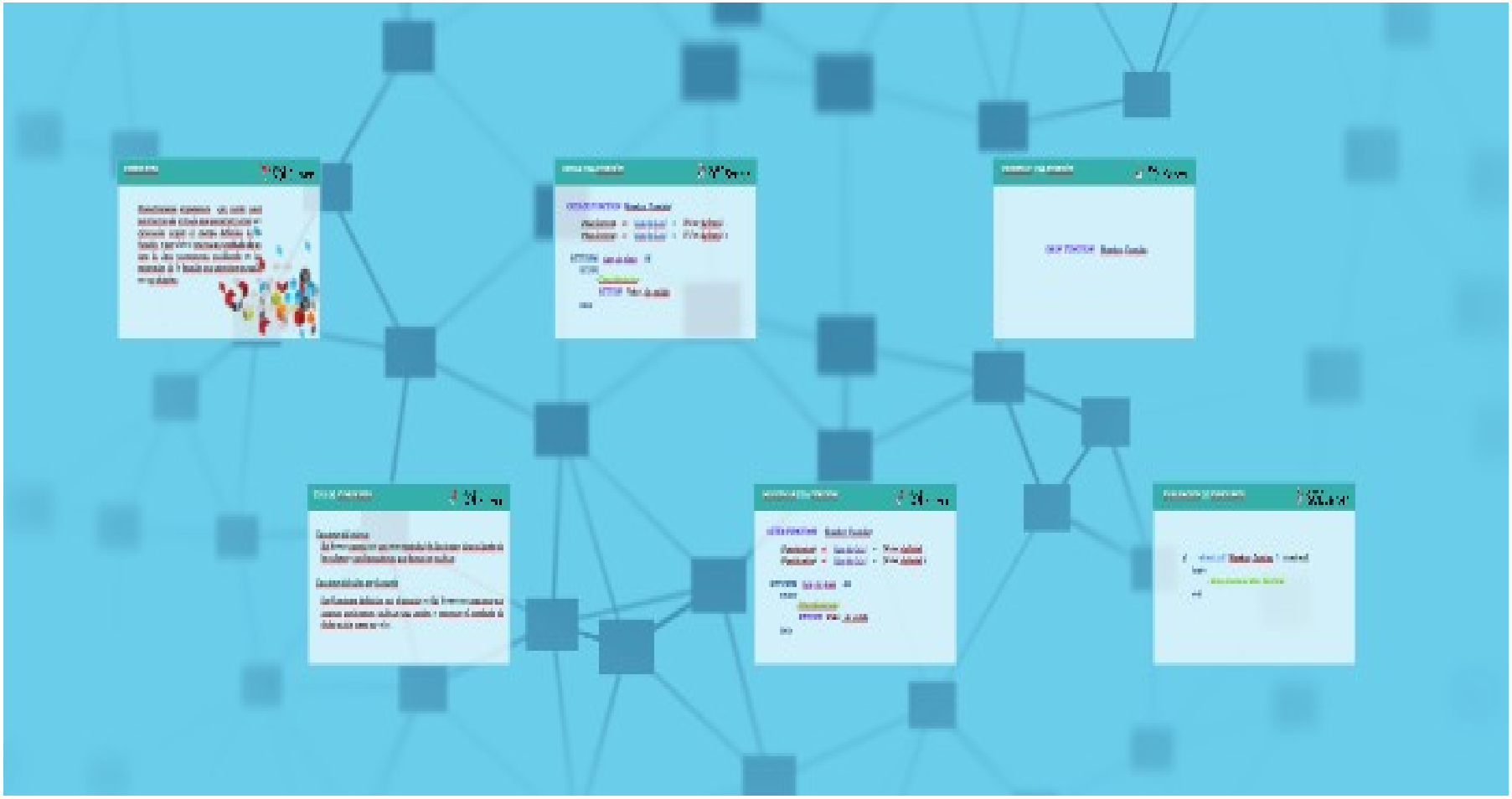
A5. Creación y depuración de Funciones de Usuario.

Índice.

A5. Creación y depuración de Funciones de Usuario.

1. ¿Qué es una Función de Usuario?



A5. Creación y depuración de Funciones de Usuario.

1. ¿Qué es una Función de Usuario?

Una **función almacenada (o de usuario)** es un conjunto de instrucciones SQL que se almacena asociado a una base de datos.

- Creación → CREATE FUNCTION
- Invocación → CALL

El procedimiento almacenado puede tener cero o muchos parámetros, que pueden ser de entrada, salida o de entrada y salida.

