

# MYSQL

By Kessusa

## Parte I: Copias de seguridad

### 1. mysqldump es la herramienta de creación de backups. Indique para qué sirven las opciones:

**--add-drop-table:** esta función lo que hace es incluir la instrucción drop table antes de cada instrucción create table de la copia que realicemos , de esta manera antes de restaurar la tabla , se eliminaría la tabla si existiese con el mismo nombre.

**--add-locks:** esta opción se aplica cuando se está importando el archivo de backup(no mientras se ejecuta mysqldump).Cada volcado de tabla se rodea con las instrucciones LOCK TABLES y UNLOCK TABLES , como resultado se obtienen inserciones más rápidas cuando se vuelve a cargar el archivo de backup.

**--complete-insert:** las sentencias de INSERTS se hacen completas incluyendo el nombre de la columna.

**--create-options:** Incluye todas las opciones específicas de MYsql para la creación de las tablas que usan la sentencia CREATE TABLE.

**--disable-keys:** Para cada tabla, rodea las instrucciones INSERT con **/ \*! 40000 ALTER TABLE tbl\_name DISABLE KEYS \* /;** y **/ \*! 40000 ALTER TABLE tbl\_name ENABLE KEYS \* /;** sentencias. Esto consigue que la carga del archivo backup sea mas rapida porque los índices se crean después de insertar todas las filas , aunque esta opción solo está disponible para índices no únicos de tablas MyISAM.

**--extended-insert:** Utiliza la sintaxis INSERT de varias filas que incluyen varias listas de valores , de esta manera se consigue el el archivo de volcado sea más pequeño y acelera los inserts cuando el archivo se vuelve a cargar.

**--lock-tables :** bloquea las tablas durante el proceso de copiado , de esta manera no se puede, ni hacer un insert, update ni modificar los datos mientras se esté copiando.

**--quick:** va junto con la opción --single transaction ya que se aplica principalmente par tablas grandes y de tipo transaccional , permite agilizar la escritura al archivo de backup al leer el registro por registro sin mandarlo a un buffer previo.

**--routines:** Incluye en la copia de las bases de datos las rutinas almacenadas (procedimientos y funciones). Con esta función se incluyen las sentencias "CREATE PROCEDURE" Y "CREATE FUNCTION" que permiten recrear completamente procedimientos almacenados y funciones. Esta opción NO está por defecto en mysqldump así que es motivo de sorpresa para DBAs nuevos enterarse que el respaldo en el que tanto confiaban no está completo al momento de necesitarlo, sino utilizaron esta opción.

**--triggers:** Incluye triggers creados en el backup. Esta opción es automática, esta por defecto, si no se quiere utilizar el respaldo de triggers utiliza **--skip-triggers**.

**--set-charset.:** Agrega SET NAMES **default\_character\_set** a la salida.

## 2. Realice una copia de seguridad de tienda Virtual, con independencia de los parámetros anteriores. Indique aquí el formato utilizado

Si queremos generar una copia de seguridad de nuestra base de datos que tenga estructura y datos podemos usar la siguiente sentencia en una terminal , para realizarla.

- ***mysqldump -h localhost -u root -p tiendaVirtual > copia\_completa.sql.***

```
ambite@ambite-VirtualBox:~$ mysqldump -u root -p tiendaVirtual > copia_completa
.sql
Enter password:
```

Ahora vamos a comprobar que se haya realizado la copia correctamente .

```
ambite@ambite-VirtualBox:~$ ls copia_completa.sql
copia_completa.sql
```

## 3. Indique cómo recuperar la base de datos utilizando la copia del apartado anterior. Indique, en todo caso, la sentencia ejecutada.

Ahora con el archivo que he generado anteriormente voy a recuperar en otro Mysql server, la base de datos copiada con el siguiente comando , y claro está teniendo el archivo de copiado.

```
m Archivo Editor Ver Base de Datos Terminal Ayuda
df t@pc-01:~/Descargas$ ls copia_completa.sql
copia_completa.sql
t@pc-01:~/Descargas$
```

Vamos a comprobar que en este servidor yo no tengo la base de datos .

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| Nombres |
| Usuariosregistrados |
| basedatos |
| mysql |
| performance_schema |
| sys |
+-----+
7 rows in set (0.00 sec)
```

Para restaurar la base de datos copiada anteriormente, usamos el siguiente comando, desde una terminal.

- ***mysql -u root -p tiendaVirtual < copia\_completa.sql***

Esto va a ser un problema ya que necesitamos que la base de datos está creada para que este comando funcione .

```
t@pc-01:~/Descargas$ mysql -u root -p tiendaVirtual < copia_completa.sql
mysql Ver 14.14 Distrib 5.7.28, for Linux (x86_64) using EditLine wrapper
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Usage: mysql [OPTIONS] [database]
        -h, --host=host_name      Connect to host_name host.
        -h, --help                Display this help and exit.
        -u, --user=user_name      Connect to user_name user.

mysql> create database tiendaVirtual;
Query OK, 1 row affected (0.00 sec)
```

Y una vez lo hemos creado podemos volver a ejecutar el comando anterior y se realizará la restauración sin problemas .

```
copia_completa.sql
t@pc-01:~/Descargas$ mysql -u root -p tiendaVirtual < copia_completa.sql
Enter password:
t@pc-01:~/Descargas$
```

Vamos a comprobarlo entrando en mysql a ver si esta todo bien copiado.

```
mysql> use tiendaVirtual;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_tiendaVirtual |
+-----+
| categorias               |
| clientes                 |
| companiaTransportes      |
| departamentos            |
| empleados                |
| lineasDePedido            |
| paises                   |
| pedidos                  |
| productos                 |
| proveedores               |
+-----+
10 rows in set (0.01 sec)
```

Como vemos podemos acceder a la base de datos y todo está perfectamente copiado.

4. ¿Cómo podríamos recuperar una base de datos para que, si no existe la base de datos, la cree y, si existe, se recupere la base de datos igualmente? ¿Depende de cómo hayamos realizado (por el uso de los diferentes parámetros) el backup? Indique todos los pasos incluyendo todos los comandos

Antes de empezar voy a borrar la base de datos anteriormente creada con el comando.

- ***drop database tiendaVirtual;***

```
mysql> drop database tiendaVirtual;
Query OK, 10 rows affected (0.07 sec)
```

Ahora vamos a hacer una copia de la base de datos pero con la siguiente opción :

<code>--databases</code>	-	Permite respaldar una o más bases de datos. Después de la opción se indica(n) el(los) nombre(s) de la(s) base de datos a respaldar. Se respalda cada base de datos completa. En la salida se incluye con esta opción las sentencias 'CREATE DATABASE' y 'USE' antes de cada nueva base de datos.
--------------------------	---	--

Con esta opción añadida a la sentencia de mysqldump, hará un backup que cuando lo intentes recuperar, podrás hacerlo sin haber creado previamente la base de datos en el servidor que quieras recuperarlo.

La sentencia será la siguiente:

- ***mysqldump --databases -u root -p tiendaVirtual > copia\_definitiva.sql***

```
ambite@ambite-VirtualBox:~$ mysqldump --databases tiendaVirtual -u root -p > copia_definitiva.sql
Enter password:
ambite@ambite-VirtualBox:~$ ls copia_definitiva.sql
copia_definitiva.sql
```

Ahora vamos a recuperarla a ver si evitamos tener el problema de antes y no necesitamos crear la base de datos inicialmente .

- ***mysql -u root -p < copia\_definitiva.sql***

```
t@pc-01:~/Descargas$ mysql -u root -p < copia_definitiva.sql
Enter password:
```

Comprobamos que el respaldo se ha creado correctamente y como podemos ver en la siguiente imagen todo está correcto.



```
mysql> use tiendaVirtual;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables ;
+-----+
| Tables_in_tiendaVirtual |
+-----+
| categorias               |
| clientes                 |
| companiaTransportes      |
| departamentos            |
| empleados                |
| lineasDePedido            |
| paises                   |
| pedidos                  |
| productos                 |
| proveedores               |
+-----+
10 rows in set (0.00 sec)
```

Pero vamos a ver porque sucede esto es muy fácil mirando los dos archivos de respaldo creados vamos a ver dónde está la diferencia y porque uno permite reponerlo sin crear la base de datos previamente el otro te obliga a tener que crear.

Estas son las primeras 25 líneas del archivo copia completa el primero, que nos obliga a tener cread ya la base de datos para hacer su recuperación .

```
t@pc-01:~/Descargas$ head -n 25 copia_completa.sql
-- MySQL dump 10.13  Distrib 5.7.29, for Linux (x86_64)
--
-- Host: localhost    Database: tiendaVirtual
--
-- Server version      5.7.29-0ubuntu0.18.04.1-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `categorias`
--
DROP TABLE IF EXISTS `categorias`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `categorias` (
```

Estas son las 25 primeras líneas del archivo copia definitiva ya con la opción --databases, veremos el porque aqui no hace falta crear previamente , la base de datos para reponerla .

```
t@pc-01:~/Descargas$ head -n 25 copia_definitiva.sql
-- MySQL dump 10.13 Distrib 5.7.29, for Linux (x86_64)
--
-- Host: localhost    Database: tiendaVirtual
-- -----
-- Server version      5.7.29-0ubuntu0.18.04.1-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `tiendaVirtual`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `tiendaVirtual` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `tiendaVirtual`;
```

Es la última sentencia de la foto la que os importa tenemos **un create database if not exists** , y luego un use que precisamente es lo que os permite o tener que tenerla creada previamente .

## 5. Realice una copia de seguridad de tiendaVirtual que incluya los posibles disparadores, funciones y procedimientos.

Para hacer esto podemos usar la sentencia --routines dentro de mysqldump , que como hemos visto antes nos permite copiar tanto las sentencias function como procedures .

- ***mysqldump --routines -u root -p tiendaVirtual > copia\_rutinas.sql***

```
ambite@ambite-VirtualBox:~$ mysqldump --routines tiendaVirtual -u root -p > co
pia_rutinas.sql
Enter password:
ambite@ambite-VirtualBox:~$ ls copia_rutinas.sql
copia_rutinas.sql
```

Este backup contendría todos los procedimientos almacenados, y funciones que estuviesen disponibles en la base de datos tiendaVirtual.

Como podemos ver en la siguiente foto al final del archivo se haría la copia, de los procedimientos y rutinas almacenadas.

