Tarefa 1. 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 co nome *vertaboas* na base de datos *utilidades*, que utilice a información contida na base de datos *information\_schema* para mostrar información das táboas que hai nas bases de datos. O procedemento recibe como parámetro de entrada unha cadea e texto que pode ser o nome dunha base de datos, ou ben, o carácter asterisco ('\*').
* Cando se lle pasa o carácter '\*' debe mostrar todas as táboas do servidor. Para cada táboa nos interesan as columnas: *table\_schema, table\_name, table\_type, engine, table\_rows* da táboa *information\_schema .tables*, ordenando o resultado polo nome do esquema e o nome da táboa.
* Cando se lle pasa o nome dunha base de datos, hai que comprobar que a base de datos existe na táboa *schemata*. No caso de existir, se mostrarán todas as táboas desa base de datos. Para cada táboa nos interesan as columnas: *table\_name, table\_type, engine, table\_rows* da táboa *information\_schema .tables*, ordenando o resultado polo nome da táboa. No caso de non existir a base de datos, se mostrará unha mensaxe de erro: ' A base de datos *xxxxx* non existe no servidor '.
* Tarefa 1.3. 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.
* Tarefa 1.4. Crear un procedemento almacenado na base de datos *tendaBD* que permita controlar os intentos de acceso errados dos usuarios da base de datos. Os parámetros de entrada son o *login* e *password* do usuario.

A táboa *usuario* garda información dos usuarios que poden acceder á base de datos, e terá o seguinte esquema:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| login | varchar(16) | N | P | Nome de usuario |
| password | char(40) | N |  | Contrasinal do usuario |

A táboa de *log\_erro\_conexion* rexistra os intentos de acceso errados, e terá o seguinte esquema:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| id | integer | N | P | Código autoincremental |
| login | varchar(16) | N |  | Nome de usuario |
| password | char(40) | N |  | Contrasinal do usuario |
| data\_hora | timestamp | N |  | Data e hora do intento de acceso |

O procedemento debe comprobar se existe na táboa *usuario* algún usuario co *login* e *password* que se pasan como parámetro. No caso de non existir, gárdase na táboa de rexistro *log\_erro\_conexion* a información correspondente ao intento de acceso errado. No caso de que o usuario faga máis de 5 intentos errados nos últimos 3 minutos, bloquearase a súa conta cambiándolle o contrasinal, poñendo unha contrasinal fixa establecida polo administrador, como por exemplo: 'H347B52(((]ERR'.

O procedemento utilizará un parámetro de saída para poder comprobar se o intento de acceso tivo éxito ou non. O parámetro ten o valor 0 se o *login* e o *password* corresponden a un usuario que existe na táboa; o valor 1 se o usuario non existe; o valor 2 no caso de bloqueo da conta por superar o número de intentos permitidos.

Solución

* Tarefa 1.1.
* Código do procedemento

/\*

u7a2tarefa0101.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: traballadores.sp\_actualizar\_depEmpregados (procedemento)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Contar número de empregados que hai en cada departamento

- Actualizar a columna depEmpregados da táboa departamento

co número de empregados que traballan no departamento.

PARAMETROS REQUERIDOS: - Non precisa parámetros

RESULTADOS PRODUCIDOS: - Columna depEmpregados actualizada

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**use** traballadores**;**

delimiter **//**

**create** **procedure** sp\_actualizar\_depEmpregados**()**

**begin**

**update** departamento

**set** depEmpregados **=** **(select** **count(\*)**

**from** empregado

**where** empDepartamento **=** depNumero**);**

**end**

**//**

delimiter **;**

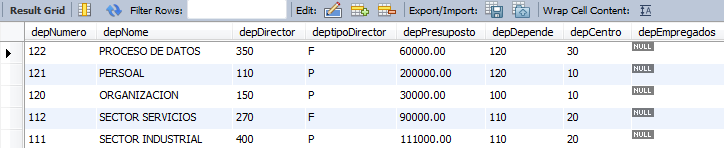
* Execución e comprobación do funcionamento do procedemento

**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*.

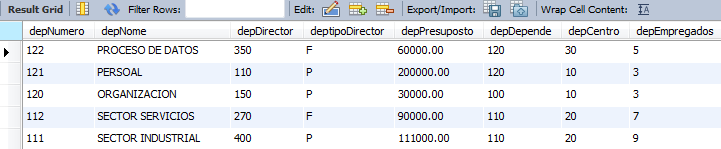
* Consulta antes de executar o procedemento:

**select** \* **from** departamento**;**



* Consulta despois de executar o procedemento:

**select** \* **from** departamento**;**



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

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



* Tarefa 1.2.
* Código do procedemento

/\*

u7a2tarefa0102.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: utilidades.vertaboas (procedemento)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Mostrar información resumida das táboas que hai nun

servidor, ou dunha base de datos concreta, tendo en conta a

información almacenada nas columnas table\_schema, table\_name

table\_name, table\_type, engine, table\_rows da táboa tables

da base de datos information\_schema. Os datos deben saír

ordenados polo nome da base de datos, e o nome da táboa.

