

# Administración de Sistemas Gestores de Bases de Datos

PROYECTO 2º CURSO

Javier Jiménez García

PROYECTO 2º CURSO

Javier Jiménez García

## ENUNCIADO DEL PROYECTO

1. Realiza un script que cree una base de datos con al menos 5 tablas e inserta registros de datos para todas las tablas.
2. Utilizando esa base de datos, realiza al menos las siguientes rutinas:
  1. Un procedimiento almacenado en el que se hagan uso de instrucciones repetitivas y de parámetros de entrada-salida
  2. Una función que a partir de algunos parámetros de entrada devuelva un valor 0 o 1
  3. Un procedimiento almacenado que haga uso al menos de dos cursores anidados y el manejo de errores mediante un handler
  4. Un disparador como aplicación al control de valores de entrada
  5. Un disparador como aplicación al mantenimiento de campos derivados
  6. Un disparador como aplicación en las estadísticas
  7. Un evento que se realice únicamente en un intervalo de tiempo determinado (por ejemplo, un mes)
  8. Un evento para la creación de un histórico de registros eliminados.
  9. Un evento para la realización de copias de seguridad periódicas

# 1. CONSTRUCCIÓN DE LA BASE DE DATOS

## Tema y recopilación de Datos.

El tema que he escogido para la Base de Datos será el propio Instituto, CIFP Majada Marcial, que servirá para organizar y recopilar los datos de los Ciclos Formativos que imparte y las características de cada uno, como las asignaturas, las capacitaciones o capacidades que los/as alumnos/as obtendrán durante su estudio y las salidas profesionales a las que podrán optar.

La información que he usado sobre los Ciclos Formativos la he recopilado de la sección "Oferta Formativa" de la web del instituto:

[http://www.cifpmajadamarcial.com/matricula2016\\_2017/ofertaformativa.pdf](http://www.cifpmajadamarcial.com/matricula2016_2017/ofertaformativa.pdf)

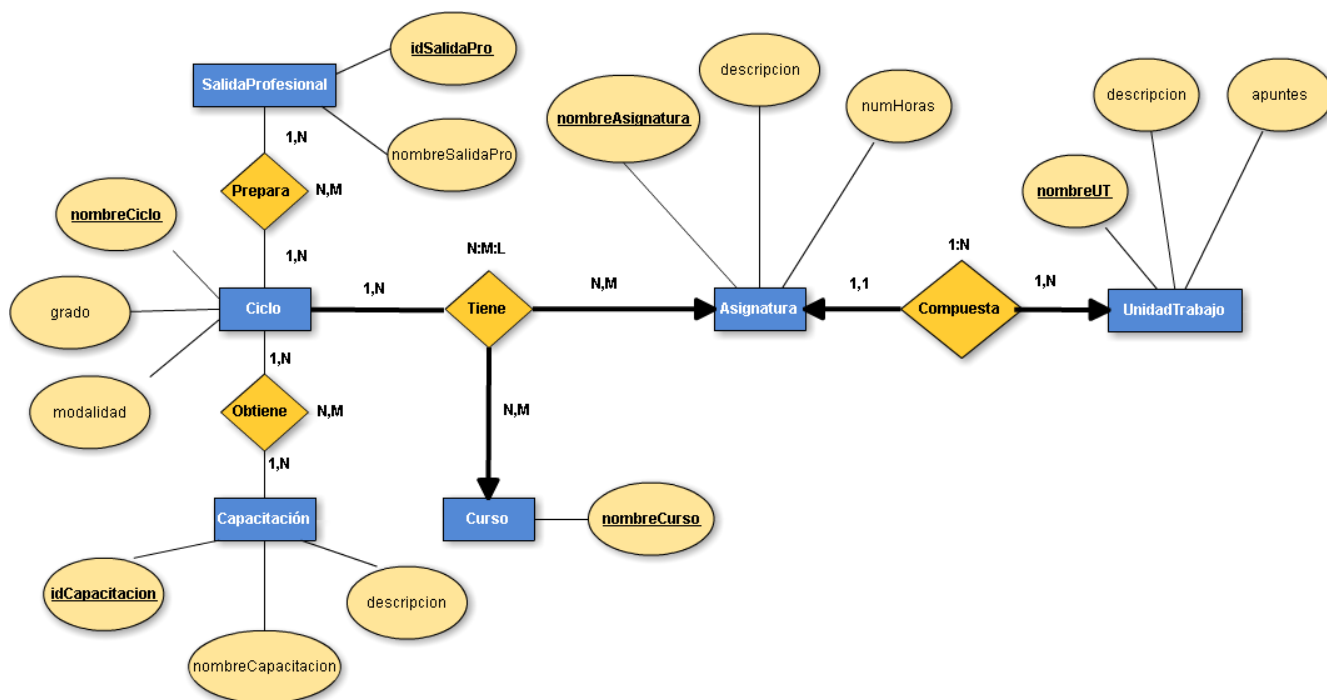
Ciclo Formativo de Grado Medio	Imagen Personal (LOE)	Estética y Belleza	1º CFGM Imagen Personal - Estética y Belleza (LOE)	Presencial
Ciclo Formativo de Grado Medio	Administración y Gestión (LOE)	Gestión Administrativa	1º CFGM Administración y Gestión - Gestión Administrativa (LOE)	Presencial
Ciclo Formativo de Grado Medio	Administración y Gestión (LOE)	Gestión Administrativa	2º CFGM Administración y Gestión - Gestión Administrativa (LOE)	Presencial
Ciclo Formativo de Grado Medio	Comercio y Marketing (LOE)	Actividades Comerciales	1º CFGM Comercio y Marketing - Actividades Comerciales (LOE)	Presencial
Ciclo Formativo de Grado Medio	Comercio y Marketing (LOE)	Actividades Comerciales	2º CFGM Comercio y Marketing - Actividades Comerciales (LOE)	Presencial
Ciclo Formativo de Grado Medio	Electricidad y Electrónica (LOE)	Instalaciones Eléctricas y Automáticas	1º CFGM Electricidad y Electrónica - Instalaciones Eléctricas y Automáticas (LOE)	Presencial
Ciclo Formativo de Grado Medio	Electricidad y Electrónica (LOE)	Instalaciones Eléctricas y Automáticas	2º CFGM Electricidad y Electrónica - Instalaciones Eléctricas y Automáticas (LOE)	Presencial
Ciclo Formativo de Grado Medio	Imagen Personal (LOE)	Peluquería y Cosmética Capilar	1º CFGM Imagen Personal - Peluquería y Cosmética Capilar (LOE)	Presencial
Ciclo Formativo de Grado Medio	Imagen Personal (LOE)	Peluquería y Cosmética Capilar	2º CFGM Imagen Personal - Peluquería y Cosmética Capilar (LOE)	Presencial
Ciclo Formativo de Grado Medio	Informática y Comunicaciones (LOE)	Sistemas Microinformáticos y Redes	1º CFGM Informática y Comunicaciones - Sistemas Microinformáticos y Redes (LOE)	Presencial
Ciclo Formativo de Grado Medio	Informática y Comunicaciones (LOE)	Sistemas Microinformáticos y Redes	2º CFGM Informática y Comunicaciones - Sistemas Microinformáticos y Redes (LOE)	Presencial
Ciclo Formativo de Grado Medio	Instalación y Mantenimiento (LOE)	Instalaciones de Producción de Calor	2º CFGM Instalación y Mantenimiento - Instalaciones de Producción de Calor (LOE)	Presencial
Ciclo Formativo de	Instalación y Mantenimiento	Instalaciones Frigoríficas y de	1º CFGM Instalación y Mantenimiento - Instalaciones	

