Dirección Xeral de Educación, Formación Profesional e Innovación Educativa

Material para

a formación profesional inicial

A01. Creación da estrutura de bases de datos relacionais

|  |  |  |
| --- | --- | --- |
| 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 | UD04 | Deseño físico de bases de datos |
| **Actividade** | **A01** | **Creación da estrutura de bases de datos relacionais** |
| 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\_V000401\_UD04\_A01\_CreaBD.docx |
| © 2015 Xunta de Galicia.  Consellería de Cultura, Educación e Ordenación Universitaria.  Este material 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](#_Toc443054194)

[Contexto da actividade 5](#_Toc443054195)

[Título da actividade 6](#_Toc443054196)

[Resultados de aprendizaxe do currículo 6](#_Toc443054197)

[Obxectivos didácticos e título e descrición da actividade 6](#_Toc443054198)

[Criterios de avaliación 6](#_Toc443054199)

[Contidos 7](#_Toc443054200)

[Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación 8](#_Toc443054201)

[2. A01. Creación da estrutura de bases de datos relacionais 10](#_Toc443054202)

[2.1 Introdución 10](#_Toc443054203)

[2.1.1 Obxectivos 10](#_Toc443054204)

[2.1.2 Software 10](#_Toc443054205)

[2.2 Actividade 11](#_Toc443054206)

[2.2.1 Deseño físico 11](#_Toc443054207)

[2.2.2 Linguaxe SQL 11](#_Toc443054208)

[2.2.2.1 Definición de datos 11](#_Toc443054209)

[2.2.2.2 Manipulación de datos 12](#_Toc443054210)

[2.2.2.3 Control de datos 12](#_Toc443054211)

[2.2.2.4 Estándares SQL 12](#_Toc443054212)

[2.2.2.5 Formas de utilizar SQL 13](#_Toc443054213)

[2.2.2.6 Conexión a un servidor MySQL. 13](#_Toc443054214)

[2.2.2.7 Forma básica das sentenzas SQL 15](#_Toc443054215)

[2.2.2.8 Notación para a sintaxe 15](#_Toc443054216)

[2.2.2.9 Comentarios en SQL 16](#_Toc443054217)

[2.2.3 Linguaxe de definición de datos 17](#_Toc443054218)

[2.2.3.1 Consulta do dicionario de datos dun servidor MySQL 17](#_Toc443054219)

[2.2.3.2 Nomes dos obxectos 18](#_Toc443054220)

[2.2.3.3 Base de datos 18](#_Toc443054221)

[Creación dunha base de datos 18](#_Toc443054222)

[Poñer en uso unha base de datos 19](#_Toc443054223)

[Borrar unha base de datos 20](#_Toc443054224)

[2.2.3.4 Táboa 20](#_Toc443054225)

[Creación dunha táboa 20](#_Toc443054226)

[Tipos de datos en MySQL 21](#_Toc443054227)

[Implementación da integridade mediante restricións 25](#_Toc443054228)

[Métodos para implementar a integridade 26](#_Toc443054229)

[Definición de restricións 27](#_Toc443054230)

[Creación de táboas e restricións 27](#_Toc443054231)

[Asociar estruturas de índices ás columnas 30](#_Toc443054232)

[Información sobre táboas 30](#_Toc443054233)

[Creación dunha táboa a partir dunha táboa que xa existe 31](#_Toc443054234)

[Borrado dunha táboa 31](#_Toc443054235)

[Motores de almacenamento en MySQL 31](#_Toc443054236)

[Táboas particionadas 33](#_Toc443054237)

[2.2.3.5 Índices 35](#_Toc443054238)

[Creación de índices 35](#_Toc443054239)

[Borrado de índices 35](#_Toc443054240)

[2.2.4 Tarefa integradora 36](#_Toc443054241)

[2.3 Tarefas 36](#_Toc443054242)

[2.3.1 Tarefa 1. Conectar cun servidor de bases de datos 36](#_Toc443054243)

[Solución 36](#_Toc443054244)

[2.3.2 Tarefa 2. Crear bases de datos empregando a sentenza CREATE 38](#_Toc443054245)

[Solución 38](#_Toc443054246)

[2.3.3 Tarefa 3. Poñer en uso as bases de datos empregando a sentenza USE 39](#_Toc443054247)

[Solución 39](#_Toc443054248)

[2.3.4 Tarefa 4. Borrar bases de datos empregando a sentenza DROP 39](#_Toc443054249)

[Solución 39](#_Toc443054250)

[2.3.5 Tarefa 5. Crear táboas empregando a sentenza CREATE 39](#_Toc443054251)

[Solución 41](#_Toc443054252)

[2.3.6 Tarefa 6. Borrar táboas empregando a sentenza DROP 43](#_Toc443054253)

[Solución 43](#_Toc443054254)

[2.3.7 Tarefa 7. Crear índices empregando a sentenza CREATE 44](#_Toc443054255)

[Solución 44](#_Toc443054256)

[2.3.8 Tarefa 8. Borrar índices empregando a sentenza DROP 44](#_Toc443054257)

[Solución 44](#_Toc443054258)

[2.3.9 Tarefa 9. Tarefa integradora 44](#_Toc443054259)

[Solución 46](#_Toc443054260)

[3. Materiais 51](#_Toc443054261)

[3.1 Documentos de apoio ou referencia 51](#_Toc443054262)

[3.2 Recursos didácticos 51](#_Toc443054263)

[3.3 Material auxiliar 51](#_Toc443054264)

[4. Avaliación 52](#_Toc443054265)

[Criterios de avaliación 52](#_Toc443054266)

[Modelo de proba combinada para PE.1, TO.1, TO.3, TO.5, TO.6, TO.7 e TO.9 52](#_Toc443054267)

[Proba para CA4.1 sobre formatos de almacenamento 53](#_Toc443054268)

[Proba para CA4.2.1 sobre crear e eliminar bases de datos e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD 54](#_Toc443054269)

[Proba para CA4.3.1 sobre crear e eliminar táboas, CA4.4 sobre seleccionar tipos de datos adecuados para as táboas e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD 54](#_Toc443054270)

[Proba para CA4.5 sobre campos clave e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD 54](#_Toc443054271)

[Proba para CA4.6 sobre aplicar restricións e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD 54](#_Toc443054272)

[Exemplo de solución para entregar en papel 54](#_Toc443054273)

[Exemplo de lista de valoración para TO.1 58](#_Toc443054274)

[Exemplo de lista de valoración para TO.3 58](#_Toc443054275)

[Exemplo de lista de valoración para TO.5 59](#_Toc443054276)

[Exemplo de lista de valoración para TO.6 59](#_Toc443054277)

[Exemplo de lista de valoración para TO.7 59](#_Toc443054278)

[Exemplo de lista de valoración para TO.9 59](#_Toc443054279)

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\_V000400\_UD04\_DesenoFisico.pdf

Título da actividade

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Título | Descrición | Duración |
| A01 | Creación da estrutura de bases de datos relacionais | Crearanse bases de datos, táboas e relacións entre elas e definiranse os tipos de datos adecuados, utilizando a linguaxe de definición de datos. | 13 |

Resultados de aprendizaxe do currículo

|  |  |
| --- | --- |
| Resultados de aprendizaxe do currículo | Completo |
| * RA4 -  Crea bases de datos, e define a súa estrutura e as características dos seus elementos segundo o modelo relacional. | Si |

Obxectivos didácticos e título e descrición da actividade

| Obxectivos específicos | | Actividade | | Descrición básica | Duración |
| --- | --- | --- | --- | --- | --- |
| O1.1 | Comprender a forma na que o SXBD almacena a información e as alternativas que ofrece. | A01 | Creación da estrutura de bases de datos relacionais. | Crearanse bases de datos, táboas e relacións entre elas e definiranse os tipos de datos adecuados, utilizando a linguaxe de definición de datos. | 13 |
| O1.2 | Crear bases de datos. |
| O1.3 | Crear táboas e relacións entre elas. |
| O1.4 | Seleccionar os tipos de datos adecuados. |
| O1.5 | Definir os campos clave nas táboas. |
| O1.6 | Aplicar as restricións reflectidas no deseño lóxico. |
| O1.7 | Utilizar a linguaxe de definición de datos. |

Criterios de avaliación

|  |
| --- |
| Criterios de avaliación |
| * CA4.1 -  Analizouse o formato de almacenamento da información. * CA4.2 -  Creáronse bases de datos.   CA4.2.1 - Creáronse e elimináronse bases de datos.   * CA4.3 -  Creáronse as táboas e as relacións entre elas.   CA4.3.1 - Creáronse e elimináronse táboas e relacións entre elas.   * CA4.4 -  Seleccionáronse os tipos de datos adecuados. * CA4.5 -  Definíronse os campos clave nas táboas. * CA4.6 -  Aplicáronse as restricións reflectidas no deseño lóxico. * CA4.8 -  Utilizáronse asistentes, ferramentas gráficas e a linguaxe de definición de datos.   CA4.8.1 - Utilizáronse asistentes, ferramentas gráficas e a linguaxe de definición de datos. |

Contidos

| Contidos |
| --- |
| * Ferramentas gráficas achegadas polo sistema xestor para a implementación da base de datos. * Linguaxe de definición de datos (DDL). * Creación, modificación e eliminación de bases de datos.   Creación e eliminación de bases de datos (CA 4.2.1).   * Creación, modificación e eliminación de táboas.   Creación e eliminación de táboas (CA 4.3.1).   * Tipos de datos. Tipos de datos definidos polo usuario. * Implementación de restricións. * Índices: características. |

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 |
| A01. Creación da estrutura de bases de datos relacionais   * Crearanse bases de datos, táboas e relacións entre elas e definiranse os tipos de datos adecuados, utilizando a linguaxe de definición de datos. | * **Tp1.1** Explicación de conceptos relativos a: deseño físico de bases de datos, introdución á linguaxe SQL (Structured Query Language) e maneiras de executar sentenzas SQL nun servidor de base de datos. | **Ta1.1** Toma de notas e formulación de dúbidas. |  | * Ordenadores con conexión a Internet, que terán instalado o sistema xestor de bases de datos MySQL, o cliente MySQL Workbench e un programas de deseño de diagramas (tipo MS Visio ou Dia). * Material didáctico subministrado polo profesorado en papel e/ou formato dixital. * Proxector. * Manual de referencia de MySQL. |  | 0.5 |
| * **Tp1.2** Demostración práctica de conexión a un servidor en modo texto dende a consola e utilizando un cliente gráfico. | * **Ta1.2** Establecemento da conexión cun servidor en modo texto dende a consola e utilizando un cliente gráfico. | * Conexión co servidor do SXBD. |  | 0.5 |
| * **Tp1.3** Explicación de conceptos sobre tipos de táboas e explicación de sintaxe das sentenzas da linguaxe de definición de datos (LDD) de SQL para: creación e borrado de bases de datos; creación e borrado de táboas; manexo de índices. | * **Ta1.3** Creación de estruturas de base de datos relacionais baseándose en diagramas Entidade/Relación e grafos relacionais e utilizando guións de sentenzas da linguaxe de definición de datos (LDD) de SQL. As operacións a realizar serán: creación e borrado de bases de datos; creación e borrado de táboas; engadido de relacións e restricións de clave foránea; creación e borrado de índices. | * Guións de sentenzas na linguaxe de definición de datos (LDD) de SQL coas solucións aos exercicios propostos. * Resultado da execución dos guións de sentenzas. |  | 10.5 |
|  | * **Ta1.4** Tarefa de avaliación utilizando os instrumentos PE.1, TO.1, TO.3, TO.5, TO.6, TO.7 e TO.9. | * Exame en papel e en formato dixital. * Resultado da execución dos guións de sentenzas. | * Máquina virtual específica para exame que terá instalado o sistema xestor de bases de datos MySQL e o cliente MySQL Workbench. * Manual de referencia de MySQL. | * **PE.1** - Cuestionario con preguntas de resposta breve sobre o formato de almacenamento da información. * **TO.1** - Documento de rexistro de creación e eliminación de bases de datos utilizando a LDD. Escala de valores (observación indirecta). * **TO.3** - Documento de rexistro de creación e eliminación de táboas e as relacións entre elas utilizando a LDD. Escala de valores (observación indirecta). * **TO.5** - Documento de rexistro de selección dos tipos de datos das táboas utilizando a LDD. Escala de valores (observación indirecta). * **TO.6** - Documento de rexistro de definición de campos clave utilizando a LDD. Escala de valores (observación indirecta). * **TO.7** - Documento de rexistro de aplicación das restricións obtidas utilizando a LDD. Escala de valores (observación indirecta). * **TO.9** - Documento de rexistro de utilización de Workbench para editar e executar os guións LDD. Escala de valores (observación indirecta). | 1.5 |

1. A01. Creación da estrutura de bases de datos relacionais
   1. Introdución
      1. Obxectivos

Os obxectivos desta actividade son:

* Analizar os detalles do almacenamento físico dunha base de datos.
* Crear e borrar bases de datos e táboas utilizando a linguaxe SQL de definición de datos, asistentes e ferramentas gráficas, seguindo as especificacións detalladas nos deseños conceptuais e lóxicos.
  + 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 para facer as prácticas asociadas a esta actividade, 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.

* En [https://www.mysql.com/products/workbench/](https://www.mysql.com/products/workbench/%20) 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 auxiliar anexo a esta actividade inclúe unha Guía básica de MySQL Workbench 6.3.
  1. Actividade
     1. Deseño físico

O deseño físico é o proceso de implantación definitiva da base de datos deseñada, sobre un SXBD concreto, utilizando directamente instrucións do xestor da base de datos, ou ben, mediante a axuda dalgunha ferramenta gráfica que facilite a creación da base de datos sen necesidade de coñecer a sintaxe da linguaxe utilizada polo xestor para ese fin. Realízase unha vez rematadas as fases de deseño conceptual e deseño lóxico.

É imprescindible empregar o manual de referencia do SXBD seleccionado para coñecer as normas de sintaxe propias do fabricante. Destácanse as seguintes precaucións:

* Nomes de bases de datos, táboas, columnas. Como regra xeral poderíamos empregar nomes que só leven letras, números e o guión baixo, tratando de evitar o uso doutros carácteres especiais incluídos o ñ, os acentos ou calquera outro símbolo. A razón desta recomendación é que aínda que o noso SXBD permita o uso de carácteres especiais pode que no futuro teñamos que migrar as nosas bases de datos a outro SXBD, ou a outro sistema operativo, que teña unhas normas máis restritivas en canto aos nomes que se poden empregar.
* Uso indistinto de maiúsculas, minúsculas, ou unha mestura de ambas, para nomear. Deberíase de empregar unha norma xeral, como por exemplo, poñer os nomes sempre en minúsculas xa que se tiveramos que migrar dun sistema Windows a outro sensible a maiúsculas e minúsculas (*case sensitive*), como pode ser Linux, pódense xerar problemas.
* Tipos de columnas. Cando se fai unha migración dunha base de datos dun SXBD a outro, é importante revisar os tipos de columnas propios dos SXBD utilizados, xa que, a pesar de existir unha norma ANSI estándar para os tipos de datos, cada fabricante introduce algunhas modificacións para mellorar o rendemento do seu motor. Así é posible que no SXBD dun determinado fabricante existan tipos de datos que non existen nos demais, ou que un mesmo tipo de dato se comporte de distinta maneira en canto a rangos de valores permitidos e espazo que ocupa no almacenamento.
* Definición das estruturas físicas de almacenamento. É importante coñecer a maneira en que o noso SXBD almacena os obxectos das bases de datos (*tablespaces*), ou os ficheiros de datos (*datafiles*).
  + 1. Linguaxe SQL

SQL corresponde ao acrónimo de *Structured Query Language* (Linguaxe Estruturado de Consultas) e é unha ferramenta para organizar, xestionar e recuperar datos almacenados nunha base de datos relacional. Naceu como software de consulta pero actualmente aínda que esa é unha das súas funcións máis importantes, utilízase para controlar todas as funcións que subministra un SXBDR aos seus usuarios, incluíndo todas as funcións das linguaxes deseñadas para o manexo de bases de datos: Linguaxe de Definición de Datos, Linguaxe de Manipulación de Datos, e Linguaxe de Control de Datos.

* + - 1. Definición de datos

A linguaxe de definición de datos ou LDD (en inglés *Data Definition Language* ou *DDL*) permite a creación do esquema da base de datos e a organización física dos datos almacenados.

* + - 1. Manipulación de datos

A linguaxe de manipulación de datos ou LMD (en inglés *Data Manipulation Language*, ou *DML*) permite:

* Recuperación de datos. SQL permite recuperar e utilizar os datos almacenados nunha base de datos a un usuario ou a un programa.
* Mantemento de datos. SQL permite actualizar a base de datos a un usuario ou a un programa, engadindo novos datos, borrando e modificando os que xa están almacenados.
  + - 1. Control de datos

A linguaxe de control de datos ou LCD (en inglés *Data Control Language* ou *DCL*) permite:

* Control do acceso. SQL pode ser utilizado para xestionar as contas dos usuarios e restrinxir a súa capacidade para recuperar, engadir e modificar datos, protexendo os datos almacenados contra accesos non autorizados.
* Control do acceso concorrente aos datos. SQL permite establecer sistemas de bloqueos de datos para permitir que varios usuarios poidan acceder ao mesmo tempo á base de datos para compartir información, evitando interferencias entre eles.
* Integridade de datos. SQL permite establecer medidas de seguridade para protexer os datos de ataques externos ou ante fallos do sistema e a recuperación da base de datos para volver a estar dispoñible para os usuarios.

