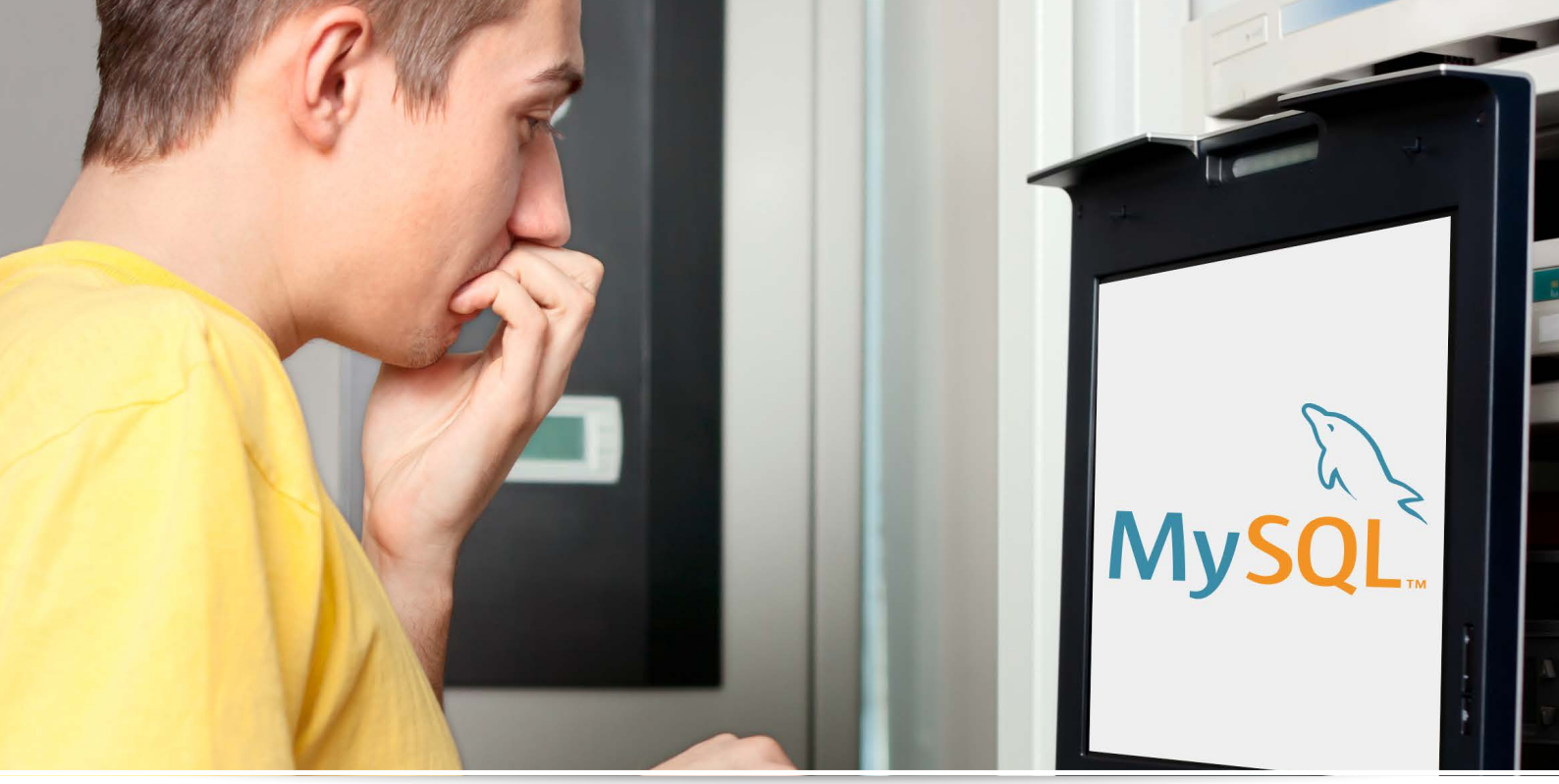

MySQL für Oracle-DBAs

Oli Sennhauser, FromDual GmbH

In den letzten Monaten wurde der Autor vermehrt von Verantwortlichen für das Datenbank-Team kontaktiert, die MySQL-Datenbanken quasi aufs Auge gedrückt bekommen haben. Ihre Mitarbeiter sind ausgebildete und erfahrene Oracle-DBAs, die zukünftig auch MySQL-Datenbanken betreiben sollen. Sie kennen ihre Datenbanken zwar aus dem Effeff, sind bei MySQL-Datenbanken dann aber doch etwas ratlos. Der Artikel zeigt, wie man MySQL-Datenbanken professionell und hochverfügbar installiert und betreibt.

