UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA
DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS

Trabajo especial
Entrega

Bases de datos

Carlos Jesús Mariano Nieves
(cnieves@alumnos.exa.unicen.edu.ar)

Francisco Murillo
(fmurillo@alumnos.exa.unicen.edu.ar)

Guillermo Tomas Cesario Provenzano
(gcesarioprovenzano@alumnos.exa.unicen.edu.ar)

A. AJUSTE DEL ESQUEMA

Se generó un script (G15_Creacion.sql) el cual se encarga de generar el esquema de tablas junto a sus restricciones de nulidad, primary key y foreign key.

B. ELABORACIÓN DE RESTRICCIONES

Escriba la restricción de la manera que considere más apropiada en SQL estándar declarativo, indicando su tipo y características correspondientes.

i. Se debe consistir que la fecha de inicio de la publicación de la edición sea anterior a la fecha de fin de la publicación del mismo si esta última no es nula.

Se trataba de una restricción de trupla, por eso utilizamos un check.

```
ALTER TABLE GR15_Evento_Edicion DROP CONSTRAINT IF EXISTS CHK_GR15_fechas;
ALTER TABLE GR15_Evento_Edicion ADD CONSTRAINT CHK_GR15_fechas
CHECK ((fecha_inicio_pub < fecha_fin_pub) OR (fecha_fin_pub IS NULL));
```

ii. Cada categoría no debe superar las 50 subcategorías.

Al ser una restricción de Tabla, lo más conveniente es utilizar nuevamente check.

```
ALTER TABLE GR15_SUBCATEGORIA ADD CONSTRAINT CHK_GR15_MaxCategorias
CHECK     (
   NOT EXISTS (
         SELECT 1
         FROM GR15_SUBCATEGORIA S
         GROUP BY (S.id_categoria)
         HAVING COUNT(I.id_comp) > 50
                  )
   );
```

iii. La suma de los aportes que recibe una edición de un evento de sus patrocinantes no puede superar el presupuesto establecido para la misma.

Como se requiere obtener tuplas de más de una de las tablas de la Base de Datos, utilizamos Assertion.

```
CREATE ASSERTION ASS_GR15_LimiteAporte
CHECK    (
   NOT EXISTS (
        SELECT e.id_evento, SUM(p.aporte)
        FROM GR15_Patrocinios P NATURAL JOIN GR15_Evento_Edicion E
        GROUP BY (e.id_Evento, e.nro_edicion)
        HAVING SUM (P.aporte) <(
                                    SELECT ee.presupuesto
                                    FROM GR15_Evento_Edicion ee
                                    WHERE(ee.id_evento = e.id_evento)
                                    )
   );
```

iv. Los patrocinantes solo pueden patrocinar ediciones de eventos de su mismo distrito.

Como se requiere obtener tuplas de más de una de las tablas de la Base de Datos, utilizamos Assertion.

```
CREATE ASSERTION ASS_GR15_DistritoPatrocinio
CHECK (
        NOT EXISTS(
                    SELECT 1
                    FROM GR15_Patrocinante P, GR15_Evento E
                    WHERE (P.id_patrocinate = new.id_patrocinate)
                        AND (E.id_evento = new.id_evento)
                        AND (P.id_distrito != E.id_distrito)
                  )
        )
   );
```

b. Implementarla en PostgreSQL de la forma más adecuada, según las posibilidades que ofrece el SGBD

Se requirió implementar de manera procedural los incisos ii, iii y iv (ya que postresql no soporta subconsultas dentro de CONSTRAINTS, ni ASSERTIONS. Dichas implementaciones se encuentran en el script GR15_Cambios.sql.

c. Provea una sentencia de modificación, borrado y/o inserción según corresponda, de modo que promueva la activación de la restricción.

Para todos los casos se contemplaron situaciones en las que la restricción diera error.

i. Para el primer caso basta con Agregar o Modificar un EVENTO_EDICION para que la fecha de fin no sea NULL y sea menor que la fecha de inicio.

ii. En este caso debemos agregar más de 51 subcategorías de la misma categoría.

iii. Se activan las restricciones de varias formas:

- Ante INSERTS en Patrocinios: Una vez que se supere el presupuesto de la edición, con la suma de los aportes, saltará la restricción.
- Ante UPDATES en Patrocinios: Si se modifica la id_evento o aporte, de uno o más patrocinios ,y con esto se supera el presupuesto para algún Evento_edicion, saltará la restricción.
- Ante UPDATES en Evento_Edicion: Sería el caso en que se modifique el valor de presupuesto y quede por debajo de la suma de los aportes.

