Entregable 3 – Bases de Datos

4.1. Script 1

```
-- Inicio del Script 1

DROP PROCEDURE IF EXISTS muestraIdsDigitalesCompradas;
DELIMITER $$
CREATE PROCEDURE muestraIdsDigitalesCompradas (
   IN partidoC VARCHAR(4),
   OUT numComprados INTEGER)
BEGIN
   IF (partidoC NOT IN (SELECT DISTINCT partido FROM senador)) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'NO EXISTE EL PARTIDO';
   END IF;
   SELECT COUNT(*) INTO numComprados
   FROM id_digital i
   INNER JOIN senador s
   ON s.dni = i.senador_corrupto
   WHERE i.comprado = 1 AND partidoC = s.partido;
   SELECT i.id, i.circunscripcion
   FROM id_digital i
   INNER JOIN senador s
   ON s.dni = i.senador_corrupto
   WHERE i.comprado = 1 AND partidoC = s.partido
   ORDER BY i.circunscripcion;
END$$
DELIMITER ;

-- Fin del Script 1
```

4.2. Script 2

```
-- Inicio del Script 2
-- FUNCIÓN 1:

DROP FUNCTION IF EXISTS mismaCircunscripcion;
DELIMITER $$
CREATE FUNCTION mismaCircunscripcion(
   idDig VARCHAR(10),
   dniSen VARCHAR(10))
   RETURNS INTEGER
   DETERMINISTIC
   READS SQL DATA
BEGIN
   IF(dniSen NOT IN (SELECT DISTINCT dni FROM senador) ) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'DNI no corresponde a un Senador';
   END IF;

   IF((SELECT COUNT(*)
      FROM id_digital i
      INNER JOIN senador s
      ON s.circ_presenta = i.circunscripcion
      WHERE idDig = i.id AND s.dni = dniSen) = 1) THEN
      RETURN 1;
   ELSE
      RETURN 0;
   END IF;
END$$
DELIMITER ;

-- FUNCIÓN 2:

DROP FUNCTION IF EXISTS votosNoCumplen;
DELIMITER $$
CREATE FUNCTION votosNoCumplen()
   RETURNS INTEGER
   DETERMINISTIC
   READS SQL DATA
BEGIN
   DECLARE resultado INTEGER;
   SET resultado = (SELECT COUNT(*)
FROM vota_sen v
INNER JOIN senador s
ON s.dni = v.dni_senador
INNER JOIN id_digital i
ON i.id = v.id_voto
WHERE mismaCircunscripcion(i.id,s.dni) = 0);
   RETURN resultado;
END$$
DELIMITER ;

-- Fin Script 2
```

4.3. Script 3

```
-- Inicio del Script 3
-- TRIGGER DELETE:

DROP TRIGGER IF EXISTS borradoDeInterventor;
DELIMITER $$
CREATE TRIGGER borradoDeInterventor
BEFORE DELETE ON interventor
FOR EACH ROW
BEGIN
   IF((SELECT COUNT(*)
      FROM centro c
      WHERE c.id_centro NOT IN
         (SELECT i.centro_asignado
          FROM interventor i
          WHERE i.dni <> OLD.dni)) > 0)
          THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No se puede eliminar este
Interventor. Todo Centro debe tener un Interventor asignado.';
   END IF;
END$$
DELIMITER ;

-- TRIGGER UPDATE:

DROP TRIGGER IF EXISTS actualizadoDeInterventor;
DELIMITER $$
CREATE TRIGGER actualizadoDeInterventor
BEFORE UPDATE ON interventor
FOR EACH ROW
BEGIN
   IF(OLD.centro_asignado <> NEW.centro_asignado)
   AND
   ((SELECT COUNT(*)
      FROM centro c
      WHERE c.id_centro NOT IN
         (SELECT i.centro_asignado
          FROM interventor i
          WHERE i.dni <> OLD.dni)) > 0)
          THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No se puede actualizar este
Interventor. Todo Centro debe tener un Interventor asignado.';
   END IF;
END$$
DELIMITER ;

-- Fin del Script 3
```