Oft wird MySQL durch die Applikation getrieben als Datenbank-Backend eingesetzt. Diese Applikationen entwickeln sich anfangs meist außerhalb des Fokus der IT-Strategen, werden dann plötzlich wichtig oder sogar geschäftskritisch. Das IT-Management kommt zu der Einsicht, dass diese Datenbank jetzt in professionelle, betriebliche Hände gehört. Was liegt da näher, als bestehende DBAs mit dieser Aufgabe zu betreuen. Es ist ja eine Datenbank und das können die DBAs ja. Eh man sich versieht, ist man verantwortlicher DBA von MySQL-Datenbanken.

Die betroffenen DBAs brauchen sich keine Sorgen bezüglich der Zuverlässigkeit von MySQL zu machen. MySQL ist eine transaktionsfähige Datenbank. Commit und Rollback im Fehlerfall sind mit MySQL genauso möglich wie bei einer Oracle-Datenbank. Auch bezüglich Hochverfügbarkeit und Leistungsfähigkeit kann MySQL mit anderen Produkten locker mithalten, sodass Datenbanken im Terabyte-Bereich mit einer Hauptspeichernutzung von 512 GB sowie Dutzenden von CPU-Cores im Einsatz sind.

Installation von MySQL

MySQL ist eine Open-Source-Datenbank und im Open-Source-Umfeld gibt es meist mehrere Wege, die zum Ziel führen. So auch bei der Installation von MySQL. Zuerst stellt sich die Frage: Community Edition oder Enterprise Edition? Technisch gibt es zwischen diesen beiden Varianten keinen Unterschied. Somit ist es dem persönlichen Geschmack jedes DBAs oder der Firmenpolitik überlassen, was eingesetzt wird. Die Community Edition wird auf jeden Fall breiter eingesetzt und Support für MySQL kann für beide Varianten von diversen Anbietern bezogen werden. Beide MySQL-Editionen können unter MySQL-Downloads *[1]* heruntergeladen werden. Hat man die Edition gewählt, besteht die nächste Entscheidung darin, auf welche Art und Weise MySQL installiert werden soll. Es stehen verschiedene Möglichkeiten zur Auswahl.

Am häufigsten werden RPM- oder Debian-Pakete eingesetzt, die mit der Distribution mitgeliefert werden. Diese Methode ist recht einfach und die Distribution stellt sicher, dass das Datenbank-Paket sauber in die Distribution integriert ist und mit allen Applikationen, die eine MySQL-Datenbank benötigen, reibungslos funktioniert. Der größte Nachteil bei der Wahl dieser Pakete ist, dass die Versionen im Stand oft etwas hinterherhinken und die allerneu-

este Version in der aktuellen Distribution noch nicht zur Verfügung steht.

Eine weitere Möglichkeit besteht darin, RPM- oder Debian-Pakete zu verwenden, die der Software-Hersteller oder Drittanbieter bereitstellen. Der Vorteil dieser Methode liegt darin, dass immer Pakete der neusten MySQL-Version vorhanden sind. Zudem kann man mit Hersteller-Paketen zu Testzwecken auch auf ein Beta-Release zugreifen. Der Nachteil bei den Hersteller-Paketen ist, dass sich diese nicht immer ganz reibungslos in die Distribution integrieren und gegebenenfalls zu Konflikten oder nicht sauber aufzulösenden Abhängigkeiten mit den übrigen Distributionspaketen führen können. Soll die MySQL-Datenbank aber ausschließlich als Backend für eine selbst geschriebene Applikation dienen, ist dies kein Problem. Zudem bieten verständlicherweise nicht alle Support-Anbieter für alle Drittanbieter-Pakete Support an.

