Samenwerking tussen ‘storage-mensen’ en ‘database-mensen’ uitermate belangrijk

Oracle en opslag

Carel-Jan Engel

Bij beheerders van opslagsystemen is vaak weinig zicht op de specifieke eisen die moeten worden gesteld aan een voor de database optimaal functionerend opslagsysteem. Dit artikel geeft een globaal overzicht van de manier waarop Oracle omgaat met opslagmedia.

De eisen aan de opslag voor een database wordt in MG, GB of TB uitgedrukt, maar zelden in termen van IO/s, I/O-operaties per seconde. Opslagsystemen maken een sterke verandering door. SAN- en NAS-systemen dringen vanuit de high-end-systemen ook steeds meer door in de mid-range. Het is aardig om de kosten van zo’n centraal opslagsysteem in €/MB uit te drukken, maar als de aankoop daarop wordt gebaseerd blijkt regelmatig achteraf dat de vereiste performance van de database niet wordt gehaald.

Verschillende typen gegevens

Dit geldt nog meer als er wordt gewerkt met parity-gebaseerde RAID-systemen. Het lijkt dan goedkoop, maar de performance is vaak het slachtoffer. Goedkoop wordt dan duurkoop. Schijven die kleiner zijn dan 36 GB zijn nog maar nauwelijks te koop.
Een grootte rond de 72 GB is een redelijk veel voorkomende standaardmaat. Als een database administrator vervolgens voorstelt de eerste zes schijven voor de transactielog te gebruiken, met 6 bestanden van 250 MB per stuk, en deze zes schijven verder leeg te laten, wordt voorzichtig aan zijn verstandelijke vermogens getwijfeld. Geheel ten onrechte, overigens. Schijven worden steeds groter in opslagcapaciteit, maar de doorvoersnelheid neemt maar fractioneel toe. Deze twee eigenschappen komen dan ook steeds verder uit elkaar te liggen. Dat betekent dat er voor een goede inrichting van een Oracle database wel eens wat offers in capaciteit moeten worden gebracht, om de doorvoersnelheid op peil te houden.

Oracle gebruikt de opslag op verschillende manieren. Daarom volgt een korte inleiding over de verschillende typen gegevens die Oracle kent. Ruwweg laten die zich in zeven typen onderscheiden:
1. De data;
2. Tijdelijke gegevensopslag voor sorteeroperaties;
3. De transactielog;
4. De gearchiveerde transactielog;
5. De rollback- of undo-gegevens;
6. De configuratiegegevens over de database;
7. Het parameterbestand met de instellingen.

Uiteraard moet ook de Oracle software zelf nog ergens een plaatsje vinden. Dan praten we over een paar Gigabyte met programma’s, die hier verder buiten beschouwing gelaten worden. De programma’s worden bij het starten van de database één keer ingelezen, en het opslagmedium waarop de programmatuur staat is niet van invloed op de verwerkingscapaciteit van een database. De opslagmedia voor de database-gegevens hebben die invloed uiteraard wel. Eerst volgt uitleg over de verschillende soorten gegevensopslag die hierboven zijn vermeld.

De data.
De data, de gegevens, vormen het belangrijkste deel van de database. Omdat een Oracle database zichzelf definieert splits ik de data hier in twee delen. Allereerst is er de *data-dictionary*, waarin de interne structuren van de database worden vastgelegd. De data-dictionary beschrijft welke tabellen in de database bestaan, de opbouw van die tabellen, indexen, views en alle andere database-objecten die in een Oracle database kunnen worden gebruikt. Deze informatie is cruciaal voor het functioneren van de database. De data-dictionary bestaat zelf ook weer uit tabellen, indexen en views.
Naast de data-dictionary kennen we nog de gegevens van de gebruikers en van de applicaties zelf. Ook hier zijn weer tabellen, indexen en alle andere soorten objecten. Daarbij is het wel zo dat in de database ondergebrachte (applicatie)logica in de data-dictionary wordt opgeslagen. Bij deze applicatielogica is bijvoorbeeld te denken aan stored procedures, packages en functions.

