Nr. 3 | Juni 2014 | ISSN 09 36-0360 | www.doag.org

DOAG News

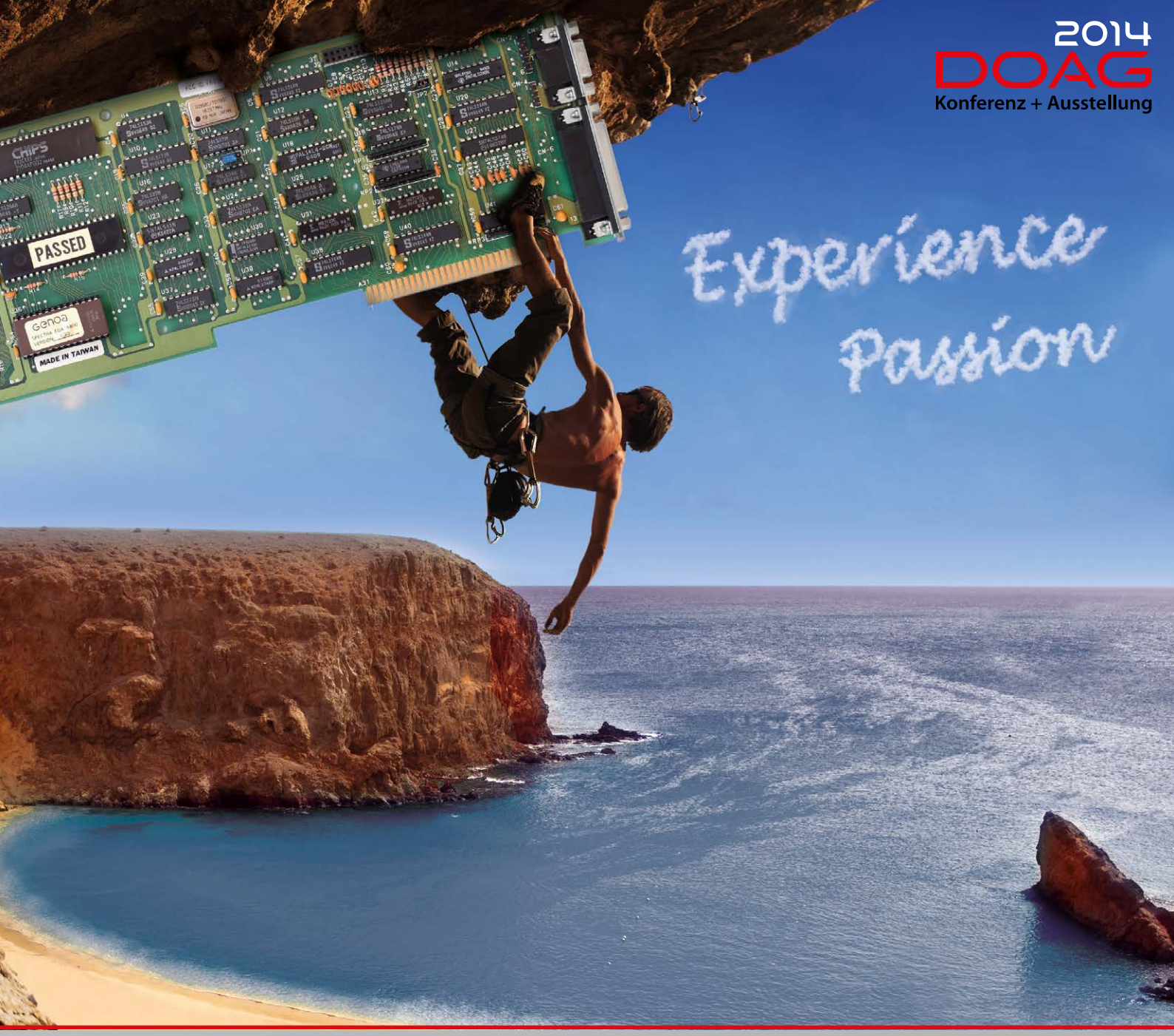
Deutsche ORACLE-Anwendergruppe e.V.

