Procedimientos, Funciones y Triggers

Ejercicios

Fuente: http://tisbddocs.dlsi.ua.es/inicio/guias/datos/sesiones#TOC-S14-Ingenier-a-inversa-de-BD-12---13-dic-2018-

1. Enunciados

P09.001- Crea con tu usuario las tablas siguientes (puedes usar CREATE TABLE ... LIKE ...):

yXXX_paciente (mismas columnas y filas que la tabla Hospital.paciente),
yXXX historial (mismas columnas y filas que la tabla Hospital.historial),
yXXX_linea_historial (mismas columnas y filas que la tabla Hospital.linea_historial),
yXXX_tiene_un (mismas columnas y filas que la tabla Hospital.tiene_un),
yXXX_trabajador (mismas columnas y filas que la tabla Hospital.trabajador),
yXXX_enfermero (mismas columnas y filas que la tabla Hospital.enfermero)
yXXX_medico (mismas columnas y filas que la tabla Hospital.medico)
yXXX_especialidad (mismas columnas y filas que la tabla Hospital.especialidad)
yXXX_auxiliar_m (colegiado varchar(9), CP: colegiado)
yXXX_auxiliar_h (colegiado varchar(9), CP: colegiado)

P09.002- Haz un procedimiento que sirva para dar de alta una especialidad nueva

P09.003- Haz un procedimiento que sirva para borrar una especialidad dada

P09.004- Haz un procedimiento que sirva para borrar todas las especialidades

P09.005- Haz un procedimiento que ponga como no activo a un paciente dado, sólo si no es de la Comunidad Valenciana

P09.006- Haz un procedimiento en el que dado como entrada el sip de un paciente, compruebe la última vez en la que dicho paciente estuvo ingresado. Si corresponde a una fecha del año anterior o anteriores, ponga dicho paciente como no activo

P09.007- Haz un procedimiento en el que todos aquellos trabajadores que no sean ni médicos ni enfermeros sean añadidos a la tabla yXXX_auxiliar_m, o a la tabla yXXX_auxiliar_h en función de su sexo

P09.008- Revisa el siguiente código. ¿Encuentras algún error? En caso afirmativo justifica la respuesta.

```
CREATE FUNCTION total_activos()
RETURNS int
DETERMINISTIC
BEGIN
DECLARE total int;
SELECT count(*) INTO total
FROM yXXX_trabajador;
END;
```

P09.009- Explica qué hace la siguiente función:

```
CREATE FUNCTION total_asignados_profesores(IN eldni char(9))
RETURNS decimal(4,2)
BEGIN
DECLARE total decimal(4,2)=0;
SET total= SELECT SUM(creditos) FROM imparte_teoria WHERE dni=eldni +
SELECT SUM(creditos) FROM imparte_practica WHERE dni=eldni;
RETURN total;
END;
```

P09.010- Indica cuándo se ejecutará el código del siguiente disparador:

```
CREATE TRIGGER activa_medico AFTER insert ON MEDICO
FOR EACH ROW
UPDATE trabajador SET active=1 WHERE colegiado=NEW.colegiado;
END;
```

P09.011- Indica cuándo se ejecutará el código del siguiente disparador:

```
CREATE TRIGGER activa_medico BEFORE delete ON MEDICO
FOR EACH ROW
UPDATE trabajador SET active=0 WHERE colegiado=OLD.colegiado;
END;
```

2. Soluciones

P09.001- Crea con tu usuario las tablas siguientes (puedes usar CREATE TABLE ... LIKE ...):