Sólo he usado parte de esa información, necesaria para el desarrollo del proyecto y prueba de los scripts y rutinas almacenadas.

Utilizaré entre 200 y 300 registros en total.

## Diagrama Entidad-Relación

Durante el proceso de inserción de registros y prueba de rutinas almacenadas, el diagrama ha sufrido algunas modificaciones en cuanto a los atributos, pero la estructura es la siguiente:



Esta Base de Datos puede servir para cualquier centro formativo, teniendo como entidad principal los Ciclos Formativos.

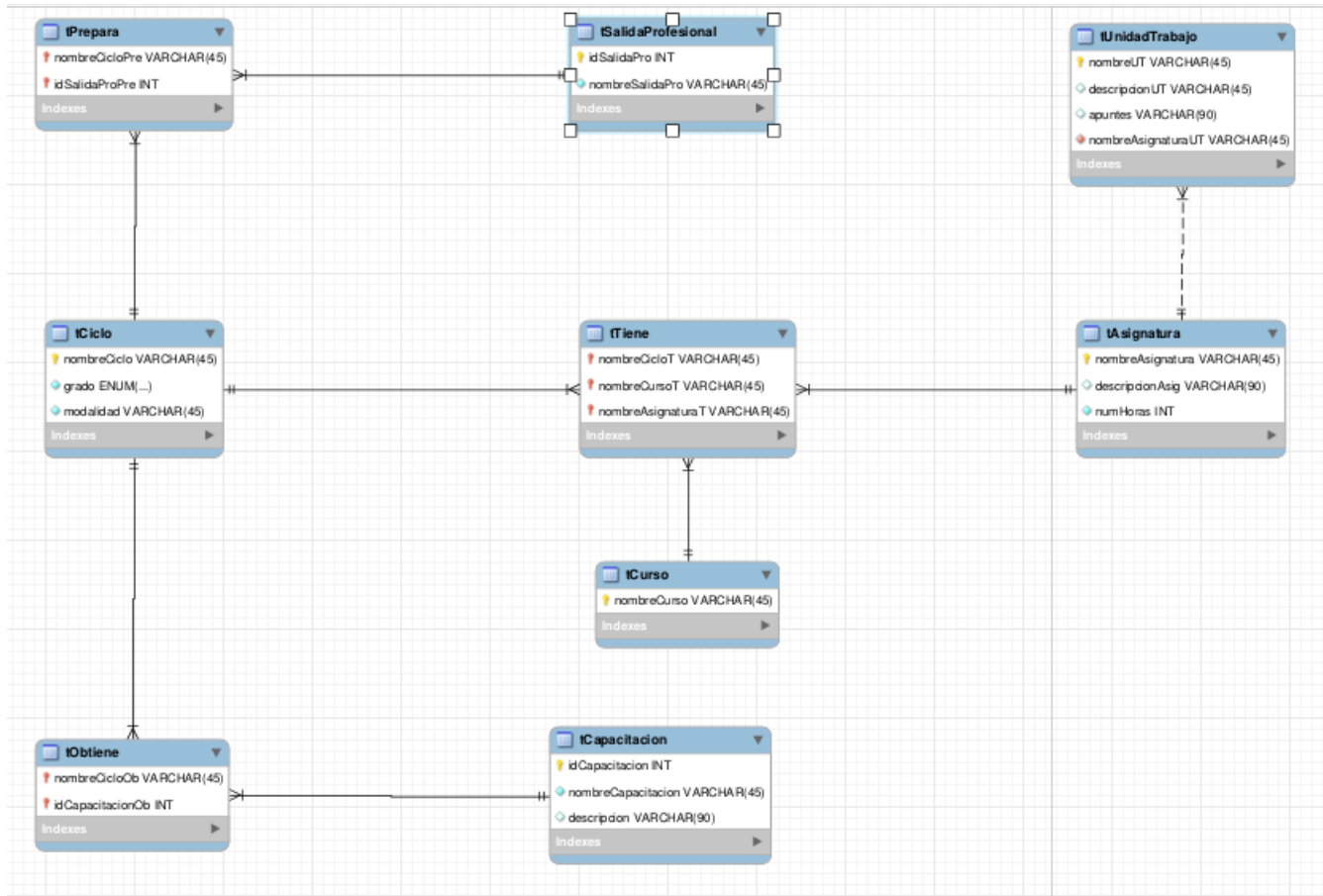
Desgranamos la estructura del diagrama:

