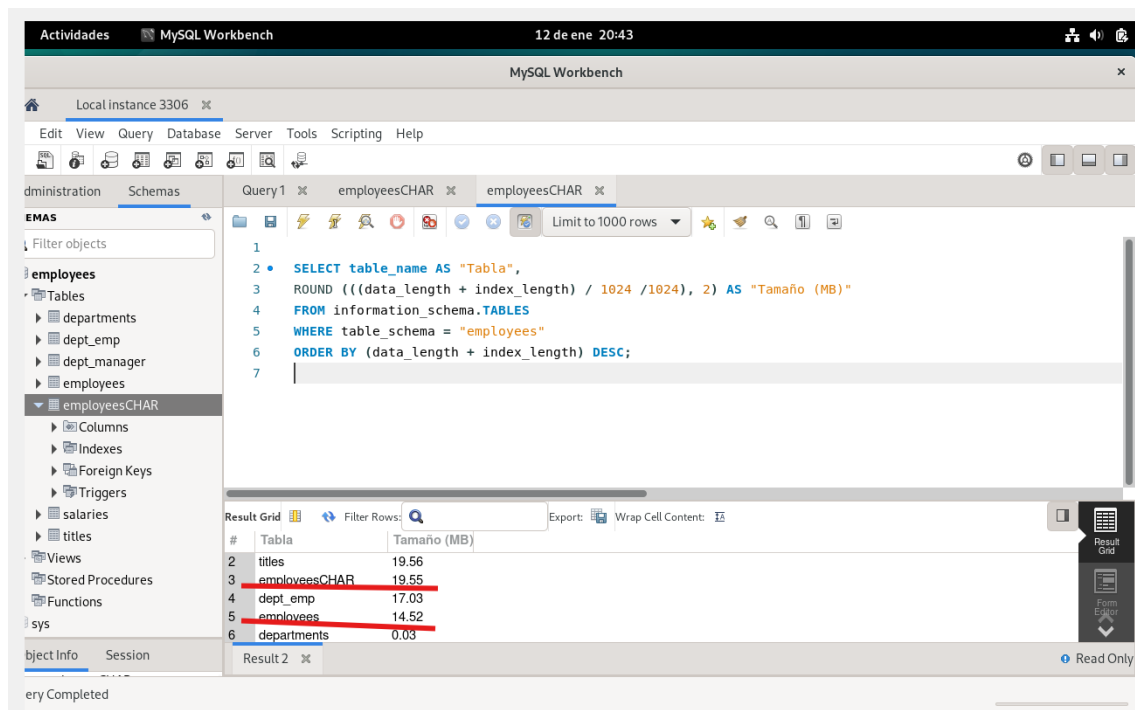
En este apartado trataremos la optimización del espacio en disco, y para ello empezaremos creando una nueva tabla llamada employeesCHAR.

CHAR y VARCHAR son tipos de datos que pueden almacenar cadenas de caracteres de diferentes longitudes. La diferencia entre ambas estriba en la forma en que asignan y manejan el espacio para las cadenas. CHAR significa cadena de caracteres de longitud fija, mientras que VARCHAR significa cadena de caracteres de longitud variable.





