Uso de bases de datos

Práctica 2: El lenguaje SQL II

En la primera parte de la práctica hemos realizado consultas y modificaciones sobre los datos utilizando SQL. En esta segunda parte, sobre la misma base de datos, añadiremos lógica dentro de la base de datos utilizando procedimientos almacenados y disparadores.

Para la correcta ejecución de la segunda parte de la práctica, es necesario volver a crear la base de datos de nuevo e insertar otra vez los datos iniciales utilizando los *scripts* que se proporcionan junto a este enunciado. Este paso es necesario debido a la introducción de los cambios siguientes en el esquema de la BD respecto a la primera parte de la práctica:

- Creación de una nueva tabla llamada *REPORT_BAND*, con las columnas *id_band, num_instruments*, *num_members_alive, longest_album_title,* y *num_short_songs.* La columna *id_band* corresponde al identificador del grupo y es clave primaria. El resto de columnas se calculan en relación al grupo. Así pues, *num_instruments* es el número **de instrumentos diferentes**, sin incluir los vocalistas, que participan en el grupo, *num_members_alive* es el número de miembros del grupo **que todavía están vivos**, *longest_album_title* es el nombre del álbum que tiene una duración mayor, y por último *num_short_songs* es el número de canciones de menos de 3 minutos que tiene el grupo. Para este ejercicio tendremos en cuenta que todo álbum tiene al menos una canción.
- Tabla *ALBUM*: se ha añadido el atributo *num_long_title_songs* que representa el **número de canciones del álbum que tienen un título con una longitud superior a 12 caracteres**.

Encontraréis las sentencias de creación y modificación de estas tablas en el archivo *create_db.sql.*

Nota importante: El SQL implementado en PostgreSQL puede aceptar diferentes variantes de sintaxis, que además pueden diferir según la versión que instaléis, y que pueden ser o no SQL estándar. Evitad (excepto que se indique lo contrario) utilizar sentencias de este tipo, y concentraros en las que se explican en los módulos didácticos. Si usáis sentencias SQL estándar vuestro código funcionará en cualquier SGBD.

Pregunta 1 (50 % puntuación)

Enunciado

Se pide que creéis un procedimiento almacenado que, dado un nombre de grupo, actualice los datos que le hacen referencia y los almacene en la tabla REPORT_BAND creada con la ejecución del archivo *create_db.sql*, **el cual deberéis ejecutar en primer término**.

Si ya existe una entrada en la tabla para el grupo (*id_band*), ésta se tendrá que modificar con los nuevos valores. Además de guardarlo en la tabla, el procedimiento devolverá el resultado del *report*.

Se tendrá que informar al usuario con un mensaje específico cuando *no exista ningún grupo con el nombre especificado*.

La firma del procedimiento pedido y el tipo que devolverá son los siguientes:

```
CREATE OR REPLACE FUNCTION update_report_band(p_name VARCHAR(255))
RETURNS SETOF report_band_type
```

donde report_band_type es de tipo:

```
CREATE TYPE report_band_type AS (
      t_id_band INTEGER,
      t_num_instruments INTEGER,
      t_num_members_alive INTEGER,
      t_longest_album_title VARCHAR(255),
      t_num_short_songs INTEGER
);
```

Criterios de evaluación

- Las propuestas de solución que no se puedan ejecutar, es decir las que den error de sintaxis, no serán evaluadas.
- Se valorará positivamente el uso de sentencias SQL estándar, al margen de otros elementos que se puedan indicar en el enunciado.
- Para obtener la máxima nota, el código SQL de vuestra solución tiene que ser eficiente. Por ejemplo, se valorará negativamente realizar más *joins* de los necesarios.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir pruebas que cubran todas las posibles situaciones descritas en el enunciado. Por ejemplo, se deberían cubrir todas las posibles situaciones de error.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir los resultados, mediante el uso de capturas de pantalla o cualquier otro mecanismo similar.

