Crear e executar procedementos almacenados

A tarefa consiste en escribir os guións de sentenzas SQL necesarios para crear procedementos almacenados atendendo a varios supostos, documentando os guións, e executando os procedementos creados.

- Tarefa 1.1. Crear un procedemento almacenado na base de datos *traballadores* que actualice a columna *depEmpregados* da táboa *departamento*, para todos os departamentos, contando o número de empregados que traballan nese departamento tendo en conta a información da columna *empDepartamento* da táboa *empregado*.
- Tarefa 1.2. Crear un procedemento almacenado que nos permita inserir datos de proba na táboa *vendas* na base de datos *tendaBD*.
  - O número de filas a inserir se lle pasa como un parámetro.
  - En cada fila, os datos para as columnas *ven_cliente*, *ven_tenda* e *ven_empregado* obtéñense buscando unha fila de maneira aleatoria nas táboas *clientes*, *tendas* e *empregados* respectivamente e collendo o código que corresponde.
  - A columna *ven_data* colle a data do sistema.
  - Nas columnas *ven_id* e *ven_factura* non se cargan datos. Na primeira porque é de tipo autoincremental e xa a calcula o servidor, e a segunda porque non se cubre ata que se facture a venda.

Solución

- Tarefa 1.1.

– Código do procedemento

```
use traballadores;
delimiter //
create procedure sp_actualizar_depEmpregados()
begin
   update departamento
     set depEmpregados = (select count(*)
                          from empregado
                          where empDepartamento = depNumero);
end
//
delimiter ;
```

– Execución e comprobación do funcionamento do procedemento

```
call sp_actualizar_depEmpregados();
```

Non produce ningunha saída en pantalla; unicamente informa na zona de saída de MySQL Workbench do número de filas modificadas. Para comprobar o correcto funcionamento do procedemento almacenado hai que consultar o contido da columna *depEmpregados* da táboa *departamento* e contrastar os valores dalgún departamento cos datos da táboa *empregado*.

– Consulta antes de executar o procedemento:

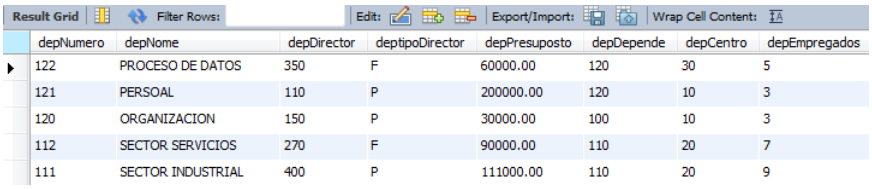
```
select * from departamento;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

depNumero	depNome	depDirector	deptipoDirector	depPresuposto	depDepende	depCentro	depEmpregados
122	PROCESO DE DATOS	350	F	60000.00	120	30	NULL
121	PERSOAL	110	P	200000.00	120	10	NULL
120	ORGANIZACION	150	P	30000.00	100	10	NULL
112	SECTOR SERVICIOS	270	F	90000.00	110	20	NULL
111	SECTOR INDUSTRIAL	400	P	111000.00	110	20	NULL

– Consulta despois de executar o procedemento:

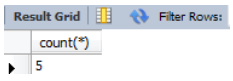
```
select * from departamento;
```



depNumero	depNome	depDirector	deptipoDirector	depPresuposto	depDepende	depCentro	depEmpregados
122	PROCESO DE DATOS	350	F	60000.00	120	30	5
121	PERSOAL	110	P	200000.00	120	10	3
120	ORGANIZACION	150	P	30000.00	100	10	3
112	SECTOR SERVICIOS	270	F	90000.00	110	20	7
111	SECTOR INDUSTRIAL	400	P	111000.00	110	20	9

– Consulta cantos empregados hai no departamento 122, na táboa empregado:

```
select count(*) from empregado where empDepartamento = 122;
```



count(*)
5

- Tarefa 1.2.

– Código do procedemento

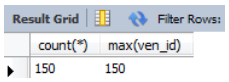
```
use tendaBD;
drop procedure if exists sp_inserir_vendas_proba;
delimiter //
create procedure sp_inserir_vendas_proba(pFilas integer)
begin
   declare vCliente, vEmpregado  smallint unsigned;
   declare vTenda tinyint unsigned;
   declare vCcontador tinyint unsigned default 0;
   while vCcontador < pFilas do
         /*seleccionar un empregado aleatoriamente*/
         select emp_id into vEmpregado
                from empregados
                order by rand()
                limit 1;
         /*seleccionar un cliente aleatoriamente*/
         select clt_id into vCliente
                from clientes
                order by rand()
                limit 1;
         /*seleccionar una tenda aleatoriamente*/
         select tda_id into vTenda
                from tendas
                order by rand()
                limit 1;
         /*inserir unha fila na táboa de vendas*/
         insert into vendas (ven_tenda,ven_empregado,ven_cliente,ven_data)
                values (vTenda, vEmpregado, vCliente, now());
         /*contar a fila nserida*/
         set vCcontador = vCcontador + 1;
   end while;
end

