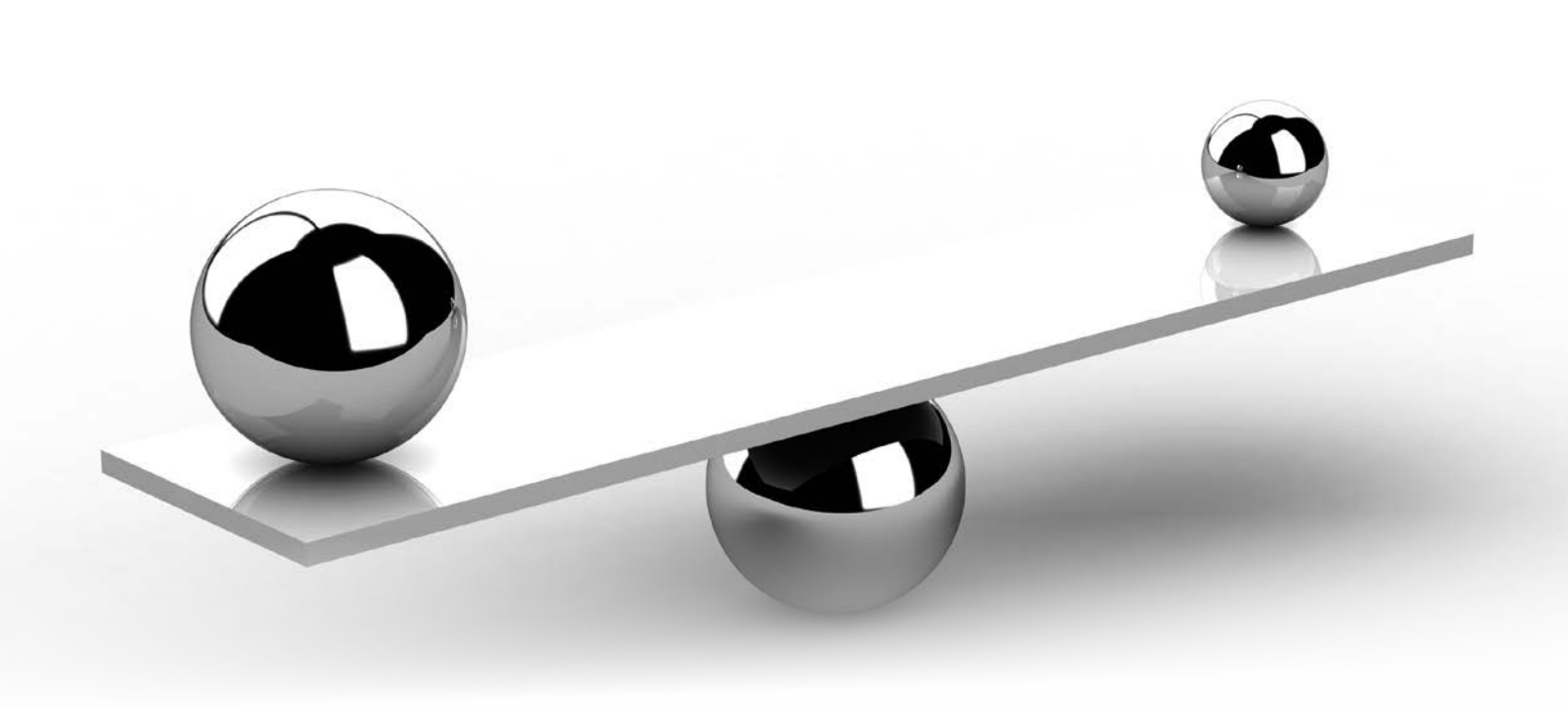

MySQL vs. Oracle-Datenbank

Oliver Sennhauser und Matthias Jung, DOAG Datenbank Community

MySQL und die Oracle-Datenbank sind zwei relationale Datenbanken aus dem Haus Oracle, die beide von sich behaupten, die ACID-Anforderungen zu erfüllen. Sie liegen beim „DB Engine Ranking" *(siehe „http://db-engines.com/en/ranking")*, das weltweit alle wichtigen Datenbank-Systeme erfasst, zurzeit auf Platz eins und zwei. MySQL wird voraussichtlich in den nächsten zwölf Monaten die Oracle-Datenbank vom ersten Platz verdrängen. Der Artikel zeigt die Unterschiede, Einsatzbereiche und die Vorzüge des jeweiligen Produkts.

