Procedimientos almacenados y disparadores, ¿para qué son necesarios?

NOMBRE Y APELLIDOS: GLORIA MARÍA MANRESA SANTAMARÍA

EJERCICIO 1 (35%)

APARTADO A

Se modifica la tabla para insertar como valor por defecto la fecha y hora de la inserción de la fila. De esta manera cada vez que se inserte una nueva fila de datos se rellenará automáticamente este campo.

```
1  -- EJERCICIO 1A
2  BEGIN WORK;
3
4  ALTER TABLE erp.tb_pump
5      ADD COLUMN updated_dt_tm timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP;
6
7  COMMIT WORK;
8
9  SELECT *
10 FROM erp.tb_pump
```

Data Output | Messages | Notifications

	pm_id [PK] integer	pm_parent_id integer	gas_id character (5)	pm_descr character varying (20)	updated_dt_tm timestamp without time zone
1	1	[null]	GS01	GS01	2022-12-10 11:44:45.883894
2	2	[null]	GS02	GS02	2022-12-10 11:44:45.883894
3	3	[null]	GS03	GS03	2022-12-10 11:44:45.883894
4	4	[null]	GS04	GS04	2022-12-10 11:44:45.883894
5	5	[null]	GS05	GS05	2022-12-10 11:44:45.883894
6	6	1	GS01	Surtidor Nº1	2022-12-10 11:44:45.883894
7	7	1	GS01	Surtidor Nº2	2022-12-10 11:44:45.883894
8	8	1	GS01	Surtidor Nº3	2022-12-10 11:44:45.883894
9	9	1	GS01	Surtidor Nº4	2022-12-10 11:44:45.883894
10	10	2	GS02	Surtidor Nº1	2022-12-10 11:44:45.883894
11	11	2	GS02	Surtidor Nº2	2022-12-10 11:44:45.883894

APARTADO B

```sql
-- EJERCICIO 1B

-- Añadir nuevo campo
ALTER TABLE erp.tb_lines_invoice
ADD COLUMN line_updated_dt_tm varchar(20);


-- Crear función
CREATE OR REPLACE FUNCTION fn_line_inserted()
RETURNS TRIGGER AS $$
BEGIN
NEW.line_updated_dt_tm = TO_CHAR(NOW(), 'yyyy-mm-dd HH24:MI');
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Crear disparador
CREATE TRIGGER tg_line_inserted
BEFORE UPDATE OR INSERT
ON erp.tb_lines_invoice
```

Para rellenar la fila de datos se han borrado los datos de la tabla y vuelto a insertar una vez creado el trigger.

Se ha comprobado que el trigger también funciona al modificar los datos.

APARTADO C

```sql
-- EJERCICIO 1C

-- CREAR TRES NUEVOS ATRIBUTOS SEGÚN EL ENUNCIADO

ALTER TABLE erp.tb_invoice
    ADD COLUMN inv_updated_dt DATE,
    ADD COLUMN inv_update_counter int DEFAULT 0,
    ADD COLUMN inv_insert_counter int DEFAULT 0;

-- CREAR FUNCIÓN
CREATE OR REPLACE FUNCTION fn_invoice_updated()
RETURNS TRIGGER AS $$
BEGIN

if TG_OP='INSERT' then
UPDATE erp.tb_invoice
SET inv_updated_dt = current_date,
    inv_insert_counter = (SELECT count(inv_id)
                          FROM erp.tb_lines_invoice
                          WHERE inv_id = new.inv_id
                         GROUP BY inv_id
                          )
WHERE inv_id = new.inv_id;
end if;

if TG_OP='UPDATE' then
UPDATE erp.tb_invoice
SET     inv_updated_dt = current_date,
        inv_update_counter = inv_update_counter+1
WHERE inv_id = old.inv_id;
end if;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Crear disparador
CREATE TRIGGER tg_invoice_updated
AFTER INSERT OR UPDATE ON erp.tb_lines_invoice
FOR EACH ROW EXECUTE PROCEDURE fn_invoice_updated();

```

La tabla tb_invoice después de crear el procedimiento anterior queda de la siguiente manera:

```
1  SELECT *
2  FROM erp.tb_invoice
```

Data Output | Messages | Notifications

	inv_id [PK] integer	inv_num character (5)	inv_date_start date	inv_date_end date	inv_amount numeric (12,2)	inv_liters_total integer	inv_updated_dt date	inv_update_counter integer	inv_insert_counter integer
1	1	1001	2022-08-01	2022-08-31	5099.52	2533	2022-12-18	0	42
2	2	1002	2022-09-01	2022-09-30	3847.98	1782	2022-12-18	0	31
3	3	1003	2022-10-01	2022-10-31	5498.16	2721	2022-12-18	0	31

EJERCICIO 2 (55%)

