

**UNIVERSIDAD PERUANA  
LOS ANDES**

**“FACULTAD DE INGENIERÍA”**



**ESCUELA PROFESIONAL “SISTEMAS Y  
COMPUTACIÓN”**

**Automatización y mantenimiento**

**CURSO: Base de Datos II**

**DOCENTE: Ing. Fernández Bejarano Raúl Enrique**

**ESTUDIANTE: Carrillo Cárdenas José**

**CICLO: V**

**HUANCAYO PERÚ**

**2025**

## **Semana 14: Automatización y mantenimiento**

**Objetivo: Administrar tareas repetitivas y asegurar la integridad del sistema.**

**Temas:**

### **1. Creación de jobs y alerts con SQL Server Agent.**

---

La automatización y el mantenimiento de SQL se refieren al uso de herramientas y procesos para realizar tareas de rutina en una base de datos, como copias de seguridad, optimización de índices y limpieza de datos, de forma automática en lugar de manual. Esto mejora la eficiencia operativa al reducir errores humanos y permite que el personal se centre en tareas más estratégicas.

#### **JOB (TRABAJOS PROGRAMADOS)**

¿Qué es un Job?

Un job es un conjunto de tareas (steps) que SQL Server ejecuta de manera automática según una programación definida (schedule).

Sirve para automatizar cualquier proceso:

Backups diarios

Limpieza de logs

Regeneración de índices

Envío de alertas

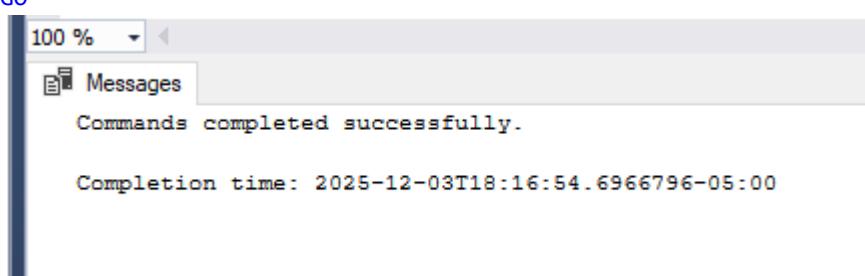
Copia a otras bases

Ejecución de scripts SQL o PowerShell

## **Script en SQL**

Crear operador

```
EXEC msdb.dbo.sp_add_operator  
    @name = N'OperadorSoporte',  
    @email_address = N'soporte@empresa.com';  
GO
```



*Crear el JOB*

```
EXEC msdb.dbo.sp_add_job  
    @job_name = N'JOB_BackupFullDiario',  
    @description = N'Backup FULL automático diario de la base QhatuPeru',  
    @enabled = 1;  
GO
```

```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:17:29.0329516-05:00
```

### Agregar el paso del job

```
EXEC msdb.dbo.sp_add_jobstep
    @job_name = N'JOB_BackupFullDiario',
    @step_name = N'BackupFull',
    @subsystem = N'TSQL',
    @command = N'
DECLARE @Ruta NVARCHAR(200) = ''C:\BackupsSQL\QhatuPeru'';
DECLARE @Fecha VARCHAR(14) = REPLACE(CONVERT(VARCHAR(20), GETDATE(), 120), '':'','');
SET @Fecha = REPLACE(@Fecha, '-'', '')';

DECLARE @Archivo NVARCHAR(300) = @Ruta + ''BackupFULL_'' + @Fecha + '''.bak'';

BACKUP DATABASE QhatuPeru
TO DISK = @Archivo
WITH INIT, COMPRESSION, CHECKSUM, STATS = 10;
';
GO
```

```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:18:09.6261228-05:00
```

### Crear el schedule

```
EXEC msdb.dbo.sp_add_schedule
    @schedule_name = N'Diario_2AM',
    @freq_type = 4, -- diario
    @freq_interval = 1,
    @active_start_time = 020000; -- 2:00 AM
GO
```

```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:18:43.1536870-05:00
```

### Asociar el schedule al job

```
EXEC msdb.dbo.sp_attach_schedule
    @job_name = N'JOB_BackupFullDiario',
    @schedule_name = N'Diario_3AM';
GO
```

```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:20:28.4372008-05:00
```