```
--
-- Dumping routines for database 'tiendaVirtual'
--
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2020-02-10 19:40:20
ambito@ambito-VirtualBox:~$
```

## Parte II: Optimización. Índices

6. Para poder continuar con la práctica, es esencial desactivar la caché de consultas, ya que lo que nos interesa es medir solo el rendimiento de los índices.. De esta forma, hay que poner a 0 las siguientes opciones:

Para poner el tamaño de caché de consultas a 0 tenemos que usar el siguiente comando dentro del terminal de mysql.



**Formato de línea de comando** --query-cache-type=#

**Obsoleto** 5.7.20

**Variable de sistema** query\_cache\_type

**Alcance** Global, sesión

**Dinámica** si

**Tipo** Enumeración

**Valor por defecto** 0

**Valores válidos** 0, 1, 2

### **0 o OFF**

No almacena en caché ni recupera resultados de la caché de consultas. Tenga en cuenta que esto no desasigna el búfer de caché de consultas. Para hacer eso, debes establecerlo query\_cache\_size en 0.

### **1 o ON**

Almacena en caché todos los resultados de consultas almacenables en caché, excepto los que comienzan con SELECT SQL\_NO\_CACHE

### **2 o DEMAND**

Resultados de caché sólo para consultas almacenables en caché que comienzan con SELECT SQL\_CACHE.

## **Pasos a seguir para desactivar la caché de preguntas .**

1 Iniciar sesión en la base de datos MySQL como administrador.

2 Escriba la instrucción

SET GLOBAL query\_cache\_size = 0;  
desactivar la caché de consultas.

3 Escriba la instrucción:

SHOW VARIABLES LIKE "query\_cache\_size";

## **Otra manera**

También puede desactivar la caché de consultas cuando se inicia el servidor escribiendo el comando **"mysqld --query\_cache\_size = 0" en el símbolo del sistema.**

Primero de todo vamos a comprobar como esta por defecto en nuestro servidor el tamaño de caché para las consultas .

Para ello ejecutamos el siguiente comando, en la terminal mysql.

- **mysql> show variables like "query\_cache\_size";**

```
mysql> show variables like "query_cache_size";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| query_cache_size | 16777216 |
+-----+-----+
1 row in set (0.02 sec)
```

Como podemos ver el valor no se encuentra a cero por defecto así que vamos a tener que ponerlo nosotros mano con el siguiente comando.

De nuevo desde dentro de la terminal mysql.

- **SET GLOBAL query\_cache\_size = 0;**

```
mysql> set global query_cache_size=0;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

**Para asegurarse de que el tamaño de caché se establece en cero.**

Vamos a comprobar si ha surtido efecto con el siguiente comando.

- **mysql> show variables like "query\_cache\_size";**

```
mysql> show variables like "query_cache_size";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| query_cache_size | 0 |
+-----+-----+
1 row in set (0.00 sec)
```

Para ver cómo han quedado todas las variables de query cache usamos.

- **mysql> show variables like '%query\_cache%';**

```
mysql> show variables like '%query_cache%';
```

Variable_name	Value
have_query_cache	YES
query_cache_limit	1048576
query_cache_min_res_unit	4096
query_cache_size	0
query_cache_type	OFF
query_cache_wlock_invalidate	OFF

```
6 rows in set (0.00 sec)
```

**NOTA IMPORTANTE DESPUÉS DE HACER CONSULTAS CON LA CACHÉ DE CONSULTAS SUPUESTAMENTE DESACTIVADA CON EL TAMAÑO A 0 ME DI CUENTA DE QUE HACIENDO LAS CONSULTAS LA CACHÉ NO SE DESACTIVABA, YA QUE CUANDO REPETÍA LAS CONSULTAS SIEMPRE ME BAJABA EL TIEMPO DE EJECUCIÓN. QUERY CACHE TYPE 2 QUE LO HACE BAJO DEMANDA, SI NO LE ESPECIFICAS TU QUE GUARDE LA CONSULTA EN CACHÉ NO LA DEBERÍA GUARDAR.**

```
mysql> mysql> show variables like '%query_cache%';
```

Variable_name	Value
have_query_cache	YES
query_cache_limit	1048576
query_cache_min_res_unit	4096
query_cache_size	16777216
query_cache_type	DEMAND
query_cache_wlock_invalidate	OFF

```
6 rows in set (0.00 sec)
```

Puedes asegurar que no está el cache operativo si al hacer una consulta un poco compleja, vemos el tiempo que ha tardado y si la repetimos a continuación, el tiempo es el mismo, quiere decir que no está usando la caché, si el tiempo baja significativamente, estará consultando el resultado de la consulta en caché.

- Indique la estructura de la base de datos solicitada en clase. Para ello, realice una copia de seguridad de la estructura de la base de datos. Posteriormente, pegue aquí su contenido.

No importa los nombres de las tablas, como tampoco los nombres de las columnas, pero sí es necesario poder reconocer cada una de ellas. A modo de ejemplo, podría ser:

**tabla\_A (con 1 millón de registros)**

- **ald:** Clave primaria. Entero auto incremental.
- **aCadenaUnica:** Clave única. Cadena de al menos 22 caracteres. Contiene valores aleatorios.
- **aCadenaIndex:** No nulos. Se le aplica un índice. Cadena con valores aleatorios (se pueden repetir).
- **aCadenaSinIndex:** No nulos. Contiene los mismos datos que **aCadenaIndex**, pero no se le aplica ningún índice.
- **aldClaveAjena:** clave ajena. Contiene valores existentes de la clave primaria de “tabla\_B”.

**tabla\_B (con al menos 100 registros)**

- **bld:** Clave primaria. Autoincremental.
- **bCadenaUnica:** Clave única. Contiene valores aleatorios.
- **bCadenaIndex:** No nulos. Se le aplica un índice. Cadena con valores aleatorios (se pueden repetir).
- **bCadenaSinIndex:** No nulos. Contiene los mismos datos que **bCadenaIndex**, pero no se le aplica ningún índice.

He creado todo y ahora he hecho un backup de solo la información de creación de la base de datos sus tablas .

Con la siguiente sentencia:

**- *mysqldump -u root -p --no-data gdb trabajo > gbi trabajo.sql***

```
t@pc-01:~$ mysqldump -u root -p --no-data gdbtrabajo > gdbtrabajo.sql
Enter password:
t@pc-01:~$ ls gdbtrabajo.sql
gdbtrabajo.sql
```

Esta es la estructura con las sentencias de creación .

```
-- MySQL dump 10.13  Distrib 5.7.28, for Linux (x86_64)
--
-- Host: localhost    Database: gbd trabajo
--
-- Server version      5.7.28-0ubuntu0.18.04.4
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
--
-- Table structure for table `tabla_A`
--
DROP TABLE IF EXISTS `tabla_A`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tabla_A` (
  `A_id` int(11) NOT NULL AUTO_INCREMENT,
  `Cadenaunica` varchar(30) DEFAULT NULL,
  `Cadenaindex` varchar(22) NOT NULL,
  `aCadenaSinindex` varchar(22) NOT NULL,
  `aidclaveajena` int(11) NOT NULL,
  PRIMARY KEY (`A_id`),
  UNIQUE KEY `aCadenaunica_UNIQUE` (`aCadenaunica`),
  KEY `fk_tabla_A_1_idx` (`aidclaveajena`),
  KEY `index4` (`aCadenaindex`),
  CONSTRAINT `fk_tabla_A_1` FOREIGN KEY (`aidclaveajena`) REFERENCES `tabla_B`
  (`idB`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Table structure for table `tabla_B`
--
DROP TABLE IF EXISTS `tabla_B`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tabla_B` (
  `idB` int(11) NOT NULL AUTO_INCREMENT,
  `bCadenaunica` varchar(45) DEFAULT NULL,
  `bCadenaindex` varchar(45) NOT NULL,
  `bCadenaSinindex` varchar(45) NOT NULL,
  PRIMARY KEY (`idB`),
  UNIQUE KEY `bCadenaunica_UNIQUE` (`bCadenaunica`),
  KEY `index3` (`bCadenaindex`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```



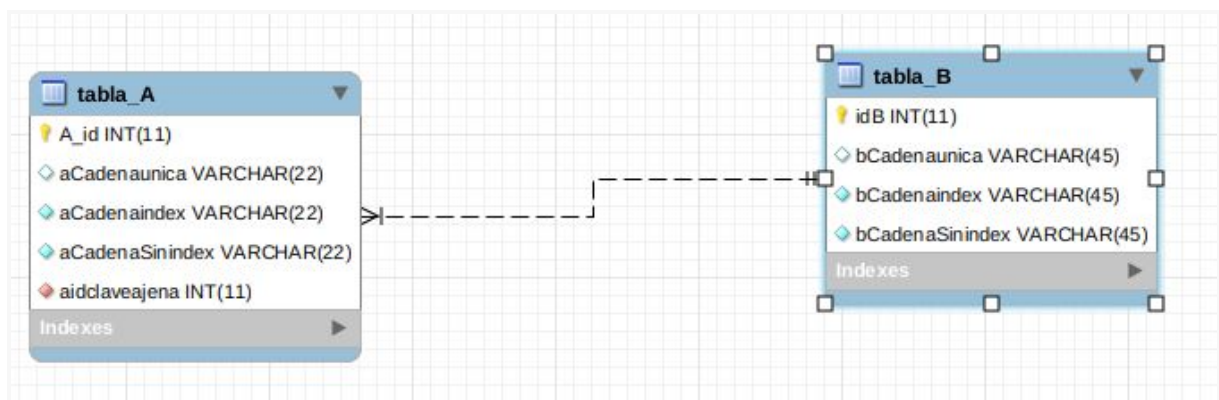
#14

```
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2020-02-10 20:38:28
```

En la siguiente foto podemos ver que están unidas las tablas por la clave primaria de la tabla 2, como queríamos .



Lo siguiente que vamos a hacer es llenar las tablas con los datos que nos han pedido.

8. Usando los “compound statements”<sup>1</sup>, Indique el script generado para insertar los datos de la tabla “tabla\_B” (recuerde que “tabla\_A” contiene una clave ajena apuntando a la clave primaria de “tabla\_B”).

```
CREATE DEFINER='root'@'localhost' PROCEDURE `insertar_tabla`()

BEGIN

declare bCadenaunica varchar(22);

declare bCadenaindex varchar(22);

declare bCadenaSinindex varchar(22);

declare contador int;
```

<sup>1</sup> <https://dev.mysql.com/doc/refman/5.7/en/sql-compound-statements.html>

```

set contador=1;

while (contador <=100) do

    set bCadenaunica := lpad(conv(floor(rand()*pow(36,8)), 10, 36), 8, 0);

    set bCadenaindex:= lpad(conv(floor(rand()*pow(36,8)), 10, 36), 8, 0);

    insert into tabla_B(idtabla_B,bCadenaunica,bCadenaindex,bCadenaSinindex)
values(`contador`,`bCadenaunica`,`bCadenaindex`,`bCadenaindex`);

set contador=contador + 1;

end while;

END

```

```

CREATE DEFINER='root'@'localhost' PROCEDURE `insertar_tabla`()
BEGIN
    declare bCadenaunica varchar(22);
    declare bCadenaindex varchar(22);
    declare bCadenaSinindex varchar(22);
    declare contador int;
    set contador=1;
    while (contador <=100) do
        set bCadenaunica := lpad(conv(floor(rand()*pow(36,8)), 10, 36), 8, 0);
        set bCadenaindex:= lpad(conv(floor(rand()*pow(36,8)), 10, 36), 8, 0);
        insert into tabla_B(idtabla_B,bCadenaunica,bCadenaindex,bCadenaSinindex) values(`contador`,`bCadenaunica`,`bCadenaindex`,`bCadenaindex`);
        set contador=contador + 1;
    end while;
END

```

Este procedimiento le hice después de montar toda la base de datos así que los datos que he cargado son diferentes a los que usare en la práctica aunque ahora me arrepiento, ya que me parece mas comoda esta forma de meter los datos.

Vamos a ver que en generar e insertar esto 100 datos ha tardado 0.123 segundos.

Time	Action	Message	Duration / Fetch
20:45:47	call insertar_tabla()	1 row(s) affected	0.124 sec
20:45:51	SELECT * FROM prueba.tabla_B LIMIT 0, 1000	100 row(s) returned	0.00042 sec / 0.000...

Ahora vamos a ver los datos que hemos creado e insertado en la tabla, para ver si tienen la estructura que queríamos darle para la tabla.

