Dirección Xeral de Educación, Formación Profesional e Innovación Educativa

Material para

a formación profesional inicial

A02. Rutinas almacenadas

|  |  |  |
| --- | --- | --- |
| Familia profesional | IFC | Informática e comunicacións |
| Ciclo formativo | CSIFC03  CSIFC02 | Desenvolvemento de aplicacións web  Desenvolvemento de aplicacións multiplataforma |
| Grao |  | Superior |
| Módulo profesional | MP0484 | Bases de datos |
| Unidade didáctica | UD07 | Programación de bases de datos |
| **Actividade** | **A02** | **Rutinas almacenadas** |
| Autores |  | Marta Fernández García  María del Carmen Fernández Lameiro  Miguel Fraga Vila  María Carmen Pato González  Andrés del Río Rodríguez |
| Nome do arquivo |  | CSIFC02\_ MP0484\_V000702\_UD07\_A02\_Rutinas.docx |
| © 2015 Xunta de Galicia.  Consellería de Cultura, Educación e Ordenación Universitaria.  Este traballo foi realizado durante unha licenza de formación retribuída pola Consellería de Cultura, Educación e Ordenación Universitaria e ten licenza CreativeCommons BY-NC-SA (recoñecemento - non comercial - compartir igual). Para ver unha copia desta licenza, visitar a ligazón http://creativecommons.org/licenses/by-nc-sa/3.0/es/. | | |

Índice

[1. Ficha técnica 5](#_Toc441432564)

[Contexto da actividade 5](#_Toc441432565)

[Título da actividade 6](#_Toc441432566)

[Resultados de aprendizaxe do currículo 6](#_Toc441432567)

[Obxectivos didácticos e título e descrición da actividade 6](#_Toc441432568)

[Criterios de avaliación 6](#_Toc441432569)

[Contidos 6](#_Toc441432570)

[Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación 8](#_Toc441432571)

[2. A02. Rutinas almacenadas 10](#_Toc441432572)

[2.1 Introdución 10](#_Toc441432573)

[2.1.1 Obxectivos 10](#_Toc441432574)

[2.1.2 Software 10](#_Toc441432575)

[2.1.3 Bases de datos de traballo 11](#_Toc441432576)

[2.2 Actividade 11](#_Toc441432577)

[2.2.1 Rutinas almacenadas en MySQL 11](#_Toc441432578)

[2.2.1.1 Creación de rutinas almacenadas 11](#_Toc441432579)

[Sentenzas CREATE PROCEDURE e CREATE FUNCTION 12](#_Toc441432580)

[Exemplos 13](#_Toc441432581)

[2.2.1.2 Utilización de rutinas almacenadas 14](#_Toc441432582)

[Exemplos 14](#_Toc441432583)

[2.2.1.3 Tratamento de erros. Sentenza SIGNAL 15](#_Toc441432584)

[2.2.1.4 Documentación de programas almacenados 16](#_Toc441432585)

[2.2.1.5 Modificación de procedementos almacenados e funcións 17](#_Toc441432586)

[Sentenzas ALTER PROCEDURE e ALTER FUNCTION 17](#_Toc441432587)

[Sentenzas DROP PROCEDURE e DROP FUNCTION 17](#_Toc441432588)

[2.3 Tarefas 17](#_Toc441432589)

[2.3.1 Tarefa 1. Crear e executar procedementos almacenados 18](#_Toc441432590)

[Solución 19](#_Toc441432591)

[2.3.2 Tarefa 2. Crear e utilizar funcións definidas polo usuario 25](#_Toc441432592)

[Solución 25](#_Toc441432593)

[2.3.3 Tarefa 3. Modificar procedementos almacenados e funcións 29](#_Toc441432594)

[Solución 29](#_Toc441432595)

[3. Materiais 31](#_Toc441432596)

[3.1 Documentos de apoio ou referencia 31](#_Toc441432597)

[3.2 Recursos didácticos 31](#_Toc441432598)

[3.3 Material auxiliar 31](#_Toc441432599)

[4. Avaliación 32](#_Toc441432600)

[Criterios de avaliación 32](#_Toc441432601)

[Modelo de proba combinada para TO.2, TO.3, TO.4, TO.5, TO.6 e TO.10 32](#_Toc441432602)

[Proba para CA7.9.1 e CA7.12 34](#_Toc441432603)

[Proba para CA7.5.1, CA7.6, CA7.7, CA7.8, e CA7.12 34](#_Toc441432604)

[Exemplo de solución para entregar en papel 34](#_Toc441432605)

[Exemplo de lista de valoración para TO.2 37](#_Toc441432606)

[Exemplo de lista de valoración para TO.3 37](#_Toc441432607)

[Exemplo de lista de valoración para TO.4 37](#_Toc441432608)

[Exemplo de lista de valoración para TO.5 37](#_Toc441432609)

[Exemplo de lista de valoración para TO.6 38](#_Toc441432610)

[Exemplo de lista de valoración para TO.9 38](#_Toc441432611)

1. Ficha técnica

Contexto da actividade

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Módulo | Duración  horas | Unidade didáctica. | Sesións 50´ | Actividades | Sesións 50´ |
| MP0484. Bases de datos | 187 | UD01. Bases de datos e sistemas de almacenamento da información | 12 | A01. Bases de datos e sistemas de almacenamento da información | 7 |
| A02. Introdución aos sistemas xestores de bases de datos | 5 |
| UD02. Deseño conceptual de bases de datos | 38 | A01. Fases do deseño de bases de datos | 3 |
| A02.Descrición e representación gráfica dos elementos do Modelo Entidade-Interrelación (MER) | 10 |
| A03. Descrición e representación gráfica dos elementos do Modelo Entidade-Interrelación Estendido(MERE) | 9 |
| A04. Construción e validación do modelo de datos | 11 |
| A05. Notación Martin e ferramentas CASE | 5 |
| UD03. Deseño lóxico de base de datos | 32 | A01. O deseño lóxico empregando o modelo relacional | 15 |
| A02. Normalización de relacións | 10 |
| A03. Operacións do MR: álxebra e cálculo | 7 |
| UD04. Deseño físico de bases de datos | 28 | A01. Creación da estrutura de bases de datos relacionais | 13 |
| A02. Modificación da estrutura de bases de datos relacionais | 9 |
| A03. Verificación e probas | 3 |
| A04. Utilización de ferramentas gráficas para a creación da estrutura de bases de datos relacionais | 3 |
| UD05. Consultas de datos | 39 | A01. Consultas simples cunha táboa. | 12 |
| A02. Consultas con datos de máis dunha táboa | 10 |
| A03. Consultas resumo con agrupamentos | 7 |
| A04. Consultas con subconsultas | 7 |
| A05. Optimización de consultas | 3 |
| UD06. Tratamento de datos | 24 | A01. Manipulación de datos | 12 |
| A02. Vistas | 5 |
| A03. Transaccións e bloqueos | 7 |
| UD07. Programación de bases de datos | **22** | A01. Introdución á programación con SQL | 4 |
| **A02. Rutinas almacenadas** | **9** |
| A03. Disparadores e eventos | 5 |
| A04. Cursores | 4 |
| UD08. Administración de bases de datos | 19 | A01. Copias de seguridade e intercambio de datos entre SXBDs | 10 |
| A02. Índices e xestión de usuarios | 9 |
| UD09. Bases de datos obxecto-relacionais | 10 | A01. Uso de bases de datos obxecto-relacionais | 10 |

