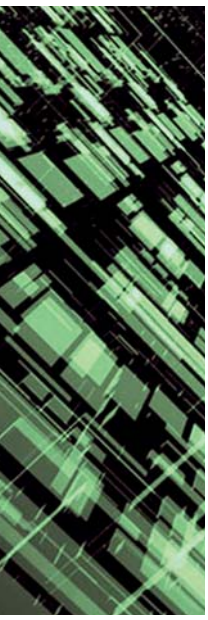


Doorontwikkeling na de overname wordt verwacht

MySQL, Sun en Oracle: de stand van zaken

Roland Bouman

Het is de laatste jaren tamelijk roerig geweest rondom het populaire open source database managementsysteem MySQL. Zo werd MySQL ab, het van oorsprong Fins-Zweedse bedrijf dat de softwareontwikkeling van het product vanaf 1995 leidde, in 2008 overgenomen door Sun Microsystems voor een totaalbedrag van 1 miljard dollar. Helaas voor Sun heeft die overname de reeds eerder ingezette dalende trend van diens aandelenkoersen niet gekeerd.

Het op 20 april van dit jaar aangekondigde voornemen tot overname van Sun door Oracle heeft onder de MySQL gebruikers tot een golf van speculatie geleid omtrent de toekomst van het MySQL databasesysteem. Inmiddels heeft Neelie Kroes, Eurocommissaris voor Mededinging haar zorgen omtrent het voortbestaan van MySQL uitgesproken indien de overname uitgevoerd zou worden, en een onderzoek ingesteld om de rechtmatigheid ervan te onderzoeken [1]. Tijd dus voor een overzicht van de stand van zaken omtrent MySQL en een analyse van het toekomstperspectief.

MySQL: algemeen overzicht

De boodschap komt wellicht reeds wat uitgekauwd over: MySQL staat vooral bekend als een RDBMS met een tamelijk beperkte feature set, dat eenvoudig installeerbaar en simpel in het gebruik is, en verder vooral uitblinkt doordat het dankzij diens open source licentie (meestal) kosteloos inzetbaar is. Daarnaast is er een enterprise editie beschikbaar op basis van dezelfde code, die met support en hot bugfixes commercieel wordt uitgebaat door Sun Microsystems (voorheen: MySQL ab).

Ook is het algemeen bekend dat MySQL vooral vaak wordt ingezet als back-end voor allerhande webtoepassingen van variërende omvang: van het gastenboek op uw persoonlijke homepage tot aan wereldwijde megasites als eBay, Facebook, Wikipedia, en zelfs Google en Yahoo! – allemaal maken ze ergens gebruik van MySQL [2].

Nu wordt vaak gedacht dat de populariteit van MySQL voornamelijk kan worden verklaard vanuit de afwezigheid van licentiekosten. Dit is zeker een belangrijke factor, maar toch is dit slechts één aspect. Het product heeft een aantal functionele kenmerken die het bij uitstek geschikt maken voor juist het bouwen van webtoepassingen. Een aantal van deze kenmerken bespreken we.

Waarom het web?

Een van MySQL's gunstige eigenschappen is het feit dat een verbinding met de server tamelijk snel tot stand kan worden gebracht in vergelijking met bijvoorbeeld Oracle. Hierdoor kunnen webpagina's die uit de database afkomstige content bevatten, snel worden geserveerd.

Ook levert MySQL vooral voor simpele SELECT query's een goede performance. Voor een grote klasse van webapplicaties is dit nu juist een veel voorkomende workload. Er zijn enkele middelen die de prestaties van dergelijke query's kunnen helpen. Zo kan er gebruik gemaakt worden van de query cache [3], waarin een reeds gematerialiseerd queryresultaat kan worden opgeslagen zodat het later snel kan worden opgezocht, zonder het statement zelfs maar een tweede keer te parsen. Verder kan er voor gekozen worden om geen gebruik te maken van transacties.