```
mysql> select * from tabla_B;
```

Database changed

```
mysql> select * from tabla_B;
```

idtabla_B	bCadenaunica	bCadenaindex	bCadenaSinindex
1	86K0BVXZ	M2E51GKJ	M2E51GKJ
2	DS809HX4	2Q0Y8YSH	2Q0Y8YSH
3	89EQI73E	X4YTTY00	X4YTTY00
4	WWLX0ZMA	T4570YZU	T4570YZU
5	AUW63R0P	2Y19HP0F	2Y19HP0F
6	I5HT72WG	9XD9K6DE	9XD9K6DE
7	V6CW9I21	I3000U2E	I3000U2E
8	WZCQPJ2A	2LPNJ00N	2LPNJ00N
9	M2I1K9SU	UJD9A7U0	UJD9A7U0
10	EHC5S4R9	GSL13V40	GSL13V40
11	4IW08SA1	88S9Y6WK	88S9Y6WK
12	RN7D2GL5	5HNZJL4I	5HNZJL4I
13	GIPUKERJ	U4LA95CL	U4LA95CL
14	T2SVMDCR	J08JE9MG	J08JE9MG
15	7SS262YF	HZ60PGT5	HZ60PGT5
16	UHL92Y2Y	QIE7C6VN	QIE7C6VN
17	5379HZ1V	HWTPJ5IT	HWTPJ5IT
18	2AIR7PCW	TQ4NIX4G	TQ4NIX4G
19	XP301CC0	ZL11NTR2	ZL11NTR2

Vemos como, tenemos las tres columnas perfectas y son datos únicos, menos en la tabla Sinindex e index, que son los mismo datos en ambas.

**9. Refleje el script Bash para generar el millón de registros de “tabla\_A” para poder volcar los datos con load data<sup>2</sup>. Indique qué dificultades ha ido encontrando y cuánto ha tardado en generar dichos valores.**

Yo modifique los archivos, para dejarlos en un formato estilo .csv separado por “;” y después use los siguientes comandos.

Primero meti los datos de la tabla\_B como es evidente dado que la tabla A tiene una dependencia de la tabla B, la tabla b con el siguiente comando ejecutado desde el terminal de mysql.

- ***load data infile '/home/ambite/Downloads/good' into table tabla\_B fields terminated by ';' lines terminated by '\n' (bcadenaunica,bcadenaindex,cadenaSinindex);***

Mi archivo de carga se llamaba **good**.

<sup>2</sup> <https://dev.mysql.com/doc/refman/5.7/en/load-data.html>

La ruta de carga era desde local **/home/ambite/Downloads**.

En la siguiente foto vemos que se creó sin problemas, las 100 rows para esta tabla.

```
mysql> load data local infile '/home/ambite/Downloads/good' into table tabla_B fields terminated by ';'
lines terminated by '\n'(bCadenaunica,bCadenaindex,BcadenaSinindex);
Query OK, 100 rows affected (0.07 sec)
Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
```

Después fui a por la tabla A, ya que ahora sí que podría meter las claves foráneas haciendo referencia a las de la tabla B anteriormente creadas .

- ***load data infile '/home/ambite/Downloads/datosA' into table tabla\_A fields terminated by ';' lines terminated by '\n' (cadenaúnica,cadena index,acadenaSinindex,aidclavajena);***

```
mysql> load data local infile '/home/ambite/Downloads/datosA' into table tabla_A fields terminated by ';' lines terminated by '\n'(aCadenaunica,aCadenaindex,acadenaSinindex,aidclavajena);
Query OK, 1000000 rows affected, 65535 warnings (1 min 29.59 sec)
Records: 1000000 Deleted: 0 Skipped: 0 Warnings: 1000000
```

En este caso tardo bastante mas como se puede ver casi un minuto y medio ya que la colección de datos era muy grande.

Realmente lo más difícil ha sido como dejar los datos para poder usar este método de carga, en mi caso he tenido que crear dos scripts de bash para modificar ficheros.

Uno de los scripts que encontré fue este aunque he conseguido alguno más rápido usando el reloj de la cpu, sin embargo este fue el que use inicialmente para generar las diferentes colecciones de datos .

Use la modificación de pasarle cuantas líneas quería genar para poder usarlo en más casuísticas.

Con este script tarde 40 minutos en crear la colección de datos, usando el reloj y lanzando 10 veces el script a la vez lo conseguí en 20 min.

Está claro que este método se puede mejorar mucho pero bueno lo di por bueno ya que la aleatoriedad de los datos está totalmente asegurada que era lo que más me preocupaba.

```
#!/bin/bash
```

```
read -p "dime cuantas lineas te genero" límite
```

```
contador=0;
```

```
while [ $límite -gt $contador ]
```

```
do
```

```
    linea=$(cat /dev/urandom | tr -dc '[:alpha:]' | head -c 22)
```

```
    echo $linea >> archivo aleatorio.txt
```

```
    contador=$((contador+1))
```

```
done
```

## SCRIPT GENERACIÓN DE FICHERO PARA EL LOAD DATA

```
#!/bin/bash
read -p "dime el archivo que queremos modificar" archivo
read -p "dime el nombre del archivo que queremos nuevo" resultado
contador=1
while read line
do
if [ 100 -ge $contador ]
then
b=";"
a="cd%"
c=$a$line$b$line$b$line$b$contador
contador=$((contador+1))
echo "$c" >> $resultado
else [ $contador -eq 101 ]
b=";"
a="cd%"
contador=1
c=$a$line$b$line$b$line$b$contador
echo "$c" >> $resultado
fi
done < $archivo
```

Con estos dos scripts cree los datos y les di formato para poder ejecutar el load data en las tablas.

10. Indique qué significa cada una de las opciones de la tabla 8.1 de <https://dev.mysql.com/doc/refman/5.7/en/explain-output.html>. Teniendo en cuenta que algunas de ellas tienen, a su vez, diferentes valores, como por ejemplo “select\_type”.



La tabla es la siguiente vamos a ir explicando que significa cada columna, una por una .

Columna	Nombre JSON	Sentido
<u>id</u>	select_id	El SELECT identificador
<u>select_type</u>	Ninguna	El SELECT tipo
<u>table</u>	table_name	La tabla para la fila de salida
<u>partitions</u>	partitions	Las particiones coincidentes
<u>type</u>	access_type	El tipo de unión
<u>possible_keys</u>	possible_keys	Los posibles índices a elegir
<u>key</u>	key	El índice realmente elegido
<u>key_len</u>	key_length	La longitud de la clave elegida
<u>ref</u>	ref	Las columnas comparadas con el índice.
<u>rows</u>	rows	Estimación de filas a examinar
<u>filtered</u>	filtered	Porcentaje de filas filtradas por condición de tabla
<u>Extra</u>	Ninguna	Información Adicional

### id(json:select\_id)

El identificador de Select es un numero secuencial de la select dentro de la consulta.

Su valor puede ser nulo si la fila es una unión de otras filas , en este caso la columna table muestra un valor para indicar a qué fila se refiere la unión de filas con los valores

<unión, N>id M.

### select\_type (Nombre JSON: ninguno)

El tipo de SELECT, que puede ser cualquiera de los que se muestran en la siguiente tabla.

Un formato JSON EXPLAIN expone el SELECT tipo como una propiedad de a query block, a menos que sea SIMPLE o PRIMARY.

Tipos de select:

- **simple**:sin usar uniones ni subconsultas.
- **primary**: la consulta ams externa
- **unión**: segunda o sucesivas selects .
- **dependent unión** : segunda o posterior select como la anterior , pero dependiente de una consulta externa.
- **unión result**: el resultado de la unión
- **subquery** :primera select de la subconsulta

- **dependent subquery** :igual que la anterior , pero además depende de una consulta externa.
- **derived** :tabla derivada de la consulta.
- **materialized**: una subconsulta realizada
- **uncachable subquery**: una consulta que no puede guardarse en caché y se tiene que volver a hacer para cada consulta externa.
- **uncachable unión**: la segunda o posterior select de una unión que pertenece a una subconsulta y no puede almacenarse en caché.

### **table(Nombre JSON: table\_name)**

El nombre de la tabla a la que se refiere la fila de salida. Este también puede ser uno de los siguientes valores:

<unión M,N>: La fila se refiere a la unión de las filas con id valores de M y N.

<derived N>: La fila se refiere al resultado de la tabla derivada para la fila con un id valor de N. Una tabla derivada puede resultar, por ejemplo, de una subconsulta en la FROM cláusula.

<subquery EN >: La fila se refiere al resultado de una subconsulta materializada para la fila con un id valor de N.

### **partitions(Nombre JSON: partitions)**

Las particiones a partir de las cuales la consulta haría coincidir los registros. El valor es NULL para tablas no particionadas.

### **type(Nombre JSON: access\_type)**

El tipo de unión. Para obtener descripciones de los diferentes tipos, consulte EXPLAIN.

### **possible\_keys(Nombre JSON: possible\_keys)**

Esta columna indica los índices de los cuales MySQL puede elegir encontrar las filas en esta tabla. Tenga en cuenta que esta columna es totalmente independiente del orden de las tablas que se muestran en la salida de EXPLAIN .Eso significa que algunas de las claves possible\_keys podrían no ser utilizables en la práctica con el orden de la tabla generada. Si esta columna es NULL(o no está definida en la salida con formato JSON), no hay índices relevantes. En este caso, puede mejorar el rendimiento de su consulta examinando la cláusula WHERE para verificar si se refiere a alguna columna o columnas que serían adecuadas para la indexación. Si es así, cree un índice apropiado y verifique la consulta EXPLAIN nuevamente.

### **key(Nombre JSON: key)**

La columna key indica el índice que MySQL ha decidido usar para la consulta.

Es posible que key nombre un índice que no esté presente en el possible\_keys valor. Esto puede suceder si ninguno de los possible\_keys índices es adecuado para buscar filas, pero todas las columnas seleccionadas por la consulta son columnas de algún otro índice. Es decir, el índice con nombre cubre las columnas seleccionadas, por lo que, aunque no se utiliza para determinar qué filas recuperar, una exploración de índice es más eficiente que una exploración de filas de datos.

Para InnoDB, un índice secundario podría cubrir las columnas seleccionadas incluso si la consulta también selecciona la clave primaria porque InnoDB almacena el valor de la clave primaria con cada índice secundario. Si key es así NULL, MySQL no encontró ningún índice para usar para ejecutar la consulta de manera más eficiente.

Para forzar a MySQL para uso o ignorar un índice que aparece en la possible\_keys columna, el uso FORCE INDEX, USE INDEX o IGNORE INDEX en su consulta.

### **key\_len(Nombre JSON: key length)**

Esta columna indica la longitud de la clave que MySQL ha usado. El valor de la key\_len permite determinar las partes de una clave que tiene más de una, usa MySQL. Si la key columna dice NULL, la KEY\_LEN columna también dice NULL.

Debido al formato de almacenamiento de la clave, la longitud de la clave es mayor para una columna que puede ser NULL que para una columna que sea not null.

### **ref(Nombre JSON: ref)**

La ref columna muestra qué columnas o constantes se comparan con el índice nombrado en la key columna para seleccionar filas de la tabla.

### **rows(Nombre JSON: rows)**

Esta columna indica el número de filas que MySQL cree que debe examinar para ejecutar la consulta.

Para las tablas con motor InnoDB este número es una estimación y puede no ser siempre exacto.

### **filtered(Nombre JSON: filtered)**

Esta columna indica un porcentaje estimado de filas de la tabla que serán filtradas por la condición de la tabla. El valor máximo es 100, lo que significa que no se produjo ningún filtrado de filas. Los valores que disminuyen de 100 indican cantidades crecientes de filtrado. rows muestra el número estimado de filas examinadas y rows × filtered muestra el número de filas que se unirán con la siguiente tabla. Por ejemplo, si rows es 1000 y filtered es 50.00 (50%), el número de filas a unir con la siguiente tabla es  $1000 \times 50\% = 500$ .

**Extra (Nombre JSON: ninguno)**

Esta columna contiene información adicional sobre cómo MySQL resuelve la consulta. Para obtener descripciones de los diferentes valores, consulte EXPLAIN .

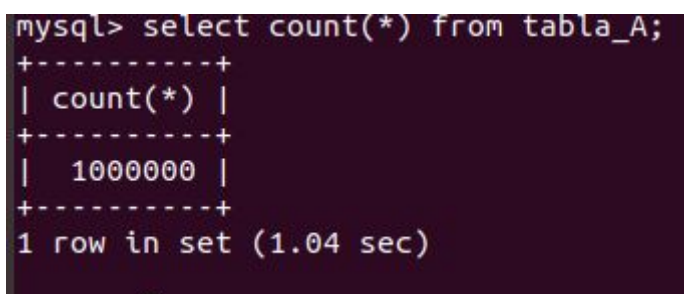
No hay una sola propiedad JSON correspondiente a la Extra columna; sin embargo, los valores que pueden aparecer en esta columna se exponen como propiedades JSON o como el texto de la message propiedad.

**11. Indique el tiempo de ejecución de las siguientes consultas (deberá adjuntar capturas de pantalla):**

Lo primero es explicar que he tenido muchos problemas para poder sacar los tiempos de consulta sin cache, así que tuve que **resetear el servicio y borrar** unos archivos de sistema donde se guardaban al parecer datos de `/proc/sys/vm/drop_caches`.

**select count(\*) from tabla\_A**

Ha tardado 1.04 segundos.



