TRABAJO ESPECIAL DE BASE DE DATOS

Miguel Agustin

Manterola Enrique

Gallego Maximiliano

29/06/2022

INTRODUCCIÓN

El Trabajo Práctico Especial consiste en la resolución de un conjunto de controles y servicios sobre una base de datos que mantiene un Sistema de Publicidad de Juegos en Línea la empresa "**PJL Sistemas Tandil**".

Nosotros realizaremos todos los scripts correspondientes para solucionar dichos controles y servicios solicitados por la cátedra.

PROCEDIMIENTO

Comenzamos copiando el Script de Tablas que nos dieron, reemplazando la nomenclatura tal como se nos indicó.

Luego, realizamos las restricciones en SQL estándar declarativo para los siguientes enunciados:

1. La fecha del primer comentario tiene que ser anterior a la fecha del último comentario si este no es nulo.
2. Cada usuario sólo puede comentar una vez al día cada juego.
3. Un usuario no puede recomendar un juego si no ha votado previamente dicho juego.
4. Un usuario no puede comentar un juego que no ha jugado.

Esto nos dio como resultado:

1. Una restricción de tupla en la cual chequeamos que la fecha_prim_com siempre sea menor que fecha_ult_com o que fecha_ult_com sea null
2. Una restricción de tabla (la cual no soporta PostgreSQL) donde chequeamos que Agrupando por fecha_comentario, id_usuario, id_juego no haya mas de una tupla por que en ese caso un mismo usuario comentaria un mismo juego en el mismo dia. (Hay un error al no extraer Año, mes y dia de fecha comentario se agrupa por Año, Mes, Día y hora.
3. Una restricción de Assertion en la cual intervienen dos tablas (Recomendacion y Voto). Aca chequeamos que no exista una recomendación para la cual no exista un voto donde el usuario y juego sean los mismo que la tupla de recomendacion.
4. Una restricción de Assertion en la cual intervienen dos tablas(Comenta y Juega).
   Aca chequeamos que no exista una tupla en comenta para la cual no exista una tupla en juega con los mismo id_usuario e id_juego.

Luego ya que PostgreSQL no soporta las restricciones 2,3 y 4 las implementamos cada cual como un Trigger distinto.

1. Soportada por PostgreSQL
2. Como nos pide que se haga un único comentario por dia, realizamos un trigger el cual se despierte antes de un insert o un update de id_juego, id_usuario y fecha_comentario en la tabla comentario. Cuando despierta el trigger este mismo va a realizar una cuenta de todos los comentarios que tengan la misma fecha_comentario, id_usuario e id_juego que la nueva tupla a insertar o a modificar. Luego vuelca esa cuenta en una variable la cual la usamos para evaluar, si conteo = 1 entonces ya comento ese dia, ese juego. Por lo tanto da excepción.
3. Como nos pide que no se pueda comentar sin antes votar, realizamos un trigger el cual se despierte antes de un insert o un update de id_juego, id_usuario en recomendación. Cuando se despierta el trigger este mismo va a realizar un conteo de las recomendaciones de id_juego, id_usuario de la tupla a insertar que no tengan tupla en la tabla voto con el mismo id_juego e id_usuario. Si el conteo da >= 1 da excepción ya que no podes comentar sin antes votar.
4. El mismo razonamiento que la anterior pero en las tablas comenta y juega.

   Todos los triggers son de granularidad For Each Row, y Before, ya que todos los chequeos se pueden realizar antes de que se inserte la nueva tupla y nos ahorramos muchos controles.

Luego teníamos que realizar un servicio que mantenga actualizado las tablas Comenta y comentario, para lo cual realizamos un trigger para los inserts en la tabla comentarios, el cual se activa antes del insert y chequea que si en la tabla comenta hay una tupla con id_juego e id_usuario que sea igual al de la tupla a insertar.
En caso de que haya mira la fecha_ult_com en la tabla comenta y la compara con la nueva fecha_comentario. Si la nueva fecha es mayor al la fecha_ult_com hace un update y cambia la actualiza.
En caso de que no haya una tupla en comenta que tenga el mismo id_juego e id_usuario crea la tabla con fecha_ult_com en null.

Para el caso del update y el delete realizamos otro trigger el cual tiene la misma granularidad (for each row) pero se activa after.
Esta decisión la tomamos ya que nos es más fácil buscar la nueva fecha_comentario con todas las tuplas insertadas.
Entonces para el Delete vamos a buscar la mayor fecha de los comentarios de ese usuario para ese juego y vamos a realizar un update en Comenta cambiando fecha_ult_com por esa fecha (Esta se va a updatear solo cuando se borre el ultimo comentario).
Y para el update utilizamos el mismo razonamiento.