- Ciclo: cada ciclo formativo tiene como atributo principal su nombre "nombreCiclo" (Aunque se podría haber añadido un código o Id). Especificaremos también su "grado", pudiendo ser "Medio" o "Superior", y su "modalidad", que será "Presencial", "A distancia" o "Semi-presencial", según la oferta formativa del Centro.
- Capacitación: durante el estudio de un ciclo se obtienen unas capacidades o "capacitaciones" que podrán ser específicas de cada ciclo u obtenidas a través de ciclos diferentes. Las capacitaciones tendrán una "idCapacitación" que será su atributo principal y una "descripción". (El nombre se ha eliminado ya que la fuente no especifica nombres para las mismas).

- Salida Profesional: cada ciclo puede preparar al alumno para unas específicas salidas profesionales, aunque puede haber varios ciclos que preparen para una misma salida profesional. Tendrán los atributos "idSalidaPro", que será el principal, y un nombre.
- Curso: los ciclos formativos tienen cursos, que pueden ser "Primero", "Segundo", y, en algunos casos, "Tercero", que coinciden con los años de duración.
- Asignaturas: hay asignaturas generales para todos los ciclos y específicas de cada uno. "nombreAsignatura" será el principal atributo y contendrá las siglas o denominación de la asignatura, que consisten en 3 letras identificativas únicas de cada una. En "descripción" contendrá el nombre completo de la asignatura (Aunque bien podrían haberse modificado estos atributos y añadir un código) y durante el curso cada asignatura tendrá un número de horas "numHoras" máximo previstas.
- UnidadTrabajo: esta entidad podría haber sido opcional, pero puede ser interesante para poder recopilar apuntes de cada asignatura y nos servirá para realizar las rutinas almacenadas y demás. Cada Unidad de Trabajo contendrá un nombre (EJ: UT1 Diseño de Base de Datos), una descripción, que será opcional y un atributo "apuntes" que también será opcional y se compondrá de un enlace hacia los mismos, por ejemplo, en un servidor FTP o un servicio en la nube como Google Drive.

## Esquema Relacional

El esquema está realizado en MySQL WorkBench, conteniendo el número final total de tablas con sus relaciones, claves primarias y foráneas.



Este esquema se adjunta en un archivo aparte.

## Scripts de creación de la Base de Datos y de Inserción de registros

Los archivos con los scripts de creación e inserción también se adjuntan aparte.

Captura script creación:

```
CREATE DATABASE IF NOT EXISTS `dbInstituto` /*!40100 DEFAULT CHARACTER SET utf8 COLLATE utf8_spanish_ci */;
USE `dbInstituto`;

--
-- Table structure for table `tAsignatura`
--

DROP TABLE IF EXISTS `tAsignatura`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tAsignatura` (
  `nombreAsignatura` char(3) NOT NULL,
  `descripcionAsig` varchar(90) DEFAULT NULL,
  `numHoras` int(11) NOT NULL,
  PRIMARY KEY (`nombreAsignatura`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tCapacitacion`
--

DROP TABLE IF EXISTS `tCapacitacion`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tCapacitacion` (
  `idCapacitacion` int(11) AUTO_INCREMENT NOT NULL,
  `descripcion` text DEFAULT NULL,
  PRIMARY KEY (`idCapacitacion`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Captura script inserción:

```
-- INSERCIÓN DE REGISTROS EN LA BD dbInstituto
--

-- TABLA tAsignatura
--
-- Asignaturas 2º ASIR - Las asignaturas comunes se omiten en las siguientes tablas
INSERT INTO tAsignatura VALUES('ADD','Administración de Sistemas Operativos',126);
INSERT INTO tAsignatura VALUES('SRD','Servicios de Red e Internet',126);
INSERT INTO tAsignatura VALUES('IMW','Implantación de Aplicaciones Web',126);
INSERT INTO tAsignatura VALUES('ADE','Administración de Sistemas Gestores de Bases de Datos',63);
INSERT INTO tAsignatura VALUES('SGY','Seguridad y Alta Disponibilidad',126);
INSERT INTO tAsignatura VALUES('EMR','Empresa e Iniciativa Emprendedora',63);
INSERT INTO tAsignatura VALUES('PMR','Proyecto de Administración de Sistemas Informáticos en Red',64);
INSERT INTO tAsignatura VALUES('FCT','Formación en Centros de Trabajo',346);
-- Asignaturas 1º ASIR
INSERT INTO tAsignatura VALUES('PNI','Planificación y Administración de Redes',160);
INSERT INTO tAsignatura VALUES('IDP','Implantación de Sistemas Operativos',256);
INSERT INTO tAsignatura VALUES('FUW','Fundamentos de Hardware',96);
INSERT INTO tAsignatura VALUES('LND','Lenguajes de Marcas y Sistemas de Gestión de Información',128);
```

## 2. RUTINAS ALMACENADAS

Usando la Base de Datos que he creado, realizaré una serie de rutinas almacenadas para la automatización de ciertos procesos, que pueden servir para ahorrar mucho tiempo y muchas líneas de código cuando la Base de Datos sea totalmente funcional y esté en producción, y también para optimizar las consultas.

Cada rutina contendrá una explicación de su funcionamiento, capturas del script, capturas de las comprobaciones y se adjuntarán en un archivo aparte.

### 2.1 PROCEDIMIENTO ALMACENADO EN EL QUE SE HAGAN USO DE INSTRUCCIONES REPETITIVAS Y DE PARÁMETROS DE ENTRADA-SALIDA

#### FUNCIONAMIENTO DEL SCRIPT:

Llamamos al procedimiento pasándole como parámetros el nombre de un Ciclo Formativo y un número. El bucle itera ese número de veces y cuenta cuantas salidas tiene el ciclo introducido dentro del número de iteraciones, comparando la variable “cont” (que empieza con el valor 0) con la “idSalidaProPre”.

**\*Nota:** En mi versión de MySQL WorkBench tengo un fallo a la hora de llamar a los procedimientos, dando error si declaro una variable (EJ: SET @x=0;) antes de la llamada, por lo que para la comprobación he modificado el script como se verá más adelante.

#### SCRIPT:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS ciclo_salidas$$
CREATE PROCEDURE ciclo_salidas(IN ciclo VARCHAR(90), IN numero INT, OUT salidas INT)
BEGIN
    DECLARE cont INT DEFAULT 0;
    WHILE cont < numero DO
        IF ciclo = (SELECT nombreCicloPre FROM tPrepara WHERE idSalidaProPre = cont) THEN
            SET salidas = salidas + 1;
        END IF;
        SET cont = cont + 1;
    END WHILE;
END $$

SET @x=0;
CALL ciclo_salidas('Administración de Sistemas Informáticos', 10, @x);

SELECT @x;
```



## COMPROBACIÓN:

Modificando el script como mencioné anteriormente, funciona correctamente: pasamos como parámetros el nombre del ciclo 'Administración de Sistemas Informáticos en Red' y el número 20, el resultado es 11, el número de salidas que hay desde la id 1 hasta la id 20 con ese nombre de ciclo.

```
1 DELIMITER $$
2 DROP PROCEDURE IF EXISTS ciclo_salidas$$
3 CREATE PROCEDURE ciclo_salidas(IN ciclo VARCHAR(90),IN numero INT)
4 BEGIN
5     DECLARE cont,salidas INT DEFAULT 0;
6     WHILE cont<numero DO
7         IF ciclo=(SELECT nombreCicloPre FROM tPrepara WHERE idSalidaProPre=cont) THEN
8             SET salidas=salidas+1;
9         END IF;
10        SET cont=cont+1;
11    END WHILE;
12    SELECT CONCAT('Salidas: ',salidas);
13 END $$
14
15
16 CALL ciclo_salidas('Administración de Sistemas Informáticos en Red',20);|
17
18
```

Grid Filter Rows: Export: Wrap Cell Content:

CONCAT('Salidas: ',salidas)
Salidas: 11

## 2.2 FUNCIÓN QUE, A PARTIR DE ALGUNOS PARÁMETROS DE ENTRADA, DEVUELVA UN VALOR 0 ó 1

### FUNCIONAMIENTO DEL SCRIPT:

Esta función nos devuelve un 1 si el ciclo introducido tiene la modalidad introducida.