//
delimiter ;
```

– Execución e comprobación do funcionamento do procedemento. Pódese executar o procedemento pasándolle como parámetro o número de filas que se van a inserir e despois execútase unha consulta con SELECT para ver os datos inseridos. Se non se desexan conservar estas filas engadidas e son as únicas feitas na data actual, pódense borrar cunha sentenza DELETE.

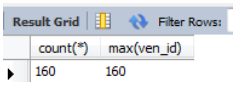
– Consulta do número de filas antes de executar o procedemento almacenado, e id da última venda.

```
select count(*), max(ven_id) from tendaBD.vendas;
```



– Execución do procedemento almacenado e consulta para saber se foron inseridas as filas.

```
call tendaBD.sp_inserir_vendas_proba(10);
select count(*), max(ven_id) from tendaBD.vendas;
```



– Borrado das filas inseridas na proba.

```
delete from vendas where ven_id between 151 and 160;
```

Crear e utilizar funcións definidas polo usuario

A tarefa consiste en escribir os guións de sentenzas SQL necesarios para crear funcións atendendo a varios supostos, e facer as probas de funcionamento utilizando as funcións creadas nunha consulta coa sentenza SELECT.

- Tarefa 1.3. Crear unha función na base de datos *utilidades* á que se lle pasa como parámetro o número do mes, e devolva o nome do mes en galego.
- Tarefa 1.4. Crear unha función na base de datos *utilidades* á que se lle pase como parámetro a nota numérica (dous enteiros e dous decimais) dun alumno, e devolva a nota en letra tendo en conta a seguinte táboa:

Nota numérica		Nota en letra
>= 0	< 5	suspenso
>= 5	< 6	aprobado
>= 6	< 7	ben
>= 7	< 9	notable
>= 9	<= 10	sobresaínte
Outro valor		erro na nota

Solución

- Tarefa 1.3.

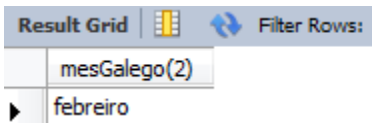
– Código da función

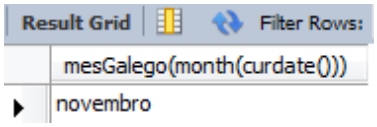
```
use utilidades;
drop function if exists mesGalego ;
delimiter //
create function mesGalego(pMes tinyint(2)) returns char(10)
  deterministic
  begin
    declare vMesLetra char(10) default null;
    case pMes
        when 1 then set vMesLetra="xaneiro";
        when 2 then set vMesLetra="febreiro";
        when 3 then set vMesLetra="marzo";
```

```
            when 4 then set vMesLetra="abril";
            when 5 then set vMesLetra="maio";
            when 6 then set vMesLetra="xuño";
            when 7 then set vMesLetra="xullo";
            when 8 then set vMesLetra="agosto";
            when 9 then set vMesLetra="setembro";
            when 10 then set vMesLetra="outubro";
            when 11 then set vMesLetra="novembro";
            when 12 then set vMesLetra="decembro";
        end case;
        return vMesLetra;
      end
    //
    delimiter ;
```

– Proba do funcionamento da función

```
select mesGalego(2); #febreiro
select mesGalego(month(curdate()));  #mes da data actual
```





- Tarefa 1.4.

– Código da función

```
use utilidades;
delimiter //
drop function if exists notaLetra //
create function notaLetra(pNota decimal(4,2))  returns char(20)
deterministic
begin
    declare vTexto char(20);
        if pNota >= 0 and pNota < 5 then set vTexto = 'suspenso';
         elseif pNota >= 5 and pNota < 6 then set vTexto = 'aprobado';
        elseif pNota >= 6 and pNota < 7 then set vTexto = 'ben';
        elseif pNota >= 7 and pNota < 9 then set vTexto = 'notable';
        elseif pNota >= 9 and pNota <= 10 then set vTexto = 'sobresaínte';
        else set vTexto = 'Erro na nota';
    end if;
    return vTexto;
end //
delimiter ;
```

– Proba do funcionamento da función

```
select notaLetra(0);    #suspenso
select notaLetra(1);    #suspenso
select notaLetra(5);    #aprobado
select notaLetra(6.9);  #ben
select notaLetra(8.5);  #notable
select notaLetra(10);   #sobresaínte
select notaLetra(11);   #Erro na nota
```