GraficoSentenzas2.emf

* + - 1. Estándares SQL

Un dos elementos clave da aceptación de SQL no mercado é a existencia dun estándar oficial adoptado polo *American National Standards Institute* (ANSI) e a *International Standards Organization* (ISO). Sen embargo, hai outros dous estándares de SQL importantes que inclúen o SQL definido por DB2 de IBM e o estándar X/OPEN de SQL para UNIX.

Os inicios do estándar oficial de SQL foron en 1982, cando ANSI encargou ao seu comité X3H2 a definición dunha linguaxe de base de datos relacional. O estándar ANSI para SQL resultante está baseado en gran medida no SQL de DB2, aínda que contén algunhas diferenzas importantes.

|  |  |
| --- | --- |
| Resumo de revisións do estándar SQL[[1]](#footnote-3) | |
| SQL-86 | Primeira publicación feita por ANSI. Confirmada por ISO en 1987. |
| SQL-89, ou SQL1 | Supón unha revisión menor da norma anterior |
| SQL-92, ou SQL2 | Supón unha revisión importante, con normas máis ríxidas. Mentres que o estándar orixinal de 1986 ocupaba menos de 100 páxinas, o estándar oficial SQL2, ou SQL-92, ocupou case 600. |
| SQL-99 ou SQL3 | Incorpora a utilización de *triggers* ou disparadores, novos tipos de datos e novas características enfocadas á programación orientada a obxectos. |
| Revisións dos anos 2003, 2005, 2008 e 2011 | Centran a maioría dos esforzos en introducir algunhas características de XML, definindo a maneira en que SQL pode ser utilizado conxuntamente co XML. |

A pesar destas normas de estandarización, cada fabricante de SXBDR incorpora o seu propio dialecto SQL co fin de ofertar un produto que o diferencie dos seus competidores e incluso chega a poñerlle un nome diferente. Exemplos:

* Access SQL de Microsoft Access
* Tansact SQL de Microsoft SQL Server
* PL/SQL de Oracle
* MySQL de MySQL
* PgSQL de PostgreSQL
  + - 1. Formas de utilizar SQL

SQL é unha linguaxe declarativa de alto nivel, é dicir, non procedimental. Isto significa que as sentenzas SQL especifican o que se quere obter, pero non a forma de conseguilo. A orde interna na que se executan as operacións asociadas a unha sentenza SQL é establecida polo optimizador de consultas do SXBD. As sentenzas SQL poden utilizarse:

* Escribindo directamente as sentenzas coa axuda dun cliente en modo texto, en modo gráfico, ou na pantalla dun terminal interactivo; o servidor devolve o resultado da petición.
* Gardando un conxunto de sentenzas nun ficheiro de ordes (*scripts*), que se executan todas seguidas de forma secuencial.
* Escribindo as sentenzas de SQL incluídas (embebidas) en programas escritos con distintas linguaxes de programación como PHP, Java ou C#.
  + - 1. Conexión a un servidor MySQL.

Para conectarse a un servidor de MySQL é necesario empregar algunha aplicación cliente que establece a conexión utilizando os parámetros que lle pasa o usuario. Os parámetros que necesita a aplicación cliente son: o nome do servidor ou a IP que identifica ao equipo na rede (por defecto, *localhost*), o porto no que o servidor escoita as peticións (por defecto, 3306), o nome de usuario que se conecta (por defecto, *root*), e a súa contrasinal (por defecto, sen contrasinal). As aplicacións cliente máis empregadas para conectase a un servidor MySQL e executar as sentenzas SQL son as seguintes:

* MySQL Workbench:

Cando se executa, hai que seleccionar o servidor co que se quere establecer a conexión, e ábrese un entorno gráfico no que se mostra información do servidor. Permite escribir sentenzas SQL mediante un editor, para ser enviadas ao servidor. Para máis información sobre o seu funcionamento, débese utilizar o manual de referencia ou ben a guía básica de funcionamento de MySQL Workbench incluída no material auxiliar desta actividade.

* Consola modo texto mysql.exe:
* No caso de ter instalado Wampserver, hai que abrir o menú de Wampserver e seleccionar 'MySQL console' no submenú de MySQL. Ábrese unha consola en modo texto que solicita introducir o contrasinal do usuario *root*. Unha vez introducido o contrasinal, móstrase o prompt *mysql>* que indica que está establecida a conexión co servidor e xa se poden empezar a escribir sentenzas para que o servidor as execute.

As sentenzas SQL que se executen nesta consola estarán completas cando terminen en punto e coma. Pódense escribir sentenzas que ocupen varias liñas. Se pulsa a tecla *Intro* e a liña non finaliza en punto e coma, mostrarase o prompt *->* que indica que a sentenza continúa. Mentres non se envíe o punto e coma, non se considera que a sentenza estea completa e non é executada polo servidor.

* No caso de non ter instalado Wampserver, hai que abrir unha consola de ordes do sistema e executar a utilidade cliente *mysql.exe* que está no directorio *bin* que se atopa no directorio de instalación de MySQL (en Windows é normalmente, c:\Program Files\MySQL\MySQL version) .
* Aplicación web phpMyAdmin:
* No caso de ter instalado Wampserver, hai que abrir o menú e seleccionar a opción phpMyAdmin.
* No caso de non ter instalado Wampserver, hai que abrir un navegador e teclear a url coa identificación do servidor web no que está instalado phpMyAdmin. Exemplos de url: http://localhost/phpMyAdmin (no caso que phpMyAdmin estea instalado nun servidor web no equipo local), http://proveedor\_web.com/phpMyAdmin (no caso que phpMyAmin estea instalado no servidor proveedor\_web.com).
* Tarefa1. Conexión cun servidor MySQL.

Aparte de poder conectarse co servidor empregando as anteriores aplicacións cliente, tamén se pode facer dende os programas de aplicación que van a procesar os datos das bases de datos, incluíndo as sentenzas de conexión no código dos programas. No seguinte exemplo móstrase unha maneira de establecer a conexión dende un programa escrito en PHP:

<?php

$servidor = 'localhost:3306';

$usuario = 'root';

$contrasinal = '';

$basedatos = 'prueba\_php';

/\* instanciar a clase para crear un novo obxecto de conexión \*/

$conexion = new mysqli($servidor, $usuario, $contrasinal, $basedatos);

if (mysqli\_connect\_errno())

{ echo '<p class="erro">Erro: Fallo na conexión co servidor ' .

$servidor . ' ou coa base de datos: ' . $basedatos . ': ' .

mysqli\_connect\_error() . '</p>';

exit(); }

/\* Aquí iría o código para manipular os datos da base de datos \*/

$conexion->close();

?>

* + - 1. Forma básica das sentenzas SQL

A linguaxe SQL consta dun conxunto de sentenzas. Cada sentenza indica unha acción específica a realizar por parte do SXBD, tal como a creación dunha nova táboa, a consulta de datos dunha ou máis táboas, ou a inserción de novos datos na base de datos.

Dibujo2.wmf

Todas as sentenzas SQL teñen a mesma estrutura:

* Empezan cun verbo, que é unha palabra clave que indica a acción que ten que executar o SXBD. Algúns destes verbos son: CREATE, INSERT, DELETE, ou SELECT.
* A sentenza continúa cunha ou máis cláusulas. Unha cláusula pode especificar os datos sobre os que debe actuar a sentenza, ou proporcionar máis detalles acerca da forma en que se ten que executar. Todas as cláusulas empezan tamén cunha palabra clave, tal como WHERE, FROM, INTO e HAVING e van seguidas de expresións, nomes de táboas ou nomes de columnas. Algunhas cláusulas teñen que aparecer de forma obrigatoria na sentenza, pero outras son opcionais.

O estándar SQL ANSI/IS0 especifica as palabras clave da linguaxe SQL que se utilizan como verbos ou como cláusulas das sentenzas. Ademais, establece que estas palabras clave non poden ser empregadas para designar obxectos da base de datos, tales como táboas, vistas, columnas ou usuarios. Moitos dialectos de SQL suavizan esta restrición, pero é recomendable evitar o uso de palabras clave ao nomear táboas, columnas, ou calquera outro obxecto da base de datos.

* + - 1. Notación para a sintaxe

SQL, do mesmo xeito que outras linguaxes, está formada por un conxunto de palabras e un conxunto de normas de sintaxe para a construción das sentenzas. A notación empregada nos manuais para a explicación da sintaxe das sentenzas resúmese na seguinte táboa.

|  |  |
| --- | --- |
| Resumo de notacións utilizadas na sintaxe das sentenzas SQL | |
| MAIÚSCULAS | O texto escrito en maiúsculas representa palabras reservadas, que non poden ser utilizadas para outros fins. Á hora de escribir a sentenza non se diferenza entre minúsculas e maiúsculas |
| Minúsculas | Representan partes da instrución nas que temos que substituír a palabra ou frase escrita en minúsculas polo que representa |
| [ texto] | A parte da sentenza encerrada entre corchetes é optativa |
| {opción1 | opción2 } | Representa unha alternativa. Ao escribir a instrución tense que elixir unha das opcións encerradas entre as chaves, que van separadas por barras verticais ( | ) |
| ... | Tres puntos seguidos indican que a parte da sentenza que está inmediatamente antes deles, pódese repetir varias veces |

No seguinte exemplo de sintaxe, as palabras CREATE, DATABASE, SCHEMA, IF, NOT, e EXISTS son palabras reservadas que non se poden utilizar noutro lugar da sentenza diferente ao que aparecen na norma de sintaxe.

Sintaxe.emf

A escritura das sentenzas SQL pode facerse en maiúsculas, minúsculas ou nunha combinación de ambas.

* + - 1. Comentarios en SQL

Os comentarios poden formar parte dunha sentenza SQL e non serán executados polo servidor. Poden estar delimitados ou non e ocupar unha liña ou máis dunha. Os símbolos utilizados para definir comentarios nun *script* de SQL ou nunha sentenza, son os que se mostran na seguinte táboa.

|  |  |
| --- | --- |
| Símbolos para definir comentarios | |
| -- | Dous guións ao comezo dunha liña indican que esa liña sexa interpretada como un comentario |
| # | O símbolo # indica que o texto que hai dende el ata o final da liña sexa interpretado como un comentario |
| /\* texto \*/ | Define o texto como comentario. O texto pode ocupar varias liñas |

Exemplo de código con comentarios:

-- Script de creación da estrutura da base de datos horas\_extra

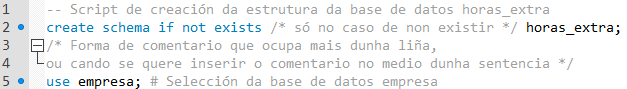
**create schema if not exists** /\* só no caso de non existir \*/ horas\_extra;

/\* Forma de comentario que ocupa máis dunha liña,

ou cando se quere inserir o comentario no medio dunha sentenza \*/

**use** empresa; # Selección da base de datos empresa

Así se verían os comentarios no editor de MySQL Workbench:



MySQL permite utilizar os símbolos /\* e \*/ propios dos comentarios para definir sentenzas executables. Esta utilización debe facerse con precaución para non ter problemas ao executar arquivos de scripts de MySQL noutros SXBDR.

|  |  |
| --- | --- |
| Utilización especial dos comentarios en MySQL | |
| /\*! código\_MySQL \*/ | En MySQL, interprétase o código como unha sentenza executable e execútase; no resto de servidores, interprétase como comentario |
| /\*!version código\_MySQL \*/ | En MySQL, interprétase o texto como unha sentenza executable e execútase nun servidor cunha versión igual ou superior á versión especificada; no resto de servidores, interprétase como comentario |

Exemplo de código e vista do mesmo editado en MySQLWorkbench:

/\*! **create database empresa** \*/;

/\*!50021 **use** empresa \*/;

**create** /\*!32302 **temporary** \*/ **table** proba (columna **integer**);



* + 1. Linguaxe de definición de datos

A linguaxe de definición de datos, LDD ou DDL, está formada polo conxunto de instrucións do SQL que permiten ao administrador crear o esquema da base de datos e facer os cambios necesarios, no seu esquema, unha vez que está creada. Permite crear, modificar e suprimir bases de datos, táboas e índices. O núcleo da LDD está baseado en tres verbos de SQL:

* CREATE, que define e crea un obxecto da base de datos.
* DROP, que elimina un obxecto existente na base de datos.
* ALTER, que modifica a definición dun obxecto existente na base de datos.
  + - 1. Consulta do dicionario de datos dun servidor MySQL

O dicionario de datos ou catálogo, garda información sobre os obxectos almacenados no servidor. O que se coñece como metadatos (datos sobre os datos).

Cando se instala MySQL créanse automaticamente catro bases de datos: *mysql*, *information\_schema*, *performance\_schema* e *test*. As dúas primeiras gardan información dos metadatos.

O estándar ANSI/ISO SQL:2003 no apartado referido ao esquema (*Part 11 'Schemata'*) define a estrutura das táboas da base de datos *information\_schema* que facilita a consulta dos metadatos. Son táboas de só lectura.

Para consultar información sobre os metadatos, MySQL dispón dun conxunto propio de sentenzas identificadas co nome SHOW, e ademais, a partir da versión 5.0.2 está dispoñible a base de datos *information\_schema* que lle permite adaptarse á norma ANSI SQL.

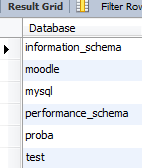
Todos os usuarios de MySQL teñen permiso para acceder ás táboas da base de datos *information\_schema*, pero só poderán ver aquelas filas correspondentes a obxectos sobre os que teñan permisos.

Exemplo de consulta dos metadatos nun servidor de MySQL. Para ver os nomes das bases de datos creadas nun servidor pódense utilizar dúas opcións:

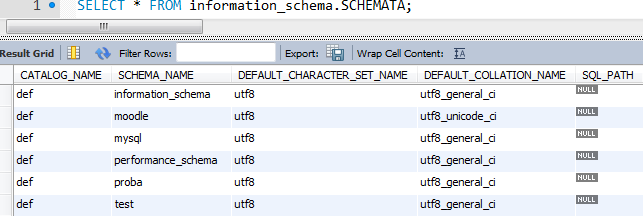
* Executar a sentenza SHOW DATABASES que só funciona en MySQL.

**show databases;**

O resultado de executar a sentenza será unha lista cos nomes das bases de datos creadas no servidor.



* Consultar a táboa SCHEMATA da base de datos *information\_schema*.



Cando se traballa utilizando clientes gráficos como MySQL Workbench ou PHPMyAdmin, ao establecer a conexión, móstranse os nomes das bases de datos. No caso de MySQL Workbench, as táboas e outros obxectos relacionados coa base de datos, móstranse na zona de navegación.

* + - 1. Nomes dos obxectos

Tódolos manuais de referencia dos SXBDR teñen especificadas as normas para nomear os obxectos do esquema das bases de datos: base de datos, táboas, índices, columnas, desencadeadores, ou procedementos almacenados. Resumo das normas para nomear obxectos en MySQL:

* Lonxitude máxima de carácteres para o nome: 64.
* Carácteres permitidos: calquera letra do alfabeto inglés (sen ñ nin acentos) tanto en minúscula como en maiúsculas, números e os carácteres especiais \_ e $. Pódese empregar, aínda que non se recomenda, o acento grave (`) para delimitar nomes de obxectos con carácteres non permitidos.
* Dentro dunha mesma base de datos non pode haber nomes de táboas repetidas. Para facer referencia a unha táboa dunha determinada base de datos, pódese utilizar a expresión: nome\_base\_de\_datos.nome\_táboa.
* Dentro dunha mesma táboa non pode haber nomes de columnas repetidas. Para facer referencia a unha columna dunha determinada táboa, pódese utilizar a expresión: nome\_táboa.nome\_columna.
  + - 1. Base de datos

Creación dunha base de datos

A sintaxe da sentenza para a creación da base de datos é:

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nome\_da\_base [opcións\_de\_creación] ...

* DATABASE e SCHEMA son sinónimos. Pódese utilizar calquera das dúas opcións.
* Cando se utiliza a parte optativa IF NOT EXISTS, créase a base de datos no caso de que non exista; se existe, móstrase unha mensaxe de advertencia (*warning*). Se non se pon esta opción, prodúcese un erro no caso de que a base de datos exista.
* O *nome\_da\_base* debe cumprir as normas vistas para nomear obxectos.
* As posibles opcións de creación son:

[DEFAULT] CHARACTER SET [=] nome\_xogo\_carácteres

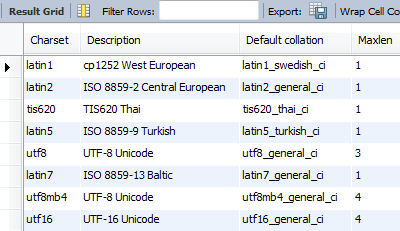
[DEFAULT] COLLATE [=] nome\_sistema\_ordenación

* A cláusula CHARACTER SET establece o conxunto de carácteres por defecto para as táboas que se crean nesa base de datos; no caso de non especificalo, tomarase o que ten definido por defecto na configuración do servidor.