Datenbank-Cloning ohne Haareraufen

Entwicklung
Edition Based Redefinition
Continuous Integration

Im Interview
Axel vom Stein, Technischer Projektleiter

Datenbank
In der Cloud
Intelligentes Monitoring



2014
DOAG
Konferenz + Ausstellung

Der Hang, den Kernel der Sache zu treffen

18. - 20. Nov. 2014

Nürnberg NCC Ost

Eventpartner:

AOUG
AUSTRIAN ORACLE USER GROUP

SOUG
Swiss Oracle User Group

2014.doag.org

Christian Trieb
Leiter der Datenbank Community

Liebe Mitglieder,
liebe Leserinnen und Leser,

Datenbank-Cloning ist eine Thematik, mit der sich jeder Datenbank-Administrator früher oder später auseinandersetzen muss. Für die unterschiedlichsten Zwecke ist ein Abbild der Produktiv-Datenbank erforderlich, sei es für die Applikationsentwicklung oder zum Testen (funktional und/oder Performance), und auch der Datenbank-Administrator selbst benötigt Datenbank-Kopien für die unterschiedlichsten Aufgaben, etwa das Testen von Patch-Einspielungen.

So unterschiedlich die Anforderungen und so verschieden die Methoden zur Erstellung der Kopie auch sind, das Ziel ist meist, eine produktionsidentische Kopie der Datenbank zu erstellen. Die gängigsten Wege und Herausforderungen sind in dieser Ausgabe beschrieben. Das Klonen einer Datenbank erfordert natürlich den Dialog zwischen dem Auftraggeber und demjenigen, der ihn erstellen soll. So zeigt sich auch in dieser Aufgabenstellung, dass ein Datenbank-Administrator auch kommunikativ tätig sein muss. Es gilt, die Anforderungen kritisch konstruktiv zu hinterfragen, um so unter Umständen zu besseren Lösungen zu gelangen.

Immer wichtiger sind dabei auch die rechtlichen Rahmenbedingungen, in denen sich derjenige bewegt, der diese Aufgabe wahrnimmt. Dazu erstellt die DOAG momentan einen Security Guide, der demnächst exklusiv für DOAG-Mitglieder zur Verfügung stehen wird.

Ich wünsche Ihnen viel Spaß beim Lesen dieser Ausgabe sowie eine schöne Sommer- und erholsame Urlaubszeit.

Ihr

Das ist Database-Cloning

Development + Operation = DevOps

Forms2ADF mal anders: Wie aus einer Oracle-Vision Praxis wird

Einleitung

Datenbank

DevOps

Entwicklung

Aktuell

Tipps & Tricks

DOAG intern

Spotlight

Donnerstag, 3. April 2014

Die DOAG 2014 BI in München schlägt vor mehr als hundert Teilnehmern die Brücke zwischen relationaler Welt und Big Data. Während der fünften Auflage der BI-Veranstaltung kündigt Christian Weinberger seinen Rücktritt als DOAG-Themenverantwortlicher an und bedankt sich bei allen Referenten, Ausstellern, Teilnehmern und DOAG-Kollegen für die jahrelangen erfolgsgekrönten DOAG-Aktivitäten.

Montag, 7. April 2014

Die Collaborate 14, von den großen amerikanischen Usergruppen Independent Oracle Users Group (IOUG), Oracle Applications Users Group (OAUG) und Quest International Users Group in Las Vegas organisiert, zählt mit mehr als 5.500 Teilnehmern zu den weltweit größten Anwenderkonferenzen. Die DOAG ist durch drei Vorstandsmitglieder vertreten: Dr. Dietmar Neugebauer, Dr. Frank Schönthaler und Fried Saacke. Sie nutzen die Konferenz, um ihre bestehende Zusammenarbeit mit den internationalen Usergruppen durch persönliche Kontakte aufzufrischen, und sind wieder einmal vom hohen Bekanntheitsgrad der DOAG auf internationaler Ebene überrascht.

Freitag, 11. April 2014

Dr. Matthias Mann, Mitglied der Datenbank Community, referiert vor mehr als 40 DOAG-Mitgliedern das Webinar zum Thema „Proxy Authentication und Remote Login ohne sichtbare Passwörter". Die große Beteiligung zeigt, welchen hohen Stellenwert die Datensicherheit in der Community hat.

