Normativa de examen

- No se permite el uso de ningún dispositivo electrónico, así como libros ni apuntes.
- Cada estudiante podrá disponer de un folio tamaño A4 con las anotaciones que considere oportunas para desarrollar su examen. El folio podrá contener texto por ambas caras.
- Durante el examen, los profesores podrán solicitar acreditar la identidad de los participantes en el mismo. Deberá tener en todo momento su documento nacional de identidad, permiso de residencia o carné de estudiante visible sobre la mesa.
- Deberá escribir su nombre y apellidos, con bolígrafo, en todas las hojas de las que consta el examen.
- No se permite abandonar el aula de examen durante los primeros 20 minutos. Transcurrido este tiempo, no se permitirá entrar al examen.
- El examen tiene una duración máxima de **2.5 horas**.
- Justifique sus respuestas lo mejor posible indicando, si fuese necesario, los pasos realizados.
- Las calificaciones provisionales serán publicadas en el aula virtual de la asignatura dentro de los 15 días naturales desde la fecha de realización del examen, publicándose ahí la fecha para su revisión.

EJERCICIO 1. Bloque "Modelo Entidad/Relación y paso a tablas"

(a) (3 Puntos) En un lugar algo olvidado de la red, donde paquetes TCP y UDP hacen cola y los hackers de sombrero negro acechan en las sombras, se encuentra la gran corporación *ByteMagic Inc.*

Tras un viernes de actualizaciones en sus sistemas (tarea que según algunos empleados tuvo más de *magic* que de *bytes*), los ingenieros comenzaron a idear lo que terminaron denominando **La Fortaleza**. Esta maravilla de la seguridad se concibió para mantener a raya a los intrusos y proteger los valiosos datos de la compañía incluyendo, por supuesto, la colección de memes de los de DevOps.

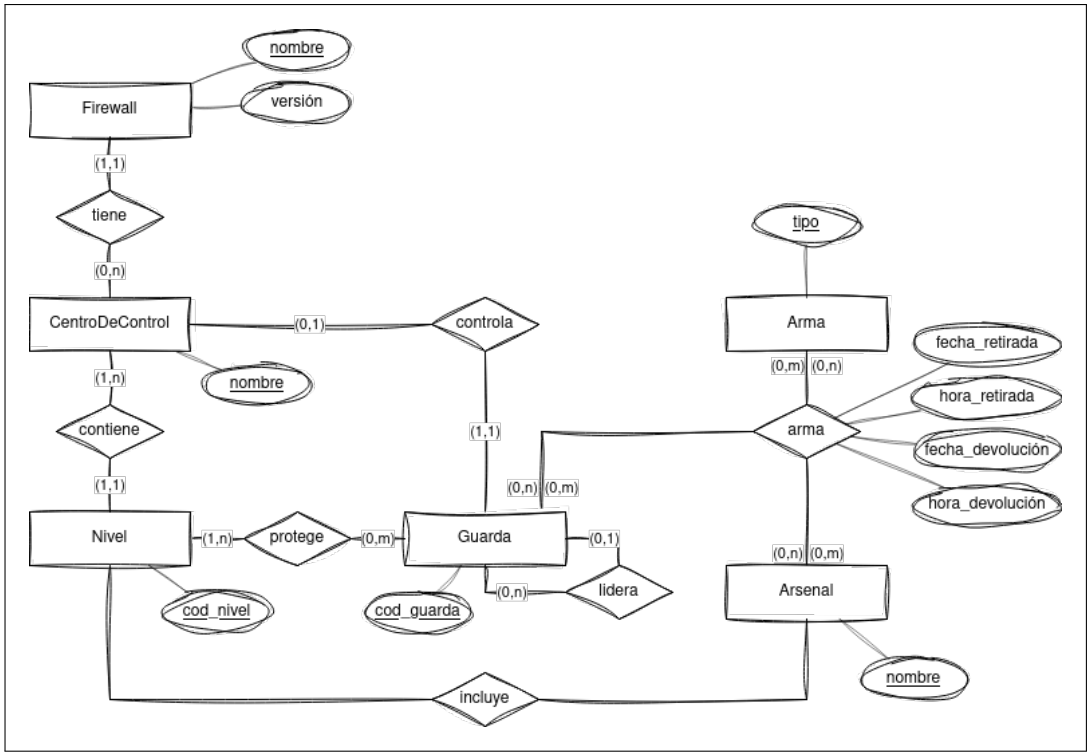
La Fortaleza estaría organizada en niveles, como las capas de una cebolla muy temperamental. Cada nivel contendría uno o más centros de control con nombre en clave único, fortificados con un tipo de *firewall*, por lo que sería necesario saber tanto qué software tiene instalado como su versión.

