**[2 puntos] CONSULTA 1**

Creamos el procedimiento sobre la base de datos, creando una variable partidoValido donde el partido de la tabla senador si no es el indicado en la tabla, muestra el error dado, mediante la condición IF donde igualando a 0 dará el mensaje de error por el que los parámetros son inesperados y donde introduciendo los nombres de los partidos con las características especificadas del tipo de entrada nos muestra las diferentes tablas.

DROP PROCEDURE IF EXISTS verIdCompradas;

DELIMITER $$

CREATE PROCEDURE verIdCompradas(

IN partidoP VARCHAR(4),

OUT personasCompradas INTEGER)

BEGIN

DECLARE partidoValido VARCHAR(4);

SELECT COUNT(\*) INTO partidoValido FROM senador

WHERE partido = partidoP;

IF partidoValido=0

THEN SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Parámetros inesperados\n\*\*\*\*\*\n====> Mensaje de error.\n\*\*\*\*\*\n';

ELSE

SELECT COUNT(\*) INTO personasCompradas

FROM id\_digital

WHERE comprado = '1'

AND senador\_corrupto IN (SELECT dni FROM senador

WHERE partido = partidoP);

SELECT id, circunscripcion FROM id\_digital

WHERE comprado = '1'

AND senador\_corrupto IN (SELECT dni FROM senador

WHERE partido = partidoP)

ORDER BY circunscripcion;

END $$

DELIMITER ;

Sabiendo los nombres de los partidos, hacemos la llamada correspondiente a cada partido, y también la llamada

CALL verIdCompradas('DAM',@personasCompradas);

CALL verIdCompradas('DAW',@personasCompradas);

CALL verIdCompradas('PRO',@personasCompradas);

CALL verIdCompradas('PHT',@personasCompradas);

CALL verIdCompradas('ASIR',@personasCompradas);

CALL verIdCompradas('1',@personasCompradas); (donde dará el error en el parámetro introducido)

**[2 puntos] CONSULTA 2**

Crea una función que reciba una id(digital) y un dni y devuelva 1 si la circunscripción de la id es la misma que la circ\_presenta del segundo, que por obligación debe ser un senador o la función falla. Si las circunscripciones son diferentes, devuelve 0.

En esta función creada se han introducido los parámetros exigidos, declarando las variables para poder almacenar la circunscripción de la tabla id\_digital, circunscripción y partido de la tabla senador y una variable para almacenar el numero de coincidencias, mediante COUNT, contando las filas que cumplen el criterio del id\_digital con el id proporcionado, cuando las coincidencias sean igual a 0 mostrará el error por los parámetros introducidos y mediante un IF creamos las condiciones que cumplan los criterios de coincidencia que la circ\_presenta debe ser un senador.

DROP FUNCTION IF EXISTS fn\_circunscripcionId; // Eliminamos función por si existe previamente

DELIMITER $$

CREATE FUNCTION fn\_circunscripcionId(

id\_digitalParam VARCHAR(10), dniParam VARCHAR(10)) // Parámetros de entrada id\_digital y dni

RETURNS INT // Devuelve un valor entero, en este caso 0 ó 1

READ SQL DATA // Función que realiza consultas, en este caso leer consulta

DETERMINISTIC

BEGIN

DECLARE circ\_id\_digital VARCHAR(10);

DECLARE circ\_senador VARCHAR(10);

DECLARE partido\_senador VARCHAR(10);

DECLARE circ\_count INT DEFAULT 0;

SELECT COUNT(\*) INTO circ\_count FROM id\_digital WHERE id = id\_digitalParam;

IF circ\_count = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Parámetros inesperados\n\*\*\*\*\*\n====> Mensaje de error.\n\*\*\*\*\*\n';

ELSE

SELECT circunscripcion INTO circ\_id\_digital FROM id\_digital WHERE id = id\_digitalParam;

SELECT circ\_presenta, partido INTO circ\_senador, partido\_senador FROM senador WHERE dni = dniParam;

IF circ\_id\_digital = circ\_senador THEN

SET circ\_count = 1;

ELSE

SET circ\_count = 0;

END IF;

END IF;

RETURN circ\_count;

END $$

DELIMITER ;

Comprobacion:

SELECT fn\_comprobarCircunscripcion('3A7F9B4C5D', '89012345H') AS COMPROBACION;

Da como resultado 1

SELECT fn\_comprobarCircunscripcion('3A7F9B4C5D', '23456789B') AS COMPROBACION;

Da como resultado 0, circunscripciones diferentes.