Dienstag, 29. April 2014

Das Berliner Expertenseminar findet wieder einmal mit Chris Antognini statt, diesmal zum Oracle Database Query Optimizer. Er erklärt weniger die internen Abläufe des Query Optimizer, sondern stellt vielmehr dessen wesentliche Merkmale vor, die ausschlaggebend für eine gute Performance sind. Die teilnehmenden Performance-Analysten, Anwendungs-Entwickler und Datenbank-Administratoren nehmen viele praktische Anregungen und Tipps für ihre tägliche Arbeit mit.

Dienstag, 29. April 2014

Nach dem Erfolg des ersten Treffens im Jahr 2014 beschäftigt sich die Regio NRW diesmal den kompletten Abend lang mit Apex. Erfahrene Anwender und Oracle-Experten berichten über ihre Erfahrungen mit dem Entwicklungstool.

Mittwoch, 7. Mai 2014

In seiner Eröffnungskeynote zur DOAG 2014 Logistik + IT schreibt Professor Michael ten Hompel, Leiter des Fraunhofer Instituts IML, der deutschen IT eine besondere Rolle zu. Beim abschließenden Rundgang durch die Räume des Fraunhofer Instituts kommen die Besucher ob der innovativen Experimentalprojekte zur künstlichen Intelligenz in der Intralogistik kaum aus dem Staunen heraus.

Mittwoch, 14. Mai 2014

Exadata-Experten aus dem ganzen Land geben sich beim DOAG 2014 Exaday im ISE Oracle Technology Center in Nürnberg ein Stelldichein. Günther Stürner von Oracle Deutschland zieht in seiner Keynote eine Bilanz über sechs Jahre Oracle Engineered Systems. Die anschließenden vielfältigen Erfahrungsberichte geben den mehr als 60 Teilnehmern einen guten Einstieg in die Welt des Exa-Stacks, den sie zum Abschluss vor Ort im Rechenzentrum der noris network AG live erleben können.*

Freitag, 16. Mai 2014

Ein wichtiger Tagesordnungspunkt der Vorstandssitzung ist die Umsetzung des Feedbacks aus der Delegiertenversammlung zu den Zielen 2016. Der Vorstand diskutiert sehr ausführlich das weitere Vorgehen zur themenorientierten Ausrichtung der DOAG. Ein weiterer wichtiger Punkt ist die Ausgestaltung der DOAG 2014 Konferenz mit der Festlegung von Schwerpunktthemen und Keynotes.

In eigener Sache:

DOAG-Fachzeitschriften im Web Alle Beiträge der DOAG-Fachzeitschriften sind seit vielen Jahren auch online verfügbar. Sie finden diese im Dokumentenarchiv. Ab dem Jahr 2014 können registrierte Mitglieder die kompletten Ausgaben auch als elektronisches Dokument unter www.doag.org/go/fachzeitschriften abrufen.

„Die neuen Funktionen begeistern mich schon, aber …“

In vielen von rund 600 Kundenprojekte sind die verschiedensten Datenbank-Versionen im Einsatz. Christian Trieb, Leiter der Datenbank Community, und Wolfgang Taschner, Chefredakteur der DOAG News, sprachen darüber mit Axel vom Stein, Technischer Projektleiter der BSS Materialflussgruppe.

Was sind bei Ihrem Unternehmen die größten Anforderungen an die IT?

vom Stein: Die BSS Materialflussgruppe ist im Anlagenbau und im Bereich der Intralogistik tätig und liefert ihren Kunden schlüsselfertige Anlagen und Gebäude. Intralogistik bedeutet hier unter anderem die Lagerung und bedarfsgerechte Bereitstellung von Material beispielsweise über ein Hochregallager. Die gesamte Abbildung der dahinterstehenden Prozesse (Materialfluss, Wareneingang, Warenausgang, Kommissionierung, Anbindung an ERP-Systeme etc.) erfolgt über eine Oracle-Datenbank. Unsere Kunden erwarten von uns deren höchste Verfügbarkeit und Performance zu moderaten Preisen. Deshalb scheuen viele Mittelstandsunternehmen den Einsatz der Enterprise Edition.

