ADMINISTRAÇÃO SQL SERVER

**Pontos a serem considerados para a instalação:**

- Quantidade em memória;

- Tamanho em disco.

Requisitos mínimos:

- 64 BITS;

- Windows Server 2012 R2;

- Framework 3.5 SP1.

**Instalação deve ocorrer em uma ou mais instâncias (ambientes) do servidor.**

Módulos de instalação:

- Database DE (“motor”)

- Analysis Services (aplicações gerenciais)

- Report Services (relatórios)

- Agent (schedule, processos em background)

- Integration Services (transferência de dados)

- Data Quality Services (limpeza de dados)

- Machine Learning Services (análise estatísticas “R”)

**O Banco de dados possui 2 ou 3 arquivos físicos:**

- Dados

- Primários (MDF)

- Secundários (NDF): não obrigatório

- Transações (LOG)

- Transações (LDF)

**Reduzindo arquivos de dados**

USE NOME\_BASE

ALTER DATABASE NOME\_BASE SET RECOVERY SIMPLE (inativa a base de dados p/ outros usuários)

DBCC SHRINKDATABASE (‘NOME\_BASE’, NOTRUNCATE) (comando p/ reduzir arquivos, etapa 1)

DBCC SHRINKDATABASE (‘NOME\_BASE’, TRUNCATEONLY) (comando p/ reduzir arquivos, etapa 2)

ALTER DATABASE NOME\_BASE SET RECOVERY FULL (ativa a base de dados p/ outros usuários)

**BACKUPs:** consiste em fazer cópias de base de dados ou arquivos de log para posterior recuperação ou migrar entre ambientes.

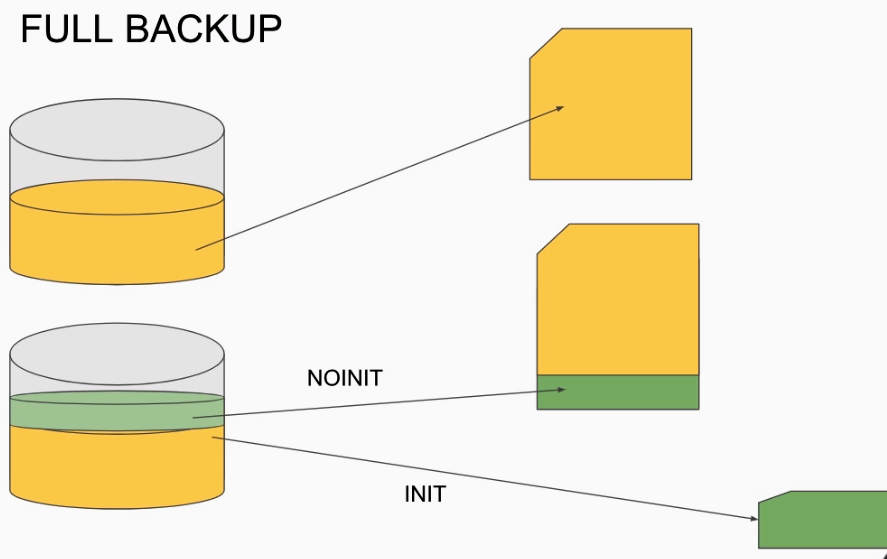
3 tipos:

- **Full:** backup completo (dados e arquivos log)

BACKUP DATABASE NOME\_BASE TO DISK = ‘DIRETÓRIO E NOME’

Cláusula INIT: usada para o primeiro BACKUP

Cláusula NOINIT: usada para adicionar dados num backup já existente



- **Diferencia**l: mapeia toda mudança efetuada desde o último backup (backup full das diferenças)

BACKUP DATABASE NOME\_BASE TO DISK = ‘DIRETÓRIO E NOME’ WHIT DIFERENCIAL

- **Transact:** representa o armazenamento apenas das modificações de dados

BACKUP LOG NOME\_BASE TO DISK = ‘DIRETÓRIO E NOME’

**Recuperando o BACKUP**

**- Status do BACKUP**:

RESTORE VERIFYONLY FROM DISK ‘NOME\_BACKUP’ WITH CHECKSUM

Se retornar 1: OK

Se retornar 0: não OK

**- Verificação do estado dos BACKUPs:**

RESTORE HEADERONLY FROM DISK = ‘DIRETÓRIO E NOME’

Retornos importantes:

- Backup Type: (1 – inicial; 2 – log; 3 – diferencial)