PARAMETROS REQUERIDOS: - IN: pBaseDatos se lle poden pasar como valores válidos

o carácter \* que significa que se quere ver información das

táboas de todas as bases de datos, ou o nome dunha base de

datos no caso de querer ver información das táboas dunha

base de datos concreta. Calquera outro valor produce unha

mensaxe de erro. O tipo de dato do parámetro ten que ser

o mesmo que a columna table\_schema para poder comparalas.

RESULTADOS PRODUCIDOS: - Mostrar en pantalla a información solicitada

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

delimiter **//**

-- creación do procedemento usando un nome cualificado

**create** **procedure** utilidades**.**vertaboas**(**pBaseDatos **varchar(**64**)** **character** **set** utf8**)**

**begin**

**declare** existe **bit** **default** 0**;**

**if** pBaseDatos**=**'\*' **then**

**select** **concat(upper(**table\_schema**),**'.'**,lower(**table\_name**))** **as** `táboa`**,**

**lower(**table\_type**)** **as** tipo**,**

**lower(engine)** **as** motor**,**

table\_rows **as** filas**,**

create\_time **as** data\_creacion

**from** information\_schema**.**tables

**order** **by** `táboa`**;**

**else**

**select** **count(\*)** **into** existe

**from** information\_schema**.**SCHEMATA

**where** SCHEMA\_NAME**=**pBaseDatos**;**

**if** existe **=** 1 **then**

**select** table\_name **as** `táboa`**,**

**lower(**table\_type**)** **as** tipo**,**

**lower(engine)** **as** motor**,**

table\_rows **as** filas**,**

create\_time **as** data\_creacion

**from** information\_schema**.**tables

**where** table\_schema**=**pBaseDatos

**order** **by** `táboa`**;**

**else**

**select** **concat(**'A base de datos "'**,**pBaseDatos**,**'" non existe'**)** **as** Error**;**

**end** **if;**

**end** **if;**

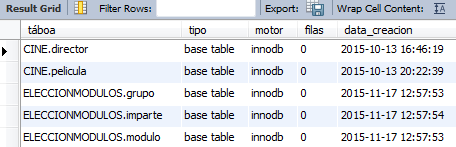
**end**

**//**

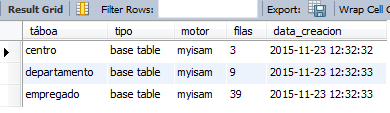
delimiter **;**

* Execución e comprobación do funcionamento do procedemento. Fanse dúas probas pasando dous valores válidos para o parámetro, e unha terceira proba cun valor que non é válido, para comprobar que se mostra a mensaxe de erro.

**call** utilidades**.**vertaboas**(**'\*'**);**



**call** utilidades**.**vertaboas**(**'traballadores'**);**



**call** utilidades**.**vertaboas**(**'pepiño'**);**



* Tarefa 1.3.
* Código do procedemento

/\*

u7a2tarefa0103.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: sp\_inserir\_vendas\_proba (procedemento)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Inserir datos de proba na táboa vendas da base de datos

tendaBD. 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, e collendo

o código que corresponde. A columna ven\_data colle a data do

sistema. Para as columnas ven\_id e ven\_factura non se cargan

datos.

PARAMETROS REQUERIDOS: - IN: pFilas - Indica o número de filas a inserir.

RESULTADOS PRODUCIDOS: - Non produce saída en pantalla. Insire na táboa vendas

filas con datos de proba. O nº de filas que se insiren

pásase como un parámetro de entrada.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**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.

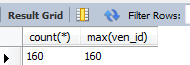
**select** **count(\*), max(**ven\_id**)** **from** tendaBD.vendas;



* 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;

* Tarefa 1.4.
* Código do procedemento

/\*

u7a2tarefa0104.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: sp\_erro\_login (procedemento)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Controlar intentos de acceso dos usuarios, comprobando se

o usuario que intenta acceder está na táboa de usuarios.

