****

Módulo Profesional 02:  
Desarrollo de aplicaciones web **Actividad UF3**

**CICLO FORMATIVO DE GRADO SUPERIOR EN**

**DESARROLLO DE APLICACIONES WEB**

**MODALIDAD ONLINE**

**JUAN BELLO FERNANDEZ**

**Actividad UF3:**

|  |  |
| --- | --- |
| **Objetivos** | |
| Interpretar el diseño lógico de bases de datos, analizando y cumpliendo las especificaciones relativas a su aplicación, para gestionar bases de datos.  Seleccionar lenguajes, objetos y herramientas, interpretando las especificaciones para  desarrollar aplicaciones web con acceso a bases de datos.  **Competencias asociadas:**   * Gestionar bases de datos, interpretando su diseño lógico y verificando integridad, consistencia, seguridad y accesibilidad de los datos. | |
| **Metodología** | **Entrega** |
| * Preparación individual * Discusión de la actividad | 26/04/2022 y formato de entrega en documento Word. |
| **Dedicación estimada** | **Documentos de referencia** |
| 50 horas | Plataforma online. |
| **Resultados de aprendizaje** | |
| * Implanta métodos de control de acceso utilizando asistentes, herramientas gráficas y comandos del lenguaje del sistema gestor de bases de datos corporativo. * Desarrolla procedimientos almacenados evaluando y utilizando las sentencias del lenguaje incorporado en el sistema gestor de bases de datos. | |
| **Criterios de evaluación** | |
| * Creación de usuarios y roles con asignación de privilegios sobre la base de datos y tablas. * Identificar las diversas formas de automatizar tareas. * Identificar las herramientas disponibles para editar guiones. * Creación de guiones o scripts para automatizar tareas. * Uso de las funciones proporcionadas por el sistema gestor. * Creación de procedimientos almacenados (procedures). * Creación de disparadores (triggers). * Utilización de asistentes, herramientas gráficas y los lenguajes de definición y control de datos. | |

**Desarrollo de la actividad**

Contenido

[1 Gestión de usuarios, roles y privilegios de acceso 4](#_Toc67859209)

[2 Programación de bases de datos 47](#_Toc67859210)

[2.1 Funciones 47](#_Toc67859211)

[2.2 Procedimientos 48](#_Toc67859212)

[2.3 Disparadores 57](#_Toc67859213)

# Gestión de usuarios, roles y privilegios de acceso

¿Qué es un rol?

|  |
| --- |
| Los roles son los conjuntos de permisos. Estos conjuntos existen a tres niveles distintos: servidor, base de datos y aplicación.  Los roles permiten agrupar los derechos y gestionar más fácilmente los diferentes usuarios y las conexiones. |

|  |
| --- |
| 1.1 Desde la consola MySQL:  Crear un rol nuevo llamado BOSSTuNombre que actúe de administrador sobre todo el motor de la base de datos. |
| -Creamos el rol con “**CREATE ROLE ‘BOSSJuan’;**”    -Le asignamos todos los privilegios con “**GRANT ALL ON \*.\* TO ‘BOSSJuan’ WITH GRANT OPTION;**”    -Comprobación privilegios mediante “**SHOW GRANTS FOR BOSSJuan;**” |
|  |