NOTA: Esta actividade está vinculada á programación recollida no arquivo CSIFC02\_MP0487\_V000700\_UD07\_ProgramacionBD.pdf

Título da actividade

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Título | Descrición | Duración |
| A02 | Rutinas almacenadas. | Crearanse procedementos almacenados e funcións definidas polo usuario, e utilizaranse para automatizar tarefas. | 9 |

Resultados de aprendizaxe do currículo

|  |  |
| --- | --- |
| Resultados de aprendizaxe do currículo | Completo |
| * RA7 - Desenvolve procedementos almacenados e guións de sentenzas, para o que utiliza e avalía as sentenzas da linguaxe incorporada no sistema xestor de bases de datos. | Si |

Obxectivos didácticos e título e descrición da actividade

| Obxectivos específicos | | Actividade | | Descrición básica | Duración |
| --- | --- | --- | --- | --- | --- |
| O2.1 | Crear, manter e executar procedementos almacenados. Tratar erros producidos na execución. | A02 | Rutinas almacenadas. | Crearanse procedementos almacenados e funcións definidas polo usuario e utilizaranse para automatizar tarefas. | 9 |
| O2.2 | Crear, manter e utilizar funcións de usuario. |
| O2.3 | Utilizar parámetros nas rutinas. |
| O2.4 | Utilizar funcións MySQL nas rutinas. |
| O2.5 | Documentar rutinas. |

Criterios de avaliación

|  |
| --- |
| Criterios de avaliación |
| * CA7.5 Creáronse, modificáronse e elimináronse procedementos almacenados.   CA 7.5.1 Creáronse, modificáronse, elimináronse e executáronse procedementos almacenados.   * CA7.6 Empregáronse parámetros no deseño de procedementos almacenados. * CA7.7 Detectáronse e tratáronse erros ao executar procedementos almacenados. * CA7.8 Usáronse as funcións proporcionadas polo sistema xestor. * CA7.9 Definíronse funcións de usuario.   CA7.9.1 - Creáronse, modificáronse, elimináronse e usáronse funcións de usuario.   * CA7.12 Documentáronse os guións codificados, e indicáronse as tarefas que automatizan e os resultados que producen. |

Contidos

| Contidos |
| --- |
| * Procedementos almacenados.   Creación, eliminación, modificación e execución de procedementos almacenados (CA 7.5.1).  Tratamento de erros producidos ao executar procedementos almacenados (CA 7.7).   * Paso de parámetros. * Funcións definidas polo usuario.   Creación, eliminación, modificación e utilización de funcións definidas polo usuario (CA 7.9.1).  Documentación (CA7.12). |

Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación

| Que e para que | Como | | | Con que | Como e con que se valora | Duración (sesións) |
| --- | --- | --- | --- | --- | --- | --- |
| Actividade (título e descrición) | Profesorado (en termos de tarefas) | Alumnado (tarefas) | Resultados  ou produtos | Recursos | Instrumentos e procedementos de avaliación |
| A02. Rutinas almacenadas   * Crearanse procedementos almacenados e funcións definidas polo usuario e utilizaranse para automatizar tarefas. | * **Tp2.1** Exposición sobre sintaxe de sentenzas SQL para crear, modificar, eliminar, executar e tratar erros de execución de procedementos almacenados. * **Tp2.2** Exposición sobre sintaxe de sentenzas SQL para crear, modificar, eliminar e utilizar funcións de usuario. * **Tp2.3** Exposición sobre utilización de funcións MySQL, parámetros e documentación das rutinas almacenadas. | * **Ta2.1** Creación, mantemento, documentación e execución ou utilización de rutinas almacenadas (procedementos almacenados e funcións de usuario) utilizando sentenzas SQL e ferramentas gráficas. | * Guións con procedementos almacenados e funcións de usuario. * Resultado da execución dos guións. | * Material didáctico subministrado polo profesorado en papel e/ou formato dixital. * Ordenadores con conexión a Internet, que terán instalado MySQL e MySQL Workbench. * Proxector. * Manual de referencia de MySQL |  | 7.5 |
|  | * **Ta2.2** Tarefa de avaliación combinada utilizando os instrumentos TO.2, TO.3, TO.4, TO.5, TO.6 e TO.9. | * Exame en papel e en formato dixital. | * Máquina virtual para exame que terá instalado MySQL e MySQL Workbench. * Manual de referencia de MySQL. | * **TO.2** Documento de rexistro de creación, modificación, eliminación e execución de procedementos almacenados. Escala de valores (observación indirecta). * **TO.3** Documento de rexistro de creación e execución de procedementos almacenados que utilicen parámetros. Escala de valores (observación indirecta). * **TO.4** Documento de rexistro de procedementos almacenados que traten erros producidos durante a execución dos mesmos. Escala de valores (observación indirecta). * **TO.5** Documento de rexistro de creación e execución de procedementos almacenados que utilicen funcións MySQL. Escala de valores (observación indirecta). * **TO.6** Documento de rexistro de creación, modificación, eliminación e utilización de funcións de usuario. Escala de valores (observación indirecta). * **TO.9** Documento de rexistro que mostre que os guións están documentados como mínimo coa indicación das tarefas que automatizan e os resultados que producen. Escala de valores (observación indirecta). | 1.5 |

1. A02. Rutinas almacenadas
   1. Introdución
      1. Obxectivos

Os obxectivos desta actividade son:

* Crear e manter rutinas almacenadas (procedementos almacenados e funcións de usuario). Utilizar parámetros.
* Utilizar funcións MySQL e rutinas almacenadas.
* Documentar e emendar erros nas rutinas almacenadas.
  + 1. Software

Utilizarase a plataforma WAMP (Windows-Apache-MySQL-PHP) WampServer 2.5 (última versión estable en outubro 2015), que inclúe MySQL Community Edition 5.6.17 como SXBDR (Sistema Xestor de Bases de Datos Relacional). As razóns de utilización deste software son que:

* É software libre, polo que o alumnado poderá descargalo de forma gratuíta e utilizalo legalmente na súa casa.
* É unha forma sinxela de facer a instalación do software necesario para desenvolver aplicacións web.
* Páxina oficial de  WampServer: <http://www.wampserver.com>
* Páxina oficial de MySQL: <https://www.mysql.com/>

Utilizarase MySQL Workbench 6.3 como ferramenta cliente gráfica xa que é a recomendada por MySQL en outubro de 2015, aínda que tamén poderían utilizarse outras como phpMyAdmin, EMS MyManager, ou MySQL Query Browser.