Asociar operador

```
EXEC msdb.dbo.sp_update_job
    @job_name = N'JOB_BackupFullDiario',
    @notify_email_operator_name = N'OperadorSoporte',
    @notify_level_email = 2;      -- 1=Sucess   2=Failure   3=Completion
```

GO

```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:23:58.1192871-05:00
```

## VALIDAR EN SQL

```
SELECT
    j.name AS JobName,
    j.notify_email_operator_id,
    o.name AS OperatorName,
    j.notify_level_email
FROM msdb.dbo.sysjobs j
LEFT JOIN msdb.dbo.sysoperators o
    ON j.notify_email_operator_id = o.id
WHERE j.name = 'JOB_BackupFullDiario';
```

```
100 % 
Results  Messages
```

	JobName	notify_email_operator_id	OperatorName	notify_level_email
1	JOB_BackupFullDiario	1	OperadorSoporte	2

## JUSTIFICACIÓN TÉCNICA

SQL Server Agent permite automatizar tareas críticas como backups, mantenimiento e integridad.

El uso de operadores garantiza que el equipo reciba alertas sin depender de supervisión manual.

sp\_update\_job es el procedimiento oficial para enlazar notificaciones a un JOB.

Evita errores usando procedimientos adecuados para cada tipo de objeto (jobs, alerts, operators).

## BUENAS PRÁCTICAS APLICADAS

- ✓ Crear un operador único para centralizar las notificaciones.
- ✓ Enviar alertas solo por fallas para evitar spam innecesario.
- ✓ Mantener nombres claros y consistentes (“OperadorSoporte”, “JOB\_BackupFullDiario”).
- ✓ Validar con consultas del sistema (sysjobs, sysoperators).
- ✓ Documentar cada job y su configuración de alertas.
- ✓ Usar cuentas SMTP corporativas configuradas en Database Mail.

## Semana 14: Automatización y mantenimiento

**Objetivo: Administrar tareas repetitivas y asegurar la integridad del sistema.**

**Temas:**

### 2. Uso de planes de mantenimiento (Database Maintenance Plans).

---

Los planes de mantenimiento en SQL Server son una herramienta que permite programar y automatizar tareas de administración esenciales para el correcto funcionamiento y rendimiento de la base de datos. Estos planes se crean como un paquete de SQL Server Integration Services (SSIS), que se ejecutan a través del Agente SQL Server y pueden incluir tareas como copias de seguridad, verificación de integridad, reorganización de índices y limpieza del historial.

### -Script en SQL

Backup full diario

```
-- Backup full diario (modificar @BackupFolder)
-- Backup full diario (modificar @BackupFolder)
DECLARE @sql NVARCHAR(MAX);
DECLARE @DBName SYSNAME = N'QhatuPERU';
DECLARE @BackupFolder NVARCHAR(260) = N'D:\Backups\';
DECLARE @DateSuffix NVARCHAR(20) = FORMAT(GETDATE(), 'yyyyMMdd_HHmmss');
DECLARE @BackupFile NVARCHAR(400) = @BackupFolder + @DBName + '_FULL_' + @DateSuffix +
'.bak';

SET @sql = N'
BACKUP DATABASE ' + QUOTENAME(@DBName) + N'
TO DISK = N''' + @BackupFile + N'''
WITH INIT, COMPRESSION, STATS = 10;
';

EXEC sp_executesql @sql;
```

Messages

```

10 percent processed.
20 percent processed.
30 percent processed.
40 percent processed.
50 percent processed.
60 percent processed.
70 percent processed.
80 percent processed.
90 percent processed.
100 percent processed.
Processed 139296 pages for database 'QhatuPERU', file 'QhatuPERU' on file 1.
Processed 1 pages for database 'QhatuPERU', file 'QhatuPERU_log' on file 1.
BACKUP DATABASE successfully processed 139297 pages in 2.748 seconds (396.015 MB/sec).

Completion time: 2025-12-03T18:36:15.1454306-05:00