|  |
| --- |
| 1.2 Crear un nuevo usuario llamado THEBOSSTuNombre asociado al rol BOSSTuNombre. |
|  |
| Verificar creación de usuario y derechos: |
| -Verificación de la creación del usuario mediante la sentencia “**SELECT USER FROM mysql.user;**”    -Verificación permisos usuario THEBOSSJuan mediante la sentencia “**SHOW GRANTS FOR THEBOSSJuan;**”    -Verificación permisos usuario THEBOSSJuan mediante la sentencia “**SELECT user, host FROM mysql.user;**” |
| 1.3 Acceder por consola al nuevo usuario THEBOSSTuNombre y realizar todas las tareas con este usuario a partir de ahora: |
| -Comprobación del usuario actual: |
| Comprobar los privilegios del usuario THEBOSSTuNombre |
| -Lo comprobamos mediante el comando **“SHOW GRANTS”** |
| 1.4 Creación de una base de datos llamada INSTITUTO |
| -Bases de datos existentes previa a la creación de ‘instituto’    -Creación base de datos ‘instituto’ |
| 1.5 Crear las siguientes tablas: |
| **ALUMNOS**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **clave\_alumno** | **nombre** | **apellido1** | **edad** | **curso\_actual** | |  |  |  |  |  | |  |  |  |  |  |   **AULAS**   |  |  | | --- | --- | | **numero\_aula** | **capacidad** | |  |  | |  |  |   **MODULOS**   |  |  |  | | --- | --- | --- | | **clave\_modulo** | **curso** | **descripcion** | |  |  |  | |  |  |  | |
| -Utilizamos la base de datos ‘**instituto**’    -Contenido base de datos previa a la creación de las tablas    -Creación tabla ‘alumnos’ y comprobación    -Creación tabla ‘aulas’ y comprobación    -Creación tabla ‘modulos’ y comprobación |
| 1.6 Ejecutar comando DESCRIBE para verificar la estructura de cada tabla: |
| **-**DESCRIBE **‘alumnos’**    **-**DESCRIBE **‘aulas’**    **-**DESCRIBE **‘modulos’** |
| 1.7 Renombrar la tabla MODULOS a ASIGNATURAS |
| -BBDD previa a la actualización    -Modificación nombre tabla y comprobación |
| 1. 8 Añadir un nuevo campo a la tabla ALUMNOS llamado apellido2 |
| -Tabla ‘**alumnos’** inicial    -Tabla ‘**alumnos’** modificada |
| 1.9 Añadir un nuevo campo a la tabla ASIGNATURAS llamada nombre\_professor |
| -Tabla ‘**asignaturas’** inicial    -Tabla ‘**asignaturas’** modificada |
| 1.10 Insertar en cada tabla 5 registros. |
| -Inserción tabla ‘**alumnos’** y comprobación    -Inserción tabla ‘**asignaturas’** y comprobación    -Inserción tabla ‘**aulas’** y comprobación |
| 1.11 Realizar alguna consulta SELECT sobre cada una de las tablas creadas. |
| -Consultas tablas ‘alumnos’        -Consultas tabla ‘asignaturas’      -Consultas tabla ‘aulas’ |
| 1.12 Añadir diversos usuarios con perfiles de acceso diferentes y dar privilegios:   * Sólo acceso local: userlocalTuNombre * Sólo acceso local: localTuNombre. Puede consultar todas las tablas y modificar la tabla ‘ALUMNOS’ y ‘ASIGNATURAS’ menos los campos ‘edad’, ‘curso\_actual’ y ‘descripcion’. * Sólo acceso externo: userexternoTuNombre |
| 1.12.1 Sólo acceso local: userlocalTuNombre. Todos los permisos.  -Usuarios antes de la creación:    -Creación usuario ‘**userlocalJuan’**    -Asignación de todos los permisos al usuario ‘**userlocalJuan’  ·**Permisos antes de la asignación  ·Permisos modificados |
| 1.12.1.1 Comprobaciones usuario userlocalTuNombre. Todos los permisos. |
| -Para comprobar los permisos del usuario ‘**userlocalJuan’**, cambiamos a ese usuario en mySql mediante la query “**SYSTEM MYSQL** **-u userlocalJuan –p**”. Mediante la query “**SELECT** **CURRENT\_USER**” comprobamos que estamos utilizando dicho usuario    -Utilizamos la BBDD ‘**instituto’** de este mismo ejercicio    **SELECT**: Realizamos la query ‘**SELECT \* FROM**’ de las 3 tablas de la base de datos    **INSERT**: Insertaremos un nuevo registro en cada una de las 3 tablas de la BBDD  -Tabla ‘**alumnos’** y comprobación    -Tabla ‘**asignaturas’** y comprobación    -Tabla ‘**aulas’** y comprobación    **UPDATE**: Realizaremos un UPDATE en cada una de las 3 tablas de la BBDD  -Tabla ‘**alumnos’** y comprobación  **Previa**    **Actualizada**    -Tabla ‘**asignaturas’** y comprobación  **Previa**    **Actualizada**    -Tabla ‘**aulas’** y comprobación  **Previa**    **Actualizada**    **DELETE**:  -Tabla ‘**alumnos’** y comprobación  **Previa**    **Actualizada**    -Tabla ‘**asignaturas’** y comprobación  **Previa**    **Actualizada**    -Tabla ‘**aulas’** y comprobación  **Previa**    Actualizada    **ALTER TABLE**:  -Tabla ‘**alumnos’** y comprobación  **Previa**    **Actualizada**    -Tabla ‘**asignaturas’** y comprobación  **Previa**    **Actualizada**  D  -Tabla ‘**aulas’** y comprobación  **Previa**    **Actualizada** |