In manchen Fällen, etwa bei Multi-Instanz-Setups mit unterschiedlichen MySQL-Versionen oder wenn ganz kurze Downtimes bei Upgrades gefordert werden, kann die Installation von MySQL über generische binäre „Tar-Balls“ („.tar.gz“) erfolgen. Bei dieser Installationsart ist die Interaktion mit der Distribution minimal, aber leider auch die Integration. Oft muss manuell das eine oder andere nachkonfiguriert werden. Diese Variante der MySQL-Installation ist eher für Spezialfälle gedacht.

Für die ganz ausgekochten DBAs besteht schließlich noch die Möglichkeit, MySQL aus den Quellen selber zu kompilieren – was bei einer Oracle-Datenbank unvorstellbar ist. Diese Variante der MySQL-Installation wird heute aber nur noch in seltenen Fällen gebraucht, zum Beispiel wenn man selber MySQL-Bugs fixen oder spezielle Patches von Drittanbietern in MySQL integrieren möchte. Nichtsdestotrotz lohnt es sich aus Verständnisgründen, diese Variante einmal durchzuprobieren. Die MySQL-Installation ist für die folgenden drei wichtigsten Linux-Distributionen möglich:

- *CentOS/RedHat/RHEL*
  shell> yum install mysql-server
- *Ubuntu/Debian*
  shell> apt-get install mysql-server
- *OpenSuSE/SLES*
  shell> zypper install mysql-community-server

Starten und Stoppen der MySQL-Instanz

Das Starten und Stoppen der MySQL-Instanz erfolgt üblicherweise durch den Betriebssystem-eigenen Start-/Stopp-Mechanismus („init“, „systemd“, „upstart“). Die Distribution sorgt dafür, dass beim Installieren des Pakets die Datenbank richtig in den Start-/Stopp-Mechanismus eingehängt wird.

Ebenfalls stellt die Distribution sicher, dass entweder beim Installieren des MySQL-Servers oder aber beim ersten Start der Instanz eine Datenbank auf der Default-Lokation („/var/lib/mysql“) angelegt wird. Diese Lokation kann natürlich nachträglich durch Verändern der MySQL-Konfiguration an jede beliebige Stelle des Filesystems verlegt werden. Leider heißt der Service, unter dem MySQL läuft, je nach Distribution leicht unterschiedlich:

- *CentOS/RedHat/RHEL*
  mysqld
- *Ubuntu/Debian*
  mysql
- *OpenSuSE/SLES*
  mysql

Der Befehl für das Starten und Stoppen des MySQL-Service lautet wie folgt *(siehe Listing 1)* oder etwas altbacken *(siehe Listing 2)*.

Diese Befehle bewirken, dass eine MySQL-Instanz gestartet oder gestoppt wird. Ob diese auch wirklich läuft, lässt sich mit „shell> service mysql status“ oder „shell> /etc/init.d/mysql status“ beziehungsweise mit Betriebssystem-Befehlen prüfen *(siehe Listing 3)*.

Hierbei wird man feststellen, dass es zwei MySQL-Prozesse gibt, den eigentlichen „mysqld“- und den „mysqld_safe“-Prozess. Letzterer wird auch als „Angel-Prozess“ bezeichnet. Er dient dazu, den „mysqld“-Prozess wieder zu starten, sollte sich dieser mal unerwarteterweise beenden. Der eigentliche Datenbank-Prozess ist jedoch der „mysqld“-Prozess.

Ob das Starten und Stoppen der MySQL-Instanz fehlerfrei funktioniert, kann im sogenannten „MySQL Error Log“ überprüft werden. Das MySQL Error Log entspricht in etwa dem „Oracle Alert Log“. Dieses liegt je nach gewählter Installationsmethode und Konfiguration an unterschiedlichen Orten und hat unterschiedliche Namen:

- *CentOS*
  „/var/lib/mysql/<hostname>.err“
- *Ubuntu*
  „/var/log/mysql/error.log“ und/oder „/var/log/syslog“
- *OpenSuSE*
  „/var/lib/mysql/<hostname>.err“