Wie haben Sie diese Anforderungen gelöst?

vom Stein: Wir betreiben für jeden unserer Kunden eine eigene Oracle-Datenbank. Der Kunde greift dann über ein Dialog-System auf seine Datenbank zu. Unsere Services sind über entsprechende Verträge für jeden einzelnen Kunden in unterschiedlichen SLA-Stufen geregelt. Mittlerweile haben sich so über die Jahre sehr viele verschiedene Kundenprojekte angesammelt. Darunter sind noch einige Datenbank-Versionen 8, etliche 9er-Versionen sowie zahlreiche Versionen 10 und 11 im Einsatz. Die Hardware steht in der Regel beim Kunden oder wird extern gehostet. Wir gehen dabei sehr flexibel auf die Wünsche unserer Kunden ein.

Wie können Sie die alten Datenbank-Versionen ohne Oracle-Support überhaupt noch sinnvoll betreiben?

vom Stein: Wir informieren unsere Kunden frühzeitig über den auslaufenden Oracle-Support und das damit verbundene Risiko und schlagen verschiedene Maßnahmen wie eine Migration auf eine aktuelle Version vor. Allerdings entscheidet ein Kunde gemeinsam mit uns oftmals, die bestehende Datenbank-Version weiter zu betreiben, sofern Änderungen an den laufenden Applikationen nur in geringem Umfang durchgeführt werden. Damit ist das Risiko eines instabilen Systems minimiert.

Wie führen Sie dann anfallende Migrationsprojekte durch?

vom Stein: Jedes unserer Kundenprojekte ist individuell zu betrachten. Von daher können wir keine Standardmigration durchführen, sondern müssen jeden Fall einzeln und mit dem Kunden im Detail absprechen. Oft bietet sich auch eine Migration an, wenn das System auch auf der Applikationsseite erweitert werden soll; das kann auch den Austausch der Hardware beinhalten.

Worin sehen Sie die Stärken neuerer Datenbank-Versionen?

vom Stein: Beispielsweise machen die SQL-Erweiterungen mit den Pivot-Tabellen in der Version 11 für unsere Entwickler vieles einfacher. Ein anderes Beispiel ist die Data Pump. Auch die Anbindung vieler externer Geräte wie Steuerungen oder Drucker können wir mit TCP/IP direkt aus der Datenbank heraus realisieren.

Haben Sie sich schon mit der Version 12c beschäftigt?

vom Stein: Ja. Die neuen Funktionen begeistern mich schon, aber generell habe ich ein Problem damit, dass neue Features oftmals nur in kostenpflichtigen Optionen der Enterprise Edition verfügbar sind. Damit entstehen dann deutlich mehr Lizenzkosten.

Christian Trieb (links) im Gespräch mit Axel vom Stein

Wie erledigen Sie die Backups für die Datenbanken?
vom Stein: Wir setzen seit der Version 9 auf RMAN.

Wie stellen Sie die Hochverfügbarkeit der Datenbanken sicher?
vom Stein: In etlichen Fällen setzen wir mit sehr guten Erfahrungen hinsichtlich Stabilität und Installation Oracle Fail Safe mit einem Microsoft-Cluster ein. Auf Wunsch kann dies auch in Kombination mit einer Standby-Datenbank betrieben werden, im Bereich der Enterprise Edition mit einem Data Guard. Bei der Standard Edition erfolgt der Abgleich zwischen den Datenbanken automatisiert mittels einer Eigenentwicklung.

Wie beurteilen Sie den Oracle-Support bei Problemfällen?
vom Stein: Es gibt hier sowohl gute als auch schlechte Erfahrungen. Die Support-Mitarbeiter sind sehr hilfsbereit und freundlich. Manche Anfragen wurden sehr schnell beantwortet, einmal hatte ich allerdings den Fall, dass der Support-Mitarbeiter mein Problem überhaupt nicht verstanden hat und nur ein aufwändiger Workaround helfen konnte.

Wie gehen Sie beim Einspielen von Patches vor?
vom Stein: Wir stellen für fast alle Kunden eine Test-Umgebung bereit, in der wir den Patch zunächst einspielen und prüfen können. Gerade in der Logistik finden wir aber in der Regel komplett abgeschlossene Systeme vor, sodass wir nicht jeden Patch zwingend anwenden müssen.