### SCRIPT:

```
DELIMITER $$
DROP FUNCTION IF EXISTS modalidad_ciclo$$
CREATE FUNCTION modalidad_ciclo(ciclo VARCHAR(90),modal VARCHAR(15)) RETURNS INT DETERMINISTIC
BEGIN
    DECLARE resultado INT DEFAULT 0;
    DECLARE moda VARCHAR(15);
    SELECT modalidad INTO moda FROM tCiclo WHERE nombreCiclo=ciclo;
    IF modal<>moda THEN
        SET resultado=0;
    ELSE
        SET resultado=1;
    END IF;
    RETURN resultado;
END $$

SELECT modalidad_ciclo('Dietética','Semi-presencial');
```

## COMPROBACIÓN:

Introducimos el ciclo 'Dietética' y la modalidad 'Semi-presencial'. El resultado es 0 porque ese ciclo no se imparte en esa modalidad.

```
17 • SELECT modalidad_ciclo('Dietética','Semi-presencial');|
```

modalidad_ciclo('Dietética','Semi-presencia
0

Sin embargo, si introducimos el mismo ciclo con la modalidad 'Presencial', nos devuelve "1" porque ese ciclo si se imparte en esa modalidad.

```
17 • SELECT modalidad_ciclo('Dietética','Presencial');|
```

modalidad_ciclo('Dietética','Presencial')
1

## 2.3 PROCEDIMIENTO ALMACENADO QUE HAGA USO AL MENOS DE DOS CURSORES ANIDADOS Y EL MANEJO DE ERRORES MEDIANTE UN HANDLER

### FUNCIONAMIENTO DEL SCRIPT:

*Este procedimiento nos cuenta el número de salidas profesionales a las que prepara cada ciclo, y nos muestra el Ciclo con mayor número de salidas y el Ciclo con menor número.*

*El script se puede complicar un poco más introduciendo un tercer cursor y comparando también el número de capacitaciones de cada ciclo.*

### SCRIPT:

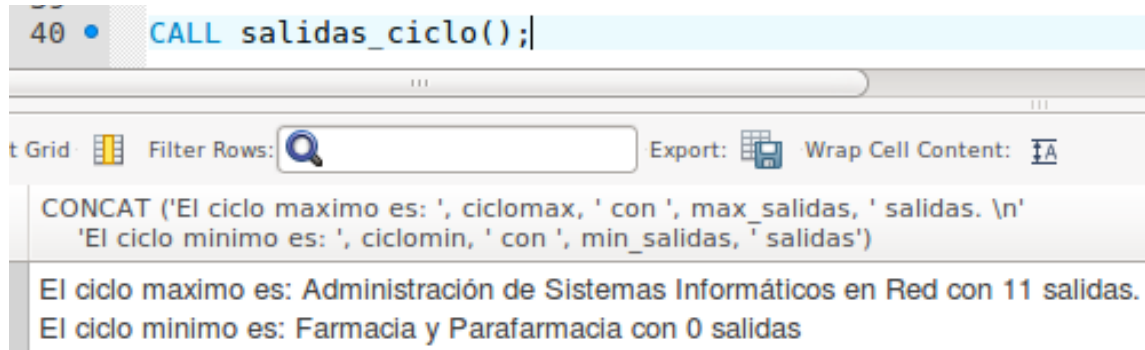
```
DROP PROCEDURE IF EXISTS salidas_ciclo$$
CREATE PROCEDURE salidas_ciclo()
BEGIN
    DECLARE num_salidas,max_salidas,min_salidas INT;
    DECLARE ciclo,ciclomax,ciclomin VARCHAR(90);
    DECLARE fin BOOL DEFAULT 0;
    DECLARE ciclo_cursor CURSOR FOR SELECT nombreCiclo FROM tCiclo;
    DECLARE salidas_cursor CURSOR FOR SELECT nombreCicloPre FROM tPrepara WHERE nombreCicloPre=ciclo;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin=1;
    SET max_salidas=0;
    OPEN ciclo_cursor;

    SELECT count(*) INTO min_salidas FROM tPrepara;
    ciclo_loop: LOOP
        FETCH ciclo_cursor INTO ciclo;
        IF fin=1 THEN LEAVE ciclo_loop; END IF;
        OPEN salidas_cursor; SET num_salidas=0;
        salidas_loop: LOOP
            FETCH salidas_cursor INTO ciclo;
            IF fin=1 THEN LEAVE salidas_loop; END IF;
            SET num_salidas=num_salidas+1;
        END LOOP salidas_loop;
        CLOSE salidas_cursor; SET fin=0;
        IF max_salidas<=num_salidas THEN
            SET max_salidas=num_salidas;
            SET ciclomax=ciclo;
        ELSEIF min_salidas>=num_salidas THEN
            SET min_salidas=num_salidas;
            SET ciclomin=ciclo;
        END IF;
    END LOOP ciclo_loop;

    CLOSE ciclo_cursor;
    SELECT CONCAT ('El ciclo maximo es: ', ciclomax, ' con ', max_salidas, ' salidas. \n'
    'El ciclo minimo es: ', ciclomin, ' con ', min_salidas, ' salidas');
END $$
```

## COMPROBACIÓN:

Con los registros actuales de la Base de Datos, el procedimiento nos indica que el ciclo 'Administración de Sistemas Informáticos en Red' es el ciclo con mayor número de salidas profesionales, y, sin embargo, como el ciclo 'Farmacia y Parafarmacia' no tiene registros insertados, nos muestra que es el que tiene menor número de salidas, con 0 salidas.