Para ver os xogos de carácteres permitidos en MySQL, pódese executar a sentenza:

**show character set;**

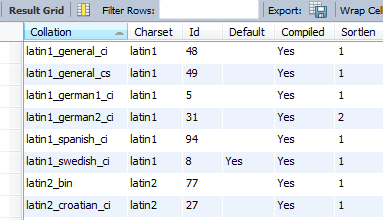
Ao executar esta sentenza, móstrase unha táboa que conten catro columnas: o nome do xogo de carácteres, a descrición, o sistema de colación por defecto e a lonxitude máxima en bytes por cada carácter. Exemplo de captura dos resultados mostrados:



Os xogos de carácteres nacionais requiren un byte para cada carácter. O xogo de carácteres Unicode (utf8, utf16 e utf32), que permiten gardar texto en calquera alfabeto ou conxunto de carácteres do mundo, poden necesitar máis espazo. Por exemplo, en MySQL o xogo de carácteres utf8 pode ser: utf8 e utf8mb4; o primeiro utiliza dende un ata tres bytes por carácter e o segundo utiliza dende un ata catro bytes por carácter.

* A cláusula COLLATE establece o conxunto de regras para comparar carácteres. Para ver os sistemas de colación permitidos en MySQL, pódese executar a sentenza:

**show collation;**



No caso de non especificar ningún sistema de colación de maneira explícita, o sistema colle o que corresponde por defecto ao xogo de carácteres elixido.

Exemplo de creación dunha base de datos:

**create** **database** **if** **not** **exists** proba

**default** **character** **set** utf8

**default** **collate** utf8\_spanish\_ci**;**

Esta sentenza crea a base de datos chamada *proba*, co xogo de carácteres por defecto *utf8* e o sistema de colación (utilizado para comparacións) *utf8\_spanish\_ci*. Os obxectos que se crean nesa base de datos utilizarán ese xogo de carácteres e ese sistema de comparación, se non se especifica ningún outro no momento da creación do obxecto.

* Tarefa 2. Crear bases de datos empregando a sentenza CREATE.

Poñer en uso unha base de datos

A sintaxe da sentenza para seleccionar unha base de datos e poder traballar con ela é:

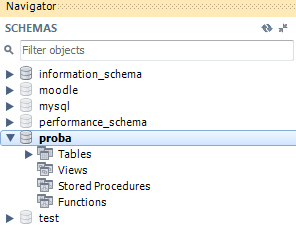
USE nome\_da\_base

Todas as sentenzas que se executen despois da sentenza USE afectan só a esa base de datos. Exemplo:

**use** proba;

Tamén se poden cualificar os nomes dos obxectos afectados por unha sentenza indicando de maneira explícita o nome da base de datos á que vai a afectar a sentenza, engadíndolle o nome da base de datos diante do nome do obxecto. Exemplo: practicas1.grupo fai referencia o obxecto *grupo* que pertence a base de datos *practicas1*.

Cando se traballa utilizando MySQL Workbench como cliente gráfico, pódese poñer en uso unha base de datos pinchando dúas veces sobre o nome da base de datos na zona de navegación. O nome da base de datos en uso aparece destacado en letra negriña.



* Tarefa 3. Poñer en uso as bases de datos empregando a sentenza USE.

Borrar unha base de datos

A sintaxe da sentenza para borrar unha base de datos é:

DROP DATABASE [IF EXISTS] nome\_da\_base

Cando se borra unha base de datos, bórranse todos os obxectos que contén a base de datos e non se poden recuperar. Exemplo:

**drop** **database** **if** **exists** proba;

* Tarefa 4. Borrar bases de datos empregando a sentenza DROP.
  + - 1. Táboa

Creación dunha táboa

Para crear unha táboa, hai que darlle un nome, definir as columnas que vai a ter e elixir as opcións da táboa e particionamento. Sintaxe resumida para MySQL:

CREATE [TEMPORARY] TABLE nome\_táboa

(

nome\_columna tipo\_de\_dato [propiedades\_da\_columna] [, ...]

)

[opcións\_de\_táboa] [opcións\_de\_partición]

* TEMPORARY é opcional, e permite crear una táboa temporal mentres dura a sesión.
* A definición de cada columna consiste en darlle un nome, asignarlle un tipo de dato e asociarlle as restricións necesarias. As dúas primeiras partes (nome e tipo) son obrigatorias.

Os SXBDR soportan un gran número de tipos de datos divididos en varias categorías: numéricos, data e hora, e cadeas de carácteres. Ver o apartado "Tipos de datos en MySQL" que se explica máis adiante.

Para definir as propiedades da columna, pódense utilizar as cláusulas:

* NOT NULL, non permite que a columna tome o valor nulo (descoñecido).
* NULL, permite que a columna tome o valor nulo.
* UNSIGNED, permite almacenar só valores positivos en columnas de tipo numérico.
* ZEROFILL, enche con ceros pola esquerda as columnas de tipo numérico.
* BINARY, diferenza entre maiúsculas e minúsculas en columnas de tipo cadea de carácteres.
* DEFAULT *valor*, indica o valor por defecto para a columna, no caso de non asignarlle ningún valor cando se engade unha fila á táboa. Está permitido utilizar algúns valores proporcionados por funcións do sistema no lugar de valores constantes definidos polo usuario. Por exemplo, USER para referirse ao usuario que estableceu a conexión, ou CURRENT\_TIMESTAMP para referirse á data e hora actual.
* AUTO\_INCREMENT, indica que o valor que toma a columna é calculado polo sistema caso de non asignarlle ningún valor á columna ou en caso de asignarlle o valor nulo. O sistema calcula o valor sumándolle 1 ao valor que toma a columna para a última fila engadida.
* PRIMARY KEY, define a columna como clave primaria. Tamén se pode definir como clave primaria creando unha restrición.
* Algunhas opcións de táboa son:

[DATA DIRECTORY= 'directorio']

[INDEX DIRECTORY= 'directorio']

[{ENGINE | TYPE} = {ISAM, MyISAM, InnodDB, ...}]

[[DEFAULT] CHARACTER SET nome\_xogo\_carácteres]

[COLLATE nome\_sistema\_colación]]

[AUTO\_INCREMENT = número],

[COMMENT texto]

* DATA DIRECTORY e INDEX DIRECTORY, especifican as rutas absolutas nas que se almacenan os datos e os índices.
* ENGINE, indica o motor de almacenamento asociado á táboa (TYPE está en desuso nas últimas versións). Ver o apartado "Motores de almacenamento en MySQL" que se explica máis adiante.
* CHARACTER SET e COLLATE, permiten establecer o conxunto de carácteres por defecto para as columnas que se crean nesa táboa, e a forma en que se ordenan e comparan os carácteres.
* AUTO\_INCREMENT, indica o número de comezo para a columna de tipo autoincremental.
* As opcións de particionamento explicaranse no apartado "Táboas particionadas".

Exemplo de código para crear unha táboa sinxela na base de datos *practicas1*:

**create** **table** test.fotografia # nome de táboa cualificado

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

titulo **varchar(**60**) not null default 'sen título',**

autor **varchar(**60**) not null comment 'Apelidos e nome do autor ou autores',**

data **date null comment 'Data en que foi tomada',**

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

**)** **engine** **=** InnoDB**;**

Tipos de datos en MySQL

A elección do tipo de columna faise en función dos datos que se queren almacenar nela. MySQL soporta unha gran cantidade de tipos de datos; a maioría cumpren o estándar ANSI e outros son propios de MySQL. A seguinte táboa contén un resumo dos tipos de datos extraído do manual de referencia de MySQL.

|  |  |  |
| --- | --- | --- |
| Resumo dos tipos de datos para MySQL, e atributos que poden ter asociadas | | |
| Numéricos | Enteiros | TINYINT[(tamaño)] [UNSIGNED] [ZEROFILL]  SMALLINT[(tamaño)] [UNSIGNED] [ZEROFILL]  MEDIUMINT[(tamaño)] [UNSIGNED] [ZEROFILL]  INT[(tamaño)] [UNSIGNED] [ZEROFILL]  INTEGER[(tamaño)] [UNSIGNED] [ZEROFILL]  BIGINT[(tamaño)] [UNSIGNED] [ZEROFILL] |
| Non enteiros | REAL[(tamaño,decimais)] [UNSIGNED] [ZEROFILL]  DOUBLE[(tamaño,decimais)] [UNSIGNED] [ZEROFILL]  FLOAT[(tamaño,decimais)] [UNSIGNED] [ZEROFILL]  DECIMAL(tamaño,decimais) [UNSIGNED] [ZEROFILL]  NUMERIC(tamaño,decimais) [UNSIGNED] [ZEROFILL]  BIT[(tamaño)]] |
| Data e hora | | DATE  TIME  TIMESTAMP  DATETIME |
| Cadeas de carácteres | | CHAR(tamaño) [BINARY | ASCII | UNICODE]  VARCHAR(tamaño) [BINARY]  BINARY(tamaño)  VARBINARY(tamaño)  TINYBLOB  BLOB  LONGBLOB  TINYTEXT [BINARY]  TEXT [BINARY]  MEDIUMTEXT [BINARY]  LONGTEXT [BINARY] |
| Outros tipos | | ENUM(valor1,valor2, valor3,...)  SET(valor1,valor2,valor3,...) |

As seguintes táboas conteñen resumos extraídos do manual de referencia de MySQL sobre o almacenamento requirido para algúns tipos de datos. No caso dos datos de tipo enteiro, móstrase o rango de valores posibles tanto para os enteiros con signo coma sen signo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resumo de tipos enteiros para MySQL | | | | |
| Tipo | Bytes | Valor mínimo con signo e sen signo | | Valor máximo con signo e sen signo |
| TINYINT | 1 | -128  0 | | 127  255 |
| SMALLINT | 2 | -32768  0 | | 32767  65535 |
| MEDIUMINT | 3 | -8388608  0 | | 8388607  16777215 |
| INT | 4 | -2147483648  0 | | 2147483647  4294967295 |
| BIGINT | 8 | -9223372036854775808  0 | | 9223372036854775807  18446744073709551615 |
| Resumo doutros tipos numéricos non enteiros para MySQL | | | | |
| Tipo | | Almacenamento requirido | | |
| FLOAT(p) | | 4 bytes se 0 <= p <= 24, 8 bytes se 25 <= p <= 53 | | |
| FLOAT | | 4 bytes | | |
| DOUBLE [PRECISION], REAL | | 8 bytes | | |
| DECIMAL(M,D), NUMERIC(M,D) | | Dende MySQL 5.0.3, os valores para columnas DECIMAL máis longos represéntanse usando un formato binario que empaqueta nove díxitos decimais en catro bytes. O almacenamento para a parte enteira e decimal determínanse separadamente. Cada múltiplo de nove díxitos require catro bytes, e o díxito "de resto" require algunha fracción de catro bytes | | |
| BIT(M) | | Aproximadamente (M + 7)/8 bytes  Para especificar valores tipo bit, para os valores pódese usar a notación b'value'. Exemplo: b'1101' | | |
| Resumo de tipos data e hora para MySQL | | | | |
| Tipo | | Almacenamento requirido | | |
|  | | Ata a versión MySQL 5.6.4 | Dende a versión MySQL 5.6.4 | |
| DATE | | 3 bytes | 3 bytes | |
| DATETIME | | 8 bytes | 5 bytes + parte fraccionaria (1 byte cada dous díxitos de precisión) | |
| TIMESTAMP | | 4 bytes | 4 bytes + parte fraccionaria (1 byte cada dous díxitos de precisión) | |
| TIME | | 3 bytes | 3 bytes + parte fraccionaria (1 byte cada dous díxitos de precisión) | |
| YEAR | | 1 byte | 1 byte | |
| Resumo de tipos cadea de carácteres para MySQL | | | | |
| Tipo | | Almacenamento requirido | | |
| CHAR(M), BINARY(M) | | M bytes, 0 <= M <= 255 | | |
| VARCHAR(M), VARBINARY(M) | | L+1 bytes, se M está entre 0 e 255, e L+2 se M é maior que 255  L representa o número de carácteres da cadea que garda. A lonxitude máxima é de 65.532 bytes | | |
| TINYBLOB, TINYTEXT | | L +1 byte, onde L < 2^8 | | |
| BLOB, TEXT | | L +2 bytes, onde L < 2^16 | | |
| MEDIUMBLOB, MEDIUMTEXT | | L+ 3 bytes, onde L < 2^24 | | |
| LONGBLOB, LONGTEXT | | L +4 bytes, onde L < 2^32 | | |
| ENUM('value1','value2',...) | | 1 o 2 bytes, dependendo do número de valores da enumeración (65.535 valores como máximo) | | |
| SET('value1','value2',...) | | 1, 2, 3, 4, o 8 bytes, dependendo do número de membros do conxunto (64 membros como máximo) | | |

Algunhas consideracións sobre estes tipos:

* MySQL soporta os tipos do estándar SQL para valores de tipo enteiro, INTEGER (ou INT) e SMALLINT. Como unha extensión ao estándar, MySQL engade os tipos TINYINT, MEDIUMINT e BIGINT.

MySQL soporta unha extensión que permite especificar o ancho en pantalla dos datos de tipo enteiro, poñendo entre parénteses un número que representa o ancho de pantalla para a columna. O ancho de pantalla non limita o rango de valores que se poden almacenar na columna. Cando se utiliza o ancho de pantalla en combinación coa opción ZEROFILL, o recheo con ceros pola esquerda para cantidades con menos díxitos que os especificados como ancho de pantalla, faise ata o tamaño do ancho de pantalla. Exemplo:

fillos **tinyint(2) unsigned zerofill**

A columna *fillos*  almacena valores numéricos enteiros que poden tomar valores entre 0 e 255, pero o 2 indica o ancho de pantalla que pode ser utilizado polas aplicacións que manexan o dato. Se toma o valor 3, móstrase como 03.

* Para poder almacenar valores numéricos con cifras decimais exactas, tamén coñecidos de coma fixa, MySQL dispón dos tipos de datos que son sinónimos, DECIMAL (ou DEC), e NUMERIC. Sintaxe:

DECIMAL(m,d)

* *m* é o número máximo de cifras en total (a escala). O rango permitido é de 1 a 65.
* *d* é o número de cifras despois de la coma (a precisión). O rango permitido é de 0 a 30 e non pode tomar un valor maior que *m*.

Se non se especifica o tamaño, o valor por defecto para *m* é 10, e para *d* é 0. Exemplo:

salario **decimal(7,2)**

A columna *salario* almacena valores numéricos que poden ter un máximo de 7 díxitos, e o 2 indica que poden ter dous decimais. O rango de valores permitido é de -99999.99 a 99999.99.

* Para almacenar valores decimais aproximados, tamén coñecidos como de coma flotante, MySQL dispón dos tipos FLOAT (para simple precisión) e DOUBLE (para dobre precisión).
* Dende MySQL 5.0.3, o tipo de datos BIT pode usarse para gardar valores dun bit. Un tipo BIT(M) permite o almacenamento de valores de M-bit. M ten un rango de 1 a 64. Para especificar valores bit, pódese usar a notación b'*value*'. A palábra *value* representa un valor binario escrito usando ceros e uns. Exemplo, b'111' e b'100000000' representan 7 e 128, respectivamente.

Cando se asigna un valor a unha columna BIT(M) con menos de M bits, o valor complétase pola esquerda con ceros. Exemplo, ao asignar un valor b'101' a unha columna BIT(6) é o mesmo que asignar b'000101'.

En versións anteriores, o tipo BIT era equivalente a un TINYINT.

* Para poder almacenar valores que representen datas e horas, MySQL dispón dos tipos DATE, TIME, DATETIME e TIMESTAMP.
* O tipo DATE utilízase para almacenar datas co formato 'AAAA-MM-DD'. O rango permitido é '1000-01-01' a '9999-12-31'. Admite utilizar guión (-) ou barra como separador (/), ou ben, omitir o separador.
* O tipo TIME utilízase para representar horas. Ten o formato HH:MM:SS[.fracción].
* Os tipos DATETIME e TIMESTAMP utilízanse para valores que conteñen data e hora co formato 'AAAA-MM-DD HH:MM:SS[.fracción]'. O rango permitido para o tipo DATETIME é '1000-01-01 00:00:00' a '9999-12-31 23:59:59', e para o tipo TIMESTAMP é '1970-01-01 00:00:01' a '2038-01-19 03:14:07'. Admítese utilizar guión (-) ou barra como separador (/) para a data.
* A partir da versión MySQL 5.6.4, os tipos TIME, DATETIME e TIMESTAMP teñen a posibilidade de gardar fraccións de segundo ata o microsegundo (6 díxitos) como máxima precisión. Hai que indicar o número de decimais entre parénteses a continuación do nome do tipo. Exemplo:

nome\_columna **timestamp(2)**

O 2 que vai entre parénteses indica que ten unha precisión de décimas de segundo.

* O tipo YEAR ocupa 1 byte e é usado para representar valores de anos. Pode ser declarado como YEAR(4) ou como YEAR(2).
* Os tipos CHAR e VARCHAR permiten almacenar cadeas de carácteres e diferéncianse no tipo de lonxitude (o primeiro ten lonxitude fixa e o segundo ten lonxitude variable) e na lonxitude máxima para as columnas (para o tipo CHAR é 255 carácteres e para o tipo VARCHAR é de 65532). Este tipo de columnas non diferencian entre minúsculas e maiúsculas, a non ser que se lles asocie a opción BINARY.