No caso de non existir o usuario na táboa se rexistra a

información do intento de acceso nunha táboa de rexistro.

No caso de que un usuario faga máis de 5 intentos errados nos

últimos 3 minutos se lle bloqueará a súa conta cambiándolle o

password por un valor fixo establecido polo administrador

PARAMETROS REQUERIDOS: - IN: pLogin: login do usuario.

- IN: pPassword: contrasinal do usuario

- OUT: pMensaxe: devolve o valor 0 se o login e password

corresponden a un usuario que existe na táboa de usuarios, o

valor 1 se o usuario non existe, e o valor 2 no caso en que

se lle cambie á password ao usuario por superar o número de

intentos permitidos.

RESULTADOS PRODUCIDOS: - Non mostra nada en pantalla, pero devolve os valores 0,1,2

no parámetro de saída.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**use** tendaBD**;**

**drop** **procedure** **if** **exists** sp\_erro\_login**;**

**delimiter** **//**

**create** **procedure** sp\_erro\_login **(**pUsuario **char(**16**),** pPalabra **char(**40**),** **out** pMensaxe **tinyint(**1**))**

**begin**

**declare** vIntentos **int;** /\*contador de intentos errados nos tres últimos minutos\*/

**declare** vUsuarioValido **boolean** **default** 0**;** /\*vale 1 cando usuario existe na táboa\*/

/\*Comprobación da existencia do usuario co login e password pasados como parámetro \*/

**select** **count(\*)** **into** vUsuarioValido

**from** usuario

**where** login**=**pUsuario **and** password**=**pPalabra**;**

/\*No caso de que non sexa correcta a conta de usuario rexístrase o intento errado en log\_erro\_conexion e cóntanse o número de intentos errados nos últimos 3 minutos\*/

**if** vUsuarioValido **=** 0 **then** /\*No caso de non existir o usuario\*/

**insert** **into** log\_erro\_conexion **(**login**,** password**)** **values** **(**pUsuario**,** pPalabra**);**

**select** **count(\*)** **into** vIntentos /\*Contar intentos nos últimos 3 minutos\*/

**from** log\_erro\_conexion

**where** login **=** pUsuario **and** timestampdiff**(minute,**data\_hora**,**now**())** **<=**3**;**

**if** vIntentos **<=** 5 **then**

**set** pMensaxe **=** 1**;**

**else**

**set** pMensaxe **=** 2**;**

**update** usuario **set** password **=** 'H347B52(((]ERR' **where** login **=** pUsuario**;**

**end** **if;**

**else** /\*No caso de existir o usuario\*/

**set** pMensaxe **=** 0**;**

**end** **if;**

**end**

**//**

**delimiter** **;**

* Execución e comprobación do funcionamento do procedemento. O primeiro para facer a comprobación é crear as táboas no caso de non existir, e dar de alta algún usuario. Para facer as probas se van a inserir usuarios co seu *password* sen cifrar, aínda que na práctica real a *password* dos usuarios debería gardarse cifrada utilizando para elo funcións que xa incorpora MySQL, como MD5, SHA1, ou SHA2.

**use** tendaBD**;**

**create** **table** **if** **not** **exists** usuario **(**

login **varchar(**16**),**

password **char(**40**),**

**primary** **key** **(**login**)**

**)engine** **=** innodb**;**

**create** **table** **if** **not** **exists** log\_erro\_conexion **(**

id **integer** unsigned auto\_increment **not** **null,**

login **varchar(**16**),**

password **char(**40**),**

data\_hora **timestamp** **default** now**(),**

**primary** **key** **(**id**)**

**)engine** **=** innodb**;**

**insert** **into** usuario **values** **(**'pepe'**,**'pepe'**);**

**insert** **into** usuario **values** **(**'pepa'**,**'pepa'**);**

Execútase o procedemento e mírase cal é o valor que devolve o parámetro de saída en cada execución. Primeiro pásanse como parámetros un *login* e un *pasword* dun usuario que exista, e compróbase que devolva o valor 0, e despois execútase 6 veces o procedemento pasando sempre o mesmo *login*, pero cunha *password* errónea. Despois dos seis intentos compróbase que o *password* do usuario foi modificado.