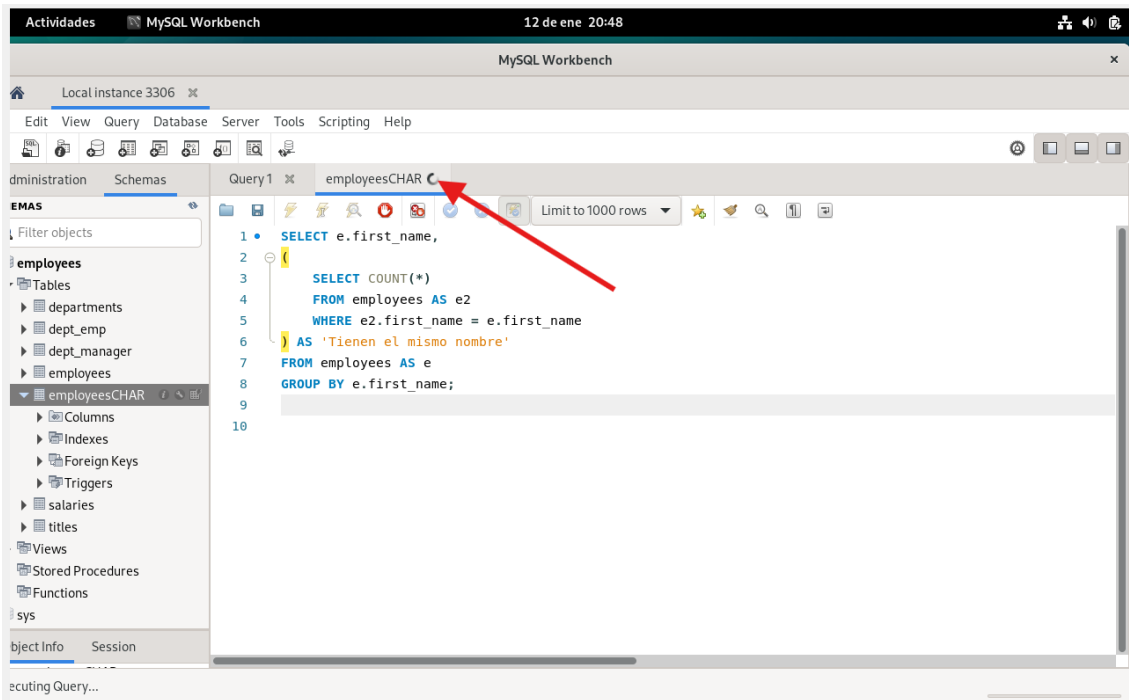
Se observará que hay una diferencia de aproximadamente 5MB entre las dos tablas.



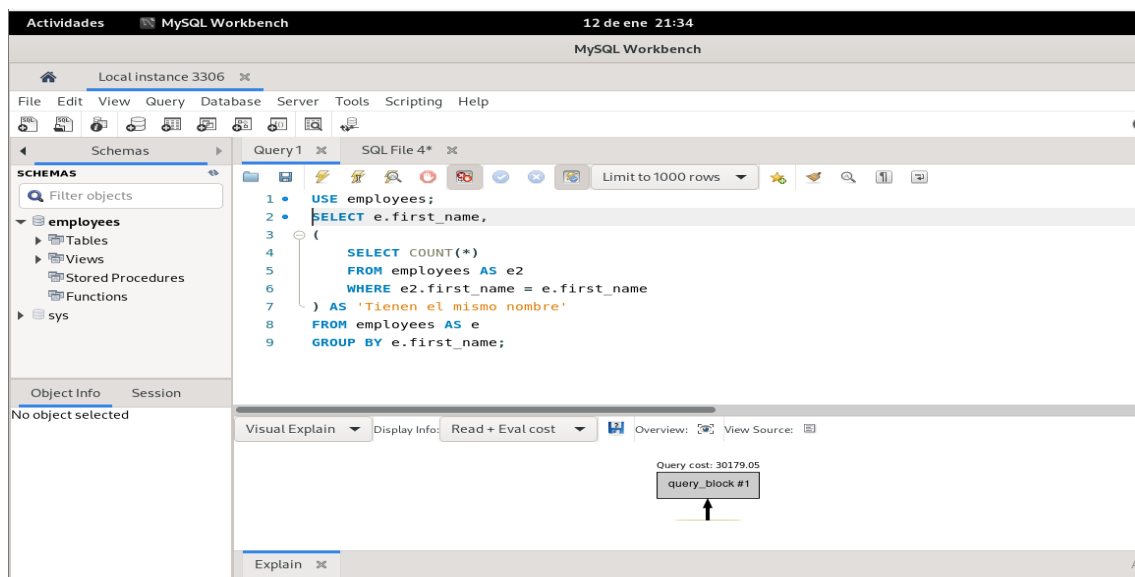
## 4.- APARTADO 3

Apartado que trata de la optimización de consultas a través de la sentencia EXPLAIN.

