Nieuwe features Oracle 11g (1)

Data Guard

Sinds een maand of twee kan de nieuwste versie van het RDBMS van Oracle worden gedownload. De geruchtenstroom (of zo u wilt – marketingmachinerie) rondom deze release was al geruime tijd op gang. Al in november 2006 gaf Tom Kyte een 'sneak preview' tijdens de jaarlijkse conferentie van de UKOUG in Birmingham. Alhoewel, 'sneak' is wat veel gezegd: er kunnen 1500 toeschouwers in het theater waarin hij deze presentatie gaf, en zoveel mensen zaten er ook minimaal in de zaal.

Uiteraard ging mijn bijzondere belangstelling uit naar de uitbreidingen die op het gebied van High Availability werden voorzien, en dan met name op het gebied van Data Guard. De punten die het meest mijn aandacht trokken waren: de mogelijkheid om een Physical Stand-by Database tegelijkertijd te openen voor Read-Only Access én te laten bijwerken met transacties vanuit de Primary Database, de mogelijkheid om het gegevenstransport te comprimeren, de (helaas niet ingeloste) belofte om eindelijk Rolling Upgrades te doen en tenslotte een trigger die wordt geactiveerd bij een rolverandering van Primary naar Stand-by database en vice versa. Met deze features zet Oracle weer een volgende stap in de perfectionering van de Stand-by Database.

Stand-by

De eerste schreden op dit gebied werden met Oracle 7 gezet. Oracle maakte het mogelijk om een back-up te maken van een database, en deze tegelijkertijd te blijven gebruiken, de zogenaamde 'hot back-up'. Omdat de database bestanden tegelijkertijd werden gekopieerd naar tape en beschreven als gevolg van transacties vanuit de applicatie konden makkelijk inconsistenties ontstaan in het back-upbestand. Een databaseblock (dat immers vaak groter is dan een fysiek block op schijf) kan zelfs in twee gedeelten van verschillende versies op tape eindigen. Dergelijke block-corrupties in het backupbestand konden na een eventuele restore worden hersteld, door de tijdens en na de backup gearchiveerde redo log files in een recovery-operatie 'af te spelen' op de teruggezette backup. Hiermee werd het mogelijk alle ge-comitte transacties terug te halen na een systeemcrash, mits uiteraard alle redo log bestanden beschikbaar waren. Als de schijven na een crash nog beschikbaar waren, was dat geen probleem. Verder was het zaak de gearchiveerde redo log files zo snel mogelijk op een ander medium (lees: tape) te plaatsen. Er is op dat gebied nog weinig veranderd. Wel bedachten een paar inventieve geesten dat het helemaal niet nodig is om te wachten op een systeemcrash voordat er een restore kan worden uitgevoerd. Door een tweede systeem neer te zetten, de back-up direct na het maken ervan op dit systeem te restoren en vervolgens de gearchiveerde redo log files direct op dit tweede systeem 'af te spelen' kon direct na een crash van het eerste systeem op het tweede systeem verder gewerkt worden. Zo werd de stand-by database geboren. In het begin was het vooral een creatief gebruik van de standaard restore en recovery faciliteiten, maar al snel pikte Oracle het idee op en breidde de commandoset uit met speciale opdrachten voor het beheer van Standby Databases. Toch bleef het lange tijd behelpen: het kopiëren van de archived redo log files moest met zelfgemaakte scriptjes worden geregeld, en dat was best wel foutgevoelig. Als er één bestand werd gemist, liep de boel vast en moest 'met de hand' weer aan de gang worden gebracht. In het ergste geval was de situatie te lang onopgemerkt gebleven en was de archived redo log file er niet meer. Dan moest er weer een back-up aan te pas komen om een nieuwe Standby Database te maken. Ook was een belangrijk nadeel dat een eenmaal als Primary Database geopende Standby Database niet meer kon worden teruggebracht in de rol van Standby Database. Er moest ook dan met een back-up een nieuwe Standby Database worden gemaakt. Op een paar platforms (Solaris, HP-UX en AIX) werd dit in Oracle 8.1 een stuk beter. Er kwam een op scripts gebaseerde versie van Data Guard. Dit werd echter niet echt aan de grote klok gehangen. Deze oerversie van Data Guard kende al automatische detectie van 'gaps' (ontbrekende archived redo log files) en ook kon in deze versie de rol tussen Primary en Standby database heen en terug worden gewisseld zonder dat er een nieuwe Standby Database moest worden aangemaakt. Deze versie had boven-

dien de mogelijkheid om de Standby Database voor 'alleen-lezen' te openen, terwijl het versturen van archived redo log files doorging. Ten slotte werd in deze versie het versturen van de redo informatie standaard gecomprimeerd uitgevoerd.