| 1.12.2 Sólo acceso local: localTuNombre. Puede consultar todas las tablas y modificar la tabla ‘ALUMNOS’ y ‘ASIGNATURAS’ menos los campos ‘edad’, ‘curso\_actual’ y ‘descripcion’. |
| -Creación usuario ‘**localJuan’**    -Asignación permisos: una vez creado el usuario, no dispone de ningún permiso asignado, cosa que haremos con el comando ‘**GRANT’**    Si cambiamos al usuario ‘**localJuan’** e intentamos acceder a la BBDD ‘**instituto’**, nos lanza un error de acceso denegado a dicha BBDD    -Asignación permisos de consulta en todas las tablas. Una vez asignado el permiso de consulta, podemos acceder tanto a la BBDD ‘**instituto’** como a una de sus tablas, ‘**aulas’** en este caso de ejemplo      -Permisos de modificación de tablas ‘**alumnos’** y ‘**asignaturas’** menos los campos ‘**edad’**, ‘**curso\_actual’** y ‘**descripcion’**.  ·Tabla ‘**alumnos’**: todos los campos menos ‘**edad’** y ‘**curso\_actual**’    ·Tabla ‘**asignaturas’**: todos los campos menos ‘**descripcion’**    -Comprobación permisos asignados |
| 1.12.2.1Comprobaciones con el usuario localTuNombre |
| -Cambiamos al Usuario ‘**localJuan’**    **SELECT**:  -Tabla ‘**alumnos’**, ‘**asignaturas’** y ‘**aulas’**    **UPDATE**:  ·Tabla ‘**alumnos’**: permiso en todos los campos menos ‘**edad’** y ‘**curso\_actual**’  **Update** campo ‘**nombre’** (permitido)    **Update** campo ‘**edad’** (restringido)    **Update** campo ‘**curso\_actual**’ (restringido)    ·Tabla ‘**asignaturas’**: permiso en todos los campos menos ‘**descripcion’**  **Update** campo **‘nombre\_professor’** (permitido)    **Update** campo **‘descripcion’** (restringido)    **DELETE**: Restricción en todas las tablas |
|  |
| Sólo acceso externo: userexternoTuNombre.  Tiene acceso de lectura en todas las tablas. Puede operar en todas las tablas menos en el campo ‘clave\_alumno’ y en la tabla ‘ASIGNATURAS’ sólo puede consultar. No puede modificar la estructura de ninguna tabla.  Nota:  Antes de crear el usuario tenemos que saber qué dirección IP tendrá el cliente que se conecte al servidor MySQL (3 opciones):  -primera opción: máquina virtual con adaptador de red en modo sólo-anfitrión (saber que IP tiene el servidor) y instalar el cliente mysql en la máquina física.  -segunda opción: si tenemos 2 máquinas virtuales en la misma máquina física podemos configurarla con el modo Host Only o Red interna (saber la IP del servidor).  -tercera opción: 2 máquinas físicas conectadas a la misma red (saber la IP del servidor). |
| La actividad se ha realizado con dos MV en el mismo equipo, la máquina cliente es la máquina donde tenemos instalado **mysql server** y la máquina cliente utiliza el software **HEIDISQL**  -Comprobamos el rango IP de la máquina cliente:    -Creación usuario ‘**userexternoJuan’**, asignando acceso restringido por IP    -Asignación permisos:  ·Tiene acceso de lectura en todas las tablas.    ·Operar en todas las tablas: -Tabla ‘**alumnos’**, todo menos en el campo ‘**clave\_alumno**’ (especificamos las columnas a las que si tiene acceso    -Tabla ‘**asignaturas**’  No se asignan nuevos permisos, ya tiene el SELECT  -Tabla ‘**aulas’**, todo menos modificar estructura de la tabla  -Verificación permisos |
| Comprobaciones con el usuario userexternoTuNombre. |
| -Conexión entre Máquina cliente y Máquina servidor    Cliente conectado de forma remota a BBDD ‘**instituto’**    **ACCESO BBDD**:    **SELECT**: Permiso para **SELECT** a todas las tablas de la BBDD ‘**instituto’**    **INSERT**: puede operar en todas las tablas menos en la tabla **'asignaturas'**  **ALUMNOS**  -Con el campo ‘**clave\_alumno**’: nos devuelve un error porque no tiene permisos para acceder a ese campo    -Sin campo **‘clave\_alumno’** también nos devuelve un error porque el campo no es autoincrementable ni tiene un valor por defecto y está definido como **NOT NULL**    **ASIGNATURAS**  -Nos dará un error porque solo tiene permiso de consulta    **AULAS**  **-**Podemos operar sin restricciones      **UPDATE**: puede operar en todas las tablas menos en el campo **'clave\_alumno**' (alumnos) y en la tabla **'asignaturas'**  **ALUMNO**  **-**Nos dará error si intentamos modificar el campo ‘**clave\_alumno’**    -Nos permite modificar cualquier otro campo        **ASIGNATURAS**  -Nos dará un error porque solo tiene permiso de consulta    **AULAS**  -Sin errores porque tiene acceso sin restricciones en **UPDATE**      **DELETE**: Nos dará error en cualquier tabla menos en ‘aulas’ porque es la única donde tiene permiso asignado de borrado  **ALUMNOS**    **ASIGNATURAS**    **AULAS**      **ALTER**: Nos dará error en las 3 tablas porque no tiene asignado permisos **ALTER**  **ALUMNOS**    **ASIGNATURAS**    **AULAS** |
|  |

