

## A1. Guiones simples de sentencias.

### Índice.

1.	¿Qué es la programación de Bases de Datos?	2
2.	Lenguaje PL/SQL	6
3.	Técnicas de mejora de SQL	9
3.1.	Vistas	11
3.2.	Guiones	14
3.3.	Eventos	16
3.4.	Procedimientos almacenados y funciones	18
3.5.	Disparadores o Triggers	22

## A1. Guiones simples de sentencias.

### 1. ¿Qué es la programación de Bases de Datos?

```
    o = e.length;
    a = M(e);
    if (n) {
        if (a) {
            for (; o > 1; i++)
                if (r = t.apply(e[i], n), r === !1) break
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === !1) break
        } else if (a) {
            for (; o > 1; i++)
                if (r = t.call(e[i], i, e[i]), r === !1) break
        } else
            for (i in e)
                if (r = t.call(e[i], i, e[i]), r === !1) break;
    return e
},
trim: b && !b.call("\uffeff\u00a0") ? function(e) {
    return null == e ? "" : b.call(e)
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
    var n = t || [];
    return null != e && (M(Object(e)) ? x.merge(n, "string" == typeof e ? [e] : e) : h.call(n, e)), n
},
isArray: function(e, t, n) {
    var r;
    if (t) {
        if (n) return m.call(t, e, n);
    }
}
```

## A1. Guiones simples de sentencias.

### 1. ¿Qué es la programación de Bases de Datos?

Hoy en día hay dos interpretaciones distintas:

- Por una parte, durante muchos años se ha entendido que la **programación de bases de datos** se refería a la creación de programas específicamente orientados a la gestión de los datos guardados en las bases de datos, como ERPs, CRMs y otros.



Enterprise Resource Planning



Customer Relationship Management

- Hoy en día se hablaría de **programación orientada a bases de datos**. Muchos sistemas de gestión de bases de datos (DBMS) incluyen lenguajes internos para programas Procedimientos Almacenados, Triggers y otros pequeños módulos que se programan y funcionan directamente con la base de datos.

```
CREATE PROCEDURE Add_User
(
    IN _Name VARCHAR(15),
    IN _lName VARCHAR(15),
    IN _Age INT,
    IN _Gender VARCHAR(15)
)
BEGIN
    INSERT INTO users (Name,lName,Age,Gender,Timezz) VALUES (_Name,_lName,_Age,_Gender,)
END

CREATE FUNCTION getProdSum(@prodID as INT)
RETURNS DECIMAL(5,2)
AS
BEGIN
    declare @sumProd DECIMAL(5,2)
    SELECT @sumProd = SUM(P.Price)
    FROM Products P
    INNER JOIN SalesOrderInfo S
    ON P.ProdID = S.ProdID
    WHERE P.ProdID = @prodID

    RETURN @sumProd
END

CREATE TRIGGER before_user_insert BEFORE INSERT ON users
FOR EACH ROW
SET NEW.full_name = CONCAT(NEW.first_name, ' ', NEW.last_name);
```

## A1. Guiones simples de sentencias.

### 1. ¿Qué es la programación de Bases de Datos?

A muchos de los lenguajes de Programación Orientada a Bases de Datos se les ha puesto nombre: Transact-SQL Server, PL/SQL de Oracle, PL/PgSQL de PostgreSQL, etc.

El uso de estos lenguajes tiene muchas aplicaciones, sobre todo en la técnica de la ciencia de los datos, porque permiten, por ejemplo, que los datos sean capaces de reaccionar ante ciertos estímulos, con independencia del software de aplicación que consulte con ellos.

