SI

Práctica 3

Christian Ramos Maroto
Adrián Lorenzo Mateo

3.1 Consultas avanzadas

3.1.1 Consulta A:

- **Análisis:**

  La consulta debe usuario que mas paginas a procesado cada mes junto con el mes, el año y el numero de paginas procesadas.

  Para obtener este resultado hay que obtener el total de paginas procesadas por cada usuario cada mes y despues quedarnos solo con el maximo de cada mes.

- **Consulta:**

```
SELECT ano_mes, login AS Usuario, numlibros AS Libros_procesados
   FROM usuario,
      (SELECT max(numlibros) AS maxlibros, anomes AS ano_mes
       FROM
         (SELECT count(*) AS numlibros, corrector
         AS cor, to_char(fecha_fin, 'yyyy-mm') AS anomes
         FROM correccion WHERE fecha_fin IS NOT NULL
         GROUP BY to_char(fecha_fin, 'yyyy-mm'), corrector)
         AS sub GROUP BY  anomes) AS max ,
         (SELECT count(*) AS numlibros, corrector AS cor,
         to_char(fecha_fin, 'yyyy-mm') AS anomes
         FROM correccion WHERE fecha_fin IS NOT NULL
         GROUP BY to_char(fecha_fin, 'yyyy-mm'), corrector) AS
        total_libros
   WHERE maxlibros = total_libros.numlibros AND usuario.id_usuario =
cor
  ORDER BY ano_mes asc;
```

3.1.2 Consulta B:

- **Análisis:**

  La consulta debe mostrar el número de páginas pendientes y procesadas durante un año agrupadas ambas por cada fase.

Para obtener las páginas procesadas simplemente hay que seleccionar aquellas páginas corregidas en la tabla corrección cuya fecha de finalización (fecha_fin) es mayor que la fecha actual de la DB menos un año y finalmente contarlas agrupándolas por la fase.

Asimismo para las páginas pendientes hay que acceder a la tabla corrección pero contando aquellas páginas cuya corrección está en trámite, es decir, que tienen el atributo fecha_fin sin asignar (a NULL).

Finalmente para obtener las dos selecciones agrupadas se realiza un LEFT JOIN con las dos subconsultas, unidas por la fase.

- **Consulta:**

```
SELECT
        pagProcesadas.id_fase AS fase,
        pendientes,
        procesadas
FROM (SELECT
        COUNT(n_pagina) AS procesadas,
        id_fase
        FROM correccion
        WHERE
        fecha_fin>=timestamp'2011-10-16'-interval' 1 year'
        GROUP BY id_fase)
        AS pagProcesadas

LEFT JOIN (SELECT
        COUNT(n_pagina) AS pendientes,
        id_fase
        FROM correccion
        WHERE
        fecha_fin is null
        GROUP BY id_fase)
        AS pagPendientes
ON pagProcesadas.id_fase=pagPendientes.id_fase
ORDER BY fase;
```

- **Comprobaciones:**

La verificación de esta consulta es sencilla, simplemente consistiría en ejecutar las dos subconsultas por separado comprobando que los resultados son equivalentes.

3.1.3 Consulta C:

- **Análisis**:

  La consulta debe mostrar agrupados por fase los libros que más tiempo han estado evaluados en el sistema, es decir, los libros cuyo intervalo en cada fase entre la primera página corregida y la última es mayor.

  Para obtener el máximo intervalo simplemente hay que restar por cada fase y libro la página más nueva (fecha_fin mayor) menos la página más antigua (fecha_inicio mínima) y seleccionar el intervalo mayor.

- **Consulta**:

```
CREATE OR REPLACE VIEW intervalo AS
SELECT
        isbn,
        MAX(date(fecha_fin))-MIN(date(fecha_inicio)) AS
tiempo,
        id_fase
FROM correccion
GROUP BY isbn,id_fase
ORDER BY tiempo DESC;


SELECT
        max_intervalo.id_fase AS fase,
        isbn,
        intervalo.tiempo
        FROM intervalo,
(SELECT
        MAX(tiempo) AS max_intervalo,
        id_fase
FROM intervalo
GROUP BY id_fase)
AS max_intervalo
WHERE
        intervalo.tiempo=max_intervalo.max_intervalo
ORDER BY fase;
```

- **Comprobaciones** :

  Para confirmar los resultados de esta consulta hemos realizado la siguiente prueba:

  Cambiar la tupla de la tabla correccion con isbn “0000000021472” , n_pagina 1 y fase PF el atributo fecha_inicio

a “2002-12-20”, lo que supone que debería aparecer en la consulta como el libro que más lleva evaluado en PF con 2957 días.