```
mysql> select count(*) from tabla_A;
+-----+
| count(*) |
+-----+
| 1000000 |
+-----+
1 row in set (1.04 sec)
```

**select aCadenaUnica from tabla\_A**

Para esta son 8.85 segundos .

```
mysql> select aCadenaunica from tabla_A;
```

```
| Cd%ZZZXTMtlacAgqtJteL |
| cd%zzZYdQZgXkOGbkdcAqu |
| cd%zzzyTvLCxBIYvDePiij |
| cd%ZzzZNTXsPIysKTVczTk |
+-----+
1000000 rows in set (8.85 sec)
```

**select aCadenaSinIndex from tabla\_A**

Esta son 1.11 segundos.

```
mysql> select aCadenaSinindex from tabla_A;
```

```
| 0jrlndgcUvSZuVKnppqonTS |
| SWnTIuNiWJEdVcHsHRaStp |
| vrUBWEK0sjbWuvXLDzUXRL |
| dWFLppHuWAIEHhWbAavfzJ |
| AsdIrEuJUxPURduojwQmws |
+-----+
1000000 rows in set (1.11 sec)
```

**select A\_id from tabla\_A order by aCadenaUnica**

Esta son 8.68 sec.

```
mysql> select A_id from tabla_A order by aCadenaunica;
```

```
| 939566 |
| 618122 |
| 835232 |
| 431729 |
| 724360 |
+-----+
1000000 rows in set (8.68 sec)
```

**select \* from tabla\_A order by aCadenaUnica**



#24

Para esta consulta tardó 1.99segundos.

```
mysql> select * from tabla_A order by aCadenaunica;
```

```
| 431729 | cd%zzzyTvlCxBiYvDePiIj | zzzyTvlCxBiYvDePiIjKCJ | zzzyTvlCxBiYvDePiIjKCJ | 55  
| 724360 | cd%ZzzZNTXsPIysKTVczTk | ZzzZNTXsPIysKTVczTkCXD | ZzzZNTXsPIysKTVczTkCXD | 89  
+-----+-----+-----+-----+  
1000000 rows in set (1.99 sec)
```

**select aCadenaUnica from tabla\_A order by aCadenaUnica**

Para esta 8.88 segundos.

```
mysql> select aCadenaunica from tabla_A order by aCadenaunica;
```

```
| cd%zzZYdQZgXkOGbkdcAqu |  
| cd%zzzyTvlCxBiYvDePiIj |  
| cd%ZzzZNTXsPIysKTVczTk |  
+-----+  
1000000 rows in set (8.88 sec)
```

**select aCadenaSinIndex from tabla\_A order by aCadenaSinIndex.**

Tiempo de consulta 1.52 segundos.

```
mysql> select aCadenaSinindex from tabla_A order by aCadenaSinindex;
```

```
| ZzzXaEzvvQaENqihJvcIPq |  
| ZzZXIPgcpwFoGweRcklDwA |  
| ZZZxtMiTaCaGquIJleLuJG |  
| zzZYdQZgXkOGbkdcAquKqt |  
| zzzyTvlCxBiYvDePiIjKCJ |  
| ZzzZNTXsPIysKTVczTkCXD |  
+-----+  
1000000 rows in set (1.52 sec)
```

**select count(\*) from tabla\_B**

Ha tardado 0.05 segundos.

```
mysql> select count(*) from tabla_B;
+-----+
| count(*) |
+-----+
|      100 |
+-----+
1 row in set (0.05 sec)
```

**select aCadenaUnica from tabla\_A**

Ha tardado 0.26 segundos.

```
mysql> select acadenaunica from tabla_A;
| cd%ZZZXIPgcPwF0GwERCKL |
| cd%ZZZxtMiTaCaGqulJleL |
| cd%zzZYdQZgXkOGbkdcAqu |
| cd%zzzyTvLCxBIYvDePij |
| cd%ZzzZNTXsPIysKTVczTk |
+-----+
1000000 rows in set (0.26 sec)
```

**select aCadenaSinIndex from tabla\_A**

Ha tardado 0.3 segundos.

```
mysql> select acadenaSinindex from tabla_A;
| SWnTluNiWJEEdVcHsHRaStp |
| vrUBWEKosjbWuvXLDzUXRL |
| dWFLppHuwAIEHhWbAavfzJ |
| AsdIrEuJUxPUrduojwQmws |
+-----+
1000000 rows in set (0.30 sec)
```

**12. Ejecute las anteriores consultas con explain. Indique los resultados y explique por qué se obtienen dichos valores.**

```
mysql> explain select count(*) from tabla_A\G;
```

Esta select tardó en ejecutarse 1.04 segundos.

```
mysql> explain select count(*) from tabla_A\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: index
possible_keys: NULL
         key: fk_tabla_A_1_idx
        key_len: 4
         ref: NULL
        rows: 939729
   filtered: 100.00
      Extra: Using index
1 row in set, 1 warning (0.00 sec)

ERROR:
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=index**, Esto significa que analiza el árbol del índice y pueden pasar dos cosas.

Primera que el índice cubra todo lo que la consulta necesita en ese caso, se escanea el árbol del índice solo y en la **columna extra** saldrá la opción **using index**.

Segunda se realiza un escaneo completo de la tabla utilizando lecturas del índice para buscar las filas de datos en orden del índice, en este caso en la **columna extra no aparecerá using index**.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=Fk\_tabla\_A\_1\_idx**, indica el índice que mysql ha decidido usar en este caso es la clave ajena que apunta a la tabla\_B.

**key\_len=4**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=NULL**, muestra qué columnas o constantes son comparadas con el índices especificado en la columna key, en este caso no se usa ninguno.

**Rows=939729**, es el número de filas aproximado que examinara Mysql, para la consulta..

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using index**, esto quiere decir que la información de las columnas se recupera usando sólo información en el árbol del índice, sin tener que hacer ninguna búsqueda adicional para leer la fila real.

**mysql> explain select aCadenaUnica from tabla\_A\G;**

**Esta select tardó en ejecutarse 8.85 segundos.**

```
mysql> explain select aCadenaUnica from tabla_A\G;
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: tabla_A
   partitions: NULL
         type: index
possible_keys: NULL
          key: aCadenaunica_UNIQUE
       key_len: 25
         ref: NULL
        rows: 939729
   filtered: 100.00
      Extra: Using index
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=index**, Esto significa que analiza el árbol del índice y pueden pasar dos cosas.

Primera que el índice cubra todo lo que la consulta necesita en ese caso, se escanea el árbol del índice solo y en la **columna extra** saldrá la opción **using index**.

Segunda se realiza un escaneo completo de la tabla utilizando lecturas del índice para buscar las filas de datos en orden del índice, en este caso en la **columna extra** no aparecerá **using index**.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=aCadena\_UNIQUE**, el índice que ha decidido usar en este caso es la clave única de la columna aCadenaúnica, para realizar la consulta, cosa que tiene sentido ya que la consulta es solo sobre esa columna.

**key\_len=25**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=NULL**, muestra qué columnas o constantes son comparadas con el índice especificado en la columna key, en este caso no se compara ninguna columna con el índice.

**Rows=939729**, es el número de filas aproximado que examinará MySQL para la consulta..



**Filtered= 100.00**,muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using index**, esto quiere decir que la información de las columnas se recupera usando sólo información en el árbol del índice, sin tener que hacer ninguna búsqueda adicional para leer la fila real.

**mysql>select aCadenaSinIndex from tabla\_A\G;**

Esta select tardó en ejecutarse 1.11 segundos.

```
mysql> explain select aCadenaSinIndex from tabla_A\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: ALL
possible_keys: NULL
         key: NULL
        key_len: NULL
         ref: NULL
         rows: 939729
    filtered: 100.00
       Extra: NULL
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary,unión,derived,etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila.Podría tener alguno de los siguientes valores.Tabla resultante de la unión de las tabas cuyos campos id son M y N,tarla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=Null**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ALL**, realiza un escaneo completo de la tabla para cada combinación de filas en las tablas anteriores, esto es malo si la tabla primera no está marcada con const y muy malo en todos los demás casos. Esto se puede evitar agregando índices a esta columna algo que deberíamos tener en cuenta si va a ser una columna muy consultada de nuestra base de datos.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=Null**, no hay ningún índice para la consulta, como es lógico ya que hemos consultado información sobre una única columna sin índice.

**key\_len=Null**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=Null**, evidentemente es nulo ya que esto te indica columnas o constantes que se comparan con el índice de la columna Key.

**Rows=939729**, es el número de filas aproximado que examinará Mysql, para la consulta..

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Null**, no tiene información adicional sobre cómo realizar la consulta.

```
mysql>explain select A_ Id from tabla_A order by aCadenaunica \G;
```

Esta select tardó en ejecutarse 8.68 segundos.

```
mysql> explain select A_id from tabla_A order by aCadenaunica \G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: index
possible_keys: NULL
         key: aCadenaunica_UNIQUE
        key_len: 25
         ref: NULL
        rows: 993673
    filtered: 100.00
      Extra: Using index
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=Simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=index**, Esto significa que analiza el árbol del índice y pueden pasar dos cosas.

Primera que el índice cubra todo lo que la consulta necesita en ese caso, se escanea el árbol del índice solo y en la **columna extra** saldrá la opción **using index**.

Segunda se realiza un escaneo completo de la tabla utilizando lecturas del índice para buscar las filas de datos en orden del índice, en este caso en la **columna extra** no aparecerá **using index**.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=aCadena\_UNIQUE**, el índice que ha decidido usar en este caso es la clave única de la columna aCadenaUnica, para realizar la consulta, cosa que tiene sentido ya que la consulta es solo sobre esa columna.

**key\_len=25**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=NULL**, evidentemente es nulo ya que esto te indica columnas o constantes que se comparan con el índice de la columna Key.

**Rows=993673**, es el número de filas aproximado que examinará MySQL, para la consulta.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using index**, esto quiere decir que la información de las columnas se recupera usando sólo información en el árbol del índice, sin tener que hacer ninguna búsqueda adicional para leer la fila real.

**mysql> explain select \* from tabla\_A order by aCadenaUnica\G;**

Esta select tardó en ejecutarse 1.99 segundos.