Datenbank-Clients

Wie auch bei Oracle gibt es bei MySQL ein Kommandozeilen-Interface (CLI) namens „mysql“. Mit diesem kann man sich gegen die MySQL-Datenbank verbinden. Eine MySQL-Instanz ist immer eindeutig durch die beiden Optionen „--hostname“ und „--port“ charakterisiert. Ein Instanz- oder Datenbank-Name, wie ihn Oracle kennt, ist bei MySQL unbekannt: „shell> mysql --host=127.0.0.1 --port=3306 --user=root --password“. Wurde MySQL frisch installiert und noch nicht gehärtet, hat der User „root“ noch kein Passwort.

```
shell> service mysql start
shell> service mysql stop
```

Listing 1

```
shell> /etc/init.d/mysql start
shell> /etc/init.d/mysql stop
```

Listing 2

```
shell> ps -ef | grep mysqld
UID        PID    PPID    C STIME    TTY      TIME CMD
mysql     1354    1       0 Jun11    ?        00:00:00 /bin/sh bin/mysqld_safe --datadir=/var/lib/mysql
--basedir=usr
mysql     1580    1354    0 Jun11    ?        00:00:43 /usr/bin/mysqld --basedir=/usr --datadir=/var/lib/mysql
--log-error=/var/lib/mysql/error.log --pid-file=/var/lib/mysql/mysql.pid --socket=/tmp/mysql.sock --port=3306
```

Listing 3

Bei der „--host“-Option gilt es zu beachten: „localhost“ ist nicht wie bei Unix-Systemen üblich ein Synonym für „127.0.0.1“, sondern zeigt dem MySQL-Client an, dass er sich nicht über TCP (Netzwerk), sondern über einen lokalen Unix-Socket verbinden soll.

Zudem muss beachtet werden, dass die MySQL-Standard-Pakete der Ubuntu- und Debian-Distributionen per Default den Zugriff auf MySQL von Remote unterbinden. Dies geschieht mit der Variable „bind_address“ in der MySQL-Konfigurations-Datei „/etc/mysql/my.cnf“. Ein Auskommentieren dieser Zeile, gefolgt von einem MySQL-Neustart, erlaubt den Zugriff auch von Remote. Wer lieber mit einem grafischen Benutzer-Interface (GUI) arbeitet, kann sich entweder der MySQL-Workbench *[2]* oder „phpMyAdmin“ *[3]* bedienen.

Schemata

Ist die Verbindung auf die Datenbank erst einmal hergestellt, kann man sich mit den Befehlen „mysql> SHOW SCHEMAS;“ oder „mysql> SHOW DATABASES;“ anzeigen lassen, welche Schemata die MySQL-Datenbank kennt. Ein Schema in MySQL entspricht in etwa einem Schema in Oracle. Der Unterschied zu Oracle besteht hauptsächlich darin, dass ein Schema nicht einem spezifischen User gehört, sondern der Instanz selbst. In MySQL können einem User nur spezifische Rechte auf bestimmte Objekte übertragen werden. Ein User ist aber nie Besitzer eines Objekts. Vier Schemata werden in MySQL typischerweise beim Erstellen der Datenbank angelegt:

- Das Schema „mysql“ enthält hauptsächlich alle Informationen über Benutzer und deren Privilegien.
- Das Schema „test“ ist, wie der Name schon sagt, für Testzwecke da. In diesem Schema kann man beliebig Tabellen anlegen und wieder löschen.
- Das „INFORMATION_SCHEMA“ („I_S“) bietet ein standardisiertes Interface für SQL-Abfragen auf Meta-Informationen der MySQL-Instanz. Hier gibt es zum Beispiel eine Tabelle namens „TABLES“, die Informationen zu allen Tabellen innerhalb der Instanz beinhaltet.
- Das „PERFORMANCE_SCHEMA“ („P_S“) beinhaltet zahlreiche Informationen und Messwerte, die für MySQL-Performance-Messungen und -Auswertungen zu Rate gezogen werden können.

„I_S“ und „P_S“ entsprechen in etwa den Oracle-„v$“-Tabellen.

Tabellen anlegen

Mit dem Befehl „mysql> use test“ wechselt man in den Kontext eines Schemas. Alle Operationen, die nicht explizit mit einem Schema-Namen spezifiziert werden, erfolgen nun auf den Objekten des gewählten Schemas. Der Befehl „mysql> SHOW TABLES;“ gibt einen Überblick darüber, welche Tabellen im entsprechenden Schema vorhanden sind. Jetzt kann man, wie von Oracle gewohnt, eine Tabelle anlegen *(siehe Listing 4)*. Neu dürfte für den Oracle-DBA das Schlüsselwort „ENGINE“ sein.

