Apex on the Rocks – Hochverfügbarkeit für eine komplette Apex-Umgebung

Moritz Klein, MT AG

Häufig starten Oracle-Application-Express-Anwendungen als kleine, von Fachabteilungen gestartete Initiativen, entwickeln sich dann aber schnell zu geschäftskritischen Anwendungen. Um diesem Umstand gerecht zu werden, reicht der Rechner unter dem Schreibtisch nicht mehr aus. Dieser Artikel zeigt eine mögliche Architektur, um Apex zukunftssicher zu betreiben.

Oracle Application Express bietet als Plattform für die Entwicklung von Web-Anwendungen einen sehr schnellen Einstieg. Zu Beginn machen sich viele deshalb meist wenig Gedanken über die notwendige Infrastruktur. Häufig wird eine vorhandene Datenbank-Installation oder auch die kostenfreie Express Edition genutzt und in der Regel werden diese Umgebungen dann durch einen versierten Mitarbeiter aus der Fachabteilung betrieben. Die Einfachheit, mit der man eine zunächst völlig ausreichende Umgebung aufbauen kann, ist eine der Stärken von Apex. Weitere Stärken, wie große Flexibilität und hohe Entwicklungs-Geschwindigkeit, führen dann zu einer Art viralem Marketing. Begeisterte Nutzer erzählen ihren Kollegen von erreichten Arbeitserleichterungen, dies führt dann zu einem größeren Nutzerkreis und zusätzlichen Applikationen.

Apex ist aber bei Weitem nicht auf kleine Applikationen beschränkt, sondern auch – unter anderem durch die Verbindung mit der Oracle-Datenbank – absolut für geschäftskritische Anwendungen tauglich. Sobald man sich an geschäftskritische Anwendungen heranwagt, sollte man sich allerdings umfangreiche Gedanken über die notwendige Umgebung für solche Anwendungen machen.

Die vom Autor betreute Apex-Umgebung stand im September 2014 genau an dieser Stelle und es musste dringend eine neue, zukunftssichere Umgebung her. Bis dahin lief Apex auf einem gemeinsam genutzten Datenbank-Server; der Web-Server war als virtuelle Maschine aufgesetzt. Durch die hohe Menge an unterschiedlichen Datenbanken auf dem gleichen Server beeinträchtigten sich diese zum Teil gegenseitig und es wurde zunehmend schwieriger, Wartungsfenster zu finden. Glücklicherweise hatten die Anwendungen schon einen so hohen Stellenwert erreicht, dass man aus dem Vollen schöpfen konnte. Schnell war klar, dass eine Cluster-Lösung zum Einsatz kommen sollte – vor allem, um Hochverfügbarkeit zu gewährleisten. Aber zunächst eine kurze Bestandsaufnahme, um die abzusichernden Komponenten zu identifizieren.

Apex-Architektur

Apex als solches besteht nur aus Metadaten-Tabellen und PL/SQL-Paketen. Lässt man die statischen Dateien wie Bilder, Stylesheets und JavaScript-Dateien außen vor, liegen alle benötigten Teile in der Datenbank. Zur Auslieferung der erzeugten Webseiten benötigt man nur noch einen Web Listener. Dies kann prinzipiell