Het niet gebruiken van transacties stuit nogal wat database ontwikkelaars tegen de borst; immers, hoe kunnen integriteit en isolatie gewaarborgd worden zonder transacties? Nu is dit inderdaad niet aan te bevelen voor typische OLTP-toepassingen, maar het past goed bij de workload van een grote klasse van websites die voornamelijk leesverkeer kennen. Denk bijvoorbeeld eens aan een site als hyves.nl. Op deze site is helemaal rechts onderaan de pagina een kader te vinden met statistieken, waarin onder meer het aantal leden en het aantal vandaag geplaatste berichten worden getoond. Hoe erg is het nu eigenlijk als deze statistieken zeggen dat er momenteel 9.200.253 leden zijn hoewel het er in werkelijkheid 1, twee of zelfs honderd meer of minder zijn? Overigens heeft de mogelijkheid om geen gebruik te hoeven maken van transacties geleid tot het nog steeds hardnekkige misverstand dat MySQL geen transacties zou ondersteunen. Om goed te begrijpen hoe die vork nu precies in de steel zit, moeten we iets dieper ingaan op MySQL's architectuur, hetgeen verderop in dit artikel besproken wordt.

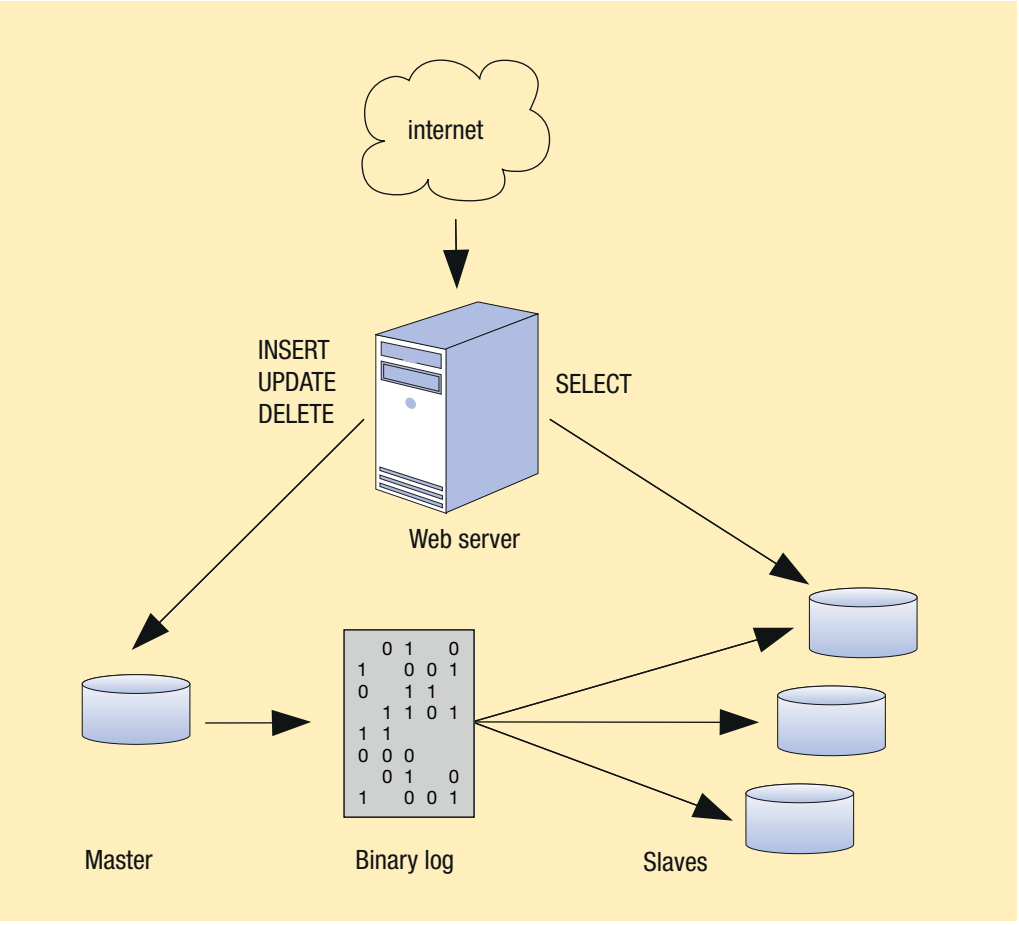
Een andere factor die van betekenis is voor veel van de bestaande webtoepassingen is dat het betrekkelijk eenvoudig is om met MySQL een master/slave replicatiecluster te bouwen [4]. Een dergelijk replicatiecluster is een simpel middel om zowel de leescapaciteit als de beschikbaarheid van een snelgroeiende website op flexibele wijze te kunnen controleren volgens het scale-out principe. Bij de grotere websites komt dit fenomeen vaak voor, ook hierop gaan we verder in dit artikel nog wat dieper in.