Ook views horen daarbij. 'Normale' views zijn in Oracle niets anders dan vooraf gedefinieerde query's, die bij het raadplegen van de view worden uitgevoerd. De definitie van de view wordt opgeslagen in de data-dictionary. Views gedragen zich bij het raadplegen wel als een tabel, maar er zijn geen gegevens opgeslagen bij een normale view. Bij materialized views worden wel gegevens opgeslagen. Deze views worden bijvoorbeeld in datawarehouses gebruikt, om langlopende aggegratiefuncties vooraf uit te voeren, en de resultaten op te slaan. Een materialized view is op die manier weer verbonden aan een tabel, en gedraagt zich ook als zodanig.
Zowel de data-dictionary als de applicatiegegevens worden opgeslagen in zogenaamde *tablespaces*. Deze worden later in detail behandeld.

Tijdelijke gegevensopslag voor sorteeroperaties.
Op het moment dat gebruikers grote hoeveelheden joins uitvoeren, of resultaten willen sorteren, is het beschikbare interne geheugen van de server niet altijd voldoende. Op die momenten worden tussenresultaten vanuit het interne geheugen tijdelijk op schijf weggeschreven. Hiervoor kent Oracle aparte, zogenaamde *temporary tablespaces*.

De transactielog.
De transactielog of, in Oracle-jargon, *redo log*, is misschien nog wel belangrijker dan de data zelf. Transacties die worden afgerond, ge-commit, worden vastgelegd op schijf in de transactielog. De gewijzigde datablokken blijven meestal gewoon gewijzigd in de cache beschikbaar. Deze worden later, als er tijd voor is, of als de cache-ruimte voor iets anders nodig is, wel op schijf gezet. Daarmee is het belang van de transactielog verklaard: als data verloren gaan, maar de transactielog is bewaard, dan kunnen hiermee de data altijd worden gereconstrueerd. Een Oracle database heeft in principe drie of meer *online redo log files*. Dit zijn de transactielog-estanden, die cyclisch worden volgeschreven. Als het laatste bestand vol is, gaat het systeem weer verder met het eerste bestand. Omdat de transactielog zo belangrijk is wordt deze meestal dubbel uitgevoerd. Van ieder online redo log-bestand bestaan dan twee kopieën, waarmee het aantal redo log-estanden in principe op zes meer komt.

De gearchiveerde transactielog.
De transactielog wordt in ten opzichte van de database relatief kleine bestanden weggeschreven. Als een bestand vol is, zorgt een archiveringsproces ervoor dat het zojuist volgeschreven bestand naar een archief wordt geschreven. Met de online transactielog, de gearchiveerde transactielog en de laatste backup, kan een database na het verloren gaan van één of meer tablespaces weer worden hersteld tot en met de laatst gecomitte transactie.
Tijdens het archiveringsproces van een online redo log-bestand gaat het systeem verder met het volgende online redo log-bestand. Het is zaak dat het archiveringsproces klaar is vóórdat het systeem het te archiveren bestand opnieuw nodig heeft. Dat is ook de reden dat er minimaal drie online redo log-bestanden zijn. Zou het archiveren van een online redo log-bestand nog niet klaar zijn, dan blijft het systeem 'hangen' totdat het archiveren klaar is. Korte piekbelastingen (in aantallen transacties) kunnen opgevangen worden door meerdere online redo log-bestanden aan te maken. Het duurt dan langer voordat de cirkel rond is, en een bestand opnieuw aan de beurt is. Dit kan alleen goed werken als er ook snel weer rustiger momenten zijn, waarbinnen het archiveringsproces de achterstand weer kan inlopen.

De undo- of rollback-gegevens.
Transacties kunnen een lange looptijd hebben, waarbij veel gegevens als één geheel kunnen worden gewijzigd. Dat heeft tot gevolg dat gegevens uit de cache al naar schijf moeten worden geschreven, om plaats te maken voor andere gegevens, voordat de transactie is ge-commit. Als de transactie dan wordt teruggedraaid, moeten de al weggeschreven wijzigingen óók worden teruggedraaid. Om dit te kunnen doen wordt de oude versie van de gegevens vastgelegd. Dat heeft meteen het voordeel dat de zogenaamde *read-consistency* kan worden gewaarborgd. De oude versie blijft bewaard totdat er geen oudere transacties of query's meer actief zijn, die zijn gestart voordat het blok werd gewijzigd. Al die tijd kunnen deze transacties teruggrijpen op de consistente stand van de gegevens zoals die was toen de transactie werd gestart. De oude versie(s) van de gegevens worden vastgelegd een apart gegevens-gebied, tot en met versie 8 de rollback-segments geheten. Vanaf versie 9 is er een alternatief beschikbaar, de undo tablespace.