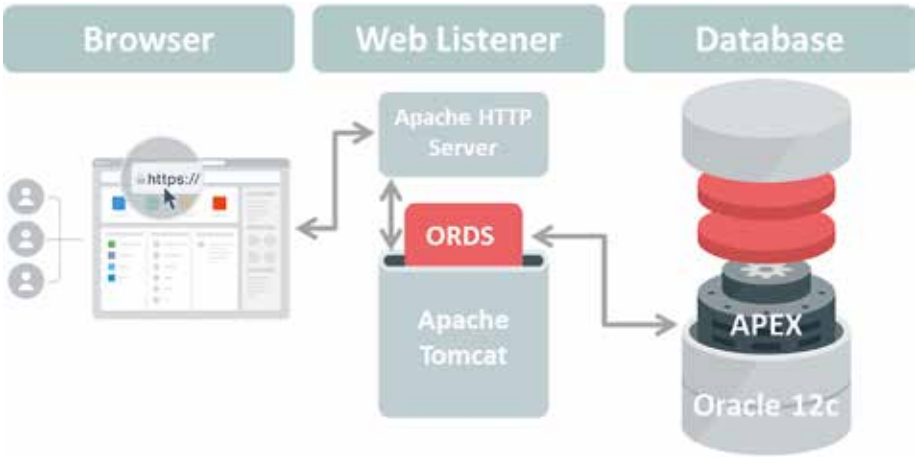

Abbildung 1: Apex-Architektur

Oracle-XDB („Embedded PL/SQL Gateway“) sein, davon sollte man allerdings in Produktions-Umgebungen Abstand nehmen. Eine heute übliche Umgebung besteht aus den folgenden Komponenten *(siehe Abbildung 1)*:

- Oracle Database Server 12*c*
- Oracle REST Data Services (ORDS)
- Apache Tomcat
- Apache HTTP Server

Basierend auf der generellen Architektur kann man die Analyse von Cluster-Lösungen auf zwei Gesichtspunkte aufteilen: Absicherung der Datenbank (inklusive Apex) und Absicherung des Web-Tier.

Absicherung der Datenbank

Um einen Überblick über von Oracle zur Verfügung gestellte Produkte und Datenbank-Optionen zu erhalten, lohnt sich der Blick in die Dokumentation, in diesem Fall „Database High Availability Overview“ *(siehe „https://docs.oracle.com/database/121/HAOVW/toc.htm“)*. Wer direkt die empfohlenen Architekturen für die verschiedenen Service-Level sehen will, kann dabei direkt zu Kapitel 7 springen. Oracle unterscheidet hier in Bronze, Silber, Gold und Platinum mit immer höheren Anforderungen an die maximal erlaubte Ausfallzeit.

Die Entscheidung im Projekt fiel hierbei auf die Silber-Architektur, dabei handelt es sich um die Nutzung von Oracle Real Application Clusters (RAC). Eine Lösung mittels Data Guard wurde alternativ untersucht, da auch hiermit die generellen Ansprüche an Hochverfügbarkeit erfüllt worden wären. Allerdings bietet RAC die Möglichkeit, durch Hinzufügen von weiteren Servern flexibel auf geänderte Anforderungen zu reagieren, um zum Beispiel später zusätzliche Apex-Installationen im Unternehmen auf einer zentralen Plattform zu konsolidieren.

Real Application Clusters

Real Application Clusters (RAC) erlaubt den Zugriff mehrerer Datenbank-Instanzen von verschiedenen Servern im Cluster auf gemeinsame Daten-Dateien, auch als Datenbank bezeichnet. Diese Datenbank erstreckt sich also über mehrere Server, stellt sich der Applikation aber als eine einheitliche Datenbank dar. Bei RAC handelt es sich um eine „shared disk“-Lösung, alle beteiligten Server benötigen also den Zugriff auf die gleichen Daten-Dateien. Die einzelnen Instanzen besitzen jeweils einen eigenen lokalen Buffer Cache, können jedoch auch auf Inhalte aus dem Buffer Cache einer anderen Instanz zugreifen, da alle Buffer Caches mittels Cache-Fusion logisch zusammengefasst sind.

RAC setzt auf Komponenten der Grid Infrastructure auf. Der Datenspeicher wird meist durch Automatic Storage Management (ASM) verwaltet und den Datenbank-Instanzen zur Verfügung gestellt. Die Oracle Clusterware übernimmt die Steuerung und Verwaltung aller Ressourcen im Cluster. Folgt man der von Oracle zur Verfügung gestellten Dokumentation und hält sich an die beschriebenen Hard- und Software-Anforderungen, ist die Installation nicht viel komplizierter als die einer Nicht-RAC-Datenbank. Die Installations-Assistenten von Oracle sind mittlerweile auch wesentlicher besser verständlich und damit auch besser nutzbar geworden.

```
SET SERVEROUTPUT ON
SPOOL apxpart.log
@apxpart partitions=auto,print,noexec
SPOOL OFF
```