- Position: ordem do backup

- BackupStartDate: data inicial

- BackupFinishDate: data final

**- Recuperação do BACKUP**

RESTORE DATABASE ‘NOME DO BANCO’ FROM DISK ‘DIRETÓRIO E NOME’ WITH FILE = NUMERO BKP, RECOVERY/NORECOVERY

- RECOVERY: quando não vai mais ler nenhum arquivo de backup

- NORECOVERY: quando vai ler mais arquivos backups

**- BACKUPs de LOG:** são melhores pra efetuar durante as rotinas, salvam as transações e alterações na base de dados.

\*\*\*\*\* Geralmente são feitos BACKUPs FULL e de LOG de modo estratégico. EX: 1 BACKUP FULL por dia e 5 BACKUP LOG\*\*\*\*

**ÍNDICE (B-TREE):**cria um índice que melhora o desempenho das “procuras” (árvore binária/balaced tree )

CREATE NONCLUSTERED INDEX NOME\_INDEX ON TABELA (VARIÁVEL)

**PLANO DE CONSULTA:** prévia do que o SQL vai fazer ao rodar uma consulta TSQL (caminho que vai percorrer)

- SCAM : varredura

- SEEK: busca

**SUGESTÃO DE ÍNDICE:** o SELECT abaixo sugere índices para as execuções.

SELECT sys.objects.name

, (avg\_total\_user\_cost \* avg\_user\_impact) \* (user\_seeks + user\_scans) AS Impact

, 'CREATE NONCLUSTERED INDEX ix\_IndexName ON ' + sys.objects.name COLLATE DATABASE\_DEFAULT + ' ( ' + IsNull(mid.equality\_columns, '') + CASE WHEN mid.inequality\_columns IS NULL

THEN ''

ELSE CASE WHEN mid.equality\_columns IS NULL

THEN ''

ELSE ',' END + mid.inequality\_columns END + ' ) ' + CASE WHEN mid.included\_columns IS NULL

THEN ''

ELSE 'INCLUDE (' + mid.included\_columns + ')' END + ';' AS CreateIndexStatement

, mid.equality\_columns

, mid.inequality\_columns

, mid.included\_columns

FROM sys.dm\_db\_missing\_index\_group\_stats AS migs

INNER JOIN sys.dm\_db\_missing\_index\_groups AS mig ON migs.group\_handle = mig.index\_group\_handle

INNER JOIN sys.dm\_db\_missing\_index\_details AS mid ON mig.index\_handle = mid.index\_handle AND mid.database\_id = DB\_ID()

INNER JOIN sys.objects WITH (nolock) ON mid.OBJECT\_ID = sys.objects.OBJECT\_ID

WHERE (migs.group\_handle IN

(

SELECT TOP (500) group\_handle

FROM sys.dm\_db\_missing\_index\_group\_stats WITH (nolock)

ORDER BY (avg\_total\_user\_cost \* avg\_user\_impact) \* (user\_seeks + user\_scans) DESC))

AND OBJECTPROPERTY(sys.objects.OBJECT\_ID, 'isusertable')=1

ORDER BY 2 DESC , 3 DESC

**SUGESTÃO DE REMOÇÃO DE ÍNDICE**

SELECT o.name, indexname=i.name, i.index\_id

, reads=user\_seeks + user\_scans + user\_lookups

, writes = user\_updates

, rows = (SELECT SUM(p.rows) FROM sys.partitions p WHERE p.index\_id = s.index\_id AND s.object\_id = p.object\_id)

, CASE

WHEN s.user\_updates < 1 THEN 100

ELSE 1.00 \* (s.user\_seeks + s.user\_scans + s.user\_lookups) / s.user\_updates

END AS reads\_per\_write

, 'DROP INDEX ' + QUOTENAME(i.name)

+ ' ON ' + QUOTENAME(c.name) + '.' + QUOTENAME(OBJECT\_NAME(s.object\_id)) as 'drop statement'

FROM sys.dm\_db\_index\_usage\_stats s

INNER JOIN sys.indexes i ON i.index\_id = s.index\_id AND s.object\_id = i.object\_id

INNER JOIN sys.objects o on s.object\_id = o.object\_id

INNER JOIN sys.schemas c on o.schema\_id = c.schema\_id

WHERE OBJECTPROPERTY(s.object\_id,'IsUserTable') = 1

AND s.database\_id = DB\_ID()

AND i.type\_desc = 'nonclustered'

AND i.is\_primary\_key = 0

AND i.is\_unique\_constraint = 0

AND (SELECT SUM(p.rows) FROM sys.partitions p WHERE p.index\_id = s.index\_id AND s.object\_id = p.object\_id) > 10000

ORDER BY reads