Solución

```
SET search_path TO ubd_20211;

CREATE OR REPLACE FUNCTION update_report_band(p_name VARCHAR(255))
RETURNS REPORT_BAND_TYPE AS $$
DECLARE
	var_return_data REPORT_BAND_TYPE;
BEGIN
	IF((SELECT COUNT(*)
		FROM band AS b
		WHERE b.name = p_name) = 0)
	THEN
	RAISE EXCEPTION 'ERROR: no band found for name "%" ', p_name;
	END IF;

	SELECT id_band
	INTO var_return_data.t_id_band
	FROM band
	WHERE "name" = p_name;

	SELECT COUNT(DISTINCT me.instrument)
		INTO var_return_data.t_num_instruments
	FROM "member" AS me
	WHERE me.instrument <> 'Vocals'
	AND me.id_band = var_return_data.t_id_band;

	SELECT COUNT(DISTINCT me.id_musician)
		INTO var_return_data.t_num_members_alive
	FROM "member" AS me
	INNER JOIN musician AS mu
	ON me.id_musician = mu.id_musician
	WHERE mu.death IS NULL
	AND me.id_band = var_return_data.t_id_band;

	SELECT a.title
		INTO var_return_data.t_longest_album_title
	FROM album AS a
	LEFT JOIN song AS s
	ON s.id_album = a.id_album
	WHERE a.id_band = var_return_data.t_id_band
	GROUP BY a.id_album
	ORDER BY SUM(s.duration) DESC
	LIMIT 1;

	SELECT COUNT(s.id_song)
		INTO var_return_data.t_num_short_songs
	FROM song AS s
	INNER JOIN album AS a
	ON s.id_album = a.id_album
	WHERE s.duration < '0:03:00'
	AND a.id_band = var_return_data.t_id_band;
```

```
	IF(NOT EXISTS(
		SELECT id_band
		FROM report_band
		WHERE id_band = var_return_data.t_id_band))
	THEN
	INSERT INTO report_band
		VALUES (
		var_return_data.t_id_band,
		var_return_data.t_num_instruments,
		var_return_data.t_num_members_alive,
		var_return_data.t_longest_album_title,
		var_return_data.t_num_short_songs);
	ELSE
	UPDATE report_band
		SET num_instruments = var_return_data.t_num_instruments,
			num_members_alive = var_return_data.t_num_members_alive,
			longest_album_title = var_return_data.t_longest_album_title,
			num_short_songs = var_return_data.t_num_short_songs
		WHERE id_band = var_return_data.t_id_band;
	END IF;
	RETURN var_return_data;
END;

$$LANGUAGE plpgsql;
```

Con las pruebas siguientes validamos el comportamiento en las diferentes casuísticas, partiendo del archivo *inserts_db.sql* que os damos de ejemplo, el cual **deberéis ejecutar antes de realizar cualquiera de ellas**

Primero comprobamos el control del error. En este caso introduciremos un nombre de grupo que no exista en la base de datos.

```
SELECT * FROM update_report_band('NOFX');
```

Y el resultado debe ser:

```
SQL Error [P0001]: ERROR: ERROR: no band found for name "NOFX"
  Where: PL/pgSQL function update_report_band(character varying) line 9 at RAISE
```

A continuación, probamos con un grupo existente:

```
SELECT * FROM update_report_band('Metallica');
```

t_id_band	t_num_instruments	t_num_members_alive	t_longest_album_title	t_num_short_songs
1	3	3	Master of Puppets	0

Si volvemos a ejecutar no da error:

```
SELECT * FROM update_report_band('Metallica');
```

t_id_band	t_num_instruments	t_num_members_alive	t_longest_album_title	t_num_short_songs
1	3	3	Master of Puppets	0

Ahora intentaremos con un grupo que no tenga ningún miembro vivo.

```
SELECT * FROM update_report_band('Ramones');
```

t_id_band	t_num_instruments	t_num_members_alive	t_longest_album_title	t_num_short_songs
5	3	0	Animal Boy	6

Fijémonos que si no tiene ningún miembro vivo, el valor de *t_num_members_alive* es 0.

