BASES DE DATOS

Hito 2: Creación y explotación de la base de datos ExpressCorreos

GRUPO CITIT21-4

Integrantes:

AKBAROV AKBAROVA FATIMA ZAHRAA
DIAZ CHAVEZ RAQUEL
GARCIA DIEZ CARLOS
SANCHEZ BARCELO IVAN
SANCHEZ MAQUEDA DAMIAN

1. Define las instrucciones SQL para crear todas las tablas necesarias del paso a tablas que se realizó en el Hito 1. Presta especial atención a las configuraciones de integridad referencial para garantizar que se cumplen todas las restricciones planteadas en el enunciado de la práctica.

Creación de la base de datos con el fichero SQL_ExpressCorreos.sql
Llenado de la base de datos ejecutando el contenido del fichero DATA.sql
Asignacion de las foreign_keys con el fichero FOREIGN_KEYS.sql

2. Resuelve las siguientes consultas SQL:
    a. Obtener las oficinas asociadas a areas de envio que incluyan calles cuyo nombre comience por "Paseo de". Además, sólo se deben obtener oficinas que estén dentro del municipio "Alcobendas".

```
SELECT DISTINCT oficina.codigo, oficina.codigo_centro_clasificacion,
oficina.id_municipio FROM oficina
INNER JOIN areaenvio ON areaenvio.id_oficina = oficina.codigo
INNER JOIN area_envio_incluye_segmento_calle ON
area_envio_incluye_segmento_calle.id_area_envio = areaenvio.id_area_envio
INNER JOIN segmentocalle ON segmentocalle.n_Segmento =
area_envio_incluye_segmento_calle.n_Segmento
INNER JOIN calle ON calle.id_calle = segmentocalle.id_calle
WHERE calle.id_municipio = oficina.id_municipio
AND calle.id_calle IN (
        SELECT id_calle FROM calle
    WHERE nombre LIKE 'Paseo de %'
)
AND
oficina.id_municipio IN (
        SELECT id_municipio FROM municipio
    WHERE municipio.nombre = "Alcobendas"
);
```

    b. Obtener el nombre y apellidos del cartero que ha realizado el reparto del paquete con menor peso, así como la matricula del coche con el que hizo dicho reparto, y el peso de dicho paquete.

```
SELECT DISTINCT cartero.nombre, cartero.apellido, reparto.matricula,
paquete.peso FROM cartero
INNER JOIN reparto ON cartero.dni = reparto.dni_cartero
INNER JOIN paquete ON reparto.id_reparto = paquete.id_reparto
WHERE paquete.peso <= ALL ( SELECT paquete.peso FROM paquete);
```

    c. Obtener el nombre, apellidos y dirección completa de los usuarios que han enviado cartas normales y certificadas

(devolver el tipo de carta también) dentro del área de envío "AR-MAD-O3".

```
SELECT ugenerico.nombre, ugenerico.apellidos, direccion.portal,
direccion.numero, direccion.piso, direccion.letra,
calle.nombre,municipio.nombre,provincia.nombre
FROM ugenerico
        INNER JOIN direccion ON direccion.id_direccion =
ugenerico.id_direccion
        INNER JOIN calle ON calle.id_calle = direccion.id_calle
   INNER JOIN municipio ON municipio.id_municipio = calle.id_municipio
   INNER JOIN provincia ON municipio.nombre_provincia = provincia.nombre
   INNER JOIN oficina ON oficina.id_municipio = municipio.id_municipio
   INNER JOIN areaenvio ON oficina.codigo = areaenvio.id_oficina
   WHERE areaenvio.nombre_area_envio = 'AR-MAD-03'
   AND ugenerico.id_ugenerico IN (
                SELECT carta.id_UGenerico_envia FROM carta
        )
UNION
SELECT uidentificado.nombre, uidentificado.apellidos, direccion.portal,
direccion.numero, direccion.piso, direccion.letra,
calle.nombre,municipio.nombre,provincia.nombre
FROM uidentificado
        INNER JOIN uidentificado_vive_en_direccion ON
uidentificado_vive_en_direccion.dni_uidentificado=uidentificado.dni_uidentifi
cado
        INNER JOIN direccion ON direccion.id_direccion =
uidentificado_vive_en_direccion.id_direccion
        INNER JOIN calle ON calle.id_calle = direccion.id_calle
   INNER JOIN municipio ON municipio.id_municipio = calle.id_municipio
   INNER JOIN provincia ON municipio.nombre_provincia = provincia.nombre
   INNER JOIN oficina ON oficina.id_municipio = municipio.id_municipio
   INNER JOIN areaenvio ON oficina.codigo = areaenvio.id_oficina
   WHERE areaenvio.nombre_area_envio = 'AR-MAD-03'
   AND uidentificado.dni_uidentificado IN (
                SELECT cartacertificada.dni_ui_envia FROM cartacertificada
);
```

