

UNIVERSIDAD PERUANA LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y
COMPUTACIÓN



ASIGNATURA: Base de Datos II

ESTUDIANTE: Espejo Quispe Luis Enrique

DOCENTE: Mg. Raúl Fernandez Bejarano

CICLO: V

SECCIÓN: A1

HYO-2025

SQL Server almacena datos en archivos físicos:

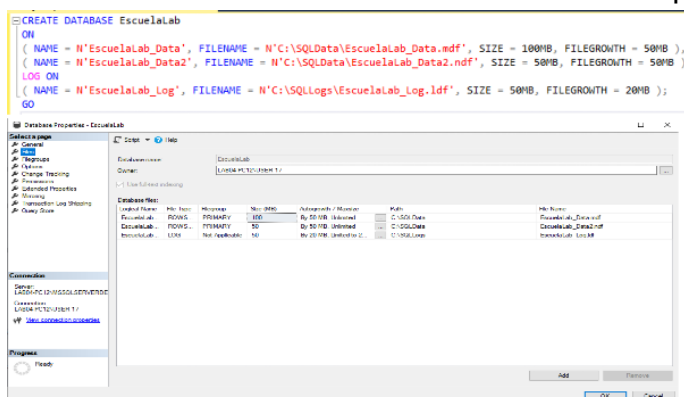
- Archivos de datos primarios (.mdf)
- Archivos de datos secundarios (.ndf)
- Registro de transacciones (.ldf). El motor gestiona páginas (8 KB) y extensiones (64 KB) para organización física.

Diseño de archivos influye en rendimiento: separar archivos de datos y logs en discos distintos, usar múltiples archivos de datos para cargas altamente concurrentes, y ajustar autogrowth en tamaños razonables (no en % en sistemas OLTP).

Mantenimiento: operaciones de shrink solo cuando es estrictamente necesario; verificar y mantener estadísticas e índices; planificar backups coherentes con el modelo de recuperación.

EJEMPLO:

Crearemos una base de datos con archivos separados



Propiedades clave de una base de datos: tamaño inicial, crecimiento automático (autogrowth), collation, modelo de recuperación, opciones de compatibilidad (compatibility level), opciones de auto shrink/auto close, y ubicación de archivos.

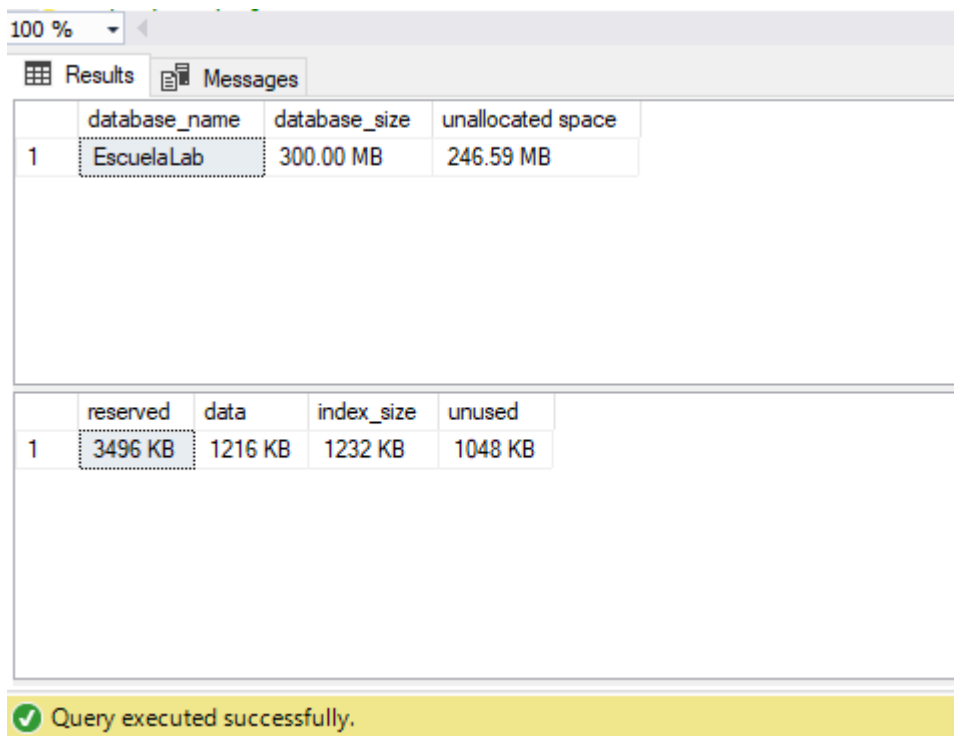
EJEMPLO:

Modificaremos las propiedades y luego comprobaremos el espacio de uso

```
ALTER DATABASE EscuelaLab
MODIFY FILE ( NAME = 'EscuelaLab_Data', SIZE = 200MB, FILEGROWTH = 100MB );
GO

USE EscuelaLab;
EXEC sp_spaceused;
GO
```


EJECUTANDO EL CODIGO:



The screenshot shows the SQL Server Enterprise Manager interface. At the top, there's a zoom level of 100%. Below it, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following data:

	database_name	database_size	unallocated space
1	EscuelaLab	300.00 MB	246.59 MB

	reserved	data	index_size	unused
1	3496 KB	1216 KB	1232 KB	1048 KB

At the bottom of the window, a yellow status bar indicates:  Query executed successfully.

Existen 3 tipos de recuperación o Backups:

1. Simple: no preserva el log para recuperaciones punto-en-tiempo; el log se trunca automáticamente. Adecuado para datos regenerables o entornos de desarrollo.
2. Full: registra todas las transacciones; permite recuperación punto-en-tiempo con backups de log; requerido para RPO bajos. Necesita backups regulares de log o crecerá el .ldf.
3. Bulk-Logged: modo intermedio que minimiza tamaño del log durante operaciones bulk (bcp, BULK INSERT, SELECT INTO), sacrificando la capacidad de recuperación punto-en-tiempo para sesiones con operaciones masivas; útil para cargas masivas controladas.

EJEMPLO:

```
--ejemplo 3
--backup actual
SELECT name, recovery_model_desc FROM sys.databases WHERE name = 'EscuelaLab';
GO

-- backup full
ALTER DATABASE EscuelaLab SET RECOVERY FULL;
GO
BACKUP DATABASE EscuelaLab TO DISK = N'C:\Backups\EscuelaLab_FULL.bak' WITH INIT;
GO

--backup de log
BACKUP LOG EscuelaLab TO DISK = N'C:\Backups\EscuelaLab_Log.trn' WITH INIT;
GO

--backup simple
ALTER DATABASE EscuelaLab SET RECOVERY SIMPLE;
GO
```

RESULTADOS:

100 %		
Results Messages		
	name	recovery_model_desc
1	EscuelaLab	FULL

La seguridad en SQL Server se compone de logins a nivel de instancia y usuarios a nivel de base de datos. Los logins pueden ser SQL Server o Windows (AD).

Roles fijos del servidor (sysadmin, serveradmin, etc.) y roles fijos de base de datos (db_owner, db_datareader, db_datawriter, db_ddladmin) facilitan delegación. También existen roles definidos por el usuario y permisos granulares (SCHEMA, OBJECT, COLUMN, EXECUTE).

EJEMPLO

```
-- EJEMPLO 4
-- Crear login a nivel de instancia (SQL auth, ejemplo de laboratorio)
CREATE LOGIN app_escuela WITH PASSWORD = 'P@ssw0rdSegura!';
GO

-- Crear usuario en la base de datos y asignarlo a rol personalizado
USE EscuelaLab;
CREATE USER usr_app_escuela FOR LOGIN app_escuela;
GO
```

RESULTADO DE LOS USUARIOS CREADOS

```
-- EJEMPLO 4
-- Crear login a nivel de instancia (SQL auth, ejemplo de laboratorio)
CREATE LOGIN app_escuela WITH PASSWORD = 'P@ssw0rdSegura!';
GO

-- Crear usuario en la base de datos y asignarlo a rol personalizado
USE EscuelaLab;
CREATE USER usr_app_escuela FOR LOGIN app_escuela;
GO
```

Principio de menor privilegio: asignar sólo los permisos necesarios. Usar roles de aplicación o grupos AD para administrar conjuntos de permisos. Emplear esquemas para organización y aplicar permisos a nivel de esquema cuando sea apropiado.