Genoemd is dat MySQL software kosteloos beschikbaar is onder de voorwaarden van een open source licentie. Maar de eigenlijke essentie van open source is natuurlijk de beschikbaarheid van de broncode, én het recht om deze code aan te passen naar eigen inzicht. Dit laatste is vooral voor de hele grote websites zoals Facebook en Google van belang. De requirements voor wat betreft het aantal simultane gebruikers, datavolumes en paginaverzoeken zijn zo extreem, dat er geen out-of-the-box oplossingen op passen. Zelf de software kunnen aanpassen aan de eigen doeleinden om deze te combineren tot een architectuur die dergelijke sites kan dragen, is voor deze bedrijven van levensbelang [5], [6].

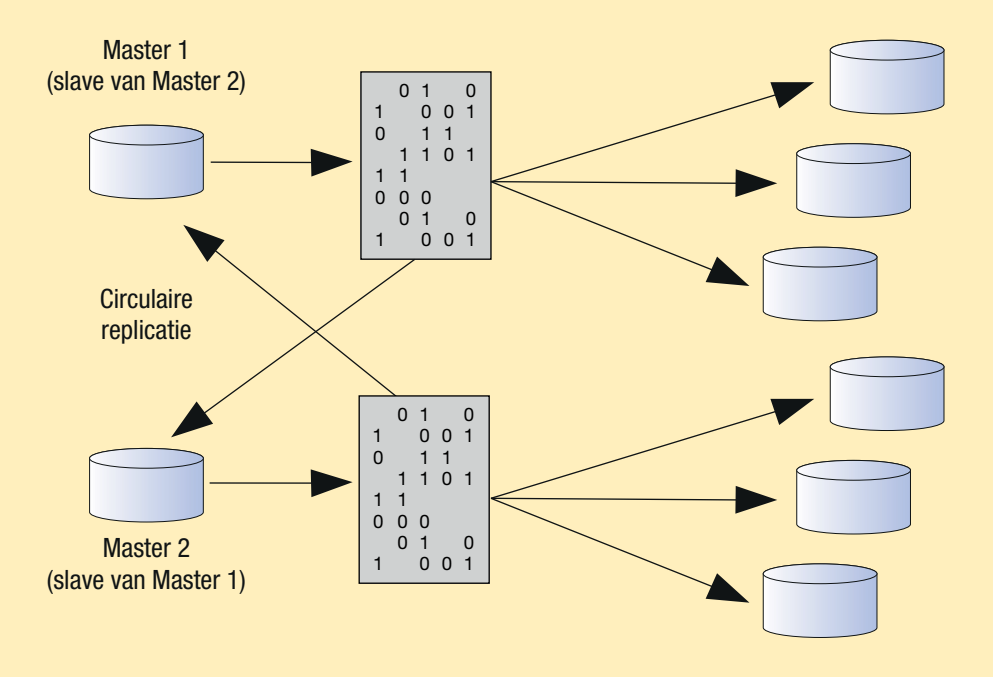
Scale-out met MySQL replicatieclusters

In een MySQL master/slave replicatiecluster wordt één database instance door de (web) applicatie gebruikt om alle wijzigingen (INSERT, UPDATE, DELETE) op de data uit te voeren. Dit is de master. De master houdt een eventlog bij van alle wijzingen: de zogenaamde binary log. Deze binary log wordt bijna gelijk doorgestuurd naar meerdere (tientallen of soms honderden) andere database instances, de slaves. De slaves worden door de applicatie exclusief gebruikt om data te lezen (SELECT), zie afbeelding 1.
De leescapaciteit kan met een dergelijk cluster flexibel worden gecontroleerd door achteraf slaves toe te voegen of af te koppelen. Dit wordt scale-out genoemd en verschilt radicaal van scale-up, waarbij de capaciteit van een enkele machine wordt vergroot door hardware upgrades.