3.2 Triggers, Funciones y Procedimientos almacenados

3.2.1 Trigger A:

- **Análisis**:

  El trigger se debe ejecutar cada vez que se actualize una fila de la tabla proyecto, si se modifica el atributo fase se realiza una notificación al coordinador del proyecto.

- **Código**:

```
CREATE OR REPLACE FUNCTION tgAlertaCambioFase()
        RETURNS TRIGGER AS $tgAlertaCambioFase$
DECLARE

        tituloLibro text;
        infoAlerta text;

BEGIN

        IF OLD.id_fase != NEW.id_fase THEN

                infoAlerta:= 'El Libro con isbn ' || OLD.isbn ||
                ' ha cambiado de la fase ' || OLD.id_fase ||
                 ' a la ' || NEW.id_fase;

                RAISE NOTICE 'El Libro % ha cambiado de la
                fase % a la %'
                ,NEW.isbn,OLD.id_fase,NEW.id_fase;

                INSERT INTO alerta(fecha,id_usuario,texto)
                        VALUES (timestamp'2011-10-
                        16',OLD.coordinador,infoAlerta);


                END IF;

                RETURN NEW;
                END;
```

```
$tgAlertaCambioFase$ LANGUAGE plpgsql;

CREATE  TRIGGER tgAlertaCambioFase AFTER
UPDATE ON proyecto FOR EACH ROW
EXECUTE PROCEDURE tgAlertaCambioFase();
```

- **Comprobaciones** :

Las pruebas que hemos realizado son las siguientes:

Modificación de una fila de proyecto pero no se cambia el atributo fase, ejecutándose el trigger pero no realizando ninguna función.

Cambiar en una fila de proyecto la fase, el trigger se ejecuta realizando las notificaciones correspondientes.

3.2.2 Trigger B:

- **Análisis:**

Este disparador se tiene que poner en funcionamiento cada vez que se corrija una página, es decir, que la fecha_fin pasa de NULL a NOT NULL.

Tiene que incrementar las páginas corregidas del proyecto en la fase correspondiente y las páginas corregidas por el usuario corrector. Debe comprobar si el usuario cumple los requisitos para promocionarlo a un nuevo nivel y de forma equivalente ascender a una nueva fase el proyecto si cumple los requisitos especificados.