Tabellen zijn niet, zoals bij sommige andere databases, elk in een eigen bestand opgeslagen

De configuratiegegevens over de database.
De configuratiegegevens over de database worden door Oracle vastgelegd in een *controlfile*. Dit controlfile bevat van de verschillende hierboven genoemde bestanden de plaats en grootte, er worden (afhakelijk van de gekozen backup-methode) gegevens over backups bewaard, en de gegevens over de transactielogs worden er vastgelegd. Er wordt ook een uniek transactievolgnummer vastgehouden. Hiermee is deze controlfile niet erg groot, maar wel zeer belangrijk voor de consistentie van de database, en noodzakelijk om de basisinformatie voor een restore/recovery te kunnen leveren als dat nodig is. Van de controlfile worden dan ook minimaal twee, en meestal drie kopieën bijgehouden.

Het parameterbestand met de instellingen.
Het starten van 'de database' is bij Oracle in werkelijkheid het starten van een *instance*. Met een instance wordt in Oracle-jargon de verzameling (kernel-)processen en het centrale geheugen daarvan aangeduid. De 'database' is de verzameling bestanden die hiervoor onder 1 t/m 5 zijn genoemd. Als de instance wordt gestart leest deze een bestand met parameters in, van oudsher het zogenaamd 'init.ora' bestand. Met Oracle 9 is er een alternatief voor dit init.ora bestand gekomen, 'spfile' genoemd. Beide formaten worden ondersteund, en elk heeft zijn eigen voor en nadelen. Het parameterbestand met instellingen wordt eenmalig gelezen bij het starten van de instance, en is daarna niet meer nodig. Voor de performance zijn dan ook alleen de in het bestand gevonden instellingen van belang, de snelheid waarmee het bestand kan worden gelezen is voor het gebruik van het systeem irrelevant. Dit bestand wordt dan ook niet verder behandeld.

Eisen aan het opslagmedium

Tabellen en indexen worden door Oracle opgeslagen in tablespaces. Een tabel of een index kan zich in maximaal één tablespace bevinden. Een tablespace is een logische eenheid, die op zijn beurt uit één of meer bestanden bestaat. Als een bestand volloopt, kan er eenvoudig een nieuw bestand aan de tablespace worden toegevoegd. Daarmee kunnen alle objecten in de tablespace weer verder groeien. Tabellen zijn dus niet, zoals bij sommige andere databases, elk in een eigen bestand opgeslagen. Er bestaan meerdere typen objecten, zoals bijvoorbeeld clusters en gepartitioneerde tabellen. Objecten zijn onderverdeeld in segments. Van de indeling van de tablespaces en de objecten wordt uiteraard een uitgebreide adminstratie bijgehouden.

> Bij het inrichten van de tablespaces moeten hotspots zoveel mogelijk over verschillende schijven worden verdeeld