MySQL Storage Engines

Ein Konzept, das MySQL zugrunde liegt, sind die sogenannten „Pluggable Storage Engines“. Das bedeutet, dass MySQL eigentlich mit verschiedenen Datenbank-Containern als Backend umgehen kann, sofern diese die entsprechenden Interfaces aufweisen. Hypothetisch wäre es sogar möglich, dass die MySQL-Instanz über ein natives Interface auf eine Tabelle einer Oracle Datenbank zugreift. Mit „mysql> SHOW ENGINES;“ kann man sehen, welche Storage Engines MySQL zurzeit gerade kennt *(siehe Listing 5)*.

Es gibt eine Storage Engine namens „InnoDB“, die in der Lage ist, Daten in transaktionalen Tabellen zu speichern („InnoDB“ ist heute „default“) sowie eine Storage Engine names „MyISAM“, die Daten in nicht-transaktionalen Tabellen speichert. Die CSV Storage Engine speichert Daten zum Beispiel in CSV-Dateien (Comma-Separated-Values) ab, die dann von anderen Anwendungen wie zum Beispiel Excel gelesen werden können.

So gibt es für verschiedene Anwendungszwecke unterschiedliche Storage Engines. Das bietet die Flexibilität, eine Storage Engine für die spezifischen Bedürfnisse zu wählen, hat aber auch den Nachteil, dass man sich falsch entscheiden kann, wenn man deren Eigenschaften nicht genau kennt. Für die meisten Anwendungsfälle ist jedoch die voreingestellte InnoDB Storage Engine die richtige Wahl. Die schematische Darstellung des „mysqld“-Prozesses in *Abbildung 1* veranschaulicht die Idee der Storage Engines.

Mit „mysql> SHOW TABLE STATUS LIKE ‚test‘\G“ oder „mysql> SHOW CREATE TABLE test\G“ kann angezeigt werden, welche Storage Engine von der Tabelle genutzt

Sparen Sie Zeit, Geld und Nerven.

Effizient und preiswert: DBConcepts.

Wir unterstützen Sie remote beim Betrieb von Oracle Datenbanken.

SLA ab 10hx5 bis 24hx7 inklusive
- proaktiver Überwachung
- rascher Reaktionszeit
- periodische Health Checks
- Backup und Recovery Tests

DBCONCEPTS

Die Oracle Experten

www.dbconcepts.at
Tel.: +43 1 890 89 990
office@dbconcepts.at

ORACLE Platinum Partner

```
mysql> CREATE TABLE test (
   id INT UNSIGNED NOT NULL
, data VARCHAR(255)
, ts TIMESTAMP
) ENGINE = InnoDB;
```

Listing 4

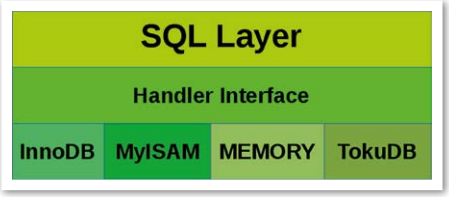

Abbildung 1: MySQL-Architekur

```
+--------------------+---------+--------------+------+
| Engine             | Support | Transactions | XA   |
+--------------------+---------+--------------+------+
| MyISAM             | YES     | NO           | NO   |
| CSV                | YES     | NO           | NO   |
| MRG_MYISAM         | YES     | NO           | NO   |
| BLACKHOLE          | YES     | NO           | NO   |
| MEMORY             | YES     | NO           | NO   |
| PERFORMANCE_SCHEMA | YES     | NO           | NO   |
| ARCHIVE            | YES     | NO           | NO   |
| InnoDB             | DEFAULT | YES          | YES  |
| FEDERATED          | NO      | NULL         | NULL |
+--------------------+---------+--------------+------+
```

Listing 5

wird. Storage Engines werden jeder Tabelle zugewiesen. Es besteht so die Möglichkeit, Abfragen beziehungsweise Joins über zwei Tabellen mit unterschiedlichen Storage Engines auszuführen.