```
mysql> explain select * from tabla_A order by aCadenaUnica\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
          table: tabla_A
    partitions: NULL
           type: ALL
possible_keys: NULL
           key: NULL
        key_len: NULL
           ref: NULL
          rows: 939729
    filtered: 100.00
      Extra: Using filesort
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ALL**, realiza un escaneo completo de la tabla para cada combinación de filas en las tablas anteriores, esto es malo si la tabla primera no está marcada con const y muy malo en todos los demás casos. Esto se puede evitar agregando índices a esta columna algo que deberíamos tener en cuenta si va a ser una columna muy consultada de nuestra base de datos.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=NULL**, no va utilizar ningún índice para la consulta.

**key\_len=NULL**, no tiene índice, así que esto es nulo.

**Ref=NULL**, muestra qué columnas o constantes son comparadas con el índice especificado en la columna key, en este caso es nulo claramente ya que no tenemos índice con el que comparar.

**Rows=939729**, es el número de filas aproximado que examinara Mysql, para la consulta.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using filesort**, esta información nos dice que mysql no tiene índices disponibles para hacer la consulta del order by, lo cual hará que la consulta sea mala en términos de rendimiento y si va a ser una consulta que necesitemos realizar con asiduidad en nuestra base de datos habrá que crear un índice para que mejore el rendimiento, es muy típico que en estas consultas sobre más de una columna ordenadas por una de ellas se apliquen los índices combinados de columnas.

**mysql> explain select aCadenaUnica from tabla\_A order by aCadenaUnica\G;**

Esta select tardó en ejecutarse 8.88 segundos.

```
mysql> explain select aCadenaUnica from tabla_A order by aCadenaUnica\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: index
possible_keys: NULL
         key: aCadenaUnica_UNIQUE
        key_len: 25
         ref: NULL
        rows: 939729
    filtered: 100.00
      Extra: Using index
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=Simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=index**, Esto significa que analiza el árbol del índice y pueden pasar dos cosas.

Primera que el índice cubra todo lo que la consulta necesita en ese caso, se escanea el árbol del índice solo y en la **columna extra** saldrá la opción **using index**.

Segunda se realiza un escaneo completo de la tabla utilizando lecturas del índice para buscar las filas de datos en orden del índice, en este caso en la **columna extra** no aparecerá **using index**.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=aCadena\_UNIQUE**, el índice que ha decidido usar en este caso es la clave única de la columna aCadenaUnica, para realizar la consulta, cosa que tiene sentido ya que la consulta es solo sobre esa columna.

**key\_len=25**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=NULL**, evidentemente es nulo ya que esto te indica columnas o constantes que se comparan con el índice de la columna Key.

**Rows=939729**, es el número de filas aproximado que examinará MySQL, para la consulta.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**mysql> explain select aCadenaSinIndex from tabla\_A order by aCadenaSinIndex\G;**

Esta select tardó en ejecutarse 1.52 segundos.

```
mysql> explain select aCadenaSinIndex from tabla_A order by aCadenaSinIndex\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
        key_len: NULL
         ref: NULL
        rows: 939729
    filtered: 100.00
      Extra: Using filesort
1 row in set, 1 warning (0.00 sec)
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.



**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ALL**, realiza un escaneo completo de la tabla para cada combinación de filas en las tablas anteriores, esto es malo si la tabla primera no está marcada con const y muy malo en todos los demás casos. Esto se puede evitar agregando índices a esta columna algo que deberíamos tener en cuenta si va a ser una columna muy consultada de nuestra base de datos.

**Possible\_keys=NULL**, esto indica que no se han encontrado índices relevantes para esta consulta.

**key=NULL**, no va a utilizar ningún índice para la consulta.

**key\_len=NULL**, no tiene índice, así que esto es nulo.

**Ref=NULL**, muestra qué columnas o constantes son comparadas con el índice especificado en la columna key, en este caso es nulo claramente ya que no tenemos índice con el que comparar.

**Rows=939729**, es el número de filas aproximado que examinará MySQL para la consulta.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using filesort**, esta opción es mala en términos de rendimiento ya que es lenta, y se da siempre que hacemos un order by de columnas que no tienen ningún índice disponible, por ello si la consulta va a ser típica en nuestra base de datos será conveniente crear un índice para esta o estas columnas, se suelen aplicar índices combinados.

**13. Cree 2 consultas, usando EXPLAIN, donde refleje la problemática de usar el operador LIKE con %). Deberá usar “tabla\_A” sobre aCadenaUnica, indicando el significado de cada columna de la salida de explain. ¿Por qué existe esa problemática?**

El problema de este tipo de consultas por where es, cuando filtramos con %(carácter comodín) y nos saltamos el inicio de la cadena, es que no vamos a poder usar, en caso de que tuviese el índice de esa columna, ya que los índices se crean y ordenan con el comienzo de los datos de la columna, es decir que si yo tuviese un valor por ejemplo en la **columna** del índice que fuese **nombre**, por ejemplo Antonio puedo hacer la consulta con el filtro **where nombre like 'Anto%'** en este caso veremos que si puede usar el índice para la consulta ya que estamos usando el comienzo del nombre y se podrá leer para buscar en el índice.

Sin embargo con la siguiente consulta

**select \* from personas where nombre like '%ntonio';**

No podría usar el índice para resolver la consulta ya que no sabe cual es la letra inicial, porque con el filtro así lo hemos decidido.

***En definitiva un índice se va a poder usar para realizar comparaciones Like siempre y cuando el argumento que le pasemos al like sea una cadena constante y no comience por un carácter comodín.***

Aquí tenemos la consulta que si usa el índice como vemos en su explain.

- **mysql> explain select aCadenaunica from tabla\_A where aCadenaunica like 'cd%upM%';**

```
mysql> explain select aCadenaunica from tabla_A where aCadenaunica like 'cd%upM%';
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tabla_A | NULL | range | aCadenaunica_UNIQUE | aCadenaunica_UNIQUE | 25 | NULL | 469864 | 100.00 | Using where; Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

Voy a poner las fotos en dos, para facilitar su lectura.

```
mysql> explain select aCadenaunica from tabla_A where aCadenaunica like '
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tabla_A | NULL | range | aCadenaunica_UNIQUE |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

Vemos que puede usar el índice aCadena única UNIQUE, como índice posible.

Lo importante está en la siguiente foto vemos como key, que como hemos visto en el punto anterior es la responsable de indicar el índice que se usa en la consulta.

En nuestro caso usa el índice creado para la columna aCadémica que se llama Cadenaunica\_Unique.

```
cd%upM%';
```

key	key_len	ref	rows	filtered	Extra
aCadenaunica_UNIQUE	25	NULL	469864	100.00	Using where; Using index

La consulta siguiente veremos que no va a utilizar el índice por lo que hemos comentado anteriormente y usar el valor comodín al principio del filtro.

- ***mysql> explain select aCadenaunica from tabla\_A where aCadenaunica like '%cd%upM%';***

```
mysql> mysql> explain select aCadenaunica from tabla_A where aCadenaunica like '%cd%upM%';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	tabla_A	NULL	index	NULL	aCadenaunica_UNIQUE	25	NULL	939729	11.11	Using where; Using index

1 row in set, 1 warning (0.00 sec)

```
mysql> mysql> explain select aCadenaunica from tabla_A where aCaden
```

id	select_type	table	partitions	type	possible_keys
1	SIMPLE	tabla_A	NULL	index	NULL

1 row in set, 1 warning (0.00 sec)

```
naunica like '%cd%upM%';
```

key	key_len	ref	rows	filtered	Extra
aCadenaunica_UNIQUE	25	NULL	939729	11.11	Using where; Using index

En este caso vemos como en el valor posible\_keys sale null, Si esta columna es NULL(o no está definida en la salida con formato JSON), no hay índices relevantes. En este caso, puede mejorar el rendimiento de su consulta examinando la cláusula WHERE para verificar si se refiere a alguna columna o columnas que serían adecuadas para la indexación que vemos que si ya que existe el index para esa tabla pero no podemos usarlo si empezamos por valor aleatorio.

14. Realice EXPLAIN de tres consultas (sobre aCadenaUnica, aCadenaConIndex y aCadenaSinIndex) utilizando filtros (uso de WHERE) de la columna seleccionada en la select, junto con el operador = (los valores filtrados deberán existir en la tabla). Indique los tiempos de ejecución así como el significado de cada columna de la salida de explain. ¿Qué conclusiones observa?

Vamos primero a recordar que hace exactamente la sentencia explain, cuando la ejecutamos sobre una select.

Explain nos permite obtener información sobre el plan de ejecución de la consulta realizada sobre nuestra base de datos, para de esta forma poder optimizar la ejecución de dichas consultas.

La sentencia nos devuelve una tabla con una serie de filas, que tienen información sobre cada una de las tablas que vamos a emplear en la consulta sobre la que hemos ejecutado el explain. Explain se puede usar sobre select, delete, insert, replace y update.

Primera consulta sobre la tabla acadenaunica.

- **mysql> explain select aCadenaunica from tabla\_A where aCadenaunica='cd%KOEXWjsxeSCiGTfuUHi';**

```
mysql> explain select aCadenaunica from tabla_A where aCadenaunica='cd%KOEXWjsxeSCiGTfuUHi';
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tabla_A | NULL | const | aCadenaunica_UNIQUE | aCadenaunica_UNIQUE | 25 | const | 1 | 100.00 | Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.02 sec)
```

Las imagenes las voy a poner en dos cachos para que se puedan leer bien.