APARTADO A

```
-- EJERCICIO 2A
-- CREAR NUEVA COLUMNA
ALTER TABLE erp.tb_refueling
ADD COLUMN rf_cost NUMERIC;

-- CREAR FUNCIÓN
CREATE OR REPLACE FUNCTION fn_calc_cost()
RETURNS TRIGGER AS $$
BEGIN

UPDATE erp.tb_refueling
SET rf_cost = (SELECT (rf_liters*fp_import)
                FROM (erp.tb_refueling NATURAL JOIN erp.tb_cars)
                        LEFT JOIN erp.tb_fuel_price ON (rf_date=fp_date AND cars_fuel=fp_fuel)
                WHERE cars_registration = new.cars_registration AND
                rf_liters = new.rf_liters AND
                rf_date = new.rf_date
              )
WHERE cars_registration = new.cars_registration AND
        rf_liters = new.rf_liters AND
        rf_date = new.rf_date;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
-- Crear disparador
CREATE TRIGGER tg_refueling_cost
AFTER INSERT ON erp.tb_refueling
FOR EACH ROW EXECUTE PROCEDURE fn_calc_cost();
```

Para que se actualicen los datos de la columna rf_cost, se han borrado los datos de la tabla y se han introducido de nuevo.

La tabla queda de la siguiente manera:

```
1  SELECT *
2  FROM erp.tb_refueling
3
```

Data Output | Messages | Notifications

	gas_id character (5)	cars_registration character (7)	pm_id integer	rf_liters integer	rf_date date	rf_km integer	rf_cost numeric
1	GS01	3685HDP	29	43	2022-09-19	303075	85.828
2	GS01	3685HDP	27	33	2022-09-30	[null]	65.637
3	GS02	3685HDP	33	33	2022-08-15	304275	62.667
4	GS02	3685HDP	35	30	2022-08-30	304825	60.660
5	GS02	3685HDP	35	20	2022-09-25	305392	38.360
6	GS03	3685HDP	37	42	2022-09-04	306108	82.194
7	GS04	3685HDP	43	33	2022-09-09	306442	63.558
8	GS05	3685HDP	55	34	2022-09-14	306992	69.054

APARTADO B

```
-- EJERCICIO 2B
-- CREAR FUNCIÓN
CREATE OR REPLACE FUNCTION fn_get_cost_by_car_type(initial_date DATE,
                                                   final_date DATE,
                                                   car_function CHARACTER(20))
    RETURNS TABLE (
        car_function_selected CHARACTER(20),
        range_date DATE,
        rf_liters_selected INT,
        rf_cost_selected NUMERIC
    )
    AS $$
    BEGIN
        RETURN QUERY
                SELECT cars_function, rf_date, rf_liters,rf_cost
                FROM erp.tb_cars NATURAL JOIN erp.tb_refueling
                WHERE cars_function=car_function AND rf_date BETWEEN initial_date AND final_date;


IF LENGTH(car_function)>20 THEN
RAISE EXCEPTION 'Longitud excesiva del valor introducido para el tipo de vehículo.';
END IF;

END;
$$LANGUAGE plpgsql;
```

Comprobamos el funcionamiento de la función:

```
SELECT *
FROM fn_get_cost_by_car_type('2022-08-18','2022-08-28','comercial')
```

Data Output | Messages | Notifications

	car_function_selected character	range_date date	rf_liters_selected integer	rf_cost_selected numeric
1	comercial	2022-08-19	16	32.944
2	comercial	2022-08-24	22	46.200
3	comercial	2022-08-27	21	43.281
4	comercial	2022-08-23	31	65.503
5	comercial	2022-08-23	28	61.964
6	comercial	2022-08-26	23	50.002
7	comercial	2022-08-20	30	65.970

APARTADO C

```
-- EJERCICIO 2C

CREATE TYPE tb_invoice_cost_summary AS(
    cars_registration   CHARACTER(10),
    cars_fuel           CHARACTER(10),
    invoice_year        INT,
    invoice_quarter     INT,
    invoice_month       INT,
    total_liters        INT,
    total_cost          NUMERIC(10,2),
    number_of_lines     INT,
    date_time           CHARACTER(20)

);


CREATE OR REPLACE FUNCTION invoice_cost_summary()
RETURNS SETOF tb_invoice_cost_summary AS $$
DECLARE
car tb_invoice_cost_summary;
BEGIN
FOR car IN
            SELECT  cars_registration,
                    cars_fuel,
                    EXTRACT(YEAR FROM inv_date_start),
                    EXTRACT(QUARTER FROM inv_date_start),
                    EXTRACT(MONTH FROM inv_date_start),
                    SUM(linv_liters),
                    SUM(linv_amount),
                    COUNT(cars_registration),
                    line_updated_dt_tm
            FROM    erp.tb_lines_invoice
                    NATURAL JOIN erp.tb_cars
                    NATURAL JOIN erp.tb_invoice
            GROUP BY    cars_registration,
                        cars_fuel,
                        EXTRACT(YEAR FROM inv_date_start),
                        EXTRACT(QUARTER FROM inv_date_start),
                        EXTRACT(MONTH FROM inv_date_start),
                        line_updated_dt_tm


              LOOP
RETURN NEXT car;
END LOOP;
RETURN;
END;
$$LANGUAGE plpgsql;
```