```

100 %

Query executed successfully. | DESKTOP-98FJL7B (16.0 RTM)

### Backup de logs

```

DECLARE @DBName sysname = N'QhatuPERU';
DECLARE @BackupFolder nvarchar(260) = N'D:\Backups\QhatuPERU\Logs\' ;
DECLARE @DateSuffix nvarchar(20) = FORMAT(GETDATE(), 'yyyyMMdd_HHmmss');
DECLARE @BackupFile nvarchar(400) = @BackupFolder + @DBName + '_LOG_' + @DateSuffix +
'.trn';
DECLARE @sql nvarchar(max);

SET @sql = N'BACKUP LOG ' + QUOTENAME(@DBName) + N' TO DISK = N''' + @BackupFile +
N''' WITH NOFORMAT, INIT, NAME = N''' + @DBName + N' Log Backup'', SKIP, NOREWIND,
NOUNLOAD, STATS = 10';

EXEC sp_executesql @sql;

```

Messages

```

10 percent processed.
20 percent processed.
30 percent processed.
40 percent processed.
50 percent processed.
60 percent processed.
70 percent processed.
80 percent processed.
90 percent processed.
100 percent processed.
Processed 225791 pages for database 'QhatuPERU', file 'QhatuPERU_log' on file 1.
BACKUP LOG successfully processed 225791 pages in 18.786 seconds (93.898 MB/sec).

Completion time: 2025-12-03T18:39:56.0795191-05:00

```

100 %

Query executed successfully. | DESKTOP-98FJL7B (16.0 RTM)

### DBCC CHECKDB con salida a tabla/log

-- Ejecutar CHECKDB y guardar resultado en tabla temporal/log (ejecutar en horarios de baja carga)

```
DECLARE @DBName sysname = N'QhatuPERU';
```

```

DECLARE @Command nvarchar(max) = N'DBCC CHECKDB(' + QUOTENAME(@DBName) + N') WITH
NO_INFOMSGS, ALL_ERRORMSG;
EXEC (@Command);
-- Recomendación: redirigir la salida a files o a Job history; o ejecutar desde
Powershell y guardar salida en archivo.

```

100 % ▶ Messages  
Commands completed successfully.  
Completion time: 2025-12-03T18:41:04.6538595-05:00

### Update statistics

```

-- Actualiza estadísticas completas (no sample) para una DB concreta
DECLARE @DBName sysname = N'QhatuPERU';
DECLARE @sql nvarchar(max) = N'USE ' + QUOTENAME(@DBName) + N'; EXEC sp_updatestats';
EXEC sp_executesql @sql;

```

100 % ▶ Messages  
Updating [dbo].[Ventas]  
[PK\_Ventas\_BC1240BD71CF9119], update is not necessary...  
0 index(es)/statistic(s) have been updated, 1 did not require update.  
  
Updating [dbo].[Pedidos]  
[PK\_Pedidos\_9D335DC32D74D3DA], update is not necessary...  
0 index(es)/statistic(s) have been updated, 1 did not require update.  
  
Updating [sys].[sqlagent\_jobs]  
[sqlagent\_jobs\_clust], update is not necessary...  
[sqlagent\_jobs\_nc1\_name], update is not necessary...  
0 index(es)/statistic(s) have been updated, 2 did not require update.  
  
Updating [sys].[sqlagent\_jobsteps]  
[sqlagent\_jobsteps\_clust], update is not necessary...  
[--]-----  
100 % ▶ DESKTOP-98FJL7B (16.0 R)  
Query executed successfully.

### Justificación técnica

- Protección de datos (Backups): backups regulares permiten recuperación ante fallos de hardware, corrupción lógica, borrado accidental o desastres. Sin backups probados, la pérdida de datos puede ser irreversible.
- Integridad de la base (DBCC CHECKDB): detecta corrupción física o lógica en los archivos o páginas; ejecutarlo periódicamente permite detectar problemas temprano y restaurar desde backups antes de propagación.
- Rendimiento (índices y estadísticas): índices fragmentados y estadísticas desactualizadas degradan consultas (planificador elige planes subóptimos). Reorganizar/reconstruir índices y actualizar estadísticas mantiene tiempos de respuesta predecibles.