Luego teníamos que realizar vistas:

1. Listar Todos los comentarios realizados durante el último mes descartando aquellos juegos de la Categoría “Sin Categorías”.
   Básicamente creamos la vista con la consulta de traer todos los comentarios de la tabla Comentarios ensamblando con la tabla juego, y luego con la tabla categoria siempre y cuando la categoría de las tuplas sea distinto a ‘Sin categoría’ y la fecha_comentario este en un intervalo de un mes del último comentario.
   Esta vista no es actualizable ya que en el front tiene 3 tablas, para que sea actualizable tenia que haber usado In o Not In. Ademas de no tener la pk.

2. Identificar aquellos usuarios que han comentado todos los juegos durante el último año, teniendo en cuenta que sólo pueden comentar aquellos juegos que han jugado.
   Aca creamos la vista con la consulta de traer todos los nombres de la tabla Usuario ensamblando con la tabla Comentarios donde el conteo de todos los distintos id_juego que comento sea igual a la cantidad de juegos que hay en la tabla juegos.
   Esta vista no es actualizable por la misma razón que el anterior, además de no tener la pk

3. Realizar el ranking de los 20 juegos mejor puntuados por los Usuarios. El ranking debe ser generado considerando el promedio del valor puntuado por los usuarios y que el juego hubiera sido calificado más de 5 veces.
   Aca creamos la vista con la consulta que nos trae toda la informacion de juego y el promedio de votos de ese juego de la tabla Juego ensamblando con Voto, siempre y cuando el juego tenga más de 5 votos.

Por último tuvimos que desarrollar una web en la cual se muestre un listado con el top 10 de los juegos más votados y un patrón de búsqueda que listará todo los datos de un usuario, la cantidad de juegos que jugo y la cantidad de votos que realizó.
Realizamos la conexión a la base de datos y realizamos las consultas necesarias para ambos pedidos.
Para la consulta del top 10 seleccionamos todo de la tabla juego, contamos los votos, agrupamos por id_juego y hacemos un ordenamiento descendente por cantidad de votos.
Para la segunda consulta el usuario ingresa un número que va a presentar el id_usuario, seleccionamos todo los datos de ese usuario, y contamos la cantidad de juego y votos que realizó.

Debido al inconveniente de no poder subir la página que desarrollamos con sus respectivas consultas, decidimos adjuntar al informe un par de imagenes demostrando el flujo y el funcionamiento de cada una.

En esta imagen vemos que tenemos los dos botones y donde uno de ellos tiene el input donde podemos colocar el id de un usuario que lo usamos como motor de búsqueda.

Top 10 de juego mas votados

Nombre juego	Descripcion
libero.	eu, ligula. Aenean euismod mauris eu elit. Nulla facilisi.
est	tellus, imperdiet non, vestibulum nec, euismod in, dolor. Fusce feugiat.
dignissim.	eget, dictum placerat, augue.
aptent	Sed neque.
mollis	amet, consectetuer

Una vez que apretamos el botón “top 10” va a generar un Listado del TOP 10 de juegos más votados.

En esta imagen mostramos que ingresamos el id de un usuario y en la próxima imagen vamos a mostrar el listado que genera.

Esto lo que devuelve una vez que apretamos el botón “Actualizar”.

model > UsuarioModel.php

```php
<?php

class UsuarioModel
{
  private $db;

  function __construct(){
    $this->db = new PDO('pgsql:host=dbases.exa.unicen.edu.ar;port=6432;dbname=cursada;user=unc_248479;password=248479');
  }

  //MOSTRAR TOP 10
  public function Mostrar(){
    $consulta = $this->db->prepare( "SELECT j.*, COUNT(v.id_voto) FROM GR20_JUEGO j JOIN GR20_VOTO v ON (j.id_juego = v.id_juego)
    GROUP BY (V.id_juego,j.id_juego) ORDER BY COUNT(v.id_voto) DESC ");
    $consulta->execute();
    $top = $consulta->fetchAll(PDO::FETCH_OBJ);
    return $top;
  }

  public function Juego($juego){
    $consulta = $this->db->prepare( "SELECT u.*,COUNT(j.id_juego) AS cant_juegos, COUNT(v.id_voto) AS cant_votos FROM gr20_usuario u JOIN gr20_juega j ON (u.id_usuario = j.id_usuario)
    JOIN gr20_voto v ON (u.id_usuario = v.id_usuario) WHERE u.id_usuario =  ?  GROUP BY (u.id_usuario) ; ");
    $consulta->execute([$juego]);
    $top = $consulta->fetchAll(PDO::FETCH_OBJ);
    return $top;
  }
  }


  ?>

```

Por último adjuntamos la imagen del codigo, donde mostramos ambas consultas y la conexión a la db.