4.4. Script 4

```
-- Inicio del Script 4
-- TRIGGER INSERT:

DROP TRIGGER IF EXISTS insertadoNumVotos;
DELIMITER $$
CREATE TRIGGER insertadoNumVotos
AFTER INSERT ON vota_sen
FOR EACH ROW
BEGIN
   DECLARE nuevoNumVotos INTEGER;
   SELECT COUNT(*) INTO nuevoNumVotos
      FROM vota_sen v
      INNER JOIN senador s
      ON s.dni = v.dni_senador
      WHERE NEW.dni_senador = s.dni;
   UPDATE senador
   SET num_votos = nuevoNumVotos
   WHERE dni = NEW.dni_senador;
END$$
DELIMITER ;

-- TRIGGER UPDATE:

DROP TRIGGER IF EXISTS actualizadoNumVotos;
DELIMITER $$
CREATE TRIGGER actualizadoNumVotos
AFTER UPDATE ON vota_sen
FOR EACH ROW
BEGIN
   DECLARE nuevoNumVotos INTEGER;

   -- SUMA EL NUEVO VOTO
   SELECT COUNT(*) INTO nuevoNumVotos
      FROM vota_sen v
      INNER JOIN senador s
      ON s.dni = v.dni_senador
      WHERE NEW.dni_senador = s.dni;
   UPDATE senador
   SET num_votos = nuevoNumVotos
   WHERE dni = NEW.dni_senador;

   -- DESCUENTA EL VOTO ANTERIOR
   SELECT COUNT(*) INTO nuevoNumVotos
      FROM vota_sen v
      INNER JOIN senador s
      ON s.dni = v.dni_senador
      WHERE OLD.dni_senador = s.dni;
   UPDATE senador
   SET num_votos = nuevoNumVotos
   WHERE dni = OLD.dni_senador;
```

```
END$$
DELIMITER ;

-- TRIGGER DELETE:

DROP TRIGGER IF EXISTS borradoNumVotos;
DELIMITER $$
CREATE TRIGGER borradoNumVotos
AFTER DELETE ON vota_sen
FOR EACH ROW
BEGIN
   DECLARE nuevoNumVotos INTEGER;

   -- DESCUENTA EL VOTO BORRADO
   SELECT COUNT(*) INTO nuevoNumVotos
      FROM vota_sen v
      INNER JOIN senador s
      ON s.dni = v.dni_senador
      WHERE OLD.dni_senador = s.dni;
   UPDATE senador
   SET num_votos = nuevoNumVotos
   WHERE dni = OLD.dni_senador;
END$$
DELIMITER ;

-- Fin del Script 4
```

4.5. Script 5

```
-- Inicio del Script 5
-- TRIGGER INSERT:

DROP TRIGGER IF EXISTS insertadoCircIguales;
DELIMITER $$
CREATE TRIGGER insertadoCircIguales
BEFORE INSERT ON vota_sen
FOR EACH ROW
BEGIN
   IF((SELECT DISTINCT circunscripcion
      FROM id_digital
      WHERE NEW.id_voto = id) <>
         (SELECT DISTINCT circ_presenta
         FROM senador
         WHERE NEW.dni_senador = dni)) THEN
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ERROR: ambas circunscripciones
deben ser iguales';
   END IF;
END$$
DELIMITER ;

-- TRIGGER UPDATE(vota_sen):

DROP TRIGGER IF EXISTS actualizadoCircIgualesEnVotaSen;
DELIMITER $$
CREATE TRIGGER actualizadoCircIgualesEnVotaSen
BEFORE UPDATE ON vota_sen
FOR EACH ROW
BEGIN
   IF((SELECT DISTINCT circunscripcion
      FROM id_digital
      WHERE NEW.id_voto = id) <>
         (SELECT DISTINCT circ_presenta
         FROM senador
         WHERE NEW.dni_senador = dni)) THEN
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ERROR: ambas circunscripciones
deben ser iguales';
   END IF;
END$$
DELIMITER ;

-- TRIGGER UPDATE (id_digital):

DROP TRIGGER IF EXISTS actualizadoCircIgualesEnIDDigital;
DELIMITER $$
CREATE TRIGGER actualizadoCircIgualesEnIDDigital
BEFORE UPDATE ON id_digital
FOR EACH ROW
BEGIN
   IF(NEW.circunscripcion <>
```

```
      (SELECT DISTINCT circ_presenta
         FROM senador s
         INNER JOIN vota_sen v
         ON s.dni = v.dni_senador
         AND NEW.id=v.id_voto)
         ) THEN
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ERROR: la circunscripcion de la
ID debe ser igual a la del Senador votado';
   END IF;
END$$
DELIMITER ;

-- TRIGGER UPDATE (senador):

DROP TRIGGER IF EXISTS actualizadoCircIgualesEnSenador;
DELIMITER $$
CREATE TRIGGER actualizadoCircIgualesEnSenador
BEFORE UPDATE ON senador
FOR EACH ROW
BEGIN
   IF(
      ((SELECT COUNT(*)
         FROM vota_sen
         WHERE NEW.dni = dni_senador) > 0)
      AND
      (NEW.circ_presenta NOT IN (SELECT DISTINCT circunscripcion
         FROM id_digital i
         INNER JOIN vota_sen v
         ON i.id = v.id_voto
         WHERE NEW.dni = v.dni_senador))
         ) THEN
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ERROR: la circ_presenta del
Senador debe ser igual a la circunscripcion de las Id que lo han votado';
   END IF;
END$$
DELIMITER ;


-- Fin del Script 5
```