CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

ACTIVIDAD EVALUABLE 3

AE3.2

BASES DE DATOS 22/23
CFGS DAM

PARTE 2 DE 2: DISEÑO DE PROGRAMAS EN MYSQL

Autor:
Abelardo Martínez

1. DESCRIPCIÓN

- **Se pide diseñar el conjunto de PROGRAMAS (SCRIPT) más adecuado para cada caso.**

Entrega **solo** las consultas referentes al bloque que corresponda con la última cifra de tu DNI/NIE
0-4 => MODELO B 5-9 => MODELO A

2. PLAZO DE ENTREGA Y PORCENTAJES

- **Porcentajes en la EVALUACIÓN**: 30% de la nota total es para las evaluables
- **Porcentajes de la ACTIVIDAD**: 50% de las evaluables (hay dos por evaluación)
- **Porcentajes de esta PARTE**: 50% de la evaluable (hay dos: AE3.1 y AE3.2)
- **Plazo de entrega de AMBAS PARTES**: 23:59 del 18 de abril de 2023 (4 semanas)

3. CALIFICACIÓN

- Le entrega no es obligatoria ni hay nota mínima. Cada parte se calificará de 0 a 10 según las indicaciones de este documento y se hará la media entre las dos partes.

4. RECURSOS

- Debes estudiar todos los materiales que te hemos proporcionado, prestando especial atención a los boletines de ejercicios.
- **Sigue los pasos indicados en la plantilla que te proporcionamos para facilitar la corrección de la actividad.**

5. PLAGIO

- Tarea INDIVIDUAL. En caso de sospecha de originalidad será requerida una entrevista oral.

6. INSTRUCCIONES DE ENTREGA

- La tarea se entregará en un único PDF que contendrá los ejercicios de ambas partes, resueltos siguiendo la plantilla indicada. **NO SE ACEPTAN ENTREGAS EN OTROS FORMATOS DISTINTOS AL PDF.**

7. SOLUCIONES Y RESULTADOS

- Recibirás la calificación desglosada por cada criterio, y el total, junto con cualquier comentario que brinde sugerencias sobre cómo podrías haberlo hecho mejor.

PASOS A SEGUIR

1. Calcula cuál es tu modelo de diagrama según la última cifra de tu documento de identidad.
2. Descarga del Aula Virtual la plantilla, el diagrama E-R que de tu modelo (A, B) y el script DDL asociado.
3. **Recuerda que deberás partir de esa BASE DE DATOS, independientemente del diagrama que presentaste en la anterior actividad evaluable.**
4. Recomendaciones:
   - Diseña primero los procedimientos, funciones y triggers y, en caso necesario, crea datos ficticios para las pruebas.
   - Usa la sintaxis MySQL.
   - Usa mayúsculas/minúsculas y tabulaciones/espacios para hacer el código lo más legible.
   - No uses variables de usuario dentro del programa. Usa variables locales si las necesitas.
   - Puedes usar esta sintaxis para los mensajes de error:

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'Operación no permitida\n*****\n====> Mensaje de error.\n*****\n';
```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'Parámetros inesperados\n*****\n====> Mensaje de error.\n*****\n';
```

   - Recuerda también usar los errores en los triggers oportunos para evitar que se realicen operaciones no permitidas.
   - Asegúrate de que tu conjunto de PROGRAMAS (script) realiza la tarea solicitada.
     - **85% de la nota de cada SCRIPT: ES CORRECTO (hace lo que se pide)**
   - Usa el mayor número de ALIAS posible, juega con las mayúsculas/minúsculas, las tabulaciones y los saltos de línea para hacer cada SCRIPT lo más legible posible:
     - **15% de la nota de cada SCRIPT: ES LEGIBLE y SIGUE LAS INDICACIONES**
   - Para completar tu estudio se recomienda realizar también las consultas y scripts correspondientes a la otra parte a modo de práctica, pues ambas actividades se complementan bastante. Esto es sólo una recomendación y no una actividad evaluable/entregable.