|  |
| --- |
| **Insertar sentencias de todos los pasos realizados en el apartado 1 en un fichero sql** |
|  |

# Programación de bases de datos

## Funciones

|  |
| --- |
| Crear una funció que retorne el valor de la hipotenusa de un triángulo a partir de los valores de sus lados. |
| Al intentar crear la funcion en la consola me dio el siguiente error:    Lo pude solucionar con la query ‘**SET GLOBAL log\_bin\_trust\_function\_creators = 1;**’ y pude implementarla **NOTA**: en el archivo sql se ha implementado la función de 3 formas diferentes, siendo esta la última y la que me parecía más correcta |

Tenemos la siguiente base de datos:

**EMPLEADO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Documento** | **Nombre** | **Apellidos** | **Sueldo** | **CantidadHijos** | **Num\_dpt** |
| **25528647P** | **Joan** | **Perez** | **1250** | **2** | **1** |
| **25458847R** | **Lluis** | **Lopez** | **900** | **0** | **1** |
| **25528847O** | **Marta** | **Perez** | **1000** | **1** | **2** |
| **25128847L** | **Susana** | **Garcia** | **800** | **2** | **3** |
| **25522847J** | **Jose Maria** | **Morales** | **1100** | **3** | **3** |
| **25528347D** | **Andres** | **Perez** | **1400** | **3** | **2** |
| **25528847P** | **Laura** | **Garcia** | **800** | **3** | **3** |

**DEPARTAMENTO**

|  |  |  |
| --- | --- | --- |
| **Num\_dpt (auto\_increment)** | **Nombre** | **Planta** |
| **1** | **Contabilidad** | **1** |
| **2** | **Sistemas** | **2** |
| **3** | **Secretaria** | **1** |
| **4** | **Comercial** | **2** |

Se adjunta el fichero SQL de creación de la base de datos y su población con los datos a insertar:



## Procedimientos

Realiza captura de pantalla de la creación de los procedimientos y su ejecución para su correcta validación.