MySQL-User

Um herauszufinden, welche User in der MySQL-Instanz existieren, hilft die Tabelle „user" im Schema „mysql". Entweder mit „mysql> use mysql;" und „mysql> SELECT user, host, password FROM user;" oder mit „mysql> SELECT user, host, password FROM mysql.user;" kann bestimmt werden, welche User aktuell in der Instanz sind *(siehe Listing 6)*.

Hierbei stellt man fest, dass es zum Beispiel den User „root" mehr als einmal gibt. Hierzu muss man wissen, dass in MySQL ein User immer aus „Username" und „Host" besteht. Die User „'root'@'localhost'" und „'root'@'127.0.0.1'" sind somit unterschiedliche User. Und wie wir aus obigen Erläuterungen erraten können, darf sich der erste User „root" nur über den lokalen Unix-Socket („localhost"), der zweite User „root" („127.0.0.1") nur über den lokalen TCP-Port 3306 und der dritte User „root" nur vom Host „centos-tmpl", der über einen DNS-Lookup aufgelöst wird, verbinden.

Der User „root" in MySQL hat typischerweise alle Privilegien und ist somit Superuser der MySQL-Instanz. Dies entspricht in etwa dem User „SYS" in Oracle. Welche Rechte ein User hat, lässt sich mit dem Befehl „mysql> SHOW GRANTS FOR 'root'@'localhost';" herausfinden.

Allgemein gilt es als sicherheitstechnisch bedenklich, dem User „root" Zugriff von Remote zu gewähren. Darüber hinaus sind auch der User „test" sowie das Schema „test" auf Produktiv-Umgebungen fragwürdig. Um eine MySQL-Instanz für den produktiven Einsatz zu härten, gibt es daher das Skript „shell> mysql_secure_installation", das einige sicherheitsrelevante Einstellungen für MySQL vornimmt.

```
+------+-------------+----------+
| user | host        | password |
+------+-------------+----------+
| root | localhost   |          |
| root | centos-tmpl |          |
| root | 127.0.0.1   |          |
|      | localhost   |          |
|      | centos-tmpl |          |
+------+-------------+----------+
```

Listing 6

MySQL-Konfiguration

Die Konfiguration von MySQL erfolgt über die Datei „/etc/my.cnf" (CentOS, SuSE) oder „/etc/mysql/my.cnf" (Ubuntu, Debian). Unter Windows liegt diese unter „C:\Program Files\MySQL\MySQL Server 5.6" und heißt „my.ini". Darin sind alle MySQL-spezifischen Variablen angegeben, um die Instanz zu konfigurieren und zu tunen.

Die Datei „my.cnf" ist unterteilt in verschiedene Abschnitte, die mit eckigen Klammern gekennzeichnet sind. Es gibt einen Abschnitt für den eigentlichen Datenbank-Prozess („mysqld"), für den „mysqld_safe"-Prozess („mysqld_safe"), für alle lokalen MySQL-Client-Prozesse („client") etc. Je nachdem, was man konfigurieren möchte, muss man beachten, dass die entsprechenden Parameter in der richtigen Sektion landen. Sonst gibt es entweder Fehler oder eine Konfiguration greift nicht.

Einige Variablen in MySQL sind Session-spezifisch, andere gelten global. Einige Variablen lassen sich dynamisch im laufenden Betrieb ändern, andere erfordern einen Instanz-Neustart, um aktiviert zu werden. Welche Variablen wie aktiviert werden, steht in der MySQL-Dokumentation *[4]*. Der Befehl „mysql> SHOW GLOBAL VARIABLES LIKE '%xyz%';" gibt an, welche Werte für MySQL-Variablen aktuell gesetzt sind.

MySQL-Dokumentation

Neben einer ganzen Reihe von Büchern zum Thema MySQL bietet MySQL auch eine Online-Dokumentation, die recht gut ist und täglich aktualisiert wird *[5]*.

Links

[1] MySQL Downloads: http://dev.mysql.com/downloads
[2] MySQL Workbench: http://dev.mysql.com/downloads/workbench
[3] phpMyAdmin: http://www.phpmyadmin.net
[4] Server Option and Variable Reference: http://dev.mysql.com/doc/refman/5.6/en/mysqld-option-tables.html
[5] MySQL Dokumentation: http://dev.mysql.com/doc

Oli Sennhauser
oli.sennhauser@fromdual.com