Listing 1

```
yum install httpd #Apache installieren#
yum install tomcat tomcat-admin-webapps #Tomcat installieren#
systemctl start httpd #Apache starten#
systemctl stop httpd #Apache stoppen#
systemctl is-active httpd #Läuft Apache?#
```

Listing 2

Apex und RAC

Um Apex für den effizienten Einsatz mit RAC fit zu machen, bedarf es der Ausführung eines Skripts. Dieses findet sich unter dem Namen „apxpart.sql“ im Unterordner „utilities“ der Apex-Installations-Dateien. Es führt einige Datenmodell-Optimierungen für eine RAC-Umgebung aus. Dabei werden einige Tabellen mit Range-Partitionierung basierend auf der Session-ID versehen. Die führenden Zahlen der Session-ID orientieren sich dabei an der jeweiligen Instanz-Nummer. Kombiniert man das Ganze mit Instanz-Affinität auf dem Web Server, müssen Daten, die von der Session abhängen, seltener zwischen den Caches der RAC-Instanzen transferiert werden.

Möchte man sich einen Überblick über die durchgeführten Änderungen machen, verbindet man sich als „SYS“ und führt die Befehle aus *Listing 1* aus. Steht keine RAC-Datenbank zum Testen zur Verfügung, kann man im Listing „partitions=auto“ durch „partitions=2“ ersetzen, um ein Zwei-Knoten-RAC zu simulieren. Die Optionen „print“ und „noexec“ sorgen dafür, dass die Befehle für die Datenmodell-Änderungen mittels „DBMS_OUTPUT“ ausgegeben, aber nicht ausgeführt werden. Trotzdem sollte man das natürlich nicht einfach auf einer Produktions-Datenbank ausprobieren.

Clusterware kann mehr als nur RAC

Die Oracle Clusterware kennen viele nur als Basis für RAC, es handelt sich jedoch um eine vollwertige Clusterware. Demnach lassen sich auch andere Applikationen und Prozesse überwachen. Bei den Überlegungen zur Absicherung der Web-Tier-Komponenten spielte dies eine maßgebliche Rolle und war einer der Gründe, RAC statt Data Guard zu benutzen. Schließlich ist es für den Betrieb der Umgebung von Vorteil, wenn man sich nur mit einer Cluster-Software vertraut machen muss.

Oracle liefert die Oracle Grid Infrastructure Agents aus, um ausgewählte Applikationen als Clusterware-Ressourcen zu verwalten. Hierzu kann man sich in der Ausgabe von „bin/agctl“ aus der Grid-Infrastructure-Installation anschauen, welche Applikationen konkret unterstützt werden. Für weitere Informationen sollte man auf jeden Fall auch den Oracle Reference Guide „Oracle Grid Infrastructure Agents“ nutzen *(siehe „http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/ogiba-2189738.pdf“)*.

Will oder kann man nicht auf die vorgefertigten Agents setzen, lohnt sich ein Blick in das Verzeichnis „crs/demo“. Weitere Informationen hierzu liefert der „Oracle Clusterware Administration and Deployment Guide“, speziell Kapitel 9, „Making Applications Highly Available Using Oracle Clusterware“ *(siehe „http://docs.oracle.com/database/121/CWADD/crschp.htm#CHDGHGJA“)*.

```
appvipcfg create -network=1 -ip=192.168.56.250 -vipname=webVIP
-user=root
```

Listing 3

```
crsctl add type systemctl_agent_type
   -basetype cluster_resource
   -attr "ATTRIBUTE=SVC_NAME, TYPE=string, FLAG=REQUIRED,
       ATTRIBUTE=ACTION_SCRIPT, TYPE=string,
       DEFAULT_VALUE=systemctl_agent.sh"
```

Listing 4

```
case $1 in
...
'check')
   /usr/bin/systemctl is-active $_CRS_SVC_NAME
   RET=$?
...
esac
if [ $RET -eq 0 ]
   then exit 0
   else exit 1
fi
```

