Replicatie met open source database

PostgreSQL: alternatief dat voor de hand ligt

Rick van Rein

Het is bekend dat PostgreSQL regelmatig met succes wordt ingezet als vervanger van Oracle. Dat het een open source-systeem is maakt het heel bijzonder want het is niet gewoon een solide database, er wordt ook hard aan door-ontwikkeld, bijvorbeeld aan replicatiemethoden.

Er zijn twee databases die veel worden toegepast in de open source-gemeenschap. De ene is MySQL, een commercieel product met een soort open source uitlaatklep. De andere is PostgreSQL, een product waaraan behalve door programmeurs over de hele wereld, veel door universitaire onderzoekers wordt bijgedragen. MySQL wordt doorgaans gebruikt voor bijvoorbeeld een hippe web-interface die wat dynamiciteit behoeft, terwijl PostgreSQL voldoende zwaar geschut is om een alternatief voor Oracle te zijn. Dat PostgreSQL zaken als transacties, triggers en backups kan regelen is vanzelfsprekend. Veel interessanter wordt het als we in speciale faciliteiten duiken, zoals replicatie van de databases, ofwel het draaien van meerdere databases met dezelfde inhoud op verschillende computers.

Waarom replicatie?

Er zijn enkele redenen te bedenken waarom replicatie van een database nuttig kan zijn. Eén reden bijvoorbeeld betreft de backup. Men kan denken aan een online-betalingssysteem dat bij blikseminslag liever niet de hele administratie kwijt wil zijn. Het systeem gebruikt minimaal een offsite backup om te voorkomen dat een lokaal probleem het gehele systeem in de problemen brengt. Maar omdat een traditionele backup achterloopt op de echte database en omdat de klanten niet gevraagd kan worden hun transacties opnieuw te doen, is replicatie in zo'n geval een mechanisme om een backup achter de hand te hebben die bij is.
Een tweede reden voor replicatie kan zijn dat zo'n gekopieerde database de rol van 'hot backup' kan vervullen. Er zijn diverse systemen te bedenken om dat te automatiseren; meestal berusten die op het detecteren van een polsslag ('heart beat') van de andere systemen, en automatische herconfiguratie op het moment dat een systeem uitvalt.
Een derde reden voor replicatie is het verdelen van werklast door middel van een cluster, mogelijk met als neveneffect dat fouten in een minderheid van systemen kunnen worden opgemerkt. Hoewel succesvolle schrijfacties naar één database ook altijd op de andere databases zullen moeten draaien, zijn pure leesacties en sommige falende schrijfacties wel goed te verdelen.
Combinaties van redenen zijn natuurlijk ook mogelijk. Nemen we weer dat online betalingsmiddel, er vanuitgaand dat men nogal wat werk te verzetten heeft, dan is het misschien zinnig om in een cluster te werken en tevens een backup te hebben op een andere locatie. Dit is nuttig omdat een cluster op een lokaal netwerk draait, waardoor het vatbaar wordt voor locatiegebonden problemen.

Openheid zit hem in licenties

PostgreSQL is, net als een aantal vrije Unix-varianten, ontstaan op de Berkeley University in California. De licentie die daarbij gebruikt wordt is uitermate open, zodat iedereen bijdragen kan leveren en desgewenst een eigen vertakking van PostgreSQL kan maken, zoals bijvoorbeeld het hier besproken Postgres-R.
Het zijn deze open licenties die er voor zorgen dat open source werkt zoals het werkt. Als een project groot genoeg wordt om een actieve groep gebruikers aan te trekken, dan is ook de ondersteuning in de praktijk goed geregeld: over de meeste problemen is al eens iemand gestruikeld, en vraag en antwoord staan meestal wel ergens in een FAQ of mailing list archive.
Het belangrijkste effect van deze hele open benadering is dat PostgreSQL zo'n indrukwekkend product is dat allerlei add-ons (denk aan ODBC drivers of stukken toegepast database-onderzoek) gratis ter beschikking worden gesteld. De makers zijn vaak allang blij hun werk te kunnen toetsen aan een 'echte' omgeving, waarbij ze dan zoveel voor niets krijgen dat hun kleine uitbouwsel vanzelfsprekend ook vrij ter beschikking wordt gesteld. Wat ditzelfde effect weer bij anderen veroorzaakt, enzovoort.
Dat open source (of 'free software' volgens sommigen) behalve vrij ook gratis is, is wat kort door de bocht. Hoewel er geen verplichte kosten zijn, moet men wel de zaak zien te installeren, configureren en later wellicht upgraden. Maar ook als je voor deze zaken externe hulp betaalt blijft het zo dat je geen licentiekosten kwijt bent per gebruiker, per werkplek of per jaar. Open source licenties schalen dus heel prettig op, en dat staat steeds meer bedrijven aan.