Behoud van ellende

Oracle 9i bracht veel veranderingen in de opzet van Data Guard. Het redo transport werd van scriptjes overgebracht naar de database kernel. Daardoor kwam Data Guard beschikbaar voor alle platforms. De Primary en Standby Database gingen rechtstreeks met elkaar communiceren, zodat ook de nog niet gearchiveerde redo informatie al naar de Standby Database kon worden gestuurd. Daarmee kon het risico op gegevensverlies tot nul worden gereduceerd. Uiteraard bleef en blijft de wet van behoud van ellende van kracht: 'gegarandeerd geen gegevensverlies' kost wel wat performance, afhankelijk van de belasting van het systeem en de bandbreedte van het netwerk. Die netwerkbelasting werd tegelijkertijd ook wat groter: er was geen mogelijkheid meer tot compressie. Als compressie beslist nodig was, moet daarvoor worden teruggevallen naar TCP/IP tunnels via gecomprimeerde ssh-sessies, of op hardwarematige oplossingen in de netwerk infrastructuur. Het beheer van Data Guard werd ook wat gecompliceerder. Oracle streeft er in de strijd met Microsoft's SQL Server nadrukkelijk naar om het beheer te vereenvoudigen, en in versie 9 werd mede daarom het aantal parameters fors gereduceerd. Deze 'vereenvoudiging' werd deels teniet gedaan door de configuratiemogelijkheden per parameter eindeloos uit te breiden. Dit wordt goed geïllustreerd door de parameter LOG_ARCHIVE_DEST_n, een van de belangrijkste parameters bij het inrichten van Data Guard. In versie 9.0 was er voor deze parameter een hoofdstuk van 35 pagina's ingeruimd in de handleiding. In versie 9.2 was dit hoofdstuk al gegroeid tot 54 pagina's. In versie 10 werd dit 55 pagina's. In versie 11g is dit eindelijk aanzienlijk vereenvoudigd: er zijn nu nog maar 22 pagina's nodig.

Sinds de presentatie van Tom Kyte zijn er vijf bèta-versies verspreid onder de bèta-testers, en 10 augustus was versie 11.1.0.6.0 eindelijk publiek beschikbaar, zij het alleen voor Linux, en dat is op het moment van schrijven nog niet veranderd.

Ongedurig

De opties die mijn meeste interesse hadden, heb ik uiteraard als eerste uitgeprobeerd. Dat heeft helaas nog niet tot onverdeeld enthousiasme geleid. Mijn ongedurig karakter, samen met het lange wachten op de software, leidt al snel tot een aanpak in de trant van 'If nothing works, then read the manual'. Nieuwe zaken onderzoek ik eerst *quick & dirty*, de gedegen testen komen later als ik de mogelijkheden (en onmogelijkheden) heb verkend. De software is dan ook snel in een verse VMWare machine met Centos5 geïnstalleerd. Met de DataBase Creation Assistant (dbca) heb ik vervolgens een set scripts voor het aanmaken van een database gegenereerd. Ik maak bij voorkeur gebruik van scripts voor het aanmaken van een database, maar bij een nieuwe versie maak ik de eerste keer altijd gebruik van de dbca om scripts te genereren. Zo kan ik snel zien op welke punten dit proces is gewijzigd. Wat opvalt zijn de grondige wijzigingen in de vertrouwde directorystructuur voor trace- en logfiles. De focus ligt in dit artikel echter op Data Guard, dus die veranderingen neem ik voorlopig even voor kennisgeving aan.

Uiteraard bleef en blijft de wet van behoud van ellende van kracht

Met ingang van versie 11g moet iedere database in een Data Guard omgeving verplicht een eigen waarde voor de parameter *db_unique_name* hebben. Naamgevingsconventies en herkenbaarheid blijven van het grootste belang in een complexe omgeving, en ik heb voor een naamgevingsconventie gekozen met de naam van de instance gevolgd door de naam van de server. Met de instance *d11a* en de servers *laphroaig* en *talisker* zijn de db_unique_names dus respectievelijk *d11a_laphroaig* en *d11a_talisker* geworden. Daarmee heb ik ook de 'verplichte' aanpassingen wel gehad, voor het overige bleek mijn script-set rondom Data Guard gewoon te werken.