Listing 5

Apache HTTP Server und Apache Tomcat installieren

Die Komponenten des Web-Tier sollten möglichst mit Betriebssystem-Mitteln installiert und betrieben werden. Solange man mit den vom Betriebssystem-Hersteller angebotenen Versionen zufrieden ist, ist diese Vorgehensweise auch generell zu bevorzugen. Die Integration in vorgegebene Verzeichnis-Strukturen und Betriebssystem-Werkzeuge ist dann einfach besser als bei einer Installation von Hand.

Im Projekt werden Red Hat Enterprise Server 7 eingesetzt und demnach per „yum“ die Web-Komponenten installiert. Als Werkzeug zur Dienste-Verwaltung kommt dann „systemctl“ zur Verwendung, das vom Betriebssystem mitgeliefert wird und entsprechend gut integriert ist. *Listing 2* zeigt hier die Befehle zur Installation und einige Beispiele zur Verwaltung der Dienste.

Der Apache HTTP Server wird als Proxy vor den Tomcat geschaltet und liefert statische Dateien direkt aus. Die generelle Konfiguration beider Komponenten kann in der jeweiligen Dokumentation nachgelesen werden und unterscheidet sich prinzipiell nicht von einer Konfiguration ohne Cluster. Bei den Oracle REST Data Services ist nur zu beachten, dass man den sogenannten „SCAN Listener“ und nicht etwa einen lokalen Listener adressiert, dies gilt aber generell, wenn man auf eine RAC-Datenbank zugreift.

Die Web-Komponenten absichern

Der Apache HTTP Server benötigt zuerst eine IP-Adresse, unter der er von den Clients erreichbar ist. Diese sollte immer die gleiche sein, egal auf welchem Knoten der Web-Server gerade läuft. Weiterhin muss die Clusterware erkennen können, ob die Komponenten laufen, und auch wissen, wie diese gestartet und gestoppt werden. Hängen Komponenten voneinander ab, sollte das natürlich auch konfiguriert werden. Schließlich muss noch entschieden werden, wie man für konsistente Konfiguration auf allen Knoten sorgt.

Virtuelle IP für den Web-Server

In der Clusterware gibt es einen Ressourcen-Typ „Virtual IP“ (VIP). Dieser wird auch für den SCAN-Listener benutzt, um die Zuordnung der IP-Adressen zu den einzelnen Knoten dynamisch verwalten zu können. Für den Web-Server kommt hier ein spezieller Typ „Application Virtual IP“ in Betracht, Ressourcen dieses Typs werden mittels appvipcfg verwaltet. Das Skript in *Listing 3* zeigt den Aufruf von „appvipcfg“ in einer Test-Umgebung, um eine entsprechende Ressource anzulegen. Es werden das Standard-Netzwerk des Clusters verwendet, die gewünschte IP bestimmt, der Ressourcen-Name festgelegt und „root“ als Besitzer spezifiziert.

Eigenen Ressourcen-Typ anlegen

Neben den von der Clusterware vordefinierten Ressourcen-Typen kann man auch eigene anlegen, um sich Vorlagen für ähnliche Ressourcen zu erstellen. Diese werden von einem Basis-Typ abgeleitet, um bereits definierte Attribute nicht erneut definieren zu müssen. Hier bietet sich der generische Typ „cluster_resource“ an.

Listing 4 zeigt den entsprechenden Aufruf, um einen eigenen Ressourcen-Typ „systemctl_agent_type“ anzulegen sowie die beiden Attribute „SVC_NAME“ und „ACTION_SCRIPT“ zu definieren. At-

```
crsctl add resource tomcat01 -type systemctl_agent_type -attr "SVC_NAME=tomcat, PLACEMENT=restricted,
 CARDINALITY=1, SERVER_POOLS=*,
 START_DEPENDENCIES='hard(webVIP,ora.data.asm_shared.acfs)
                pullup(webVIP,ora.data.asm_shared.acfs)',
 STOP_DEPENDENCIES='hard(webVIP,ora.data.asm_shared.acfs)'"
```

Listing 6

tribute werden durch die Clusterware als Umgebungsvariablen mit dem Präfix „_CRS_" durchgereicht, das Attribut „SVC_NAME" ist also über eine Umgebungsvariable namens „_CRS_SVC_NAME" verfügbar. Das Attribut wird hier als erforderlich markiert, da das unter „ACTION_SCRIPT" als Standard definierte Skript diese Variable benötigt.