Scale-out vertaalt zich in een grotere throughput: het aankunnen van grotere aantallen verzoeken. Maar grotere throughput kan ook benut worden om responstijden te verlagen: in plaats van in één paginaverzoek meerdere database query's sequentieel uit te voeren, kan een webapplicatie zo gebouwd worden dat elk databaseverzoek in een eigen webpaginaverzoek wordt uitgevoerd. Met behulp van AJAX-technieken kan de hoofdpagina meerdere van deze verzoeken asynchroon en parallel uitvoeren, waardoor de laadtijd van de pagina als geheel afneemt. Hierdoor neemt het aantal verzoeken op de webserver weliswaar toe, maar neemt de wachttijd tengevolge van database I/O af, met lagere responstijden als resultaat.



Afbeelding 1: MySQL master-slave replicatie.



Afbeelding 2: MySQL multi-master-slave replicatie.

Replicatieclusters en beschikbaarheid

Tot slot is er nog een bijkomend voordeel van het werken met een replicatiecluster. Omdat er meerdere slaves zijn neemt ook de beschikbaarheid van de toepassing toe, ten minste voor wat betreft het lezen van gegevens: als één van de vele slaves ten gronde gaat is er altijd nog een pool van andere slaves van waaruit data gelezen kunnen worden. En door slaves over verschillende locaties te spreiden, neemt de beschikbaarheid nog meer toe: als één datacentrum in zijn geheel uitvalt, blijven de slaves bij het andere datacentrum ongemoeid.

De achilleshiel van de zojuist besproken replicatieclusters is natuurlijk de master. De master is slechts enkelvoudig aanwezig en vormt dus een bottleneck indien de applicatie veel schrijfacties uitvoert. Verder is de master een single point of failure, en op zijn minst een deel van de functionaliteit van de applicatie zal niet beschikbaar zijn indien de master (al dan niet gepland) wegvalt. Een zogenaamd multi-master replicatiecluster kan in deze gevallen soelaas bieden.

Multi-master replicatieclusters

Een multi-master replicatiecluster laat zich het gemakkelijkst begrijpen als twee separate replicatieclusters, waarbij de master van ieder cluster tevens ook slave is van de master van het andere cluster. Dit fenomeen wordt aangeduid met de term circulaire replicatie [7]. Dus de master van het ene cluster voedt behalve zijn slaves óók de master van het andere cluster. De master van het andere cluster voedt op zijn beurt ook weer zijn eigen slaves, maar ook de andere master, zie afbeelding 2. (Het lijkt op het eerste gezicht alsof de circulaire replicatie tussen de masters tot een oneindige loop zou leiden, maar dit wordt voorkomen doordat de events in de binaire log een nummer meekrijgen van de master waar het event oorspronkelijk vandaan komt, en een slave alleen events met een andere dan de eigen id verwerkt.)

Met de multi-master aanpak kan de applicatie voortaan uit meerdere masters kiezen. Ook kan er nu onderhoud aan de masters worden gepleegd zonder de beschikbaarheid van de applicatie te verliezen. Tot slot kunnen nu ook de masters in verschillende locaties worden geplaatst, waardoor de praktische beschikbaarheid dicht bij de 99,999 procent (five nines) komt, zonder daarvoor gespecialiseerde en vaak kostbare hardware aan te schaffen.

Overigens is een multi-master replicatiecluster niet in staat om eventuele bottlenecks in het schrijven van data op te lossen. Hoewel meerdere masters wel meer totale schijfcapaciteit voor de applicaties betekenen is de praktische schrijfcapaciteit toch nog steeds gebonden aan het vermogen van de slaves om die wijzigingen ook te verwerken. In het theoretische geval dat bij een verdubbeling van het aantal masters de applicaties tweemaal zoveel schrijfverzoeken kunnen afhandelen, zouden er per slave ineens ook tweemaal zoveel schrijfverzoeken moeten worden afgehandeld, en dit kan natuurlijk niet.

MySQL scale-up?

Voor de meer traditionele databasesystemen als Oracle en Microsoft SQL Server is een scale-out model zoals bij het MySQL replicatiecluster meestal niet aan de orde. Het spreekt bijna vanzelf dat een grootschalig replicatiecluster minder aantrekkelijk wordt indien ook de licentiekosten voor de database server en het besturingssysteem in de vergelijking meedoen. In de praktijk draaien scale-out architecturen dan ook volledig op open source software.

