/\* SCRIPT 1

Crea un procedimiento almacenado que muestre la id y circunscripción

de las id\_digital que han sido compradas por un partido determinado.

El partido se recibe como parámetro y se almacena el número de personas

que han sido compradas por el partido mediante otro parámetro. \*/

DELIMITER //

DROP PROCEDURE IF EXISTS compras\_partido //

CREATE PROCEDURE compras\_partido (IN pi\_partido VARCHAR(4), OUT po\_num\_pers INT)

BEGIN

IF NOT EXISTS (SELECT 1 FROM senador WHERE partido = pi\_partido) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Parámetro inesperado\n\*\*\*\*\*\n====> Siglas no válidas.\n\*\*\*\*\*\n';

END IF;

SELECT id, circunscripcion

FROM id\_digital, senador

WHERE senador\_corrupto = dni AND partido = pi\_partido

AND comprado = TRUE

ORDER BY circunscripcion ASC;

/\* Esta función devuelve el número de filas afectadas por la última consulta,

pero ha sido descontinuada en MySQL 8.0.17. por lo que no se recomienda \*/

-- SET po\_num\_pers = FOUND\_ROWS();

SELECT COUNT(\*) INTO po\_num\_pers

FROM id\_digital, senador

WHERE senador\_corrupto = dni AND partido = pi\_partido

AND comprado = TRUE;

END //

DELIMITER ;

-- Comprobación

CALL compras\_partido('DAW', @num\_pers); -- 6 votos comprados

SELECT @num\_pers;

CALL compras\_partido('NO', @num\_pers); -- Error

/\* SCRIPT 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. \*/

DELIMITER //

DROP FUNCTION IF EXISTS misma\_circ//

CREATE FUNCTION misma\_circ (pi\_id VARCHAR(10), pi\_dni VARCHAR(10))

RETURNS BOOLEAN

NOT DETERMINISTIC

READS SQL DATA

BEGIN

DECLARE v\_circ\_id VARCHAR(25);

DECLARE v\_circ\_dni VARCHAR(25);

SELECT circ\_presenta INTO v\_circ\_dni

FROM senador

WHERE dni = pi\_dni;

SELECT circunscripcion INTO v\_circ\_id

FROM id\_digital

WHERE id = pi\_id;

IF v\_circ\_dni IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Parámetro inesperado\n\*\*\*\*\*\n====> Senador no válido.\n\*\*\*\*\*\n';

END IF;

IF v\_circ\_id IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Parámetro inesperado\n\*\*\*\*\*\n====> ID no válida.\n\*\*\*\*\*\n';

END IF;

IF v\_circ\_id = v\_circ\_dni THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

END //

DELIMITER ;

-- Comprobación

SELECT misma\_circ('I1J2K3L4M5', '56789012E') AS circ\_ok; -- 1

SELECT misma\_circ('I1J2K3L4M5', '67890123F') AS circ\_ok; -- 0

SELECT misma\_circ('AA', '56789012E'); -- ID no válida

SELECT misma\_circ('I1J2K3L4M5', 'AA'); -- Senador no válido

/\* 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 \*/

DELIMITER //

DROP FUNCTION IF EXISTS votos\_no\_validos//

CREATE FUNCTION votos\_no\_validos ()

RETURNS INT

NOT DETERMINISTIC

READS SQL DATA

BEGIN

DECLARE v\_votos INT;

SELECT COUNT(\*) INTO v\_votos FROM vota\_sen;

RETURN v\_votos - (SELECT SUM( misma\_circ(id\_voto, dni\_senador) ) FROM vota\_sen);

-- Otra opción más directa;

-- RETURN SELECT COUNT(\*) FROM vota\_sen WHERE NOT misma\_circ(id\_voto, dni\_senador);

END //

-- Comprobación

SELECT votos\_no\_validos() AS votos\_no\_validos; -- 0, todos son válidos.

-- Añadimos uno incorrecto para acabar la comprobación

START TRANSACTION;

INSERT INTO vota\_sen VALUES ('I1J2K3L4M5', '67890123F');

SELECT votos\_no\_validos() AS votos\_no\_validos; -- 1

ROLLBACK;

/\* SCRIPT 3

Crea los triggers necesarios para asegurar una participación (1,n) entre el centro y el

interventor, es decir, que todos los centros tienen al menos un interventor asociado.