## 2.4 DISPARADOR COMO APLICACIÓN AL CONTROL DE VALORES DE ENTRADA

### FUNCIONAMIENTO DEL SCRIPT:

Imaginemos que el Gobierno indica un número máximo de horas para cualquier asignatura de 126 horas. Este Trigger limita el número de horas a 126 y, si intentamos introducir un valor superior, nos lo deja en esa cantidad.

### SCRIPT:

```
DELIMITER $$
DROP TRIGGER IF EXISTS control_horas$$
CREATE TRIGGER control_horas BEFORE INSERT ON tAsignatura FOR EACH ROW
BEGIN
    IF NEW.numHoras>126 THEN
        SET NEW.numHoras=126;
    END IF;
END $$
```

## COMPROBACIÓN:

Insertamos una nueva asignatura que tendrá 128 horas y como vemos, la limitación del trigger nos actualiza el número de horas a 126.

```
7 • insert into tAsignatura VALUES('AAA','Asignatura nueva',128);
8 • SELECT * FROM tAsignatura WHERE nombreAsignatura='AAA';
```

nombreAsignatura	descripcionAsig	numHoras
AAA	Asignatura nueva	126

## 2.5 DISPARADOR COMO APLICACIÓN AL MANTENIMIENTO DE CAMPOS DERIVADOS

### FUNCIONAMIENTO DEL SCRIPT:

Cuando damos de alta un nuevo registro en la tabla tCiclo, este disparador comprueba si es de grado superior. Si es así, le asigna unas asignaturas por defecto que son comunes para todos los ciclos de grado superior, además, si el ciclo es de modalidad 'A distancia' tendrá las asignaturas comunes pero repartidas en un tercer año de curso. Se puede añadir también una condición para los Ciclos Formativos de Grado Medio.

### SCRIPT:

```
DELIMITER $$
DROP TRIGGER IF EXISTS asig_comunes$$
CREATE TRIGGER asig_comunes AFTER INSERT ON tCiclo FOR EACH ROW
BEGIN
    IF NEW.grado='Superior' THEN
        IF NEW.modalidad='A distancia' THEN
            INSERT INTO tTiene VALUES(NEW.nombreCiclo,'Primero','FOL');
            INSERT INTO tTiene VALUES(NEW.nombreCiclo,'Tercero','EMR');
            INSERT INTO tTiene VALUES(NEW.nombreCiclo,'Tercero','FCT');
        ELSEIF NEW.modalidad='Presencial' THEN
            INSERT INTO tTiene VALUES(NEW.nombreCiclo,'Primero','FOL');
            INSERT INTO tTiene VALUES(NEW.nombreCiclo,'Segundo','EMR');
            INSERT INTO tTiene VALUES(NEW.nombreCiclo,'Segundo','FCT');
        END IF;
    END IF;
END $$
```

## COMPROBACIÓN:

Insertamos un nuevo Ciclo de Grado Superior y modalidad Presencial y comprobamos que se le asignan (dando de alta los registros correspondientes en la tabla tTiene) las asignaturas comunes:

```
10 • INSERT INTO tCiclo VALUES('Administración y Finanzas','Superior','Presencial');
11 • SELECT * FROM tTiene WHERE nombreCicloT='Administración y Finanzas';
```

nombreCicloT	nombreCursoT	nombreAsignaturaT
Administración y Finanzas	Primero	FOL
Administración y Finanzas	Segundo	EMR
Administración y Finanzas	Segundo	FCT

## 2.6 DISPARADOR COMO APLICACIÓN EN LAS ESTADÍSTICAS

### FUNCIONAMIENTO DEL SCRIPT:

Queremos llevar un control en la Base de Datos de las Unidades de Trabajo que los profesores dan de alta durante el curso. Para ello programamos un disparador para que cada vez que se de de alta una nueva Unidad de Trabajo, los datos de esta se copien en la tabla “registroUT”

### SCRIPT:

```
DROP TABLE IF EXISTS registroUT;
CREATE TABLE registroUT(
    nombreUTreg VARCHAR(90) PRIMARY KEY NOT NULL,
    descripcionUTreg TEXT DEFAULT NULL,
    apuntes VARCHAR(90) DEFAULT NULL,
    nombreAsigReg CHAR(3) NOT NULL,
    fecha DATE NOT NULL
);

DELIMITER $$
DROP TRIGGER IF EXISTS reg_UT$$
CREATE TRIGGER reg_UT AFTER INSERT ON tUnidadTrabajo FOR EACH ROW
BEGIN
    INSERT INTO registroUT VALUES(NEW.nombreUT,NEW.descripcionUT,NEW.apuntes,NEW.nombreAsignaturaUT,
    CURDATE());
END $$
```

## COMPROBACIÓN:

Comprobamos insertando nuevos registros en la tabla tUnidadTrabajo y vemos que también se insertan en la tabla registroUT, añadiendo la fecha en la que se insertó:

```
13 • INSERT INTO tUnidadTrabajo
14   VALUES('UT1 Administración','Sobre Administración','www.apuntes.local','AAA');
15 • SELECT * FROM registroUT;
```

nombreUTreg	descripcionUTreg	apuntes	nombreAsigReg	fecha
UT1 Administración	Sobre Administración	www.apuntes.local	AAA	2017-02-24

## 2.7 EVENTO QUE SE REALICE ÚNICAMENTE EN UN INTERVALO DE TIEMPO DETERMINADO

### FUNCIONAMIENTO DEL SCRIPT:

Durante el verano el número de horas máximo para el siguiente curso de las asignaturas que tienen 126 horas se incrementa a 130, por lo que queremos conservar los datos de estas asignaturas por si vuelven a cambiar más adelante, o para llevar un control. Además queremos actualizar estos registros, por lo que programamos el siguiente evento.