**call** sp\_erro\_login**(**'pepe'**,**'pepe'**,**@proba**);**

**select** @proba**;**

**call** sp\_erro\_login**(**'pepe'**,**'aa'**,**@proba**);**

**select** @proba**;**

**call** sp\_erro\_login**(**'pepe'**,**'ee'**,**@proba**);**

**select** @proba**;**

**call** sp\_erro\_login**(**'pepe'**,**'el'**,**@proba**);**

**select** @met**;**

**call** sp\_erro\_login**(**'pepe'**,**'es'**,**@met**);**

**select** @proba**;**

**select** **\*** **from** usuario**;**

**call** sp\_erro\_login**(**'pepe'**,**'ex'**,**@proba**);**

**select** @proba**;**

**call** sp\_erro\_login**(**'pepe'**,**'EX'**,**@proba**);**

**select** @proba**;**

**select** **\*** **from** log\_erro\_conexion**;**

**select** **\*** **from** usuario**;**

Tarefa 2. 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 2.1. 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 2.2. 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 |

* Tarefa 2.3. Crear unha función na base de datos *utilidades* que pasándolle como parámetro as 8 cifras correspondentes ao número do DNI, devolva a letra que lle corresponde.

A letra do DNI obtense mediante un algoritmo coñecido como módulo 23. O algoritmo consiste en dividir o número correspondente ao DNI entre 23 e obter o resto da división enteira. O resultado é un número comprendido entre o 0 e o 22. A cada un destes números se lles fai corresponder unha letra tendo en conta a seguinte táboa:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| T | R | W | A | G | M | Y | F | P | D | X | B | N | J | Z | S | Q | V | H | L | C | K | E |

Non se utilizan as letras: I, Ñ, O, e U. As letras I e O se descartan para evitar confusións con outros carácteres, como 1, l o 0.

* Tarefa 2.4. Crear unha función na base de datos *utilidades* que pasándolle como parámetro os 8 primeiros carácteres correspondentes ao Número de Identidade de Estranxeiro (NIE), devolva a letra que lle corresponde.

Utilízase o mesmo algoritmo que para o DNI, coa diferenza de que o NIE pode empezar por unha letra, polo que hai que engadirlle as seguintes restricións:

* No caso de que o NIE empece pola letra X, se calcula desprezando a X, e utilizando os 7 díxitos restantes.
* No caso de que o NIE empece pola letra Y, se substitúe a letra Y polo número 1.
* No caso de que o NIE empece pola letra Z, se substitúe a letra Z polo número 2.
* Tarefa 2.5. Crear unha función na base de datos *utilidades* á que se lle pasa como parámetro os 20 díxitos dunha conta bancaria española, e retorne como saída o IBAN que lle corresponde.

*O documento Cálculo do IBAN contén a explicación de como se calcula o IBAN para as contas bancarias de España, noutros países as contas bancarias poden ter ata 34 díxitos, e poden incluír letras.*

Solución

* Tarefa 2.1.
* Código da función

/\*

u7a2tarefa0201.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: mesGalego (función)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Obter o nome do mes en galego partindo do número do mes

PARAMETROS REQUERIDOS: - Número do mes

RESULTADOS PRODUCIDOS: - Nome do mes en galego

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**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 2.2.
* Código da función

/\*

u7a2tarefa0202.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: notaLetra (función)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Obter a nota en forma de texto partindo dunha nota numérica

PARAMETROS REQUERIDOS: - Cifra de dous enteiros e dous decimais correpondente á nota

RESULTADOS PRODUCIDOS: - Cadea de 20 carácteres coa nota en forma de texto

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**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**

* Tarefa 2.3.
* Código da función

/\*

u7a2tarefa0203.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: letraDni (función)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Obter a letra correspondente a un DNI a partir do algoritmo

coñecido como módulo 23

PARAMETROS REQUERIDOS: - Número enteiro, correspondente ao número dun DNI

RESULTADOS PRODUCIDOS: - Cadea de 1 carácter correspondente a letra do DNI

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**use** utilidades**;**

**create** **function** letraDni **(**pDni **integer)** **returns** **char(**1**)**

**deterministic**

**return** **substring(**'TRWAGMYFPDXBNJZSQVHLCKE'**,** pDni **%** 23 **+** 1**,** 1**);**

* Proba do funcionamento da función

**select** letraDni**(**33585123**);**



* Tarefa 2.4.
* Código da función

/\*

u7a2tarefa0204.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: letraNIE (función)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Obter a letra correspondente a un NIE a partir do algoritmo

coñecido como módulo 23, engadindo as restricións:

a) Se a primeira letra é unha X se despreza a primeira letra

b) Se a primeira letra é unha Y se substitúe polo número 1

c) Se a primeira letra é unha Z se substitúe polo número 2

d) Se empeza por calquera outro carácter devolve un cero

PARAMETROS REQUERIDOS: - Cadea de 8 carácteres, correspondentes a un NIE

RESULTADOS PRODUCIDOS: - Cadea de 1 carácter correspondente a letra do NIE ou un 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**use** utilidades**;**