2. Sintaxis.

La sintaxis de para la creación de una función de usuario es la siguiente:

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }

routine_body:
    Valid SQL routine statement
```

3. Componentes.

Algunos componentes importantes son los siguientes:

- **Delimitadores** → la definición de un procedimiento almacenado necesita, temporalmente, modificar el carácter separador de las sentencias SQL que, por defecto, es ';'.

  Modificación del delimitador a '//': `DELIMITER //`

  Finalización del área: `// DELIMITER ;`

- **Parámetros de entrada** → TODOS los parámetros de una función SON de entrada y no requieren la palabra reservada IN antes del parámetro.

```
CREATE FUNCTION contar_productos(gama VARCHAR(50))
```

- **Resultado de salida** → una función SIEMPRE devolverá un valor de salida asociado al nombre de la función. En la definición de la cabecera de la función hay que definir el tipo de dato que devuelve con la palabra reservada RETURNS y en el cuerpo de la función debemos incluir la palabra reservada RETURN para devolver el valor de la función.

```
DELIMITER $$
DROP FUNCTION IF EXISTS contar_productos$$
CREATE FUNCTION contar_productos(gama VARCHAR(50))
  RETURNS INT UNSIGNED
  ...
BEGIN
  ...

  RETURN total;
END
$$
```

4. Características de la función.

Tras de la definición del tipo de dato que devolverá la función con la palabra reservada RETURNS, hay que indicar las características de la función. Las opciones disponibles son las siguientes:

- **DETERMINISTIC**: Indica que la función siempre devuelve el mismo resultado cuando se utilizan los mismos parámetros de entrada.
- **NOT DETERMINISTIC**: Indica que la función no siempre devuelve el mismo resultado, aunque se utilicen los mismos parámetros de entrada. Esta es la opción que se selecciona por defecto cuando no se indica una característica de forma explícita.
- **CONTAINS SQL**: Indica que la función contiene sentencias SQL, pero no contiene sentencias de manipulación de datos. Algunos ejemplos de sentencias SQL que pueden aparecer en este caso son operaciones con variables (Ej: SET @x = 1) o uso de funciones de MySQL (Ej: SELECT NOW();) entre otras. Pero en ningún caso aparecerán sentencias de escritura o lectura de datos.
- **NO SQL**: Indica que la función no contiene sentencias SQL.
- **READS SQL DATA**: Indica que la función no modifica los datos de la base de datos y que contiene sentencias de lectura de datos, como la sentencia SELECT.
- **MODIFIES SQL DATA**: Indica que la función sí modifica los datos de la base de datos y que contiene sentencias como INSERT, UPDATE o DELETE.

Para poder crear una función en MySQL es necesario indicar al menos una de estas tres características:

- DETERMINISTIC
- NO SQL
- READS SQL DATAas

5. Declaración de variables locales.

En las Funciones, como en los Procedimientos, se puede declarar variables locales con la palabra reservada DECLARE.

```
DECLARE var_name [, var_name] ... type [DEFAULT value]
```

El ámbito de la variable local es el bloque BEGIN ... END de la función en la que está definida.

Una restricción que hay que tener en cuenta es que se deben declarar antes de los cursores y de los handlers.

```
DELIMITER $$
DROP FUNCTION IF EXISTS contar_productos$$
CREATE FUNCTION contar_productos(gama VARCHAR(50))
  RETURNS INT UNSIGNED
BEGIN
  -- Paso 1. Declaramos una variable local
  DECLARE total INT UNSIGNED;

  -- Paso 2. Contamos los productos
  SET total = (
    SELECT COUNT(*)
    FROM producto
    WHERE producto.gama = gama);

  -- Paso 3. Devolvemos el resultado
  RETURN total;
END
$$

DELIMITER ;
```

6. Invocación de las Funciones.

La invocación (o llamada) a los procedimientos almacenados se realiza con la función SELECT:

```
SELECT contar_productos('Herramientas');
```

En este ejemplo, se llama a la función contar_productos pasando como parámetro de entrada '**Herramientas**', que es la gama de herramientas que se desea contar, y su número se devuelve y se muestra a través de la función SELECT.

7. Eliminación de una Función.

La eliminación de una función se realiza a través del comando DROP:

```
DROP  FUNCTION [IF EXISTS] sp_name
```

```
DELIMITER $$
DROP PROCEDURE IF EXISTS contar_productos$$
```