Uso de bases de datos

Práctica 1: El lenguaje SQL I

Queremos disponer de una base de datos para registrar información sobre grupos musicales, canciones y músicos. A continuación, se describen cada una de las relaciones.

Las relaciones con las que trabajamos son las siguientes (claves primarias subrayadas, claves foráneas en cursiva) y los atributos no pueden tener valor nulo a menos que se diga lo contrario.

MUSICIAN (id_musician, name, birth, death, age, gender, nationality)

La relación *MUSICIAN* contiene los datos generales sobre los músicos que aparecen en la BD. En concreto, para cada músico se guarda un número identificador (*id_musician*) que es clave primaria, el nombre (*name*), la fecha de nacimiento (*birth*), fecha de defunción (*death*) que puede ser *NULL*, la edad (*age*), el género (*gender*) y la nacionalidad (*nationality*).

El atributo *gender* solo puede tomar los valores {'*M*', '*F*'}. *M* para el masculino y *F* para el femenino.

BAND (id_band, name, year_formed, year_dissolution, style, origin)

La relación contiene información sobre los grupos musicales. En concreto, para cada grupo, se guarda el identificador (*id_band*) que es clave primaria, el nombre del grupo (*name*), el año de formación (*year_formed*), el año de disolución (*year_dissolution*) que puede ser *NULL*, el estilo musical (*style*) y el país de origen del grupo (*origin*).

El atributo *style* sólo puede tomar los valores {'*Blues*', '*Country*', '*Heavy*', '*Jazz*', '*Pop*', '*Punk*', '*Reggae*', '*Rock*', '*Soul*', '*Thrash*', '*Techno*'}.

ALBUM (id_album, title, year, *id_band*)

Información sobre los álbumes. Per a cada àlbum se almacena el seu identificador (*id_album*) que és clave primaria, el título (*title*), el año de publicación (*year*) y el identificador del grupo musical que ha grabado el álbum (*id_band*), que es clave foránea de *BAND*.

SONG (id_song, title, duration, *id_album*)

Guarda información sobre las canciones. Por cada canción guarda el identificador (*id_song*) que es clave primaria, el título (*title*), la duración en minutos y segundos (*duration*), que no puede ser negativa ni cero y el identificador del álbum al que pertenece la canción (*id_album*), que puede ser NULL y que es clave foránea de *ALBUM*, con política de anulación en caso de borrado.

MEMBER (*id_musician*, *id_band*, instrument)

Información sobre los músicos que forman parte de un grupo o grupos musicales. Se almacenan los valores (*id_musician, id_band, instrument*) que son clave primaria, donde *id_musician* es clave foránea de *MUSICIAN* con política de borrado en cascada, *id_band* es clave foránea de *BAND* con política de actualización en cascada y, finalmente, *instrument* es el instrumento que utiliza el músico en el grupo; la voz (*Vocals*) se considera un instrumento. Un músico puede tocar más de un instrumento en un grupo.

El atributo *instrument*, sólo puede tomar los valores {'*Bass*', '*Drums*', '*Guitar*', '*Keyboard'*, '*Vocals', 'Trumpet', 'Clarinet', 'Oboe', 'Flute'*}.

COMPOSER (*id_musician*, *id_song*, year)

Información sobre las canciones y los compositores de las mismas. Se almacenan los valores (*id_musician, id_song*) que son clave primaria, donde *id_musician* es clave foránea de *MUSICIAN, id_song* es clave foránea de *SONG* y el año en que se compuso la canción (*year*).

ACLARACIONES

En el fichero `create_db.sql` se proporcionan las sentencias SQL necesarias para crear la base de datos. En el fichero `inserts_db.sql` se proporcionan las sentencias de inserción de datos que hay que ejecutar para responder a las preguntas de la práctica.

Recordad que para poder trabajar sobre tablas que son de un esquema de base de datos concreto se debe utilizar el nombre del esquema como prefijo, o hay que tener actualizada la variable `search_path`. Para vuestra comodidad, al principio de cada sesión haced:

```
SET search_path TO "nombre_bd_que_utilicéis";
```

Así mismo, acordaos de eliminar de la solución que entreguéis aquellas sentencias auxiliares que utilicéis para vuestras pruebas, como *DROP*, inserciones de prueba, etc. que puedan alterar las salidas de los resultados esperados.

Nota importante: El SQL implementado en PostgreSQL puede aceptar diferentes variantes de sintaxis, que además pueden variar según la versión que instaléis, y que pueden ser o no SQL estándar. Evitad (a menos que se indique lo contrario) utilizar sentencias de este tipo, y concentraros en las que se explican en los módulos didácticos. Esto es especialmente relevante en el caso del módulo 4 (evaluado en esta primera parte de la práctica), donde se explica SQL estándar. Si utilizáis sentencias SQL estándar, vuestro código funcionará en cualquier SGBD.