ENUNCIADO MODELO A: BASE IMPERIAL

SCRIPT 1A: LISTAR Y CONTAR NAVES POR MATRÍCULA Y TIPO (2 PUNTOS)

Crea un procedimiento almacenado para obtener la matrícula y el nombre de las naves que han realizado algún transporte cuya matrícula comienza por la letra pasada como parámetro (de entrada) y son del tipo pasado como parámetro (de entrada) (T para TieFighter y C para Crucero) y devuelve el número de resultados en un tercer parámetro (de salida). Ordena los resultados por MATRÍCULA y NOMBRE.

Se pide:

- Crea el procedimiento **pListarNavesTransporte_porLetrayTipo**
- Muestra un mensaje de error si no se recibe una 'T' o una 'C' como 2º parámetro.

SCRIPT 2A: LISTAR TRANSPORTES POR NÚMERO DE COSTES Y POR EUROS GASTADOS (2 PUNTOS)

Crea una función para obtener, dado el código de misión, el número de costes asociados y una segunda función para obtener, también dado un código de misión, los euros totales que ha costado. Usando las dos funciones, lista las 2 misiones que más costes asociados tienen y otro listado para las 2 misiones que más dinero han costado.

Se pide:

- Crea la función **fNumcostesMision**
- Crea la función **fTotalCostesMision**

SCRIPT 3A: TRIGGERS PARA LA TABLA COSTE (2 PUNTOS)

Crea los triggers necesarios para asegurar que los números de línea de un mismo transporte son consecutivos cuando se inserta una nueva línea de costes. Para hacerlo más sencillo, ignora los borrados, gestiona las inserciones y prohíbe las actualizaciones del campo LINEA en esa tabla.

Se pide:

- Crea el trigger **tAntesActualizarLineasCoste**
  - Muestra un mensaje de error cuando se intenta cambiar el campo LINEA.
- Crea el trigger **tAntesInsertarLineasCoste**
  - El campo LINEA debe ser >0 y siempre consecutivo para un mismo código de misión, de manera que, si insertamos la línea 7 de coste de la misión MA001, debe existir antes la línea de coste 6. Si no existe esa línea 6, debe cancelarse la operación con un mensaje de error.

SCRIPT 4A: TRIGGERS PARA LA P+D DE NAVES (2 PUNTOS)

Al traducir del diseño conceptual al modelo relacional, existía una pérdida semántica en las especializaciones que no eran Parcial+Solapada (PS), quedándose como una restricción de integridad que dijimos que "ya resolveríamos más adelante".

Pues bien, mediante triggers podemos traducir cualquier especialización (TS, TD o PD) y superar así esa pérdida semántica, aunque no de manera sencilla. Solo las disjuntas tienen una solución con triggers más o menos sencilla.

Te proponemos crear los triggers necesarios para asegurar la **especialización DISJUNTA de NAVE en Tiefighter y Crucero durante las inserciones**, ignorando las actualizaciones y los borrados por su complejidad.

Se pide:

- Crea el trigger **necesario para prevenir inserciones incorrectas en Tiefighter**
- Crea el trigger **necesario para prevenir inserciones incorrectas en Crucero**

SCRIPT 5A: TRIGGERS PARA LAS PARTICIPACIONES 1:N (2 PUNTOS)

Al traducir del diseño conceptual al modelo relacional, existía una pérdida semántica en las participaciones 1:N, quedándose como una restricción de integridad que dijimos que "ya resolveríamos más adelante".

Pues bien, mediante triggers podemos traducir cualquier participación 1:N y superar así esa pérdida semántica de manera sencilla.

Te proponemos crear los triggers necesarios para asegurar la **participación 1:N de la relación entre HANGAR y NAVE**, ignorando las inserciones porque no afectan a la 1:N.

Se pide:

- Crea el trigger **necesario para prevenir borrados de NAVE** que rompan la participación 1:N en la relación de HANGAR con NAVE.
- Crea el trigger **necesario para prevenir actualizaciones del campo HANGAR de la tabla NAVE** que rompan la participación 1:N en la relación de HANGAR con NAVE.