Redo transport

De eerste optie die ik uitprobeerde was de compressie van het redo transport. Dat liep op een teleurstelling uit. In een 'bèta draft' werd bij de nieuwe features van Data Guard beschreven dat het redo transport zou worden gecomprimeerd. (Ik nam niet deel aan het bèta-programma, maar al snel was links en rechts uitgelekt dat op de documentatiesite van Oracle, tahiti.oracle.com, alle documentatie rondom versie 11 onbeschermd toegankelijk was (een beetje fantasievol met wat url's omgaan leverde alle documentatie op). Het eerste wat ik dan ook deed was via de index het woord COMPRESSION opzoeken. Door het opnemen van de optie COMPRESSION=ENABLE in de instellingen van de LOG_ARCHIVE_DEST_2 parameter wordt de compressie ingeschakeld.

```
SQL> alter system set LOG_ARCHIVE_DEST_2 = 'SERVICE=d11a2_dg LGWR SYNC
COMPRESSION=ENABLE DELAY=5';
```

Vervolgens heb ik een benchmark gedraaid, een paar keer met en een paar keer zonder compressie. Er was geen verschil… Daarom maar, volgens de gebruikte methode, de manual er op nageslagen. Die is keurig aangepast aan het in de productieversie van 11g waargemaakte resultaat: als compressie wordt ingeschakeld wordt het redo transport bij het verzenden van *ontbrekende archived redo log files* gecomprimeerd. Als er weinig bandbreedte beschikbaar is, en het komt regelmatig voor dat het netwerk uitvalt, dan kan het een voordeel bieden. Het is iets, maar tegelijkertijd toch wel erg teleurstellend, en zeker de extra licentiekosten (zie kader) niet of nauwelijks waard. Ik zou in dergelijke gevallen toch maar uitkijken naar een router met hardware-compressie.

Trigger

De volgende optie benaderde ik wel vanuit de manual: Rolling Upgrades. Oracle claimt al jaren dat het mogelijk is, maar nadere observatie leert steeds weer dat het nog niet is gerealiseerd. Een Rolling Upgrade maakt het mogelijk 'te verbouwen terwijl de winkel open blijft'. Vertaald naar Oracle-beheer: de database wordt ge-upgrade, maar de applicatie draait gewoon door. Eerdere claims maakten gebruik van een Data Guard Logical Standby Database. Een losse patch of patchset (geen nieuwe release) kan dan worden aangebracht op de Logical Standby Database, terwijl de applicatie gebruik blijft maken van de Primary Database. Vervolgens worden via een *switchover* de rollen van beide databases gewisseld, waarna de applicatie gebruik kan maken van de nieuwere versie in. Helaas moeten bij een switchover de applicaties uitloggen. Voor de gebruikers is dit dus geen Rolling Upgrade. Daarnaast is de hele operatie vrij complex, en is een Logical Standby Database vereist. De Logical Standby Database kent beperkingen met betrekking tot ondersteunde datatypes, en is daarnaast bij zwaar belaste systemen ook niet altijd in staat de Primary Database bij te houden. Maar goed, Oracle 11g beloofde opnieuw een Rolling Upgrade. In de beschrijving van de 'new features' van Data Guard wordt verwezen naar Hoofdstuk 12 van de handleiding. Het hoofdstuk begint met een opsomming van de voordelen van een Rolling Upgrade. Het eerste voordeel is: 'Uw database zal slechts gedurende een minimale periode niet beschikbaar zijn'. Daar ben ik maar weer opgehouden met lezen. Oracle databases hebben nog steeds geen echte 'Rolling Upgrade'.

De trigger die Tom Kyte besprak, kan volgens hem worden gebruikt om dynamisch parameters voor bijvoorbeeld de geheugenconfiguratie aan te passen, als deze in de Standby rol anders zou moeten zijn dan in de Primary rol. Ik vind dat geen goed idee. Als triggers worden geschreven om parameters aan te passen, betekent dat er opnieuw een plaats ontstaat waar informatie over de configuratie wordt vastgelegd. Dat maakt het geheel ondoorzichtig. Toen een aantal parameters werd uitgebreid met de 'VALID_FOR' optie, om verschillende instellingen te kunnen vastleggen voor verschillende rollen, is het rollenprobleem maar gedeeltelijk opgelost. Doordat maar een paar parameters rolafhankelijk kunnen worden geconfigureerd wordt een route met triggers ingeslagen. Veel beter zou het zijn als alle parameters kunnen worden voorzien van gelabelde waarden, en een instance kan worden gestart met de waarden die bij een bepaald label horen. Dan heeft de beheerder de vrijheid om een instance 'zuinig' te configureren als deze als Standby samen met een acceptatie-systeem op één server moet draaien, en een full-size configuratie vast te leggen voor de Primary rol. Als de database in de Primary rol moet draaien bij systeemuitval wordt de acceptatieomgeving stilgelegd om productie door te laten gaan. Eventueel kan dan voor een Read Only rol ten behoeve van een datawarehouse 's nachts nog een derde parameterset worden geconfigureerd.