```
mysql> explain select aCadenaunica from tabla_A where aCadenaunica='cd%KO
+----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tabla_A | NULL | const | aCadenaunica_UNIQUE |
+----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.02 sec)
```

```
EOEXWjsxeSCiGTfuUHi';
+-----+-----+-----+-----+-----+-----+
| key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+
| aCadenaunica_UNIQUE | 25 | const | 1 | 100.00 | Using index |
+-----+-----+-----+-----+-----+-----+
```

Vamos a explicar columna por columna.

En este caso vemos como **id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=const**, muestra el tipo de unión usada en la consulta.

**Possible\_keys=aCadenaunica\_UNIQUE**, muestra qué índices puede escoger MySQL para buscar las filas de la tabla. Si esta columna es NULL, significa que no hay índices que puedan ser usados para mejorar la consulta.

**key=aCadenaunica\_UNIQUE**, muestra el índice que MySQL ha decidido usar para ejecutar la consulta. Es posible que el nombre del índice no esté presente en la lista de possible\_keys.

**key\_len=25**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=const**, muestra qué columnas o constantes son comparadas con el índices especificado en la columna key. El valor const, es que se da cuando una consulta compara primary keys o unique index con valores constantes, en este caso tiene sentido ya que el índice que tiene creada esta columna es de tipo Unique.

**Rows=1**, muestra el número de filas que MySQL cree que deben ser examinadas para ejecutar la consulta. Este número es un número aproximado.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using index**, nos indica que ha utilizado el índice para sacar la información de la columna que le hemos solicitado..



Siguiente consulta sobre acadenaindex.

- **explain select acadenaindex from tabla\_A where aCadenaindex='jLvDXVQcCphiDuxHiMhCvq';**

```
mysql> explain select acadenaindex from tabla_A where aCadenaindex='jLvDXVQcCphiDuxHiMhCvq';
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tabla_A | NULL | ref | index4 | index4 | 24 | const | 1 | 100.00 | Using index |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.02 sec)
```

```
+----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key |
+----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tabla_A | NULL | ref | index4 | index4 |
+----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.02 sec)
```

```
x='jLvDXVQcCphiDuxHiMhCvq';
+----+-----+-----+-----+-----+-----+-----+
| key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+
| index4 | 24 | const | 1 | 100.00 | Using index |
+----+-----+-----+-----+-----+-----+-----+
```

En este caso vemos como **Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ref**, este tipo se usa si la combinación de las filas de las tablas, usan un prefijo más a la izquierda de la clave o si la clave no es un índice de tipo primary key o unique, es decir si la combinación no puede seleccionar una sola fila en función de la clave). En nuestro caso está claro que está usando el index4, que no es ni tipo primary key ni unique.

**Possible\_keys=index4**, muestra qué índices puede escoger MySQL para buscar las filas de la tabla. Si esta columna es NULL, significa que no hay índices que puedan ser usados para mejorar la consulta.

**key=Index 4**, muestra el índice que MySQL ha decidido usar para ejecutar la consulta. Es posible que el nombre del índice no esté presente en la lista de possible\_keys.

**key\_len=24**, muestra el tamaño del índice que MySQL ha decidido usar para ejecutar la consulta.

**Ref=const**, muestra qué columnas o constantes son comparadas con el índices especificado en la columna key.

**Rows=1**, muestra el número de filas que MySQL cree que deben ser examinadas para ejecutar la consulta. Este número es un número aproximado.

**Filtered= 100.00**,muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using index**, indica que la consulta se ha realizado utilizando el índice, para sacar la información de las columnas..

Siguiente consulta de este tipo sobre acadenaSinindex.

Voy a hacer la consulta con el mismo filtro ya que son iguales ambas columnas solo que una no tiene índice y la otra si.

- ```
- mysql> explain select aCadenaSinindex from tabla_A where  
aCadenaSinindex='jLvDXVQcCphiDuxHiMhCvg';
```

```
mysql> explain select aCadenaSinIndex from tabla_A where aCadenaSinIndex='jLVdXVQCcPpHiDuxHiMhCvq';
```

| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra       |
|----|-------------|---------|------------|------|---------------|------|---------|------|--------|----------|-------------|
| 1  | SIMPLE      | tabla_A | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 993673 | 10.00    | Using where |

```
1 row in set, 1 warning (0.00 sec)
```

```
mysql> explain select aCadenaSinindex from tabla_A where aCadenaSinindex=
+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table  | partitions | type | possible_keys | key |
+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE      | tabla_A | NULL        | ALL  | NULL          | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```



```

nindex='jLvDXVQcCphiDuxHiMhCvq';
+-----+-----+-----+-----+-----+
key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+
NULL | NULL | NULL | 993673 | 10.00 | Using where |
+-----+-----+-----+-----+-----+

```

En este caso vemos como **Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ALL**, Este tipo de unión lo que hace es un escaneo completo de la tabla para cada combinación de filas de las tablas anteriores, esto es malo si la tabla primera no está marcada con const y muy malo en todos los demás casos. Esto se puede evitar agregando índices a esta columna algo que deberíamos tener en cuenta si va a ser una columna muy consultada de nuestra base de datos.

**Possible\_keys=NULL**, Si esta columna es NULL, significa que no hay índices que puedan ser usados para mejorar la consulta.

**key=NULL**, pues no tenemos índice para esta consulta.

**key\_len=NULL**, No tiene clave así que normal que sea nulo su tamaño.

**Ref=NULL**, evidentemente es nulo ya que esto te indica columnas o constantes que se comparan con el índice de la columna Key.

**Rows=993637**, vemos que nos devuelve un valor cercano a la totalidad de las filas que tiene la tabla, por lo que podemos suponer que la consulta va a ser bastante lenta.

**Filtered= 10.00**,muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using where**, nos indica que MYSQL HA UTILIZADO UNA CONDICIÓN DE TIPO WHERE PARA ENCONTRAR LA INFORMACIÓN DE LA CONSULTA QUE QUEREMOS .

## CONCLUSIONES

En cuanto a las conclusiones que saco del uso de explain en estas consultas es que, el uso de índices hace que la consulta sea mucha mas eficiente ,solo tenemos que fijarnos en los valores de rows, y filtered para ver que con los índices básicamente no va a tener que inspeccionar casi nada y con la columna que no tiene índices básicamente tiene que recorrer la totalidad de los datos para poder encontrar lo que buscamos.

Por tanto ya que el uso de índices,supone que haya un gasto mayor de espacio y recursos de nuestra base de datos,pero que luego hace que las consultas sean mucho más eficientes , debemos plantearnos su uso siempre y cuando en mi opinión la columna vaya a ser muy solicitada o poco solicitada.

Si la columna va ser de uso extensivo un índice es necesario, si no pues posiblemente no merezca la pena ponerlo.

en cuanto a los tiempos de ejecución de los explain de las diferentes consultas vemos que a más cosa tiene que comprobar más tanra , por ejemplo la columna sin indices sale tiempo 0.00 y las otras ronda los 0.02, 0.03 segundos, ya que tiene que hacer más comprobaciones

Sin embargo ahora voy a colgar los tiempos de la select sin explain , para ver cuanto mejora realmente la consulta, tener o no indices.

Vamos a verlo ejecutó exactamente la misma consulta y veremos el caso más importante para mi es el de columna con índice y columna sin índice ya que es exactamente la columna con los mismos valores y es la misma consulta así que podemos hacernos una idea de cuánto mejoró la consulta tener o no el índice.

```
mysql> select aCadenaunica from tabla_A where aCadenaunica='cd%KOEXWjsxeSCiGTfuUHi';
+-----+
| aCadenaunica |
+-----+
| cd%KOEXWjsxeSCiGTfuUHi |
+-----+
1 row in set (0.03 sec)

mysql> select acadenaindex from tabla_A where aCadenaindex='jLvDXVQcCphiDuxHiMhCvq';
+-----+
| acadenaindex |
+-----+
| jLvDXVQcCphiDuxHiMhCvq |
+-----+
1 row in set (0.04 sec)

mysql> select aCadenaSinindex from tabla_A where aCadenaSinindex='jLvDXVQcCphiDuxHiMhCvq';
+-----+
| aCadenaSinindex |
+-----+
| jLvDXVQcCphiDuxHiMhCvq |
+-----+
1 row in set (1.11 sec)
```

Perfecto y vemos como las consultas con índice apenas tardan nada, y como la misma consulta hecha sin índice tarda 1,11 segundos si queremos ver cuanto mejora el rendimiento la consulta solo tenemos que dividirlo y veremos cuantas veces mas rápido es uno que otro.

Si usamos la diferencia entre la búsqueda con índice y sin índice de lo mismo en tiempo sería  $1,11 \times 100 = 111$  centésimas de segundo

$0,04 \times 100 = 4$  centésimas de segundo.

Si dividimos  $111/4 = 27.75$  es decir que la consulta con índice es 27,75 veces más rápida que sin índice , por tanto la conclusión es clara.

Si lo necesitas usar mucho el índice es imprescindible

#### 15. Analice el resultado de las siguientes select con explain (indique, como en los casos anteriores, los resultados y su significado):

- `mysql> explain select aCadenaunica from tabla_A where aCadenaunica='cd%EZsxoPeHoDfGcFNggXG'`

```
mysql> explain select aCadenaunica from tabla_A where aCadenaunica='c
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: const
possible_keys: aCadenaunica_UNIQUE
         key: aCadenaunica_UNIQUE
        key_len: 25
         ref: const
         rows: 1
    filtered: 100.00
      Extra: Using index
1 row in set, 1 warning (0.00 sec)

ERROR:
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=Const**, muestra qué columnas o constantes son comparadas con el índice especificado en la columna key. El valor const, es que se da cuando una consulta compara primary keys o unique index con valores constantes, en este caso tiene sentido ya que el índice que tiene creada esta columna es de tipo Unique.

**Possible\_keys=aCadenaunica**, este es el índice que podría usar.

**key=aCadenaúnica**, el índice que usa para la consulta.

**key\_len=25**, el tamaño de la clave Cadena única.

**Ref=const**, evidentemente es nulo ya que esto te indica columnas op constantes que se comparan con el índice de la columna Key.

**Rows=1**, vemos que va a examinar una única fila para la consulta gracias al índice.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Using index**, nos muestra que la consulta se ha realizado mirando solo en el índice para obtener la información de la tabla.

- **mysql> explain select aCadenaunica,aCadenaSinindex from tabla\_A where aCadenaunica='cd%EZsxoPeHoDfGcFNggX**

```
mysql> explain select aCadenaunica,aCadenaSinindex from tabla_A
FNggX
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: const
possible_keys: aCadenaunica_UNIQUE
         key: aCadenaunica_UNIQUE
        key_len: 25
         ref: const
         rows: 1
    filtered: 100.00
       Extra: NULL
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta, en nuestro caso es una consulta sobre una sola tabla así que es normal que solo haya 1.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila. Podría tener alguno de los siguientes valores. Tabla resultante de la unión de las tablas cuyos campos id son M y N, tabla resultante de una tabla derivada cuyo id es N. Una tabla derivada puede ser, por ejemplo, una subconsulta en la cláusula FROM o tabla resultante de una subconsulta materializada cuyo id es N.

**Partitions=Null**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=Const**, muestra qué columnas o constantes son comparadas con el índice especificado en la columna key. El valor const, es que se da cuando una consulta compara primary keys o unique index con valores constantes, en este caso tiene sentido ya que el índice que tiene creada esta columna es de tipo Unique.

**Possible\_keys=aCadenaúnica**, este es el índice que podría usar.

**key=aCadenaúnica**, el índice que usa para la consulta.

**key\_len=25**, el tamaño de la clave Cadena única.

**Ref=const**, evidentemente es nulo ya que esto te indica columnas o constantes que se comparan con el índice de la columna Key.

**Rows=1**, vemos que va a examinar una única fila para la consulta gracias al índice.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=Null**, no tiene información adicional de cómo ejecutar la consulta.