Para o tipo VARCHAR, os requirimentos de almacenamento dependen da lonxitude da cadea almacenada e o valor que vai entre parénteses representa a lonxitude máxima que pode ter a cadea, en carácteres. Exemplo:

|  |  |  |
| --- | --- | --- |
| Tipo de columna | Valor asignado | Bytes que ocupa |
| CHAR(30) | ‘pepe’ | 30 |
| VARCHAR(30) | ‘pepe’ | 5 |

O tipo VARCHAR aproveita mellor o espazo de almacenamento, pero ten como contrapartida que require máis tempo de proceso do servidor.

* Os tipos BINARY e VARBINARY son similares a CHAR e VARCHAR, pero almacénanse como cadeas de carácteres binarias. Conteñen cadeas de bytes, en lugar de cadeas de carácteres, polo que diferencian as minúsculas das maiúsculas.
* Os tipos BLOB e TEXT son de lonxitude variable e permiten tratar unha gran cantidade de datos variables. As columnas tipo BLOB trátanse como cadeas de carácteres binarias (de byte). As columnas tipo TEXT trátanse como cadeas de carácteres non binarias (de carácteres). Cada valor BLOB ou TEXT represéntase internamente como un obxecto a parte.

As columnas tipo BLOB non teñen conxunto de carácteres e ordénanse e compáranse baseándose nos valores numéricos dos bytes.

Hai catro tipos BLOB e catro tipos TEXT que se diferencian entre eles só na lonxitude máxima dos valores que poden tratar.

* Tipos BLOB: TINYBLOB, BLOB, MEDIUMBLOB e LONGBLOB.
* Tipos TEXT: TINYTEXT, TEXT, MEDIUMTEXT e LONGTEXT.

As columnas tipo BLOB son obxectos binarios que poden almacenar calquera tipo de información, dende un arquivo de texto, una imaxe, arquivos de son ou vídeo.

* Os tipos ENUM e SET representan unha cadea de carácteres á que se engade unha restrición de valor, indicando entre parénteses a lista de valores permitidos separados por comas. No caso de ENUM, só pode almacenar un deses valores; no caso de SET, pode almacenar máis dun deses valores. Exemplos:

talla **enum('superpequena','pequena','mediana','grande','supergrande')**

aficions **set('deporte','fotografía','lectura','maquetismo')**

* A columna *talla* pode almacenar unicamente algún dos cinco valores representados na lista, ou ben o valor nulo. Pode estar asociada a un campo tipo *radio* dun formulario HTML.
* A columna *aficions* pode almacenar un ou máis dos valores da lista. Pode estar asociada a un campo tipo *checkbox* dun formulario HTML.

Estes tipos de columnas funcionan dunha maneira similar aos *strings* de cadeas de carácteres. Internamente almacénanse como valores enteiros que representan a posición do valor dentro da lista, o que supón un importante aforro de espazo de almacenamento. Exemplo: Se a táboa na que está a columna *talla* tivese un millón de filas que toman o valor 'mediana' nesa columna, requírense 1 millón de bytes de almacenamento, en lugar dos 8 millóns que se utilizarían para almacenar o valor 'mediana' se a columna fose definida como tipo VARCHAR.

A elección do tipo de columna é unha tarefa delicada que require prestarlle atención antes de empezar a crear as táboas. Hai que ter en conta dous factores que poden influír na decisión: o espazo de almacenamento utilizado polo tipo de columna e o tempo de proceso que require o servidor para manexar ese tipo de columna.

O manual de Mysql contén recomendacións para optimizar a estrutura dunha base de datos, facendo referencia especial ao tamaño dos datos e aos tipos de columnas[[2]](#footnote-4).

Implementación da integridade mediante restricións

Un paso importante no deseño dunha base de datos é decidir a mellor forma de implementar a integridade dos datos. A integridade fai referencia á coherencia e a precisión dos datos que están almacenados nunha base de datos. Tipos de integridade:

* Integridade de dominio (ou columna). Especifica o conxunto de valores de datos que son válidos para unha columna e se admite valores nulos. A integridade de dominio adóitase implementar mediante o uso de comprobacións de validez e tamén mediante a restrición do tipo de datos, o formato ou o intervalo dos valores posibles permitidos nunha columna.
* A integridade de entidade (ou táboa). Require que todas as filas dunha táboa teñan un identificador exclusivo, coñecido como clave primaria. A modificación do valor da clave primaria ou a eliminación da fila enteira depende do nivel de integridade requirido entre a clave primaria e calquera outra táboa.
* Integridade referencial. Esta integridade asegura que sempre se manteñen as relacións entre as claves principais (na táboa á que se fai referencia) e as claves externas (nas táboas que fan referencia). Non se pode eliminar unha fila, nin se pode modificar a clave primaria, se unha clave externa fai referencia á fila, salvo que se permita a acción 'en cascada'. Pódense definir relacións de integridade referencial entre táboas diferentes ou entre columnas da mesma táboa.

No seguinte gráfico ilústranse os tres tipos de integridade:

RestricionsIntegridade.emf

Métodos para implementar a integridade

A integridade dos datos pódese esixir mediante dous métodos: integridade de datos declarativa ou integridade de datos procedimental.

* Coa integridade declarativa, defínense os criterios que teñen que cumprir os datos como parte da definición dun obxecto; despois, o xestor asegura automaticamente que os datos cumpran eses criterios. O método preferido para implementar a integridade de datos básica é a integridade declarativa. Hai que ter en conta os feitos seguintes sobre o método declarativo:
* Declárase como parte da definición da base de datos, mediante o uso de restricións declarativas que se definen directamente nas táboas e nas columnas.
* Impleméntase mediante a utilización de restricións, valores predeterminados e regras.
* Coa integridade procedimental, escríbense secuencias de comandos que definen os criterios que teñen que cumprir os datos e comproban que os devanditos criterios se cumpren. Débese limitar o uso da integridade procedimental a situacións excepcionais e a aquelas cunha lóxica complicada. Por exemplo, utilízase a integridade procedimental cando se desexa implementar un tipo de eliminación do que non dispón o xestor. Os feitos seguintes aplícanse á integridade procedimental:
* Pódese implementar no cliente ou no servidor mediante outras linguaxes e ferramentas de programación.
* Impleméntase utilizando disparadores (*triggers*) e procedementos almacenados.

Definición de restricións

As restricións son o método máis adecuado para conseguir a integridade dos datos e son un método estándar ANSI para implementar a integridade dos datos. Cada tipo de integridade de datos (dominio, entidade e referencial) impleméntase con tipos de restricións diferentes. As restricións aseguran que os datos que se escriben nas columnas sexan válidos e que se manteñan as relacións entre as táboas. A táboa seguinte describe os diferentes tipos de restricións.

|  |  |  |
| --- | --- | --- |
| Tipos de integridade | Tipo de restrición | Descrición |
| **Dominio** | DEFAULT | Especifica o valor que se proporciona para a columna cando non se especifica explicitamente cando se insire unha nova fila. |
| CHECK | Especifica mediante unha expresión os valores dos datos que se aceptan nunha columna. |
| REFERENTIAL | Especifica os valores de datos que se aceptan como actualización en función dos valores dunha columna doutra táboa |
| **Entidade** | PRIMARY KEY | Identifica de forma exclusiva cada unha das filas; asegura que os usuarios non escriban valores duplicados e que se cre un índice para aumentar o rendemento. Non se permiten valores nulos. |
| UNIQUE | Impide a duplicación de claves alternativas (non principais) e asegura que se cree un índice para aumentar o rendemento. |
| **Referencial** | FOREIGN KEY | Define unha columna ou combinación de columnas nas que os seus valores coinciden coa clave primaria da mesma ou outra táboa. |
| CHECK | Especifica os valores dos datos que se aceptan nunha columna en función dos valores doutras columnas da mesma táboa. |

Creación de táboas e restricións

Dende a norma ANSI SQL2, na sentenza CREATE TABLE, ademais da definición de columnas, pódense introducir algunhas cláusulas que nos permiten completar a descrición da táboa, introducindo algunhas restricións. Entre estas cláusulas merecen ser destacadas as seguintes:

* Restrición de clave primaria. Sintaxe:

[CONSTRAINT nome\_restrición] PRIMARY KEY [tipo\_índice] (nome\_de\_columna [, ...])

Define unha clave primaria nunha táboa para identificar de forma exclusiva cada unha das súas filas. Hai que ter en conta que:

* Só se pode definir unha restrición PRIMARY KEY por táboa.
* A clave primaria pode ser unha columna ou unha combinación de columnas.
* Os valores que se gardan na clave primaria teñen que ser exclusivos. Non pode haber dúas filas que tomen o mesmo valor para a clave primaria.
* Ningunha columna que forme parta da clave primaria pode tomar o valor NULL.
* Restrición de clave foránea (allea ou externa). Sintaxe:

[CONSTRAINT nome\_restrición] FOREING KEY ( nome\_de\_columna [, ...] )

REFERENCES nome\_de\_táboa (nome\_de\_columna [,...])

[{ON DELETE|ON UPDATE} {RESTRICT|SET NULL|SET DEFAULT|CASCADE}]

É importante recordar que a clave foránea e a clave primaria á que fai referencia teñen que pertencer ao mesmo dominio (mesmo tipo, tamaño e propiedades). O código de erro 150, móstrase cando non se pode crear unha clave foránea e a maioría das veces prodúcese porque non se cumpre esta restrición.

A restrición FOREIGN KEY implementa a integridade referencial. Esta restrición define unha columna que fai referencia a unha columna que ten que ter unha restrición PRIMARY KEY ou UNIQUE na mesma ou noutra táboa.

Hai que ter en conta que:

* A clave foránea pode estar formada por unha ou máis columnas. O número de columnas e os tipos de datos que se especifican na instrución FOREIGN KEY ten que coincidir co número de columnas e os tipos de datos da cláusula REFERENCES.
* As columnas definidas como clave foránea deben ter asociado un índice para optimizar o seu rendemento. A partir da versión 5.0 de MySQL, cando se crea unha clave foránea créase o índice de forma automática.
* Cando se intenta realizar unha operación de modificación ou borrado dunha fila da táboa pai (a que figura en REFERENCES) e hai filas nalgunha táboa que conteñen claves foráneas que sinalan a esa fila, terase en conta a acción especificada nos apartados ON UPDATE e ON DELETE da cláusula FOREIGN KEY. As accións permitidas para conservar a integridade referencial nas operacións de borrado (DELETE) e modificación (UPDATE), son:
* CASCADE: borra ou actualiza a fila na táboa pai, e automaticamente borra ou actualiza as filas con claves foráneas que fan referencia a esa fila. Exemplo: No caso de cambiar o valor da clave primaria para unha fila na táboa de clientes, cámbiase 'en cascada' o valor da columna definida como clave foránea na táboa de facturas para facer referencia ao cliente ao que pertence a factura, nas filas que fan referencia a ese cliente.

**constraint** fk\_facturas\_clientes **foreign key** **(**cliente**)** **references** clientes **(**id**)**

**on** **delete** **restrict**

**on** **update** **cascade**

* SET NULL: borra ou actualiza a fila na táboa pai e garda o valor NULL na columna que é clave foránea e fai referencia a esa fila. Para poder empregar esta opción, a columna que é clave foránea debe permitir almacenar o valor NULL, é dicir, non ter NOT NULL na súa definición.
* NO ACTION: no estándar ANSI SQL-92, a opción NO ACTION significa que non está permitido borrar ou modificar unha fila na táboa pai, se hai algunha fila que teña unha clave foránea que faga referencia a ela.
* RESTRICT: en MySQL, NO ACTION e RESTRICT son equivalentes.
* SET DEFAULT: borra ou actualiza a fila na táboa pai e garda o valor definido por defecto para a columna que é clave foránea e fai referencia a esa fila. Non está implementada en MySQL.
* Restricións de unicidade para crear índices que non permiten valores repetidos nunha columna. Sintaxe:

[CONSTRAINT nome\_restrición] UNIQUE [INDEX|KEY]

[nome\_de\_índice] [tipo\_índice] (nome\_de\_columna [, ...])

A restrición UNIQUE ou de clave única especifica que dúas filas non poden ter o mesmo valor na columna. Esta restrición implementa a integridade de entidade creando un índice único para a columna. É útil cando xa se ten unha clave primaria, como por exemplo un identificador de empregado, pero se desexa garantir que outras columnas, como o número do permiso de conducir dun empregado ou o DNI, tamén sexan exclusivos.

**constraint** pk\_empregado **primary** **key** **(**identificador**),**

**constraint** iu\_empregado\_dni **unique** **(**dni**), # a columna dni non admite valores repetidos**

Hai que ter en conta que:

* Pode permitir o valor nulo.
* Pode haber varias restricións UNIQUE nunha mesma táboa.
* Pode aplicar a restrición UNIQUE a unha columna ou a unha combinación de varias columnas que teñen que ter valores exclusivos.
* Restricións CHECK para comprobación de valores do dominio válidos.

CHECK (expresión\_lóxica).

Restrinxe os valores que se poden almacenar nunha columna, que teñen que cumprir a condición representada pola expresión lóxica. Exemplo: o código de oficina non pode ter un valor inferior a 1, nin superior a 100, é dicir, que o código da oficina estea nun rango comprendido entre 1 e 100.

**check** **(**codOficina **between** 1 **and** 200**),**

Hai que ter en conta que:

* A restrición comproba os datos cada vez que se insire unha nova fila (INSERT), ou cando se modifica unha fila que xa existe (UPDATE).
* A expresión pode facer referencia a outras columnas da mesma táboa.
* A expresión non pode conter subconsultas.

Esta restrición non está implementada en MySQL actualmente; non dá erro de sintaxe pero non fai a comprobación de valores.

Para poder borrar ou modificar as restricións, hai que facer referencia ao nome da restrición, polo que é recomendable asignarlles un nome no momento que se crean empregando unha regra. No caso de non darlles un nome, MySQL asígnalles un nome de maneira automática. Unha regra posible para nomear as restricións podería ser:

* Para claves primarias: pk\_nomeTáboa.
* Para claves foráneas fk\_nomeTáboa\_nomeTáboareferenciada.
* Para índices únicos iu\_nomeTaboa\_nomeColumna.

Exemplo de sentenza para a creación da táboa de oficinas tendo en conta as seguintes restricións de integridade impostas polo modelo:

* A clave primaria é a columna *num\_oficina*.
* A columna *num\_oficina* só permite valores entre 1 e 200, ambos incluídos.
* En cada cidade só pode haber unha oficina.
* A columna *provincia* toma por defecto o valor 'Lugo'.
* A columna *director* é unha clave foránea que fai referencia á táboa *empregado*. Contén o *id* do empregado que dirixe a oficina. No caso de executar esta sentenza podudiríase un erro porque á táboa *empregado* non existe.

**create** **table** oficina

**(**

num\_oficina **integer** **not** **null,**

nome **varchar(**60**),**

enderezo **varchar(**60**),**

codpostal **char(**5**),**

director **integer,**

cidade **char(**20**)** **,**

provincia **char(**20**)** **default** "Lugo"**,**

salario **decimal(**8**,**2**),**

**constraint** pk\_oficina **primary** **key** **(**num\_oficina**),**

**constraint** iu\_oficina\_cidade **unique** **(**cidade**), # só unha oficina por cidade**

**constraint** fk\_oficina\_empregado **foreign key** **(**director**)** **references** empregado **(**id**)**

**on** **delete** **set** **null**

**on** **update** **cascade,**

**check** **(**num\_oficina **between** 1 **and** 200**)**

**)** **engine** **=** InnoDB**;**

Pódense crear novas restricións e modificar ou borrar as restricións que xa existen utilizando a sentenza ALTER TABLE, que se ve na seguinte actividade.

Asociar estruturas de índices ás columnas

No momento da creación da táboa, pódense crear estruturas de índices asociadas ás columnas, utilizando a cláusula INDEX. Sintaxe:

{KEY|INDEX} [nome\_de\_índice] (nome\_de\_columna[(lonxitude)] [ASC|DESC] [, ...])

Os índices serven para axilizar as operacións de busca. Poden estar formados por máis dunha columna ou incluso por parte dunha columna. MySQL permite seleccionar a orde do índice (ascendente ou descendente). Exemplos:

-- índice asociado á columna nome

**index** idx\_oficina\_nome (nome)

-- índice asociado aos 15 primeiros carácteres da columna enderezo

**key** idx\_oficina\_enderezo (enderezo(15))

-- índice asociado á combinación das columnas numero\_oficina e nome en orde descendente

**index** idx\_oficina\_numeroNome (numero\_oficina,nome) **desc**

Para poder borrar ou modificar as estruturas de índices, hai que facer referencia ao nome do índice, polo que é recomendable asignarlle un nome empregando unha regra. Exemplo: *idx\_ nomeTaboa\_nomeColumna*. No caso de non asignarlle un nome, o sistema asígnalle un de forma automática.