Der Einsatz von Microsoft kommt in dieser Breite sicher nicht sehr häufig vor. Wie sind hier Ihre Erfahrungen?
vom Stein: Unsere Erfahrungen sind mittlerweile sehr positiv. In den Ursprüngen mit Windows NT und Windows 2000 gab es natürlich noch große Probleme; seit Windows Server 2008 R2 spricht aber nichts mehr dagegen, Oracle in Verbindung mit Windows einzusetzen.

Benutzen Sie Tools zum Administrieren der Datenbank?
vom Stein: Ein zentraler Enterprise Manager mit Zugriff auf alle Datenbanken funktioniert leider nicht, da wir zahlreiche unterschiedliche VPN-Verbindungen zu unseren Kunden unterhalten. Wir setzen deshalb das Monitoring-Tool von Herrmann & Lenz ein. Der Kontakt entstand übrigens auf einer DOAG-Veranstaltung.

Können Sie sich vorstellen, für Ihre Anwendung ein Engineered System von Oracle einzusetzen?
vom Stein: Es gibt schon Überlegungen in dieser Richtung, aber selbst unsere Großkunden bevorzugen momentan die Microsoft-Lösung, zumal die Investition in ein Engineered System sehr hoch ist.

Welche Produkte der Firma Oracle setzen Sie sonst noch ein?
vom Stein: Wir setzen aktuell nur die Datenbank selbst ein.

Was erwarten Sie von einem IT-Unternehmen wie Oracle?
vom Stein: Ich würde mir wünschen, dass Oracle künftig auch Optionen bei der Standard Edition einführt, beispielsweise Data

Zur Person: Axel vom Stein
Axel vom Stein hat im Jahr 2001 sein Studium mit Diplom in Mathematik mit Nebenfach Informatik abgeschlossen. Er ist seitdem für die BSS Materialflussgruppe tätig, zunächst als Projektingenieur und ab dem Jahr 2007 als Technischer Projektleiter. Der Schwerpunkt seiner Tätigkeiten hat sich im Laufe der Zeit vom reinen Entwickler (unter anderem PL/SQL, C, C#, Delphi) in Richtung Datenbank-Administration und Technische Projektleitung verschoben. Er ist verantwortlich für die Standardisierung der BSS-Software und die Migrationsprojekte. Zudem plant und realisiert er Sicherheits- und Hochverfügbarkeitskonzepte. Seine besonderen Interessen liegen auf der Oracle Standard Edition, Oracle auf Windows sowie in agilen Software-Methoden.

Wichtig ist die Abschätzung der Kompressionsrate

Guard, damit auch der Mittelstand diese Funktionen zu vernünftigen Kosten erhalten kann. Darüber hinaus gibt es einige Dinge wie beispielsweise die Erweiterung des UTLTCP-Packages um Socket-Server-Routinen oder das Auslesen eines Directory, die ohne Umwege direkt aus der Datenbank heraus funktionieren sollten. Ein weiterer Wunsch wäre der offizielle Support vieler Underscore-Parameter.

Wie sehen Sie den Stellenwert einer Anwendergruppe wie der DOAG?

vom Stein: Die DOAG besitzt für mich einen sehr hohen Stellenwert. Ich bin per Zufall im Jahr 2003 im Internet auf einen Artikel gestoßen, in dem auf ein Regionaltreffen der DOAG hingewiesen wurde. Seitdem besuche ich die Veranstaltungen regelmäßig sowohl für Vorträge als auch den Austausch mit anderen Anwendern. Zudem erweitert die DOAG ihr Angebot laufend, wie z.B. die Webinare oder Fachkonferenzen, so dass auch mein Arbeitgeber von der Mitgliedschaft überzeugt ist und diese unterstützt.

Die BSS Materialflussgruppe

Im Jahr 1991 gegründet zählt die BSS Materialflussgruppe heute zu den weltweit wenigen Unternehmen, die Komplettlösungen der Intralogistik wie automatische Logistikanlagen, Hochregallager, Sortiersysteme, Kommissionierlösungen oder auch komplette Logistik- und Verteilzentren als Generalunternehmer für Industrie und Handel planen und realisieren. Die Bandbreite reicht dabei von einfachen, teilautomatisierten Systemen bis hin zu komplexen Hochleistungsanlagen. Die BSS Materialflussgruppe ist in den letzten Jahren stetig gewachsen und beschäftigt derzeit etwa 150 Mitarbeiter. Mit den selbst entwickelten und zertifizierten Softwaresystemen für Verwaltung, Steuerung und Organisation werden die kompletten kundenspezifischen Logistikprozesse abgebildet. Die Systeme sind modular aufgebaut und an alle marktüblichen Systemumgebungen ankoppelbar.