**drop** **function** **if** **exists** letraNIE**;**

**delimiter** **//**

**create** **function** letraNIE**(**pNIE **char(**8**))** **returns** **char(**1**)** **deterministic**

**begin**

**declare** vBase **integer;**

**case** **left(**pNIE**,**1**)**

**when** 'X' **then** **set** vBase **=** **right(**pNIE**,**7**);**

**when** 'Y' **then** **set** vBase **=** **concat(**'1'**,right(**pNIE**,**7**));**

**when** 'Z' **then** **set** vBase **=** **concat(**'2'**,right(**pNIE**,**7**));**

**else** **return** '0'**;**

**end** **case;**

**return** **substring(**'TRWAGMYFPDXBNJZSQVHLCKE'**,** vBase **%** 23 **+** 1**,** 1**);**

**end**

**//**

**delimiter** **;**

* Proba do funcionamento da función

**select** letraNIE**(**'X7128990'**);**



**select** letraNIE**(**'Y0801462'**);**



**select** **if(**letraNIE**(**'30801462'**)=**0**,**'Erro no NIE'**,**letraNIE**(**'30801462'**))** **as** letraNIE**;**



* Tarefa 2.5.
* Código da función

/\*

u7a2tarefa0205.sql

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NOME RUTINA: calcularIBAN (función)

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR: - Obter o código internacional de conta bancaria (IBAN) para

contas en España. O algoritmo para o cálculo descríbese no

documento 'Cálculo de IBAN.pdf'

PARAMETROS REQUERIDOS: - Cadea de 20 carácteres que identifican unha conta bancaria

- Cadea de dous carácteres co código do país: ES - España

RESULTADOS PRODUCIDOS: - Cadea de 25 carácter correspondente ao IBAN da conta

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*/

**use** utilidades**;**

**drop** **function** **if** **exists** calcularIBAN**;**

**delimiter** **//**

**create** **function** calcularIBAN**(**pCCC **char(**20**),** pPais **char(**2**))** **returns** **char(**25**)** **charset latin1**

**deterministic**

**begin**

**declare** vBase **decimal(**30**,**0**);**

**declare** vControl **tinyint(**2**)** zerofill**;**

**set** vBase **=** **concat(**pCCC**,**locate**(left(**pPais**,**1**),**'ABCDEFGHIJKLMNOPQRSTUVWXYZ'**)+**9**,**

locate**(right(**pPais**,**1**),**'ABCDEFGHIJKLMNOPQRSTUVWXYZ'**)+**9**,**'00'**);**

**set** vControl **=** 98 **-** vBase **%** 97**;**

**return** **concat(**pPais**,**vControl**,**' '**,**pCCC**);**

**end**

**//**

**delimiter** **;**

* Proba do funcionamento da función

**select** calcularIBAN**(**'00120345030000067890'**,**'ES'**);**



**select** calcularIBAN**(**'90000001210123456789'**,**'ES'**);**



**select** calcularIBAN**(**'21000418450200051332'**,**'ES'**);**



Tarefa 3. Modificar procedementos almacenados e funcións

A tarefa consiste en facer modificacións nos seguintes procedementos almacenados e funcións:

* Tarefa 3.1. Cambiar as seguintes características do procedemento almacenado *utilidades.vertaboas()* creado na tarefa 1.2.:
* Se teñan en conta os privilexios do usuario que o executa (INVOKER).
* Engadir como comentario o texto: 'Mostra a información das táboas das bases de datos que se pasan como parámetro'.
* Tarefa 3.2. Borrar a función letraNIE, creada na tarefa 2.4.

Solución

Para facer os cambios pódese facer referencia á rutina cualificándoa co nome da base de datos, ou ben activar a base de datos. Nas solucións propostas móstranse as dúas opcións. A comprobación dos cambios pódese facer consultando o dicionario de datos. No caso de MySQL, as táboas *mysql.proc* ou *information\_schema.routines*.

* Tarefa 3.1

**alter** **procedure** utilidades**.**vertaboas

**comment** 'Mostra información das táboas das bases de datos que se pasan como parámetro'

**sql** security invoker**;**

* Tarefa 3.2

**use** utilidades**;**

**drop** **function** **if** **exists** letraNIE**;**