O resultado das probas dos guións SQL desta actividade, ilustrarase normalmente cunha imaxe capturada da zona de manipulación de datos de Workbench. Para completar esa información, por exemplo, cando non é posible mostrar o resultado completo, mostraranse outras zonas de Workbench, como por exemplo a zona de saída (*output*).

* En <https://www.mysql.com/products/workbench/> pode obterse información detallada sobre a ferramenta MySQL Workbench e descargar o software.
* En <http://dev.mysql.com/doc/index-gui.html> pode descargarse o manual de MySQL Workbench.
* O material anexo a esta actividade inclúe unha Guía básica de MySQL Workbench 6.3.
  + 1. Bases de datos de traballo

As bases de datos *practicas1*, *traballadores*, *utilidades* e *tendaBD,* utilizaranse nalgúns exemplos e tarefas desta actividade. Antes de empezar a probar os exemplos ou realizar as tarefas, hai que executar os scripts de creación no servidor e poñer en uso as bases de datos cando corresponda. Os scritps atópanse no cartafol anexo a esta actividade descrito no apartado '3.3 Material auxiliar'.

* 1. Actividade
     1. Rutinas almacenadas en MySQL

Unha rutina almacenada é un conxunto de sentenzas que poden ser almacenadas no servidor cun nome que se lle asigna no momento da creación. As aplicacións cliente poden executar as rutinas almacenadas, indicando o nome da rutina e pasándolle, opcionalmente, os parámetros necesarios. O termo rutinas almacenadas fai referencia tanto aos procedementos almacenados como ás funcións definidas polo usuario.

Cando se crea unha rutina almacenada, o conxunto de instrucións almacenadas no servidor queda enlazado e optimizado para a súa execución, o que supón unha mellora no rendemento e unha redución do tráfico na rede, aínda que a cambio supón un aumento de carga de traballo no servidor de bases de datos.

As rutinas almacenadas (procedementos e funcións) son soportadas por MySQL a partir da versión 5.0, e segue a sintaxe da norma ANSI SQL:2003 para procedementos almacenados, igual que IBM DB2.

* + - 1. Creación de rutinas almacenadas

O proceso de creación dunha rutina almacenada fai que quede asociado á base de datos que estea activa nese momento ou á que se indica explicitamente cun nome cualificado como *nomeBD.nomeRutina*. Isto ten varias implicacións:

* Cando se queiran utilizar o procedemento almacenado ou a función, ten que estar activa a base de datos asociada ou ben hai que cualificar o nome da rutina co nome da base de datos. Exemplo: *utilidades.calculoNnota(),* ou *utilidades.letraDNI()*, para facer referencia ás rutinas *calculoNota()* e *letraDNI()* que están na base de datos *utilidades*.
* Nunha base de datos non pode haber dúas rutinas almacenadas que teñan o mesmo nome.
* Cando se borra unha base de datos, se borran todos os procedemento almacenados e todas as funcións asociadas a esa base de datos.

No caso de existir un conxunto de procedementos almacenados e de funcións definidas polo usuario que podan ser de utilidade en máis dunha base de datos, pode ser útil ter creada unha base de datos cunha librería de procedementos almacenados e de funcións definidas polo usuario, en lugar de crear eses procedementos ou funcións en cada unha das bases de datos nas que se vai a utilizar.

Nesta actividade, utilizarase unha base de datos chamada *utilidades* cunha serie de táboas con sistemas de codificación (*provincias*, *países*, ...) e unha librería de rutinas almacenadas de uso xeral que poden ser utilizados dende calquera base de datos mediante nomes cualificados. Exemplo: *utilidades.calculo\_cif()*.

A información das rutinas creadas gárdase no dicionario de datos, igual có resto de obxectos das bases de datos. No caso de MySQL, a información sobre as rutinas almacenadas pódese consultar nas táboas *mysql.proc* e *information\_schema.routines*.

Sentenzas CREATE PROCEDURE e CREATE FUNCTION

Estas sentenzas permiten crear procedementos almacenados e funcións definidas polo usuario. Sintaxe:

CREATE [DEFINER = { usuario | CURRENT\_USER }]

PROCEDURE nome\_procedemento ( [parámetro\_procedemento [,...]])

[característica ...] corpo\_rutina

CREATE [DEFINER = { usuario | CURRENT\_USER }]

FUNCTION nome\_función ( [parámetro\_función [,...]])

RETURNS tipo\_dato

[característica ...] corpo\_rutina

* As cláusulas DEFINER e SQL SECURITY (forma parte de *característica*) especifican o contexto de seguridade no momento da execución da rutina. Con DEFINER pódese indicar o nome do usuario que vai ser considerado o creador da rutina. Se non se especifica nada, tómase CURRENT\_USER que fai referencia ao usuario actual que está creando a rutina.
* Os parénteses que van despois do nome do procedemento son obrigatorios e conteñen a lista de parámetros. Non se escribe nada dentro deles se o procedemento non ten parámetros.
* *parámetro\_procedemento* ten a forma:

[IN | OUT | INOUT] nome\_parámetro tipo\_dato

* IN, OUT e INOUT é o tipo de parámetro.

Un parámetro de entrada (IN - input) indica que cando se chame ao procedemento almacenado, hai que escribir nesa posición un valor que se corresponda co tipo de dato asociado a ese parámetro. O parámetro pode cambiar de valor durante a execución do procedemento, pero o seu valor non é visible para quen o chama.

Un parámetro de saída (OUT - output) indica que nesa posición hai que poñer unha variable de usuario que almacenará o resultado que devolverá o parámetro, para poder manexalo posteriormente. O seu valor inicial é null.

Un parámetro de entrada-saída (INOUT) indica que nesa posición hai que poñer unha variable de usuario que ten un valor inicial que se pasa como entrada cando se chama ao procedemento, e no que almacenará o resultado que devolverá o parámetro despois de executarse o procedemento.

No caso de chamar a un procedemento dende outro procedemento ou función, tamén se poden utilizar variables locais para os parámetros OUT e INOUT, en lugar de variables de usuario.

O procedemento almacenado pode devolver resultados mediante os parámetros OUT e INOUT.

Se non se indica IN, OUT ou INOUT, considérase que é de entrada (IN).

* O nome do parámetro pódese escribir en minúsculas ou maiúsculas indistintamente.
* O tipo de dato asociado do parámetro pode ser calquera tipo de dato válido en MySQL.
* *parámetro\_función* ten a forma:

nome\_parámetro tipo\_dato

O tipo de dato pode ser calquera tipo de dato válido en MySQL.

* A cláusula RETURNS só se pode utilizar na creación de funcións, onde é obrigatoria. Indica o tipo de dato que retorna a función. No corpo da función ten que existir unha sentenza RETURN para indicar o valor que retorna a función. Sintaxe da sentenza RETURN:

RETURN expresión

O tipo de dato da expresión ten que ser o mesmo có que se especifica na cláusula RETURNS.

Unha función só pode retornar un valor. No caso de necesitar unha rutina que devolva máis dun valor, utilizarase un procedemento almacenado con máis dun parámetro de saída (OUT).