d. Obtener el numero total de repartos realizados por cada cartero en las diferentes oficinas ordenado de mayor a menor.

```
SELECT dni_cartero, coche.codigo_oficina, COUNT(*)
FROM reparto
INNER JOIN coche ON reparto.matricula = coche.matricula
WHERE codigo_oficina IN(SELECT DISTINCT coche.codigo_oficina
                                FROM coche)
GROUP BY dni_cartero, coche.codigo_oficina
ORDER BY dni_cartero,COUNT(*) DESC;
```

e. Obtener el peso total de los paquetes y el nivel mayor de urgencia de las cartas certificadas que ha llevado los coches de

la oficina “OF-MAD-O1” en los repartos que se les han asignado en los últimos 7 días.

```
/*
Obtener el peso total de los paquetes que ha llevado los coches de la oficina “OF-MAD-O1”
en los repartos que se les han asignado en los últimos 7 días.
*/
SELECT SUM(paquete.peso) FROM reparto
INNER JOIN paquete ON paquete.id_reparto = reparto.id_reparto
INNER JOIN coche ON coche.matricula = reparto.matricula
INNER JOIN oficina ON oficina.codigo = coche.codigo_oficina
WHERE oficina.codigo = "3"
AND fecha_creacion < CURDATE() AND fecha_creacion >=
DATE_SUB(CURDATE(), INTERVAL 7 DAY);
/*
11.5
*/
/*
Obtener el nivel mayor de urgencia de las cartas certificadas que ha llevado los coches de la oficina “OF-MAD-O1”
en los repartos que se les han asignado en los últimos 7 días.
*/
DELIMITER //
CREATE FUNCTION nivelMaximo()
RETURNS VARCHAR(20)
NO SQL
BEGIN
        DECLARE done INT DEFAULT FALSE;
    DECLARE nivelActual VARCHAR(20) DEFAULT '';
        DECLARE nivelMayor VARCHAR(20) DEFAULT '';
    DECLARE cur1 CURSOR FOR (
                SELECT cartacertificada.nivel_urgencia FROM reparto
                INNER JOIN cartacertificada ON cartacertificada.id_reparto =
reparto.id_reparto
                INNER JOIN coche ON coche.matricula = reparto.matricula
                INNER JOIN oficina ON oficina.codigo = coche.codigo_oficina
                WHERE oficina.codigo = "3"
                AND fecha_creacion < CURDATE() AND fecha_creacion >=
DATE_SUB(CURDATE(), INTERVAL 7 DAY)
    );
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
                OPEN cur1;
                read_loop: LOOP
                FETCH cur1 INTO nivelActual;
                IF nivelActual='ALTO' AND nivelMayor<>'ALTO' THEN
                        SET nivelMayor = nivelActual;
                ELSEIF nivelActual='MEDIO' OR nivelMayor<>'ALTO' THEN
                        SET nivelMayor = nivelActual;
                ELSEIF nivelActual='BAJO' AND nivelMayor<>'ALTO' AND
nivelMayor<>'MEDIO' THEN
                        SET nivelMayor = nivelActual;
                END IF;
                IF done THEN
                        LEAVE read_loop;
```

```
                END IF;
                END LOOP;
                CLOSE cur1;
    RETURN (nivelMayor);
END //
DELIMITER ;

SELECT nivelMaximo();

/*
ALTO
*/
```

f. Aumentar en un 10% la capacidad de los todos los coches que únicamente se hayan utilizado para realizar repartos de cartas.

```
UPDATE coche
INNER JOIN reparto ON coche.matricula=reparto.matricula
SET capacidad = capacidad * 1.10
WHERE reparto.matricula NOT IN(
                SELECT reparto.matricula
                FROM reparto
                INNER JOIN paquete ON paquete.id_reparto =
reparto.id_reparto
);
```

