PARELTJE

Het zal geen enkele lezer van dit blad zijn ontgaan dat Microsoft binnenkort, op 7 november om precies te zijn, uitkomt met de nieuwe 2005 versie van SQL Server. Het feit dat Microsoft met een groot team van ontwikkelaars vijf jaar heeft gewerkt aan de ontwikkeling ervan betekent dat het product op veel vlakken is verbeterd en uitgebreid. Over de grote veranderingen in SQL Server is al veel gezegd en geschreven, maar bij zoveel vernieuwingen wil het wel eens gebeuren dat sommige pareltjes aan de aandacht ontsnappen. Over een van die pareltjes wil ik het nu hebben: *snapshot isolation*. Sommigen zullen beter bekend zijn met de gerelateerde term *row versioning*, daar kom ik later op terug.

In multi-user omgevingen waarin meer gebruikers/sessies gelijktijdig toegang hebben tot dezelfde database werkt SQL Server als een verkeerslicht dat ervoor zorgt dat er geen 'botsingen' tussen transacties plaatsvinden. Op deze manier kan SQL Server de data- integriteit waarborgen. Traditioneel werkt SQL Server met pessimistische locking wat inhoudt dat ongelukken worden voorkomen door sommige verkeersstromen (transacties) tijdelijk te laten wachten op andere verkeersstromen. In een typisch scenario is het zo dat een transactie die gegevens in de database wijzigt het licht op rood zet voor andere transacties die dezelfde gegevens willen uitlezen of wijzigen. Pas als de eerste transactie klaar is, komt voor andere transacties het licht weer op groen te staan. Het precieze locking-gedrag hangt af van de *transaction isolation level* dat wordt gebruikt.

Snapshot isolation werkt daarentegen met optimistische locking. Dit houdt in dat alle verkeerslichten altijd op groen staan, transacties hoeven dus nooit op andere transacties te wachten. Als er een botsing plaatsvindt tussen twee transacties waardoor de data-integriteit in gevaar komt, dan zorgt SQL Server ervoor dat één van de transacties tijdens de *commit* faalt en de wijzigingen van deze transactie worden teruggedraaid waardoor wijzigingen van de andere transactie niet ongedaan worden gemaakt of overschreven. Het is dus alsof het ongeluk nooit heeft plaatsgevonden! De applicatie kan op deze situatie anticiperen, bijvoorbeeld door de gefaalde transactie opnieuw aan SQL Server aan te bieden.

Het grote voordeel van *snapshot isolation* is een betere *concurrency*, wat in veel gevallen leidt tot een betere performance en schaalbaarheid van de applicatie. Dit voordeel geldt vooral in situaties waarbij er langlopende transacties of *report queries* zijn die vaak andere transacties ophouden. *Snapshot isolation* is echter niet altijd de beste oplossing: als blijkt dat er erg vaak botsingen tussen transacties plaatsvinden is het beter om de traditionele pessimistische locking te gebruiken. Ook duren *updates* langer wat een negatief effect op de prestaties kan hebben bij applicaties die veel data wijzigen.

Snapshot isolation is een optie die je in SQL Server expliciet per database moet aanzetten. Om de compatibiliteit te waarborgen met vorige versies gebruikt SQL Server 2005 namelijk standaard de traditionele pessimistische locking. Het aanpassen van een bestaande applicatie zodat deze werkt met *snapshot isolation* is trouwens meer werk dan het enkel omzetten van deze instelling. De applicatiecode kan namelijk impliciete aannames bevatten over de manier waarop de locking werkt. Bovendien moet je bij *snapshot isolation* ervoor zorgen dat foutmeldingen van SQL Server tijdens de *commit* van een transactie (die duiden op een botsing tussen transacties) intelligent door de applicatie worden afgehandeld.

Naast de genoemde voordelen van *snapshot isolation* is er nog een situatie waarin deze nieuwe functie van SQL Server 2005 zeer goed van pas komt: bij het migreren van Oracle databaseapplicaties naar SQL Server. Het alleen maar vertalen van PL/SQL (Oracle's SQL dialect) naar T-SQL voldoet in de praktijk meestal niet omdat de bovenliggende applicatiecode impliciete aannames maakt over het locking-gedrag van de database. Oracle heeft als enige grote databasefabrikant altijd optimistische locking ondersteund in plaats van pessimistische locking. Simpel gezegd: in de praktijk komt het erop neer dat een Oracle-applicatie na een vertaling van het SQL-dialect op SQL Server vaak *concurrency*-problemen heeft, waardoor ze slecht presteert. Om deze problemen op te lossen moest je in het verleden daarom de gehele applicatiecode doorspitten om alle impliciete aannames over het locking-gedrag van de onderliggende database op te sporen en de programmatuur daarop aanpassen.

Omdat SQL Server 2005 nu zowel pessimistische als optimistische locking ondersteunt, is het migreren van Oracle-applicaties naar SQL Server een stuk eenvoudiger geworden. Bovendien zijn er gereedschappen die veel werk uit handen nemen, zoals programma's die PL/SQL-code converteren in T-SQL (zie www.swissql.com en www.adventnet.com). Daarnaast biedt Microsoft ook een Database Migration Wizard die het omzetten van een database (metadata, stored procedures en data) van Oracle naar SQL Server grotendeels automatiseert.

Technische noot: de manier waarop SQL Server en Oracle optimistische locking implementeren is door middel van *row versioning*. Dit houdt in dat als een transactie data in de database wijzigt, er in feite een nieuwe versie van de datarij wordt aangemaakt waardoor andere transacties de oorspronkelijke waarde kunnen blijven benaderen. Bij het *commit*-ten van de transactie wordt vervolgens gedetecteerd of er een conflict heeft plaatsgevonden, en zo niet, dan wordt de versie van de data die bij de huidige transactie hoort 'permanent' gemaakt. Overigens verschillen SQL Server en Oracle op een aantal punten in de wijze waarop ze optimistische locking implementeren. Wie het naadje van de kous wil weten raad ik aan deze artikelen te lezen:
http://www.microsoft.com/technet/prodtechnol/sql/2005/SQL05B.mspx
http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=5

De conclusie van dit verhaal: of je nu een nieuwe applicatie wilt ontwikkelen, of dat je een bestaande Oracle-applicatie wilt migreren naar SQL Server, *snapshot isolation* is een pareltje in SQL Server 2005 dat je als applicatieontwikkelaar mogelijk veel werk uit handen neemt en dat de prestaties van je applicatie verbetert.

Andreas de Ruiter
Developer & Platform Group, Microsoft Nederland
aruiter@microsoft.com