* *característica* ten a forma:

LANGUAGE SQL | [NOT] DETERMINISTIC | SQL SECURITY { DEFINER | INVOKER } |

{ CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA } | COMMENT 'string'

* LANGUAGE SQL indica a linguaxe na que están escritas as sentenzas do corpo da rutina. Actualmente non se ten en conta porque só se permite o uso de sentenzas SQL, aínda que está previsto poder utilizar, nun futuro, sentenzas doutras linguaxes, como por exemplo PHP.
* Unha rutina considérase de tipo DETERMINISTIC se sempre devolve o mesmo valor de saída para os mesmos valores dos parámetros de entrada. Unha rutina que contén unha función NOW() ou unha función RAND() sería de tipo NOT DETERMINISTIC. Se non se especifica nada, o valor que toma o servidor por defecto é NOT DETERMINISTIC.
* SQL\_SECURITY permite indicar se na execución da rutina almacenada se utilizan privilexios do creador da rutina (DEFINER - valor por defecto) ou do usuario que a chama (INVOKER).
* Hai unha serie de características que fan referencia á natureza dos datos manexados pola rutina: CONTAINS SQL indica que a rutina non contén sentenzas que len ou escriben datos. NO SQL indica que a rutina non contén sentenzas SQL. READS SQL indica que contén sentenzas que len datos, por exemplo select, pero non sentenzas que escriben datos. MODIFIES SQL DATA indica que a rutina contén sentenzas que poden escribir datos, por exemplo insert ou update. O valor por defecto é CONTAINS SQL.
* COMMENT permite introducir un comentario que se verá ao executar a sentenza SHOW CREATE PROCEDURE ou SHOW CREATE FUNCTION.
* *corpo\_rutina* pode ser unha instrución ou un conxunto de instrucións SQL en forma de bloque de programación empezando por BEGIN e rematando en END. O corpo da rutina pode incluír sentenzas de declaración de variables, de asignación de valores, de control de fluxo, de manipulación de datos e a maioría das sentenzas da linguaxe SQL, e ademais, pode facer chamadas a outras rutinas almacenadas (procedementos e funcións definidas polo usuario).

Exemplos

* Exemplo de creación dun procedemento almacenado

**use** practicas1**;**

**drop** **procedure** **if** **exists** contarEmpregados **;**

**delimiter** **//**

**create** **procedure** contarEmpregados **(**pSexo **char(**1**),** **out** pContador **smallint)**

**begin**

**select** **count(\*)** **into** pContador

**from** practicas1**.**empregado

**where** sexo **=** **convert(**pSexo **using** utf8**) collate** utf8\_spanish\_ci**;**

**end**

**//**

**delimiter** **;**

O procedemento creado conta os empregados que hai na táboa *practicas1.empregado* que teñan na columna *sexo* o valor que se pasa como primeiro parámetro, e devolve o resultado nun parámetro de saída que aparece na segunda posición da lista de parámetros. A función *convert* converte o parametro ao sistema de colación adecuado, xa que a colación por defecto para utf8 é utf8\_general\_ci e o sistema de colación que utilizan as columanas da táboa é utf8\_spanish\_ci. Aos nomes dos parámetros se lles puxo ao inicio unha letra p para diferencialos dos nomes de columnas, aínda que isto so é unha recomendación.

Utilízanse as sentenzas BEGIN e END para identificar as sentenzas que forman o corpo do procedemento, aínda que non sería obrigatorio porque o corpo do procedemento só contén unha sentenza.

A sentenza DELIMITER cambia o delimitador de fin de sentenza mentres se crea o procedemento, e despois da creación vólvese a deixar o valor normal que é o carácter punto e coma (;).

* Exemplo de creación dunha función

**use** practicas1**;**

**create** **function** saudo**(**pEntrada **char(**20**))** **returns** **char(**27**)**

**deterministic**

**return** **concat(**'Hola, '**,**pEntrada**,**'!'**);**

A función recibe como parámetro de entrada unha cadea de carácteres de lonxitude 20, e retorna unha cadea de carácteres de lonxitude 27 formada pola concatenación do texto literal 'Hola ,' e a cadea recibida como parámetro de entrada. Hai que poñerlle a característica DETERMINISTIC, como a case todas asa funcións, pois o valor por defecto é NOT DETERMINISTIC.

Como o corpo da función ten só unha liña, non é obrigatorio utilizar as sentenzas END e BEGIN, e tampouco fai falta cambiar o delimitador de final de sentenza porque o remate da creación da función coincide co remate da sentenza que forma o corpo da función.

A función pódese utilizar como parte dunha expresión nunha sentenza SQL igual que as funcións que xa ten definidas MySQL.

* + - 1. Utilización de rutinas almacenadas

Para chamar a un procedemento almacenado hai que utilizar a sentenza CALL, mentres que unha función utilízase como calquera outra función das que xa ten definidas MySQL, engadíndoa a unha expresión que pode utilizarse dentro dunha sentenza SQL, como por exemplo, SELECT, INSERT, UPDATE ou DELETE. A función devolve sempre un valor no momento de avaliar a expresión.

Sintaxe da sentenza CALL:

CALL nome\_procedemento([parámetro [, ...]])

Exemplos

* Sentenzas para a execución do procedemento almacenado *contarEmpregados*:

**call** practicas1.contarEmpregados**(**'m'**,** @resultado**);**

**select** @resultado **as** numero\_empregados**;**

Resultado:



* Exemplo de uso da función *saudo*.

**select** practicas1.saudo**(**'Pepe'**);**

Resultado:



* Tarefa 1: Crear e executar procedementos almacenados.
* Tarefa 2: Crear e utilizar funcións definidas polo usuario.
  + - 1. Tratamento de erros. Sentenza SIGNAL

A sentenza SIGNAL devolve un erro como resultado da execución dun programa almacenado. Sintaxe:

SIGNAL valor\_condición\_erro

[SET nome\_información = valor\_información ...]

* *valor\_condición\_erro* indica o valor de erro que se vai a devolver. Pode ser un valor SQLSTATE (unha cadea de 5 carácteres), ou un nome de condición, que ten que estar declarada previamente. Os dous primeiros carácteres de SQLSTATE indican a clase de erro que pode ser un erro grave que produce a finalización da sentenza, unha advertencia (*warning*) ou un erro de tipo *'not found'*. O valor SQLSTATE para unha sentenza SIGNAL non debe empezar nunca por '00' porque eses valores corresponden a operacións rematadas con éxito. Para sinalar un valor SQLSTATE xenérico pódese utilizar o valor '45000' que significa 'excepción non controlada definida polo usuario'.
* A sentenza pode incluír, de maneira opcional, unha cláusula SET cunha lista de informacións asociadas á sinal de erro. Esta lista ten a forma:

nome\_información = valor\_información

Algúns dos *nomes\_información* máis empregados son:

* MESSAGE\_TEXT = 'texto da mensaxe que se devolve'.
* MYSQL\_ERRNO = número de erro MySQL que se devolve.

