Fernández Fernández, Francisco Javier 33413458R

Gracias por empezar cada ejercicio en una nueva cara

[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)
```

Fernández Fernández, Francisco Javier 33413458R

Gracias por empezar cada ejercicio en una nueva cara

[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.

Fernández Fernández, Francisco Javier 33413458R

Gracias por empezar cada ejercicio en una nueva cara

[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';`

Fernández Fernández, Francisco Javier 33413458R

Gracias por empezar cada ejercicio en una nueva cara

[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