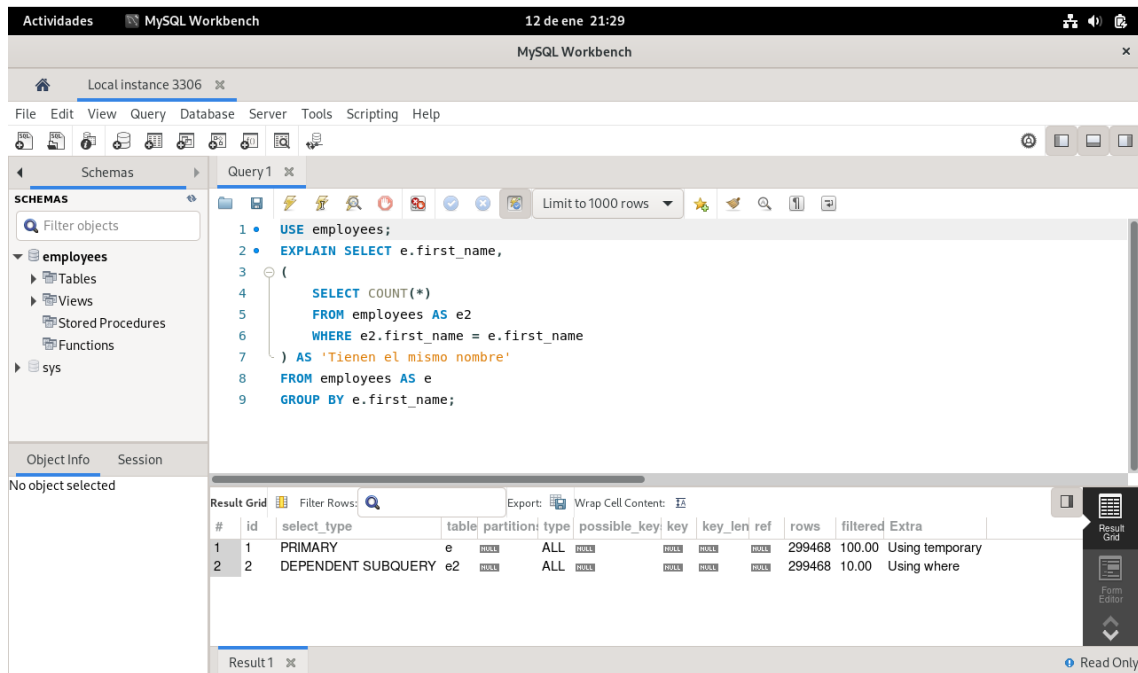
Primero hacemos una consulta normal pidiendo un listado con los nombres de los empleados y la cantidad de ellos que tienen el mismo nombre (Consulta pesada).



El tiempo de ejecución se dispara a los 1000 segundos que tenemos por defecto como límite. El resultado final es un error.



Aplicaremos la sentencia EXPLAIN delante de SELECT y observamos que no sólo no da error, sino que da un resultado inmediato.



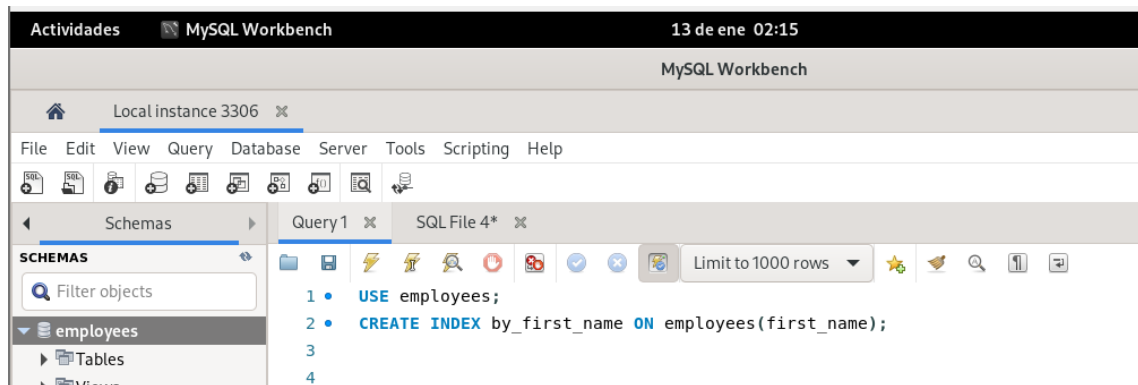
The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
1 • USE employees;  
2 • EXPLAIN SELECT e.first_name,  
3 (  
4     SELECT COUNT(*)  
5     FROM employees AS e2  
6     WHERE e2.first_name = e.first_name  
7 ) AS 'Tienen el mismo nombre'  
8 FROM employees AS e  
9 GROUP BY e.first_name;
```

The 'Result Grid' at the bottom displays the execution plan:

#	id	select_type	table	partition	type	possible_key	key	key_len	ref	rows	filtered	Extra
1	1	PRIMARY	e		ALL					299468	100.00	Using temporary
2	2	DEPENDENT SUBQUERY	e2		ALL					299468	10.00	Using where

Ahora vamos a crear un índice y ejecutamos una consulta. Esto hará que ésta se ejecute en menos de 1 segundo.



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