800 Java-Enthusiasten füllen JavaLand mit Leben und Ideen

Unter dem Motto „Zwei Tage lang das JavaLand besiedeln“ bot das JavaLand 2014 ein innovatives Konzept für die Teilnehmer. Die von der DOAG Dienstleistungen GmbH ausgerichtete und gemeinsam mit dem Heise Zeitschriften Verlag präsentierte Veranstaltung stellte der Java-Community einen attraktiven Rahmen zum Lernen, Erfahren und Austauschen zur Verfügung, der zusammen mit den mittlerweile 21 unter dem Dach des Interessenverbund der Java User Groups e.V. (iJUG) vertretenen Java User Groups gestaltet wurde.

Neben mehr als 100 Vorträgen in sieben parallelen Streams gab es die ganze Zeit über im Hackergarten ein Labor zum Experimentieren sowie ein Ort für Diskussionen und zum Kennenlernen. Unter dem Motto „NightHacking” übertrug Stephen Chin, Java Technology Ambassador für Oracle, Live-Interviews aus dem JavaLand. Arun Gupta, Direktor Developer Advocacy bei Red Hat, arbeitete mit den Teilnehmern an kürzlich nach GitHub umgezogenen Java-EE-7-Beispielprojekten.

Im Java Innovation Lab präsentierten Java-Entwickler innovative Projekte, deren Fokus die reale mit der virtuellen Welt verbindet. Daneben wurden die Community-getriebenen Programme „Adopt a JSR“ und „Adopt OpenJDK“ vorgestellt und es gab Gelegenheit zum Ausprobieren mit den Java-Experten Ed Burns, Mani Sarkar, Anatole Tresch und Daniel Bryant. Unter dem Motto „Lambdafy Your Project“ konnten die Teilnehmer anhand von Beispielen erlernen, wie die Umstellung auf Lambda-Ausdrücke in Java 8 funktioniert. Darüber hinaus wurden mit „Code Katas“ in kleinen Teams überschaubare Programmieraufgaben gelöst. Ein weiteres Highlight der Veranstaltung war die Übertragung des Java-8-Lauch im Rahmen des Abendevents.

„Der große Erfolg der Konferenz ist das Ergebnis der gemeinsamen Arbeit aller Beteiligten“, sagt Fried Saacke, DOAG-Vorstand und Geschäftsführer sowie Vorstandsvorsitzender des iJUG. „Wir freuen uns schon auf eine Fortsetzung im kommenden Jahr.“

Das ist Database-Cloning

Johannes Ahrends, CarajanDB GmbH

Viele von uns sind mit der Filmserie „Star Wars“ aufgewachsen und haben schon diverse Schlachten mit Clone-Kriegern von Lego geschlagen. Das Schaf „Dolly“ hat bereits in den 1990er Jahren für Aufregung gesorgt und die laut Wikipedia „durch ungeschlechtliche Vermehrung entstandene Nachkommenschaft eines Lebewesens“ ist zumindest für Menschen in Deutschland verboten. Der Hintergedanke beim Klonen ist jedes Mal derselbe: Es sollen eine oder mehrere Kopien einer Sache hergestellt werden, die die gleichen Eigenschaften wie das Original besitzen. Das gilt auch für Datenbanken und ist prinzipiell nichts Neues. Wenn es da nicht zwei wichtige Punkte gäbe: Größe und Anzahl von Datenbanken nehmen immer mehr zu.

Kopien von Datenbanken gibt es, seit diese existieren, sei es per RMAN Duplicate, Standby Database oder auf Storage-Ebene über Mirroring. Das Verfahren ist einfach und schnell und wird seit vielen Jahren erfolgreich genutzt. Allerdings kommen wir mehr und mehr an die Grenzen dieser Verfahren, denn es ist nicht selten, dass 20 oder mehr Kopien einer Datenbank für Test und Entwicklung zur Verfügung stehen müssen. Bei Datenbanken, die mehrere Terabyte groß sind, bedeutet dies, dass Unsummen für Plattenspeicher anfallen.