Información sobre táboas

Para obter información sobre as táboas creadas, e as restricións asociadas a elas, pódense utilizar sentenzas SHOW propias de MySQL, ou as táboas da base de datos estándar ANSI INFORMATION\_SCHEMA. Exemplos:

* Coa sentenza SHOW, propia de MySQL:

**show tables from** practicas1; # mostra os nomes das táboas da base de datos practicas1

**show create table** fotografia; # mostra a sentenza CREATE TABLE para esa táboa

* Consultando a base de datos INFORMATION\_SCHEMA:

-- consulta das táboas do servidor

**select** **\*** **from** INFORMATION\_SCHEMA**.**TABLE

-- restricións de táboa: claves primaria, únicas e foráneas

**select** **\*** **from** INFORMATION\_SCHEMA**.**TABLE\_CONSTRAINTS

-- restricións de integridade referencial

**select** **\*** **from** INFORMATION\_SCHEMA**.**REFERENTIAL\_CONSTRAINTS

Na seguinte táboa móstranse os nomes e a información que conteñen algunhas das táboas da base de datos INFORMATION\_SCHEMA que poden ser de utilidade nesta actividade:

|  |  |  |
| --- | --- | --- |
| Nome da táboa | Información que contén | Equivalencia SHOW |
| SCHEMATA | Bases de datos | SHOW DATABASES |
| TABLES | Táboas da base de datos | SHOW TABLES |
| COLUMNS | Columnas das táboas | SHOW COLUMNS FROM táboa |
| TABLE\_CONSTRAINTS | Restricións de táboas |  |
| REFERENTIAL\_CONSTRAINTS | Restricións referenciais |  |
| ENGINES | Motores de almacenamento | SHOW ENGINES |
| STATISTICS | Información sobre os índices | SHOW INDEX FROM táboa |

Creación dunha táboa a partir dunha táboa que xa existe

MySQL permite crear unha táboa partindo do esquema doutra táboa que xa existe. Sintaxe:

CREATE [TEMPORARY] TABLE nome\_táboa {LIKE nome\_táboa\_existente | sentenza SELECT}

Consideracións sobre a sintaxe anterior:

* No caso de utilizar LIKE, créase unha táboa baleira baseada na definición doutra táboa, incluíndo algúns atributos das columnas e os índices definidos na táboa orixinal.
* No caso de empregar unha sentenza SELECT, créase a estrutura da táboa e cópianse os datos resultantes da sentenza. No caso de producirse un erro cando se copian só datos, non se crea a táboa.

En calquera dos dous casos hai atributos que non se van a copiar, e haberá que telo en conta[[3]](#footnote-5). Algunha das cousas a ter en conta cando se crea unha táboa partindo doutra:

* Claves foráneas. As restricións FOREIGN KEYS non se copian en ningún dos casos. Pódense engadir utilizando unha sentenza ALTER TABLE para incluílas despois de facer a copia.
* En táboas co motor de almacenamento MyISAM, as opcións de táboa DATA DIRECTORY e INDEX DIRECTORY tampouco se copian. Os ficheiros de datos e índices gárdanse no directorio de datos especificado para a base de datos destino.

Exemplo:

**create** **table** copia\_oficina **like** oficina**;**

* Tarefa 5. Crear táboas empregando a sentenza CREATE.

Borrado dunha táboa

A sentenza DROP TABLE permite borrar táboas. Sintaxe:

DROP [TEMPORARY] TABLE [IF EXISTS] nome\_táboa [, nome\_táboa] ...

Exemplo:

**drop** **table** copia\_oficina**;**

* Tarefa 6. Borrar táboas empregando a sentenza DROP.

Motores de almacenamento en MySQL

Unha característica que distingue a MySQL doutros SXBDR é a dispoñibilidade de distintos motores de almacenamento, que ofrecen distintas prestacións e características. Cando se empezou a desenvolver MySQL, o seu creador Michael Windenis buscaba un xestor de bases de datos que fora moi rápido nas consultas xa que ningún dos que había no mercado era o suficientemente rápido. Por iso decidiu crear un novo motor de consultas SQL baseado no formato ISAM[[4]](#footnote-6) que son as iniciais de *Indexed Sequential Access Method* (Método de Acceso Secuencial Indexado), pero engadindo novas extensións que dan lugar ao primeiro motor de almacenamento de MySQL que se coñece co nome MyISAM. Posteriormente fóronse incorporando outros motores, atendendo ás necesidades dos usuarios.

Nunha mesma base de datos pódense crear táboas que utilicen distintos motores de almacenamento.

Pódese consultar os motores de almacenamento que pode manexar o servidor utilizando a sentenza SHOW ENGINES (é unha sentenza propia de MySQL), ou consultando a táboa *engines* da base de datos INFORMATION\_SCHEMA. Só se verán os motores de almacenamento que se habilitaron no momento da compilación da distribución.

Cando se crea unha táboa, a cláusula ENGINE permite indicar o motor de almacenamento asociado á táboa. No caso de non utilizar esta cláusula opcional, o servidor asócialle á táboa o motor de almacenamento predeterminado que indica a variable de sistema default-storage-engine. Na versión MySQL 5.6 o motor predeterminado é InnoDB.

Os motores de almacenamento máis utilizados son:

* MyISAM. As táboas que utilizan este motor de almacenamento son táboas non transaccionais, que se caracterizan pola rapidez na recuperación de datos.

As táboas deste tipo almacénanse no directorio de datos especificado na configuración do servidor (*datadir*). Cada táboa ten asociados tres ficheiros que teñen as extensións MYD para os datos, MYI para os índices, e FRM para os metadatos (esquema).

Cando se insiren ou actualizan filas, prodúcese un bloqueo a nivel de táboa, é dicir, a táboa enteira queda bloqueada e non pode ser utilizada por outros usuarios, excepto para consultas. As lecturas non producen bloqueos.

Estas táboas son as máis utilizadas nos casos en que se fan moitas consultas ao mesmo tempo e poucas actualizacións que son feitas só polo administrador. É o caso de moitas aplicacións web.

* InnoDB. As táboas que utilizan este motor de almacenamento son táboas transaccionais e con elas poden ser utilizados os comandos START TRANSATION e COMMIT ou ROLLBACK para crear transaccións ACID[[5]](#footnote-7). Permiten a recuperación automática tras un fallo grave e a utilización de claves foráneas para controlar a integridade referencial.

Utiliza funcións de bloqueo a nivel de fila; isto fai que sexan recomendables cando se fan moitas operacións de inserción e de actualización simultáneas, comparando co número de consultas.

As táboas deste tipo poden almacenarse nun único espazo de táboas compartido (*ibdata*), ou ben utilizar arquivos específicos para cada táboa como fan o resto de motores. O espazo de táboa pode estar formado por varios arquivos diferentes. Tamén se pode configurar para usar espazos de táboa individuais para cada táboa; neste caso, cada táboa ten asociado un arquivo coa extensión .ibd. O tamaño inicial deste espazo é de 16 Mb e vai medrando a medida que se gravan máis datos.

* Merge. Permite realizar consultas nun conxunto de táboas MyISAM como se foran unha mesma unidade lóxica. A colección de táboas MyISAM teñen que ter a mesma estrutura; é necesario definir as columnas e os índices das táboas cos mesmos nomes e na mesma orde. Ademais da comodidade de poder facer referencia a varias táboas sen usar varias consultas, as táboas Merge ofrecen as seguintes vantaxes:
* Permiten crear unha entidade lóxica que supere o tamaño máximo para as táboas MyISAM.
* Permite incluír táboas comprimidas no grupo.

As táboas Merge admiten operacións DELETE e UPDATE, e no caso das insercións poden ter distintos tratamentos, segundo a cláusula INSERT\_METHOD.

* Memory. Almacenan todos os datos na memoria RAM, non en disco. Antes coñecíanse como HEAP.
* Archive. Utilizadas para gardar información histórica ou información de rexistro de operacións (auditoría), xa que unha vez gravadas as filas non se poden modificar.
* CSV. Garda os datos en ficheiros de texto plano, usando formato de valores separados por comas.
* Federated. Permite o acceso remoto a táboas creadas noutro servidor. Por defecto, está desactivado no arquivo de configuración (my.ini ou my.cnf).
* NDBCLUSTER (tamén coñecido como NBD). É o motor de almacenamento utilizado por MySQL Cluster para implementar táboas que se particionan en varias máquinas.

Todos estes motores de almacenamento crean táboas non transaccionais, excepto InnoDB e NDBCLUSTER.

|  |  |
| --- | --- |
| Resumo das vantaxes das táboas transaccionais e as non transaccionais | |
| Transaccionais | * Permiten o manexo de transaccións ACID. * Máis seguras. Cando falla unha transacción por un fallo do sistema ou de hardware, pode recuperar os datos, mediante recuperación automática ou dende unha copia de seguridade máis o log de transaccións. Con táboas non transaccionais, todos os cambios son permanentes. * Permite executar o comando COMMIT para confirmar os cambios feitos por unha transacción, ou o comando ROLLBACK para ignorar os cambios (se AUTOCOMMIT está desactivado). * Proporcionan mellor rendemento para táboas nas que se realizan ao mesmo tempo un número importante de actualizacións e de lecturas. |
| Non transaccionais | * Máis rápidas. * Menor requirimento de espazo. * Requiren menos memoria para actualizacións. |

Cando se crea unha nova táboa, pódese indicar ao servidor MySQL o motor de almacenamento que se vai utilizar, engadindo a opción de táboa ENGINE no comando CREATE TABLE. Unha vez creada, pódese cambiar o motor de almacenamento coa sentenza ALTER TABLE que se explicará máis adiante. En caso de non especificar a opción ENGINE, crearase co motor de almacenamento definido por defecto na configuración do servidor na opción de arranque *default-storage-engine*, ou na variable de sistema *storage\_engine*.

Táboas particionadas

MySQL 5.1 e versións posteriores, admiten táboas particionadas. As particións son similares conceptualmente ao motor Merge, xa que permiten usar contidos de táboa almacenados en diferentes localizacións. Sen embargo, unha táboa particionada é unha única táboa, non unha construción lóxica que se asigna a varias táboas subxacentes. Ademais, poden utilizar motores de almacenamento diferentes a MyISAM.

Existen dous tipos de particionados: particionado horizontal, no que as distintas filas da táboa asígnanse a unha partición en concreto; e o particionado vertical, no que as distintas columnas da táboa asígnanse a unha partición en concreto.

Actualmente, en MySQL, só está soportado o particionamento horizontal de catro maneiras distintas:

* RANGE: a asignación das filas da táboa ás diferentes particións realízase segundo un rango de valores definido sobre unha determinada columna da táboa ou expresión. É dicir, indicarase o número de particións a crear, e para cada partición, o rango de valores válidos, de forma que cando se vai a introducir unha fila nova na táboa, compróbase o valor da columna/expresión indicada para facer a partición e a fila inserirase na partición correspondente.
* LIST: a asignación das filas da táboa ás diferentes particións realízase segundo unha lista de valores definida sobre unha determinada columna da táboa ou expresión. É dicir, indicarase o número de particións a crear, e para cada partición, a lista de valores que serán a condición para inserir nela, de forma que cando un rexistro se vaia a introducir na táboa, mirarase en que partición hai que gardalo tendo en conta a lista de valores válidos para cada partición.
* HASH: este tipo de partición está pensado para repartir de forma equitativa as filas da táboa entre as diferentes particións. Mentres nas dúas formas anteriores é unha persoa quen decide a partición á que se levan as filas segundo uns valores indicados, na partición HASH é MySql quen fai ese traballo. Para definir este tipo de particionado, indicarase unha columna do tipo integer ou unha función aplicada a unha determinada columna que devolverá un valor enteiro. En función do valor enteiro, MySql decide en que partición se insire a fila.
* KEY: similar a HASH, pero a función para o particionado a proporciona MySQL automaticamente (coa función MD5). Pódense indicar as columnas para o particionado, pero sempre han de ser clave primaria da táboa ou índice único.

Mysql permite ademais realizar subparticionado, é dicir, a división de cada partición en múltiples subparticións mediante a opción SUBPARTITIONS.

A definición de particións non é estática, é dicir, MySQL ten ferramentas para poder cambiar a configuración do particionado a posteriori, para engadir ou suprimir particións existentes, fusionar particións en outras, dividir unha partición en varias, etc.

O particionado ten as súas limitacións e as súas restricións, pois non se pode realizar sobre calquera tipo de columna ou expresión, hai un límite de particións a definir, e haberá que ter en conta algunhas cousas para mellorar o rendemento das consultas e evitar que estas recorran todas as particións dunha táboa. Por exemplo, unha táboa de vendas que almacena as vendas realizadas en distintas tendas dunha cadea particionada pola identificación de cada tenda. Ao buscar os datos da tenda de Albacete, a busca limítase á partición asignada a esta cidade, xa que sería absurdo buscar datos de vendas da tenda de Albacete en calquera outra partición.

A vantaxe principal do particionado de táboas é a optimización á hora de acceder aos datos que almacena MySQL, xa que o optimizador pode buscar en particións concretas ou en particións de forma paralela. Noutras palabras, á hora de realizar unha consulta a unha táboa particionada, en lugar de realizar buscas por toda a táboa, redúcese a busca a aquelas particións nas que se sabe que hai datos que interesan.

Outra vantaxe do particionado é que o almacenamento pódese distribuír en varios dispositivos, o que mellora o tempo de acceso por paralelismo E/S. Isto aporta moita flexibilidade se pensamos en escalar a nosa base de datos, xa que, por exemplo, podemos gardar cada partición ou cada grupo de particións en distintos discos físicos.

Hai que ter en conta que estas vantaxes teñen os seus inconvenientes, xa que por exemplo, a complexidade de administración da nosa base de datos incrementarase ao igual que o deseño das táboas. O ideal é alcanzar un equilibrio entre os beneficios que conseguiremos e o custe de manter o sistema.

Exemplo: Creación dunha táboa de *vendas* particionada polo rango de valores, tendo en conta o ano da venda (*data\_venda*):

**create** **table** vendas

**(** cliente **int** **not** **null,**

importe **decimal(**10**,**2**)** **default** **null,**

data\_venda **date** **default** **null**

**)** **engine=**myisam

**partition** **by** **range** **(year(**data\_venda**))**

**(partition** p1 **values** **less** **than** **(**2000**),**

**partition** p2 **values** **less** **than** **(**2005**)** **,**

**partition** p3 **values** **less** **than** **(**2010**),**

**partition** p4 **values** **less** **than** **(**2015**)** **,**

**partition** p5 **values** **less** **than** **maxvalue** **);**

A función year(*data\_venda*) devolve o ano da expresión tipo data. Os datos que se almacenan na táboa vendas distribúense en cinco particións. Na primeira partición (p1) gárdanse os datos das vendas feitas antes do ano 2000; na segunda (p2), as vendas feitas antes do ano 2005; na terceira (p3), as vendas feitas antes do ano 2010; na cuarta (p4), as vendas feitas antes do ano 2015; e na quinta (p5), as vendas feitas dende o ano 2015 ata o valor máis alto que tome o ano da venda.

* + - 1. Índices

Os índices son o principal medio para acelerar o acceso aos datos contidos nas táboas, en especial en aquelas consultas nas que se mostran datos de varias táboas. Hai algúns motores de almacenamento de MySQL que non permiten utilizar índices: Merge, Archive e CSV.

O SXBD sempre establece un índice para a clave primaria dunha táboa e pódense crear índices a maiores para as para as claves foráneas e as columnas que son utilizadas frecuentemente en condiciones de busca. Á hora de tomar unha decisión deste tipo, hai que ter en conta que aínda que a busca sexa máis rápida, o índice ocupa un espazo de almacenamento extra e ademais debe ser actualizado cada vez que se executa unha sentenza que insire unha fila nova e cada vez que se borre ou modifica unha fila da táboa.

A creación e borrado de índices pódese facer no momento da creación da táboa coa sentenza CREATE TABLE, ou posteriormente, utilizando a sentenza ALTER TABLE. Tamén se poden crear e modificar coas sentenzas propias para manexo de índices.

Creación de índices

A sentenza CREATE INDEX permite a creación de novas estruturas de índices. Sintaxe:

CREATE [UNIQUE] INDEX nome\_de\_índice ON nome\_de\_táboa ( lista\_columnas [ASC | DESC] ).

Para poder borrar ou modificar as estruturas de índices hai que facer referencia ao nome do índice, polo que é recomendable asignarlle un nome empregando unha regra como por exemplo: *idx\_ nomeTaboa\_nomeColumna*.

Exemplo de sentenza que crea un índice asociado á columna *dni* da táboa *alumno* :

**create** **index** idx\_alumno\_dni

**on** alumno **(**dni**);**

* Tarefa 7. Crear índices empregando a sentenza CREATE.

Borrado de índices

A sentenza DROP INDEX permite borrar un índice. Sintaxe:

DROP INDEX nome\_índice ON nome\_táboa

Exemplo de sentenza para borrar un índice asociado á columna *dni* da táboa *alumno*.

**drop** **index** idx\_alumno\_dni **on** alumno;

* Tarefa 8. Borrar índices empregando a sentenza DROP.
  + 1. Tarefa integradora

Esta tarefa créase coa intención de utilizar de forma secuencial a maioría das sentenzas tratadas nesta actividade.