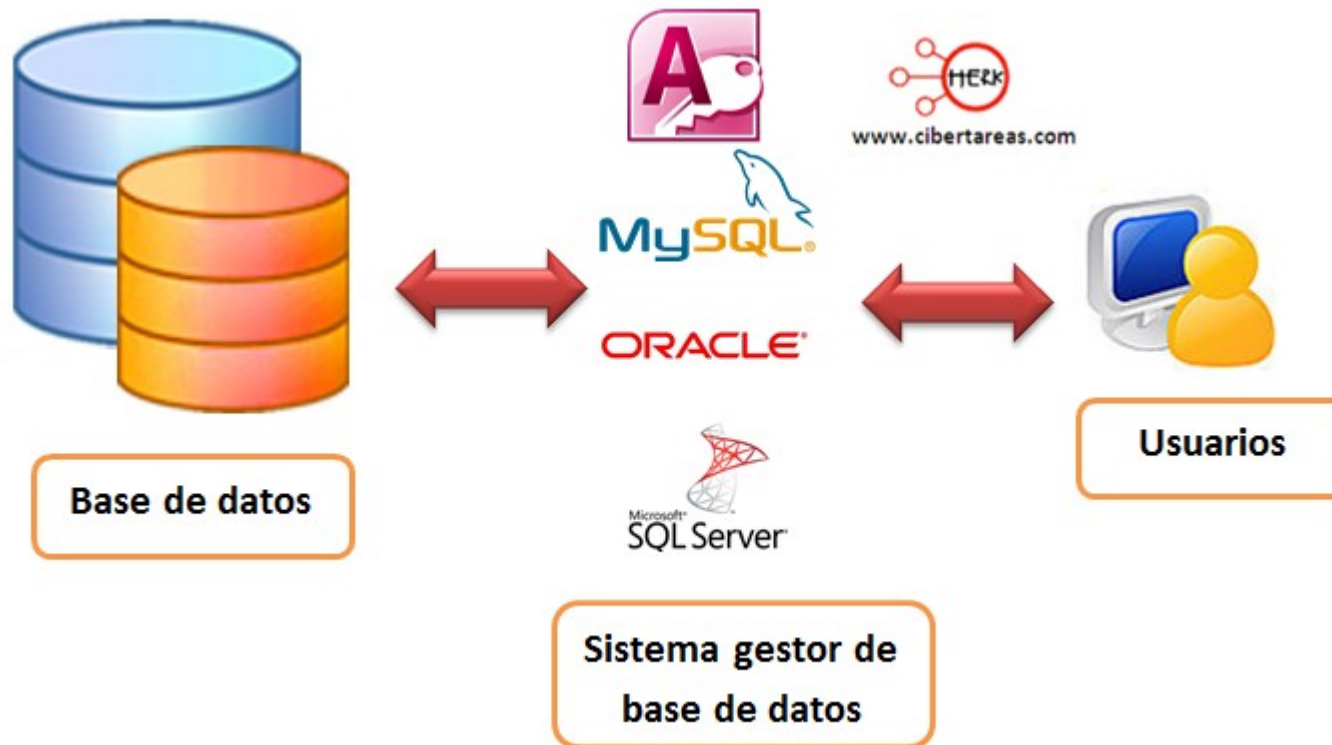
Este tipo de programación es fantástica porque permite crear el Modelo de Negocio en la propia Base de Datos, y además, hacer que los datos estén almacenados de un modo estructurado y reaccionen y adquieran un nivel más importante, en vez de ser meras figuras estáticas que sean meramente consultadas.



## A1. Guiones simples de sentencias.

### 1. ¿Qué es la programación de Bases de Datos?

La **Programación de Bases de Datos** es una rama de la informática con mucho futuro, aunque se perciba como complementaria frente a la programación tradicional.



## A1. Guiones simples de sentencias.

### 2. Lenguaje PL/SQL.

---

Hasta ahora la forma de interrogar a las bases de datos es mediante el empleo de sentencias SQL a través de un lenguaje declarativo con fuerte base matemática y cimentado sobre el álgebra relacional.

No obstante, SQL ofrece una gran potencia de interrogatorio y administración de la base de dato: pero también percibimos la existencia de ciertos tipos de preguntas (o acciones) que no se pueden realizar y que requieren un lenguaje más potente.



## A1. Guiones simples de sentencias.

### 2. Lenguaje PL/SQL.

---

El lenguaje PL/SQL (lenguaje procedimental) fue desarrollado por Oracle y extiende la potencia que ofrece SQL: PL/SQL (Procedural Language/Structured Query Language), un lenguaje de programación incrustado en Oracle.



## A1. Guiones simples de sentencias.

### 2. Lenguaje PL/SQL.

---

#### Características:

- Soporta el lenguaje de consultas (SQL), pero NO órdenes de definición de datos (DDL) ni de control (DCL).
- Presenta características propias de un lenguaje de programación:
  - × Uso de variables.
  - × Estructuras de control de flujo y de toma de decisiones.
  - × Control de excepciones.
  - × Reutilización de código a través de paquetes, procedimientos y funciones.
- Los programadores pueden escribir procedimientos (o funciones), bloques de código anónimo (como scripts) para ejecutarse desde SQL.
- El código desarrollado se puede almacenar como objetos en la base de datos, creando paquetes, funciones o procedimientos, que podrán ser reutilizados por los usuarios autorizados.
- El código se ejecuta desde el servidor y supone un ahorro en recursos por parte de los usuarios.
- El uso de disparadores (o triggers) permite la realización de una acción concreta sobre la base de datos cuando se va a realizar (o después) una modificación, inserción o borrado de registros sobre una tabla.
- La necesidad del uso de lenguajes procedimentales para trabajar con bases de datos motivó a Oracle la creación y desarrollo de PL/SQL para su SGBD.
- No todos los SGBD permiten utilizar PL/SQL aunque algunos lo han incorporado: DB2 y Times Ten in-memory.
- Otras compañías ofrecen soluciones parecidas, como PL/pgSQL de PostgreSQL.



## A1. Guiones simples de sentencias.

### 3. Técnicas de mejora de SQL.

Los SGBD han ido desarrollando todo un abanico de técnicas con le objetivo de ampliar la capacidad del lenguaje SQL y aumentar así la eficiencia en el desarrollo de aplicaciones sobre la base de datos.