MySQL wurde in den 1990er-Jahren zu Beginn des Web-Booms als SQL-Layer um C-ISAM-Dateien entwickelt. Es war um die Jahrtausendwende eine relativ einfache und frei verfügbare Datenbank, die in zahlreiche neu aufkommende Web-Anwendungen integriert wurde und heute als eine der Haupt-Komponenten des sogenannten „LAMP-Stacks" (Linux, Apache, MySQL, PHP) gilt. Daher ist MySQL aus der Web-Welt nicht mehr wegzudenken. Ohne MySQL würden sich die meisten Angebote des Internets quasi in nichts auflösen.

Die Oracle-Datenbank ist in den 1970er-Jahren in den USA entstanden und kommt seitdem in zahlreichen Unternehmen für geschäftskritische Prozesse zum Einsatz. Daher ist dieses Produkt meist eng mit den Geschäftsprozessen verflochten und oft nur schwer daraus herauszulösen.

Eigenschaften

Beide Datenbanken bezeichnen sich als relationales Datenbank Management System (RDBMS), wobei beide auch über den Tellerrand hinausschauen und versuchen, im Umfeld der relationalen Welt weitere Funktionalitäten abzudecken.

MySQL bietet mit seinem Storage-Engine-API zahlreiche Möglichkeiten, um die Funktionalität des Produkts zu erweitern. Dies führt dazu, dass wir heute die Wahl haben zwischen einer transaktionalen, Crash-sicheren, relationalen Storage-Engine namens „InnoDB", einer Storage-Engine, die Daten ausschließlich im Speicher ablegt („In-Memory"), einer spaltenorientierten Storage-Engine („Column Store"), einer Graphen-Storage-Engine und einer Storage-Engine, die für das „Sharding" zuständig ist. Auf der anderen Seite verfolgt die Oracle-Datenbank eher den Ansatz von Integration. XML und OODBMS sind in das Datenbanksystem integriert; Java läuft im Datenbank-Kernel; In-Memory und Column Stores sind in das Produkt integriert. Die Qual der Wahl hat man eher nicht, man darf oder muss nehmen, was der Hersteller für einen als gut befunden hat. Dies kann ein Vor- oder auch ein Nachteil sein. Von Nachteil ist jedoch die komplizierte Lizenzpolitik seitens Oracle, die zusätzlich beim Einsatz bestimmter Funktionen der Oracle-Datenbank zu betrachten ist.

Umfang

MySQL hat aus der Not eine Tugend gemacht und verfolgt recht stark das KISS-Prinzip (Keep It Stupid and Simple). Das heißt in der Praxis, dass sich in MySQL al-

les recht einfach, schnell und schnörkellos gestaltet. Dafür hat man nicht denselben Funktionsumfang und dieselben Möglichkeiten wie bei der Oracle-Datenbank und muss sich gegebenenfalls selbst einen Weg überlegen, um eine vergleichbare Lösung zu implementieren. Das beginnt schon mit der Installation: Jede gängige Linux-Distribution bietet von Haus aus eine MySQL-Distribution an. Die Installation erfolgt oft über das Packet-Management-System der Distribution. Alternativ lässt sich auch das Repository von Oracle einbinden. Eine Installation von MySQL benötigt in der Regel genau einen Befehl wie „yum install mysql-server". In zwei bis drei Minuten ist ein lauffähiges MySQL-System verfügbar, das je nach Bedarf noch weiter konfiguriert werden kann.

Eine Oracle-Datenbank-Installation ist meist deutlich aufwändiger und beginnt mit dem Download von mehrere GB umfassenden Software-Archiven. Im Anschluss daran muss die Software auf den entsprechenden Server kopiert und der Oracle Universal Installer (OUI) aufgerufen werden. Über mehrere Dialog-Seiten hinweg werden Parameter zur Installation der Software und – sofern gewünscht – zum direkten weiteren Aufbau einer Datenbank ermittelt. Natürlich lässt sich dieser Prozess auch automatisieren („Silent Installation"). Allerdings bedarf es hier zur Vorbereitung einer gewissen Einarbeitungszeit, um sich mit den Mechanismen dieses Verfahrens vertraut zu machen.