Pregunta 1 (30 % puntuación)

Enunciado

En el fichero `create_db.sql` se proporcionan las sentencias SQL necesarias para crear las tablas *MUSICIAN*, *BAND*, *ALBUM, MEMBER* y *COMPOSER*.

Después de un último análisis de la base de datos, se ha llegado a la conclusión que se necesita introducir las mejoras siguientes:

1. ¿Qué sentencias son necesarias para **crear** la tabla *BAND* según la definición dada? (12,5%)
2. También se pide dar las sentencias SQL de **alteración** que permiten hacer algunos cambios en la base de datos:
    2.1. En la tabla *MEMBER* queremos añadir las siguientes restricciones (5%):
        a. Restricción de la clave primaria de *MEMBER*.
        b. Restricción de la clave foránea de *BAND* con política de actualización en cascada en caso de modificación.
    2.2. En la tabla *ALBUM* queremos añadir la restricción de la clave foránea de *BAND* con política de actualización en cascada en caso de modificación (2,5%).
    2.3. En la tabla *BAND* queremos hacer la siguiente modificación, añadir una nueva columna *awards* (premios ganados por el grupo musical) que tendrá por defecto valor 0 y no puede ser negativo (2,5%):
    2.4. En la tabla *MUSICIAN* queremos añadir las siguientes modificaciones (5%):
        a. La columna *death* puede tener valor *NULL* y en caso de tener un valor diferente a NULL, este no puede ser nunca inferior ni igual a *birth*.
        b. La columna age no es necesaria y se debe suprimir.
    2.5. En la tabla *COMPOSER* queremos añadir la restricción de la clave foránea de *MUSICIAN* con política de actualización en cascada en caso de borrado (2,5%).

En el fichero `inserts_db.sql` encontraréis las sentencias de inserción que debéis lanzar sobre las tablas **una vez modificadas**. Tened en cuenta que el número de columnas que deben incluir las tablas es el que viene dado en el fichero `inserts_db.sql`, por lo cual no se aceptarán como soluciones válidas aquellas prácticas que contemplen otro tipo de columna no incluida en este fichero.

Nota importante: Para aplicar los cambios requeridos **NO** debéis utilizar los dominios explicados en el módulo teórico.

Nota: En el siguiente enlace encontraréis información sobre el orden de ejecución de las expresiones.

https://www.postgresql.org/docs/current/static/sql-expressions.html#SYNTAX-EXPRESS-EVAL

Criterios de evaluación

- *Las preguntas tienen un peso que se especifica, según el número de apartados y la dificultad.*

- *Las preguntas no contestadas no penalizan.*
- *Las sentencias SQL que no se puedan ejecutar (den error de sintaxis) no serán evaluadas.*
- *Las sentencias de creación de tablas que propongáis que den error al ejecutar el fichero inserts_db.sql que os proporcionamos no serán evaluadas (en definitiva, el número de columnas que han de incluir las tablas es el que viene dado en el fichero inserts_db.sql).*
- *No se evaluarán propuestas de solución que utilicen dominios (CREATE DOMAIN).*

Solución

1. Creación de la tabla *BAND*

```
CREATE TABLE BAND (
  id_band SMALLINT,
  name VARCHAR(255) NOT NULL,
  year_formed SMALLINT NOT NULL,
  year_dissolution SMALLINT,
  style VARCHAR(255) NOT NULL,
  origin VARCHAR(255) NOT NULL,
  CONSTRAINT PK_BAND PRIMARY KEY(id_band),
    CONSTRAINT STYLE_VALID CHECK (style IN ('Blues', 'Country',
  'Heavy', 'Jazz', 'Pop', 'Punk', 'Reggae', 'Rock', 'Soul', 'Thrash',
  'Techno')));
```

2.1. En la tabla *MEMBER* queremos añadir las siguientes restricciones (5%):
   a. Restricción de la clave primaria de *MEMBER*.
   b. Restricción de la clave foránea de *BAND* con política de actualización en cascada en caso de modificación.

```
ALTER TABLE MEMBER
ADD CONSTRAINT PK_MEMBER PRIMARY KEY(id_musician, id_band,
  instrument),
ADD CONSTRAINT FK_BAND_MEMBER FOREIGN KEY (id_band) REFERENCES
  BAND(id_band) ON UPDATE CASCADE;
```

2.2. En la tabla ALBUM queremos añadir la restricción de la clave foránea de BAND con política de actualización en cascada en caso de modificación.

```
ALTER TABLE ALBUM
ADD CONSTRAINT FK_BAND_ALBUM FOREIGN KEY (id_band) REFERENCES
  BAND(id_band) ON UPDATE CASCADE;
```