## A1. Guiones simples de sentencias.

### 3. Técnicas de mejora de SQL.

Las técnicas empleadas para mejorar la eficiencia en SQL son las siguientes:

- **Vistas** → tabla virtual con contenido definido a partir de una consulta.
- **Guiones** → secuencia de sentencias SQL almacenadas en un fichero.
- **Eventos** → tareas programadas en un determinada fecha y hora.
- **Procedimientos almacenados y funciones** → códigos asociados para realizar tareas que han de ser invocados.
- **Disparadores o triggers** → objetos asociados a una tabla que se activan a través de un evento (insert, delete, update).



## A1. Guiones simples de sentencias.

### 3.1. Vistas.

La vista no es más que una **tabla virtual** cuyo contenido se define a través de una consulta.

Al ser una tabla, se halla dotada de columnas y filas de datos, y dotadas de un nombre.

NO EXISTE como un conjunto de valores de datos almacenados en la base de datos → SALVO que se halle indizada.

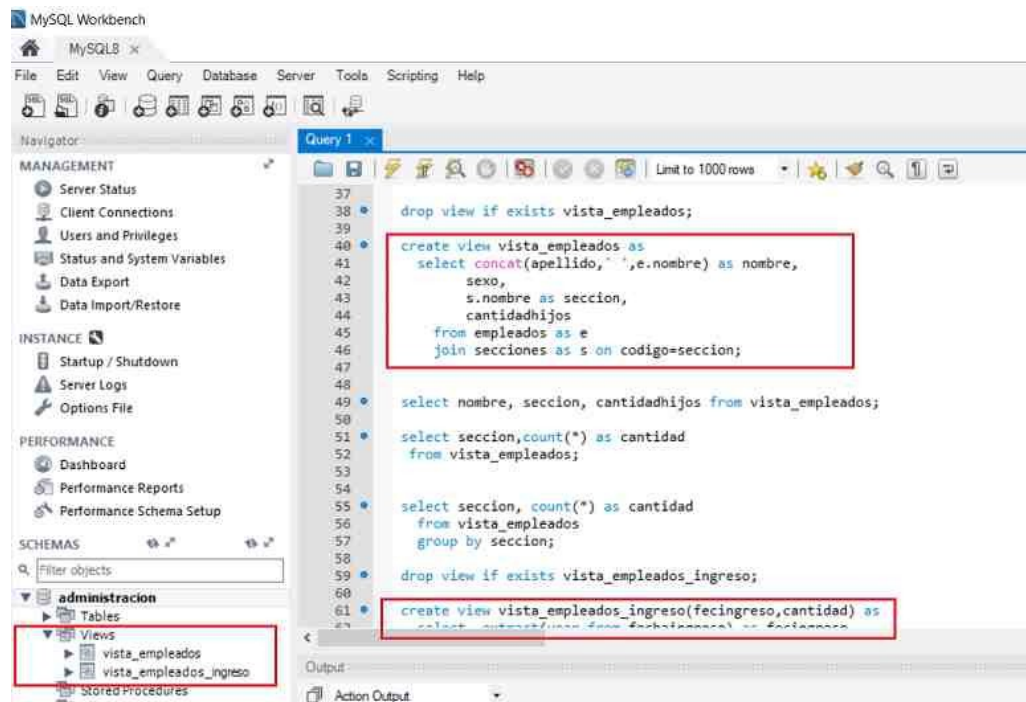


# A1. Guiones simples de sentencias.

## 3.1. Vistas.

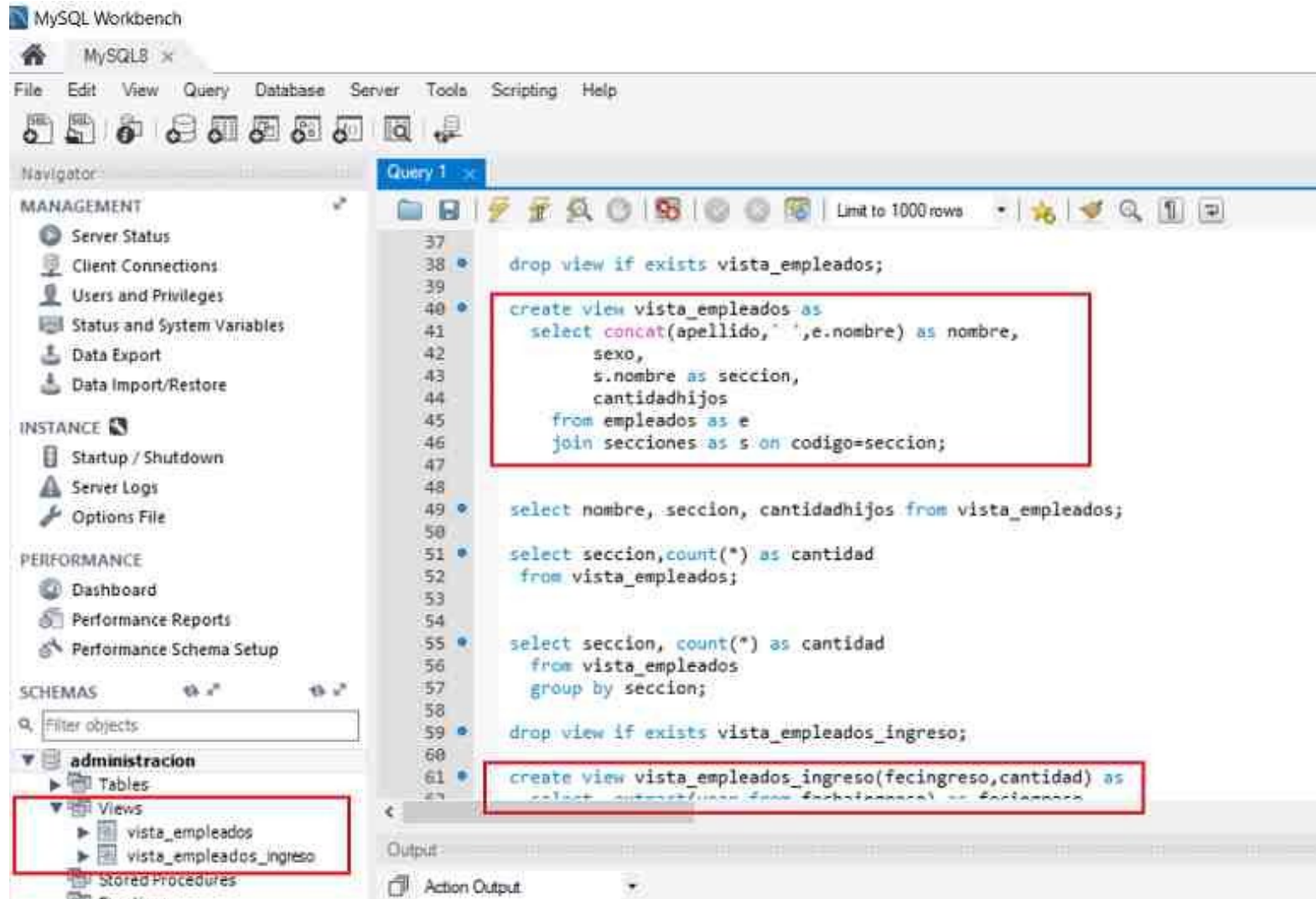
Algunas características que presentan la Vistas son las siguientes:

- Actúan como un filtro de las tablas a las que hace referencia.
- Se suelen utilizar para centrar, simplificar y personalizar la visión de la base de datos por cada usuario.
- Son un mecanismo de seguridad de acceso a datos, PERO sin tener acceso directo a las tablas.
- Proporcionan una interfaz compatible con versiones anteriores, simulando la existencia de una tabla cuyo esquema ha cambiado.
- Sirven para copiar datos entre sistemas SQL Server para mejorar el rendimiento y crear particiones de datos.