Wie auch die Oracle-Datenbank unterstützt MySQL die gängigsten Plattformen wie Oracle Linux, Red Hat Enterprise Linux, SuSE Enterprise Linux, Ubuntu Server und natürlich auch Windows. Auf den Linux-Plattformen wird man am wenigsten auf Probleme stoßen und Bugs sowie entsprechende Workarounds dazu finden. Da MySQL die Pakete regelmäßig und automatisiert baut, kommen die Versionen für alle unterstützten Plattformen praktisch zeitgleich heraus. Ein neues Minor-Release wie MySQL 5.7.17 (in der Oracle-Sprache „Patch") kann etwa alle sechs bis acht Wochen erwartet werden, ein Major-Release (wie MySQL 8.0) etwa alle achtzehn Monate. Aktuelle Versionen bei MySQL sind zurzeit MySQL 5.6 und 5.7. Die nächste Version wird MySQL 8.0 heißen und ist bereits als Beta-Release verfügbar. Die Freigabe dieser neuen Version wird wahrscheinlich im Jahr 2017 für den produktiven Einsatz erfolgen.

Wie auch die Oracle-Datenbank verfügt MySQL über eine Konfigurationsdatei namens „my.cnf" (auf Windows „my.ini"). Diese liegt üblicherweise unter „/etc" (Red Hat und SuSE) oder „/etc/mysql" (Ubuntu und Debian). Sie kann aber auch an einer beliebigen anderen Stelle liegen. In diesem Fall ist einfach dem Start-Prozess mitzuteilen, wo er sie findet.

Im praktischen Einsatz

Das Starten und Stoppen einer MySQL-Instanz erfolgt über den Betriebssystem-eigenen Mechanismus („init"-Prozess). Zurzeit sind die Linux-Distributionen im Übergang von SytemV/Upstart nach SystemD. Es herrscht daher noch etwas Verwirrung, wie es richtig geht, und es hat sich möglicherweise die eine oder andere unsaubere Implementierung eingeschlichen. Dies sollte aber nur ein vorübergehendes Problem während dieser Übergangsphase sein. Multi-Instanzen-Set-ups sind in der Oracle-Welt üblich. Dies ist unter anderem auf die Lizenz-Politik von Oracle zurückzuführen, da die Lizenzierung auf Sockel und Kernen basiert. Bei MySQL fallen in den meisten Fällen sowieso keine Lizenz-, sondern höchstens Support-Subskriptions-Kosten an. Von daher wird MySQL typischerweise nicht als Multi-Instanzen-Set-up installiert. Im Bedarfsfall kann man MySQL natürlich auch in diesem Modus betreiben.

Was für den Oracle-DBA „SQL*plus", ist für den MySQL-DBA „mysql". Viele Administratoren arbeiten gerne mit dem Kommandozeilen-Interface (CLI). Im Vergleich zur Oracle-Datenbank ist das MySQL-CLI wesentlich weniger mächtig. Dies ist unter anderem darauf zurückzuführen, dass zu der Zeit, als MySQL entwickelt wurde, der Bedarf nach starken CLIs nicht mehr so groß war, weil grafische Benutzeroberflächen schon weitverbreitet waren. Im Unterschied dazu war man in den Anfängen der Oracle-Datenbank noch auf eine gute CLI zwingend angewiesen.

Die grafische Oberfläche der Wahl ist bei MySQL heute die MySQL-Workbench. Sie beinhaltet Werkzeuge für den Admin auf der einen, für den Entwickler auf der anderen Seite. Zusätzlich bietet dieses Toolset einen recht brauchbaren ER-Diagrammer sowie ein Tool für die Daten-Migration von anderen Datenbank-Systemen wie Oracle. Sämtliche Anwendungen, die eine ODBC- oder JDBC-Schnittstelle nutzen, sollten in der Lage sein, mit MySQL zu kommunizieren und Daten zu verarbeiten. Sogar der SQL Developer von Oracle kann mit MySQL arbeiten.