```
1 • USE employees;  
2 • CREATE INDEX by_first_name ON employees(first_name);  
3  
4
```

The 'Schemas' panel on the left shows the 'employees' database selected.



Actividades MySQL Workbench 13 de ene 02:57

MySQL Workbench

Local instance 3306

File Edit View Query Database Server Tools Scripting Help

Schemas Query1 departments employees Limit to 1000 rows

1 • SHOW INDEX FROM employees;  
2  
3

**SCHEMAS**

Filter objects

employees

- Tables
  - departments
  - dept\_emp
  - dept\_manager
  - employees
- Columns
- Indexes
- Foreign Keys

Object Info Session

Connection Details  
Name: Local instance 3306  
Host: localhost  
Port: 3306  
Login User: root  
Current User: root@localhost  
SSL cipher: SSL not used  
Server

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

#	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packe
1	employees	0	PRIMARY	1	emp_no	A	299468		
2	employees	1	by_first_name	1	first_name	A	1282		

Result 7

Hacemos una consulta con el índice.

Actividades MySQL Workbench 13 de ene 02:59

MySQL Workbench

Local instance 3306

File Edit View Query Database Server Tools Scripting Help

Schemas

SCHEMAS

Filter objects

employees

Tables

departments

dept\_emp

dept\_manager

employees

Columns

Indexes

Foreign Keys

Object Info Session

Connection Details

Name: Local instance 3306

Host: localhost

Port: 3306

Login User: root

Current User: root@localhost

SSL cipher: SSL not used

Server

Query1

departments

employees

Limit to 1000 rows

```

1 • SELECT e.first_name,
2 (
3     SELECT COUNT(*)
4     FROM employees AS e2
5     WHERE e2.first_name = e.first_name
6 ) AS 'Tienen el mismo nombre'
7 FROM employees AS e
8 GROUP BY e.first_name;
9
10
11
12

```

Result Grid

Filter Rows:

Export: Wrap Cell Content: Fetch rows:

#	first_name	Tienen el mismo nombre
1	Aamer	228
2	Aamod	216
3	Abdelaziz	227
4	Abdelghani	247
5	Abdelkader	222
6	Abdelwaheb	241

Result 1

Query Completed

CTRL DERECHA

Y una segunda consulta con EXPLAIN

Actividades MySQL Workbench 13 de ene 03:01

MySQL Workbench

Local instance 3306

File Edit View Query Database Server Tools Scripting Help

Schemas

SCHEMAS

Filter objects

employees

Tables

departments

dept\_emp

dept\_manager

employees

Columns

Indexes

Foreign Keys

Object Info Session

Connection Details

Name: Local instance 3306

Host: localhost

Port: 3306

Login User: root

Current User: root@localhost

SSL cipher: SSL not used

Server

Query1

departments

employees

Limit to 1000 rows