### SCRIPT:

```
DROP TABLE IF EXISTS numHorasAntiguas;
CREATE TABLE numHorasAntiguas(
  nombreAsig CHAR(3),
  numHoras INT(11)
);

DELIMITER $$
DROP EVENT IF EXISTS actualiza_hora$$
CREATE EVENT actualiza_hora
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 MONTH ENABLE
DO
BEGIN
  DECLARE nombre CHAR(3);
  DECLARE horas INT(11);
  SELECT nombreAsignatura,numHoras INTO nombre,horas FROM tAsignatura WHERE numHoras=126;
  INSERT INTO numHorasAntiguas VALUES(nombre,horas);
  UPDATE tAsignatura SET numHoras=130 WHERE numHoras=126;
END;$$
```

## COMPROBACIÓN:

Para comprobar el funcionamiento modificamos el script para que se ejecute en 2 minutos y mostramos el resultado de la tabla numHorasAntiguas:

```
48 • select * from numHorasAntiguas;|
49
```

nombreAsig	numHoras
AAA	126
ADD	126
IMW	126

Y en la tabla tAsignatura comprobamos que esos valores se actualizan:

```
47 • SELECT * FROM tAsignatura;|
```

nombreAsignatura	descripcionAsig	numHoras
AAA	Asignatura nueva	130
ADD	Administración de Sistemas Operativos	130



## 2.8 EVENTO PARA LA CREACIÓN DE UN HISTÓRICO DE REGISTROS ELIMINADOS

### FUNCIONAMIENTO DEL SCRIPT:

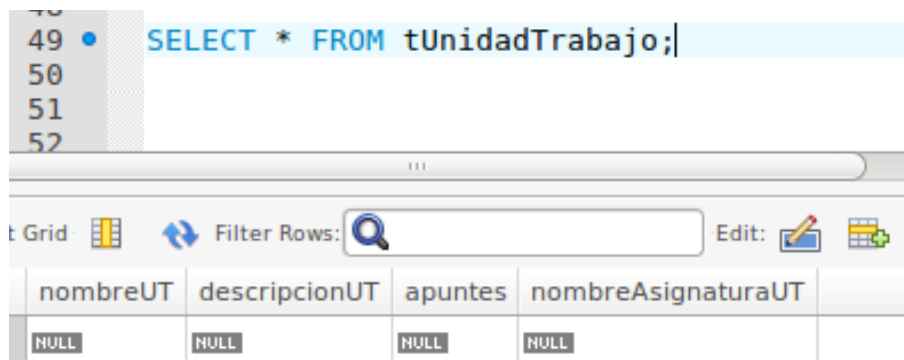
Cada año al llegar Agosto, guardamos todas las Unidades de Trabajo insertadas por los profesores en la tabla borradosUT para su almacenamiento, control e incluso para poder recuperarlos, y, al mismo tiempo, borramos los datos de la tabla tUnidadTrabajo para que el nuevo curso no tenga ningún apunte subido y cada profesor valore y suba los que necesite.

### SCRIPT:

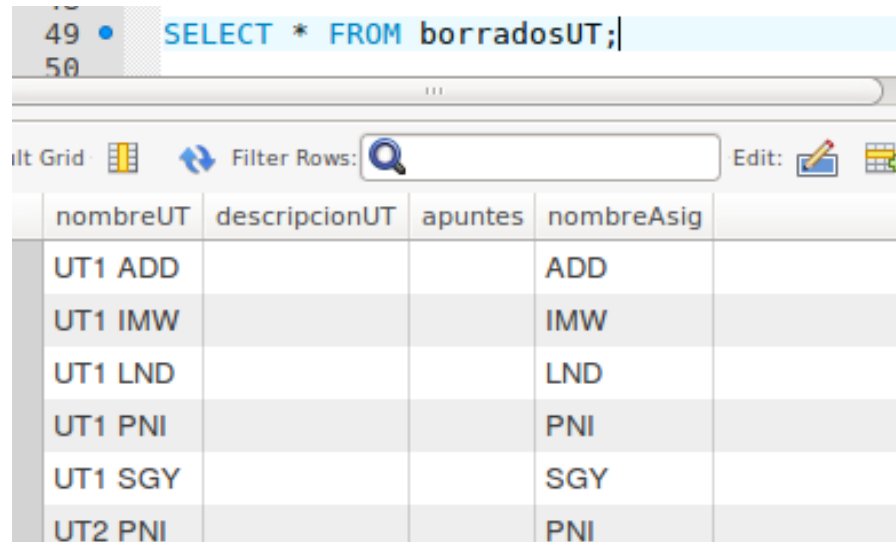
```
DROP TABLE IF EXISTS borradosUT;  
CREATE TABLE borradosUT(  
    nombreUT VARCHAR(90) PRIMARY KEY NOT NULL,  
    descripcionUT TEXT DEFAULT NULL,  
    apuntes VARCHAR(90) DEFAULT NULL,  
    nombreAsig CHAR(3) NOT NULL  
);  
  
DELIMITER $$  
DROP EVENT IF EXISTS borrados$$  
CREATE EVENT borrados  
ON SCHEDULE EVERY 1 YEAR STARTS '2017-08-01'  
DO  
BEGIN  
  
    INSERT INTO borradosUT SELECT * FROM tUnidadTrabajo;  
    DELETE FROM tUnidadTrabajo;  
  
END;$$
```