yXXX_paciente (mismas columnas y filas que la tabla Hospital.paciente),
yXXX historial (mismas columnas y filas que la tabla Hospital.historial),
yXXX_linea_historial (mismas columnas y filas que la tabla Hospital.linea_historial),
yXXX_tiene_un (mismas columnas y filas que la tabla Hospital.tiene_un),
yXXX_trabajador (mismas columnas y filas que la tabla Hospital.trabajador),
yXXX_enfermero (mismas columnas y filas que la tabla Hospital.enfermero)
yXXX_medico (mismas columnas y filas que la tabla Hospital.medico)
yXXX_especialidad (mismas columnas y filas que la tabla Hospital.especialidad)
yXXX_auxiliar_m (colegiado varchar(9), CP: colegiado)
yXXX_auxiliar_h (colegiado varchar(9), CP: colegiado)

```
CREATE TABLE yXXX_paciente as SELECT * from hospital.paciente;
CREATE TABLE yXXX_trabajador as SELECT * from hospital.trabajador;
CREATE TABLE yXXX_medico as SELECT * from hospital.medico;
CREATE TABLE yXXX_enfermero as SELECT * from hospital.enfermero;
CREATE TABLE yXXX_historial as SELECT * from hospital.historial;
CREATE TABLE yXXX_especialidad as SELECT * from hospital.especialidad;
CREATE TABLE yXXX_linea_historial as SELECT * from hospital.linea_historial;
CREATE TABLE yXXX_tiene_un as SELECT * from hospital.tiene_un;
CREATE TABLE yXXX_auxiliar_m (colegiado varchar(9), primary key (colegiado));
CREATE TABLE yXXX_auxiliar_h (colegiado varchar(9), primary key (colegiado));

DROP TABLE yXXX_tiene_un;
DROP TABLE yXXX_linea_historial;
DROP TABLE yXXX_historial;
DROP TABLE yXXX_especialidad;
DROP TABLE yXXX_paciente;
DROP TABLE yXXX_medico;
DROP TABLE yXXX_enfermero;
DROP TABLE yXXX_auxiliar_m;
DROP TABLE yXXX_auxiliar_h;
DROP TABLE yXXX_trabajador;
```

Solución alternativa:

```
CREATE TABLE yXXX_paciente LIKE hospital.paciente;
CREATE TABLE yXXX_trabajador LIKE hospital.trabajador;
CREATE TABLE yXXX_medico LIKE hospital.medico;
CREATE TABLE yXXX_enfermero LIKE hospital.enfermero;
CREATE TABLE yXXX_historial LIKE hospital.historial;
CREATE TABLE yXXX_especialidad LIKE hospital.especialidad;
CREATE TABLE yXXX_linea_historial LIKE hospital.linea_historial;
CREATE TABLE yXXX_tiene_un LIKE hospital.tiene_un;
```

P09.002- Haz un procedimiento que sirva para dar de alta una especialidad nueva

```
delimiter //
create procedure yXXX_alta_especialidad (in idesp int(10), in tipesp varchar(50))
  begin
    insert into yXXX_especialidad (id, tipo) values (idesp, tipesp);
  end;
//
delimiter ;

call yXXX_alta_especialidad(8, 'nuevapal');

select * from yXXX_especialidad; -- para comprobar que se ha insertado.
```

P09.003- Haz un procedimiento que sirva para borrar una especialidad dada

```
delimiter //
create procedure yXXX_borra_especialidad (in idesp int(10))
  begin
    delete from yXXX_especialidad where id=idesp;
  end;
//
delimiter ;

call yXXX_borra_especialidad(8);

select * from yXXX_especialidad; -- para comprobar que se ha borrado.
```

P09.004- Haz un procedimiento que sirva para borrar todas las especialidades

```
delimiter //
create procedure yXXX_borra_todo_especialidad ()
  begin
    delete from yXXX_especialidad;
  end;
//
delimiter ;

call yXXX_borra_todo_especialidad();

select * from yXXX_especialidad; -- para comprobar que se ha borrado.
```

P09.005- Haz un procedimiento que ponga como no activo a un paciente dado, sólo si no es de la Comunidad Valenciana

```
delimiter //
create procedure yXXX_desactiva_paciente (in sippac varchar(7))
  begin
    declare prov varchar(50);
    declare act tinyint(1);
    select provincia, activo into prov,act from yXXX_paciente where sip=sippac;
    if (prov <> 'ALICANTE') && (prov <> 'VALENCIA' ) && (prov <>'CASTELLON') then
       update yXXX_paciente set activo=0 where sip=sippac;
    end if;
  end;
//
delimiter ;

select sip, activo, provincia from yXXX_paciente where sip = '1000007'; --paciente de Huelva activo
call yXXX_desactiva_paciente('1000007');

select sip, activo, provincia from yXXX_paciente where sip = '1000007'; --paciente de Huelva desactivado
```