iv. Se activan las restricciones de varias formas:

- Ante INSERTS en Patrocinios: Si se quiere agregar una tupla que no cumpla con la condición P.id_distrito <> E.id_distrito, saltará la restricción.
- Ante UPDATES en Patrocinante: Si al modificar el campo 'id_distrito' de una tupla, entra en conflicto con la condición P.id_distrito <> E.id_distrito, saltara la restricción.
- Ante UPDATES en Evento: Si al modificar el campo 'id_distrito' de una tupla, entra en conflicto con la condición P.id_distrito <> E.id_distrito, saltara la restricción

Las sentencias se encuentran comentadas en el script GR15_cambios.sql

C. SERVICIOS

a. Cuando se crea un evento, debe crear las ediciones de ese evento, colocando como fecha inicial el 1 del mes en el cual se creó el evento y como presupuesto, el mismo del año pasado más un 10 %, en caso de que no hubiera uno el año pasado, colocar 100.000

Para resolver este servicio, se creó un trigger por INSERT en la tabla EVENTO, que se propagara a la tabla EVENTO_EDICION.

b. Todas las fechas, entre EVENTO y EVENTO_EDICION (recordar en EVENTO_EDICION también) tienen que ser coherentes.

Para resolver este servicio, se creó un trigger por INSERT or UPDATE en EVENTO y en EVENTO_EDICION, que mantendrá la coherencia entre las fechas.

Las sentencias se encuentran en el script GR15_cambios.sql

D. DEFINICIÓN DE VISTAS

1. Identificador de los Eventos cuya fecha de realización de su último encuentro esté en el primer trimestre de 2020.

Se crea una vista sobre EVENTO. Es actualizable, ya que contiene la clave primaria.

2. Datos completos de los distritos indicando la cantidad de eventos en cada uno

Se crea una vista que involucra a las tablas Evento y Distrito. No es actualizable, ya que es una vista de más de una tabla, además de poseer una función de agrupamiento.

3. Datos Categorías que poseen eventos en todas sus subcategorías.

Se crea una vista que involucra a Categoría, Subcategoría y Evento. No es actualizable ya que es una vista de más de una tabla.

Las sentencias se encuentran en el script GR15_cambios.sql

E. TUDAI - SITIO A REALIZAR

● ***Generar un Listado del TOP 10 de usuarios que participa en más eventos.***

```
$sentencia = $this->db->prepare("SELECT id_usuario FROM gr15_evento GROUP BY id_usuario
ORDER BY COUNT(id_usuario) LIMIT 10");
      $sentencia->execute();
      $Bases = $sentencia->fetchAll(PDO::FETCH_OBJ);
      return $Bases;
```

● ***Generar un Listado que devuelva el o los usuarios de acuerdo a un patrón de búsqueda que contenga todos los datos del usuario junto con la cantidad de eventos publicados y ediciones de dichos eventos.***

```
$sentencia = $this->db->prepare ("SELECT u.id_usuario,u.nombre,u.apellido, u.e_mail,
u.password,
            (SELECT COUNT(id_evento)
               FROM GR15_Evento e
               WHERE e.id_usuario = '%$id_usuario%'
               GROUP BY e.id_usuario) as Cantidad eventos publicados,
            (SELECT COUNT(ee.id_evento)
               FROM GR15_Evento_Edicion ee
                  JOIN GR15_Evento e ON (ee.id_evento = e.id_evento)
               WHERE e.id_usuario = '%$id_usuario%'
               GROUP BY (e.id_usuario)
            ) as Cantidad de ediciones
            FROM GR15_Usuario u
               WHERE (u.id_usuario is NULL OR U.id_usuario = '%$id_usuario%')
               AND (u.nombre is NULL OR u.nombre LIKE '%$nombre%')
               AND (u.apellido is NULL OR u.apellido LIKE '%$apellido%')
               AND (u.e_mail is NULL OR u.e_mail LIKE '%$email%')
               AND (u.password is NULL OR u.password LIKE '%$password%')");

      $sentencia->execute(array($id_usuario,$nombre,$apellido,$email,$password));
      $Bases = $sentencia->fetchAll(PDO::FETCH_OBJ);
```

```
    return $Bases;
```

$id_usuario, $nombre, $apellido, $email, $password son variables.