Wenn man sich mit der MySQL-Datenbank verbunden hat, wird der Oracle-DBA sehr wahrscheinlich etwas verwirrt sein. Sowohl die User als auch die Schemata und Datenbank-Objekte sowie deren Beziehungen zum User sind anders, als dies bei der Oracle-Datenbank implementiert wurde. Ein Account setzt sich immer aus einem User-Namen und einem Host oder einer Host-Range zusammen, von wo aus sich ein User verbinden darf. Das kann dazu führen, dass es zwei User mit demselben Namen in der Datenbank geben kann, etwa „oli@localhost" und „oli@192.168.%". Diese können zu allem Überdruss zusätzlich noch unterschiedliche Passwörter und Privilegien haben.

Bei der Oracle-Datenbank entsteht ein Schema mit dem Kreieren eines Objekts, das einem User gehört. Der User darf als Besitzer des Objekts dieses vollständig verwalten. Bei MySQL ist das anders: Ein Objekt (Tabelle, View, Trigger etc.) gehört immer dem System. Ein User kann nur Rechte haben, mit dem Objekt bestimmte Aktionen durchzuführen (EXECUTE, INSERT, SELECT etc.). Rollen wie bei der Oracle-Datenbank kennt MySQL zurzeit noch nicht. Das ist allerdings nicht weiter tragisch, da heute Rollen-Konzepte üblicherweise in der Applikation und nicht wie früher üblich in der Datenbank abgebildet sind. Mit der Version 8.0 wird es in naher Zukunft jedoch auch Rollen in MySQL geben.

Eine Oracle-Datenbank ist mit mehr Objekt-Typen ausgestattet als MySQL. Dort kennt man nur Tabellen, Indizes, Views und Stored Programs (Procedures, Functions, Triggers, Events), wobei man Letztere heute tendenziell eher nicht mehr verwendet, da Business-Logik heute im Business-Layer angesiedelt ist und weniger häufig im Daten-Layer (DBMS), wie das früher noch häufig der Fall war und auch von den Datenbank-Herstellern stark gefördert wurde. Wenn ein Oracle-PL/SQL-Entwickler Stored Programs für MySQL erstellen soll, wird er voraussichtlich etwas enttäuscht sein. Der Funktionsumfang, die Integration in zahlreiche Entwicklungs-Umgebungen, die Debugging-Möglichkeiten und vor allem auch das intuitive Arbeiten wird man bei MySQL

stark vermissen. Wie bereits gesagt, heute entwickelt man Software anders als früher. Eine stark integrierte Sprache ist heute nicht mehr zwingend erforderlich und bindet Business-Prozesse unnötig an dedizierte Datenbank-Systeme.

Bei den Daten-Typen kennt MySQL alle wichtigen Basiselemente wie „INT" (klein bis groß), „FLOAT", „DECIMAL", „VARCHAR", „DATE", „TIME", „DATETIME", „BLOB", „TEXT" (= CLOB) etc. Komplexe oder zusammengesetzte Datentypen wie bei der Oracle-Datenbank kennt MySQL derzeit nicht. Neue Datentypen wie „GIS" (POINT, POLYGON etc.) oder „JSON" gehen zwar in diese Richtung, sind aber eher aus Sicht der Applikation und nicht von der Datenbank aus zu betrachten.

Da MySQL schon früh weltweite Verbreitung fand und auch auf Feedback von weit entfernten Kulturen einging, ist die Verwendung verschiedener Character-Sets wie UTF8, latin1 oder auch ein Mischen von Sets kein Problem. MySQL kann das Character-Set auf Tabellen- oder sogar auf Spalten-Ebene festlegen. Ein nachträgliches Ändern des Character-Sets auf Instanz, Schema, Tabelle oder Spalte ist kein Hexenwerk.

Wie die Oracle-Datenbank kennt MySQL inzwischen auch das Konzept von Tablespaces. Aus verschiedenen Gründen ist dieses allerdings nicht so ausgefeilt wie bei der Oracle-Datenbank, was den Vorteil bietet, dass es im Handling recht einfach ist. MySQL kann entweder alle Tabellen in einem Tablespace ablegen (System Tablespace, ähnlich wie es ganz früher auch bei der Oracle-Datenbank der Fall war), oder aber jede Tabelle hat ihren eigenen Tablespace. Das führt natürlich zu Problemen, wenn man Instanzen mit Zehntausenden bis Millionen von Tabellen hat. Seit Kurzem kennt MySQL daher auch das Konzept von allgemeinen Tablespaces („General Tablespaces"). Darin lassen sich, wie von der Oracle-Datenbank gewohnt, Tabellen nach Wunsch ablegen. Das hat auf der einen Seite den Vorteil von weniger Dateien im Filesystem, auf der anderen Seite aber den Nachteil, dass man sich Gedanken darüber machen muss, wo die Tabellen abgelegt werden sollen.