Políticas de acceso: en entornos corporativos, integrar con Active Directory, habilitar autenticación y cifrado (TLS), usar Always Encrypted o Dynamic Data Masking para datos sensibles, y auditar accesos.

EJEMPLO

Para el ejemplo seguiremos usando los datos anteriores

```
-- Asignar permisos mínimos: sólo insertar/select en esquema dbo.Alumnos
CREATE ROLE rol_app_lectura_insercion;
ALTER ROLE rol_app_lectura_insercion ADD MEMBER usr_app_escuela;
GO

GRANT SELECT, INSERT ON SCHEMA::dbo TO rol_app_lectura_insercion;
GO

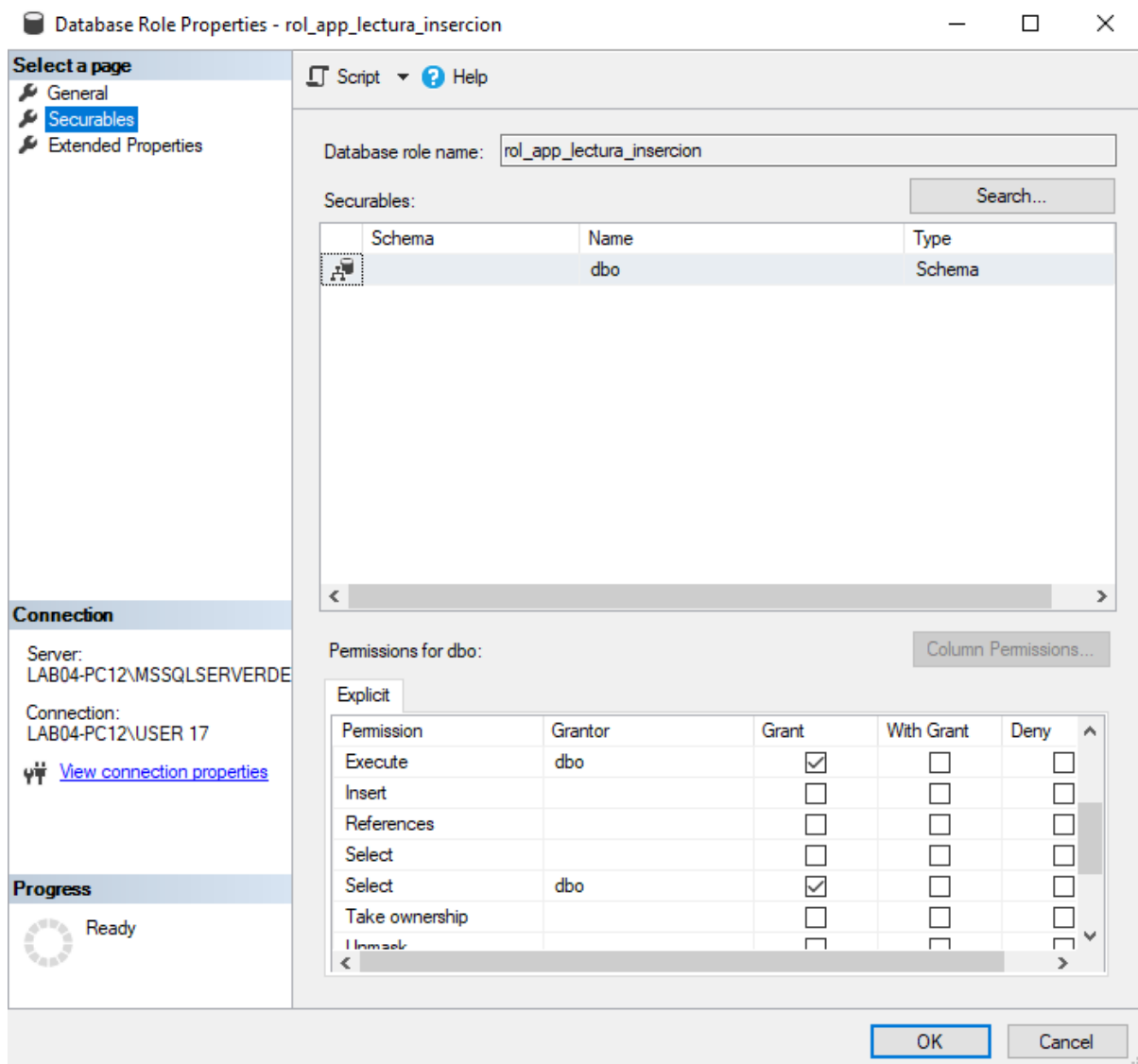
--conceder permiso execute
GRANT EXECUTE ON SCHEMA::dbo TO rol_app_lectura_insercion;
GO

--quitar permiso
REVOKE INSERT ON SCHEMA::dbo FROM rol_app_lectura_insercion;
GO

-- ver permisos de un usuario
SELECT * FROM fn_my_permissions(NULL, 'DATABASE');
-- 0 revisar vistas de catalogo: sys.database_permissions, sys.database_principals
GO
```