ENUNCIADO MODELO B: ACADEMIA JEDI

SCRIPT 1B: ACADEMIAS Y CURSOS POR COSTES ASOCIADOS (2 PUNTOS)

Crea una función para obtener, dado el código de academia y código de curso, el número de costes asociados y una segunda función para obtener, también dado un código de academia y código de curso, los euros totales que ha costado. Usando las dos funciones, lista los 2 cursos que más costes asociados tienen y otro listado para los 2 cursos que más dinero han costado.

Se pide:

- Crea la función **fNumcostesAcadCurso**
- Crea la función **fTotalCostesAcadCurso**

SCRIPT 2B: LISTAR ALUMNOS POR LETRA Y TIPO (2 PUNTOS)

Crea un procedimiento almacenado para obtener el nombre completo y la descripción del curso en que se encuentra matriculado de los alumnos cuyo nombre comienza por la letra pasada como parámetro (de entrada) y son del tipo pasado como parámetro (de entrada) (P para Padawan y S para Senior) y devuelve el número de resultados en un tercer parámetro (de salida). Ordena los resultados por el nombre completo.

Se pide:

- Crea el procedimiento **pListarAlumnos_porLetrayTipo**
- Muestra un mensaje de error si no se recibe una ‘P’ o una ‘S’ como 2º parámetro.
- Para el nombre completo usa la función CONCAT.

SCRIPT 3B: PÉRDIDA SEMÁNTICA EN ALUMNOS (T+D) (2 PUNTOS)

Al traducir del diseño conceptual al modelo relacional, existía una pérdida semántica en las especializaciones que no eran Parcial+Solapada (PS), quedándose como una restricción de integridad que dijimos que "ya resolveríamos más adelante".

Pues bien, mediante triggers podemos traducir cualquier especialización (TS, TD o PD) y superar así esa pérdida semántica, aunque no de manera sencilla. Solo las disjuntas tienen una solución con triggers más o menos sencilla.

Te proponemos crear los triggers necesarios para asegurar la **especialización DISJUNTA de los ALUMNOS en PADAWAN y SENIOR durante las inserciones**, ignorando las actualizaciones y los borrados por su complejidad. (Los datos del script DDL pueden no cumplir estas restricciones, pero el trigger servirá a partir de ahora)

Se pide:

- Crea el trigger **necesario para prevenir inserciones incorrectas en PADAWAN**
- Crea el trigger **necesario para prevenir inserciones incorrectas en SENIOR**

SCRIPT 4B: TRIGGERS PARA LAS PARTICIPACIONES 1:N (2 PUNTOS)

Al traducir del diseño conceptual al modelo relacional, existía una pérdida semántica en las participaciones 1:N, quedándose como una restricción de integridad que dijimos que "ya resolveríamos más adelante".

Pues bien, mediante triggers podemos traducir cualquier participación 1:N y superar así esa pérdida semántica de manera sencilla.

Te proponemos crear los triggers necesarios para asegurar la **participación 1:N de la relación entre CURSO y ALUMNO**, ignorando las inserciones porque no afectan a la 1:N.

Se pide:

- Crea el trigger **necesario para prevenir borrados de ALUMNO** que rompan la participación 1:N en la relación de CURSO con ALUMNO.
- Crea el trigger **necesario para prevenir actualizaciones del campo CODCURSO de la tabla ALUMNO** que rompan la participación 1:N en la relación de CURSO con ALUMNO.

SCRIPT 5B: LÍNEAS DE COSTES CONSECUTIVAS (2 PUNTOS)

Crea los triggers necesarios para asegurar que los números de línea de un mismo coste son consecutivos cuando se inserta una nueva línea de coste de un curso. Para hacerlo más sencillo, ignora los borrados, gestiona las inserciones y prohíbe las actualizaciones del campo NUMLINEA en esa tabla.

Se pide:

- Crea el trigger **tAntesActualizarLineasCoste**
  - Muestra un mensaje de error cuando se intenta cambiar el campo LINEA
- Crea el trigger **tAntesInsertarLineasCoste**
  - El campo NUMLINEA debe ser >0 y siempre consecutivo para un mismo COSTE, de manera que, si por ejemplo insertamos la línea 6 del coste del curso '6050' y academia 'GLEE', debe existir antes el 5. Si no existe esa línea 5, debe cancelarse la operación con un mensaje de error.