```

1 • EXPLAIN SELECT e.first_name,
2 (
3     SELECT COUNT(*)
4     FROM employees AS e2
5     WHERE e2.first_name = e.first_name
6 ) AS 'Tienen el mismo nombre'
7 FROM employees AS e
8 GROUP BY e.first_name;
9
10
11
12

```

Result Grid

Filter Rows:

Export: Wrap Cell Content:

#	id	select_type	table	partition	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	PRIMARY	e		range	by_first_name	by_first_name	58		1283	100.00	Using index for group-by
2	2	DEPENDENT SUBQUERY	e2		ref	by_first_name	by_first_name	58	func	233	100.00	Using index

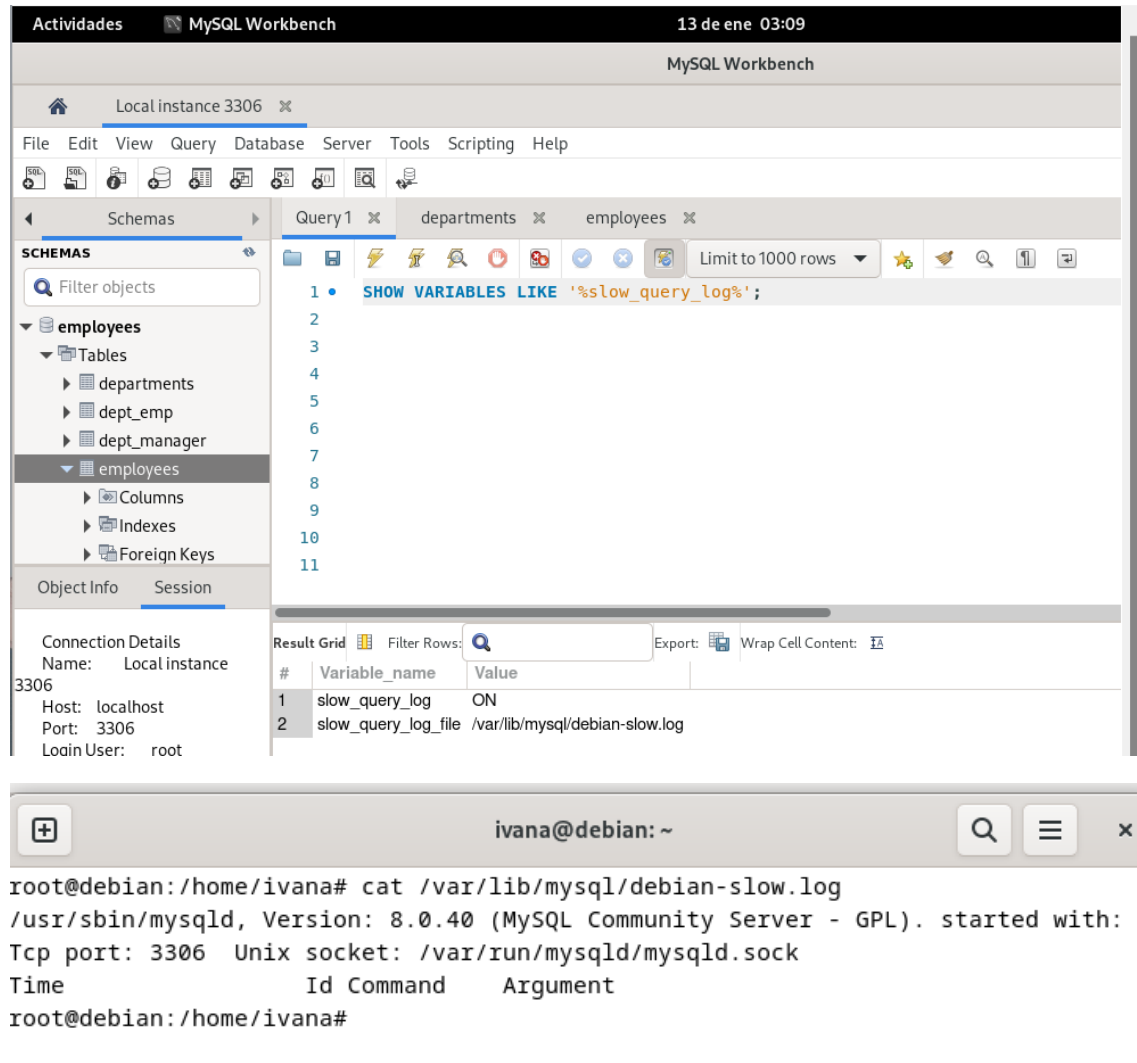
Result 2

Read Only

## 5.- Apartado 4

Fichero log de consultas lentas, es decir, fichero en el que se registrarán las consultas que hayan sobrepasado el límite de tiempo establecido.

Procedo a averiguar si tengo activado dicho fichero, y su ruta.