-Disponibilidad y cumplimiento: para SLAs es necesario un proceso repetible y automatizado que garantice copias consistentes y retención de backups por políticas regulatorias.

-Auditoría y trazabilidad: Planes y jobs generan historial que facilita auditar que las tareas críticas se ejecutaron según política.

---

#### **4) Buenas prácticas (prácticas recomendadas y tangibles)**

-Política de retención clara: define qué backups conservar (full, differential, log) y por cuánto tiempo; mantén copias fuera del sitio (offsite) o en cloud.

-Prueba de restauración periódica: un backup no es válido hasta que se restaura con éxito. Programa restores regulares (ej. semanal en entornos de test) y documenta tiempos de recuperación (RTO/RPO).

-Separar responsabilidades y jobs: crea jobs independientes para backups, integrity checks y mantenimiento de índices. Así, fallos en una tarea no afectan a las otras y es más fácil depurar.

-Ventanas de mantenimiento: programa tareas intensivas (REBUILD, CHECKDB) en horarios de baja carga; para bases grandes, considera ejecutar CHECKDB en réplicas o usar estrategias como snapshots si aplica.

-Uso de opciones ONLINE: cuando la edición de índices lo permita y el edition/licencia lo soporte, usar ALTER INDEX ... REBUILD WITH (ONLINE = ON) para minimizar bloqueo; si no está disponible, planear downtime.

-Compresión de backups: reduce espacio y tiempo de I/O; verificar compatibilidad con versión/edición de SQL Server.

-Monitoreo y alertas: configura Database Mail y alertas en SQL Agent para notificar fallas de jobs. Automatiza la recolección de métricas (duración de job, tamaño de backup).

-Evitar xp\_cmdshell si es posible: por seguridad, realizar tareas de sistema (borra archivos, copia a NAS) mediante PowerShell o tareas del sistema con cuentas controladas.

-Documentación y runbooks: tener pasos operativos para recuperar (playbook): cómo restaurar, qué backups usar, pasos para recovery parcial, contactos, y validaciones post-restore.

-Automatizar retención y rotación: no acumular backups en disco local. Implementa limpieza basada en fecha y políticas (ej. mover diarios a cold storage cada X días).

-Versionado de scripts y cambios: guarda los scripts de mantenimiento en control de versiones (Git) para auditoría y rollback.

-Considerar soluciones probadas: utiliza soluciones comunitarias/robustas (scripts conocidos y probados) en lugar de reinventar desde cero. (Ej. scripts de mantenimiento ampliamente usados en la comunidad).

#### **Semana 14: Automatización y mantenimiento**

**Objetivo:** Administrar tareas repetitivas y asegurar la integridad del sistema.

**Temas:**

### 3. Automatización con T-SQL y PowerShell.

En el contexto de la automatización con T-SQL y PowerShell, SQL se refiere al motor de base de datos de Microsoft SQL Server y su lenguaje, Transact-SQL (T-SQL). T-SQL es el lenguaje de programación nativo para ejecutar lógica dentro de la base de datos, como almacenar valores, aplicar condicionales y ejecutar procedimientos. PowerShell es un lenguaje de scripting que, en este contexto, se utiliza para automatizar tareas administrativas externas a la base de datos, como crear copias de seguridad, restaurar bases de datos o ejecutar consultas T-SQL de forma remota mediante el uso de módulos específicos