Auf physischer Ebene verwendet MySQL das Filesystem des Betriebssystems. Eine entsprechende Zwischenschicht wie Oracle ASM gibt es nicht. Das kommt dem Linux-Administrator entgegen, der neben MySQL noch zahlreiche andere Anwendungen zu betreuen hat und sich nicht in solche Spezialitäten einarbeiten muss (KISS). Der Spatial-Option der Oracle-Datenbank entsprechen bei MySQL die GIS-Datentypen und -Indizes (zweidimensionale Indizes). In MySQL sind diese OpenGIS standardkonform und für zahlreiche neue Anwendungen nutzbar, die dank OpenStreetMap und Google Maps sowie der mobilen Devices mit Standort-Angaben möglich werden. So lassen sich Abfragen wie „Welche Restaurants liegen im Umkreis von hundert Metern von meiner Position und haben jetzt gerade geöffnet?" problemlos verarbeiten.

Was die Oracle-Datenbank mit seiner Text-Option anbietet, entspricht der MySQL-Volltext-Suche mit Volltext-Indizes. Dies ist eine recht praktische Möglichkeit, ganze Texte von kleinerer bis mittlerer Größe in der Datenbank zu durchsuchen. Bei großen Textmengen im Bereich von Tera- bis PetaByte weicht man besser auf eine spezialisierte Lösung aus, etwa das heute moderne ElasticSearch. Vor rund fünfzehn Jahren wurde XML populär. Oracle hat diese Strömung stark unterstützt und mit der XML-Option in die Datenbank integriert (heute Standard mit 12*c*). Es hat sich aber, bedingt durch seine Komplexität, nicht in allen Bereich gleichermaßen durchgesetzt. MySQL unterstützt ebenfalls marginale XML-Funktionalitäten. Zielführender und zukunftsweisender ist jedoch der neue JSON-Trend, der zahlreiche XML-Einsatzgebiete effizienter abdecken kann – angefangen bei einem eigenen Datentyp „JSON" und zahlreichen Funktionen, um die Suche und die Manipulation von JSON-Dokumenten zu erleichtern und auch stark zu beschleunigen. Mit der Einführung von JSON in der Version 5.7 ist MySQL jetzt also auch zum Document-Store geworden. Dies stößt aktuell auf eine sehr gute Resonanz in der Oracle-Entwickler-Gemeinde bei der Neuentwicklung von Projekten. Die Oracle-Datenbank hat mit ihrer neuesten Version ebenfalls im Bereich von JSON Fortschritte erzielt und unterstützt auch diesen neuen Trend.

Wie bereits erwähnt, bietet MySQL mit seinen vielseitigen Plug-in-APIs zahlreiche Möglichkeiten für Erweiterungen. Dies sind einerseits Storage-Engines zum Ablegen der Daten in unterschiedlichen Formaten oder Speicherformen. Das Volltext-Parse-Plug-in bietet die Möglichkeit, Volltextsuchen für westliche und asiatische Schriften zu implementieren. Die Auditing-, Authentication- und Password-Validation-APIs erlauben hier die Anbindung an sämtliche verfügbaren externen Quellen und das Query-Rewrite-API ermöglicht es, Plug-ins zur Manipulation von Queries zu erstellen, wenn sich zum Beispiel die alte Applikation nicht an das neue Datenbank-Verhalten adaptieren lässt.

Datensicherheit

Im Bereich „Backup/Restore" hat MySQL dem RMAN der Oracle-Datenbank nichts Gleichwertiges entgegenzusetzen. Zwar schreibt MySQL sogenannte „Binary Logs", die in etwa den Oracle-Archive-Logs entsprechen, und hat ebenfalls eine physische Backup-Methode. Diese entspricht aber eher den alten Oracle-Tablespaces.