Exemplo de utilización:

**drop** **procedure** **if** **exists** demoSignal**;**

delimiter **//**

**create** **procedure** demoSignal**(**pValor **tinyint)**

**begin**

**declare** especialidad condition **for** **sqlstate** '45000'**;**

**if** pValor **=** 0 **then**

signal **sqlstate** '01000'**;**

elseif pValor **=** 1 **then**

signal **sqlstate** '45000' **set** message\_text **=** 'Ocorreu un erro (1)'**;**

elseif pValor **=** 2 **then**

signal especialidad **set** message\_text **=** 'Ocorreu un erro (2)'**;**

**else**

signal **sqlstate** '01000'

**set** message\_text **=**'Ocorreu unha advertencia'**,** mysql\_errno **=** 1000**;**

signal **sqlstate** '45000'

**set** message\_text **=** 'Ocorreu unha advertencia'**,** mysql\_errno **=** 1001**;**

**end** **if;**

**end;**

**//**

delimiter **;**

En función do valor que se pase como parámetro, o procedemento devolve unha condición de erro:

* No caso de que *pValor* tome o valor 0, devólvese unha advertencia (*warning*), porque o valor SQLSTATE empeza por '01'.



* No caso de que *pValor* tome o valor 1, devólvese un erro e unha mensaxe . O erro provoca que termine a execución do procedemento e móstrase a mensaxe coa información do erro.



* No caso de que *pValor* tome o valor 2, devólvese o mesmo erro pero asociado ao nome de condición.



* No caso de que *pValor* tome calquera outro valor, primeiro devólvese unha advertencia que non supón o final da execución do procedemento e execútase a seguinte sentenza que devolve: un erro, a mensaxe coa información do erro, o código do erro, e ademais, provoca que termine a execución do procedemento.



* + - 1. Documentación de programas almacenados

É moi recomendable documentar o código co obxecto de facilitar as labores de mantemento dos programas almacenados (facer modificacións nos programas almacenados debido á aparición de novas necesidades, novas normas, ...).

A documentación do código pódese incluír nos propios guións, inserindo liñas de comentarios que conteñan informacións referentes ao autor, a data de creación, tarefas que automatizan, parámetros que necesita, e resultados que producen.

Cando se fan cambios no código tamén se deberían inserir liñas de comentarios que conteñan a data da modificación, autor, cambios realizados, e causas polas que se fan os cambios.

Cando se utiliza MySQL Workbench para editar os guións de creación de programas almacenados pódese crear un fragmento de código (*snippet*) e gardalo para utilizalo cada vez que se crea un novo ficheiro coa creación ou modificación de programas almacenados. Para máis información sobre snippets, pódese consultar a Guía básica de MySQL Workbench 6.3 anexionada a esta actividade. Exemplo de contido do snippet *docu\_creaRutina*:

/\*

nome\_arquivo.sql

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

NOME RUTINA:

DATA CREACIÓN:

AUTOR:

TAREFA A AUTOMATIZAR:

PARAMETROS REQUERIDOS:

RESULTADOS PRODUCIDOS:

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

\*/

* + - 1. Modificación de procedementos almacenados e funcións

Os procedementos e función existentes pódense modificar empregando as sentenzas ALTER e DROP.

Sentenzas ALTER PROCEDURE e ALTER FUNCTION

Esta sentenza permite cambiar as características dunha rutina almacenada (procedemento ou función). Na mesma sentenza pódese cambiar máis dunha característica. Sen embargo, non se permite facer cambios nos parámetros nin no corpo da rutina, para facer estes cambios é necesario borrar a rutina almacenada e volvela a crear. Sintaxe :

ALTER {PROCEDURE | FUNCTION} nome\_rutina [características ...]

As características que se poden modificar son algunhas das que se viron na sintaxe da orden de creación de rutinas almacenadas:

LANGUAGE SQL | SQL SECURITY { DEFINER | INVOKER } |

{ CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA } | COMMENT 'string'

* LANGUAGE SQL indica a linguaxe na que están escritas as sentenzas do corpo da rutina. Actualmente non se ten en conta porque só se permite o uso de sentenzas SQL, aínda que está previsto poder utilizar, nun futuro, sentenzas doutras linguaxes, como por exemplo PHP.
* SQL\_SECURITY permite indicar se na execución da rutina almacenada se utilizan privilexios do creador da rutina (DEFINER - valor por defecto) ou do usuario que a chama (INVOKER).
* Características que fan referencia á natureza dos datos manexados pola rutina almacenada: CONTAINS SQL indica que a rutina non contén sentenzas que len ou escriben datos. NO SQL indica que a rutina non contén sentenzas SQL. READS SQL indica que a rutina contén sentenzas que len datos, por exemplo *select*, pero non sentenzas que escriben datos. MODIFIES SQL DATA indica que a rutina contén sentenzas que poden escribir datos, por exemplo *insert* ou *update*. O valor por defecto é CONTAINS SQL.
* COMMENT permite introducir un comentario que se verá ao executar a sentenza SHOW CREATE PROCEDURE ou SHOW CREATE FUNCTION.

Sentenzas DROP PROCEDURE e DROP FUNCTION

Estas sentenzas permiten borrar procedementos almacenados. Sintaxe:

DROP {PROCEDURE | FUNCTION} [IF EXISTS] nome\_rutina

A opción IF EXISTS comproba primeiro se existe a rutina almacenada, e só fai o borrado no caso de que exista, evitando así que se produza un erro grave que faga que se corte a execución dun guión de sentenzas e dá lugar só a unha mensaxe de advertencia (*warning*).

* Tarefa 3. Modificar procedementos almacenados e funcións.
  1. Tarefas

As tarefas propostas son as seguintes:

* Tarefa 1. Crear e executar procedementos almacenados.
* Tarefa 2. Crear e utilizar funcións definidas polo usuario.
* Tarefa 3. Modificar procedementos almacenados e funcións.
  + 1. 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: 9/11/2015

AUTOR: Grupo licenza 2015

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 **;**

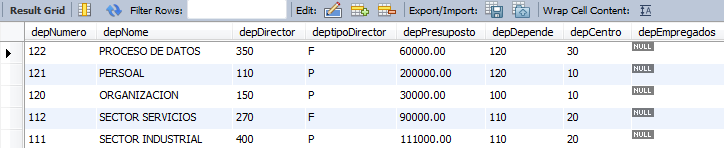
* Execución e comprobación do funcionamento do procedemento

**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*.

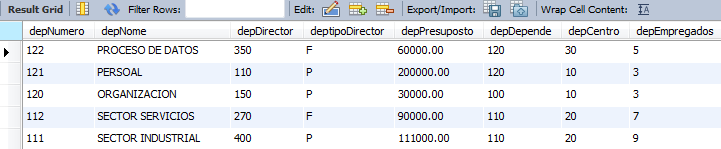
* Consulta antes de executar o procedemento:

**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: 9/11/2015

AUTOR: Grupo licenza 2015

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;**

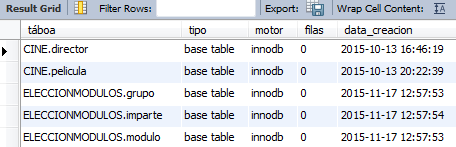
**end**

**//**

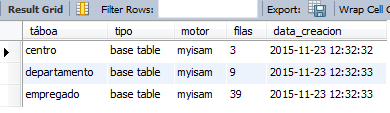
delimiter **;**

* 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: 9/11/2015

AUTOR: Grupo licenza 2015

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.