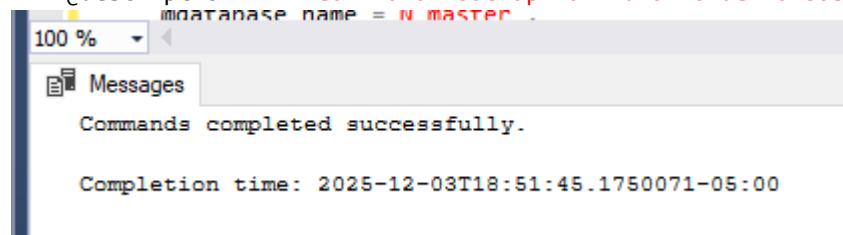
#### -Script en SQL

Crear un Job que realice un Backup Full automáticamente

```
EXEC sp_add_job
    @job_name = N'Job_BackupFull_Automatico',
    @description = N'Realiza un backup full diario de la base de datos MiBaseDatos';
    @database_name = N'master';

-- 2. Crear el Step (código T-SQL que realiza el respaldo)
EXEC sp_add_jobstep
    @job_name = N'Job_BackupFull_Automatico',
    @step_name = N'Realizar Backup Full',
    @subsystem = N'TSQL',
    @database_name = N'master',
    @command = N'
        DECLARE @File NVARCHAR(200);
        SET @File = ''D:\Backups\MiBaseDatos_FULL_''
                    + CONVERT(VARCHAR(20), GETDATE(), 112)
                    + '''.bak'';

        BACKUP DATABASE MiBaseDatos
        TO DISK = @File
        WITH INIT, COMPRESSION, STATS = 10;
    ',
    @on_fail_action = 2,
    @on_success_action = 1;
GO
```



```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:52:04.9461139-05:00

-- 3. Crear un horario diario 2:00 AM
-----
EXEC sp_add_jobschedule
    @job_name = N'Job_BackupFull_Automatico',
    @name = N'Schedule_Diario_2AM',
    @freq_type = 4,          -- diario
    @freq_interval = 1,
    @active_start_time = 020000;   -- 02:00 AM
GO
```

```
100 % 
Messages
Commands completed successfully.

Completion time: 2025-12-03T18:52:17.7246133-05:00

-- 4. Asignar el job al servidor actual
-----
EXEC sp_add_jobserver
    @job_name = N'Job_BackupFull_Automatico',
    @server_name = N'(local)';
GO
```

¿Qué automatiza este script?

- ✓ Crea un Job.
- ✓ Hace backup full automáticamente.
- ✓ Programa ejecución diaria.
- ✓ Ejecuta sin intervención humana.

## Justificación Técnica

Automatizar con T-SQL y PowerShell es fundamental porque:

1. Reduce errores humanos

Las tareas manuales como backups, mantenimiento o limpieza de archivos son propensas a fallos. La automatización estandariza procesos y evita omisiones.

## 2. Asegura continuidad operativa (24/7)

Los trabajos se ejecutan a horas donde no hay personal, eliminando la dependencia del operador.

## 3. Mejora rendimiento y salud del sistema

Al programar:

mantenimiento de índices

actualización de estadísticas

verificación de integridad

backups confiables

...se evita degradación del rendimiento y corrupción silenciosa.

## 4. Integración total del ecosistema Microsoft

PowerShell tiene acceso a:

SQL Server

Sistema operativo

Servicios del servidor

Archivos del sistema

Automatización de notificaciones

Combinando T-SQL + PowerShell se logra una administración completa.

## 5. Facilita auditorías y cumplimiento

La automatización genera historial verificable:

qué se ejecutó

cuándo

duración

resultado

Es un estándar requerido en entornos profesionales.

---

## Buenas Prácticas para Automatización con T-SQL y PowerShell

### 1. Separar trabajos por función

Ejemplo recomendable:

Job de Backups

Job de Mantenimiento de índices

Job de CHECKDB

Job de limpieza de archivos

Job de monitoreo

Evita que fallos en una tarea detengan otras.

## Semana 14: Automatización y mantenimiento

**Objetivo: Administrar tareas repetitivas y asegurar la integridad del sistema.**

**Temas:**

---

### 4. Monitoreo proactivo (correo, logs, alertas de rendimiento).

---

En SQL, el monitoreo proactivo consiste en configurar alertas y recopilar datos de forma continua para detectar y solucionar problemas antes de que afecten a los

usuarios. Esto se logra usando herramientas para monitorear logs de eventos, métricas de rendimiento (como uso de CPU y memoria) y estado general de la base de datos, y enviando notificaciones automáticas por correo electrónico o mediante otras herramientas cuando se detecta una anomalía.