**16. Realice, con joint, dos select similares donde en una se utilice dos índices y la otra uno o ninguno. Explique los resultados de explain. ¿Por qué uno de ellos utiliza dos índices y el otro no?**

- `mysql> explain select bCadenaSin index,aCadenaSinindex from tabla_A inner join tabla_B on tabla_A.aidclaveajena =tabla_B.idB;`

```
mysql> explain select bCadenaSinindex,aCadenaSinindex from tabla_
laveajena =tabla_B.idB\G ;
***** 1. row *****
      id: 1
    select_type: SIMPLE
      table: tabla_B
    partitions: NULL
       type: ALL
possible_keys: PRIMARY
         key: NULL
       key_len: NULL
         ref: NULL
        rows: 100
   filtered: 100.00
      Extra: NULL
***** 2. row *****
      id: 1
    select_type: SIMPLE
      table: tabla_A
    partitions: NULL
       type: ref
possible_keys: fk_tabla_A_1_idx
         key: fk_tabla_A_1_idx
       key_len: 4
         ref: gbd.tabla_B.idB
        rows: 9304
   filtered: 100.00
      Extra: NULL
2 rows in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

En estas consultas que hacemos sobre más de una tabla vemos que sale un explain por cada tabla que consultamos , que nos dará información de como Mysql hace la consulta en cada tabla.

En este caso primero sale la información de tabla\_B.

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta,.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary,unión,derived,etc en función de la complejidad de la consulta.

**Table=tabla\_B**, muestra la tabla a la que se refiere la información de la fila.



**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ALL**, Este tipo de unión lo que hace es un escaneo completo de la tabla para cada combinación de filas de las tablas anteriores, esto es malo si la tabla primera no está marcada con const y muy malo en todos los demás casos. Esto se puede evitar agregando índices a esta columna algo que deberíamos tener en cuenta si va a ser una columna muy consultada de nuestra base de datos.

**Possible\_keys=PRIMARY**, este es el índice que puede suar que además nos indica que es la clave primaria de la tabla

**key=NULL**, no usa ningún índice.

**key\_len=NULL**, como no usa ningún índice pues el tamaño de la clave es nulo.

**Ref=NULL**, ninguna columna es comparada con el índice.

**Rows=100**, vemos que va a examinar todas las filas de la tabla.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=NULL**, no tiene información adicional de cómo ejecutar la consulta.

### Ahora vamos a ver cómo hará la consulta de la tabla\_A.

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ref**, Todas las filas con valores de índice coincidentes se leen de esta tabla para cada combinación de filas de las tablas anteriores. Si la clave que se usa coincide solo con unas pocas filas, este es un buen tipo de combinación.

**Possible\_keys=fk\_tabla\_A\_1\_idx**, nos muestra como posible índice la clave ajena que apunta a la tabla\_B.

**key=fk\_tabla\_A\_1\_idx**, usa la clave de unión de las tablas como índice de la consulta.

**key\_len=4**, es el tamaño que tiene la clave **fk\_tabla\_A\_1\_idx**, .

**Ref=gbd.tabla\_B.idB**, vemos que para comparar utiliza la información de esta columna, algo que es normal ya que la unión de las tablas se ha hecho usando esa columna.

**Rows=9838**, las columnas aproximadas que va a examinar.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=NULL**, no tiene información adicional de cómo ejecutar la consulta.

- **mysql> explain select tabla\_B.bCadenaunica,tabla\_A.aCadenaindex from tabla\_A inner join tabla\_B on tabla\_A.aidclaveajena =tabla\_B.idB\G ;**

```
mysql> explain select tabla_B.bCadenaunica,tabla_A.aCadenaindex
      1. row
      id: 1
      select_type: SIMPLE
      table: tabla_B
      partitions: NULL
      type: index
      possible_keys: PRIMARY
      key: bCadenaunica_UNIQUE
      key_len: 48
      ref: NULL
      rows: 100
      filtered: 100.00
      Extra: Using index
      2. row
      id: 1
      select_type: SIMPLE
      table: tabla_A
      partitions: NULL
      type: ref
      possible_keys: fk_tabla_A_1_idx
      key: fk_tabla_A_1_idx
      key_len: 4
      ref: gbd.tabla_B.idB
      rows: 9304
      filtered: 100.00
      Extra: NULL
      2 rows in set, 1 warning (0.00 sec)
```

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_B**, muestra la tabla a la que se refiere la información de la fila.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=index**, Esto significa que analiza el árbol del índice y pueden pasar dos cosas.

Primera que el índice cubra todo lo que la consulta necesita en ese caso, se escanea el árbol del índice solo y en la **columna extra** saldrá la opción **using index**.

Segunda se realiza un escaneo completo de la tabla utilizando lecturas del índice para buscar las filas de datos en orden del índice, en este caso en la **columna extra no aparecerá using index**.

**Possible\_keys=PRIMARY**, este es el índice que puede usar que además nos indica que es la clave primaria de la tabla.

**key=bCadenaunica\_UNIQUE**, va a usar esta clave, de la columna bCadenaunica.

**key\_len=48**, como no usa ningún índice pues el tamaño de la clave es nulo.

**Ref=NULL**, ninguna columna es comparada con el índice.

**Rows=100**, vemos que va a examinar todas las filas de la tabla.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=using index**, nos dice que va a utilizar el index para realizar la consulta..

**Ahora vamos a ver cómo hará la consulta de la tabla\_A.**

**Id = 1**, esto muestra el número secuencial que identifica cada una de las tablas que vamos a usar en la consulta.

**Select type=simple**, pues eso quiere decir que el tipo de consulta de la select en esta ocasión es una consulta simple, pero podría ser primary, unión, derived, etc en función de la complejidad de la consulta.

**Table=tabla\_A**, muestra la tabla a la que se refiere la información de la fila.

**Partitions=NULL**, esta columna muestra las particiones cuyos registros coinciden con los de la consulta, sin embargo esta columna sólo se muestra si empleamos la sentencia EXPLAIN PARTITIONS.

**Type=ref**, Todas las filas con valores de índice coincidentes se leen de esta tabla para cada combinación de filas de las tablas anteriores. Si la clave que se usa coincide solo con unas pocas filas, este es un buen tipo de combinación.

**Possible\_keys=fk\_tabla\_A\_1\_idx**, nos muestra como posible índice la clave ajena que apunta a la tabla\_B.

**key=fk\_tabla\_A\_1\_idx**, usa la clave de unión de las tablas como índice de la consulta.

**key\_len=4**, es el tamaño que tiene la clave **fk\_tabla\_A\_1\_idx**, .

**Ref=gbd.tabla\_B.idB**, vemos que para comparar utiliza la información de esta columna, algo que es normal ya que la unión de las tablas se ha hecho usando esa columna.

**Rows=9838**, las columnas aproximadas que va a examinar.

**Filtered= 100.00**, muestra el porcentaje estimado de filas que serán filtradas por la condición de la consulta.

**Extra=NULL**, no tiene información adicional de cómo ejecutar la consu

**17. Realice y muestre los cambios necesarios para obtener los resultados de la página 18 del fichero mysql Explain explained.pdf que se refleja en Google Classroom.**

- **mysql> explain select \* from tabla\_A order by aCadenaSinIndex\G;**

```
mysql> explain select * from tabla_A order by aCadenaSinIndex\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
          table: tabla_A
    partitions: NULL
           type: ALL
possible_keys: NULL
           key: NULL
        key_len: NULL
           ref: NULL
          rows: 993673
    filtered: 100.00
      Extra: Using filesort
1 row in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

Aquí vemos que va utilizar filesort , que es una opción más lenta que un índice por ello vamos a intentar evitar esto añadiendo un índice a la columna aCadenaSinindex, para mejorar la velocidad de la consulta.

Para ello vamos a ejecutar las siguiente sentencia .

```
mysql> alter table tabla_A add index nuevo_sinindex (aCadenaSinindex);
```

```
mysql> alter table tabla_A add index nuevo_sinindex (aCadenaSinindex);
Query OK, 0 rows affected (12.71 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Con esto simplemente hemos añadido un índice a la columna que antes no tenía.

Si volvemos a ejecutar el explain anterior veremos como ahora cambia la opción de filesort por using index.

```
mysql> explain select aCadenaSinIndex from tabla_A order by aCadenaSinIndex\G;
***** 1. row *****
      id: 1
    select_type: SIMPLE
        table: tabla_A
    partitions: NULL
         type: index
possible_keys: NULL
          key: nuevo_sinindex
        key_len: 24
           ref: NULL
        rows: 993673
   filtered: 100.00
      Extra: Using index
1 row in set, 1 warning (0.00 sec)
```

Cosa lógica puesto que ahora sí que tiene un index que puede usar para mejorar la eficiencia de la consulta , podemos ver que si ahora miramos la descripción de la tabla:A, la columna tiene un índice nuevo.

```
mysql> desc tabla_A;
```

| Field           | Type        | Null | Key | Default | Extra          |
|-----------------|-------------|------|-----|---------|----------------|
| A_id            | int(11)     | NO   | PRI | NULL    | auto_increment |
| aCadenaunica    | varchar(22) | YES  | UNI | NULL    |                |
| aCadenaindex    | varchar(22) | NO   | MUL | NULL    |                |
| aCadenaSinindex | varchar(22) | NO   | MUL | NULL    |                |
| aidclaveajena   | int(11)     | NO   | MUL | NULL    |                |

```
5 rows in set (0.00 sec)
```

18. Indique 3 consultas, con join, de la base de datos de tienda Virtual, que crea que se realicen asiduamente. ¿Realizaría algún o algunos índices? ¿Sobre qué columnas? ¿Por qué? Refleje los cambios obtenidos antes y después obtenidos con explain.

- select nombre,idPedido from pedidos inner join clientes using (idCliente) where nombre like 'a%\G;

```
mysql> select nombre,idPedido from pedidos inner join clientes using (idCliente) where nombre like 'a%';
```

| nombre                             | idPedido |
|------------------------------------|----------|
| Ana Trujillo Emparedados y helados | 10308    |
| Antonio Moreno Taqueria            | 10365    |
| Around the Horn                    | 10355    |
| Around the Horn                    | 10383    |

- explain select nombre,idPedido from pedidos inner join clientes using (idCliente) where nombre like 'a%\G;

```
mysql> explain select nombre,idPedido from pedidos inner join clientes using
like 'a%\G;
***** 1. row *****
id: 1
select_type: SIMPLE
table: clientes
partitions: NULL
type: ALL
possible_keys: PRIMARY
key: NULL
key_len: NULL
ref: NULL
rows: 91
filtered: 11.11
Extra: Using where
***** 2. row *****
id: 1
select_type: SIMPLE
table: pedidos
partitions: NULL
type: ref
possible_keys: idCliente
key: idCliente
key_len: 4
ref: tiendaVirtual.clientes.idCliente
rows: 2
filtered: 100.00
Extra: Using index
2 rows in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

Ahora vamos a crear un nuevo índice que engloba a ambas columnas, como esta va a ser una consulta muy realizada en nuestra base de datos para saber que compras ha realizado cada cliente, nos parece una buena idea crear un índice para mejorar su eficiencia.

Lo primero que vamos a hacer es crear el nuevo índice con la siguiente sentencia.



- **mysql> alter table clientes add index nombre id (nombre,idCliente);**

```
mysql> alter table clientes add index nombr_id (nombre,idCliente);
Query OK, 0 rows affected (0.30 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Y ahora vamos a volver a ejecutar el explain, a ver como ha cambiado el resultado.

```
mysql> explain select nombre,idPedido from pedidos inner join clientes using (idCl
like 'a%'\G;
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: clientes
  partitions: NULL
         type: range
possible_keys: PRIMARY,nombr_id
          key: nombr_id
        key_len: 767
          ref: NULL
         rows: 4
    filtered: 100.00
      Extra: Using where; Using index
***** 2. row *****
      id: 1
  select_type: SIMPLE
        table: pedidos
  partitions: NULL
         type: ref
possible_keys: idCliente
          key: idCliente
        key_len: 4
          ref: tiendaVirtual.clientes.idCliente
         rows: 2
    filtered: 100.00
      Extra: Using index
2 rows in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

Podemos ver varias diferencias entre las más relevantes que el número de filas que tenemos que buscar baja drásticamente en este caso solo busca las filas que va a mostrarnos, ya que ahora puede usar el índice para las dos columnas no como anteriormente que la columna nombre mio tenía un índice sobre el que poder buscar.

Como podemos ver usa el nuevo índice , ya que puede elegir entre idCliente o , nombre id, y escoje nombre id como índice para la consulta.