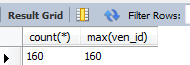
**select** **count(\*), max(**ven\_id**)** **from** tendaBD.vendas;



* 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: 9/11/2015

AUTOR: Grupo licenza 2015

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**;**

* + 1. 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.
* TA1. 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: 9/11/2015

AUTOR: Grupo licenza 2015

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: 9/11/2015

AUTOR: Grupo licenza 2015

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: 9/11/2015

AUTOR: Grupo licenza 2015

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: 9/11/2015

AUTOR: Grupo licenza 2015

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: 9/11/2015

AUTOR: Grupo licenza 2015

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'**);**



* + 1. 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**;**

1. Materiais
   1. Documentos de apoio ou referencia

* HUESO IBAÑEZ, Luis. *Bases de datos*. Ciclos Formativos Ra-Ma, 2012.
* SANCHEZ, Jorge. *Manual de Gestión e bases de datos*.

<http://www.jorgesanchez.net/bd/sgbd.html>

* *Bases de datos. Máster oficial de Software libre*. UOC.

<http://ocw.uoc.edu/informatica-tecnologia-y-multimedia/bases-de-datos/Course_listing>

* *Manual de referencia de MySQL 5.6*. MySQL. <http://dev.mysql.com/doc/>
* GROFF, James.*SQL.*McGraw-Hill.
* VIQUEZ, Pablo. *Mi vida y cosas relacionadas*. Pablo Viquez Blog.

<http://www.pabloviquez.com/2009/05/illegal-mix-of-collations%E2%80%A6/>

* 1. Recursos didácticos
* Ordenadores con conexión a Internet, que terán instalado o sistema xestor de bases de datos MySQL e o cliente MySQL Workbench.
* Material didáctico subministrado polo profesorado en papel e/ou formato dixital.
* Máquina virtual para exame que terá instalado MySQL e MySQL Workbench.
* Manual de referencia de MySQL.
* Proxector.
  1. Material auxiliar

O material auxiliar anexo a esta actividade está almacenado na carpeta CSIFC02\_MP0484\_V000702\_UD07\_A02\_Rutinas\_Anexos que contén:

* O arquivo GuiaWorkbench.docx que é unha guía básica de MySQL Workbench 6.3.
* O arquivo V000702\_scriptsSQL.zip cos scripts necesarios para crear as bases de datos empregadas nos exemplos, tarefas e proba de avaliación da actividade.
* O arquivo calculoIBAN.pdf co texto de apoio TA1.: Cálculo do IBAN. O documento para a *'Definición e adaptación do Código internacional de conta bancaria (IBAN) para contas en España*' foi descargado da páxina de oficial de Kutxabank, S.A. no seguinte [enlace](https://kutxa.kutxabank.es/cs/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=content-type&blobheadername2=Expires&blobheadername3=MDT-Type&blobheadername4=Content-disposition&blobheadervalue1=application%2Fpdf&blobheadervalue2=Thu%2C+10+Dec+2020+16%3A00%3A00+GMT&blobheadervalue3=abinary%3Bcharset%3DUTF-8&blobheadervalue4=inline%3B+filename%3D%22c%C3%A1lculo+de+iban.pdf%22&blobkey=id&blobtable=MungoBlobs&blobwhere=1311844840805&ssbinary=true).

1. Avaliación

Criterios de avaliación

|  |  |  |  |
| --- | --- | --- | --- |
| Criterios de avaliación seleccionados  para esta actividade | Tipo de  evidencia de aprendizaxe | Instrumento de avaliación | Peso na cualificación  da UD |
| * CA7.5 Creáronse, modificáronse e elimináronse procedementos almacenados.   CA7.5.1 - Creáronse, modificáronse, elimináronse e executáronse procedementos almacenados. | * Exame en formato dixital e en papel. | * TO.2Documento de rexistro de creación, modificación, eliminación e execución de procedementos almacenados. Escala de valores (observación indirecta). | 15 |
| * CA7.6 Empregáronse parámetros no deseño de procedementos almacenados. | * TO.3 Documento de rexistro de creación e execución de procedementos almacenados que utilicen parámetros. Escala de valores (observación indirecta). | 10 |
| * CA7.7 Detectáronse e tratáronse erros ao executar procedementos almacenados. | * TO.4 Documento de rexistro de procedementos almacenados que traten erros producidos durante a execución dos mesmos. Escala de valores (observación indirecta). | 5 |
| * CA7.8 Usáronse as funcións proporcionadas polo sistema xestor. | * TO.5 Documento de rexistro de creación e execución de procedementos almacenados que utilicen funcións MySQL. Escala de valores (observación indirecta). | 5 |
| * CA7.9 Definíronse funcións de usuario.   CA7.9.1 - Creáronse, modificáronse, elimináronse e usáronse funcións de usuario. | * TO.6 Documento de rexistro de creación, modificación, eliminación e utilización de funcións de usuario. Escala de valores (observación indirecta). | 15 |
| * CA7.12 Documentáronse os guións codificados, e indicáronse as tarefas que automatizan e os resultados que producen. | * TO.9 Documento de rexistro que mostre que os guións están documentados como mínimo coa indicación das tarefas que automatizan e os resultados que producen. Escala de valores (observación indirecta). | 8 |

Modelo de proba combinada para TO.2, TO.3, TO.4, TO.5, TO.6 e TO.10

Proponse unha proba de dous apartados de tipo práctico. A solución deberá ser entregada polo alumnado nun documento escrito e nun arquivo dixital.

* O documento escrito debe conter:
* Código da rutina almacenada solicitada.
* Sentenza ou sentenzas que permitan utilizar ou executar a rutina anterior.
* Imaxe capturada co resultado da execución da rutina almacenada dende o cliente Workbench.
* O arquivo dixital debe conter:
* Arquivo .sql co código da rutina almacenada.