Crea otra función que, usando la función anterior, devuelva el número de votos en vota\_sen que no cumplen la restricción de que las circunscripciones sean iguales

DROP FUNCTION IF EXISTS fn\_circunscripcionId; // Eliminamos función por si existe previamente

DELIMITER $$

CREATE FUNCTION fn\_numeroVotosCircunscripcion();

RETURNS INTEGER // Devuelve un valor entero, en este caso 0 ó 1

READ SQL DATA // Función que realiza consultas, en este caso leer consulta

DETERMINISTIC

BEGIN

DECLARE total\_votos INT;

DECLARE votos\_incorrectos INT;

SET total\_votos = (SELECT COUNT(\*) FROM vota\_sen);

SET votos\_incorrectos = (SELECT COUNT(\*) FROM vota\_sen vs

LEFT JOIN senador s ON vs.dni\_senador = s.dni

LEFT JOIN id\_digital id ON s.circ\_presenta = id.circunscripcion

WHERE vs.dni\_senador IS NOT NULL AND id.id IS NOT NULL

AND NOT fn\_comprobarCircunscripcion(id, dni ));

RETURN votos\_incorrectos;

END $$

DELIMITER ;

Llamada a la función.

SELECT fn\_numeroVotosCircunscripcion();

Sale 0 en la función.

**[2 puntoS] CONSULTA 3**

Trigger de antes del borrado, donde queremos proteger los borrados no deseados por medio del contador inerventoresContador y el contador de centros. Si intentamos borrar un interventor nos dará error.

DROP TRIGGER IF EXISTS asegurarParticipacion1n;

DELIMITER $$

CREATE TRIGGER asegurarParticipacion1n;

BEFORE DELETE ON centro

FOR EACH ROW

BEGIN

DECLARE interventoresContador INTEGER

// Contador de los interventores

SELECT COUNT(\*) INTO interventoresContador FROM interventor

WHERE id\_centro = OLD.id\_centro;

// Condición que realiza la comprobación

IF interventoresContador < 1 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT ='Operación no permitida\n\*\*\*\*\*\n====> Mensaje de error.\n\*\*\*\*\*\n';

END IF;

END $$

DELIMITER ;

Haremos la comprobación de la prueba de borrado con

DELETE FROM centro WHERE id\_centro = 'C007';

Creamos el trigger antes de la actualización, para proteger las actualizaciones no deseadas Declaramos una variable centrosContador para conocer cuántos centros se actualizan. La condición incluye la comprobación de que hay al menos un interventor asociado al centro. Saltaría mensaje de error si no se dieran las condiciones que no permitiría actualizar la tabla centro.

DROP TRIGGER IF EXISTS asegurarParticipacion1n;

DELIMITER $$

CREATE TRIGGER asegurarParticipacion1n;

BEFORE UPDATE ON centro

FOR EACH ROW

BEGIN

DECLARE interventoresContador INTEGER

DECLARE centrosContador INTEGER

// Contador de los interventores

SELECT COUNT(\*) INTO interventoresContador FROM interventor

WHERE id\_centro = NEW.id\_centro;

// Contador de los centros

SELECT COUNT(\*) INTO centrosContador FROM centro;

// Condición que realiza la comprobación.

IF interventoresContador < 1 OR centrosContador <=1 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT ='Operación no permitida\n\*\*\*\*\*\n====> Mensaje de error.\n\*\*\*\*\*\n';

END IF;

RETURN NEW;

END $$

Haremos la comprobación de la prueba de borrado con:

UPDATE centro SET id\_centro = 'C009’ WHERE id\_centro = 'C007';

**[2 puntos] CONSULTA 4**

No me ha dado tiempo a realizar los dos últimos ejercicios pero los haré poco a poco. En los dos primeros ejercicios he realizado comprobaciones de la función y el procedimiento por la herramienta MYSQL Workbench, ya que me ha dado problemas el ponerlo en la maquina virtual de Linux introduciendo las sentencias, no he podido visualizarlo en mysql normal de Ubuntu porque me daba errores de sintaxis, pero en el Workbench si me daba resultados que creo pueden ser los solicitados en los eje