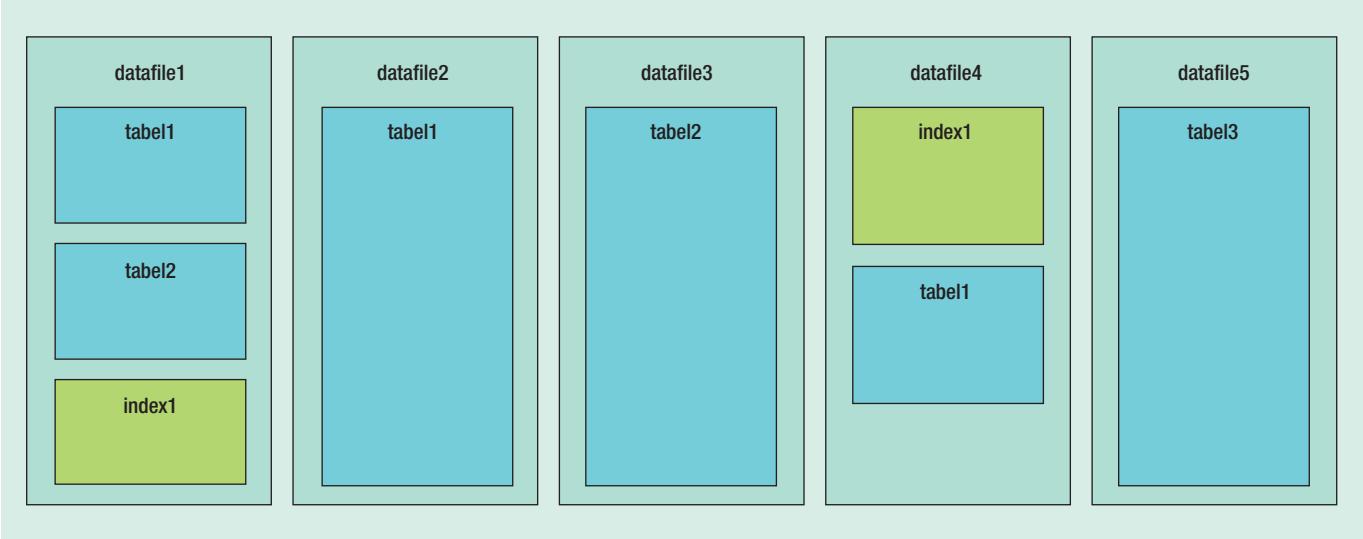
In afbeelding 1 is schematisch een tablespace 'DATA' weergegeven. Aan deze tablespace zijn vijf datafiles toegewezen, datafile1 tot en met datafile5. In de eerste datafile is te zien dat een deel van tabel1 en van tabel2 is opgeslagen. Ook is er nog een stukje van index1 vastgelegd in deze datafile1. De tabellen groeien, en zo is tabel1 doorgegroeid in datafile2, en vervolgens nog een stukje in datafile4. Tabel2 is doorgegroeid in datafile3. Kennelijk zaten al deze datafiles al vol toen tabel3 werd gecreëerd. Deze is dan ook in datafile5 terecht gekomen. Alle wijzigingen in de tablespaces worden in eerste instantie in de online redo logfiles beschreven, later worden ze pas in de datafiles zelf doorgevoerd. Daarop is een uitzondering: wijzigingen in een temporary tablespace worden niet in de transactielog bijgehouden.
De verschillende soorten gegevens stellen ieder hun eigen eisen aan het opslagmedium. Uiteraard hangt alles af van de totale werklast die het systeem ondergaat, een kleine database kan in principe op 1 schijf staan, die dan bij voorkeur wel is gemirrored. Alle gegevens op 1 schijf zetten levert echter wel snel schaalbaarheidsproblemen op, zoals uit de volgende beschrijving zal blijken.

De data.
De tablespaces met data worden meestal random gelezen en geschreven. Sommige plaatsen op een schijf worden daarbij vaker bezocht dan andere. De meestbezochte plaatsen noemen we ook wel *hotspots*. Het is zaak bij het inrichten van de tablespaces ervoor te zorgen dat hotspots zoveel mogelijk over verschillende schijven worden verdeeld. Immers, als een schijf meerdere hotspots kent wordt deze schijf al snel de beperkende factor in de totale systeemcapaciteit. Met verschillende schijven bedoel ik fysiek aparte schijven, ook wel *spindles* genoemd. Een schijf van een windows-systeem die in een C: en een D: 'schijf' is verdeeld telt toch maar voor 1 spindle.

Tijdelijke gegevensopslag voor sorteeroperaties.
De tijdelijke gegevensopslag wordt gebruikt als het interne geheugen niet toereikend is. Probeer het gebruik van de temporary tablespace daarom in de eerste plaats te voorkomen door voor voldoende geheugen en de goede bijbehorende instellingen te zorgen. Wordt er dan toch gebruik van gemaakt, dan is het zaak dat deze gegevens zo snel mogelijk kunnen worden geschreven en gelezen. Verder is het niet interessant om deze gegevens te kunnen bewaren. Als het systeem crasht zijn de tijdelijke gegevens niet langer nodig. Men moet daarom voor de temporary tablespace een snelle schijf gebruiken, die niet voor teveel andere zaken wordt gebruikt.