Maar ook voor MySQL is het opzetten van een replicatiecluster natuurlijk niet in alle gevallen voordelig. Het heeft alleen zin indien de kosten van het toevoegen en onderhouden van een

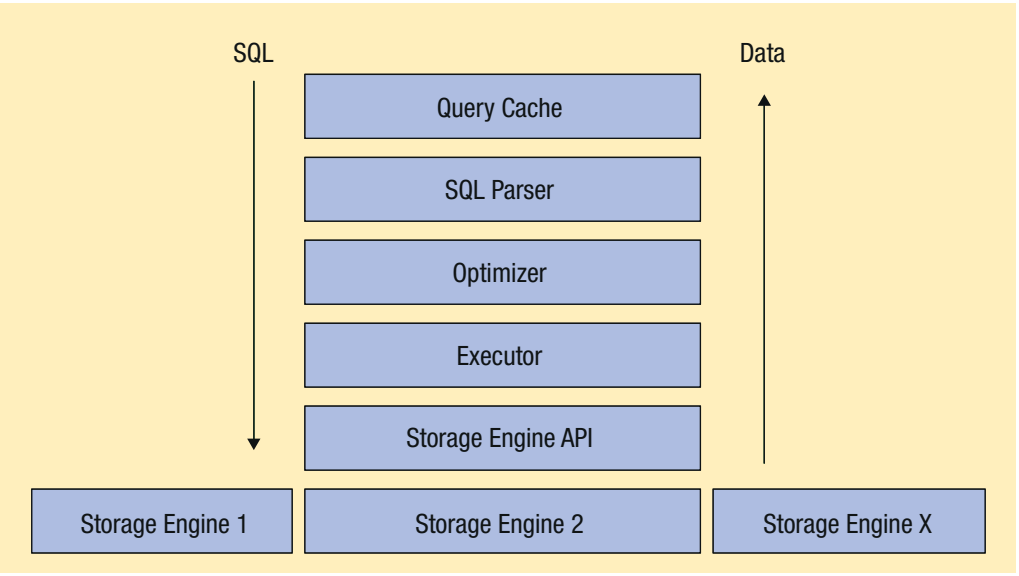
slave lager uitvallen dan het upgraden naar beter presterende (indien men de throughput wil verhogen) of intrinsiek redundante (indien men de beschikbaarheid wil verhogen) hardware van een enkelvoudige database server (scale-up).
Dit brengt ons op een andere reden voor het bouwen van MySQL scale-out clusters. MySQL zelf schaalt niet zo best op. Meer dan Oracle en MS SQL Server heeft MySQL moeite met het benutten van de beschikbare processorcapaciteit, terwijl de trend nu juist zo is dat die almaar toeneemt. De laatste jaren is dit een steeds vaker herhaald punt van kritiek op MySQL, juist vanuit de hoek van ervaren MySQL gebruikers.

MySQL serverarchitectuur

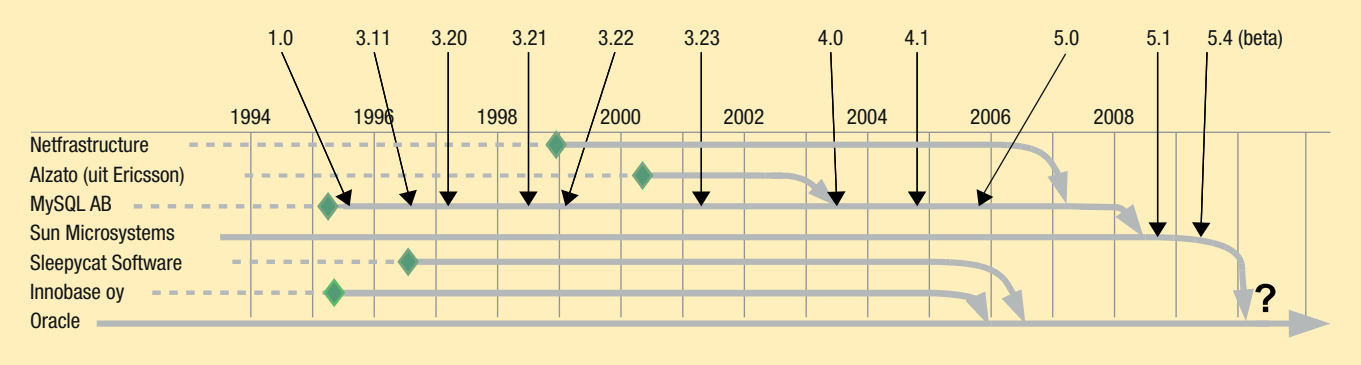
Afbeelding 3 toont een versimpelde schematische weergave van de MySQL serverarchitectuur. SQL statements worden ontvangen van verbonden clients, en worden waar mogelijk direct vanuit de query cache beantwoord. Indien het statement niet in cache staat, dan wordt het geparset en zo ontrafeld tot afzonderlijke tabel- en kolomnamen, sleutelwoorden en en operatoren. De optimizer ontvangt de symbolen van de parser en construeert daaruit het queryplan. Het queryplan is een lijst van te nemen acties om het queryresultaat te berekenen. Hierin staan instructies zoals 'data uit tabel ophalen', 'filteren', 'groeperen', 'joinen' etcetera. Alle bewerkingsoperaties in het plan worden uitgevoerd door de executor, die uiteindelijk daaruit het queryresultaat genereert.
De queryplan-instructies om data te op te slaan of te verkrijgen worden afgehandeld via de storage engine API. De functies die tezamen deze API vormen hebben zelf geen directe toegang tot de data. In plaats daarvan worden deze functieaanroepen afgehandeld door een specifieke storage engine, in het MySQL jargon ook bekend als 'handler'. De storage engine is de eigenlijke database, dat wil zeggen de programmatuur die interacteert met de datadrager en verantwoordelijk is voor het opslaan en terughalen ervan.