g. Obtener los carteros que no hayan llevado a cabo recogidas de paquetes en la misma dirección varias veces.

```
SELECT DISTINCT nombre, apellido
        FROM correosexpress.cartero INNER JOIN correosexpress.recogida
                ON cartero.dni = recogida.dni_cartero
        WHERE dni_cartero NOT IN (
                SELECT dni_cartero
        FROM correosexpress.recogida
        GROUP by  id_direccion, dni_cartero
                HAVING COUNT(*) > 1
);
```

h. Obtener el código y la información del área de envío asociada a las oficinas en la que trabajan carteros con todos los turnos existentes.

```
SELECT areaenvio.id_area_envio, areaenvio.nombre_area_envio
,areaenvio.id_oficina FROM areaenvio
INNER JOIN oficina ON oficina.codigo = areaenvio.id_oficina
```

```
INNER JOIN trabaja ON trabaja.codOf = oficina.codigo
INNER JOIN turno ON turno.jornada = trabaja.jornadaT
GROUP BY oficina.codigo
HAVING COUNT(distinct turno.jornada) = 3;
```

i. Obtener las rutas que incluyen todos los segmentos de la calle "Avenida de América" de "Alcorcón" y que se hayan realizado más de 3 repartos en ella.

```
SELECT ruta.id FROM ruta
INNER JOIN ruta_incluye_segmento ON ruta.id =
ruta_incluye_segmento.id_ruta
INNER JOIN segmentocalle ON segmentocalle.n_Segmento =
ruta_incluye_segmento.n_segmento
INNER JOIN calle ON calle.id_calle = segmentocalle.id_calle
INNER JOIN municipio ON municipio.id_municipio = calle.id_municipio
INNER JOIN reparto ON reparto.id_ruta = ruta.id
WHERE calle.nombre = 'Avenida de América'
AND municipio.nombre = 'Alcorcón'
GROUP BY ruta.id
HAVING COUNT(distinct segmentocalle.n_Segmento) = (
                SELECT COUNT(*) FROM segmentocalle
                INNER JOIN calle ON calle.id_calle = segmentocalle.id_calle
                INNER JOIN municipio ON municipio.id_municipio =
calle.id_municipio
                WHERE calle.nombre = 'Avenida de América'
                AND municipio.nombre = 'Alcorcón'
)
AND COUNT(distinct reparto.id_reparto) > 3
```

3. Implementa el siguiente procedimiento y función:
    a. Define un procedimiento que devuelva los DNIs (separados por "#") tanto del usuario receptor, como del usuario autorizado, para recibir la entrega de una carta certificada que se pasará como parámetro de entrada.

```
DELIMITER &&
CREATE PROCEDURE dni_receptor_autorizado_inner_join(IN id_carta_cert INT)
BEGIN
        SELECT dni_uidentificado, dni_autorizado_por_id_uidentificado FROM
UIdentificado
   INNER JOIN CartaCertificada ON CartaCertificada.dni_ui_recibe =
UIdentificado.dni_uidentificado
   WHERE CartaCertificada.id_carta_certificada = id_carta_cert;
END &&
DELIMITER ;
```