Replicerende mechanismen

Er bestaan vier soorten mechanismen voor replicatie. Eigenlijk zijn het er twee keer twee, want er zijn twee keuzen die het soort replicatiemechanisme bepalen. De eerste keuze die gemaakt wordt door een replicatiemechanisme is of er een master is met meerdere slaves, of dat elke replica beschrijfbaar is. Een benaming die wel eens gebruikt wordt voor dat laatste is multi-master, ter onderscheid van single-master.
Het systeem met een enkele master is niet altijd handig. Als je begint met een transactie is voor de database nog niet te zien of er meer zal gebeuren dan alleen lezen, dus een doorverwijzing naar de master moet door de client worden geregeld; iets wat weer vervelend is omdat je op die plek eigenlijk niet een master wil vastpinnen. Een single-master schema wordt wel regelmatig gebruikt bij LDAP, omdat dat systeem vaak wordt gebruikt voor toepassingen die weinig schrijfacties verwerken. Voor SQL-systemen is toepassen van een multi-master een veel handiger keus.

De tweede keuze die de replicatiemechanismen indeelt is daarom veel interessanter. Het gaat hier om het moment waarop veranderingen worden gepropageerd van een master naar de slaves. Dat kan binnen de transactie vallen of er na. Als alle slaves worden bijgewerkt binnen de transactie, dan moeten ze wel allemaal tot dezelfde conclusie komen over het slagen of falen van de transactie; de standaardoplossing daarvoor is een *two-phase commit*. Dat kost tijd, zodat de algehele performance van de database minder lijkt. Om het nog wat erger te maken blijven locks gedurende die hele tijd aangevraagd staan, dus andere processen worden ook nog eens opgehouden of ze riskeren te worden afgebroken wegens deadlock.

> Voor SQL-systemen is toepassen van een multi-master een veel handiger keus

Van de andere kant, als slaves pas worden bijgewerkt na de afloop van een transactie, ontstaan mogelijke consistentieproblemen. Als twee masters tegelijk een transactie krijgen te verwerken en die daarna doorsturen naar hun soortgenoten, dan ontstaat de mogelijkheid dat ze met elkaar strijdige processen afhandelen. Het grote probleem hier is dat serialisation van transacties in gevaar komt als je meerdere masters tegelijk van buitenaf kunt aanspreken – niet elke database in het replicatieschema krijgt de transacties in dezelfde volgorde te verwerken en dus kunnen de databases met elkaar uit de pas gaan lopen.

PostgreSQL kan overigens beide varianten ondersteunen. Er is ondersteuning voor het two-phase commit protocol, zodat gedistribueerde transacties mogelijk zijn (zie kader 2PC). Verder bestaat er Slony-I, een uitbreiding op PostgreSQL die single-master replicatie implementeert. Hoewel het praktisch is dat een systeem als Slony-I bestaat, is het wat ongeïnspireerd om buiten transacties om kopietjes van een enkele leidende database te trekken. Het voorkomt de inconsistenties die een multi-master schema heeft, maar het heeft wel de nadelen van de single-master opzet.

Postgres-R combineert de voordelen

Universiteiten beschouwen een open source-systeem als PostgreSQL of Linux doorgaans als een uitstekende kans om aan te tonen dat onderzoeksprincipes werken in een praktisch product. Immers, een hele brede en praktisch bruikbare basis is reeds aanwezig, zodat alleen de wijzigingen die het gevolg zijn van de eigen ideeën hoeven te worden uitgeprogrammeerd. Dat is heel wat geschikter dan het wiel proberen uit te vinden, en het levert ook veel betere mogelijkheden om het onderzoek te toetsen; bijvoorbeeld door in benchmarks de performance voor en na de aanpassing te vergelijken.
Een voorbeeld van zulk praktisch onderzoek- en testwerk is Postgres-R, het promotiewerk van Bettina Kemme van het ETH in Zürich. Zij ontwikkelde een geheel nieuwe manier om replicatie in een database in te bouwen. Die manier regelt de replicatie

Wat was 2PC ook alweer?