P09.006- Haz un procedimiento en el que dado como entrada el sip de un paciente, compruebe la última vez en la que dicho paciente estuvo ingresado. Si corresponde a una fecha del año anterior o anteriores, ponga dicho paciente como no activo

```
delimiter //
create procedure yXXX_desactiva_paciente2 (in sippac varchar(7))
  begin
    declare his  int(11);
    declare anyo int(5);
    select idhistorial into his from yXXX_tiene_un where idpaciente=sippac;
          select date_format(now(),'%Y')-date_format(max(fecha),'%Y')
into anyo from yXXX_linea_historial where idhistorial=his;
    if anyo >=1 then
         update yXXX_paciente set activo = 0 where sip=sippac;
  end if;
  end;
//
delimiter ;

select activo from yXXX_paciente where sip='1000035'; -- paciente activo
select idhistorial from yXXX_tiene_un where idpaciente='1000035'; -- vemos historial
select max(fecha) from yXXX_linea_historial where idhistorial= 718153; -- vemos fecha

call yXXX_desactiva_paciente2('1000035');

select activo from yXXX_paciente where sip='1000035'; -- paciente desactivado
```

P09.007- Haz un procedimiento en el que todos aquellos trabajadores que no sean ni médicos ni enfermeros sean añadidos a la tabla yXXX_auxiliar_m, o a la tabla yXXX_auxiliar_h en función de su sexo

```
delimiter //
create procedure yXXX_alta_auxiliar ()
  begin
    declare done bool default 0;
    declare col varchar(9);
    declare sex varchar(1);
    declare aux cursor for select colegiado,sexo from yXXX_trabajador where colegiado not in (select colegiado from medico) and colegiado not in (select colegiado from enfermero);
    declare continue handler for sqlstate '02000' set done=1;
    open aux;
    repeat
        fetch aux into col,sex;
        if not done then
       if sex='M' then
           insert into yXXX_auxiliar_m (colegiado) values (col);
       else
           insert into yXXX_auxiliar_h (colegiado) values (col);
       end if;
         end if;
    until done end repeat;
    close aux;
  end;
//
delimiter ;


select count(*) from yXXX_auxiliar_m; -- inicialmente tiene 0 filas
select count(*) from yXXX_auxiliar_h; -- inicialmente tiene 0 filas

select count(*),sexo
from yXXX_trabajador
where colegiado not in (select colegiado from medico)
and colegiado not in (select colegiado from enfermero); -- insertar 239 en h y 241 en m

call yXXX_alta_auxiliar();

select count(*) from yXXX_auxiliar_m -- 241 filas
select count(*) from yXXX_auxiliar_h -- 239 filas
```

P09.008- Revisa el siguiente código. ¿Encuentras algún error? En caso afirmativo justifica la respuesta.

```
CREATE FUNCTION total_activos()
RETURNS int
DETERMINISTIC
BEGIN
DECLARE total int;
SELECT count(*) INTO total
FROM yXXX_trabajador;
END;
```

Las funciones han de devolver un valor obligatoriamente. Falta, por tanto, el RETURN.

```
CREATE FUNCTION total_activos()
RETURNS int
DETERMINISTIC
BEGIN
    DECLARE total int;
    SELECT count(*) INTO total
        FROM yXXX_trabajador;
    RETURN total;
END;
```

P09.009- Explica qué hace la siguiente función:

```
CREATE FUNCTION total_asignados_profesores(IN eldni char(9))
RETURNS decimal(4,2)
BEGIN
DECLARE total decimal(4,2)=0;
SET total= SELECT SUM(creditos) FROM imparte_teoria WHERE dni=eldni +
SELECT SUM(creditos) FROM imparte_practica WHERE dni=eldni;
RETURN total;
END;
```

Calcula el total de créditos de teoría y de práctica que imparte un profesor dado, los suma y el valor resultante lo devuelve como salida de la función.

P09.010- Indica cuándo se ejecutará el código del siguiente disparador:

```
CREATE TRIGGER activa_medico AFTER insert ON MEDICO
FOR EACH ROW
UPDATE trabajador SET active=1 WHERE colegiado=NEW.colegiado;
END;
```

Después de cualquier operación de inserción de datos en la tabla medico.

P09.011- Indica cuándo se ejecutará el código del siguiente disparador:

```
CREATE TRIGGER activa_medico BEFORE delete ON MEDICO
FOR EACH ROW
UPDATE trabajador SET active=0 WHERE colegiado=OLD.colegiado;
END;
```

Antes de cualquier borrado que se realice sobre la tabla medico.