b. Define una función que devuelva cual es la ruta más adecuada de las predefinidas existentes para realizar una reparto especificado por parámetro. Dicha ruta será aquella que incluya el mayor número de segmentos de calles de todas las direcciones de los paquetes y cartas (de ambos tipos) contenidos en el reparto. Se debe utilizar como mínimo un cursor para implementar esta función.

```
DELIMITER %%
CREATE FUNCTION ruta_mejor(reparto INT)
RETURNS INT
DETERMINISTIC
BEGIN
	DECLARE done INT DEFAULT FALSE;
   DECLARE id_mejor_ruta INT default -1;
	DECLARE n_segmentos_mejor_ruta INT;
   DECLARE ruta_id, n_segmentos INT;
   DECLARE cur1 CURSOR FOR (
		/*RUTA Y TOTAL DE SEGMENTOS QUE ESTAN EN LA RUTA Y EN LA DIRECCION DE REPARTO*/
		SELECT Ruta.id, calculaSegmentos(Ruta.id, reparto) as segmentosRutaReparto FROM Ruta
   );
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
	SET n_segmentos_mejor_ruta = 0;
		OPEN cur1;
		read_loop: LOOP
		FETCH cur1 INTO ruta_id, n_segmentos;
		IF n_segmentos>n_segmentos_mejor_ruta THEN
			SET n_segmentos_mejor_ruta = n_segmentos;
			SET id_mejor_ruta = ruta_id;
		END IF;
		IF done THEN
			LEAVE read_loop;
		END IF;
		END LOOP;
		CLOSE cur1;
   RETURN (id_mejor_ruta);
END %%
DELIMITER ;

DELIMITER //
CREATE FUNCTION calculaSegmentos(RUTA_ID INT, REPARTO_ID INT)
RETURNS INT
DETERMINISTIC
BEGIN
	DECLARE total INT;
   SET total = (
		SELECT COUNT(*) FROM (
			SELECT distinct segmentocalle.n_Segmento FROM segmentocalle
				WHERE segmentocalle.n_Segmento IN (
					SELECT distinct segmentocalle.n_Segmento FROM reparto
```

```
                                        INNER JOIN carta ON reparto.id_reparto =
carta.id_Reparto
                                        INNER JOIN ugenerico ON
carta.id_UGenerico_recibe = ugenerico.id_ugenerico
                                        INNER JOIN direccion ON
ugenerico.id_direccion = direccion.id_direccion
                                        INNER JOIN calle ON calle.id_calle =
direccion.id_calle
                                        INNER JOIN segmentocalle ON
segmentocalle.id_calle = calle.id_calle
                                        WHERE direccion.numero BETWEEN
segmentocalle.n_Inicio AND segmentocalle.n_Final AND reparto.id_reparto =
REPARTO_ID
                                        AND segmentocalle.n_Segmento IN (
                                                SELECT distinct
segmentocalle.n_Segmento FROM segmentocalle
                                                INNER JOIN calle ON
segmentocalle.id_calle = calle.id_calle
                                                INNER JOIN direccion ON
direccion.id_calle = calle.id_calle
                                                INNER JOIN ruta_incluye_segmento
ON ruta_incluye_segmento.n_segmento = segmentocalle.n_segmento
                                                WHERE id_ruta = RUTA_ID
                                        )
                                        UNION
                                        SELECT distinct segmentocalle.n_Segmento
FROM reparto
                                        INNER JOIN paquete ON reparto.id_reparto =
paquete.id_reparto
                                        INNER JOIN ugenerico ON
paquete.id_ug_recibe = ugenerico.id_ugenerico
                                        INNER JOIN direccion ON
ugenerico.id_direccion = direccion.id_direccion
                                        INNER JOIN calle ON calle.id_calle =
direccion.id_calle
                                        INNER JOIN segmentocalle ON
segmentocalle.id_calle = calle.id_calle
                                        WHERE direccion.numero BETWEEN
segmentocalle.n_Inicio AND segmentocalle.n_Final AND reparto.id_reparto =
REPARTO_ID
                                        AND segmentocalle.n_Segmento IN (
                                                SELECT distinct
segmentocalle.n_Segmento FROM segmentocalle
                                                INNER JOIN calle ON
segmentocalle.id_calle = calle.id_calle
                                                INNER JOIN direccion ON
direccion.id_calle = calle.id_calle
                                                INNER JOIN ruta_incluye_segmento
ON ruta_incluye_segmento.n_segmento = segmentocalle.n_segmento
                                                WHERE id_ruta = RUTA_ID
                                        )
                                )
        ) segmentos_en_ruta_y_reparto
    );
    RETURN total;
```

```
END //
DELIMITER ;
```

4. Implementa los siguientes triggers:
    a. Define un trigger para impedir que se puedan registrar envíos de paquetes cuyo identificador no cumpla que empiece por “PQ” y tenga 10 números después, que además sus dimensiones estén entre 5x5 y 25x25 y que su peso entre 1 y 100. Si no se cumple alguna de estas restricciones de datos se deber´a devolver un error con un mensaje que indique claramente el error concreto que se ha producido.

```
DELIMITER $$
CREATE TRIGGER formatoPaqueteErroneo before INSERT ON Paquete
FOR EACH ROW
BEGIN
/*
ID_paquete correcta
*/
if char_length(NEW.id_paquete) > 12
OR char_length(NEW.id_paquete) < 12
or NEW.id_paquete not regexp 'PQ+[0-9]+$' then
        signal sqlstate '02000'
        SET message_text = 'ID incorrecto';

/*
dimension del paquete correcta
*/
elseif NEW.dimension not regexp '[0-9]+$+x+[0-9]+$'
/*
Primera dimension
*/

or substr(NEW.dimension, 1,locate('x', NEW.dimension)-1) + 0 < 5
or substr(NEW.dimension, 1,locate('x', NEW.dimension)-1) + 0 > 25
/*
Segunda dimension
*/
or substr(NEW.dimension, locate('x',NEW.dimension)+1, 3) + 0 < 5
or substr(NEW.dimension, locate('x',NEW.dimension)+1, 3) + 0 > 25 then
        signal sqlstate '02000'
        SET message_text = 'Dimensiones del paquete incorrecta';

/*
peso del paquete correcto
*/
elseif NEW.PESO > 100 then
        signal sqlstate '02000'
        SET message_text = 'Paquete demasiado pesado';
elseif NEW.PESO < 1 then
        signal sqlstate '02000'
```

```
        SET message_text = 'Paquete demasiado ligero';
end if;
END$$

DELIMITER ;
```