2.3. En la tabla BAND queremos hacer la siguiente modificación, añadir una nueva columna awards (premios ganados por el grupo musical) que tendrá por defecto valor 0 y no puede ser negativo.

```
ALTER TABLE BAND
ADD COLUMN awards SMALLINT NOT NULL DEFAULT 0 CHECK
  (awards>=0);
```

2.4. En la tabla MUSICIAN queremos añadir las siguientes modificaciones:

a. La columna death puede tener valor NULL y en caso de tener un valor diferente a NULL, este no puede ser nunca inferior ni igual a birth.
b. La columna age no es necesaria y se debe suprimir.

```
ALTER TABLE MUSICIAN
ADD CONSTRAINT CHECK_DEATH CHECK (death IS NULL OR death >
  birth),
DROP COLUMN age;
```

2.5. En la tabla COMPOSER queremos añadir la restricción de la clave foránea de MUSICIAN con política de actualización en cascada en caso de borrado.

```
ALTER TABLE COMPOSER
ADD CONSTRAINT FK_MUSICIAN_COMPOSER FOREIGN KEY (id_musician)
  REFERENCES MUSICIAN(id_musician) ON DELETE CASCADE;
```

Pregunta 2 (45 % puntuación)

Enunciado

a) Diseñad una consulta que devuelva los músicos que no sean americanos, de grupos que han lanzado algún álbum en la década de los 90, así como el número de canciones que han compuesto a lo largo de su carrera. En concreto, queremos saber el nombre y la edad del músico, el título del álbum y el año de lanzamiento, y el número de canciones que ha compuesto, (0 si no ha compuesto ninguna), ordenados por edad de mayor a menor y en caso de empate por orden alfabético del nombre.

b) Diseñad una consulta que devuelva las canciones de duración superior a 3', publicadas en algún álbum del año 1986, compuestas por músicos de bandas en las que todos sus miembros estén vivos. En concreto, queremos que nos muestre el nombre y la duración de la canción, el nombre de la banda, y el título del álbum que se publicó.

c) Diseñad una vista (*seven_top_composers*) que obtenga el nombre y el año de nacimiento de los 7 músicos que más canciones han compuesto, así como el número de canciones compuestas, la duración media de las canciones y el número de bandas de las que han sido miembros. Queremos el resultado ordenado por el número de canciones compuestas de mayor a menor y, en caso de empate, por orden alfabético inverso del nombre.

Nota : En los siguientes enlaces encontraréis información sobre:

- funciones agregadas: https://www.postgresql.org/docs/8.2/functions-aggregate.html

- la cláusula LIMIT: https://www.postgresql.org/docs/8.1/queries-limit.html
- funciones de fecha y hora: https://www.postgresql.org/docs/8.1/functions-datetime.html

Criterios de evaluación

- *Todas las preguntas tienen el mismo peso.*
- *Las preguntas no contestadas no penalizan.*
- *Las sentencias SQL que no se puedan ejecutar (den error de sintaxis) no serán evaluadas.*
- *Se valorará positivamente el uso de sentencias SQL estándar (al margen de otros elementos indicados en el enunciado).*
- *Para obtener la máxima nota, la propuesta de solución de cada pregunta debe incluir el resultado (captura de pantalla o similar).*
- *Para obtener la máxima nota en cada pregunta, la propuesta de solución se tiene que ajustar estrictamente a lo que se pide en el enunciado (por ejemplo, tiene que incorporar todas las columnas que se esperan en el resultado, se deben hacer las ordenaciones pedidas …)*
- *Igualmente para obtener la máxima nota la presentación de los resultados debe ser precisa y amigable (nombres de las columnas, redondeo de decimales, …)*

Solución

a)
```
SELECT DISTINCT m.name, EXTRACT (year from AGE(m.birth)) AS AGE,
a.title, a.year, COUNT(DISTINCT c.id_song) AS Songs
FROM MUSICIAN m NATURAL JOIN MEMBER me NATURAL JOIN ALBUM a LEFT JOIN
COMPOSER c ON (m.id_musician=c.id_musician)
WHERE m.nationality <> 'American' AND a.year BETWEEN 1990 AND 1999
GROUP BY m.name,  AGE, a.title, a.year
ORDER BY AGE DESC, m.name ASC;
```

```
      name      | age |   title    | year | songs
----------------+-----+------------+------+-------
 Pep Sala       |  61 | Quina nit | 1990 |     2
 Carles Sabater |  59 | Quina nit | 1990 |     0
(2 rows)
```