* Tarefa 9. Tarefa integradora.
  1. Tarefas

As tarefas propostas son as seguintes:

* Tarefa 1. Conectar cun servidor de bases de datos.
* Tarefa 2. Crear bases de datos empregando a sentenza CREATE.
* Tarefa 3. Poñer en uso as bases de datos empregando a sentenza USE.
* Tarefa 4. Borrar bases de datos empregando a sentenza DROP.
* Tarefa 5. Crear táboas empregando a sentenza CREATE.
* Tarefa 6. Borrar táboas empregando a sentenza DROP.
* Tarefa 7. Crear índices empregando a sentenza CREATE.
* Tarefa 8. Borrar índices empregando a sentenza DROP.
* Tarefa 9. Tarefa integradora.
  + 1. Tarefa 1. Conectar cun servidor de bases de datos

A tarefa consiste en realizar tres conexións cun servidor de bases de datos utilizando diferentes aplicacións cliente:

* Tarefa 1.1. Conectar utilizando a ferramenta gráfica MySQL Workbench 6.2 instalada no ordenador cliente.
* Tarefa 1.2. Conectar utilizando unha consola modo texto de MySQL, tendo instalado Wampserver.
* Tarefa 1.3. Conectar mediante un navegador empregando a aplicación web phpMyAdmin.

Solución

Para conectarse a un servidor de MySQL é necesario empregar algunha aplicación cliente que establece a conexión utilizando os parámetros que lle pasa o usuario. Os parámetros que necesita a aplicación cliente son: o nome do servidor ou a IP que identifica ao equipo na rede (por defecto, *localhost*), o porto no que o servidor escoita as peticións (por defecto, 3306), o nome de usuario que se conecta (por defecto, *root*), e a súa contrasinal (por defecto, sen contrasinal).

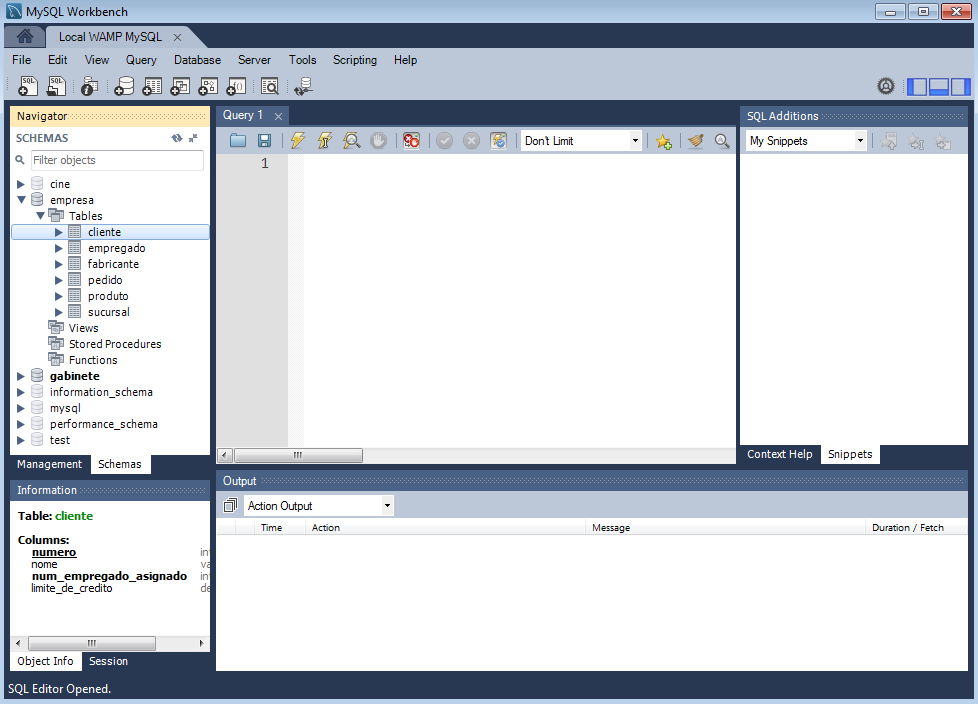
* Tarefa 1.1

Execútase a aplicación MySQL Workbench e aparece a pantalla de inicio na que se pode escoller o servidor co que se quere establecer a conexión, ou crear unha nova conexión.

workbench1.emf

No caso de ter creada a conexión, selecciónase, ábrese unha contorna gráfica na que se mostra información do servidor, que permite mediante un editor, escribir sentenzas SQL para ser enviadas ao servidor, como se pode observar na seguinte imaxe. No caso de non ter creada a conexión, hai que indicar o nome do equipo ou IP na que está instalado o servidor, o porto polo que escoita as peticións, e o nome e contrasinal do usuario MySQL.

Pódese ver máis información sobre o funcionamento de MySQL Workbench consultando a guía básica de MySQL Workbench 6.3 que se adxunta como material auxiliar.



* Tarefa 1.2

Desprégase o menú de Wampserver e selecciónase 'MySQL console' no submenú de MySQL. Ábrese unha consola en modo texto que solicita introducir o contrasinal do usuario *root*. Unha vez introducida o contrasinal, móstrase o prompt *mysql>* que indica que está establecida a conexión co servidor e se poden empezar a escribir sentenzas para que o servidor as execute.

|  |  |
| --- | --- |
| wamp_consolaMySQL.emf | wamp_consolaMySQL2.PNG |

Poderíanse teclear sentenzas SQL e tamén se poden executar comandos propios da consola. Para ver unha relación destes comandos hai que executar o comando help ou \h.

* Tarefa 1.3

Desprégase o menú de Wampserver e selecciónase a opción 'phpMyAdmin' no menú principal. Ábrese o navegador configurado por defecto na instalación de WampServer que chama á aplicación phpMyAdmin na máquina local (localhost), que é un dos compoñentes que instala WampServer.

Na configuración estándar de Wampserver establécese a conexión co servidor MySQL utilizando a conta de usuario do administrador (root), aínda que isto se pode cambiar nos ficheiros de configuración de Wampserver.

No caso de non ter instalado Wampserver, habería que abrir un navegador, indicar a url do servidor no que está instalado phpMyAdmin e identificarse mediante un nome de usuario e unha contrasinal como un usuario de MySQL.

|  |  |
| --- | --- |
| wamp_phpMyAdmin.emf | phpMyAdmin.PNG |

* + 1. Tarefa 2. Crear bases de datos empregando a sentenza CREATE

A tarefa consiste en crear unha base de datos con nome *practicas2*, utilizando o xogo de carácteres latin1 e o idioma español.

Solución

-- Creación da base de datos practicas2

**create** **database** **if** **not** **exists** practicas2

**character** **set** latin1

**collate** latin1\_spanish\_ci**;**

/\* ou tamén \*/

**create** **schema** **if** **not** **exists** practicas2

**default** **character** **set** latin1

**collate** latin1\_spanish\_ci **;**

Cando se traballa con MySQL Workbench como editor, pode ocorrer que despois de executar o *script* anterior, a base de datos non apareza na relación de bases de datos (*schemas*) que hai na parte esquerda da pantalla. Nese caso, pódese refrescar a información sobre as bases de datos seleccionando o botón que está situado xunto ao título SCHEMAS.

workbench_refrescar.emf

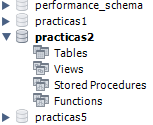
* + 1. Tarefa 3. Poñer en uso as bases de datos empregando a sentenza USE

A tarefa consiste en poñer en uso a base de datos *practicas2* creada na tarefa anterior e observar o que sucede na zona de navegación (SCHEMAS) de MySQL Workbench.

Solución

**use** practicas2**;**

Utilizando MySQL Workbench, o nome da base de datos en uso aparece destacado en letra negriña na zona de navegación situada na parte esquerda da pantalla (SCHEMAS), indicando que esa é a base de datos activa. Todas as sentenzas que se executen a partir dese momento afectarán a esa base de datos, a non ser que se indique o nome doutra base de datos na propia sentenza.



* + 1. Tarefa 4. Borrar bases de datos empregando a sentenza DROP

A tarefa consiste en borrar a base de datos *practicas2*.

Solución

**drop** **database** **if** **exists** practicas2**;**

* + 1. Tarefa 5. Crear táboas empregando a sentenza CREATE

A tarefa consiste en crear varias táboas:

* Tarefa 5.1. Crear a base de datos *practicas1*, empregando o sistema de codificación utf8 en español e crear nesa base de datos a táboa *fabricante* co motor de almacenamento MyISAM. A estrutura da táboa é a seguinte:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome columna | Tipo | Null | Clave | Observacións |
| idFabricante | char(5) | N | Primaria | Código que identifica a cada fabricante |
| nome | varchar(80) | N |  |  |

* Tarefa 5.2. Crear na base de datos *practicas1* as táboas transacionais correspondentes á interpretación do seguinte diagrama E/R e ao seu grafo relacional:

|  |  |
| --- | --- |
| F5_a_AtribitosMultivaluados.emf | F5_a_AtribitosMultivaluados_R.emf |

Terase en conta que:

* O atributo *nacionalidade* é un atributo multivaluado da entidade *pelicula*.
* O xogo de carácteres para almacenar os datos nas columnas destas táboas será latin1 e a colación en español.
* O atributo *codPelicula* será un valor numérico que se irá incrementando, de forma automática, nunha unidade cada vez que insira unha nova fila na táboa *pelicula*.
* O atributo *titulo* ocupa como máximo 100 carácteres, e non admitirá valores nulos. Os valores desta columna almacenaranse utilizando o xogo de carácteres latin1 e o sistema de colación *general*.
* O atributo *duración* gardarase como unha cantidade numérica positiva, con dous díxitos enteiros e dous decimais.
* Tarefa 5.3. Crear na base de datos *practicas1*, as táboas que correspondan ao seguinte diagrama E/R, interpretando a relación de tipo 1:N.

F3_Relacions1_N_CICLISTA.emf

Terase en conta que o código do equipo está composto por unha combinación de letras e números cun tamaño fixo de 9 carácteres.

* Tarefa 5.4. Crear na base de datos *practicas1*, as táboas que correspondan ao seguinte diagrama E/R, interpretando a relación de tipo N:M.

F3_RelacionsN_M_CONCERTOS.emf

Terase en conta que:

* O caché do grupo é o mesmo para todos os concertos e suponse que non pode ser superior a 5 millóns de euros.
* Para facilitar as buscas debe crearse un índice para a columna *nome* da táboa *grupo*.
* O *aforo* representa o número máximo de espectadores que poden asistir ao concerto e como máximo será 100000 persoas.
* O *prezo* das entradas ven dado en euros con dous decimais e o prezo máximo é 200 euros.
* Tarefa 5.5. Crear na base de datos *practicas1*, as táboas que correspondan ao seguinte diagrama E/R, interpretando a relación de tipo 1:1.

F42_Relacions1_1_MER.emf

Terase en conta que só existen tres modelos de contadores que se identifican como 'aeg235','samsung20', e 'aeg55'.

Solución

* Tarefa 5.1

/\*creación da base de datos practicas1\*/

**create** **database** **if** **not** **exists** practicas1

**default** **character** **set** utf8

**default** **collate** utf8\_spanish\_ci**;**

/\*Se podería crear a táboa sen necesidade de activar a

base de datos, utilizando a notación nomeBd.nomeTaboa: \*/

**drop** **table** **if** **exists** practicas1**.**fabricante**;**

**create** **table** practicas1**.**fabricante # nomeBd**.**nomeTaboa

**(**idFabricante **char(**5**)** **not** **null,**

nome **varchar(**80**)** **not** **null,**

**primary** **key** **(**idFabricante**)**

**)** **engine** **=** MyISAM**;**

Outra forma de facelo:

/\*creación da base de datos practicas1 utilizando a opción schema\*/

**create** **schema** **if** **not** **exists** practicas1

**default** **character** **set** utf8

**default** **collate** utf8\_spanish\_ci**;**

/\*Activar a base de datos practicas1 \*/

**use** practicas1**;**

/\*Borrar a táboa no caso de que exista coa sentenza DROP e creala coa sentenza CREATE

As operacións fanse sobre a base de datos activa, que é practicas1 \*/

**drop** **table** **if** **exists** fabricante**;**

**create** **table** fabricante

**(**idFabricante **char(**5**)** **not** **null,**

nome **varchar(**80**)** **not** **null,**

**primary** **key** **(**idFabricante**)**

**)** **engine** **=** MyISAM**;**

* Tarefa 5.2

/\*activación da base de datos practicas1\*/

**use** practicas1**;**

/\*Creación da táboa pelicula, se non existe\*/

**create** **table** **if** **not** **exists** pelicula

**(**codPelicula **smallint** unsigned auto\_increment **not** **null,**

anoRodaxe **date** **null,**

titulo **varchar(**100**)** **character** **set** latin1 **collate** latin1\_general\_ci **not** **null,**

xenero **varchar(**30**)** **not** **null,**

duracion **decimal(**4**,**2**)** unsigned **not** **null,**

**primary** **key** **(**codPelicula**)**

**)** **engine** **=** Innodb

**default** **character** **set** latin1 **default** **collate** latin1\_spanish\_ci**;**

/\*Creación da táboa nacionalidade, se non existe\*/

**create** **table** **if** **not** **exists** nacionalidade

**(**codPelicula **smallint** unsigned **not** **null,**

nacionalidade **varchar(**30**)** **not** **null,**

**primary** **key** **(**codPelicula**,**nacionalidade**),**

**constraint** fk\_nacionalidade\_pelicula **foreign** **key** **(**codPelicula**)**

**references** pelicula**(**codPelicula**)**

**on** **delete** **restrict**

**on** **update** **cascade**

**)** **engine** **=** Innodb

**default** **character** **set** latin1 **default** **collate** latin1\_spanish\_ci**;**

* Tarefa 5.3

/\*activación da base de datos practicas1\*/

**use** practicas1**;**

/\*Creación da táboa equipo, se non existe\*/

**create** **table** **if** **not** **exists** equipo

**(**

codEquipo **char(**9**)** **not** **null,**

nome **varchar(**50**)** **not** **null,**

director **varchar(**80**) not** **null,**

urlWeb **varchar(**150**) not** **null,**

**primary** **key** **(**codEquipo**)**

**)** **engine** **=** Innodb**;**

/\*Creación da táboa ciclista, se non existe\*/

**create** **table** **if** **not** **exists** ciclista

**(**

fichaFederativa mediumint unsigned **not** **null,**

dataNacemento **date not** **null,**

nome **varchar(**40**)** **not** **null,**

apelidos **varchar(**80**) not** **null,**

codEquipo **char(**9**)** **not** **null,** # **not** **null:** cardinalidade mínima 1

**primary** **key** **(**fichaFederativa**),**

**constraint** fk\_ciclista\_equipo **foreign** **key** **(**codEquipo**)**

**references** equipo**(**codEquipo**)**

**on** **delete** **restrict**

**on** **update** **cascade**

**)** **engine** **=** Innodb**;**

* Tarefa 5.4

/\*activación da base de datos practicas1\*/

**use** practicas1**;**

/\*Creación da táboa grupo, se non existe\*/

**create** **table** **if** **not** **exists** grupo

**(**

codGrupo **smallint** unsigned **not** **null,**

nome **varchar(**80**)** **not** **null,**

cache **mediumint** unsigned **not** **null,**

dataFormacion **date not** **null,**

**primary** **key** **(**codGrupo**),**

**index** idx\_grupo\_nome (nome)

**)** **engine** **=** Innodb**;**

/\*Creación da táboa concerto, se non existe\*/

**create** **table** **if** **not** **exists** concerto

**(**

codConcerto mediumint unsigned **not** **null,**

nome **varchar(**100**)** **not** **null,**

data **date not** **null,**

aforo **mediumint** unsigned **not** **null comment** 'numero máximo de espectadores'**,**

prezo **decimal(**5**,**2**)** **not** **null comment** 'prezo da entrada'**,**

**primary** **key** **(**codConcerto**)**

**)** **engine** **=** Innodb**;**

/\*Creación da táboa actuacion que sae da transformación

da relación N:M nunha nova táboa\*/

**create** **table** **if** **not** **exists** actuacion

**(**

codConcerto mediumint unsigned **not** **null,**

codGrupo **smallint** unsigned **not** **null,**

**primary** **key** **(**codConcerto**,**codGrupo**),**

**constraint** fk\_actuacion\_concerto **foreign** **key** **(**codConcerto**)**

**references** concerto**(**codConcerto**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_actuacion\_grupo **foreign** **key** **(**codGrupo**)**

**references** grupo**(**codGrupo**)**

**on** **delete** **restrict**

**on** **update** **cascade**

**)** **engine** **=** Innodb**;**

* Tarefa 5.5

/\*activación da base de datos practicas1\*/

**use** practicas1**;**

/\*Creación da táboa piso, se non existe\*/

**create** **table** **if** **not** **exists** piso

**(**

codPiso **smallint** unsigned **not** **null,**

domicilio **varchar(**150**)** **not** **null,**

localidade **varchar(**80**) not** **null,**

codPostal **char(**5**) not** **null,**

**primary** **key** **(**codPiso**)**

**)** **engine** **=** Innodb**;**

/\*Creación da táboa contador, se non existe\*/

**create** **table** **if** **not** **exists** contador

**(**

codContador **mediumint unsigned auto\_increment** **not** **null,**

modelo **enum(**'aeg235'**,**'samsung20'**,**'aeg55'**)** **not** **null,**

codPiso **smallint** unsigned **not** **null,** # **not** **null:** cardinalidade mínima 1

**primary** **key** **(**codContador**),**

**constraint** fk\_contador\_piso **foreign** **key** **(**codPiso**)**

**references** piso **(**codPiso**)**

**on** **delete** **restrict**

**on** **update** **cascade**

**)** **engine** **=** Innodb**;**

* + 1. Tarefa 6. Borrar táboas empregando a sentenza DROP

A tarefa consiste en escribir a sentenza necesaria para eliminar a táboa *fotografia* da base de datos *test*.