## A1. Guiones simples de sentencias.

### 3.1. Vistas.

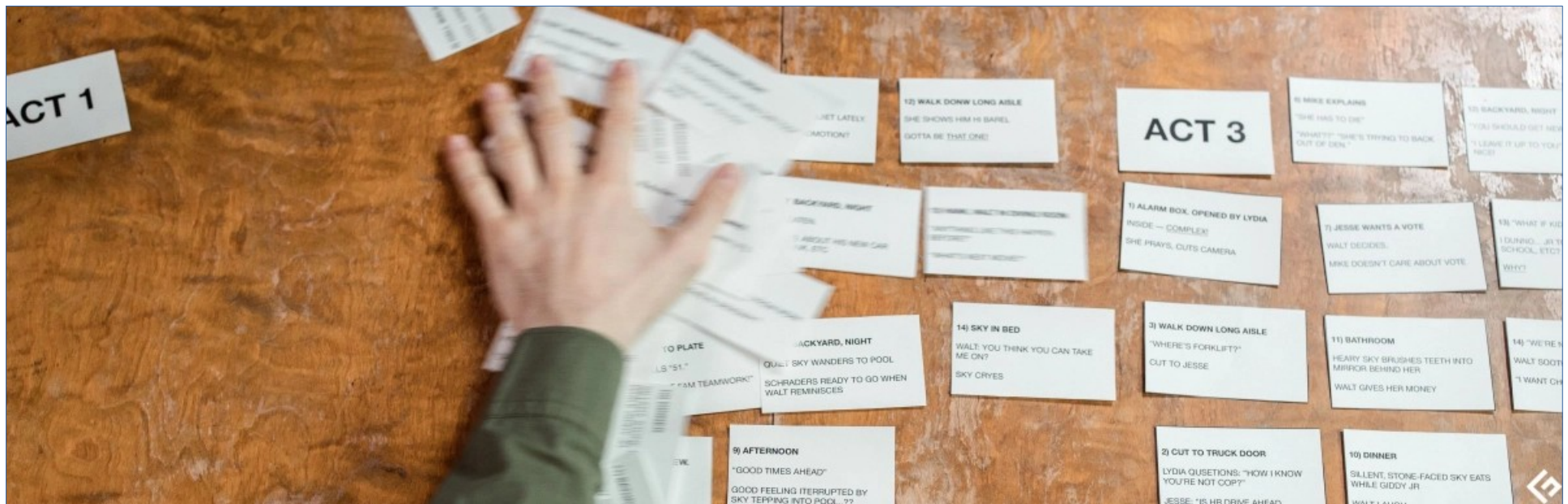




## A1. Guiones simples de sentencias.

### 3.2. Guiones.

Los **guiones** (o scripts), se guardan en texto plano bajo la extensión .sql y sirven para definir una serie de tareas que se ejecutarán en el SGBD.



Su principal ventaja recae en su facilidad de uso desde Consola o mediante una interfaz gráfica de gestión, permitiendo la generación de código SQL para la creación de Bases de Datos alojadas en el Servidor.

## A1. Guiones simples de sentencias.

### 3.2. Guiones.

---

Como ejemplo de creación de un guion, se creará un fichero SQL en el que:

- Se usará la base de datos SGR.
- Se creará una vista que ofrecerá los primeros diez comentarios.
- Finalizará dando permiso de selección al usuario Consultor.

```
USE sgr;  
CREATE OR REPLACE VIEW comentarios_p10 AS  
    SELECT id, nombre, fecha, comentario  
        FROM comentarios  
        LIMIT 10  
GRANT SELECT ON sgr.comentarios_p10 TO Consultor;
```