Ignora las inserciones por motivos de simplicidad de gestión. \*/

/\* Sólo se necesitan comprobar las operaciones sobre los interventores, puesto que la clave foránea ya gestiona borrados y actualizaciones en cascada si cambia la tabla de centros \*/

DELIMITER //

-- Borrar interventores de un centro

DROP TRIGGER IF EXISTS int\_cen\_del//

CREATE TRIGGER int\_cen\_del

AFTER DELETE ON interventor FOR EACH ROW

BEGIN

IF NOT EXISTS (SELECT 1 FROM interventor WHERE centro\_asignado = OLD.centro\_asignado) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = '\n\*\*\*\*\*\nNo se puede eliminar el último interventor de un centro.\n\*\*\*\*\*\n';

END IF;

END //

-- Comprobación

START TRANSACTION//

DELETE FROM interventor WHERE dni = '78901234G'//-- No lo permite, es el único

DELETE FROM interventor WHERE dni = '55667788O'// -- Lo Permite, hay varios

ROLLBACK;

-- Actualizar centro de un interventor

DROP TRIGGER IF EXISTS int\_cen\_upd//

CREATE TRIGGER int\_cen\_upd

AFTER UPDATE ON interventor FOR EACH ROW

BEGIN

IF OLD.centro\_asignado <> NEW.centro\_asignado THEN

IF NOT EXISTS (SELECT 1 FROM interventor WHERE centro\_asignado = OLD.centro\_asignado) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = '\n\*\*\*\*\*\nNo se puede cambiar al único interventor del centro.\n\*\*\*\*\*\n';

END IF;

END IF;

END //

-- Comprobación

START TRANSACTION//

UPDATE interventor SET centro\_asignado = 'C006' WHERE dni = '78901234G'//-- No lo permite

UPDATE interventor SET centro\_asignado = 'C006' WHERE dni = '55667788O'// -- Lo Permite

ROLLBACK//

DELIMITER ;

/\* SCRIPT 4

Crea los triggers necesarios para convertir senador.num\_votos en un campo calculado que se actualiza automáticamente cada vez que se modifica la tabla vota\_sen. Ten en cuenta todos los casos posibles. \*/

DELIMITER //

CREATE TRIGGER vota\_sen\_ins

AFTER INSERT ON vota\_sen FOR EACH ROW

BEGIN

UPDATE senador

SET num\_votos = (SELECT COUNT(\*) FROM vota\_sen WHERE dni\_senador = NEW.dni\_senador)

WHERE dni = NEW.dni\_senador;

-- Se podría usar SET num\_votos = num\_votos + 1; más rápido pero menos robusto, ya que

-- no arregla los errores que num\_votos pueda tener o cambios manuales del dato.

END //

-- Comprobación, no usamos una transacción esta vez pues aprovechamos para borrar este voto después

SELECT num\_votos FROM senador WHERE dni = '56789012E'// -- 25000, datos originales aleatorios

INSERT INTO vota\_sen VALUES ('I1J2K3L4M5', '56789012E')//

SELECT num\_votos FROM senador WHERE dni = '56789012E'// -- 4, se corresponde con vota\_sen y este cambio será permanente.

DROP TRIGGER IF EXISTS vota\_sen\_del//

CREATE TRIGGER vota\_sen\_del

AFTER DELETE ON vota\_sen FOR EACH ROW

BEGIN

UPDATE senador

SET num\_votos = (SELECT COUNT(\*) FROM vota\_sen WHERE dni\_senador = OLD.dni\_senador)

WHERE dni = OLD.dni\_senador;

END //

-- Comprobación

DELETE FROM vota\_sen WHERE id\_voto = 'I1J2K3L4M5' AND dni\_senador = '56789012E'//

SELECT num\_votos FROM senador WHERE dni = '56789012E'// -- 3, se ha actualizado correctamente

DROP TRIGGER IF EXISTS vota\_sen\_upd//

CREATE TRIGGER vota\_sen\_upd

AFTER UPDATE ON vota\_sen FOR EACH ROW

BEGIN

-- Que cambie la id de un voto no importa, si cambia el senador votado, se actualiza

IF OLD.dni\_senador <> NEW.dni\_senador THEN

-- Actualizar al senador que pierde el voto

UPDATE senador

SET num\_votos = (SELECT COUNT(\*) FROM vota\_sen WHERE dni\_senador = OLD.dni\_senador)

WHERE dni = OLD.dni\_senador;

-- Actualizar al senador que gana el voto

UPDATE senador

SET num\_votos = (SELECT COUNT(\*) FROM vota\_sen WHERE dni\_senador = NEW.dni\_senador)