Veelbelovend

Alle hoop was daarmee gericht op de laatste feature: Real-time Query of Physical Standby het voor 'alleen lezen' openen van een Physical Standby Database die tegelijkertijd wordt bijgewerkt vanuit de Primary Database. De eerste teleurstelling was dat voor deze feature betaald moet worden (zie kader). Deze feature heb ik eveneens via de handleiding benaderd. De handleiding is veelbelovend: er worden geen beperkingen genoemd die de eerder gewekte verwachtingen temperen. De werkwijze is ook erg eenvoudig en bestaat uit slechts drie stappen: stop het verwerken van de redo, open de database en vervolg het verwerken van de redo. In syntax is dat:

```
SQL> alter database recover managed standby database cancel;
SQL> alter database open;
SQL> alter database recover managed standby database using current log-
file disconnect;
```

Het lijkt kinderlijk eenvoudig, en dat is het ook. Direct na het uitvoeren van deze commando's heb ik in de Primary Database een tabelletje aangemaakt, met één column van het type date:

```
SQL> create table ts (ts date);
```

Direct daarna heb ik op de Standby Database gekeken of de tabel al was 'gearriveerd'. Deze bleef weg…..

Advertentie

Licentieperikelen

Data Guard en Standby Databases zijn altijd omgeven geweest met veel misverstanden als het om noodzakelijke licenties gaat. De licentievoorwaarden van Oracle zijn voor de leek niet echt doorzichtig, en het feit dat er voor verschillende High Availability oplossingen verschillende voorwaarden worden gehanteerd maakt het niet eenvoudiger. Met name de regel dat een server wel mag zijn voorzien van Oracle-software, maar dat daarvoor geen licentie nodig is als de server (vrij vertaald) niet meer dan tien dagen per jaar wordt gebruikt draagt bij aan de verwarring. Deze regel heeft uitdrukkelijk betrekking op een server waarop de Oracle-software alleen wordt gestart als de productie-server uitvalt. Een Standby Database wordt voortdurend bijgewerkt, daarvoor is de Oracle software gestart, en daarvoor is dan ook gewoon een licentie nodig. Dat geldt voor iedere Standby Database, of het nu de ouderwetse 'houtje-touwtje-scriptje' Standby Database op basis van Standard Edition (one) is of een geavanceerde 'zero data loss' Data Guard configuratie. Voor Data Guard is altijd de Enterprise Edition nodig, op alle servers, primary of standby. De budgetverantwoordelijken vinden dit altijd een beetje zuur: er staat een server, er draait software op, daar moet fors voor worden betaald, maar het geheel draagt normaal gesproken niet bij aan de gegevensverwerking. Ik begrijp dat niet. We nemen voorzorgsmaatregelen: een extra server, extra schijven, extra netwerkonderdelen, misschien zelfs een extra computerzaal op een andere locatie, en dan niet te vergeten de implementatie van het geheel. In dit grote plaatje wordt voor alles betaald, alleen de Oracle-software zou gratis moeten zijn? De hardware-leverancier schuift z'n dozen ook niet gratis naar binnen als het om een stand-by site gaat. Oracle heeft dit sentiment opgemerkt, en heeft ervoor gezorgd dat een Physical Standby Database nu kan worden gebruikt voor raadplegingen, terwijl deze tegelijkertijd wordt bijgewerkt. Het productiesysteem kan op deze manier van zware raadpleegacties kunnen worden ontlast. Zo wordt de Stand-by Database nuttig gebruikt. Er moet echter wel voor betaald worden: er wordt 25% van de listprijs per CPU berekend voor deze optie. Dat geldt voor de CPU's van zowel de Standby Database àls van de Primary Database. Eenzelfde opslag wordt berekend voor het gebruik van de compressie-optie. Deze optie kent meerdere toepassingsgebieden: onder adere RMAN, Datapump en OLTP Table compressie. Als compressie op die gebieden niet nodig is zou ik, gezien de beperkte functionaliteit van compressie in combinatie met Data Guard, zorgvuldig afwegen of deze optie z'n geld wel waard is.