A proba realizarase na base de datos *liga\_acb*. Subminístrase o arquivo *ligaAcb.sql* co script que permite crear esa base de datos. A estrutura da base de datos é a seguinte:

* Táboa *equipo*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| codigo | char(4) |  |  | Código do equipo. Para enlazar con outras táboas |
| nome\_club | varchar(80) |  |  | Nome do club ao que pertence o equipo |
| pavillon | varchar(70) |  |  | Nome do pavillón no que xogan os partidos como local |
| capacidade\_pavillon | smallint unsigned |  |  | Número de espectadores que entran no seu pavillón |
| web | varchar(100) |  |  | Url da web |
| nome\_equipo | varchar(80) |  |  | Nome oficial do equipo na liga |

* Táboa *xogador*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| codigo\_equipo | char(4) |  |  | Código do equipo no que xoga o xogador |
| numero | tinyint unsigned |  |  | Número de dorsal do xogador |
| nome | char(30) |  |  | Nome do xogador |
| posicion | char(1) |  |  | Posición na que xoga: B = base, P = pivot, ...... |
| nacionalidade | char(3) |  |  | Nacionalidade do xogador |
| ficha | char(3) |  |  | Tipo de ficha: EXT=estranxeiro EUR=europeo JFL=nacional |
| estatura | Decimal(3,2) |  |  | Estatura do xogador en metros |
| idade | tinyint unsigned |  |  | Idade do xogador |
| temporadas | tinyint |  |  | Número de tempadas que leva no equipo |

* Táboa *partido*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| id | smallint unsigned | N | **P** | Un identificador para cada partido. AUTOINCREMENTAL |
| xornada | smallint unsigned |  |  | Número da xornada na que se xoga o partido |
| equipo\_local | char(4) |  |  | Código do equipo que xoga como local |
| equipo\_visitante | char(4) |  |  | Código do equipo que xoga como visitante |
| puntos\_local | tinyint unsigned |  |  | Puntos conseguidos polo equipo local no partido |
| puntos\_visitante | tinyint unsigned |  |  | Puntos conseguidos polo equipo visitante no partido |

* Táboa *estatistica*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| equipo | char(4) |  |  | Código do equipo |
| xogados\_local | tinyint unsigned |  |  | Número de partidos xogados como local |
| xogados\_visitante | tinyint unsigned |  |  | Número de partidos xogados como visitante |
| ganados\_local | tinyint unsigned |  |  | Número de partidos ganados como local |
| ganados\_visitante | tinyint unsigned |  |  | Número de partidos ganados como visitante |
| puntos\_favor\_local | smallint |  |  | Número de puntos conseguidos xogando como local |
| puntos\_favor\_visitante | smallint |  |  | Número de puntos conseguidos xogando como visitante |
| puntos\_contra\_local | smallint |  |  | Número de puntos recibidos xogando como local |
| puntos\_contra\_visitante | smallint |  |  | Número de puntos recibidos xogando como visitante |

* Táboa *clasificacion*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| posicion | tinyint unsigned |  |  | Nº de orde que ocupa o equipo na clasificación |
| equipo | char(4) |  |  | Código do equipo |
| xogados | tinyint unsigned |  |  | Número de partidos xogados |
| ganados | tinyint unsigned |  |  | Número de partidos ganados |
| puntos\_favor | smallint |  |  | Número de puntos conseguidos en total |
| puntos\_contra | smallint |  |  | Número de puntos recibidos en total |

Proba para CA7.9.1 e CA7.12

* 1.- Escribir o código da función *ficha\_texto* á que se lle pasa como parámetro o tipo de ficha dun xogador, e devolve o texto correspondente ao valor do parámetro segundo a seguinte táboa. A función debe borrarse se xa existe. Escribir a sentenza que permite probar a función. Documentar o guión indicando a lo menos as tarefas que automatizan e os resultados que producen.

|  |  |
| --- | --- |
| Valor | Texto |
| EXT | Xogador estranxeiro |
| EUR | Xogador europeo |
| JFL | Xogador nacional |
| COT | Trámites pendentes de confirmación |
| Calquera outro valor | Erro: Non existe ese tipo de ficha |

Proba para CA7.5.1, CA7.6, CA7.7, CA7.8, e CA7.12

* 2.- Escribir o procedemento almacenado *estatisticos* ao que se lle pasa como parámetro o código dun equipo e mostra a información seguinte. O procedemento debe borrarse se existe. Escribir a sentenza que permite executar o procedemento. Documentar o guión indicando a lo menos as tarefas que automatizan e os resultados que producen.
* Se o equipo existe, mostrará as seguintes estatísticas dese equipo:

Número de partidos xogados como local

Número de partidos xogados como visitante

Número de partidos ganados como local

Número de partidos ganados como visitante

Número de puntos conseguidos xogando como local

Número de puntos conseguidos xogando como visitante

Número de puntos recibidos xogando como local

Número de puntos recibidos xogando como visitante

* Se o equipo non existe, o procedemento debe provocar un erro, deter a execución e mostrar a mensaxe: "Código de equipo non existe".

Exemplo de solución para entregar en papel

* Solución ao apartado 1
* O código da función podería ser o seguinte:

/\*

funcion\_ficha\_texto.sql

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

NOME RUTINA: ficha\_texto (función)

DATA CREACIÓN: 18/11/2015

AUTOR: Grupo licenza 2015

TAREFA A AUTOMATIZAR: - devolve o texto correspondente ao valor do tipo de ficha:

Valor Texto

EXT Xogador estranxeiro

EUR Xogador europeo

JFL Xogador nacional

COT Trámites pendentes de confirmación

Calquera outro valor Erro: Non existe ese tipo de ficha

PARAMETROS REQUERIDOS: - Tipo de ficha dun xogador

RESULTADOS PRODUCIDOS: - Texto correspondente segundo a relación anterior

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

\*/

**use** liga\_acb**;**

**drop** **function** **if** **exists** ficha\_texto**;**

delimiter **//**

**create** **function** ficha\_texto**(**pFicha **char(**3**))**

**returns** **varchar(**50**)**

**deterministic**

**begin**

**declare** vTexto **varchar(**50**);**

**case** pFicha

**when** 'EXT' **then** **set** vTexto **=** 'Xogador estranxeiro'**;**

**when** 'EUR' **then** **set** vTexto **=** 'Xogador europeo'**;**

**when** 'JFL' **then** **set** vTexto **=** 'Xogador nacional'**;**

**when** 'COT' **then** **set** vTexto **=** 'Trámites pendentes de confirmación'**;**

**else** **set** vTexto **=** 'Erro: Non existe ese tipo de ficha'**;**

**end** **case;**

**return** vTexto**;**

**end**

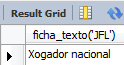
**//**

delimiter **;**

* Unha sentenza moi sinxela para probar o funcionamento da función podería ser:

**select** ficha\_texto**(**'JFL'**);**

* A saída do resultado dende Workbench sería:



* Solución ao apartado 2
* O código do procedemento podería ser o seguinte:

/\*

procedemento\_estatisticos.sql

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

NOME RUTINA: estatisticos (procedemento)

DATA CREACIÓN: 18/11/2015

AUTOR: Grupo licenza 2015

TAREFA A AUTOMATIZAR: Mostre as seguintes estatísticas dun equipo:

Número de partidos xogados como local

Número de partidos xogados como visitante

Número de partidos ganados como local

Número de partidos ganados como visitante

Número de puntos conseguidos xogando como local

Número de puntos conseguidos xogando como visitante

Número de puntos recibidos xogando como local

Número de puntos recibidos xogando como visitante

Se o código do equipo non existe, o procedemento debe provocar

un erro, deter a execución e mostrar a mensaxe:

"Código de equipo non existe"

PARAMETROS REQUERIDOS: - Código dun equipo

RESULTADOS PRODUCIDOS: - Mostrar os estatísticos anteriores ou a mensaxe de erro.