**NOTA**: al no haber especificación concreta, se han realizado todas las operaciones con el usuario ‘**root’**

|  |
| --- |
| 2.2.1. Crear un procedimiento almacenado llamado insertar\_departamentos\_tunombre que inserte un nuevo departamento con los datos (nombre y planta) introducidos como parámetros. |
| -Creación procedimiento    -Llamada procedimiento    -Resultado |
| 2.2.2. Crear un procedimiento almacenado llamado empleados\_sueldo\_tunombre que seleccione los nombres, apellidos y sueldos de los empleados. |
| -Creación procedimiento    -Llamada procedimiento y resultado |
| 2.2.3. Crear un procedimiento almacenado llamado empleados\_hijos\_tunombre que seleccione los nombres, apellidos y cantidad de hijos de los empleados con hijos. |
| -Creación procedimiento    -Llamada procedimiento y resultado |
| 2.2.4. Crear un procedimiento almacenado llamado empleados\_mayor\_sueldo\_tunombre que seleccione los nombres, apellidos y sueldos de los empleados que tengan un sueldo superior o igual al enviado como parámetro. |
| -Creación procedimiento    -Llamada procedimiento y resultado: la asignación del sueldo a comparar puede hacerse bien mediante un **SET** asignando a la variable que recibe el SP el valor a comparar (se ha hecho de esta forma), o podría hacerse también colocando el valor directamente en la llamada al SP |
| 2.2.5. Crear una tabla llamada "Indefinido" con los siguientes campos: documento, nombre, apellido. Crear otra tabla llamada "Temporal" con los siguientes campos: documento, nombre, apellido. |
| -Sin SP    -Con SP |
| 2.2.6. Crear un procedimiento almacenado llamado clasificar\_empleados\_tunombre que inserte a la tabla "Indefinidos" a todos los empleados que cobren más de 1000 €, y que inserte a la tabla "Temporales" a todos los empleados que cobren menos o igual de 1000 €. |
| -Creación procedimiento    -Llamada procedimiento y resultado |
| 2.2.7. Crear un procedimiento que elimine un departamento, verificando que exista y que no tenga empleados. El procedimiento recibirá como parámetro de entrada el nombre del departamento. En caso de no existir, o de tener empleados, el procedimiento ha de informar de este hecho con sus respectivos mensajes:  El departamento no existe o está mal escrito.  El departamento tiene empleados y no se puede borrar. |
| -Creación procedimiento    -Llamada procedimiento y resultados  ·Estado tabla ‘**departamento’** antes de ejecución y elemento a eliminar ‘**Playa y** **piscina’** (departamento **SIN** empleados)    ·Ejecución y resultado    ·Estado tabla ‘**departamento’** antes de ejecución y elemento a eliminar ‘**Contabilidad’** (departamento **CON** empleados)    ·Ejecución y resultado    ·Estado tabla ‘**departamento’** antes de ejecución y elemento a eliminar ‘**CContabilidadD’** (departamento **MAL ESCRITO**)    ·Ejecución y resultado |

## Disparadores

Una biblioteca tiene la siguiente base de datos:

**LIBRO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Codigo** | **Titulo** | **Autor** | **Editorial** | **Precio** | **Stock** |
| **1** | **Alicia en el país de las maravillas** | **Lewis Carroll** | **Planeta** | **30** | **10** |
| **2** | **Dracula** | **Bram Stoker** | **DeBolsillo** | **60** | **20** |
| **3** | **Don Quijote de la Mancha** | **Miguel de Cervantes** | **Galaxia** | **15** | **15** |
| **4** | **El guardian entre el centeno** | **J.D. Salinger** | **Alianza** | **25** | **30** |

**SOCIO**

|  |  |  |  |
| --- | --- | --- | --- |
| **Codigo** | **Nombre** | **Direccion** | **telefono** |
| **1111** | **Vicente Romero** | **c/ Camps Blancs, 23** | **610575392** |
| **1243** | **Ramses Moreno** | **c/ Eusebi Güell, 10** | **637288106** |
| **5656** | **Jessica Ruiz** | **Ronda San Ramón, 231** | **630567120** |
| **1789** | **Belén Gallego** | **c/ Industria, 49** | **647933872** |

**VENTA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cod\_libro** | **Cod\_socio** | **fecha** | **cantidad** | **precio** |

Esta tabla aún no contiene datos.