Noot: Ik ben niet juridisch geschoold, aan deze tekst kunnen dan ook geen rechten worden ontleend.

Na een

```
SQL> alter system switch logfile;
```

was deze er echter al snel. Ik was vergeten dat mijn demo-instellingen standaard een DELAY van vijf minuten kennen voor het verwerken van redo.

Na de logswitch werd echter de 'using current logfile' optie geactiveerd, en deze zorgt ervoor dat een eventueel gedefinieerde DELAY wordt genegeerd. Het is noodzakelijk dat zogenaamde standby redo log files beschikbaar zijn bij de Standby Database, anders kan niet met de 'using current logfile' optie worden gewerkt. Het werkt ook snel. Op een zwaar belaste laptop met twee 11g databases in twee Virtual Machines waren de transacties binnen vijf seconden verwerkt op de Standby Database. Kortom, Real-time Query maakt de beloften waar.

Een Standby Database die wordt gebruikt met Real-time Query biedt geen bescherming meer tegen menselijke fouten! Door een vertraging op te nemen in het verwerken van de redo kan een eventuele menselijke fout relatief eenvoudig worden hersteld met behulp van een stand-by database. Als bijvoorbeeld data per ongeluk is verwijderd, of er is per ongeluk veel te veel ge-update door een verkeerde WHERE-clause, dan kan een vertraging van een paar uur op de Standby Database ervoor zorgen dat de oude situatie vanaf de Standby Database kan worden hersteld, zonder dat daarvoor de Primary Database down gebracht hoeft te worden. Voor het herstel van menselijke fouten zijn ook de verschillende Flashback-opties beschikbaar. Daarnaast wordt in omgevingen met hoge beschikbaarheidseisen vaak gebruik gemaakt van twee Standby Databases. In dat geval kan de tweede Standby Database nog steeds met een DELAY worden geconfigureerd om herstel van menselijke fouten mogelijk te maken.

Bladerend door de Data Guard Concepts & Administration Manual kwam ik in Appendix B nog steeds de in mijn ogen foute beschrijving voor het upgraden van een database in een Data Guard configuratie tegen. De handleiding beschrijft hoe de Standby Database actief redo verwerkt terwijl de Primary Database wordt ge-upgrade. In mijn ogen is dit in strijd met het streven naar hoge beschikbaarheid: als het upgrade-proces om één of andere reden faalt, dan zijn beide databases onbruikbaar. Veel beter vind ik het om de instance van de Standby Database te stoppen tijdens het upgrade proces. Als er dan tijdens de upgrade van de Primary Database wat fout gaat dan volstaat het starten en activeren (als Primary Database) van de Standby Database om het systeem weer in

de lucht te brengen. Pas nadat upgrade van de Primary Database succesvol is verlopen wordt de Standby Database met de nieuwe versie van de RDBMS software gestart, en worden de archived redo log files met de wijzigingen uit het upgrade proces verwerkt.

Conclusie

Sommige features die eerder werden beloofd zijn nog niet helemaal uitgekristalliseerd. De 'Rolling Upgrade' is nog steeds marketing speak. De downtime bij een upgrade wordt minder, maar het blijft een complex geheel dat nog steeds zijn naam niet waarmaakt. Compressie van redo transport heeft het kennelijk niet helemaal gered voordat de productieversie van 11g eruit moest. Alleen maar comprimeren bij het wegwerken van gaten in de redo stream is iets, maar niet voldoende. Juist compressie van de hele redo stream is voor veel sites van belang. Dit zal hopelijk bij een volgende release worden gerealiseerd. De huidige feature biedt onvoldoende waar voor een 25% prijsverhoging. De role-change trigger is aardig, maar ik zie het nut niet zo sterk. Voor dynamisch aanpassen van parameters zou ik deze zeker niet inzetten. Real-time query tenslotte maakt de toezegging volledig waar. Toch is ook hier een 25% opslag op de listprijs aan de forse kant.

Carel-Jan Engel werkt als onafhankelijk Oracle-consultant. Hij is lid van het Oak Table Network. E-mail: cjpengel.dbalert@xs4all.nl.