MsSQL: Basal performance tuning, part 2

Grundlæggende brug af indeks for bedre performance.

Skrevet den **10. Feb 2009** af **trer** I kategorien **Databaser / MS SQL** | ★★☆☆☆

Disse artikler om basal performance tuning er uddrag af en intern anbefaling om SQL som jeg har skrevet til udviklerne på min arbejdsplads, NNIT a/s.

Korrekt brug af indeks er nødvendig for at opnå god performance i en database applikation, men indeks er et tve-ægget sværd - på den ene side øger de hastigheden når data skal ud af basen, og på den anden side sænker de hastigheden når data skal ind i.

Man kan derfor ikke ukritisk smide indeks på ens database, men må overveje hvor de kan gøre gavn.

Det er i øvrigt værd at bemærke, at SQL Server kan traversere et indeks lige godt forfra og bagfra. Der er derfor ikke noget vundet ved at angive et indeks som stigende eller faldende.

Undtagelsen er naturligvis sammensatte indeks med skiftende orden i de enkelte kolonner.

Benyt "Index Tuning Wizard"
Det første og nemmeste er, at man afvikler Query Analyzers indbyggede værktøj, Index Tuning Wizard, med de forskellige queries man skal køre mod databasen.

Det bedste er her, at have en profilertrace der giver et typisk billede af databasen under brug så ITW har denne at gå ud fra. ITW vil så afvikle tracen med forskellige index-sammensætninger, og efter det fortæller den, indeks der bør tilføjes og hvilke der kan fjernes. ITW kan endda simulere indeks brug ved større data-mængder end der reelt findes i databasen.

Bemærk at såfremt der i større omfang benyttes views kan Index Tuning Wizard give et ret fejlagtigt billede da den ikke opløser views i de underliggende SELECTs.

Benyt ikke Query og Index hints
Undgå at benytte Query/Indeks Hints i produktionssystemer. Korrekt skrevne queries / stored procedures vil vælge det bedste index og join-type udfra de aktu-elle indeks-statistikker og server belastning.

Tvinger man Query Optimizeren til at benytte et bestemt indeks eller join-type vil denne ikke længere kunne vælge den optimale tilgang til data.

I forbindelse med test og udvikling kan det dog være fordelagtigt at benytte in-deks hints til at checke effekten af forskellige index og join typer.

Benyttes Query / Indeks-hints vil man være tvungen til at gen-teste ved hver server og/eller database-opdatering idet selv små ændringer kan påvirke effekten af hints.

Fjern ubrugte indeks
Alle indeks sløver performance på en tabel ved indsættelse og opdatering af data. Såfremt et indeks generelt ikke benyttes af nogen query giver det derfor bedst mening af fjerne det.

Via Query Analyzers Index Tuning Wizard og et profilertrace kan man identificere indeks som muligvis ikke

benyttes.

Fjern indeks på små tabeller.
For at benytte en tabel skal der altid læses mindst 1 extent og for at benytte et indeks skal der også altid læses mindst 1 extent. Optimizeren vælger derfor ofte ikke at benytte indeks på tabeller der ligger på 1 til 2 extents i størrelse (ca. 128 KB).

Undlad derfor indeks på små tabeller (men erklær stadig PRIMARY KEY constraints på tabellen af hensyn til mulighed for fremmednøgler og unikke række-identifikation).

Størrelsen på en given tabel kan altid beregnes ud fra forventet antal rækker * række-definitionen. I en database i brug kan størrelsen ses direkte i KB via Enterprise Manager.

Begræns brug af Clustered Index
På en tabel i SQL Server kan der være ét og kun ét clustered index. Typisk vil SQL Server vælge at lægge det på primær nøglen, men der er det sjældent til gavn.

Et clustered index styrer den fysiske orden hvori data lagres - derfor må det ikke bruges på data der opdateres (f.eks. en LastUpdated kolonne). Det skal istedet bruges på kolonner der kun én gang tildeles en værdi (f.eks. en CreatedDate kolonne).