Für kleinere Datenbanken empfiehlt es sich, ein logisches Backup mit dem Werkzeug „mysqldump" zu erstellen. Dieses ähnelt „exp/imp" bei der Oracle-Datenbank. Im Unterschied dazu kann jedoch auf einem mit „mysqldump" erzeugten Backup ein Point-in-Time-Recovery mit den Binary-Logs erfolgen. Allerdings muss der DBA selbst dafür sorgen, dass er die richtige Position des benötigten Binary-Log erwischt – ansonsten können beim PiTR Lücken mit fehlenden Daten entstehen. Mit dieser Methode können praktisch statementgenau Recoveries erfolgen, wenn der Admin das genaue Statement beziehungsweise den Zeitpunkt (Log-Position des Statements) kennt. Bei größeren Datenmengen ist ein logisches Backup mittels „mysqldump" nicht mehr das adäquate Mittel der Wahl. In diesem Fall empfiehlt es sich, die physische Backup-Methode namens „MySQL Backup" zu verwenden. Damit lassen sich Datenbanken bis in den Terabyte-Bereich sichern, sofern die darunterliegende Infrastruktur auch für diese Datenmengen ausgelegt ist. Nach dem erfolgten Restore findet, wie bereits bei der logischen Backup-Methode, ein Point-in-Time-Recovery mithilfe der MySQL-Binary-Logs statt. Auch in diesem Fall gilt es wieder, die richtige Position im richtigen Log-File zu finden. Diese kann in der Regel jedoch – bei sorgsamer Vorbereitung – problemlos ermittelt werden.

Hochverfügbarkeit

Bei den Hochverfügbarkeits-Lösungen kann sich MySQL durchaus mit der

Oracle-Datenbank messen. Failover-Cluster lassen sich bei MySQL genauso implementieren, da diese hauptsächlich durch Betriebssystem-Mittel beziehungsweise durch eine entsprechende Cluster-Software abgedeckt sind. Es gilt nur, die richtigen Module zu finden und MySQL entsprechend richtig mit diesen zu integrieren. Failover-Cluster sind jedoch tendenziell eine eher veraltete Technologie. Heute löst man Hochverfügbarkeit mit MySQL durch den Ansatz der Replikation. Analog zu Oracle Data Guard kennt MySQL die sogenannte „Master/Slave-Replikation". Dies klingt zwar weniger spektakulär, ist aber ähnlich: Die Applikation schreibt in den aktiven Master-Knoten, der wiederum im Streaming-Verfahren die ganzen Daten-Änderungen auf den Slave repliziert. Dort werden diese abgearbeitet und auf die Datenbank appliziert. Im Unterschied zur Oracle-Datenbank können die Daten auf dem Slave-System gelesen und auch geschrieben werden. Letzteres sollte allerdings unterlassen werden, wenn die Daten-Konsistenz der Replikation gewährleistet werden soll.

Der Replikationsvorgang in der MySQL-Master/Slave-Replikation ist asynchron. Im Falle eines Hardware-Ausfalls kann es daher sein, dass Transaktionen auf dem Master bereits committet wurden, auf dem Slave hingegen noch nicht eingetroffen sind. Im schlimmsten Fall würde dies bedeuten, dass einige Transaktionen verloren gehen können. Wenn dies unter gar keinen Umständen geschehen darf, bietet MySQL die semi-synchrone Replikation. Im Unterschied zur asynchronen Replikation wird hier die Transaktion erst als vollständig akzeptiert, wenn mindestens ein Slave deren Erhalt bestätigt hat. Dies hat den großen Vorteil, dass bei einem Hardware-Ausfall garantiert ist, dass kein Datenverlust mehr entstehen kann. Es hat aber auch den Nachteil, dass der Commit verzögert wird, bis mindestens ein Knoten die Transaktion bestätigt hat, was zu höheren Commit-Latenzen führt und damit einen Einfluss auf die Performance der Applikation hat.

Eine Master/Slave-Replikation hat gewisse operative Nachteile, insbesondere bei Upgrades und sonstigen Wartungsarbeiten, denn die Rollen von Master und Slave müssen wechseln. Das ist zwar kein Hexenwerk, erfordert jedoch eine gewisse Konzentration seitens des DBA, da ansonsten unnötig lange Unterbrechungen oder gar Fehler auftreten können.