Action-Skript erstellen

Die Komponenten des Web-Tier werden wie beschrieben mittels „systemctl" gesteuert und der angelegte Ressourcen-Typ legt das vom Clusterware Agent aufzurufende Skript fest. Die Anforderungen an das Skript sind in der entsprechenden Dokumentation vermerkt, hier die wichtigsten: Das Skript muss auf die vier Kommandos „start", „stop", „check" und „clean" reagieren können und im Erfolgsfall mit Status „0" enden. Unter Unix/Linux-Betriebssystemen gehört es zum guten Ton, dass Skripte im Erfolgsfall „0" und bei einem Fehler einen Wert zwischen „1" und „255" als Status-Code zurückgeben, „systemctl" verhält sich hier auch entsprechend.

Ein eigenes Action-Skript als Clusterware-tauglicher Wrapper für „systemctl" kann unter *„https://github.com/commi235/ocw-systemctl"* inklusive einer kurzen Anleitung zur Integration heruntergeladen werden. Im Prinzip werden nur das Kommando („$1") vom Clusterware-Agent entgegengenommen, der entsprechende „systemctl"-Aufruf ausgeführt und der erhaltene Status an den Agent zurückgegeben. *Listing 5* zeigt eine gekürzte Fassung, in der nur der Zweig für das Kommando „check" sowie die Rückgabe an den Clusterware Agent detailliert dargestellt sind.

Die Konfigurations-Dateien

Mit dem Ressourcen-Typ und dem Action-Skript hat man die grundlegenden Bausteine, um eine eigene Ressource anzulegen. Allerdings ist es doch aufwändig und fehleranfällig, wenn man selbst für eine Synchronisation der Konfigurationsdateien und anderer von allen Knoten benötigten Dateien sorgen muss. Auch das Action-Skript muss natürlich auf allen Knoten, auf denen die Komponenten des Web-Tier laufen sollen, verfügbar sein. Hier gibt es verschiedene Möglichkeiten, wie zum Beispiel ein per NFS oder CIFS gemountetes Verzeichnis oder auch „rsync".

Setzt man bereits ASM ein, kann man mithilfe von Automatic Storage Management Cluster File System (ACFS) auch einen Teil des von ASM verwalteten Speichers auf Betriebssystem-Ebene zur Verfügung stellen, wenn das verwendete Betriebssystem unterstützt wird. Zu Beginn des Projekts war ACFS noch nicht einsetzbar, da der notwendige Patch einen Konflikt mit dem Januar-PSU hatte, dies ist jedoch seit dem April-PSU behoben.

Hat man diese Hürde überwunden, kann man mit dem ASM-Konfigurations-Assistent (ASMCA) auf einer vorhandenen Disk Group ein Volume anlegen und im Betriebssystem mounten lassen. Abschließend kopiert man dann die benötigten Dateien auf den neu angelegten Speicherort, löscht die lokalen Kopien und zeigt per symbolischen Link auf den neuen Speicherort. Im Projekt wurden die Konfigurationsdateien des Apache HTTP Server, Apache Tomcat und ORDS, das Action-Skript und das Apex-Images-Verzeichnis so für alle Knoten im Cluster verfügbar gemacht.

