# Verbinde die Welten

**Ronny Fauth**
**DB Systel GmbH**
**Schlachthofstrasse 80, 99085 Erfurt**

**Schlüsselworte**
MySQL, Oracle, ODBC, dg4odbc, mysql-odbc, unix-odbc, Datenbanklinks

**Einleitung**
Die Zeiten, als sich ein Datenbanksystem als Allround-Lösung darstellen und durchsetzen konnte, gehören lange der Vergangenheit an. Die Anwender, die wachsenden Anforderungen und nicht zuletzt die wachsenden Anzahl an verschiedenen Datenbanken belehren die Firmen immer mehr eines Besseren. So wird die Forderung laut, die Datenbanken untereinander verbinden zu müssen um effizienter arbeiten zu können. Ein solcher Weg soll hier aufgezeigt werden. Es soll gezeigt werden, dass es mit zwei Oracle-Produkten sehr einfach ist ein inhomogenes System zu bauen welches stabil funktioniert.

**Verwendung**
Die Anwendungsfälle für eine solche Lösung wie die hier vorgestellte, sind recht vielfältig. Verschiedene Datenbanksysteme haben verschiedene Vorteile. MySQL zum einen ist leicht zu bedienen, und zum anderen für, bis auf wenige Ausnahmen, alle Plattformen verfügbar und nicht zu vergessen: kostenfrei in der Community-Edition. Oracle hingegen ist sehr leistungsfähig, für großen Durchsatz ausgelegt und durch DataGuard und Real Application Cluster sehr stark im Höher- und Hochverfügbarkeitssektor vertreten. Doch die Lizenzkosten unterbinden eine Verwendung in kleinen Bereichen, wie zum Beispiel einem Webserver. Natürlich gibt es die Möglichkeit sich die Express-Edition zu installieren, doch auch hier gibt es Einschränkungen, wie zum Beispiel dass die Verfügbarkeit auf wenige Plattformen beschränkt ist. Doch was tun, wenn man die Vorteile von beiden Datenbanksystemen verbinden möchte? Ein denkbares Szenario könnte folgendermaßen aussehen:

Eine kleine Firma betreibt verschiedenen Onlineshops, in diesen wird MySQL verwendet. Nun sollen die Daten aber gemeinsam gesichert werden. Auch soll ein zentrales Reporting eingerichtet werden welches die Umsätze über alle Onlineshops aggregiert und gleichzeitig eine Bestandsführung ermöglicht. Ein sicher denkbarer, aber mit Nachteilen behafteter Weg wäre es, die Datenbanken auf den Webservern einmal nachts zu exportieren und auf einem zentralen Server zu importieren. Schon allein der Abstand zwischen den Importen stellt einen Nachteil dar.

Eine Lösung wäre folgende:

Man baut als zentralen Server eine Oracle-Datenbank auf, diese erhält über ODBC und Datenbanklinks Zugriff auf die Webserverdatenbanken. Über eingerichtete Datenbankjobs werden die Shop-Datenbanken einmal alle 10 Minuten (die Zeit kann je nach Anforderung variiert werden) abgeglichen mit den Daten welche gespeichert sind. Auf dieser Oracle-Instanz kann zum Beispiel über Apex der Umsatz visualisiert oder ein Abgleich mit einer Bestandsdatenbank durchgeführt werden, welche dann gleichzeitig eine Bestellung auslöst sollte ein Artikel nur noch in geringer Stückzahl vorhanden ist. Auch die Sicherung kann über RMAN direkt auf dieser Datenbank erfolgen. Sind die Onlineshop-Datenbanken klein genug, so kann auch eine Vollsicherung in der Oracle-Datenbank

erfolgen.