RESULTADO DE LA ASIGNACION DE PERMISOS

RESULTADO DE REVOCAR LOS PERMISOS



Activity Monitor en SSMS ofrece una vista rápida de la actividad: procesos activos, CPU, I/O, bloqueos, consultas más costosas y espera de recursos. Es útil para diagnóstico inicial y para identificar sesiones bloqueadas o queries de larga duración.

Limitaciones: Activity Monitor es una herramienta GUI de diagnóstico puntual; para monitorización continua y retención histórica conviene usar Query Store, Extended Events, Performance Monitor (PerfMon) o soluciones APM/Monitoring (Azure Monitor, SCOM, etc.).

Métricas clave: CPU, Page Life Expectancy, waits (LCK_M_X, PAGEIOLATCH_XX), Batch Requests/sec, Log Flush Waits, y latencia de I/O.

EJEMPLOS:

RESULTADO

```

--ejemplo 6

-- Ver detalles de sesión
DBCC INPUTBUFFER(<SPID>);
SELECT * FROM sys.dm_exec_sessions WHERE session_id = <SPID>;
SELECT * FROM sys.dm_exec_requests WHERE session_id = <SPID>;
GO

-- usar query store
-- Habilitar Query Store en la base de datos (si no está)
ALTER DATABASE EscuelaLab SET QUERY_STORE = ON;
GO

-- Consultar top 10 queries por tiempo total
SELECT TOP 10 qsqt.query_id, qsqt.query_sql_text, qsp.avg_duration
FROM sys.query_store_query_text qsqt
JOIN sys.query_store_query qsq ON qsqt.query_text_id = qsq.query_text_id
JOIN sys.query_store_plan qsp ON qsq.query_id = qsp.query_id
ORDER BY qsp.avg_duration DESC;
GO

```

SQL Server Agent es el componente para automatización: jobs (con pasos que ejecutan T-SQL, SSIS, PowerShell), schedules, alerts y operators. Permite programar backups, mantenimiento de índices, cargas ETL y notificaciones.

Estructura: Job → Steps (cada uno con su propio tipo y subsistema) → Schedules → Alerts → Notifications. Jobs pueden ejecutarse bajo cuentas con permisos específicos (proxy) para limitar privilegios.

Buenas prácticas: separar jobs de mantenimiento de jobs de negocio; registrar salida (logging) y alertas; evitar jobs largos en horarios pico; usar retención de historial para no llenar msdb.

EJEMPLO:


```
--ejemplo 7
USE msdb;
GO
EXEC sp_add_job @job_name = N'Backup_EscuelaLab_Diario';
GO

EXEC sp_add_jobstep
    @job_name = N'Backup_EscuelaLab_Diario',
    @step_name = N'Backup Completo',
    @subsystem = N'TSQL',
    @command = N'BACKUP DATABASE EscuelaLab TO DISK = N'C:\Backups\EscuelaLab_FULL.bak' WITH INIT, COMPRESSION;',
    @retry_attempts = 1,
    @retry_interval = 5;
GO

EXEC sp_add_schedule
    @schedule_name = N'Diario_02AM',
    @freq_type = 4, -- diario
    @freq_interval = 1,
    @active_start_time = 20000; -- 02:00:00
GO

EXEC sp_attach_schedule
    @job_name = N'Backup_EscuelaLab_Diario',
    @schedule_name = N'Diario_02AM';
GO

EXEC sp_add_jobserver @job_name = N'Backup_EscuelaLab_Diario';
GO
```