Der genannte Rollentausch wird mit dem virtuell synchronen Multi-Master-Galera-Cluster für MySQL vermieden. Dabei kann gleichzeitig auf alle drei Knoten (eine ungerade Anzahl größer eins) geschrieben und von allen Knoten gelesen werden. Es ist daher ein voll symmetrischer Cluster mit mehreren gleichwertigen Knoten. Mit Galera-Cluster für MySQL lässt sich im laufenden Betrieb ein Knoten aus dem Cluster entfernen. Auf diesem Knoten können dann Wartungsarbeiten erfolgen. Anschließend wird der Knoten wieder in den Cluster eingefügt und synchronisiert sich selbstständig. Fortlaufend wird derselbe Vorgang mit den weiteren Knoten im Cluster durchgeführt. Diese Operation nennt man auch „Rolling-Cluster-Restart". Der Datenbank-Cluster lässt sich also wartungsfensterfrei im laufenden Betrieb pflegen. Das Äquivalent dazu ist der Real Application Cluster (RAC) bei der Oracle-Datenbank.

Beim Monitoring für MySQL gibt es zwei Ansätze: Entweder verwendet man den MySQL-hauseigenen MySQL Enterprise Monitor, der wohl eine der häufigsten Monitoring-Lösungen für MySQL ist, oder das Oracle Cloud Control Plug-in für MySQL. Dieses ist noch wesentlich weniger mächtig als die MySQL-Monitoring-Lösung. Über kurz oder lang sollen wohl Pläne im Hause Oracle bestehen, den vollen Funktionsumfang vom MySQL Enterprise Monitor in Cloud Control zu integrieren. Das wird allerdings wohl noch eine Zeit dauern, was verständlich ist, wenn man die Mächtigkeit und die Spezifika des MySQL Enterprise Monitor näher betrachtet.

Einsatzgebiet der Oracle-Datenbank und von MySQL

Welche Datenbank soll man wann und wo einsetzen? Dies ist eine schwere Frage, die man praktisch selten mit technischen Stärken oder Schwächen beantworten kann. MySQL ist sehr stark im Web-Umfeld verankert und lässt sich wohl nur schwer aus dem LAMP-Stack durch eine Oracle-Datenbank substituieren. Hat man auf der anderen Seite eine Anwendung, die nur mit einem spezifischen Datenbank-Backend funktioniert, ist eigentlich bereits klar, welche Lösung einzusetzen ist. Daher empfiehlt es sich, als Erstes auf die Empfehlung des Herstellers der Applikation zu hören. Weitere Fragen sind im Bereich der Kosten zu klären. Lohnt sich der Kostenaufwand für den Betrieb einer Oracle-Datenbank bei der Wertschöpfung des zu betreibenden Dienstes? Weitere wichtige Punkte sind: Gibt es bereits Know-how in der entsprechenden Technologie beziehungsweise wie schnell kann man dieses aufbauen (DBA und Entwickler) und wie teuer kommt es, sich dieses Know-how zu generieren oder einzukaufen? Darüber hinaus kommen natürlich auch politische Punkte zum Tragen: Ist eine Anwendung vollständig in Oracle-PL/SQL geschrieben, so ist der Aufwand enorm, diese Anwendung in eine andere Sprache auf eine andere Datenbank zu portieren. Wir haben hier den perfekten Vendor-Lock-in.

Eine Studie, die bereits vor etlichen Jahren durchgeführt wurde, hat ergeben, dass etwa 75 bis 80 Prozent der Anwendungen aufgrund der technischen Anforderungen mit MySQL hätten gelöst werden können. Die Probleme liegen also meist nicht im technischen Bereich, sondern im Umfeld. Eine technische Einschränkung besteht mit MySQL allerdings: Der Einsatz im zwei- oder gar dreistelligen Terabyte-Bereich ist mit MySQL schwierig und gewisse Probleme sind nicht mehr mit vernünftigem Aufwand lösbar. Aber bei solchen Größen dürfte auch die klassische Oracle-Datenbank Schwierigkeiten bekommen …

Oliver Sennhauser
oliver.sennhauser@doag.org

Matthias Jung
matthias.jung@doag.org