De transactielog.
OLTP-systemen en batch-georiënteerde systemen kunnen bij grote werklast snel tegen beperkingen in de schrijfcapaciteit van de transactielog aanlopen. Dit online redo log-bestand ontvangt alle transacties. Is het vol dan wordt door het archiveerproces direct een kopie gemaakt. Om nu te voorkomen dat kopiëren van het ene en het schrijven van het volgende online redo log-bestand elkaar in de weg zitten, worden de online redo log-bestanden op verschillende schijven (met bij voorkeur verschillende controllers) geplaatst. De aanbevolen twee kopieën van de online redo log-bestanden moeten zeker op verschillende schijven staan. Zouden ze op dezelfde schijf staan, dan leidt het (synchroon) schrijven van deze kopieën tot extra wachttijd en daarmee tot vertraging van het voltooien van een transactie. Bovendien levert het geen voordeel op met betrekking tot beschikbaarheid. Als de schijf kapot gaat zijn beide bestanden weg.

De gearchiveerde transactielog.
In principe moet er net zo snel kunnen worden geschreven naar



Afbeelding 1: Wijzigingen in de tablespaces worden in eerste instantie in de online redo logfiles beschreven.

de gearchiveerde redo log-bestanden als naar de online redo log-bestanden. Het archiveren loopt weliswaar iets achter, en er is een kans dat het proces door het continue karakter iets efficiënter met de middelen kan omgaan, maar tijdens een wat langer durende piek kan het archiveringsproces de flessenhals gaan vormen. Bij voorkeur krijgen dan ook de gearchiveerde redo logfiles hun eigen schijf.

De undo- of rollback-gegevens.
Bij grote transacties is de doorvoersnelheid mede afhankelijk van de snelheid van het medium waarop deze gegevens worden opgeslagen. Als dit bijvoorbeeld op dezelfde schijf als de data gebeurt, dan kan veelvuldig lezen van data het schrijven van de undo- of rollback-gegevens hinderen. Opnieuw: voor systemen met een grote werklast is een eigen schijf hiervoor optimaal.

De configuratiegegevens over de database.
De controlfile wordt frequent bijgewerkt (iedere drie seconden). Het is aan te bevelen meerdere kopieën (minimaal twee) van de controlfile aan te houden. Als deze op dezelfde schijf staan geldt hetzelfde als voor de online redo log-bestanden: het synchroon schrijven van de kopieën kost extra tijd, en extra beschikbaarheidsvoordelen zijn er dan niet.

Optimale indeling

Een optimale indeling vraagt om een minimale set van 13 schijven: een voor de data dictionary, een voor de data zelf, een voor de temporary tablespace, drie keer twee = zes voor drie keer twee kopieën van online redo log-bestanden, een voor de gearchiveerde redo log-bestanden, een voor de undo/rollback en twee voor de twee kopieën van de controlfile. Dat zal in de praktijk niet voor veel systemen nodig zijn. Wel is het belangrijk gegevens goed te scheiden, en goed met de eigenschappen van de onderliggende opslagsystemen rekening te houden. RAID-5 is nu eenmaal per definitie minder geschikt voor (zeer) frequente schrijfoperaties: zet dus de redo log-bestanden op RAID 1+0 systemen. Men moet er bij grote virtuele op RAID gebaseerde systemen rekening mee houden, dat hotspots van twee *volumes* onbedoeld op 1 schijf terecht kunnen komen. Ik heb in de praktijk applicaties elkaar op die manier in de weg zien zitten, zonder dat men snel kon ontdekken wat nu de oorzaak was.
Hopelijk is met dit artikel iets meer inzicht gegeven in de storage-wensen van een Oracle database. Meer informatie is bij de links op www.baarf.com, en verder is veel informatie te vinden in het (gratis downloadbare) boek 'Scaling Oracle 8i' van de website van James Morle, www.scaleabilities.com. Ondanks de titel is dit een zeer lezenswaardig boek.

Een optimale indeling vraagt om een minimale set van 13 schijven

Oracle gaat inmiddels met ASM het managen van schijven automatiseren. Daarbij worden de schijven bij voorkeur als ruwe devices bij Oracle aangemeld, en het optimaliseren van de indeling wordt vervolgens automatisch en voortdurend gedaan, ook als er schijven worden toegevoegd. Dan worden de gegevens 'on-the-fly' herverdeeld. ASM kan het leven van de DBA aanzienlijk vereenvoudigen. Maar ook met ASM blijft het uitermate belangrijk dat binnen een organisatie de storage-mensen met de database-mensen samenwerken om een goede dienst te leveren.

Carel-Jan Engel (cjpengel.dbalert@xs4all.nl) is onafhankelijk Oracle-specialist op het gebied van database-ontwerp, trouble-shooting en tuning.