Se adjunta script de creación base de datos y población de las tablas



Realiza captura de pantalla de la creación de los disparadores y su ejecución con las opciones de comprobación para su correcta validación.

|  |
| --- |
| 2.3.1. Las ventas se irán añadiendo a la tabla correspondiente mediante un procedimiento. Este recibirá como parámetros de entrada el código del libro, el código del socio, la fecha de la venta y la cantidad vendida. El precio de la venta lo calculará el procedimiento, y será la cantidad del libro por su precio unitario. |
| Creación del procedimiento    Valores antes del procedimiento:    Ejecutamos procedimiento: se han añadido las ventas a la tabla ‘**venta’** y se ha actualizado el ‘**stock’** disponible en la tabla ‘**libro’** |
| 2.3.2. Cada vez que se venda un libro, el sistema ha de controlar que la cantidad vendida no exceda del stock de ese libro. Si lo intentamos hacer nos tiene que dar el siguiente mensaje "No queda stock para comprar, sentimos las molestias." |
| Creación del disparador:    Valores antes del disparador:    Ejecutamos disparador:  -Hacemos dos **INSERT** en ventas con el procedimiento anterior ‘**insert\_Ventas’**, uno con exceso de stock y otro en un rango correcto  ·**Con stock disponible**: actualiza la tabla ‘**venta’** con la nueva venta y actualiza el stock en la tabla ‘**libro’**      ·**Sin stock**: salta el error de ‘falta de stock’ y no cambia ningún parámetro en ninguna de las dos tablas |
| 2.3.3. El sistema no ha de dejarnos modificar el nombre de un socio. Si lo intentamos hacer nos tiene que dar el siguiente mensaje "No se puede cambiar el nombre de socio." |
| Creación del disparador:    Valores antes del disparador:  Ejecutamos disparador:  -Realizamos un **UDPATE** del nombre del socio ‘**Vicente** **Romero’** |
| 2.3.4. El sistema no ha de dejarnos modificar ni el título ni el autor de los libros. Si lo intentamos hacer nos tiene que dar el siguiente mensaje " No se puede cambiar el nombre del libro ni el autor. " |
| Creación del disparador:    Valores antes del disparador:    Ejecutamos disparador:  --Realizamos un **UDPATE** del título del libro 2 y el autor del libro 3 |
| 2.3.5. A causa de la crisis, se ha producido un aumento en la morosidad. Se ha de crear una tabla llamada Morosos con los mismos datos que la tabla Socio a partir de un procedimiento llamado crear\_morosos. Después poblar la tabla con los mismos datos (INSERT). |
| Creación del procedimiento:    NOTA: he añadido un **TRUNCATE** morosos antes de insertar pensando que, si se ejecuta de nuevo el trigger nos dará errores de datos duplicados. La idea es que vacíe la tabla ‘**morosos’** antes de poblarla ya que la puebla con todos los registros de la tabla ‘**socio’.** Obviamente no es la implementación perfecta pero sí la que pide el ejercicio.  Ejecutamos procedimiento: |
| 2.3.6. Hay que evitar que los socios que deben dinero se den de baja. |
|  |
| 2.3.7. Si modificamos cualquier campo de los socios (que no sea el nombre), el sistema ha de automáticamente cambiar los mismos datos a la tabla de morosos, si el socio editado está en esa tabla. |
| Creación del procedimiento    Estado previo    Ejecutamos procedimiento: haremos un **UPDATE** del socio **1111** de los 3 campos, primero el número de código y posteriormente sobre el nuevo código los campos dirección y teléfono |

|  |
| --- |
| **Insertar ficheros sql del apartado 2.1,2.2 y 2.3** |
|  |

|  |  |
| --- | --- |
| **Ponderación actividad UF3 sobre 10** | **NOTA sobre 10** |
| 1. Gestión de usuarios, roles y privilegios    1. Creación rol, usuario y tablas………………….1,5 puntos    2. Añadir usuarios con privilegios en las tablas y comprobación……………………………………………2,5 puntos   Distribución 2,5 puntos:  userlocalTuNombre…………………………………..0,5 puntos  localTuNombre………………………………………… 1 punto  userexternoTuNombre……………………………..1 punto | 4 |
| 1. Programación de bases de datos    1. Funciones…………………………………………….……..0,5 puntos    2. Procedimientos………………………………………..…3 puntos    3. Disparadores ……………………………………………… 2,5 puntos | 6 |