Gleichwohl könnte man dieses Szenario noch größer Beschreiben, anstelle der einfachen MySQL-Datenbank könnte man sich eine Master-Slave-Replikation vorstellen, bei der aus einem der Slaves die entsprechenden Daten gezogen werden. Auf der anderen Seite wiederum könnte man die einfache Oracle-Datenbank gegen einen DataGuard oder ein Real Application Cluster austauschen welches die Ausfallsicherheit um ein vielfaches erhöht. Und noch globaler könnte man Standorte überall auf der Welt mit diesem Szenario auf dem neuesten Stand halten. Erwähnt sei hier auch, dass es sich nicht allein auf MySQL-Oracle-Verbindungen beschränken muss. Es besteht auch die Möglichkeit MSSQL-Oracle-Verbindungen herzustellen und in einem vollkommen inhomogenen System eine Möglichkeit zum Datenaustausch zu schaffen.

Der geneigte Leser mag sehen, dass dieses nur ein Szenario von vielen möglichen ist. Die Anwendungsgebiete sind fast unbegrenzt.

**Voraussetzungen**
Es müssen auf Seiten der Oracle-Datenbank einige Treiber installiert sein um diese Art der Kommunikation nutzen zu können. Diese sollen hier genannt werden:

- unix-odbc
- mysql-odbc

und ihre Abhängigkeiten. Es wird unix-odbc extra erwähnt, da es zwar eine Abhängigkeit von mysql-odbc ist, aber es keine Fehlermeldung gibt, falls dieser nicht vorhanden ist. Dieses führt zu seltsam unverständlichen Fehlermeldungen.

**Aufbau**
Der einfachste Fall stellt 1 MySQL Datenbank (Version egal) und 1 Oracle Datenbank (Version 11.1 oder Version 11.2) dar. Eine Oracle 10 Datenbank ist zwar prinzipiell auch verwendbar, aber es muss trotzdem die ODBC-Treiber der Version 11 (dg4odbc) in ein eigenes Oracle_Home installiert werden. Um den Fall dazu noch einfach und problemfrei zu halten, verwenden wir für die Oracle-Datenbank Oracle Unbreakable Linux als Betriebssystem. Auch ist es möglich beide Datenbanken auf Oracle Unbreakable Linux zu betreiben, dann entfällt die Netzwerkverbindung und man kann auf die MySQL-Datenbank per socket zugreifen. Dieser Fall wird hier auch der Einfachheit halber angenommen. Normalerweise ist das Betriebssystem der MySQL-Datenbank unwichtig, da auf diese per TCP zugegriffen wird.

Die “Installation” braucht nur vier Dateien:

```
/etc/odbc.ini

[ODBC Data Sources]
mysql = MySQL ODBC Driver 5.1

[mysqlora]
DATABASE       = oracle_test
DESCRIPTION = MySQL ODBC 5.1.5 Connector
PORT           = 3306
SERVER         = localhost
#UID           = oracle_test_user
```

```
#PWD           = oracle_test
Socket         = /var/lib/mysql/mysql.sock
CHARSET        = latin1
TRACEFILE   = /tmp/myodbc-nicdsn.trc
TRACE          = OFF
```

```
/home/oracle/app/oracle/product/11.2.0/dbhome_1/hs/admin/initmysql.ora

HS_FDS_CONNECT_INFO=mysqlora
HS_FDS_SUPPORT_STATISTICS=FALSE
HS_FDS_TRACE_LEVEL=255
HS_LANGUAGE=AMERICAN_AMERICA.WE8ISO8859P15
HS_FDS_SHAREABLE_NAME=/usr/lib/libmyodbc5.so
#
# ODBC specific environment variables
#
set ODBCINI=/etc/odbc.ini
```

und

```
/home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.ora

SID_LIST_LISTENER =
  (SID_LIST =
               (SID_DESC =
               (ORACLE_HOME    =    /home/oracle/app/oracle/product/11.2.0/
dbhome_1)
               (SID_NAME = mysql)
               (PROGRAM = dg4odbc)
               )
  )

/home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora

MYSQL =
  (DESCRIPTION =
               (ADDRESS_LIST =
               (ADDRESS = (PROTOCOL = TCP)
               (HOST = localhost)
               (PORT = 1521)
               )
               )
               (CONNECT_DATA =
               (SID = mysql)
               )
               (HS = OK)
  )
```

