SQL05 – SQL Server: miti da sfatare

Andrea Benedetti

Architect | SQL Server & BI Regional Lead

Microsoft

andrea.benedetti@microsoft.com

@anBenedetti

https://blogs.technet.com/b/andrea_benedetti_blog

Grazie a

Sponsor

SELECT

- Scrivere la stessa SELECT in maniera diversa è lo stesso

Dove andiamo

- Conosciamo così a fondo il db che utilizziamo?
- Non tutto è come sembra… provare per credere…
- In generale NON prendere mai per oro colato

Partiamo…

Quanti DEV ci sono in sala?

User Function

- Usare le user function alla «Visual Basic» va bene

- Un db lavora in maniera diversa
- Una funzione scalare lavora riga per riga
  - Un database no
- In generale se vedo > 10 UDF mi spavento ☺

Transazioni read committed

- Non potrò mai leggere due volte una riga

- Capiamo sempre cosa stiamo facendo
- Poniamoci sempre nel caso «peggiore»

- Il db fa quello che gli chiediamo noi… ☺

Qualche ragionamento a livello sistemistico

DBCC Checkdb

- È bloccante

- Non causa blocchi al database
  - Utilizza database snapshot da SQL 2005 +
- È uno dei nostri migliori amici.
- Usatelo!

- Blocking-free da SQL Server 2000 ☺

Database backup

- Un backup può causare blocchi.

- I backup non acquisiscono lock
- Le performance db possono risentire di I/O contention

Tempdb

- Il tempdb deve avere un data file per core

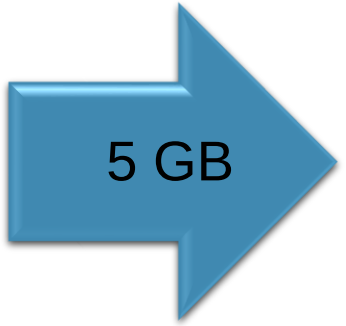
- Esiste un solo tempdb → può essere un collo di bottiglia
- Raccomandazione «vecchia», quando non c'erano tanti core

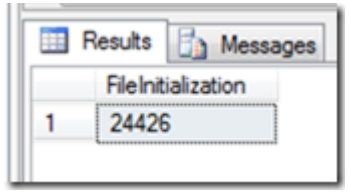
- *PAGEIOLATCH_XX* → I/O contention
- Attenzione:
  - cost of file switching
  - requires more IAM pages
  - increases the manageability overhead

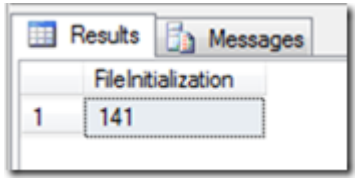
Instant file inizialization

- Solo su Enterprise.

- Sempre disponibile
- Funziona solo su file dati
- Il Tlog è sempre in zero-out per integrità







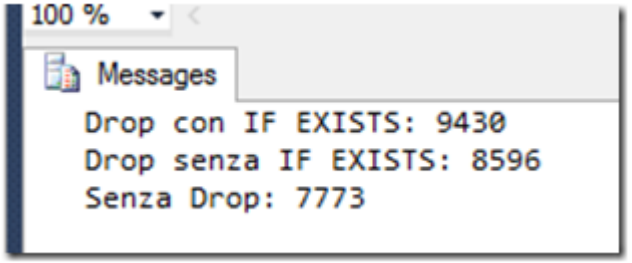
http://community.ugiss.org/blogs/abenedetti/archive/2012/02/22/sql-server-instant-initialization.aspx

Qualche ragionamento a livello sviluppo

Drop

- Il DROP esplicito o implicito di una tabella temporanea non fa differenza

- Nel tuning sono molti i dettagli che possono portare a fare la differenza
- Se in una SP utilizzo una #temp ne chiamo la DROP o no?



Tipi di dato

- Nvarchar(4000) o Nvarchar(35) è uguale…

(msec)								
1 MLN Rows	INSERT	ADD COL	REBUILD	SELECT	UPDATE	LIKE	EQUALS	DELETE
Nvarchar(MAX)	2176	15006	1106	313	1373	200	200	1103
Nvarchar(4000)	863	13723	790	173	1046	100	86	1076
Nvarchar(35)	700	12376	473	133	646	53	36	906
Varchar(35)	526	12120	423	116	596	53	36	876
2 MLN Rows	INSERT	ADD COL	REBUILD	SELECT	UPDATE	LIKE	EQUALS	DELETE
Nvarchar(MAX)	4386	31066	2076	526	2650	363	316	2280
Nvarchar(4000)	2183	27433	1636	373	2060	190	170	3310
Nvarchar(35)	1220	23333	950	253	1246	103	76	2963
Varchar(35)	1186	22530	843	243	1216	101	74	2200

Dati e cluster

- I dati sono memorizzati separatamente dal cluster.

- I dati sono nell'indice cluster
- In altri db ok, in SQL Server no

Chiave cluster

- Gli indici noncluster non hanno la chiave cluster.

- Il livello foglia NC:
  - NON ha un puntamento alla riga
  - Ha la chiave cluster

Rebuild Indici Cluster

- Ricostruire un indice cluster ricostruisce anche tutti i noncluster.

- Gli indici noncluster hanno un puntamento
- Il puntamento non cambia alla rebuild

- Rebuild cluster = nessun effetto su NC

Count(*)

- SELECT COUNT(*) effettua sempre un table scan.

- Query processor sceglie la strada più corta
  - Scan dell'indice nonclustered più piccolo
    - Purchè sia non filtrato

Variabili tabella

- Sono gestite solo in memoria.

- Tabelle temporanee e variabili tabella sono sempre create sul tempdb e non in memoria
- Occhio
  - Optimizer assume che ci sia una sola riga
  - Non hanno statistiche
  - Non sono indicizzabili
    - Solo indici cluster e vincoli UNIQUE
  - Non sono usabili parallelamente

GUID

- È una buona cluster key.

- GUIDs è randomico → frammentazione
  - NEWSEQUENTIALID non è randomico
- 16 bytes
- La scelta migliore? INT, BIGINT
  - 4, 8 bytes
  - Append-only insertion pattern

Ordinamento (1)

- Le tabelle hanno un ordine garantito.

- Per definizione, le tabelle NON hanno un ordine
- Nemmeno in presenza di indici cluster

- Ci sono diversi operatori (execution plan) che possono influenzare l'ordinamento
  - Stream aggregates, partitions, parallelism, …
- Usiamo ORDER BY

Ordinamento (2)

- Un ORDER BY non impatta le performance.

- ORDER BY consuma preziose risorse server
- Viene utilizzato processor time
- Se serve, viene utilizzato TEMPDB

NOLOCK (1)

- NOLOCK funziona anche su istruzioni DML.

- Non è la panacea di tutti i mali ☺
- Funziona solo con SELECT
- Una scrittura richiede, quanto meno, un lock Sch-S (Schema Stability)
- Quindi SQL Server lo ignora…

NOLOCK (2)

- NOLOCK si propaga anche sulle colonne calcolate.

- Se la tabella contiene una colonna calcolata, NOLOCK non ha effetti

In sintesi

- Molto si legge, molto è sbagliato
- Occhio a prendere per «oro colato»
- Provare per credere funziona sempre
- Mettersi nella condizione «peggiore»

- Test, test, test ☺

Grazie!

Q&A

Tutto il materiale di questa sessione su

http://www.communitydays.it/

Lascia il feedback su questa sessione,

potrai essere estratto per i nostri premi!

Seguici su

Twitter @CommunityDaysIT

Facebook http://facebook.com/cdaysit

#CDays14