Una vez guardado el fichero, se puede ejecutar en cualquier momento.

La orden REPLACE VIEW permite reemplazar si existe una vista con el mismo nombre.

## A1. Guiones simples de sentencias.

### 3.3. Eventos.

Un **evento** es un acontecimiento que ocurre independientemente del funcionamiento de la aplicación.



El uso de eventos permite programar tareas, como eliminar registros, actualizarlos, pasarlos a un histórico, realizar nóminas a fin de mes, ...



## A1. Guiones simples de sentencias.

### 3.3. Eventos.

---

El evento puede ser lanzado en su creación, se puede indicar cada cuánto debe ejecutarse y durante cuánto tiempo durará su ejecución.

El uso de eventos debe activar el planificador de mysql con la variable global **event\_schedule** a ON:

```
SET GLOBAL event_schedule = ON;
```

Un ejemplo de evento es el siguiente:

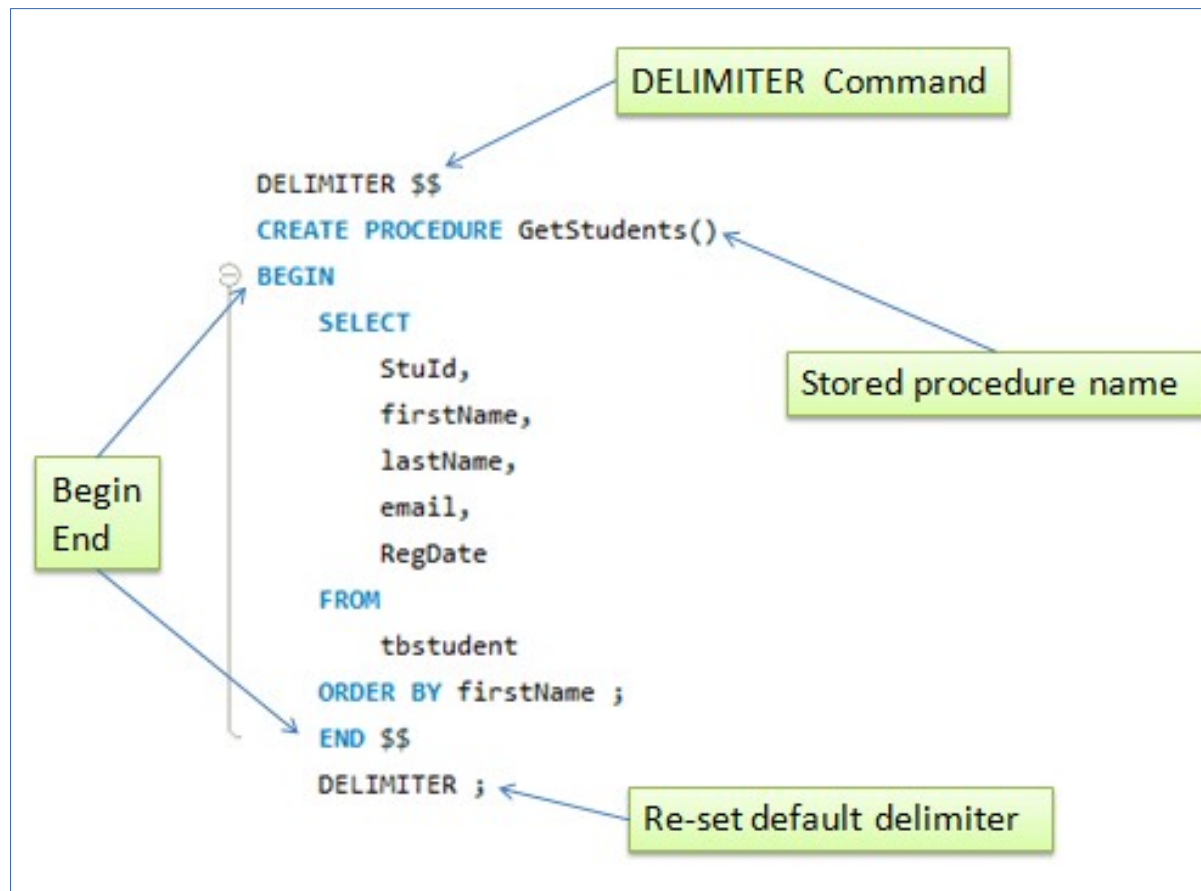
```
CREATE EVENT aumento_sueldo  
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 YEAR  
DO  
  UPDATE db.trabajador SET sueldo *= 1,05 ;
```