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

\*/

**use** liga\_acb**;**

**drop** **procedure** **if** **exists** estatisticos**;**

delimiter **//**

**create** **procedure** estatisticos**(**pcodigo **char(**4**))**

**begin**

**declare** vEquipo\_existe **tinyint** unsigned **default** 0**;**

**declare** vPartidos\_xogados\_local **smallint** unsigned **default** 0**;**

**declare** vPartidos\_xogados\_visitante **smallint** unsigned **default** 0**;**

**declare** vPartidos\_ganados\_local **smallint** unsigned **default** 0**;**

**declare** vPartidos\_ganados\_visitante **smallint** unsigned **default** 0**;**

**declare** vPuntos\_conseguidos\_local **smallint** unsigned **default** 0**;**

**declare** vPuntos\_conseguidos\_visitante **smallint** unsigned **default** 0**;**

**declare** vPuntos\_recibidos\_local **smallint** unsigned **default** 0**;**

**declare** vPuntos\_recibidos\_visitante **smallint** unsigned **default** 0**;**

**select** **count(\*)** **into** vEquipo\_existe **from** equipo **where** codigo **=** pcodigo**;**

**if** vEquipo\_existe **=** 0 **then**

signal **sqlstate** '45000'

**set** message\_text **=** 'Código de equipo non existe'**;**

**else**

**select** **count(\*)** **into** vPartidos\_xogados\_local

**from** partido **where** equipo\_local **=** pcodigo**;**

**select** **count(\*)** **into** vPartidos\_xogados\_visitante

**from** partido **where** equipo\_visitante **=** pcodigo**;**

**select** **count(\*)** **into** vPartidos\_ganados\_local

**from** partido

**where** equipo\_local**=**pcodigo **and** puntos\_local**>**puntos\_visitante**;**

**select** **count(\*)** **into** vPartidos\_ganados\_visitante

**from** partido

**where** equipo\_visitante**=**pcodigo **and** puntos\_local**<**puntos\_visitante**;**

**select** **sum(**puntos\_local**)** **into** vPuntos\_conseguidos\_local

**from** partido **where** equipo\_local **=** pcodigo**;**

**select** **sum(**puntos\_visitante**)** **into** vPuntos\_conseguidos\_visitante

**from** partido **where** equipo\_visitante **=** pcodigo**;**

**select** **sum(**puntos\_visitante**)** **into** vPuntos\_recibidos\_local

**from** partido **where** equipo\_local **=** pcodigo**;**

**select** **sum(**puntos\_local**)** **into** vPuntos\_recibidos\_visitante

**from** partido **where** equipo\_visitante **=** pcodigo**;**

**select**

vPartidos\_xogados\_local**,**

vPartidos\_xogados\_visitante**,**

vPartidos\_ganados\_local**,**

vPartidos\_ganados\_visitante**,**

vPuntos\_conseguidos\_local**,**

vPuntos\_conseguidos\_visitante**,**

vPuntos\_recibidos\_local**,**

vPuntos\_recibidos\_visitante**;**

**end** **if;**

**end**

**//**

delimiter **;**

* A sentenza para probar o funcionamento da función podería ser:

**call** estatisticos**(**'ESTU'**);**

* A saída de resultado dende Workbench sería:



Exemplo de lista de valoración para TO.2

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.2 (EV) (observación indirecta: documento de rexistro de execución de procedementos almacenados e xeracións de guións para a creación, modificación e eliminación de procedementos almacenados).

| Nome | Data | |
| --- | --- | --- |
| Indicadores de  CA7.5.1 - Creáronse, modificáronse, elimináronse e executáronse procedementos almacenados. | Valoración máxima  15 | Cualificación |
| Escribiuse o código do procedemento almacenado *estatisticos* e cumpre as condicións solicitadas. | 11 |  |
| O código inclúe a eliminación do procedemento se é que existía. | 1 |  |
| O procedemento asociouse á base de datos *liga\_acb*. | 1 |  |
| Creouse o procedemento. | 1 |  |
| Executouse o procedemento almacenado e mostrouse o resultado utilizando o cliente Workbench. | 1 |  |

Exemplo de lista de valoración para TO.3

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.3 (EV) (observación indirecta: documento de rexistro de xeración de guións que deseñen procedementos almacenados con parámetros).

| Nome | Data | |
| --- | --- | --- |
| Indicadores de  CA7.6 Empregáronse parámetros no deseño de procedementos almacenados. | Valoración máxima  10 | Cualificación |
| Empregáronse os parámetros requiridos no procedemento almacenado. | 10 |  |

Exemplo de lista de valoración para TO.4

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.4 (EV) (observación indirecta: documento de rexistro de procedementos almacenados que traten erros producidos durante a execución dos mesmos).

| Nome | Data | |
| --- | --- | --- |
| Indicadores de  CA7.7 Detectáronse e tratáronse erros ao executar procedementos almacenados. | Valoración máxima  5 | Cualificación |
| O código do procedemento inclúe o tratamento de erros solicitado. | 5 |  |

Exemplo de lista de valoración para TO.5

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.5 (EV) (observación indirecta: documento de rexistro de rutinas almacenadas que utilicen utilice funcións MySQL).

| Nome | Data | |
| --- | --- | --- |
| Indicadores de  CA7.8 Usáronse as funcións proporcionadas polo sistema xestor. | Valoración máxima  5 | Cualificación |
| Usáronse as función proporcionadas polo sistema xestor. | 5 |  |

Exemplo de lista de valoración para TO.6

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.6 (EV) (observación indirecta: documento de rexistro de xeración de guións que definan e utilicen funcións de usuario).

| Nome | Data | |
| --- | --- | --- |
| Indicadores de  CA 7.9.1 Creáronse, modificáronse, elimináronse e usáronse funcións de usuario. | Valoración máxima  15 | Cualificación |
| Escribiuse o código da función de usuario e cumpre as condicións solicitadas. | 10 |  |
| O código inclúe a eliminación da función se é que existía. | 1 |  |
| A función asociouse á base de datos *liga\_acb*. | 1 |  |
| Creouse a función. | 1 |  |
| Escribiuse o código que permite usar a función. | 1 |  |
| Executouse o código anterior e mostrouse o resultado utilizando o cliente Workbench. | 1 |  |

Exemplo de lista de valoración para TO.9

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.9 (EV) (observación indirecta: documento de rexistro que mostre que os guións están documentados como mínimo coa indicación das tarefas que automatizan e os resultados que producen).

| Nome | Data | |
| --- | --- | --- |
| Indicadores de  CA7.12 Documentáronse os guións codificados, e indicáronse as tarefas que automatizan e os resultados que producen. | Valoración máxima  8 | Cualificación |
| Documentáronse as rutinas indicando a lo menos as tarefas que automatizan, os resultados que producen e os parámetros utilizados. | 8 |  |