WHERE dni = NEW.dni\_senador;

END IF;

END //

/\* Comprobación

'I1J2K3L4M5' ha votado a 12345678A, cambiamos su voto al senador '56789012E' \*/

START TRANSACTION//

UPDATE vota\_sen SET dni\_senador = '56789012E'

WHERE id\_voto = 'I1J2K3L4M5' AND dni\_senador = '12345678A'//

SELECT num\_votos FROM senador WHERE dni = '12345678A'// -- 6, tenía 7

SELECT num\_votos FROM senador WHERE dni = '56789012E'// -- 4, tenía 3

ROLLBACK//

DELIMITER ;

-- SCRIPT 5

/\* Crea los triggers necesarios para asegurar que los votos en vota\_sen cumplan la restricción de que las circunscripciones de senador (circ\_presenta) y votante sean iguales. Ten en cuenta todos los casos posibles. \*/

/\* Se deben controlar 4 circunstancias: inserción y actualización de votos, y que la circunscripción de senadores o id digitales cambien, haciendo inválidos votos existentes. \*/

DELIMITER //

DROP TRIGGER IF EXISTS vota\_sen\_ins2//

CREATE TRIGGER vota\_sen\_ins2

BEFORE INSERT ON vota\_sen FOR EACH ROW

BEGIN

IF NOT misma\_circ(NEW.id\_voto, NEW.dni\_senador) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = '\n\*\*\*\*\*\nLa circunscripción del voto no coincide con la del senador.\n\*\*\*\*\*\n';

END IF;

END //

DROP TRIGGER IF EXISTS vota\_sen\_upd2//

CREATE TRIGGER vota\_sen\_upd2

BEFORE UPDATE ON vota\_sen FOR EACH ROW

BEGIN

IF NOT misma\_circ(NEW.id\_voto, NEW.dni\_senador) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = '\n\*\*\*\*\*\nLa circunscripción del voto no coincide con la del senador.\n\*\*\*\*\*\n';

END IF;

END //

-- Comprobación sobre inserciones o actualizaciones

START TRANSACTION//

INSERT INTO vota\_sen VALUES ('I1J2K3L4M5', '56789012E')//-- Correcto

INSERT INTO vota\_sen VALUES ('I1J2K3L4M5', '67890123F')//-- Incorrecto

ROLLBACK//

DROP TRIGGER IF EXISTS senador\_circ\_upd//

CREATE TRIGGER senador\_circ\_upd

BEFORE UPDATE ON senador FOR EACH ROW

BEGIN

IF OLD.circ\_presenta <> NEW.circ\_presenta THEN

-- Se considera que un senador no puede cambiar de dni, resulta absurdo.

IF EXISTS (SELECT 1 FROM vota\_sen WHERE dni\_senador = OLD.dni

AND misma\_circ(id\_voto, dni\_senador)) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = '\n\*\*\*\*\*\nEl senador tiene votos en otra circunscripción\n\*\*\*\*\*\n';

END IF;

END IF;

END //

DROP TRIGGER IF EXISTS iddig\_circ\_upd//

CREATE TRIGGER iddig\_circ\_upd

BEFORE UPDATE ON id\_digital FOR EACH ROW

BEGIN

IF OLD.circunscripcion <> NEW.circunscripcion THEN

-- Se considera que las id\_digitales tampoco cambian de id

IF EXISTS (SELECT 1 FROM vota\_sen WHERE id\_voto = OLD.id

AND misma\_circ(id\_voto, dni\_senador)) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = '\n\*\*\*\*\*\nEsta ID ya ha votado en otra circunscripción\n\*\*\*\*\*\n';

END IF;

END IF;

END //

-- Comprobación sobre actualizaciones de circunscripciones

UPDATE senador SET circ\_presenta = 'Ladronia' WHERE dni = '56789012E'//-- Incorrecto

UPDATE id\_digital SET circunscripcion = 'Ladronia' WHERE id = 'I1J2K3L4M5'//-- Incorrecto

-- Borramos los votos del senador y la id para poder cambiar su circunscripción

START TRANSACTION//

DELETE FROM vota\_sen WHERE dni\_senador = '56789012E' or id\_voto='I1J2K3L4M5'//

UPDATE senador SET circ\_presenta = 'Ladronia' WHERE dni = '56789012E'//

UPDATE id\_digital SET circunscripcion = 'Ladronia' WHERE id = 'I1J2K3L4M5'//

ROLLBACK//

DELIMITER ;