- **Código**:

```
CREATE OR REPLACE FUNCTION tgCheckCorreccion() RETURNS
TRIGGER AS $tgCheckCorreccion$
        DECLARE

                proyectoRow proyecto%ROWTYPE;
                usuarioRow usuario%ROWTYPE;
                promocion integer;
                mensaje text;

        BEGIN
                IF OLD.fecha_fin IS NULL AND NEW.fecha_fin IS NOT
                NULL THEN
```

```
                SELECT INTO proyectoRow * FROM proyecto
                WHERE proyecto.isbn=NEW.isbn;

            IF NEW.id_fase='PF' THEN

                UPDATE pagina SET id_fase='PF' WHERE
                isbn=NEW.isbn AND n_pagina=NEW.n_pagina;
                UPDATE proyecto SET
                n_paginaspf=proyectoRow.n_paginaspf+1
            WHERE isbn=OLD.isbn;
                SELECT INTO proyectoRow * FROM proyecto
                WHERE isbn=OLD.isbn;

            IF proyectoRow.n_paginas=proyectoRow.n_paginaspf
        THEN

                UPDATE proyecto SET id_fase='PF' WHERE
            isbn=NEW.isbn;

            RAISE NOTICE 'Proyecto correspondiente al
        libro %, ha cambiado de la fase F0 a PF.',
NEW.isbn;

            END IF;



                SELECT INTO usuarioRow * FROM usuario
            WHERE usuario.id_usuario=NEW.corrector;

                UPDATE usuario SET
                n_paginaspf=usuarioRow.n_paginaspf+1
            WHERE id_usuario=NEW.corrector;
                 SELECT INTO usuarioRow * FROM usuario
            WHERE id_usuario=NEW.corrector;

                SELECT INTO promocion valor FROM var_config
            WHERE variable='pag_promotf';

            IF usuarioRow.nivel='F0' AND
        usuarioRow.fecha<=timestamp'2011-10-16'-
interval'21 days' AND
usuarioRow.n_paginaspf=promocion THEN

            UPDATE usuario SET nivel='PF' WHERE
            id_usuario=NEW.corrector;
                RAISE NOTICE 'El usuario % ha sido
            promocionado a nivel PF', usuarioRow.login;
```

```
                END IF;

                 ELSIF NEW.id_fase='SF' THEN

                        UPDATE pagina SET id_fase='SF' WHERE
                        isbn=NEW.isbn AND n_pagina=NEW.n_pagina;
                        UPDATE proyecto SET
                        n_paginassf=proyectoRow.n_paginassf+1 WHERE
                isbn=OLD.isbn;
                        SELECT INTO proyectoRow * FROM proyecto WHERE
                        isbn=OLD.isbn;

                IF proyectoRow.n_paginas=proyectoRow.n_paginassf THEN

                UPDATE proyecto SET id_fase='SF' WHERE
        isbn=NEW.isbn;

                RAISE NOTICE 'Proyecto correspondiente al libro %,
                ha cambiado de la fase PF a SF.', NEW.isbn;

                END IF;



                SELECT INTO usuarioRow * FROM usuario WHERE
                usuario.id_usuario=NEW.corrector;

                UPDATE usuario SET
                n_paginassf=usuarioRow.n_paginassf+1 WHERE
        id_usuario=NEW.corrector;
                SELECT INTO usuarioRow * FROM usuario WHERE
                id_usuario=NEW.corrector;

                SELECT INTO promocion valor FROM var_config WHERE
        variable='pag_promosf';

                IF usuarioRow.nivel='PF' AND
        usuarioRow.fecha<=timestamp'2011-10-16'-
interval'41     days' AND
usuarioRow.n_paginassf=promocion THEN

                UPDATE usuario SET nivel='SF' WHERE
                id_usuario=NEW.corrector;
                RAISE NOTICE 'El usuario % ha sido promocionado a
                nivel   SF', usuarioRow.login;

        END IF;
```

```
 ELSIF NEW.id_fase='TF' THEN

        UPDATE pagina SET id_fase='TF' WHERE isbn=NEW.isbn
        AND n_pagina=NEW.n_pagina;
         UPDATE proyecto SET
        n_paginastf=proyectoRow.n_paginastf+1 WHERE
    isbn=OLD.isbn;
        SELECT INTO proyectoRow * FROM proyecto WHERE
        isbn=OLD.isbn;

        IF proyectoRow.n_paginas=proyectoRow.n_paginastf THEN

        UPDATE proyecto SET id_fase='TF' WHERE
    isbn=NEW.isbn;

        RAISE NOTICE 'Proyecto correspondiente al libro %,
        ha cambiado de la fase SF a TF.', NEW.isbn;

        END IF;



        SELECT INTO usuarioRow * FROM usuario WHERE
        usuario.id_usuario=NEW.corrector;

        UPDATE usuario SET
        n_paginastf=usuarioRow.n_paginastf+1 WHERE
    id_usuario=NEW.corrector;
        SELECT INTO usuarioRow * FROM usuario WHERE
        id_usuario=NEW.corrector;

        mensaje:='El libro con isbn '||NEW.isbn||' esta listo para
    ser cerrado.';
         INSERT INTO alerta VALUES(timestamp'2011-10-
    16',proyectoRow.coordinador,mensaje);


        SELECT INTO promocion valor FROM var_config WHERE
    variable='pag_promotf';

        IF usuarioRow.nivel='SF' AND
    usuarioRow.fecha<=timestamp'2011-10-16'-
interval'63 days'
        AND usuarioRow.n_paginastf=promocion THEN

        UPDATE usuario SET nivel='TF' WHERE
        id_usuario=NEW.corrector;
        RAISE NOTICE 'El usuario % ha sido promocionado a
        nivel TF', usuarioRow.login;
```

```
        END IF;

    END IF;

    END IF;


    RETURN NEW;

    END;

    $tgCheckCorreccion$ LANGUAGE plpgsql;

    CREATE TRIGGER tgCheckCorreccion AFTER UPDATE
    ON correccion FOR EACH ROW
    EXECUTE PROCEDURE tgCheckCorreccion();
```

- **Comprobaciones** :

    Para confirmar el correcto funcionamiento del trigger se ha generado un escenario de pruebas en el que ficticiamente un usuario corrige todas las páginas de un libro en una determinada fase. Al finalizar la correción se genera la notificación de que el proyecto correspondiente al libro corregido asciende a la siguiente fase.

    Asimismo se ha diseñado otro escenario en el que un usuario de un determinado nivel corrige páginas hasta cumplir los requisitos para ser promocionado al siguiente nivel.

3.2.3 Procedimiento A:

- **Análisis:**

    El procedimiento tiene que eliminar los usuarios que se registraron en un cierto intervalo de tiempo que son sólo correctores y no tienen ninguna página corrregida en ese intervalo.

Para obtener la selección de usuarios que cumplen estas características hemos obtenido el conjunto de los correctores que se registraron en ese tiempo y el conjunto de los usuarios que corrigieron páginas en ese intervalo. Teniendo estos dos conjuntos hemos seleccionado los usuarios que estan en el primero pero no en el segundo.