Two-phase commit, vaak afgekort tot 2PC, is een commit-protocol dat het transactie-regime van afzonderlijke databases samenvoegt tot een gedistribueerde transactie, dus een transactie die alle deelnemende databases overspant. Overigens kunnen ook niet-database systemen deelnemen in zo'n transactie, zolang ze dit protocol maar ondersteunen.
Het 2PC-protocol bestaat uit twee fasen. In de eerste wordt al datgene geconcentreerd wat een lokale fout zou kunnen opleveren in één van de deelneemers in de transactie. Dus gebrek aan schijfruimte, consistentieproblemen en deadlocks moeten allemaal op de spits worden gedreven in de eerste fase, die 'prepare to commit' heet. Het SQL-commando daarvoor is PREPARE. Elke deelnemer aan zo'n gedistribueerde transactie rapporteert succes of falen. Bij succes staat alles klaar voor een commit, een knip met de vinger is bij wijze van spreken alles wat nodig is voor de echte commit.
In de tweede fase meldt de aanvoerder van de gedistribueerde transactie of de gehele transactie moet worden teruggedraaid of niet. Terugdraaien gebeurt als er minstens één lokale deelnemer was die problemen meldde, en alle lokale deelnemers reageren daarop met een rollback. In alle andere gevallen volgt de bevestigende vingerknip die de lokale deelnemers doet committen.
Helemaal waterdicht is dit proces niet. Als tijdens de tweede fase een probleem optreedt dat niet met programmatuur te ondervangen is, krijgt men alsnog een gefragmenteerde uitkomst; denk aan netwerkproblemen, blikseminslag of stroomstoringen. Maar de invloed van zulke storingen is met 2PC zo zeldzaam gemaakt dat handmatige correctie daarvan doenlijk wordt.

PostgreSQL in plaats van Oracle, kan dat?

Zo af en toe komen verhalen naar voren over de vervanging van Oracle door PostgreSQL. Ikzelf heb die optie ooit online gedocumenteerd en ben vervolgens benaderd door de gemeente Haren, waar de coördinator
informatiebeheer zeer vooruitstrevend bezig was met de invoering van open source software. Hij wilde zich loswerken van de destijds net aangetrokken licentievoorwaarden van Oracle, en overwoog er een applicatie naartoe te laten porten.
Dat is inmiddels realiteit geworden. De problemen die we tegenkwamen waren vrij beperkt: er moesten wat type-namen worden omgeschreven van de MS SQL- of Oracle-notatie naar die van PostgreSQL (het betrof hier uitbreidingen ten opzichte van standaard SQL); de database moest op het wat weerbarstige HP UX van de source worden opgebouwd en er waren drivers nodig voor ODBC onder Windows.

binnen de transactie, maar veel sneller dan met een standaard two-phase commit-schema mogelijk is. De uitdaging daarbij was een multi-master systeem binnen een cluster te ondersteunen. Goedbeschouwd is het centrale probleem bij een multi-master cluster dat de volgorde van de transacties in de war kan raken. Nu zijn er netwerkprotocollen die regelen dat multi-casts overal in dezelfde volgorde aankomen, en dat is voor Postgres-R als communicatiebasis gebruikt. Van de zekerheden die dat bood kon vervolgens slim gebruik worden gemaakt in de rest van het replicatiesysteem.

> Het is nog zaak vooraf bewust te kiezen welke variant geïnstalleerd moet worden

Allereerst volgt een observatie: een transactie die leest kan overal draaien, zolang het maar niet interfereert met locks van schrijvende transacties. Je kunt zeggen dat een transactie die alleen leest overal wel tussen te proppen is. Om dat te bevorderen maakt Postgres-R gebruik van de in PostgreSQL ingebouwde mogelijkheid om op een shadow copy van de database te werken; dat is een kopie die louter voor de huidige transactie zichtbaar is. Komt er tijdens het werken aan zo'n shadow copy een lezende transactie binnen, dan kan die gewoon lokaal worden afgehandeld op basis van de dan bestaande state. De database doet dus net alsof de lezende transactie binnenkwam voor de schrijvende transactie die op een shadow copy werkt. Dat soort *concurrency control* valt binnen de alledaagse verantwoordelijk van een database. Wat dus rest is het repliceren van schrijvende transacties. Het algemene idee is heel simpel: schrijvende transacties komen binnen, worden op het multi-cast protocol gezet en daarna komen ze overal – ook op de verzendende database – terecht in dezelfde volgorde, waarna ze kunnen worden uitgevoerd. Weliswaar is het daarbij mogelijk dat er nog wat mis gaat, maar dat gebeurt dan ook overal omdat de databases keurig in de pas lopen. Hooguit kan de ene database er wat eerder achter komen dan een andere, dus het blijft nuttig andere databases van zulke bevindingen op de hoogte te stellen. De eerste van zulke bevindingen kan als resultaat van de oorspronkelijke transactie worden teruggerapporteerd.
Het verwerken van de schrijfoperatie in elke database in het replicatieschema wordt gedaan op een shadow copy. In zo'n schaduw-database is het voor een transactie mogelijk om de eerder zelf weggeschreven data te bekijken en om constraints te controleren, alvorens tot een commit over te gaan, en dat alles terwijl de database voor andere transacties onveranderd lijkt.