Los guardas estarían meticulosamente identificados y jerarquizados. Cada uno de ellos, salvo el Comandante en Jefe, podrá tener una banda de subordinados a su cargo, cual gallinas bajo el ala protectora de otra gallina, aunque esta más grande y ligeramente paranoica. Los guardas tendrían la noble misión de proteger, al menos, un nivel. Además, cada centro de control contaría con un único guarda responsable para asegurar de que todo funciona como un reloj suizo.

En caso de que el caos decidiera hacer una visita, los guardas pueden dirigirse a los depósitos de armas (también cada uno con su nombre en clave) que hay repartidos por los niveles, ya que todos los niveles tienen al menos uno de estos arsenales al lado de la cafetería. Allí, se pertrecharían con todos los tipos de arma que quisieran, pero no sin antes cumplir con el tedioso pero necesario protocolo: el registro meticuloso sobre qué tipo de arma han tomado, la fecha y hora exactas de retirada y, por supuesto, la fecha y hora de la devolución en el momento que la devuelvan. Porque en la fortaleza, hasta el caos se maneja con un orden impecable y un desmedido amor por el papeleo.

realizar un modelo conceptual de datos mediante la técnica del **modelo Entidad-Relación de Chen** que modele **La Fortaleza**.

Solución propuesta:


nombre
Firewall
versión
(1,1)
tiene
(0,n)
CentroDeControl
(0,1)
controla
nombre
(1,n)
contiene
(1,1)
Nivel
(1,n)
protege
(0,m)
Guarda
(0,1)
cod_nivel
cod_guarda
(0,n)
lidera
(1,1)
(0,n)
(0,m)
tipo
Arma
(0,m)
(0,n)
arma
fecha_retirada
hora_retirada
fecha_devolución
hora_devolución
(0,n)
(0,m)
Arsenal
nombre
incluye

EJERCICIO 2. Bloque "SQL"

Dado el siguiente modelo relacional:

`CRIATURA (`<u>`Nombre`</u>`, Sexo, Especie)`

`TRATAMIENTO (`<u>`Nombre`</u>`, Descripción, Enfermedad)`

`PERSONAL (`<u>`Nombre`</u>`, Sexo, Especialidad, Rol)`

`APLICA (`<u>*`Criatura`*</u>`, `<u>*`Tratamiento`*</u>`, `<u>*`Personal`*</u>`, Fecha, Hora)`

Se pide:

(a) (½ Punto) Escriba una consulta en SQL que devuelva el `nombre` de las criaturas que nunca han recibido un `tratamiento` cuyo `nombre` sea '*tónico de bailes*'.

Solución propuesta:

```
SELECT nombre
FROM critura
WHERE nombre NOT IN (SELECT critaura
                     FROM aplica
                     WHERE tratamiento = 'tónico de bailes')
```

(b) (½ Punto) Escriba una consulta en SQL que devuelva el `nombre` del personal que ha aplicado algún tratamiento a las criaturas cuyo `nombre` sea '*Risueño Rob*' y '*Caféinomano Eduardo*'.

Solución propuesta:

```
SELECT nombre
FROM personal
WHERE nombre IN (SELECT personal
                 FROM aplica
                 WHERE criatura = 'Risueño Rob')
  AND nombre IN (SELECT personal
                 FROM aplica
                 WHERE criatura = 'Cafeinómano Eduardo')
```

(c) (½ Punto) Escriba una consulta en SQL que devuelva el `nombre` del tratamiento que se haya aplicado a todas las criaturas de la `especie` '*zombi*'.

Solución propuesta:

```
SELECT nombre
FROM tratamiento
WHERE NOT EXISTS (SELECT *
                  FROM criatura
                  WHERE especie = 'zombi'
                    AND NOT EXISTS (SELECT *
                                    FROM aplica
```

```
                                        WHERE aplica.tratamiento
                                          =
                                              tratamiento.
                                                 nombre
                                          AND aplica.criatura =
                                              criatura.nombre))
```

(d) (½ Punto) Escriba una consulta en SQL que devuelva el `nombre` del tratamiento o tratamientos más veces aplicado.

Solución propuesta:

```
SELECT tratamiento
FROM aplica
GROUP BY tratamiento
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM aplica
                        GROUP BY tratamiento);
```

(e) (½ Punto) Escriba una consulta en SQL que devuelva el `nombre` de las criaturas de la `especie` '*sirena*' a las que se le haya aplicado algun tratamiento por personal de la `especialidad` '*Pociones de la Eternidad*'.

Solución propuesta:

```
SELECT DISTINCT criatura.nombre
FROM aplica
    INNER JOIN criatura ON criatura.nombre = aplica.criatura
    INNER JOIN personal ON personal.nombre = aplica.personal
WHERE criatura.especie = 'sirena'
  AND personal.especialidad = 'Pociones de la Eternidad';
```

(f) (1 Punto) Escriba un procedimiento almacenado en SQL que dado el nombre de una `enfermedad`, que se pasará como parámetro de entrada, liste por pantalla el número de veces que se han aplicado tratamientos para dicha enfermedad a cada `especie`. Si una especie nunca ha recibido tratamientos no es necesario que aparezca en el listado. El listado debe aparecer ordenador de mayor a menor número de tratamientos aplicados.

