SQL Performance tips & trucs

I N H O U D S O P G A V E

SQL Performance tips & trucs (5-2-2010) 1
1. Zo weinig mogelijk OR gebruiken 2
1.1 De grootste tabel als laatste 2
1.2 Liever geen SELECT * 2
1.3 Zo weinig mogelijk DISTINCT gebruiken 2
1.4 Pas op met UNION 2
1.5 Snelheid van operatoren 3
1.6 Gebruik indien mogelijk BETWEEN 3
1.7 BETWEEN boven IN 3
1.8 Vermijd indien mogelijk de SUBSTRING functie 3
1.9 Vermijd de combinatie NOT IN 3
1.10 IN of EXISTS 3
2. User defined functions 4
2.1 Weeknummers 4
2.2 Controle sofinummers 4

1. Zo weinig mogelijk OR gebruiken

Waarom? In dat geval gebruikt SQL meestal geen index en dat scheelt natuurlijk snelheid. Er zijn twee alternatieven: IN en UNION.

Voorbeeld OR vervangen door IN:

Met OR:

```
SELECT *
FROM klanten
WHERE klantcode = 1
OR klantcode = 2
OR klantcode = 3
```

Met IN:

```
SELECT *
FROM klanten
WHERE klantcode IN (1,2,3)
```

Voorbeeld OR vervangen door UNION:

Met OR:

```
SELECT *
FROM klanten
WHERE klantcode = 1
OR klplaats = 'GRONINGEN'
```

Met UNION:

```
SELECT *
FROM klanten
WHERE klantcode = 1
UNION
SELECT *
FROM klanten
WHERE klplaats = 'GRONINGEN'
```

1.1 De grootste tabel als laatste

De volgorde van de tabellen bij FROM kan effect hebben op de snelheid van verwerking. Plaats voor de zekerheid de grootste tabel achteraan.

*1.2 Liever geen SELECT **

Vraag geen kolommen op die je niet nodig hebt. Bovendien brengt SELECT * met zich mee dat er geen indexen gebruikt worden wat ook voor een slechtere performance zorgt.

1.3 Zo weinig mogelijk DISTINCT gebruiken

Als er dubbele rijen verwijderd moeten worden, kan dat zeer tijdrovend zijn; de software moet dan elke rij met alle andere vergelijken. DISTINCT is zeker niet noodzakelijk in subqueries.

1.4 Pas op met UNION

Het UNION commando voert automatisch het equivalent van een SELECT DISTINCT uit, ook als er geen dubbele rijen zijn. Als je er zeker van bent dat er geen dubbele rijen zijn, kun je beter UNION ALL gebruiken.

1.5 Snelheid van operatoren

Bij WHERE bepalen de verschillende operatoren hoe snel een query wordt uitgevoerd. Vaak heb je geen keus, maar als je die wel hebt, geeft onderstaande volgorde van hoog naar laag de snelheid van uitvoering aan:

- =
- >, >=, <, <=
- LIKE
- <>

Conclusie: gebruik zo veel mogelijk = en zo weinig mogelijk <>.

1.6 Gebruik indien mogelijk BETWEEN

Aansluitend op het vorige geniet het de voorkeur BETWEEN te gebruiken boven een constructie met >= en <=. Bij gebruik van BETWEEN doet SQL namelijk wel een beroep op een eventuele index op de betreffende kolom.

1.7 BETWEEN boven IN

Als je keus hebt, gebruik dan BETWEEN en geen. Bijvoorbeeld:

```
SELECT customer_number, customer_name
FROM customer
WHERE customer_number in (1000, 1001, 1002, 1003, 1004)
```

Is veel minder efficiënt dan:

```
SELECT customer_number, customer_name
FROM customer
WHERE customer_number BETWEEN 1000 and 1004
```

Gesteld dat er een index zit op customer_number dan zal een reeks nummer veel sneller gelokaliseerd worden met BETWEEN dan met IN.

1.8 Vermijd indien mogelijk de SUBSTRING functie

De SUBSTRING functie kan een scan van de hele tabel tot gevolg hebben in plaats van de eventueel aanwezige index te gebruiken.
Dus in plaats van:

```
WHERE SUBSTRING(column_name,1,1) = 'b'
```

Probeer:

```
WHERE column_name LIKE 'b%'
```

1.9 Vermijd de combinatie NOT IN

Gebruik één van de volgende alternatieven:

- EXISTS of NOT EXISTS
- IN
- LEFT OUTER JOIN in combinatie met een NULL voorwaarde

1.10 IN of EXISTS

Als je de keus hebt tussen het gebruik van IN of EXISTS, EXISTS zal gewoonlijk sneller zijn.

2. User defined functions

2.1 Weeknummers

De volgende functies distilleert het weeknummer uit een datum volgens de Nederlandse ISO-norm:

```
CREATE FUNCTION fn_weeknummer
  (@d datetime )
RETURNS int
        AS
BEGIN
        DECLARE
                @jaar  int,
                @datum datetime,
                @weeknummer int
        SET @jaar = Year(@d)
        SET @datum = '1/1/' + str(@jaar)
        IF datepart(dw,@datum) > 1 and datepart(dw,@datum) <7
                SET @weeknummer = (datepart(wk,@d) )
        ELSE
                SET @weeknummer = (datepart(wk,@d) -1)
        IF @weeknummer = 0
                SET @weeknummer = datepart(wk,'12/31/' + str(@jaar-1)         )
        RETURN (@weeknummer)
END
```

2.2 Controle sofinummers

De volgende functie controleert sofinummers. Correct levert een 1 op; incorrect een 0.

```
CREATE FUNCTION fn_sofinummer
  (@sofi char(9) )
RETURNS int
        AS
BEGIN
        DECLARE
                @getal  int,
                @sofinummer int

                set @getal = 9 * substring(@sofi, 1, 1)
                set @getal = @getal + 8 * (substring(@sofi, 2, 1))
                set @getal = @getal + 7 * (substring(@sofi, 3, 1))
                set @getal = @getal + 6 * (substring(@sofi, 4, 1))
                set @getal = @getal + 5 * (substring(@sofi, 5, 1))
                set @getal = @getal + 4 * (substring(@sofi, 6, 1))
                set @getal = @getal + 3 * (substring(@sofi, 7, 1))
                set @getal = @getal + 2 * (substring(@sofi, 8, 1))
                set @getal = @getal + -1 * (substring(@sofi, 9, 1))

        IF @getal % 11 = 0
                SET @sofinummer = 1
        ELSE
                SET @sofinummer = 0
        RETURN (@sofinummer)
END
```