Si buscamos un grupo que no tenga ningún álbum veremos que el atributo *t_longest_album_title* es NULL.

```
SELECT * FROM update_report_band('Els Pets');
```

t_id_band	t_num_instruments	t_num_members_alive	t_longest_album_title	t_num_short_songs
7	3	3	[null]	0

Si añadimos un álbum al grupo *Els Pets* nos tendrá que salir cómo el que tiene el título más largo.

```
INSERT INTO album VALUES(14,'Calla i Balla',1991,7);
INSERT INTO song VALUES(32,'S''ha acabat','00:04:49',14);
SELECT * FROM update_report_band('Els Pets');
```

t_id_band	t_num_instruments	t_num_members_alive	t_longest_album_title	t_num_short_songs
7	3	3	Calla i Balla	0

Pregunta 2 (50 % puntuación)

Enunciado

En la tabla ALBUM tenemos la columna *num_long_title_songs* con el objetivo de almacenar el **número de canciones con un título de más de 12 caracteres** del álbum.

Cread un disparador o disparadores, sobre la tabla o tablas que sean necesarias, de manera que se mantenga correctamente actualizada la columna *num_long_title_songs* de la tabla ALBUM.

En concreto, se pide que esta columna refleje los valores pedidos y que éstos siempre se mantengan actualizados en función de los cambios.

Podemos suponer que los usuarios o programas nunca actualizarán directamente la columna *num_long_title_songs* de la tabla ALBUM, y que, en el momento de insertar un nuevo álbum, el valor de la columna *num_long_title_songs* será cero.

Criterios de evaluación

- Las propuestas de solución que no se puedan ejecutar, las que den error de sintaxis, no serán evaluadas.
- Se valorará positivamente el uso de sentencias SQL estándar, al margen de otros elementos que se puedan indicar en el enunciado.
- Para obtener la máxima nota, el código SQL de vuestra solución tiene que ser eficiente. Por ejemplo, se valorará negativamente realizar más *joins* de los necesarios.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir pruebas que cubran todas las posibles situaciones descritas en el enunciado.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir los resultados, mediante capturas de pantalla o de alguna forma similar.
- Para obtener la máxima puntuación, el código que actualice *num_long_title_songs* tiene que pertenecer a una única función.

Solución

La solución se divide en dos partes:

1. **Creación de la función de tratamiento de actualización**: implementación y creación de la función *update_album_statistics* que trata las casuísticas siguientes:
    - Inserción de una nueva canción (*SONG*): si la nueva fila tiene un título de canción con una longitud mayor de 12 caracteres.
    - Modificación del título de una canción del álbum (*id_album*, *title*) (*SONG*)

        - Si la nueva fila (*NEW*) tiene un título con una longitud mayor de 12 caracteres, y la anterior (*OLD*) no, sumamos 1 a *num_long_title_songs* del álbum referenciado por *NEW.id_album*.
        - Si la nueva fila (*NEW*) no tiene un título con una longitud mayor de 12 caracteres, y la anterior (*OLD*) sí, restamos 1 a *num_long_title_songs* del álbum referenciado por *NEW.id_album*.
    - Modificación del identificador del álbum de una canción (*id_album*, *title*) (*SONG*)
        - Si la nueva fila (*NEW*) tiene un título con una longitud mayor de 12 caracteres sumamos 1 a *num_long_title_songs* del álbum referenciado por *NEW.id_album* y restamos 1 a *num_long_title_songs* de *OLD.id_album*.
    - Borrado de una canción (*SONG*): Si la fila a borrar tenía un título con una longitud mayor de 12 caracteres, restamos 1 a *num_long_title_songs* del álbum referenciado por *OLD.id_album*.

2. **Creación de los disparadores**: en segundo lugar, se tienen que crear los disparadores que cubran las casuísticas pedidas en el enunciado. Nos tenemos que preocupar de cubrir las inserciones, borrados y modificaciones de la tabla SONG, que es la que provoca la modificación del valor de *num_long_title_songs* de la tabla ALBUM. El disparador será *AFTER* ya que no queremos que actualice la tabla hasta haber hecho todas las comprobaciones.