Solución

**use** test**;**

**drop** **table** **if** **exists** fotografia**;**

Tamén se podería executar nunha únca sentenza, cualifinaco o nome da táboa

**drop** **table** **if** **exists** test.fotografia**;**

* + 1. Tarefa 7. Crear índices empregando a sentenza CREATE

Supoñendo que a columna *nome* da táboa *concerto* da base de datos *practicas1* utilízase frecuentemente como condición para localizar os datos dun concerto determinado, pídese crear un índice asociado a esta columna para que esas consultas tarden menos tempo en executarse.

Solución

**create** **index** idx\_concerto\_nome

**on** concerto **(**nome**);**

* + 1. Tarefa 8. Borrar índices empregando a sentenza DROP

Supoñendo que có paso do tempo chegouse á conclusión de que era mellor eliminar o índice asociado á columna *nome* da táboa *concerto* da base de datos *practicas1.* Pídese escribir a sentenza SQL que permite borrar ese índice.

Solución

**drop** **index** idx\_ concerto\_nome **on** concerto**;**

Neste caso pódese ver a importancia que ten o utilizar unha regra para darlle o nome aos índices no momento en que se crean, pois hai que utilizalos cando se queren borrar. No caso de non darlle un nome no momento da creación, hai que buscar cal é o nome que lle deu o servidor.

* + 1. Tarefa 9. Tarefa integradora
* Tarefa 9.1. A tarefa consiste en escribir un script de sentenzas SQL para realizar as seguintes operacións na base de datos *practicas1*:
* Borrar a táboa *pelicula* e todas as relacionadas con ela.
* Crear as táboas que correspondan ao seguinte diagrama E/R, interpretando a seguinte relación ternaria.

F7_TernariasNMP.emf

* Crear un índice asociado á columna *titulo* da táboa *pelicula*, e outro índice composto polas columnas *apelido1*, *apelido2* e *nome* da táboa *actor*.
* Borrar o índice composto creado no apartado anterior.
* Borrar a táboa *roda*.
* Tarefa 9.2. Crear a base de datos *gabinete*, tendo en conta o seguinte grafo relacional:

u4a3_gabinete.emf

Tarefa 9.3. Diagrama MERE para o deseño dunha base de datos para unha companía formada por empregados, departamentos e proxectos.

exercico proxecto_MERE.emf

Partindo deste diagrama MERE, crear unha base de datos co nome proxectos, utilizando o sistema de colación utf8 español, e as táboas que resultan de aplicar as regras de tranformación de:

* entidades (fortes e débiles).
* atributos de entidades.
* atributos multivaluados.
* interrelacións N:M. Crear só as táboas, non crear as restricións de integridade referencial.
* interrelacións 1:N:M. Crear só as táboas, non crear as restricións de integridade referencial.
* atributos de interrelacións.

Solución

* Tarefa 9.1

/\*activación da base de datos practicas1\*/

**use** practicas1**;**

/\*borrado da táboa película e táboas relacionadas\*/

**drop** **table** **if** **exists** nacionalidade**;**

**drop** **table** **if** **exists** pelicula**;**

/\*Creación da táboa pelicula, se non existe\*/

**create** **table** **if** **not** **exists** pelicula

**(**

codPelicula **smallint** unsigned auto\_increment **not** **null,**

titulo **varchar(**100**)** **not** **null,**

xenero **varchar(**30**),**

duracion **decimal(**4**,**2**)** unsigned**,**

**primary** **key** **(**codPelicula**)**

**)** **engine** **=** Innodb **;**

/\*Creación da táboa actor, se non existe\*/

**create** **table** **if** **not** **exists** actor

**(**

idActor **mediumint** unsigned **not** **null,**

nome **varchar(**40**)** **not** **null,**

apelido1 **varchar(**40**),**

apelido2 **varchar(**40**),**

**primary** **key** **(**idActor**)**

**)** **engine** **=** Innodb**;**

/\*Creación da táboa personaxe, se non existe\*/

**create** **table** **if** **not** **exists** personaxe

**(**

codPersonaxe **char(**10**)** **not** **null,**

descricion **varchar(**40**)** **not** **null,**

**primary** **key** **(**codPersonaxe**)**

**)** **engine** **=** Innodb**;**

/\*Creación da táboa roda que sae da transformación

da relación ternaria N:M:P nunha nova táboa, se non existe\*/

**create** **table** **if** **not** **exists** roda

**(**

codRodaxe **smallint** unsigned auto\_increment **not** **null,**

codPelicula **smallint** unsigned **not** **null,**

idActor **mediumint** unsigned **not** **null,**

codPersonaxe **char(**10**)** **not** **null,**

**primary** **key** **(**codRodaxe**),**

**constraint** fk\_roda\_pelicula **foreign** **key** **(**codPelicula**)**

**references** pelicula**(**codPelicula**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_roda\_actor **foreign** **key** **(**idActor**)**

**references** actor**(**idActor**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_roda\_personaxe **foreign** **key** **(**codPersonaxe**)**

**references** personaxe**(**codPersonaxe**)**

**on** **delete** **restrict**

**on** **update** **cascade**

**)** **engine** **=** Innodb**;**

/\*creación do índice composto\*/

**create** **index** idx\_actor\_apelido1\_apelido2\_nome **on** actor **(**apelido1**,**apelido2**,**nome**);**

/\*borrado do índice composto\*/

**drop** **index** idx\_actor\_apelido1\_apelido2\_nome **on** actor**;**

/\*borrado da táboa roda\*/

**drop** **table** roda**;**

* Tarefa 9.2

/\*BORRADO DA BASE DE DATOS, NO CASO QUE EXISTA

(só cando se están facendo probas)\*/

**drop** **database** **if** **exists** gabinete**;**

/\* CREACIÓN DA BASE DE DATOS\*/

**create** **database** **if** **not** **exists** gabinete

**character** **set** utf8

**collate** utf8\_spanish\_ci**;**

/\* ou tamén \*/

**create** **schema** **if** **not** **exists** gabinete

**default** **character** **set** utf8

**collate** utf8\_spanish\_ci **;**

/\* ACTIVACIÓN DA BASE DE DATOS \*/

**use** gabinete**;**

/\* CREACIÓN DAS TÁBOAS \*/

**create** **table** cliente **(**

dni **char(**9**)** **not** **null,**

nome **varchar(**100**)** **not** **null,**

enderezo **varchar(**150**)** **not** **null,**

**primary** **key** **(**dni**),**

**key** nome **(**nome**)**

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

**create** **table** telefono **(**

dni **char(**9**)** **not** **null,**

telefono **char(**12**)** **not** **null,**

**primary** **key** **(**dni**,**telefono**),**

**constraint** fk\_telefono\_cliente **foreign** **key** **(**dni**)**

**references** cliente **(**dni**)**

**on** **delete** **cascade**

**on** **update** **cascade**

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

**create** **table** avogado **(**

dni **char(**9**)** **not** **null,**

nome **varchar(**100**)** **not** **null,**

enderezo **varchar(**150**)** **not** **null,**

**primary** **key** **(**dni**)**

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

**create** **table** asunto **(**

expediente **int(**10**)** unsigned **not** **null** auto\_increment**,**

dataInicio **datetime** **not** **null** **default** **current\_timestamp,**

descricion text **not** **null,**

dataArquivo **date** **default** **null,**

estado enum**(**'recopilación'**,**'tramite'**,**'xuizo'**,**'arquivado'**)** **not** **null,**

dniCliente **char(**9**)** **not** **null,**

**primary** **key** **(**expediente**),**

**key** dniCliente **(**dniCliente**),**

**constraint** fk\_asunto\_cliente **foreign** **key** **(**dniCliente**)**

**references** cliente **(**dni**)**

**on** **delete** **restrict**

**on** **update** **cascade**

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

**create** **table** avogado\_asunto **(**

dniAbogado **char(**9**)** **not** **null,**

expediente **int(**10**)** unsigned **not** **null,**

**primary** **key** **(**dniAbogado**,**expediente**),**

**key** expediente **(**expediente**),**

**constraint** fk\_avogado\_asunto\_avogado **foreign** **key** **(**dniAbogado**)**

**references** avogado **(**dni**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_avogado\_asunto\_asunto **foreign** **key** **(**expediente**)**

**references** asunto **(**expediente**)**

**on** **delete** **restrict**

**on** **update** **cascade**

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

* Tarefa 9.3

**drop** **database** **if** **exists** proxectos**;**

-- creación da base de datos proxectos

**create** **database** proxectos

**default** **character** **set** utf8

**collate** utf8\_general\_ci**;**

-- activación da base de datos proxectos

**use** proxectos**;**

-- ENTIDADES

-- creación da táboa persoal

**create** **table** persoal

**(**

dni **integer(**8**)** unsigned **not** **null,**

nome **varchar(**80**)** **not** **null,**

enderezo **varchar(**100**)** **not** **null,**

codigoPostal **char(**5**)** **not** **null,**

localidade **varchar(**80**)** **not** **null,**

dataNacemento **date** **not** **null,**

sexo enum**(**'home'**,**'muller'**)** **not** **null,**

**primary** **key** **(**dni**)**

**)** **engine** **=** InnoDB**;**

-- creación da táboa empregado

**create** **table** empregado

**(**

dni **integer(**8**)** unsigned **not** **null,**

nss **char(**10**)** **not** **null,**

salario **decimal(**8**,**2**)** unsigned **not** **null** **comment** 'salario mensual do empregado'**,**

**primary** **key** **(**dni**),**

**constraint** uk\_empregado\_nss **unique** **key** **(**nss**)**

**)** **engine** **=** InnoDB**;**

-- creación da táboa departamento

**create** **table** departamento

**(**

codDepartamento **smallint** unsigned **not** **null,**

nome **varchar(**50**)** **not** **null,**

**primary** **key** **(**codDepartamento**)**

**)** **engine** **=** InnoDB**;**

-- creación da táboa proxecto

**create** **table** proxecto

**(**

codProxecto **smallint** unsigned auto\_increment**,**

nome **varchar(**50**)** **not** **null,**

descricion text **not** **null,**

**primary** **key** **(**codProxecto**)**

**)** **engine** **=** InnoDB**;**

-- creación da táboa curso

**create** **table** curso

**(**

codCurso **smallint** unsigned **not** **null,**

nome **varchar(**50**)** **not** **null,**

descricion text **not** **null,**

nivel enum**(**'iniciación'**,**'básico'**,**'medio'**,**'alto'**,**'profesional'**),**

numeroHoras **smallint** unsigned **not** **null,**

**primary** **key** **(**codCurso**)**

**)** **engine** **=** InnoDB**;**

-- creación da táboa edicion

**create** **table** edicion

**(**

codCurso **smallint** unsigned **not** **null,** # hereda clave primaria da entidade forte

codEdicion **smallint** unsigned **not** **null,**

nPrazasEmpregados **tinyint** unsigned **not** **null,**

nPrazasPracticas **tinyint** unsigned **not** **null,**

dataRealizacion **date** **not** **null,**

**primary** **key** **(**codCurso**,**codEdicion**)** # clave composta por ser entidade débil

**)** **engine** **=** InnoDB**;**

-- ATRIBUTOS MULTIVALUADOS DE ENTIDADES

-- creación da táboa telefono

**create** **table** telefono

**(**

dni **integer(**8**)** unsigned **not** **null,**

numeroTelefono **char(**12**)** **not** **null,**

**primary** **key** **(**dni**,**numeroTelefono**)**

**)** **engine** **=** InnoDB**;**

-- creación da táboa localizacion

**create** **table** localizacion

**(**

codDepartamento **smallint** unsigned **not** **null,**

localizacion **char(**50**)** **not** **null,**

**primary** **key** **(**codDepartamento**,** localizacion**)**

**)** **engine** **=** InnoDB**;**

-- INTERRELACIÓNS N:M (créase unha táboa)

-- Relación asiste (N:M).

-- Creación da táboa asiste, sen restricións de integridade referencial

**create** **table** asiste

**(**

codAsistencia **integer** unsigned auto\_increment**,**

dniPersoal **integer(**8**)** unsigned **not** **null,**

codCurso **smallint** unsigned **not** **null,**

codEdicion **smallint** unsigned **not** **null,**

**primary** **key** **(**codAsistencia**)**

**)** **engine** **=** InnoDB**;**

-- Relación participa (N:M).

-- Creación da táboa participa, sen restricións de integridade referencial

**create** **table** participa

**(**

codParticipa **integer** auto\_increment**,**

dataInicio **date** **not** **null,**

dataFin **date** **null,**

horas **smallint** unsigned **not** **null,**

dniEmpregado **integer(**8**)** unsigned **not** **null,**

codProxecto **smallint** unsigned**,**

**primary** **key** **(**codParticipa**)**

**)** **engine** **=** InnoDB**;**

-- Relación documenta (N:M).

-- Creación da táboa documenta, sen restricións de integridade referencial

**create** **table** documenta

**(**

codDocumenta **integer** auto\_increment**,**

detalle text **not** **null,** # multivaluado

codProxecto **smallint** unsigned**,**

dniPracticas **integer(**8**)** unsigned **not** **null,**

**primary** **key** **(**codDocumenta**)**

**)** **engine** **=** InnoDB**;**

-- Relación podesupervisar (N:M).

-- Creación da táboa podesupervisar, sen restricións de integridade referencial

**create** **table** podesupervisar

**(**

codProxecto **smallint** unsigned**,**

dniEmpregado **integer(**8**)** unsigned **not** **null,**

**primary** **key** **(**codProxecto**,** dniEmpregado**)**

**)** **engine** **=** InnoDB**;**

-- RELACIÓNS ENTRE AS ENTIDADES. RELACIÓNS DE TIPO 1:N:M (créase unha táboa)

-- Relación supervisa (1:N:M).

-- Creación da táboa supervisa, sen restricións de integridade referencial

**create** **table** supervisa

**(**

codSupervisa **integer** auto\_increment**,**

observacions text **not** **null,** # multivaluado

codProxecto **smallint** unsigned **not** **null,**

dniSupervisor **integer(**8**)** unsigned **not** **null,**

dniSupervisado **integer(**8**)** unsigned **not** **null,**

**primary** **key** **(**codSupervisa**)**

**)** **engine** **=** InnoDB**;**

1. Materiais
   1. Documentos de apoio ou referencia

* DUBOIS, Paul. *La biblia de MySQL*. Anaya Multimedia, 2009.
* GROFF, James.*SQL.* McGraw-Hill.
* HUESO IBAÑEZ, Luis. *Bases de datos*. Ciclos Formativos Ra-Ma, 2012.
* *ANSI SQL Standards and Guidelines*. <http://www.whoishostingthis.com/resources/ansi-sql-standards/>
* *Manual de referencia de MySQL 5.6*. <http://dev.mysql.com/doc/>
* CAMPS PARÉ, Rafael, CASILLAS SANTILLÁN, Luís Alberto, COSTAL COSTA, Dolors, GILBERT GINESTÁ, Marc, MARTÍN ESCOFET, Carme, PÉREZ MORA, Óscar. *Bases de datos.* UOC, 2007.

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

* POZO CORONADO, Salvador. *MySQL con Clase.*  [http://mysql.conclase.net/curso/?cap=000#](http://mysql.conclase.net/curso/?cap=000), 2004.
* CAMBRA, Pedro. *Duplicar o clonar tablas en MySQL*. <http://cambrico.net/mysql/duplicar-o-clonar-tablas-en-mysql>, 2008.
* WiseRatel. Curso MySQL - Comandos básicos. <http://www.wiseratel.com/cursos/bases-de-datos/curso-mysql/>
  1. Recursos didácticos
* Ordenadores con conexión a Internet, que terán instalado o sistema xestor de bases de datos MySQL, o cliente MySQL Workbench e un programas de deseño de diagramas (tipo MS Visio ou Dia).
* Proxector.
* Material didáctico subministrado polo profesorado en papel e/ou formato dixital.
* Máquina virtual específica para exame que terá instalado o sistema xestor de bases de datos MySQL e o cliente MySQL Workbench.
* Manual de referencia de MySQL.
  1. Material auxiliar

O material auxiliar anexo a esta actividade está almacenado na carpeta CSIFC02\_MP0484\_V000401\_CreaBD\_Anexo que contén o arquivo GuiaWorkbench.docx que é unha guía básica de MySQL Workbench 6.3.