### COMPROBACIÓN:

Comprobamos que se borran de la tabla tUnidadTrabajo:



Y que se guardan en la tabla borradosUT:



The screenshot shows a database query window with the following SQL statement: `SELECT * FROM borradosUT;`

Below the query, there is a table with the following data:

nombreUT	descripcionUT	apuntes	nombreAsig
UT1 ADD			ADD
UT1 IMW			IMW
UT1 LND			LND
UT1 PNI			PNI
UT1 SGY			SGY
UT2 PNI			PNI

## 2.9 EVENTO PARA LA REALIZACIÓN DE COPIAS DE SEGURIDAD PERIÓDICAS

### FUNCIONAMIENTO DEL SCRIPT:

La única forma que he podido encontrar para realizar copias de seguridad mediante eventos es la siguiente, aunque podríamos haber programado un procedimiento que realice las copias y llamarlo a través de un evento periódicamente, pero no lo he probado. Este evento exporta las tablas a ficheros de tipo txt:

### SCRIPT:

```
DELIMITER $$
DROP EVENT IF EXISTS backups$$
CREATE EVENT backups
ON SCHEDULE EVERY 1 MONTH STARTS NOW()
DO
BEGIN
    SELECT * FROM registroUT INTO OUTFILE 'copia_registroUT.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tAsignatura INTO OUTFILE 'copia_tAsignatura.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tCapacitacion INTO OUTFILE 'copia_tCapacitacion.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tCiclo INTO OUTFILE 'copia_tCiclo.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tCurso INTO OUTFILE 'copia_tCurso.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tObtiene INTO OUTFILE 'copia_tObtiene.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tPrepara INTO OUTFILE 'copia_tPrepara.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tSalidaProfesional INTO OUTFILE 'copia_tSalidaProfesional.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tTiene INTO OUTFILE 'copia_tTiene.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tUnidadTrabajo INTO OUTFILE 'copia_tUnidadTrabajo.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
END;$$
```

## COMPROBACIÓN:

*Modificamos el script para que se active en 1 minuto:*

```
DELIMITER $$
DROP EVENT IF EXISTS backups$$
CREATE EVENT backups
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 MINUTE
DO
BEGIN
    SELECT * FROM registroUT INTO OUTFILE 'copia_registroUT.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tAsignatura INTO OUTFILE 'copia_tAsignatura.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tCapacitacion INTO OUTFILE 'copia_tCapacitacion.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tCiclo INTO OUTFILE 'copia_tCiclo.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tCurso INTO OUTFILE 'copia_tCurso.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tObtiene INTO OUTFILE 'copia_tObtiene.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tPrepara INTO OUTFILE 'copia_tPrepara.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tSalidaProfesional INTO OUTFILE 'copia_tSalidaProfesional.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tTiene INTO OUTFILE 'copia_tTiene.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
    SELECT * FROM tUnidadTrabajo INTO OUTFILE 'copia_tUnidadTrabajo.txt' FIELDS TERMINATED BY ';'
    OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n\r';
END;$$
```

*Y comprobamos que crea todos los ficheros con las copias de seguridad de cada tabla.*

```
root@Boqueron:/var/lib/mysql/dbInstituto# ls
asig_comunes.TRN      copia_tSalidaProfesional.txt  tCapacitacion.frm      tPrepara.ibd
control_horas.TRN     copia_tTiene.txt             tCapacitacion.ibd     tSalidaProfesional.frm
copia_registroUT.txt  copia_tUnidadTrabajo.txt     tCiclo.frm            tSalidaProfesional.ibd
copias                db.opt                       tCiclo.ibd            tTiene.frm
copia_tAsignatura.txt registroUT.frm               tCiclo.TRG            tTiene.ibd
copia_tCapacitacion.txt registroUT.ibd              tCurso.frm            tUnidadTrabajo.frm
copia_tCiclo.txt      reg_UT.TRN                  tCurso.ibd            tUnidadTrabajo.ibd
copia_tCurso.txt      tAsignatura.frm             tObtiene.frm          tUnidadTrabajo.TRG
copia_tObtiene.txt    tAsignatura.ibd            tObtiene.ibd
copia_tPrepara.txt    tAsignatura.TRG            tPrepara.frm
```