En este ejemplo, cada año se incrementa el sueldo de los trabajadores un 5%.

## A1. Guiones simples de sentencias.

### 3.4. Procedimientos almacenados y funciones.

Un **procedimiento almacenado** es una función alojada en la propia base de datos que puede ser llamada como si fuese un procedimiento de cualquier otro lenguaje de programación y que realiza una determinada tarea.



## A1. Guiones simples de sentencias.

### 3.4. Procedimientos almacenados y funciones.

Un ejemplo de un **procedimiento almacenado** es el siguiente:

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view containing 'administracion', 'libros', and 'Stored Procedures'. The 'Stored Procedures' folder is expanded, and 'pa\_libros\_limite\_stock' is selected. The main editor shows a SQL script for 'Query 1' with the following content:

```
27 insert into libros(titulo,autor,editorial,precio,stock)
28 values('Martin Fierro','Jose Hernandez','Emece',39.00, 70);
29 insert into libros(titulo,autor,editorial,precio,stock)
30 values('Uno','Richard Bach','Planeta',10.00, 120);
31
32 drop procedure if exists pa_libros_limite_stock;
33
34 delimiter //
35 create procedure pa_libros_limite_stock()
36 begin
37     select * from libros
38     where stock<=10;
39 end //
40 delimiter ;
41
42 call pa_libros_limite_stock();
43
```

Below the script, the 'Result Grid' shows the output of the stored procedure. The grid has columns: 'codigo', 'titulo', 'autor', 'editorial', 'precio', and 'stock'. The results are as follows:

	codigo	titulo	autor	editorial	precio	stock
▶	1	Alicia en el pais de las maravillas	Lewis Carroll	Emece	20.00	9
	3	Aprenda PHP	Mario Molina	Siglo XXI	40.00	3
	6	Java en 10 minutos	Mario Molina	Siglo XXI	50.00	7
	7	Martin Fierro	Jose Hernandez	Planeta	20.00	3

## A1. Guiones simples de sentencias.

### 3.4. Procedimientos almacenados y funciones.

Una **función** es un conjunto de sentencias, de forma similar a un procedimiento almacenado, pero con la diferencia de que debe devolver un valor simple (int, varchar, float, ...).

Una función tiene nombre, acepta parámetros de entrada y devuelve un valor **obligatoriamente**.

Las funciones pueden ser llamadas desde sentencias SELECT.

```
CREATE FUNCTION FunctionName
(
    @parameter1 Datatype,
    @Parameter2 datatype,
    @parametern datatype
)
Returns <Return attribute Data type>
AS
BEGIN
    --Function Body
    Return <Return data type>
End
```

## A1. Guiones simples de sentencias.

### 3.4. Procedimientos almacenados y funciones.

Un ejemplo de una **función** es el siguiente:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'administration' expanded, and 'Functions' containing 'f\_estrellas' highlighted with a red box. The main editor shows a SQL script with the following code:

```
15
16 • drop function if exists f_estrellas;
17
18 delimiter //
19 • create function f_estrellas(
20   cant tinyint)
21   returns varchar(15)
22   deterministic
23 begin
24   declare estrellas varchar(15) default '';
25   declare x int default 0;
26   while x<cant do
27     set estrellas=concat(estrellas,'*');
28     set x=x+1;
29   end while;
30   return estrellas;
31 end //
32 delimiter ;
33
34 • select url f_estrellas(estrellas) from sitios;
35
```