Siguiente consulta podría ser por ejemplo qué empleados están en un departamento, y el código de ese departamento.

```
mysql> select empleados.nombre,departamentos.nombre,departamentos.Departamento from empleados inner join departamentos using(idDepartamento) order by idDepartamento;
```



```
mysql> select empleados.nombre, departamentos.nombre, departamentos.idDepartamento
r join departamentos using(idDepartamento) order by idDepartamento;
+-----+-----+-----+
| nombre | nombre | idDepartamento |
+-----+-----+-----+
Luisa	Informática	4
Margaret	Ventas	6
Robert	Ventas	6
Andrew	Ventas	6
Adam	Ventas	6
Steven	Ventas	6
Laura	Ventas	6
Janet	Ventas	6
Michael	Ventas	6
Nancy	Ventas	6
Anne	Ventas	6
+-----+-----+-----+
11 rows in set (0.00 sec)
```

Perfecto aquí podemos ver nuestros empleados, con su departamento y el id del mismo.

**explain select empleados.nombre, departamentos.nombre, departamentos.Departamento from empleados inner join departamentos using(idDepartamento) order by idDepartamento\G ;**

```
mysql> explain select empleados.nombre, departamentos.nombre, departamentos
dos inne
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: empleados
    partitions: NULL
         type: ALL
possible_keys: idDepartamento
          key: NULL
        key_len: NULL
         ref: NULL
         rows: 11
    filtered: 100.00
      Extra: Using temporary; Using filesort
***** 2. row *****
      id: 1
  select_type: SIMPLE
        table: departamentos
    partitions: NULL
         type: ALL
possible_keys: PRIMARY
          key: NULL
        key_len: NULL
         ref: NULL
         rows: 3
    filtered: 33.33
      Extra: Using where; Using join buffer (Block Nested Loop)
2 rows in set, 1 warning (0.00 sec)

ERROR:
No query specified
```

Vemos que en estas consultas no se está usando ningún índice, se crea con el using temporary una tabla temporal donde se almacenan los resultados de la consulta a la primera tabla para poder compararlos con los de la segunda y esto, junto con el filesort hacen que la consulta sea lenta.

Para evitar esto vamos a crear un índice combinado de ambas columnas de esta manera será más rápida la consulta

```
mysql> alter table empleados add index nombre_departamento(nombre,idDepartamento);
```

Ahora vamos a volver a ejecutar el explain

```
explain select empleados.nombre,departamentos.nombre,departamentos.Departamento from
empleados inner join departamentos using(idDepartamento) order by idDepartamento\G ;
```

```
mysql> explain select empleados.nombre,departamentos.nombre,departamentos.Departamento from
empleados inner join departamentos using(idDepartamento) order by idDepartamento\G ;
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: empleados
  partitions: NULL
         type: index
possible_keys: idDepartamento
          key: nombre_departamento
       key_len: 770
          ref: NULL
         rows: 11
    filtered: 100.00
      Extra: Using index; Using temporary; Using filesort
***** 2. row *****
      id: 1
  select_type: SIMPLE
        table: departamentos
  partitions: NULL
         type: ALL
possible_keys: PRIMARY
          key: NULL
       key_len: NULL
          ref: NULL
         rows: 3
    filtered: 33.33
      Extra: Using where; Using join buffer (Block Nested Loop)
2 rows in set, 1 warning (0.00 sec)
```

Como podemos ver ahora usa la clave que hemos creado para la consulta, nombre\_departamento, en extra nos parece que ahora si está usando index para la consulta.

No estoy seguro de que las consultas sean vayan a ser más rápidas ya que tambien veo aparecer using filesort en extra y eso es una opción lenta de mysql.

La siguiente consulta interesante que podríamos tener es por ejemplo que proveedor tienes para cada producto de la tienda, ordenados por la id de producto.

**mysql> select productos.nombre,proveedores.nombre from products inner join proveedores using (idProveedor) order by idProducto ;**

```
mysql> select productos.nombre,proveedores.nombre from products inner join proveedores using (idProveedor) order by idProducto ;
```

| nombre                          | nombre                             |
|---------------------------------|------------------------------------|
| Chais                           | Exotic Liquid                      |
| Chang                           | Exotic Liquid                      |
| Aniseed Syrup                   | Exotic Liquid                      |
| Chef Anton's Cajun Seasoning    | New Orleans Cajun Delights         |
| Chef Anton's Gumbo Mix          | New Orleans Cajun Delights         |
| Grandma's Boysenberry Spread    | Grandma Kelly's Homestead          |
| Uncle Bob's Organic Dried Pears | Grandma Kelly's Homestead          |
| Northwoods Cranberry Sauce      | Grandma Kelly's Homestead          |
| Mishi Kobe Niku                 | Tokyo Traders                      |
| Ikura                           | Tokyo Traders                      |
| Queso Cabrales                  | Cooperativa de Quesos 'Las Cabras' |
| Queso Manchego La Pastora       | Cooperativa de Quesos 'Las Cabras' |
| Konbu                           | Mayumi's                           |
| Tofu                            | Mayumi's                           |
| Genen Shouyu                    | Mayumi's                           |
| Pavlova                         | Pavlova, Ltd.                      |
| Alice Mutton                    | Pavlova, Ltd.                      |
| Carnarvon Tigers                | Pavlova, Ltd.                      |
| Teatime Chocolate Biscuits      | Specialty Biscuits, Ltd.           |
| Sir Rodney's Marmalade          | Specialty Biscuits, Ltd.           |
| Sir Rodney's Scones             | Specialty Biscuits, Ltd.           |

**mysql> explain select productos.nombre,proveedores.nombre from products inner join proveedores using (idProveedor) order by idProducto\G ;**

```

mysql> explain select productos.nombre,proveedores.nombre from
(idProveedor) order by idProducto\G ;
***** 1. row *****
      id: 1
    select_type: SIMPLE
          table: proveedores
    partitions: NULL
           type: index
possible_keys: PRIMARY
           key: nombre
        key_len: 767
           ref: NULL
          rows: 29
     filtered: 100.00
      Extra: Using index; Using temporary; Using filesort
***** 2. row *****
      id: 1
    select_type: SIMPLE
          table: productos
    partitions: NULL
           type: ref
possible_keys: idProveedor
           key: idProveedor
        key_len: 4
           ref: tiendaVirtual.proveedores.idProveedor
          rows: 2
     filtered: 100.00
      Extra: NULL
2 rows in set, 1 warning (0.00 sec)

ERROR:
No query specified

```

Al ver este explain podemos ver que utiliza un índice único en la búsqueda de la tabla proveedores que es nombre, por tanto nombre debe estar como clave única de es tabla, y luego usa en productos la clave primaria para hacer la busqueda por tanto creo que ya está bastante bien optimizada esta búsqueda ,en las dos tablas .



## Parte III: Optimización. Caché de consultas

19. Active la caché. Ejecute de nuevo las consultas de la pregunta 11. Recuerde que deberá ejecutar dos veces cada consulta, cuando esté la caché activada, para comprobar si los tiempos bajan.

Bueno como dije anteriormente en mi versión de Mysql, no se puede desactivar la caché de consultas , con ninguno de los métodos que aparecen en la documentación mysql.

```
(base) ambite@ambite-desktop:~$ mysql --version
mysql Ver 14.14 Distrib 5.7.27, for Linux (x86_64) using EditLine wrapper
(base) ambite@ambite-desktop:~$
```

Por tanto ya explique como yo para poder hacer la consultas sin coche tenia que reiniciar el servicio y borrar un directorio cada vez que hacía una consulta, para poder asegurar que no tenia guardado nada en caché.

**mysql> select count(\*) from tabla\_A;**

```
mysql> select count(*) from tabla_A;
+-----+
| count(*) |
+-----+
| 1000000 |
+-----+
1 row in set (0.13 sec)
```

El tiempo con caché 0.13 sin cache fueron 1.04 segundos.

**mysql> select aCadenaunica from tabla\_A;**

```
| cd%ZZZXIPgcpwFoGweRckL |
| cd%ZZZxtMiTaCaGqulJleL |
| cd%zzZYdQZgXkOGbkdcAqu |
| cd%zzzyTvLCxBIYvDePiij |
| cd%ZzZZNTXsPIysKTVczTk |
+-----+
1000000 rows in set (0.25 sec)
```

La primera vez sin la caché tardó 8.85 segundos con caché ha tardado 0.25segundos.

**mysql> select aCadenaSinindex from tabla\_A;**

```

| ZzZXIPgcpwFoGweRcklDwA |
| ZZZxtMiTaCaGquLJleLuJG |
| zzZYdQZgXkOGbkdcAquKqt |
| zzzYtVLCxBIYvDePiijKCJ |
| ZzzZNTXsPIysKTVczTkCXd |
+-----+
1000000 rows in set (0.25 sec)

```

La primera vez sin cache fueron 1.11 segundos y con cache han sido 0.25 segundos.

**mysql>select A\_id from tabla\_A order by aCadenaUnica**

```

| 835232 |
| 431729 |
| 724360 |
+-----+
1000000 rows in set (0.27 sec)

```

La primera vez fueron 8.68 segundos y con la consulta cacheada han sido 0.27 segundos.

**mysql>select \* from tabla\_A order by aCadenaUnica**

```

| 431729 | cd%zzzyTvLCxBIYvDePiij | zzzYtVLCxBIYvDePiijKCJ | zzzYtVLCxBIYvDePiijKCJ | 55
| 724360 | cd%ZzzZNTXsPIysKTVczTk | ZzzZNTXsPIysKTVczTkCXd | ZzzZNTXsPIysKTVczTkCXd | 89
+-----+-----+-----+-----+-----+
1000000 rows in set (1.32 sec)

```

La primera vez tardó 1.99 segundos y con la caché tarda 1.32 segundos , vemos que en este caso no ha bajado tanto y eso es por el order by, ya que se hace un filesort y eso ralentiza mucho la consulta podría ser un buen ejemplo de tabla a la que meter índices para mejorar el rendimiento de la consulta.

**mysql>select aCadenaUnica from tabla\_A order by aCadenaUnica;**

```

| cd%zzZYdQZgXkOGbkdcAqu |
| cd%zzzyTvLCxBIYvDePiij |
| cd%ZzzZNTXsPIysKTVczTk |
+-----+
1000000 rows in set (0.26 sec)

```

Para esta consulta sin la caché se tardó 8.88 segundos y con la cache se han tardado 0.26 segundos.



```
mysql>select aCadenaSinIndex from tabla_A order by aCadenaSinIndex;
```

```
| ZZZxtMiTaCaGqulJleLuJG |
| zzZYdQZgXkOGbkdcAquKqt |
| zzzyTvLCxBIYvDePiijKCJ |
| ZzzZNTXsPIysKTVczTkCXd |
+-----+
1000000 rows in set (0.26 sec)
```

Esta consulta tardaba sin cachear 1.25 segundos y ahora con la caché ha tardado 0.26 segundos.

Conclusión sobre la caché habla por si sola se ha conseguido reducir un tiempo de 8 segundos a consulta a 0.4 segundos, por lo tanto la eficiencia de búsqueda se mejora siempre con la cache, no me extraña que MySQL haya decidido dejarla por defecto y no poder actuar sobre ella, ya que lo que mejora el servicio un

20. Muestre otra consulta con join agrupada, sobre tablaA y tablaB donde la caché mejore los tiempos (indique los tiempos antes y después).

NO SE TIENE QUE HACER