Código SQL:

```
SET search_path TO ubd_20211;

CREATE OR REPLACE FUNCTION update_album_statistics()
RETURNS trigger AS $$
BEGIN
	IF (TG_OP = 'INSERT') THEN
		IF(char_length(NEW.title) > 12) THEN
			UPDATE album
			SET num_long_title_songs = num_long_title_songs + 1
			WHERE NEW.id_album = id_album;
		END IF;
	ELSIF (TG_OP = 'UPDATE') THEN
		IF(char_length(NEW.title) > 12) THEN
			UPDATE album
			SET num_long_title_songs = num_long_title_songs + 1
			WHERE NEW.id_album = id_album;
		END IF;
		IF(char_length(OLD.title) > 12) THEN
			UPDATE album
			SET num_long_title_songs = num_long_title_songs - 1
			WHERE OLD.id_album = id_album;
		END IF;
	ELSIF (TG_OP = 'DELETE') THEN
		IF(char_length(OLD.title) > 12) THEN
			UPDATE album
			SET num_long_title_songs = num_long_title_songs - 1
```

```
                WHERE OLD.id_album = id_album;
            END IF;
        END IF;
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_album_statistics AFTER INSERT OR DELETE OR UPDATE OF
id_album, title ON song FOR EACH ROW EXECUTE PROCEDURE update_album_statistics();
```

Con las pruebas siguientes podremos validar cada una de las casuísticas propuestas utilizando el fichero *inserts_db.sql* que os damos de ejemplo como base.

Es importante volver a crear la base de datos, de modo que el orden de ejecución de los scripts sea:

1. **create_db.sql**
2. **Creación del procedimiento y el disparador/disparadores**
3. **inserts_db.sql**

Primero haremos la consulta sobre ALBUM para ver si se ha calculado correctamente el valor de *num_long_title_songs* una vez insertados los datos.

```
SELECT * FROM album ORDER BY num_long_title_songs DESC, id_album;
```

id_album	title	year	id_band	num_long_title_songs
1	Master of Puppets	1.986	1	2
2	Fire and Water	1.970	3	2
3	Aftermath	1.966	2	2
5	Free	1.969	3	2
9	Entre el cielo y el suelo	1.986	8	2
7	Bad Company	1.974	4	1
8	Quina nit	1.990	6	1
10	Ramones	1.976	5	1
4	Ride the lightning	1.984	1	0
6	Tattoo You	1.981	2	0
11	Animal Boy	1.986	5	0
12	Zapatillas	2.005	10	0

13	Destrangis	2.001	9	0

Ahora cambiamos el nombre de la canción ‘Battery’ de ‘Metallica’ a ‘Battery reloaded’.

```
UPDATE song SET title = 'Battery reloaded' WHERE id_song = 1;
SELECT * FROM album WHERE id_album = 1;
```

id_album	title	year	id_band	num_long_title_songs
1	Master of Puppets	1.986	1	3

Vemos que se ha incrementado el número de canciones con título largo.

Ahora eliminamos la canción anterior, y vemos como se vuelve a decrementar el valor de *num_long_title_songs*:

```
DELETE FROM composer WHERE id_song = 1;
DELETE FROM song WHERE id_song = 1;
SELECT * FROM album WHERE id_album = 1;
```

id_album	title	year	id_band	num_long_title_songs
1	Master of Puppets	1.986	1	2

Ahora cambiaremos una canción de un álbum a otro. En concreto pasaremos la canción ‘Ready for Love’ del álbum ‘Bad Company’ al álbum ’Ramones’. Veremos como se ha decrementado el valor de canciones largas de ‘Bad Company’ y se ha incrementado en el álbum ‘Ramones’.

```
UPDATE song SET id_album = 10 WHERE id_song = 17;
SELECT * FROM album WHERE id_album IN (7, 10);
```

id_album	title	year	id_band	num_long_title_songs
10	Ramones	1.976	5	2
7	Bad Company	1.974	4	0

Finalmente, añadiremos dos nuevas canciones al álbum ‘Destrangis’ de ‘Estopa’, una de las cuales con una longitud superior a 12 caracteres y otra con una longitud inferior. Veremos que se incrementa el valor del atributo *num_long_title_songs* pero solo en una unidad.

```
INSERT INTO song VALUES (32,'Vino tinto','0:03:19',13), (33,'Destrangis in the
night','0:03:28',13);
SELECT * FROM album WHERE id_album = 13;
```

id_album	title	year	id_band	num_long_title_songs
13	Destrangis	2.001	9	1