Solución propuesta:

```
DELIMITER $$
CREATE PROCEDURE num_tratamientos_especie (IN enf VARCHAR(30)
   )
BEGIN
    SELECT criatura.especie, COUNT(*) AS num_veces
    FROM aplica
```

```
        INNER JOIN criatura ON criatura.nombre = aplica.
            criatura
        INNER JOIN tratamiento ON tratamiento.nombre = aplica
            .tratamiento
    WHERE tratamiento.enfermedad = enf
    GROUP BY criatura.especie
    ORDER BY num_veces DESC;
END$$
DELIMITER ;
```

(g) (1 Punto) Escriba un *trigger* en SQL que impida que se inserten nuevas aplicaciones de tratamiento a criaturas si ya se han aplicado antes aunque sean realizadas por personal diferente.

Solución propuesta:

```
DELIMITER $$
CREATE TRIGGER evita_tratamiento_duplicado BEFORE INSERT ON
    aplica
FOR EACH ROW
BEGIN
    DECLARE num_trat INTEGER;

    SELECT COUNT(*) INTO num_trata
    FROM aplica
    WHERE criatura = NEW.criatura
      AND tratamiento = NEW.tratamiento;

    IF num_trata > = THEN
        SIGNAL SQLSTATE '481516'
        SET MESSAGE_TEXT = 'No se puede aplicar dos veces el
            mismo
        tratamiento';
    END IF;
END$$
DELIMITER ;
```

(h) (1 Punto) Cree una vista en SQL que permita conocer el número de tratamientos aplicado por cada personal (las columnas de la vista deberán ser el `nombre` del personal y el número total de tratamiento aplicados). A continuación, cree un usuario denominado '*gerente*' que tenga únicamente permisos de consulta sobre esa vista.

Solución propuesta:

```
CREATE VIEW num_tratamientos_personal AS
SELECT personal, COUNT(*) AS num_tratamiento
FROM aplica
GROUP BY personal;
```

```
CREATE USER 'gerente' IDENTIFIED BY '1234';

GRANT SELECT ON num_tratamientos_personal TO 'gerente';
```

EJERCICIO 3. Bloque "Acceso programático"

(a) (1 Punto) Tomando como punto de partida un `SCHEMA` de bases de datos basado en el modelo relacional del bloque anterior, complete los huecos del siguiente código fuente en `python` de tal manera que se satisfaga la funcionalidad indicada como comentarios.

```
import mysql.connector

# establecimiento de conexion
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root"
)

cursor = mydb.cursor()

# añade personal
nombre = "Atenea la Moderada"
sexo = "M"
especialidad = "Curación de Maldiciones Graves"
rol = "Asistente"

cursor.execute("INSERT INTO personal
    ----------------------------------",
     (nombre, sexo, especialidad, rol))

mydb.commit()

# listar los nombres y sexo de las criaturas de especie "fénix"
especie = "fénix"
consulta = "SELECT nombre, sexo FROM criatura WHERE especie = %s
    "

cursor._______________(
    ------------------------------------------------)

for (nombre, sexo) in cursor:
     print(f"{nombre} ({sexo})")

# cierre de recursos
cursor.close()
cnx.close()
mydb.close()
```

Solución propuesta:

```
import mysql.connector

# establecimiento de conexion
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root"
```

```
)

cursor = mydb.cursor()

# añade personal
nombre = "Atenea la Moderada"
sexo = "M"
especialidad = "Curación de Maldiciones Graves"
rol = "Asistente"

cursor.execute("INSERT INTO personal VALUES (%s, %s, %s, %s)"
    ,
     (nombre, sexo, especialidad, rol))

mydb.commit()

# listar los nombres y sexo de las criaturas de especie "fé
    nix"
especie = "fénix"
consulta = "SELECT nombre, sexo FROM criatura WHERE especie =
     %s"

cursor.execute(consulta, (especie,))

for (nombre, sexo) in cursor:
    print(f"{nombre} ({sexo})")

# cierre de recursos
cursor.close()
cnx.close()
mydb.close()
```

EJERCICIO 4. Bloque "Acceso seguro"

(a) (½ Punto) Explique brevemente en qué consiste un ataque de *SQL Injection*

Solución propuesta:
Un ataque de *SQL Injection* es cuando un hacker inserta código malicioso en los campos de entrada de una aplicación web. Si la aplicación no filtra o valida correctamente estos datos, el código malicioso se ejecuta en la base de datos y el atacante puede obtener acceso no autorizado, robar información o manipular la base de datos. Para prevenir estos ataques, es importante que los desarrolladores implementen medidas de seguridad como el filtrado de entradas y el uso de consultas preparadas.