Lægges et clustered index på en kolonne der opdateres skal SQL Server reorganisere data ved hver opdatering, dette er ekstremt tidskrævende sammenlignet med en normal indsættelse.

Man bør forbeholde clustered index til kolonner hvorpå der foretages RANGE-scan, dvs. typisk en BETWEEN operator, eller hvor man ønsker en implicit ORDER BY.

Sammensatte indeks
I visse tilfælde vil det være fordelagtigt at benytte sammensatte indeks (indeks der indeholder flere kolonner) - mens de i andre situationer vil være ufleksible, i forhold til optimeringsplanen.

Brug sammensatte indeks når flere kolonner alle benyttes i betingelser og vil understøttes af indeks. Sammensatte indeks vil ofte fylde mindre og kan gøre at læsning af data extents er unødvendig når indekset dækker de kolonner der indgår i querien.

Query Optimizeren kan kun benytte et sammensat indeks i kolonne-orden - oprettelse af separate indeks giver ofte mere fleksibilitet. Det betyder at et indeks der dækker kolonne A,B og C ikke kan anvendes hvis en WHERE betingelse kun benytter kolonne B.

I SQL Server 7.0 og 2000 kan Query Optimizeren anvende flere indeks i udarbejdelsen af queryplanerne - dermed kan flere enkeltkolonne indeks indgå og sammensatte indeks er knap så nødvendige som på f.eks. Oracle.

Bemærk at sammensatte indeks altid skal opbygges så den kolonne med største spredning af data står først i indekset. Det er kun denne kolonne der benyttes til at afgøre om indeks vil blive brugt eller ej.

Understøt joins med indeks
Alle kolonner der indgår i en JOIN eller WHERE betingelse bør være understøttet af indeks. Vurder om det bør være enkeltkolonne indeks eller sammensatte indeks.

Overvej værdien af indeks.
Indeks er ofte mest anvendelige såfremt der er en rimelig spredning i data. Har man fx. 1.000.000 rækker i en person tabel vil et indeks på personens køn være af begrænset effekt (*).

Spredningen i data på en kolonne kan vises i procent ved nedenstående SQL:

```
SELECT CAST(COUNT(DISTINCT [col1]) AS FLOAT) /
CAST(COUNT(*) AS FLOAT)*100) AS [Spredning]
FROM [dbo].[mytable]
```

Hvor [col1] er den kolonne man ønsker at placere indeks på. En grov tommelfin-ger regel siger at spredningen bør være 70% eller højere før indeks er godt.

På historik tabeller og lignende hvorfra data kun sjældent læses men ofte indsæt-tes bør man undlade indeks - evt. kombineret med stored procedures til at opret-te / nedlægge indeks før rapport-udtræk

*) En sandhed med modifikationer. Hvis man f.eks. kun ønsker at få samtlige mænd ud, vil indekset med ét opslag være i stand til at skære samtlige kvinder fra og vice-versa, og så er indekset rart.

Understøt sorteringer med indeks
Alle kolonner der indgår i implicitte eller eksplicitte sorteringer bør understøttes af indeks.

Implicit er fx. DISTINCT og UNION
Explicit er fx ORDER BY, GROUP BY

Det vil ofte være en fordel ved f.eks. DISTINCT at understøtte hele dataudtrækket med ét indeks.

God fornøjelse
Troels

Kommentar af driis d. 19. Dec 2004 | 1

God artikel om emnet.

Kommentar af skwat d. 17. Jun 2004 | 2

smukt og så til de point

Kommentar af lomse d. 05. Feb 2004 | 3

Jamen, for søren da osse, skriv dog lidt mere, om ´hvad det handler om. Man kan jo ikke se det ud fra din overskrift.

Kommentar af arne_v d. 18. Dec 2004 | 4

Meget godt beskrevet.

Kommentar af net-base.dk d. 03. Jun 2004 | 5

Sag lige og ledte efter hvordan kan trak alt data fra 2 tabeller på en gang, så prøvede lige her, men fandt ikke noget, men hvordan skulle jeg også vide hvad det var jeg ville få frem, tam overskrift. Men har ikke gidet at læse teksten