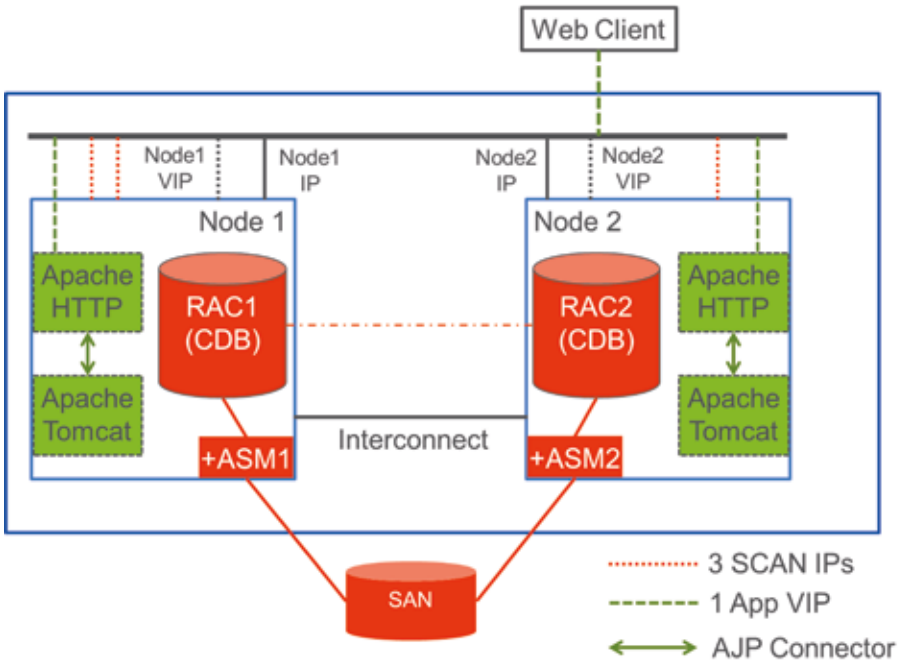

Abbildung 2: Fertige Cluster-Architektur

Cluster-Ressourcen für Web-Tier anlegen

Nachdem jetzt alle Bausteine beisammen sind, können wir die einzelnen Ressourcen für die Komponenten des Web-Tier anlegen. Bevor man jedoch einfach eine Ressource einrichtet, sollte man sich

Gedanken um deren Abhängigkeiten machen. Diese sollte man der Clusterware mitteilen, sonst kann sie die Ressource nicht korrekt kontrollieren und verwalten. Liegen die Konfigurationsdateien für den Apache Tomcat auf einem ACFS-Speicherort, kann dieser auf einem Knoten ohne diesen Speicherort nicht starten. Ist die angelegte virtuelle IP auf dem Knoten nicht verfügbar, kann auch der Apache HTTP Server nicht richtig starten.

In der Clusterware werden diese Abhängigkeiten über die Attribute „START_DEPENDENCIES“ und „STOP_DEPENDENCIES“ der Ressource definiert. *Listing 6* zeigt das Anlegen der Ressource für Apache Tomcat exemplarisch, hierbei werden auch der vorher angelegte Ressourcen-Typ verwendet und das dort definierte Attribut „SVC_NAME“ mit dem entsprechenden Service-Namen belegt.

Einen Knoten-Ausfall simulieren

Nachdem alle benötigten Komponenten installiert und konfiguriert sind, sollte man natürlich einige Tests fahren. Als initiale Minimal-Tests kann man mittels „crsctl“ oder „srvctl“ Ausfälle der einzelnen Komponenten simulieren. Die Unterscheidung, wann man welches Skript benutzt, ist relativ einfach; beginnt eine Ressource mit „ora.“, sollte man prinzipiell „srvctl“ benutzen, ansonsten landet man auch mal bei einer nicht mehr von Oracle unterstützten Konfiguration.

Zuerst verschafft man sich mit „crsctl status resource –t“ einen Überblick darüber, in welchem Zustand die Ressourcen sind und auf welchem Knoten sie laufen. Im angenommenen Ausgangszustand laufen die Ressourcen „apache01“, „tomcat01“ und „webVIP“ auf Knoten „1“, aufgrund der definierten Abhängigkeiten liegen alle drei auf dem gleichen Knoten. Das Kommando „crsctl eval relocate resource webVIP –f“ zeigt an, was passiert, wenn die Ressource „webVIP“ auf einen anderen Knoten umziehen muss. Als Erstes gehen „apache01“ und „tomcat01“ auf Knoten „1“ in den Zustand „offline“. Sobald die IP-Adresse auf Knoten „2“ „online“ ist, starten dort auch die beiden Web-Server.