## -Script en SQL

Habilitar Database Mail (Correo SQL Server)

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'Database Mail XPs', 1;
RECONFIGURE;
GO
```

```
100 % Messages
Configuration option 'show advanced options' changed from 1 to 1. Run the RECONFIGURE statement to install.
Configuration option 'Database Mail XPs' changed from 0 to 1. Run the RECONFIGURE statement to install.

Completion time: 2025-12-03T18:58:10.6148901-05:00
```

```
-- Crear perfil de correo
EXEC msdb.dbo.sysmail_add_profile_sp
    @profile_name = 'PerfilNotificaciones',
    @description = 'Perfil para enviar alertas de SQL Server';
```

```
100 % Messages
Commands completed successfully.

Completion time: 2025-12-03T18:58:26.7096346-05:00
```

```
-- Crear cuenta asociada
EXEC msdb.dbo.sysmail_add_account_sp
    @account_name = 'CuentaSQL',
    @description = 'Cuenta SMTP',
    @email_address = 'alertas.sql@midominio.com',
    @display_name = 'SQL Server Monitor',
    @mailserver_name = 'smtp.midominio.com',
    @port = 587;
```

```
100 % Messages
Commands completed successfully.

Completion time: 2025-12-03T18:58:39.3655442-05:00
```

```
-- Asociar cuenta con el perfil
EXEC msdb.dbo.sysmail_add_profileaccount_sp
    @profile_name = 'PerfilNotificaciones',
    @account_name = 'CuentaSQL',
    @sequence_number = 1;
GO
```

```
100 % ▾ Messages
Commands completed successfully.

Completion time: 2025-12-03T18:58:49.8854987-05:00
```

Enviar un correo de prueba

```
EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'PerfilNotificaciones',
    @recipients = 'soporte@midominio.com',
    @subject = 'Prueba de correo',
    @body = 'El servicio de correo está funcionando correctamente.';
```

```
100 % ▾ Messages
Mail (Id: 1) queued.

Completion time: 2025-12-03T18:59:10.4259871-05:00
```

Crear una alerta por error crítico (Severity 16 o superior)

```
EXEC msdb.dbo.sp_add_alert
    @name = 'Error_Critico_16',
    @severity = 16,
    @message_id = 0,
    @delay_between_responses = 300;
```

```
100 % ▾ Messages
SQLServerAgent is not currently running so it cannot be notified of this action.

Completion time: 2025-12-03T19:00:06.2231585-05:00
```

Enviar alertas cuando la base de datos crece demasiado

```
IF (SELECT size*8/1024 FROM sys.database_files WHERE name='MiBaseDatos') > 5000
BEGIN
    EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'PerfilNotificaciones',
        @recipients = 'soporte@midominio.com',
        @subject = 'ALERTA: Crecimiento inusual de BD',
        @body = 'La base de datos ha superado 5 GB.';
END
```

```
100 % ▾ Messages
Commands completed successfully.

Completion time: 2025-12-03T19:01:18.0483202-05:00
```

## **Justificación Técnica**

El monitoreo proactivo es crucial en la administración de bases de datos porque:

1. Detecta fallos antes que ocurran

Permite identificar:

caídas de servicios

errores de discos

saturación de CPU

crecimiento anormal de archivos

fallos en backups

Evita tiempos muertos y pérdida de datos.

---

2. Reduce tiempos de respuesta

Los operadores reciben alertas por correo inmediatamente.

Permite actuar antes de que el usuario note el problema.

---

3. Garantiza la integridad y disponibilidad

SQL Server funciona en un entorno crítico:

ERP, facturación, ventas, producción, servidores web.

El monitoreo evita:

corrupción de datos

reprocesos lentos

interrupciones de servicio

---

4. Permite auditoría y análisis histórico

Los logs permiten ver:

cuándo el sistema falló

qué error ocurrió

qué usuario ejecutó un proceso

Información clave para auditorías.

---

5. Permite cumplir normativas y estándares

ISO, PCI, leyes de protección de datos exigen monitoreo permanente.

---

## **★ Buenas Prácticas de Monitoreo Proactivo en SQL Server**

1. Activar Database Mail y configurar alertas críticas

Obligatorio para ambientes productivos.

---

2. Registrar logs en tablas propias

Permite análisis histórico incluso si SQL Agent falla.