Pervasive ondersteunt PostgreSQL

Pervasive heeft Pervasive Postgres introduceerd, het eerste geïntegreerde pakket van open source-software en -diensten van een gevestigde databaseleverancier. PostgreSQL levert een bedrijfsklare kerntechnologie met views, triggers, stored procedures en beveiliging. De software wordt gedistribueerd met een bedrijfsvriendelijke BSD-licentie, die het gebruik zonder royalty's in een commerciële context mogelijk maakt.
Pervasive zorgt voor formele ondersteuning, een duidelijke roadmap en technische uitbreidingen, zoals een naadloze geautomatiseerde installatie en gebruiksvriendelijke beheer-tools. Ook gaat Pervasive intensief samenwerken met de wereldwijde PostgreSQL gemeenschap. Verbeteringen en uitbreidingen aan de open source PostgreSQL server zullen weer teruggegeven worden aan de open source-gemeenschap.
Gilbert van Cutsem, VP EMEA bij Pervasive meldt desgevraagd aan DB/M: "Pervasive is nu in staat een grotere markt aan te spreken dan voorheen. De markt voor 'embedded' databases genereert ongeveer 300 miljoen to 500 miljoen dollar in totale verkoop. Terwijl de 'mainstream' database-markt circa 11,5 miljard dollar groot is. En wij bevinden ons op dit moment niet in deze markt.
Open Source-oplossingen krijgen steeds meer voet aan de grond, ook in Europa. Kijk maar naar de Europese overheden, waar veel open source-projecten lopen of op de agenda staan. Pervasive Postgres is een open source-alternatief voor mainstream bedrijfstoepassingen; de software wordt gedistribueerd met een BSD-licentie. Middelgrote en kleine bedrijven die de stap naar open source doen, hebben ook tools voor migratie en real-time oplossingen nodig. En deze technologie hebben wij in huis."

Dankzij locks wordt wel gezorgd dat meerdere transacties niet tegelijk dezelfde data kunnen veranderen. Op het moment dat de lokale database waar de transactie op werd losgelaten akkoord gaat met de data, kunnen alle wijzigingen als geheel aan de andere databases in het gerepliceerde schema worden verzonden. Doordat een hele transactie in één keer wordt verzonden blijft de communicatie-overhead minimaal. Bovendien is het mogelijk voor zo'n replica om alle locks voor een transactie in één keer te bemachtigen en dus al vroeg te ontdekken dat het uitvoeren van de transactie geen zin heeft.

Postgres-R is op dit moment niet in de standaard-releases van PostgreSQL opgenomen. De toevoeging -R betreft een zogenaamde 'fork' van de normale codeboom, dus een aangepaste versie op basis van een wat oudere versie van de originele PostgreSQL. Er is beslist interesse om deze integratie te regelen, zowel bij gebruikers als bij de actieve ontwikkelaars van PostgreSQL en Postgres-R, dus het zal er nog wel eens van komen. Tot dan is het nog zaak vooraf bewust te kiezen welke variant geïnstalleerd moet worden.

In vergelijking

Het Postgres-R systeem presteert beter dan een standaard two-phase commit-protocol doordat het al bij de eerste terugmelding kan beslissen of een transactie doorgang moet vinden of niet. Ook de optimalisatie door het tegelijk aanvragen van alle locks voorkomt verspilling van processor-tijd. Verder is het denkbaar dat een database puur als slave wordt opgezet, om daarmee een offsite backup te implementeren. Zolang alle schrijvende transacties maar in de juiste volgorde worden doorgezonden naar de backup, doet het er niet veel toe als die ze wat later ontvangt dan de systemen op een LAN. Immers, er hoeft met Postgres-R niet te worden gewacht op succesconfirmatie van de backup, dus de backup hoeft geen vertragende factor te zijn.
Het is opmerkelijk dat PostgreSQL aardig wat keuzes biedt op het gebied van replicatie. En replicatie is echt geen uitzondering, in het algemeen is PostgreSQL een vooruitstrevend systeem dat veel slimme ideeën bevat. Het wordt dan ook niet ontwikkeld vanuit een rechtlijnige bedrijfsvisie, maar veeleer vanuit een wens naar technologische vooruitgang. En dat houdt in dat er veel nieuwe technologieën in worden uitgeprobeerd, zelfs als die in wezen elkaars tegengestelde zijn. Wat uiteindelijk in PostgreSQL terechtkomt is dan vaak de beste, of een combinatie – met als eindresultaat het ideaal van vrije software: de gebruiker krijgt een vrije keuze uit de beste alternatieven.

Rick van Rein
Dr. ir. H. van Rein (rick@openfortress.nl) is ontwikkelaar en beheerder bij OpenFortress Digital signatures.