In een MySQL server kunnen meerdere verschillende storage engines tegelijkertijd gebruikt worden. Verschillende storage engines hebben verschillende eigenschappen voor wat betreft indicering, transacties, constraint afhandeling, data-opslagformaat (bijvoorbeeld de toepassing van compressie) en gegevensdrager (bijvoorbeeld disk, werkgeheugen of een remote host). Het idee is dat dit de ontwikkelaar en/of DBA in staat stelt om voor elk bepaald type data de best bijpassende storage engine te kiezen.
Om een idee te geven van de mogelijkheden volgt een korte beschrijving van MySQL's twee meestgebruikte storage engines:
MyISAM, ondersteunt geen transacties, en gebruikt table level locking. Deze engine is snel voor alleen het lezen van query's. MyISAM kan ook snel zijn voor schrijfquery's, maar door table locking gaat de performance snel achteruit naarmate de concurrency toeneemt. Verdere features: fulltext en spatial indexes. Voorbeeldtoepassingen: (kleine) datawarehouses, logging. Vanwege het ontbreken van transacties minder geschikt voor typische OLTP-doeleinden.
InnoDB, een engine met ondersteuning voor transacties op basis van multi-version concurrency control. Bij lage concurrency is er bij deze engine meer tuning vereist om dezelfde prestaties te krijgen als bij MyISAM, maar in het geval van meer concurrency blijft deze engine beter overeind. Wel vereist deze engine meer werkgeheugen en schijfruimte dan MyISAM. Verdere features: clustered en adaptive hash indexes en ondersteuning voor foreign key constraints. Voorbeeldtoepassing: OLTP.

Zoals blijkt zijn er nogal wat verschillen, en omdat niet alle features in alle engines aanwezig zijn is het soms moeilijk kiezen. Aan de andere kant is het idee om verschillende soorten data en verschillende workloads met daarop aangepaste storage engines aan te pakken zo gek nog niet. Neem bijvoorbeeld de net besproken techniek van replicatieclustering: voor de masters die met name het schrijfverkeer te verwerken krijgen is de InnoDB



Afbeelding 3:
MySQL architectuur.



Afbeelding 4: MySQL timeline.