Um den Ausfall von ACFS auf einem Knoten zu simulieren, gibt man „srvctl predict filesystem -device /dev/asm/asm_shared-331“ ein, wobei der Pfad nach „-device“ an die eigene Konfiguration angepasst werden muss. Abschließend lässt man zumindest einmal die Web-Komponenten von einem auf den anderen Knoten wandern, indem man „crsctl relocate resource apache01 –f“ aufruft.

Fazit

Es bedeutet schon einigen Aufwand, eine Umgebung, wie im Artikel aufgezeigt, aufzubauen *(siehe Abbildung 2)*. Speziell die notwendige Recherche um die Clusterware Agents zu verstehen und auch ein grundsätzliches Verständnis von RAC zu bekommen, benötigt doch einige Zeit.

Dieser Initial-Aufwand wurde allerdings auch belohnt, die Umgebung ist seit März 2015 in Betrieb und läuft seitdem völlig fehlerlos. Ganz klarer Vorteil der Umgebung ist auch, dass wirklich alle benötigten Komponenten innerhalb eines Clusters gemeinsam verwaltet werden und man sich möglichst wenige zusätzliche Abhängigkeiten einhandelt. Die Entscheidung für Oracle 12*c* und RAC erlaubt es zukünftig auch, zum Beispiel „Apex as a Service“ anzubieten. Durch die Multi-Tenant-Option können schnell zusätzliche Datenbanken angelegt werden, ohne zusätzliche Datenbank-Instanzen erstellen zu müssen, und mit RAC ist man in der Lage, flexibel auf gestiegene Anforderungen zu reagieren.

Moritz Klein
moritz.klein@mt-ag.com

Lizenzierung Forms 12*c*: Oracle bittet Stand-Alone-Kunden zur Kasse

DOAG Online

Kunden, die Forms mit einer „Forms and Reports“-Lizenz nutzen, müssen bei einem Upgrade auf Forms 12*c* die Repository-Datenbank nachlizenzieren. Als Stand-Alone-Version war zuvor keine Repository-Datenbank notwendig. Erst mit dem Release des Weblogic Server 12*c* wurde der Oracle Platform Security Service (OPSS), eine Layerschicht für Rollen und Authentifizierungen, in die Datenbank eingebettet. Da Forms den Enterprise Manager nutzt, der wie alle Anwendungen auf ADF-Basis auf die OPSS-Verwaltung zugreift, wird mit der Version 12*c* auch für Forms die Datenbank als Repository benötigt.

„Hier muss schleunigst nachgebessert werden“, fordert Michael Paege, Leiter Competence Center Lizenzierung bei der DOAG. Kunden, die Forms and Reports als Teil der Internet Application Server Enterprise Edition oder Weblogic Server Suite erworben haben, benötigen keine weitere Lizenz, sondern können die Repository-Datenbank als kostenfreie „Restricted Use“-Version nutzen. „Oracle sollte Kunden, die eine ‚Forms and Reports‘-Lizenz erworben haben, die benötigte Repository-Datenbank ebenfalls als ‚Restricted Use‘-Lizenz bereitstellen“, so Paege.

Auch Jan-Peter Timmermann, Leiter der Infrastruktur & Middleware Community, zeigt sich bestürzt: „Viele Kunden haben beim Umstieg von Forms 6*i* auf 10*g* aus gutem Grund die Stand-Alone-Version erworben – Oracle Portal und Discoverer wurden nicht benötigt, damit war dies der einfachste und kostengünstigste Weg, Forms im Web zu betreiben.“ Lediglich als Teil der Internet Applikation Server Enterprise Edition wurde damals für Forms eine Datenbank als Repository benötigt.