Comprobamos el funcionamiento de la función:

```
1  SELECT *
2  FROM invoice_cost_summary()
```

Data Output | Messages | Notifications

	cars_registration character (10)	cars_fuel character (10)	invoice_year integer	invoice_quarter integer	invoice_month integer	total_liters integer	total_cost numeric (10,2)	number_of_lines integer	date_time character (20)
1	0019GVM	gasoil	2022	3	8	301	601.43	5	2022-12-18 18:13
2	0815GYR	gasoil	2022	3	9	308	615.42	5	2022-12-18 18:13
3	6392KPT	gasoil	2022	3	9	166	331.83	4	2022-12-18 18:13
4	4273GFK	gasoil	2022	4	10	478	964.19	5	2022-12-18 18:13
5	3685HDP	gasoil	2022	3	9	188	373.93	2	2022-12-18 18:13
6	8806KZN	gasolina	2022	3	8	108	235.58	3	2022-12-18 18:13
7	3685HDP	gasoil	2022	3	8	188	373.93	2	2022-12-18 18:13
8	2093GSW	gasoil	2022	3	9	394	787.19	5	2022-12-18 18:13
9	0815GYR	gasoil	2022	3	8	308	615.42	5	2022-12-18 18:13
10	2093GSW	gasoil	2022	4	10	394	787.19	5	2022-12-18 18:13

EJERCICIO 3 (10%)

APARTADO A

```
1  -- EJERCICIO 3A
2
3  SELECT  event_object_table AS table_name ,trigger_name
4  FROM information_schema.triggers
5  GROUP BY table_name , trigger_name
6
```

Data Output | Messages | Notifications

	table_name name	trigger_name name
1	tb_lines_invoice	tg_invoice_updated
2	tb_lines_invoice	tg_line_inserted
3	tb_refueling	tg_refueling_cost

En la captura anterior podemos observar que tenemos 3 triggers en la base de datos. También podemos ver a que tabla hace referencia cada uno de los triggers.

APARTADO B

Statistics collector es un subsistema de PostgreSQL que recopila información sobre la actividad del servidor y nos aporta información sobre como está trabajando PostgreSQL.

Podemos, por ejemplo, ver una tabla con todos los usuarios y roles de la base de datos (pg_roles), obtener información sobre los clientes conectados a la base de datos (pg_stat_activity) o obtener información de todas las tablas de usuario, entre otras posibilidades.

Query | Query History | Scratch

```
1  SELECT *
2  FROM pg_stat_user_tables
```

Data Output | Messages | Notifications

	relid oid	schemaname name	relname name	seq_scan bigint	seq_tup_read bigint	idx_scan bigint	idx_tup_fetch bigint	n_tup_ins bigint	n_tup_upd bigint	n_tup_del bigint	n b
1	17236	erp	tb_fuel_price	141	7093	0	0	122	0	0	
2	17272	erp	tb_invoice	75	222	17584	17520	3	117	0	
3	17277	erp	tb_lines_invoice	5300	548944	25	128	225	527879	104	
4	17258	erp	tb_refueling	383	28777	[null]	[null]	243	458	118	
5	17241	erp	tb_pump	264	9173	164	164	56	0	0	
6	17222	erp	tb_cars	95	940	16342	16342	10	0	0	
7	17229	erp	tb_gas_station	2	0	15915	15915	5	0	0	

También se puede obtener información sobre el tamaño de la base de datos. En la siguiente imagen podemos observar la consulta para obtener dicha información, en bytes.

```
1  select pg_database_size('dbdw_pec3');
2
```

Data Output | Messages | Notifications

	pg_database_size bigint
1	12157807

Criterios de valoración

En el enunciado se indica el peso/valoración de cada ejercicio.

Para conseguir la puntuación máxima en los ejercicios, es necesario explicar con claridad la solución que se propone.

Formato y fecha de entrega

Tenéis que enviar la PEC al buzón de Entrega y registro de EC disponible en el aula (apartado Evaluación). El formato del archivo que contiene vuestra solución puede ser **.pdf, .doc y .docx**. **Para otras opciones, por favor, contactar previamente con vuestro consultor**. El nombre del fichero debe contener el código de la asignatura, vuestro apellido y vuestro nombre, así como el número de actividad (PEC3).

La fecha límite para entregar la PEC3 es el **19/12/2022**.

Nota: **Propiedad intelectual**

Al presentar una práctica o PEC que haga uso de recursos ajenos, se tiene que presentar junto con ella un documento en que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL etc.). El estudiante tendrá que asegurarse que la licencia que sea no impide específicamente su uso en el marco de la práctica o PEC. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por el copyright.

Será necesario, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente, si así corresponde.