b)
```
SELECT DISTINCT s.title, s.duration, b.name, a.title
FROM SONG s NATURAL JOIN COMPOSER C NATURAL JOIN MEMBER m NATURAL
JOIN BAND b INNER JOIN ALBUM a ON (s.id_album=a.id_album AND
m.id_band=a.id_band)
WHERE s.duration > '00:03:00' AND a.year = 1986 AND
m.id_band NOT IN (    SELECT me.id_band
                      FROM MEMBER me NATURAL JOIN MUSICIAN mu
                      WHERE mu.death IS NOT NULL);
```

```
      title      | duration |  name  |           title
-----------------+----------+--------+---------------------------
 Hijo de la luna | 00:04:00 | Mecano | Entre el cielo y el suelo
(1 row)
```

```
c) CREATE OR REPLACE VIEW seven_top_composers AS (
        SELECT m.name, EXTRACT (YEAR FROM m.birth) birth_year,
             COUNT (DISTINCT s.id_song) songs,
             date_trunc('second',AVG(s.duration)) duration_avg,
             COUNT (DISTINCT me.id_band) Bands
        FROM MUSICIAN m NATURAL JOIN COMPOSER c NATURAL JOIN SONG s
             NATURAL JOIN MEMBER me
        GROUP BY m.name, birth_year
        ORDER BY songs DESC, m.name DESC
        LIMIT 7);

   SELECT * FROM seven_top_composers;

      name      | birth_year | songs | duration_avg | bands
----------------+------------+-------+--------------+-------
 Paul Rodgers   |       1949 |     7 | 00:04:26     |     2
 Mick Jagger    |       1943 |     5 | 00:03:57     |     1
 Lars Ulrich    |       1963 |     4 | 00:06:53     |     1
 Joey Ramone    |       1951 |     4 | 00:02:11     |     1
 James Hetfield |       1963 |     4 | 00:06:53     |     1
 Keith Richards |       1943 |     3 | 00:03:07     |     1
 Pep Sala       |       1960 |     2 | 00:02:55     |     1
(7 rows)
```

Pregunta 3 (25 % puntuación)

Enunciado

Se ha descubierto que los compositores de alguna canción de duración superior a 5', que todavía están vivos y que pertenecen a bandas en que todos los miembros son de la misma nacionalidad, nacieron realmente 6 meses antes.

Proponed una **única sentencia SQL** para corregir los registros incorrectos (en caso de utilizar más de una, el ejercicio se considerará incorrecto). Por otro lado, mostrad el conjunto de las filas que se actualizan.

Nota importante: cuando probéis la implementación de esta actualización, tened en cuenta que siempre debéis partir de la misma base de datos, de otro modo podéis encontrar incoherencias en vuestros análisis.

Criterios de evaluación

- *Las sentencias SQL que no se puedan ejecutar (den error de sintaxis) no serán evaluadas.*

- *Para obtener la máxima nota en cada pregunta, la propuesta de solución se tiene que ajustar estrictamente a lo que se pide en el enunciado (hay que actualizar los datos estrictamente necesarios) y con una única sentencia de UPDATE.*
- *Para obtener la máxima nota, la solución debe ser eficiente (por ejemplo, se valorará negativamente hacer más joins de las necesarias).*
- *Se debe argumentar a la respuesta proporcionada. Para obtener la máxima puntuación se debe mostrar la relación de filas que se actualizan.*
- *Igualmente para obtener la máxima nota la presentación de los resultados debe ser precisa y amigable (nombres de las columnas, redondeo de decimales, …)*

Solución

Filas afectadas antes de hacer la actualización:

```
SELECT DISTINCT m.name, m.birth
FROM MUSICIAN m NATURAL JOIN COMPOSER c NATURAL JOIN SONG s NATURAL JOIN
      MEMBER me
WHERE m.death IS NULL AND s.duration > '00:05:00' AND
      m.nationality = ALL (SELECT m1.nationality
                           FROM MUSICIAN m1 NATURAL JOIN MEMBER me2
                           WHERE me.id_band = me2.id_band)
ORDER BY m.birth;
```

```
     name     |   birth
--------------+------------
 Mick Jagger  | 1943-07-26
 Simon Kirke  | 1949-07-28
 Paul Rodgers | 1949-12-17
(3 rows)
```

Actualización:

```
UPDATE MUSICIAN
      SET birth = birth + interval '6 month'
      WHERE id_musician IN (
            SELECT DISTINCT m.id_musician
            FROM MUSICIAN m NATURAL JOIN COMPOSER c NATURAL JOIN SONG s
                  NATURAL JOIN MEMBER me
            WHERE m.death IS NULL AND s.duration > '00:05:00' AND
                  m.nationality = ALL (SELECT m1.nationality
                                       FROM MUSICIAN m1 NATURAL JOIN MEMBER me2
                                       WHERE me.id_band = me2.id_band));
```

Filas afectadas después de hacer la actualización:

```
UPDATE 3
     name     |   birth
--------------+------------
 Mick Jagger  | 1944-01-26
 Simon Kirke  | 1950-01-28
 Paul Rodgers | 1950-06-17
(3 rows)
```