engine bijvoorbeeld een betere keuze dan de MyISAM engine. Voor de slaves zal in veel gevallen juist de MyISAM engine weer beter zijn.
Overigens is er nog een belangrijk niet-functioneel verschil tussen deze engines: MyISAM is een origineel product afkomstig van de MySQL ontwikkelaars zelf. InnoDB niet: dit product werd oorspronkelijk geleverd door het Finse bedrijf InnoBase oy. Omdat InnoDB zelf ook open source is kan het zonder meer worden meegeleverd in de open source MySQL server. (Voor meelevering van InnoDB in de enterprise editie van de MySQL server wordt door Sun een fee betaald).
In 2005 is InnoBase oy overgenomen door Oracle. (Voor een overzicht van overnames rondom Sun, Oracle, MySQL en Innobase, zie afbeelding 4. Toentertijd werd gevreesd dat Oracle hiermee de competitie die het van MySQL ondervond wilde beteugelen, maar daar is weinig van te merken geweest: Oracle is InnoDB blijven leveren en heeft de license fee ervoor niet verhoogd. Tegelijkertijd is Oracle InnoDB verder blijven ontwikkelen, en heeft daaraan zelfs belangrijke features toegevoegd, zoals compressie en online wijzigbare indexes.

De toekomst met Oracle

Oracle heeft inmiddels duidelijk uitsluitsel gegeven voor wat betreft Sun's hardware-tak: Oracle heeft aangekondigd hierin meer te gaan investeren en de competitie met Sun's traditionele concurrent IBM voort te zetten [8]. En duidelijk bewijs daarvan is ook al geleverd: de tweede editie van Oracle's Exadata appliance is gebouwd met behulp van Sun hardware, met als belangrijkste nieuwe feature de flash-fire technology waardoor de performance voor zowel OLTP- als datawarehousingtoepassingen drastisch verbeterd wordt [9].
Oracle heeft zich evenwel nog niet uitgesproken over de toekomst die voor MySQL in het verschiet ligt na de overname. Vrijwel iedereen in de MySQL community is het er over eens dat in ieder geval de open source versie van MySQL server zal blijven voortbestaan. De algemene verwachting is dat zelfs als Oracle besluit het product niet verder te ontwikkelen, bedrijven zoals Monty Program ab (recentelijk opgericht door de oorspronkelijke ontwikkelaar van MySQL Michael 'Monty' Widenius) de ontwikkeling zullen overnemen. Maar goed bekeken is het logischer als Oracle juist doorgaat met MySQL te ontwikkelen. Via MySQL heeft Oracle een kans om actief deel te nemen aan de technologie waarop de meerderheid van het world wide web draait.

Minder zeker is de toekomst van de third-party leveranciers van storage engines en appliances. Dit zijn bedrijven die op commerciële basis MySQL uitrusten met een op een bepaald doeleinde toegesneden storage engine (en bijbehorende optimizer en executor), soms in combinatie met gespecialiseerde hardware. Voorbeelden hiervan zijn analytische closed source databaseproducten van Calpont, Infobright en Kickfire. Nu mogen dergelijke bedrijven een aangepaste closed source versie van MySQL leveren omdat zij daarvoor van Sun een licentie hebben. Het is zeer de vraag of Oracle ook van plan is om een dergelijke licentie te blijven verlenen. In ieder geval kan Oracle het deze vendors erg moeilijk maken en wat dat betreft is het onderzoek van de EU-commissie zo gek nog niet. De commissie heeft tot uiterlijk 19 januari 2010 de tijd om het onderzoek af te ronden, dus over deze zaak is het laatste woord nog niet gezegd.

Literatuur

1. *Mergers: Commission opens in-depth investigation into proposed takeover of Sun Microsystems by Oracle – http://europa.eu/rapid/pressReleasesAction.do?reference=IP/09/1271*
2. *MySQL Customers by Industry – www.mysql.com/customers/industry/*
3. *MySQL 5.4 Reference Manual, 7.5.5. The MySQL Query Cache – http://dev.mysql.com/doc/refman/5.1/en/query-cache.html*
4. *MySQL 5.4 Reference Manual, 16 – Replication – http://dev.mysql.com/doc/refman/5.4/en/replication.html*
5. *300 Million and On – http://blog.facebook.com/blog.php?post=136782277130*
6. *Scaling out, notities van Facebook ontwikkelaar Jason Sobel – www.facebook.com/note.php?note_id=23844338919*
7. *Advanced MySQL Replication Techniques, door Giuseppe Maxia, http://onlamp.com/pub/a/onlamp/2006/04/20/advanced-mysql-replication.html*
8. *Sun Customers – www.oracle.com/features/suncustomers.html*
9. *Exadata – www.oracle.com/database/exadata.html*

Roland Philippe Bouman Msc. is webapplicatie- en BI-ontwikkelaar voor Strukton Rail en auteur van 'Pentaho Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL'.