b. Define un trigger que compruebe que si se han incluido paquetes en un reparto con un peso mayor de 500Kg en total se le asigne un cartero. Como el reparto al crearse tiene que tener asignado un coche, el cartero que se le deberá asignar tiene que ser uno que este trabajando en la misma oficina que el coche que tiene asignado previamente el reparto, en un turno que este dentro del horario actual, y que no tenga asignado otro reparto en ese momento. Se puede utilizar la función CURRENT TIMESTAMP1 para obtener la fecha.

```
DROP TRIGGER IF EXISTS comprobarReparto;
DELIMITER //
CREATE TRIGGER comprobarReparto after INSERT ON Reparto
FOR EACH ROW
BEGIN
declare dnicartero varchar(9) default null;
DECLARE done INT DEFAULT FALSE;
DECLARE repartoID, pesoTotal INT;
DECLARE cur1 CURSOR FOR (
                select Reparto.id_reparto, sum(peso) as pesoReparto
                from Reparto join Paquete on Reparto.id_reparto =
Paquete.id_reparto
                group by Reparto.id_reparto
);
DECLARE cartero_DNI VARCHAR(20);
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
SET cartero_DNI = seleccionarCarteroDisponible();
        OPEN cur1;
   read_loop: LOOP
                FETCH cur1 INTO repartoID, pesoTotal;
      IF pesoTotal>500 THEN
                        UPDATE reparto SET dni_cartero = cartero_DNI
         WHERE id_reparto = repartoID;
      END IF;
      IF done THEN
                        LEAVE read_loop;
                END IF;
        END LOOP;
   CLOSE cur1;
END //
DELIMITER ;

DELIMITER &&
CREATE FUNCTION seleccionarCarteroDisponible()
RETURNS VARCHAR(20)
DETERMINISTIC
```

```
BEGIN
	DECLARE dni VARCHAR(20) DEFAULT '';
   SET dni = ( SELECT cartero.dni IN (
                        SELECT cartero.dni FROM cartero
                        INNER JOIN coche ON coche.matricula =
reparto.matricula
                        INNER JOIN oficina ON oficina.codigo =
coche.codigo_oficina
                        INNER JOIN reparto ON cartero.dni = reparto.dni_cartero
                        INNER JOIN trabaja ON trabaja.dniC = cartero.dni
                        INNER JOIN turno ON turno.jornada = trabaja.jornadaT
                        WHERE reparto.fecha_creacion = current_date()
                        AND current_time() > time(substr(horario, 1,
locate('-',horario)-1)) and current_time() < time(substr(horario,
locate('-',horario)+1, 2))
                        UNION
                        SELECT cartero.dni FROM cartero
                        INNER JOIN coche ON coche.matricula =
reparto.matricula
                        INNER JOIN oficina ON oficina.codigo =
coche.codigo_oficina
                        INNER JOIN reparto ON cartero.dni = reparto.dni_cartero
                        INNER JOIN trabaja ON trabaja.dniC = cartero.dni
                        INNER JOIN turno ON turno.jornada = trabaja.jornadaT
                        AND current_time() > time(substr(horario, 1,
locate('-',horario)-1)) and current_time() < time(substr(horario,
locate('-',horario)+1, 2))
                        GROUP BY cartero.dni
                        HAVING COUNT(distinct reparto.id_reparto) = 0
   )
   LIMIT 1
   );
	RETURN(dni);
END &&
DELIMITER ;
```

Código SQL en Repositorio privado en github:
https://github.com/aprentix/ExpressCorreos_DB