Danach kann ein Datenbanklink angelegt werden mit dem auf die MySQL-Datenbank zugegriffen werden kann.

```
create database link mysql connect to "oracle_test_user" identified
by "oracle_test" using 'MYSQL';
```

```
Database link created.
```

Das einzig nicht intuitive an der Konfiguration ist die Benennung des init<SID>.ora, hier in diesem Falle initmysql.ora
Hier ein Beweis für die Funktionstüchtigkeit:

```
SQL> select "a" as TEST from "a"@mysql;

              TEST
----------
              1
              2
```

**Probleme**
Es sei hier erwähnt, dass es auch Konfigurationen gab, welche nicht funktioniert haben. Dieses lag größtenteils an mysql-odbc. Dieser Treiber funktioniert zum Beispiel nicht auf ubuntu 10.04 64bit, auch die Verwendung unter Solaris 10 ist problematisch, da hier der unix-odbc-Treiber nicht als Paket vorliegt sondern kompiliert werden muss. Hier muss Oracle nachbessern. Auch kann ein kleiner Fehler in den Konfigurationsdateien dazu führen, dass das gesamte Konstrukt nicht mehr lauffähig ist. Die Probleme mit dem mysql-odbc-Treiber sind schwerwiegender, denn die Log-files (zu finden unter `/home/oracle/app/oracle/product/11.2.0/dbhome_1/hs/log/*.trc`)
sind weniger aussagekräftig und eher kryptisch. Eine Möglichkeit im vornherein ein Problem damit auszuschließen ist die Verwendung von ldd:

```
[oracle@localhost log]$ ldd /usr/lib/libmyodbc5.so
              linux-gate.so.1 =>  (0x00ca4000)
              libdl.so.2 => /lib/libdl.so.2 (0x00450000)
              libcrypt.so.1 => /lib/libcrypt.so.1 (0x00351000)
              libnsl.so.1 => /lib/libnsl.so.1 (0x00153000)
              libm.so.6 => /lib/libm.so.6 (0x006bd000)
              libpthread.so.0 => /lib/libpthread.so.0 (0x00d1a000)
              libodbcinst.so.1 => /usr/lib/libodbcinst.so.1 (0x00207000)
              libc.so.6 => /lib/libc.so.6 (0x00454000)
              /lib/ld-linux.so.2 (0x00736000)
```

wie hier zu sehen ist, sind alle Abhängigkeiten erfüllt, besondere Wichtigkeit hat hier die libodbcinst.so.1, denn diese wird durch unix-odbc bereitgestellt. Sollte an dieser Stelle ein “not found” stehen, ist es ausgeschlossen, dass die Verbindung funktionieren kann.

**Weiter interessante Architekturen**
Es ist auch möglich die Gateway Software getrennt von der eigentlichen Oracle-Datenbank auf einem anderen Server zu installieren. Ohne weitere Lizenzkosten. Dieses ermöglicht es, das Gateway auf einem Server zu betreiben welcher unix-odbc und mysql-odbc ohne Probleme unterstützt. So ist es möglich einen Server dediziert für die Gateway Aufgabe zu konfigurieren und weitere Oracle-Datenbanken ohne die Gateway Software betreiben zu können. Somit steht auch der Verwendung in großen Datenbanklandschaften nichts im Wege. Es ist möglich die Gateways zentral auf einem Server zu bündeln und die Verwaltung somit zu vereinfachen. Auch wenn das nach einem Single-Point-of-Failure klingt, so ist es keiner, denn auch hier kann man Methoden zum Failover einbauen.

**Fazit**
Die Verwendung von MySQL und Oracle ist, unter gewissen Vorrausetzungen, einfach zu konfigurieren und effektiv nutzbar. Es steht also einer Verbindung der Welten nichts im Wege.

**Kontaktadresse:**
Ronny Fauth
DB Systel
Schlachthofstrasse 80
D-99085 Erfurt

Telefon: +49 (0) 361 300 5944
E-Mail ronny.fauth@deutschebahn.com
Internet: www.deutschebahn.com