The image shows two screenshots. The top screenshot is of the MySQL Workbench interface. The 'Schemas' pane on the left shows the 'employees' database selected. The 'Query' editor in the center contains the SQL command: `SHOW VARIABLES LIKE '%slow_query_log%';`. The 'Result Grid' at the bottom right displays the output of this query:

#	Variable_name	Value
1	slow_query_log	ON
2	slow_query_log_file	/var/lib/mysql/debian-slow.log

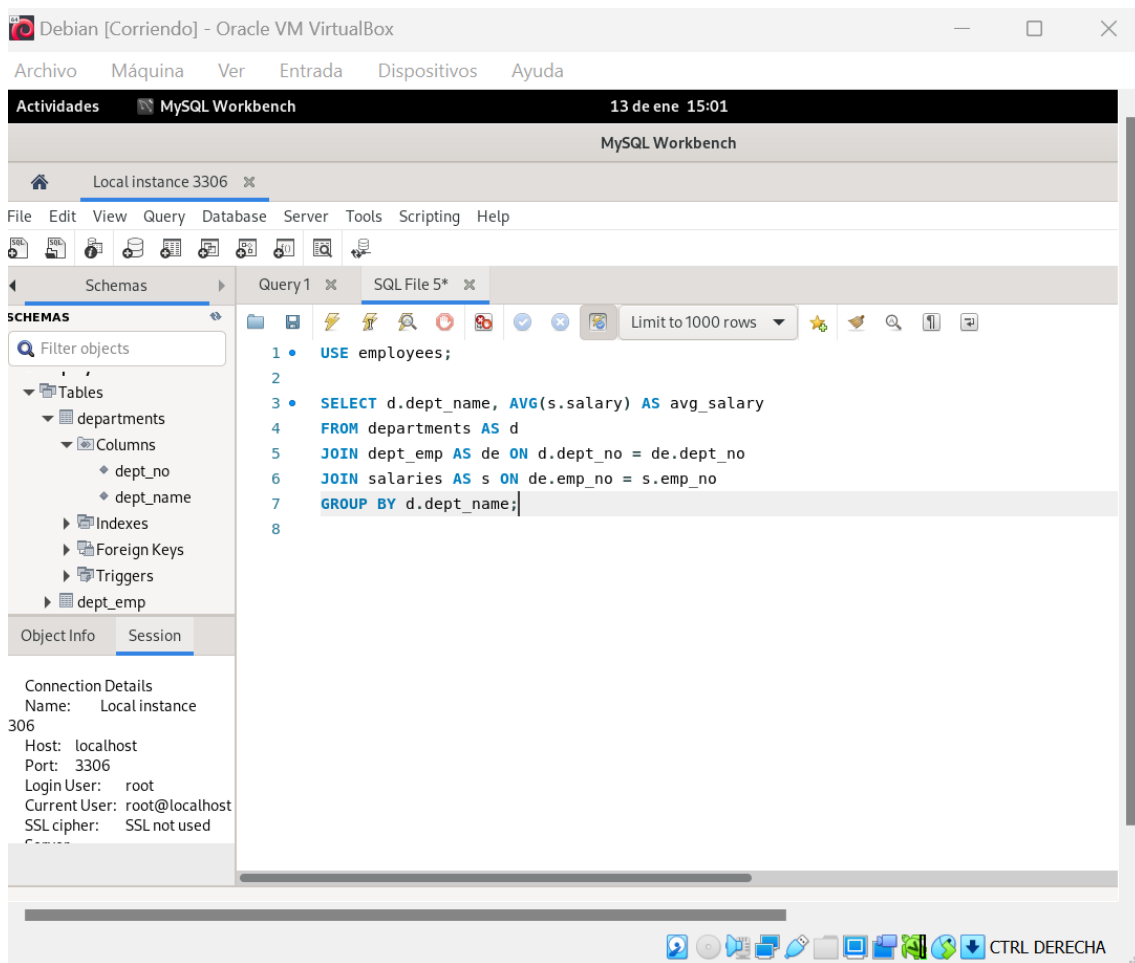
The bottom screenshot is a terminal window titled 'ivana@debian: ~'. It shows the command `cat /var/lib/mysql/debian-slow.log` being executed, which displays the MySQL startup log. The log indicates that the MySQL server (Version: 8.0.40) started with TCP port 3306 and the Unix socket `/var/run/mysqld/mysqld.sock`. The log also shows the 'slow\_query\_log' variable is set to 'ON' and the 'slow\_query\_log\_file' is `/var/lib/mysql/debian-slow.log`.