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 |
| * CA4.1 -  Analizouse o formato de almacenamento da información. | * Exame en papel e en formato dixital. * Resultado da execución dos guións de sentenzas. | * **PE.1** - Cuestionario con preguntas de resposta breve sobre o formato de almacenamento da información. | 5 |
| * CA4.2 -  Creáronse bases de datos.   CA4.2.1 - Creáronse e elimináronse bases de datos. | * **TO.1** - Documento de rexistro de creación e eliminación de bases de datos utilizando a LDD. Escala de valores (observación indirecta). | 10 |
| * CA4.3 -  Creáronse as táboas e as relacións entre elas.   CA4.3.1 -  Creáronse e elimináronse táboas e relacións entre elas. | * **TO.3** - Documento de rexistro de creación e eliminación de táboas e as relacións entre elas utilizando a LDD. Escala de valores (observación indirecta). | 15 |
| * CA4.4 -  Seleccionáronse os tipos de datos adecuados. | * **TO.5** - Documento de rexistro de selección dos tipos de datos das táboas utilizando a LDD. Escala de valores (observación indirecta). | 7 |
| * CA4.5 -  Definíronse os campos clave nas táboas. | * **TO.6** - Documento de rexistro de definición de campos clave utilizando a LDD. Escala de valores (observación indirecta). | 7 |
| * CA4.6 -  Aplicáronse as restricións reflectidas no deseño lóxico. | * **TO.7** - Documento de rexistro de aplicación das restricións obtidas utilizando a LDD. Escala de valores (observación indirecta). | 10 |
| * CA4.8 - Utilizáronse asistentes, ferramentas gráficas e a linguaxe de definición de datos.   CA4.8.1  -  Utilizáronse asistentes, ferramentas gráficas e a linguaxe de definición de datos. | * **TO.9** - Documento de rexistro de utilización de Workbench para editar e executar os guións LDD. Escala de valores (observación indirecta). | 6 |

Modelo de proba combinada para PE.1, TO.1, TO.3, TO.5, TO.6, TO.7 e TO.9

Proponse un cuestionario con preguntas de resposta breve para avaliar CA4.1 e unha proba práctica para avaliar CA4.2.1, CA4.3.1, CA4.4, CA4.5 CA4.6 e CA4.8.1 mediante observación indirecta sobre un documento de rexistro da práctica.

A solución da práctica deberá ser entregada polo alumnado nun documento escrito e en dixital.

* O documento escrito debe conter:
* Bloque de sentenzas solicitado.
* Imaxe capturada co resultado da execución do bloque dende o cliente Workbench.
* O arquivo dixital debe conter:
* Arquivos .sql co código correspondente ao bloque de sentenzas.

A parte práctica consistirá en escribir o guión de sentenzas que permita realizar as operacións solicitadas para dúas bases de datos: unha da que se ten o diagrama Entidade/Relación e outra da que se ten información das táboas.

A base de datos *federacion\_futbol* utilizarase para levar o rexistro das competicións dos clubs que participan nas competicións organizadas pola federación de fútbol. O diagrama Entidade/Relación é:

exame_ER.emf

A base de datos *musica* está formada polas táboas que se describen a continuación.

* Táboa CANCION con motor de almacenamento innodb.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Tipo | Null | Clave | Observacións |
| id\_disco | smallint (sin signo) | No | Primary | Código identificativo do disco |
| numero\_pista | tinyint (sin signo) | No |  |
| titulo | varchar(90) | No |  |  |
| duración | decimal(5,2) |  |  | Duración da canción en minutos e segundos |

* Táboa DISCO con motor de almacenamento innodb.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Tipo | Null | Clave | Observacións |
| id | smallint (sin signo) | No | Primary | Código que identifica ao disco |
| titulo | varchar(80) | No |  |  |
| autor | char(3) |  |  | Código do autor do disco |
| id\_discografica | char(5) | No |  | Código que identifica á discográfica que edita o disco |

* Táboa DISCOGRAFICA con motor de almacenamento innodb.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Tipo | Null | Clave | Observacións |
| id | char(5) | No | Primary | Código da discográfica |
| nome | varchar(80) | No |  |  |

Proba para CA4.1 sobre formatos de almacenamento

* 1.- Explicar de forma breve cómo se almacena fisicamente a información nas táboas que usan motores de almacenamento Innodb e MyISAM.

Proba para CA4.2.1 sobre crear e eliminar bases de datos e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD

* 2.- Borrar a base de datos *federacion\_futbol*, no caso de que exista, e creala empregando o sistema de ordenación e comparación en español do xogo de carácteres *latin1*.
* 3.- Crear a base de datos *musica*, no caso de que non exista, empregando o sistema de ordenación e comparación de español do xogo de carácteres utf8.

Proba para CA4.3.1 sobre crear e eliminar táboas, CA4.4 sobre seleccionar tipos de datos adecuados para as táboas e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD

* 4.- Crear as táboas da base de datos *federacion\_futbol* e as relacións entre elas. Todas as táboas serán transaccionais. Hai que elixir os tipos de datos e as propiedades máis adecuadas para cada columna, tendo en conta a información que vai a almacenar. Unha columna que vai almacenar o prezo dun artigo maior que cero, ten que ser numérica con decimais e sen signo; por exemplo, decimal(6,2) unsigned. Unha columna que vai almacenar os apelidos dunha persoa, debería ser declarada alfanumérica de lonxitude variable; por exemplo, varchar(50).
* 5.- Borrar as táboas *cancion*, *disco* e *discográfica*, no caso de que existan, e crealas cun motor de almacenamento transacional, implementando os tipos de datos indicados para cada columna e as propiedades dos atributos que figuran na descrición das táboas da base de datos *musica*.

Proba para CA4.5 sobre campos clave e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD

* 6.- Definir os campos clave nas táboas da base de datos *federacion\_futbol.*
* 7.- Definir os campos clave nas táboas da base de datos *musica.*

Proba para CA4.6 sobre aplicar restricións e CA4.8.1 sobre utilizar asistentes, ferramentas gráficas e LDD

* 8.- Aplicar as restricións reflectidas no diagrama Entidade/Relación da base de datos *federacion\_futbol.*
* 9.- Aplicar as restricións reflectidas no diagrama relacional da base de datos *musica.*

Exemplo de solución para entregar en papel

Proponse a seguinte solución:

* 1.- Explicar de forma breve cómo se almacena fisicamente a información nas táboas que usan motores de almacenamento Innodb e MyISAM
* Almacenamento de información nas táboas que usan motor de almacenamento MyISAM. As táboas MyISAM almacénanse no directorio de datos (datadir), e os ficheiros que xeran teñen as extensións MYD, para os datos, MYI, para os índices, e FRM para os metadatos (esquema).
* Almacenamento de información nas táboas que usan motor de almacenamento InnoDB. De forma predeterminada, InnoDB xestiona as táboas nun único espazo de táboas compartido (ibdata), en lugar de utilizar arquivos específicos para cada táboa como fan o resto de motores. O espazo de táboa pode estar formado por varios arquivos diferentes. Tamén se pode configurar para usar espazos de táboa individuais para cada táboa. Neste caso, cada táboa inclúe un arquivo .ibd no directorio de datos. O tamaño inicial deste espazo é de 16 Mb e vai medrando a medida que se gravan máis datos.
* Preguntas 2, 4, 6 e 8, relacionadas coa base de datos *federacion\_futbol.*

**drop** **database** **if** **exists** federacion\_futbol**;**

/\*creación da base de datos, no caso de que non exista\*/

**create** **database** **if** **not** **exists** federacion\_futbol

**character** **set** latin1

**collate** latin1\_spanish\_ci**;**

/\*activación da base de datos\*/

**use** federacion\_futbol**;**

/\*creación da táboa competicion\*/

**create** **table** competicion

**(**

codigo **char** **(**6**)** **not** **null,**

nome **varchar** **(**60**)** **not** **null,**

**primary** **key** **(**codigo**)**

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

/\*creación da táboa club\*/

**create** **table** club

**(**

nome **char(**30**)** **not** **null,**

socios **integer** **unsigned** **not** **null,**

**primary** **key** **(**nome**)**

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

/\*creación da táboa clasificacion\*/

**create** **table** clasificacion

**(**

idClasificacion **mediumint** **unsigned auto\_increment** **not** **null,**

codigoCompeticion **char** **(**6**)** **not** **null,**

temporada **char(**7**)** **not** **null,**

posicion **tinyint** **unsigned** **not** **null,**

nomeClub **char(**30**)** **not** **null,**

**primary** **key** **(**idClasificacion**),**

**constraint** fk\_clasificacion\_competicion

**foreign** **key** **(**codigoCompeticion**)** **references** competicion **(**codigo**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_clasificacion\_club

**foreign** **key** **(**nomeClub**)** **references** club **(**nome**)**

**on** **delete** **restrict**

**on** **update** **cascade**

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

/\* Tamén se podería crear a táboa cunha clave primaria composta:

create table clasificacion

(

codigoCompeticion char (6) not null,

temporada char(7) not null,

posicion tinyint unsigned not null,

nomeClub char(30) not null,

primary key (codigoCompeticion,temporada,posicion),

constraint fk\_clasificacion\_competicion

foreign key (codigoCompeticion) references competicion (codigo)

on delete restrict

on update cascade,

constraint fk\_clasificacion\_club

foreign key (nomeClub) references club (nome)

on delete restrict

on update cascade

) engine = innodb;

\*/

/\*creación da táboa persoa\*/

**create** **table** persoa

**(**

codigo **smallint** **unsigned auto\_increment** **not** **null,**

nome **varchar(**70**)** **not** **null,**

dataNacemento **date not** **null,**

**primary** **key** **(**codigo**)**

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

/\*creación da táboa contrato\*/

**create** **table** contrato

**(**

idContrato **smallint** **unsigned auto\_increment** **not** **null,**

nomeClub **char(**30**)** **not** **null,**

codigoPersoa **smallint** unsigned **not** **null,**

dataContrato **date** **not** **null,**

importe **mediumint** **unsigned** **not** **null,**

dataCese **date,**

**primary** **key** **(**idContrato**),**

**constraint** fk\_contrato\_club

**foreign** **key** **(**nomeClub**)** **references** club **(**nome**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_contrato\_persoa

**foreign** **key** **(**codigoPersoa**)** **references** persoa **(**codigo**)**

**on** **delete** **restrict**

**on** **update** **cascade**

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

/\* Tamén se podería crear a táboa cunha clave primaria composta:

create table contrato

(

nomeClub char(30) not null,

codigoPersoa smallint unsigned not null,

dataContrato date not null,

importe mediumint unsigned,

dataCese date,

primary key (nomeClub, codigoPersoa, dataContrato),

constraint fk\_contrato\_club

foreign key (nomeClub) references club (nome)

on delete restrict

on update cascade,

constraint fk\_contrato\_persoa

foreign key (codigoPersoa) references persoa (codigo)

on delete restrict

on update cascade

) engine = innodb;

\*/

/\*creación da táboa xogador\*/

**create** **table** xogador

**(**

codigo **smallint** **unsigned** **not** **null,**

peso **decimal(**5**,**2**) not** **null,**

altura **decimal(**3**,**2**) not** **null,**

**primary** **key** **(**codigo**),**

**constraint** fk\_xogador\_persoa

**foreign** **key** **(**codigo**)** **references** persoa **(**codigo**)**

**on** **delete** **restrict**

**on** **update** **cascade**

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

/\*creación da táboa axudante\*/

**create** **table** axudante

**(**

codigo **smallint** **unsigned** **not** **null,**

nome **varchar(**60**) not** **null,**

**primary** **key** **(**codigo**)**

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

/\*creación da táboa entrenador\*/

**create** **table** entrenador

**(**

codigo **smallint** **unsigned** **not** **null,**

dataInicio **date not** **null,**

codigoAxudante **smallint** **unsigned** **not** **null,**

**primary** **key** **(**codigo**),**

**constraint** fk\_entrenador\_persoa

**foreign** **key** **(**codigo**)** **references** persoa **(**codigo**)**

**on** **delete** **restrict**

**on** **update** **cascade,**

**constraint** fk\_entrenador\_axudante

**foreign** **key** **(**codigoAxudante**)** **references** axudante **(**codigo**)**

**on** **delete** **restrict**

**on** **update** **cascade**

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

A imaxe capturada dende a zona de saída (Output) do cliente Workbench co resultado da execución do bloque de sentenzas é:



* Preguntas 3, 5, 7 e 9, relacionadas coa base de datos *musica.*

/\*creación da base de datos, no caso de que non exista\*/

/\*creación da base de datos, no caso de que non exista\*/

**create** **database** **if** **not** **exists** musica

**default** **character** **set** utf8

**collate** utf8\_spanish\_ci **;**

/\*activación da base de datos\*/

**use** musica**;**

/\*borrar táboas, no caso de que existan\*/

**drop** **table** **if** **exists** cancion**,** disco**,** discográfica**;**

/\*creación da táboa discografica\*/

**create** **table** discografica

**(**

id **char(**5**)** **not** **null,**

nome **varchar(**80**)** **not** **null,**

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

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

/\*creación da táboa disco\*/

**create** **table** disco

**(**

id **smallint** **unsigned** **not** **null,**

titulo **varchar(**80**)** **not** **null,**

autor **char(**3**),**

id\_discografica **char(**5**)** **not** **null,**

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

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

/\*creación da táboa cancion\*/

**create** **table** cancion

**(**

id\_disco **smallint** **unsigned** **not** **null,**

numero\_pista **tinyint** **unsigned** **not** **null,**

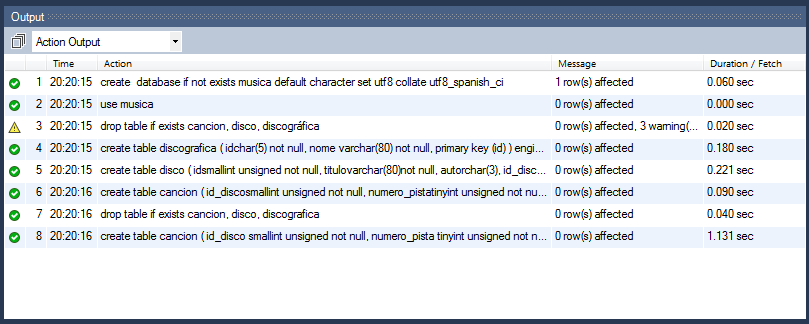
titulo **varchar(**90**)** **not** **null,**

duracion **decimal(**5**,**2**),**

**primary** **key** **(**id\_disco**,** numero\_pista**)**

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

A imaxe capturada dende a zona de saída (Output) do cliente Workbench co resultado da execución do bloque de sentenza é:



Exemplo de lista de valoración para TO.1

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.1 (EV) (observación indirecta: documento de rexistro de creación e eliminación de bases de datos utilizando a LDD).

| Nome | Data | |
| --- | --- | --- |
| Indicadores para  CA4.2.1 - Creáronse e elimináronse bases de datos. | Valoración máxima  10 | Cualificación |
| Creouse a base de datos coas especificacións pedidas. | 6 |  |
| Borrouse a base de datos se xa existía. | 2 |  |
| Activouse a base de datos. | 2 |  |

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 creación e eliminación de táboas e as relacións entre elas utilizando a LDD).

| Nome | Data | |
| --- | --- | --- |
| Indicadores para  CA4.3.1 - Creáronse e elimináronse táboas e relacións entre elas. | Valoración máxima  15 | Cualificación |
| Creáronse as táboas coas especificacións pedidas. | 13 |  |
| Borráronse as táboas se xa existían. | 2 |  |

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 selección dos tipos de datos das táboas utilizando a LDD).

| Nome | Data | |
| --- | --- | --- |
| Indicadores para  CA4.4 - Seleccionáronse os tipos de datos adecuados. | Valoración máxima  7 | Cualificación |
| Seleccionáronse os tipos de datos coas especificacións pedidas. | 7 |  |

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 definición de campos clave utilizando a LDD).

| Nome | Data | |
| --- | --- | --- |
| Indicadores para  CA4.5 - Definíronse os campos clave nas táboas. | Valoración máxima  7 | Cualificación |
| Definíronse os campos clave. | 7 |  |

Exemplo de lista de valoración para TO.7

Proponse a seguinte lista de valoración para o instrumento de avaliación TO.7 (EV) (observación indirecta: documento de rexistro de aplicación das restricións obtidas utilizando a LDD).

| Nome | Data | |
| --- | --- | --- |
| Indicadores para  CA4.6 - Aplicáronse as restricións reflectidas no deseño lóxico. | Valoración máxima  10 | Cualificación |
| Aplicáronse as restricións reflectidas no diagrama entidade/relación ou no diagrama relacional. | 10 |  |

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 de utilización de Workbench para editar e executar os guións LDD).

| Nome | Data | |
| --- | --- | --- |
| Indicadores para  CA4.8.1 - Utilizáronse asistentes, ferramentas gráficas e a linguaxe de definición de datos. | Valoración máxima  6 | Cualificación |
| Executáronse os bloques de sentenzas SQL creados e xustificouse capturando a imaxe da zona Output de Workbench co resultado da execución. | 10 |  |

1. As copias impresas dos estándares SQL teñen restricións de copyright. Pódense atopar á venda na páxina web de [ANSI](http://www.ansi.org/), seleccionando 'Access Standards', e en 'Standards Store' buscar (*search*): "SQL Language." [↑](#footnote-ref-3)
2. Sección *8.4.2 Optimizing MySQL Data Types* do manual de referencia MySQL 5.6. [↑](#footnote-ref-4)
3. <http://cambrico.net/mysql/duplicar-o-clonar-tablas-en-mysql> [↑](#footnote-ref-5)
4. <https://es.wikipedia.org/wiki/ISAM> [↑](#footnote-ref-6)
5. Atomicity Consistency Isolation Durability. [↑](#footnote-ref-7)