The function definition (lines 19-31) is enclosed in a red box. Below the editor, the 'Result Grid' shows the output of the query. The first column is 'url' and the second is 'f\_estrellas(estrellas)'. The results are:

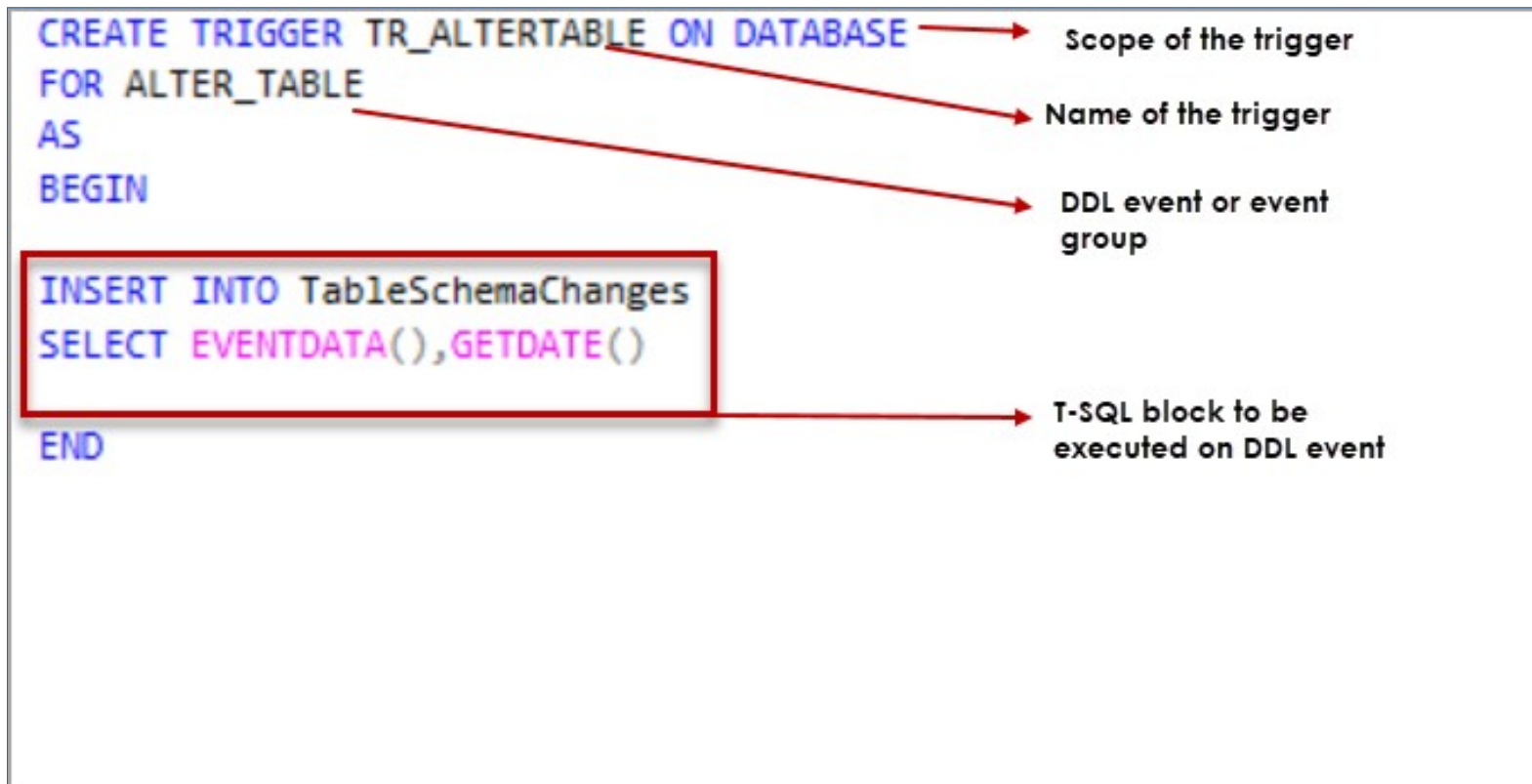
url	f_estrellas(estrellas)
clarin.com	***
infobae.com	*****
lanacion.com.ar	***
lavoz.com.ar	**

## A1. Guiones simples de sentencias.

### 3.5. Disparadores o Triggers.

Un **trigger** es un objeto de la base de datos que se halla asociado a una tabla y cuya activación dependerá de la ejecución de una acción definida: INSERT, UPDATE, DELETE.