- **Código**:

```
CREATE OR REPLACE FUNCTION
borraUsuariosInactivos(integer) RETURNS void AS $$
DECLARE

        idUsuario integer;
        dias ALIAS FOR $1;
        contador integer;

BEGIN

IF dias<0 THEN
        RAISE EXCEPTION 'El parametro debe ser mayor o
        igual que 0';
END IF;

contador:=0;

FOR idUsuario IN SELECT id_usuario FROM usuario
        WHERE escolaborador=false AND escorrector=true
AND escoordinador=false
        AND usuario.fecha<=timestamp'2011-10-16'-
dias*interval '1 days'
        EXCEPT SELECT DISTINCT corrector FROM correccion
WHERE fecha_fin IS NOT null
        AND fecha_fin>=timestamp'2011-10-16'-
dias*interval '1 days' ORDER BY id_usuario
LOOP

        DELETE FROM usuario WHERE id_usuario=idUsuario;
        contador:=contador+1;
        RAISE NOTICE 'Usuario con id % usuarios
borrado',idUsuario;
END LOOP;

RAISE NOTICE '% usuarios borrados',contador;

END;
$$ LANGUAGE plpgsql;
```

- **Comprobaciones** :

La comprobación la hemos realizado de la siguiente manera, hemos dado un intervalo de tiempo muy grande (la fecha 2011-10-16 menos 4000 días) y hemos añadido un usuario que se registró en el sistema en 1999-11-25 sin corregir ninguna página hasta la fecha. El procedimiento al ejecutarse únicamente borra del sistema ese usuario.

3.2.4 Función B:

- **Análisis:**

  Esta función únicamente devuelve un integer según el argumento de entrada, si es 'F0' un 0, si es 'PF' un 1, etc. Esta información está situada en el tabla fase.

- **Código**:

```
CREATE OR REPLACE FUNCTION dameOrdenFase(text) RETURNS integer AS $$
    DECLARE

        idFase ALIAS FOR $1;
        infoFase fase%ROWTYPE;

    BEGIN

        SELECT INTO infoFase * FROM fase WHERE id_fase=idFase;

        RETURN infoFase.orden;

    END;
    $$ LANGUAGE plpgsql;
```

- **Comprobaciones** :

  Las verificación de esta función se hace en funcBtest que simplemente devuelve un entero según el argumento de entrada.

3.2.5 Función C:

- **Análisis:**

- **Código**:

- **Comprobaciones** :

3.2.6 Procedimiento D:

- **Análisis:**

  Este procedimiento tiene que notificar los libros que están estancados en el sistema en un intervalo de tiempo determinado. Para obtener esos libros se hacen dos selecciones: Aquellos libros que no están dados por finalizados y que cumplen la condición de tener una página corregida en ese intervalo de tiempo y aquellos que no están terminados y no están dentro del anterior conjunto.

- **Código**:

```
CREATE OR REPLACE FUNCTION alertaParado(integer)
RETURNS void  AS
$$
  DECLARE

        dias ALIAS FOR $1;
        isbnLibro text;
        mensaje text;
        userCoordinador integer;
        contador integer;

        BEGIN

             IF dias<0 THEN
                  RAISE EXCEPTION 'El parametro debe ser
            mayor o igual que 0';
        END IF;

        contador=0;


        FOR isbnLibro IN SELECT DISTINCT correccion.isbn
            FROM correccion,proyecto WHERE
            correccion.isbn=proyecto.isbn
            AND proyecto.isbn!='FI' AND fecha_fin IS NOT
            NULL EXCEPT
            SELECT DISTINCT correccion.isbn FROM
            correccion,proyecto WHERE
```

```
                correccion.isbn=proyecto.isbn
                AND proyecto.isbn!='FI'   AND fecha_fin IS NOT
        NULL
                 AND fecha_fin>=timestamp'2011-10-16'-
        90*interval'1 days'
                AND fecha_fin<=timestamp'2011-10-16'
        LOOP

                SELECT INTO userCoordinador coordinador
        FROM proyecto WHERE
isbnLibro=proyecto.isbn;

                mensaje:='El libro '||isbnLibro||' lleva
        estancado'  || ' más de '|| dias || ' dias.';

                INSERT INTO alerta(fecha,id_usuario,texto)
                VALUES (timestamp'2011-10-
                16',userCoordinador,mensaje);

                contador:=contador+1;

        END LOOP;

        RAISE NOTICE ' % libros estancados en el
sistema',contador;

END;
$$ LANGUAGE plpgsql;
```

- **Comprobaciones** :

  La verificacíon del procedimiento es simple, se elige un intervalo (por ejemplo 5 días) y se obtienen los libros que tienen una página con fecha_fin en ese lapso de tiempo y se compara con el resultado del conjunto contrario, es decir, los que no tienen ninguna página corregida en ese intervalo. Se sobreentiende que los libros finalizados no se tienen en cuenta para este cálculo.