

# Eventos y Triggers

Conceptos fundamentales



The background is a solid dark blue. A large, semi-transparent light blue circle is positioned on the right side, partially overlapping a vertical line of a slightly different shade of blue that runs from the top to the bottom of the frame.

EVENTOS

# Definición

- Los **eventos** son tareas programadas que se ejecutan automáticamente en momentos específicos, similares a los *cron jobs* en sistemas Unix o MacOS, o al Programador de tareas en Windows.
- Su propósito es automatizar tareas repetitivas dentro del servidor de base de datos.
- A diferencia de los *triggers*, que responden a cambios en los datos, los *events* se activan en función del tiempo.

# Funcionamiento del Event Scheduler



`SET GLOBAL event_scheduler = ON;`




`SHOW PROCESSLIST;`



`SET GLOBAL event_scheduler = OFF;`

# Ciclo de Vida

- **Creación:** CREATE EVENT
  - **Activación:** ALTER EVENT ... ENABLE
  - **Modificación:** ALTER EVENT
  - **Desactivación:** ALTER EVENT ... DISABLE
  - **Eliminación:** DROP EVENT
- 

# Creación de Eventos Programados

- **Evento de una sola vez:**

```
CREATE EVENT nombre_evento  
ON SCHEDULE AT '2025-04-22 10:00:00'  
DO  
    -- instrucciones SQL;
```

- **Evento recurrente:**

```
CREATE EVENT nombre_evento  
ON SCHEDULE EVERY 1 DAY  
STARTS '2025-04-22 00:00:00'  
DO  
    -- instrucciones SQL;
```

# Consideraciones

Los eventos pueden ser **persistentes** o **no persistentes**.

Se pueden definir eventos que se repitan en **intervalos** específicos.

# Administración

- **Modificación de un evento:**

```
ALTER EVENT nombre_evento  
ON SCHEDULE EVERY 1 WEEK  
DO  
    -- nuevas instrucciones SQL;
```

- **Eliminación de un evento:**

```
DROP EVENT nombre_evento;
```

- **Visualización de eventos existentes:**

```
SHOW EVENTS;
```





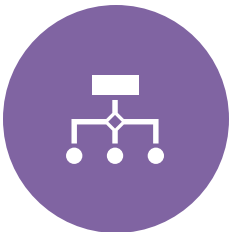
# Casos de Uso



**Respaldo de datos:**  
Automatizar copias de seguridad periódicas.



**Depuración de datos:**  
Eliminar registros antiguos para mantener el rendimiento.



**Generación de reportes:**  
Crear informes diarios o mensuales automáticamente.



**Mantenimiento:**  
Optimizar tablas o reconstruir índices en horarios de baja actividad.

# Mejores Prácticas

- **Seguridad:**  
Asegurar que solo usuarios autorizados puedan crear o modificar eventos.
- **Monitoreo:**  
Revisar regularmente los eventos programados y sus resultados.
- **Documentación:**  
Mantener una documentación clara de los eventos y sus propósitos.

# Conclusiones

- Los **eventos** son herramientas poderosas para la automatización de tareas dentro de la base de datos.
- Reducen la intervención manual.
- Mejoran la eficiencia y consistencia en tareas repetitivas.

The background consists of a dark blue field with a lighter blue square on the left and a semi-transparent blue circle on the right.

TRIGGERS

# Definición

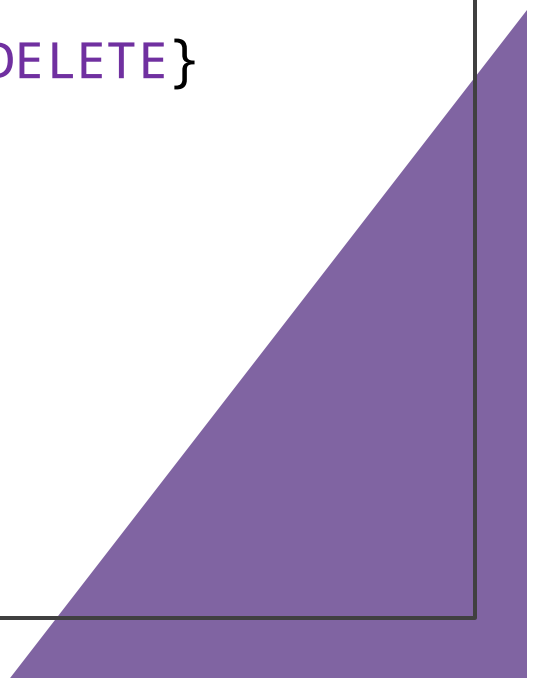
- Un ***trigger*** es un programa almacenado que se ejecuta automáticamente en respuesta a eventos como: INSERT, UPDATE o DELETE en una tabla asociada.
- Sirven para automatizar tareas como validación de datos, mantenimiento de auditorías o sincronización de tablas.
- A diferencia de los procedimientos almacenados, los *triggers* se activan automáticamente sin necesidad de ser llamados explícitamente.

# Tipos de Triggers en MySQL


- Eventos que pueden activar un *trigger*:
  - INSERT
  - UPDATE
  - DELETE
- Momentos de activación:
  - BEFORE: Antes de que se ejecute el evento.
  - AFTER: Después de que se ejecute el evento.
- Desde MySQL 5.7.2, es posible crear múltiples *triggers* para el mismo evento y momento de acción en una tabla.

# Sintaxis básica

```
CREATE TRIGGER nombre_trigger  
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}  
ON nombre_tabla  
FOR EACH ROW  
BEGIN  
    -- instrucciones  
END;
```



# Consideraciones

- Utilizar **NEW** y **OLD** para referirse a los valores nuevos y antiguos de las filas.
  - Los *triggers* no pueden contener instrucciones que inicien o finalicen transacciones como **START TRANSACTION**, **COMMIT** o **ROLLBACK**.
- 



# Administración

- Visualizar *triggers* existentes:

`SHOW TRIGGERS;`

- Eliminación un *trigger*:

`DROP TRIGGER IF EXISTS nombre_trigger;`

- Llamar procedimiento almacenado desde un *trigger*:

`CALL nombre_procedimiento();`

El procedimiento llamado no debe tener parámetros `OUT` o `INOUT`.

# Casos de Uso

- **Auditoría de cambios:**  
Utilizar un *trigger* **AFTER UPDATE** para registrar cambios en una tabla de auditoría.
- **Validación de datos:**  
Implementar un *trigger* **BEFORE INSERT** para validar datos antes de su inserción.
- **Mantenimiento de tablas resumen:**  
Usar un *trigger* **AFTER DELETE** para actualizar una tabla que mantiene un resumen de datos.

## Buenas prácticas

- No se pueden utilizar instrucciones que gestionen transacciones dentro de un *trigger*.
- Evitar operaciones complejas que puedan afectar el rendimiento.
- Mantener el código del *trigger* lo más simple y eficiente posible.
- Documentar claramente la finalidad de cada *trigger*.
- Realizar pruebas exhaustivas para asegurar el comportamiento esperado.

# Conclusiones

- Los ***triggers*** permiten automatizar tareas en respuesta a eventos específicos en las tablas.
- Son herramientas poderosas para mantener la integridad y coherencia de los datos.

# Preguntas y Discusión

Espacio para resolver dudas y discutir aplicaciones prácticas