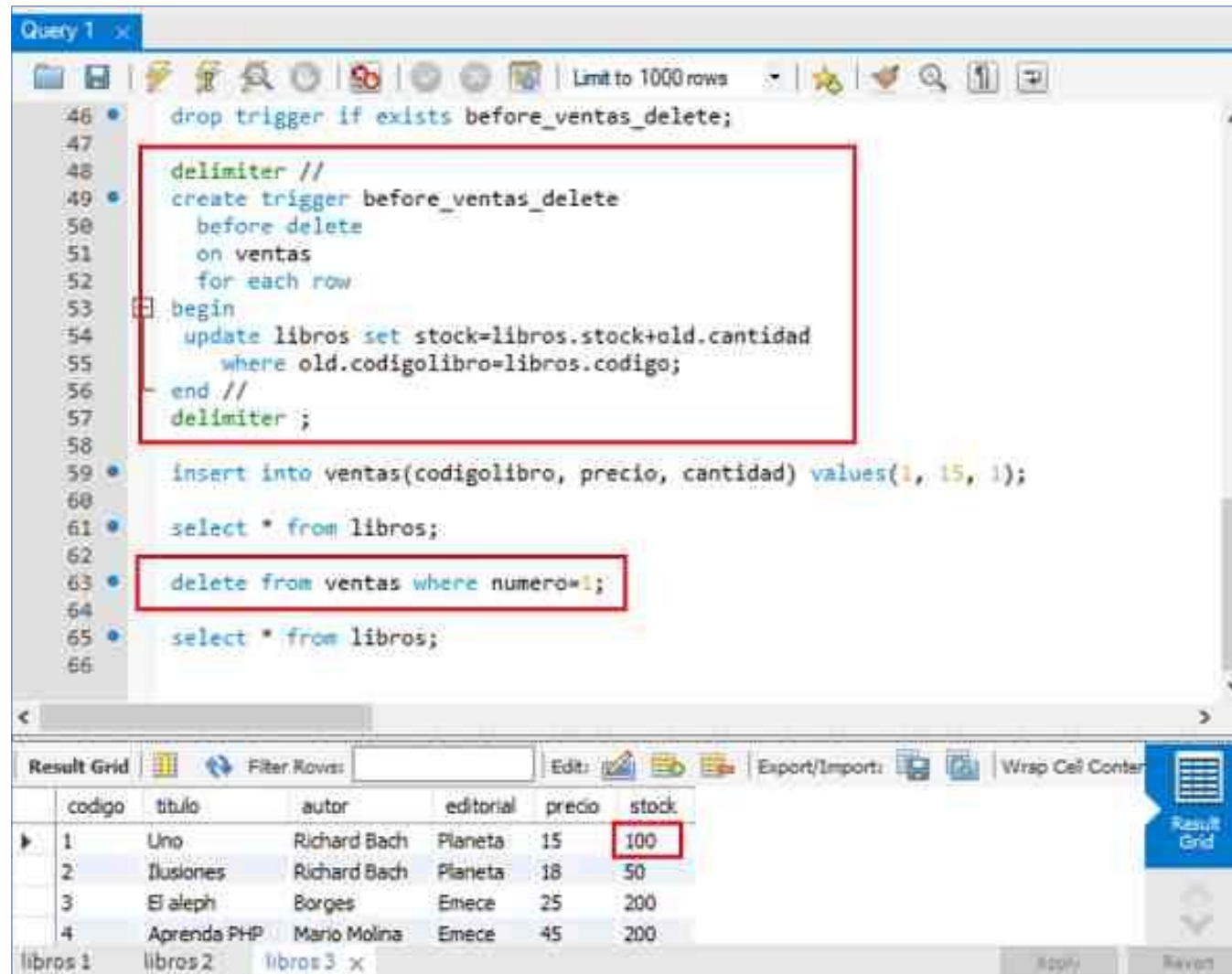
El **trigger** se puede invocar antes o después del evento.



## A1. Guiones simples de sentencias.

### 3.5. Disparadores o Triggers.

Un ejemplo de un **trigger** es el siguiente:



The screenshot shows a SQL query editor window titled "Query 1". The query is as follows:

```
46 • drop trigger if exists before_ventas_delete;
47
48 • delimiter //
49 • create trigger before_ventas_delete
50 •   before delete
51 •   on ventas
52 •   for each row
53 • begin
54 •   update libros set stock=libros.stock+old.cantidad
55 •   where old.codigolibro=libros.codigo;
56 • end //
57 • delimiter ;
58
59 • insert into ventas(codigolibro, precio, cantidad) values(1, 15, 1);
60
61 • select * from libros;
62
63 • delete from ventas where numero=1;
64
65 • select * from libros;
66
```

The script is designed to create a trigger named `before_ventas_delete` that updates the stock in the `libros` table whenever a row is deleted from the `ventas` table. The trigger is tested by inserting a row into `ventas`, selecting all rows from `libros`, deleting the first row from `ventas`, and selecting all rows from `libros` again.

The results of the query are displayed in a table below the script:

	codigo	titulo	autor	editorial	precio	stock
1	1	Uno	Richard Bach	Planeta	15	100
2	2	Ilusiones	Richard Bach	Planeta	18	50
3	3	El aleph	Borges	Emece	25	200
4	4	Aprenda PHP	Mario Molina	Emece	45	200

The `stock` value for the first row (codigo 1) is highlighted with a red box, showing the result of the trigger's update operation.