```
ivana@debian: ~  
# Query_time: 0.000768 Lock_time: 0.000007 Rows_sent: 0 Rows_examined: 0  
SET timestamp=1736768302;  
SELECT st.* FROM performance_schema.events_stages_history_long st WHERE st.nesting_event_id = 54;  
# Time: 2025-01-13T11:38:22.441999Z  
# User@Host: root[root] @ localhost [::1] Id: 9  
# Query_time: 0.000967 Lock_time: 0.000005 Rows_sent: 0 Rows_examined: 0  
SET timestamp=1736768302;  
SELECT st.* FROM performance_schema.events_waits_history_long st WHERE st.nesting_event_id = 54;  
# Time: 2025-01-13T11:38:32.126766Z  
# User@Host: root[root] @ localhost [::1] Id: 10  
# Query_time: 3.877829 Lock_time: 0.000009 Rows_sent: 1000 Rows_examined: 2844047  
use employees;  
SET timestamp=1736768308;  
SELECT s.salary,  
  
COUNT(*) AS 'Número de empleados con este título'  
FROM salaries AS s  
GROUP BY s.salary  
LIMIT 0, 1000;  
# Time: 2025-01-13T11:42:46.925057Z  
# User@Host: root[root] @ localhost [::1] Id: 9  
# Query_time: 0.000262 Lock_time: 0.000002 Rows_sent: 0 Rows_examined: 0  
SET timestamp=1736768566;  
SELECT st.* FROM performance_schema.events_stages_history_long st WHERE st.nesting_event_id = 71;  
# Time: 2025-01-13T11:42:46.926216Z  
# User@Host: root[root] @ localhost [::1] Id: 9  
# Query_time: 0.000235 Lock_time: 0.000001 Rows_sent: 0 Rows_examined: 0  
SET timestamp=1736768566;  
SELECT st.* FROM performance_schema.events_waits_history_long st WHERE st.nesting_event_id = 71;  
root@debian:/home/ivana#
```

## 6.- Apartado 5

Para este apartado voy a centrarme en la tabla salaries para obtener el salario promedio de los empleados de cada departamento.

a) Consulta inicial



## b) Consulta con el comando EXPLAIN

The screenshot shows the MySQL Workbench interface with the same SQL query as before, but with the `EXPLAIN` command added at the beginning of the `SELECT` statement:

```

1 • USE employees;
2
3 • EXPLAIN SELECT d.dept_name, AVG(s.salary) AS avg_salary
4 FROM departments AS d
5 JOIN dept_emp AS de ON d.dept_no = de.dept_no
6 JOIN salaries AS s ON de.emp_no = s.emp_no
7 GROUP BY d.dept_name;
8

```

The 'Result Grid' pane at the bottom displays the execution plan for the query. The table below represents the data shown in the grid:

#	id	select_type	table	partition	type	possible_keys	key	key_len	ref	rows	filtered
1	1	SIMPLE	de	NULL	index	PRIMARY,dept_no	dept_no	16	NULL	331143	100
2	1	SIMPLE	d	NULL	eq_ref	PRIMARY,dept_name	PRIMARY	16	employees.de.dept_no	1	100
3	1	SIMPLE	s	NULL	ref	PRIMARY	PRIMARY	4	employees.de.emp_no	9	100

The 'Connection Details' pane on the left remains the same as in the previous screenshot.

### c) Creación de un índice

The first screenshot shows the MySQL Workbench interface with the following SQL queries in the editor:

```
1 • USE employees;
2 • CREATE INDEX idx_emp_no ON salaries(emp_no);
3
4
```

The second screenshot shows the same interface after running the queries. The SQL editor now contains:

```
1 • show index FROM salaries;
```

The bottom panel displays the 'Result Grid' with the following data:

#	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
1	salaries	0	PRIMARY	1	emp_no	A	294479				BTREE
2	salaries	0	PRIMARY	2	from_date	A	2838426				BTREE
3	salaries	1	idx_emp_no	1	emp_no	A	299318				BTREE

The left sidebar shows the 'Schemas' tree with 'departments' expanded, and the 'Object Info' tab is selected.

Debian [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Actividades MySQL Workbench 13 de ene 15:06

MySQL Workbench

Local instance 3306

File Edit View Query Database Server Tools Scripting Help

Schemas

Filter objects

Tables

- departments
  - Columns
    - dept\_no
    - dept\_name
  - Indexes
  - Foreign Keys
  - Triggers
  - dept\_emp

Object Info Session

Connection Details

Name: Local instance 3306

Host: localhost

Port: 3306

Login User: root

Current User: root@localhost

SSL cipher: SSL not used

Query1

SQL File 5\*

SQL File 6\*

Limit to 1000 rows

```

1 • USE employees;
2
3 • EXPLAIN SELECT d.dept_name, AVG(s.salary) AS avg_salary
4 FROM departments AS d
5 JOIN dept_emp AS de ON d.dept_no = de.dept_no
6 JOIN salaries AS s ON de.emp_no = s.emp_no
7 GROUP BY d.dept_name;
8

```

Result Grid

#	id	select_type	table	partition	type	possible_keys	key	key_len	ref	rows	filtered
1	1	SIMPLE	de	NULL	index	PRIMARY,dept_no	dept_no	16	NULL	331143	10
2	1	SIMPLE	d	NULL	eq_ref	PRIMARY,dept_name	PRIMARY	16	employees.de.dept_no	1	10
3	1	SIMPLE	s	NULL	ref	PRIMARY,idx_emp_no	PRIMARY	4	employees.de.emp_no	9	10

Result 4

CTRL DERECHA

ivana@debian: ~

```

# Time: 2025-01-13T14:02:03.348889Z
# User@Host: root[root] @ localhost [::1] Id: 9
# Query_time: 11.524787 Lock_time: 0.000114 Rows_sent: 9 Rows_examined: 380530
1
SET timestamp=1736776911;
SELECT d.dept_name, AVG(s.salary) AS avg_salary
FROM departments AS d
JOIN dept_emp AS de ON d.dept_no = de.dept_no
JOIN salaries AS s ON de.emp_no = s.emp_no
GROUP BY d.dept_name
LIMIT 0, 1000;
# Time: 2025-01-13T14:05:19.293790Z
# User@Host: root[root] @ localhost [::1] Id: 8
# Query_time: 0.000200 Lock_time: 0.000001 Rows_sent: 0 Rows_examined: 0
SET timestamp=1736777119;
SELECT st.* FROM performance_schema.events_stages_history_long st WHERE st.nesti
ng_event_id = 148;
# Time: 2025-01-13T14:05:19.295047Z
# User@Host: root[root] @ localhost [::1] Id: 8
# Query_time: 0.000178 Lock_time: 0.000000 Rows_sent: 0 Rows_examined: 0
SET timestamp=1736777119;
SELECT st.* FROM performance_schema.events_waits_history_long st WHERE st.nestin
g_event_id = 148;
root@debian: /home/ivana#
ost: localhost
ort: 3306
gin User: root

```

#	id	select_type	table	partition	type	possible_keys	key	key_len	ref	rows	filtered
2	1	SIMPLE	d	NULL	eq_ref	PRIMARY,dept_name	PRIMARY	16	employees.		
3	1	SIMPLE	s	NULL	ref	PRIMARY,idx_emp_no	PRIMARY	4	employees.		