Aber werden diese Datenbanken überhaupt genutzt? In vielen Fällen benötigen die Entwicklungsteams nur einen Bruchteil der Datenbank; manchmal werden zwar viele Daten gelesen, jedoch nur sehr wenige für die neue Entwicklung geändert. Das bedeutet, dass in der Regel 90 Prozent der kopierten Daten gar nicht erforderlich sind und weniger als 5 Prozent tatsächlich geändert werden. Außerdem müssen Datenbanken immer schneller provisioniert beziehungsweise aktualisiert werden. Der Stand der Produktion von vor einem Monat ist für die Entwicklung eventuell noch akzeptabel, für einen Qualitätstest allerdings schon nicht mehr. Das stellt uns vor folgende Herausforderungen:

- Viele Datenbanken aufzubauen, die wenig Speicherplatz benötigen
- Datenbanken schnell zur Verfügung zu stellen
- Möglichst den DBA von dieser Aufgabe zu entlasten, also den Verantwortlichen selbst die Möglichkeit zu geben, solche Datenbanken zu erstellen,

Auch wenn Oracle mit RMAN Duplicate einen einfach zu bedienenden Befehl für das Klonen von Datenbanken zur Verfügung stellt, reicht das für diese Anforderungen nicht aus:

- Es werden zu viele/alle Daten kopiert
- Bei einer Datenbankgröße von einigen Terabyte dauert das etliche Stunden
- Das ist für den Anwender unzumutbar

Als Alternative bietet sich zunächst Data Pump an, das einige der genannten Anforderungen erfüllt:

- Es können Teilmengen übertragen und die gleichen Exports für mehrere Fachbereiche genutzt werden
- Die Teilmengen reduzieren die Dauer für die Erstellung

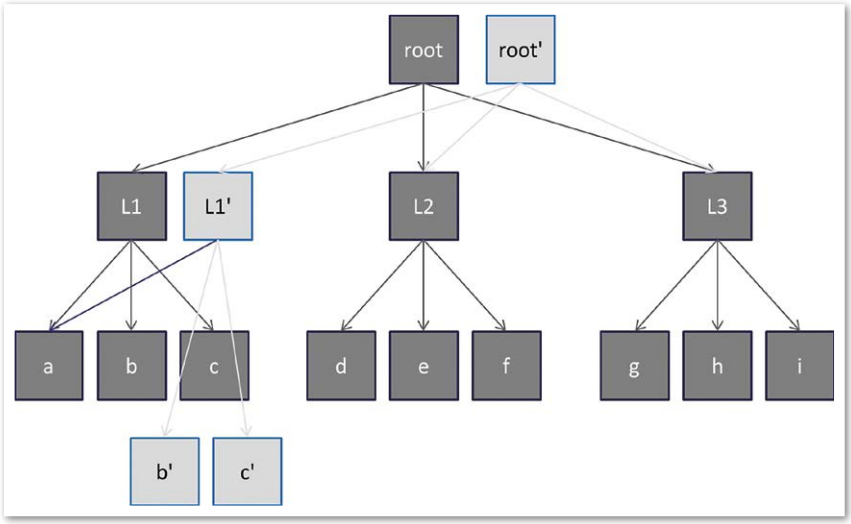


Abbildung 1: Schema des „Copy On Write“

- Wenn schon eine Datenbank existiert, kann der Ex-/Import per Skript erfolgen, sodass sich die Fachabteilung selbst bedienen kann

Das Verfahren wird heute in vielen Unternehmen für den Aufbau von Entwicklungsdatenbanken benutzt, hat aber folgende Einschränkungen:

- Es ist schwierig, die Teilmengen klar zu definieren. Dadurch wächst die Gefahr der Inkonsistenz.
- Auch wenn 90 Prozent der Daten nicht benötigt werden. Wer sagt, welche das sind?
- Die Teilmenge eignet sich nicht für Qualitätstests beziehungsweise Performance-Analysen.
- Der DBA muss zumindest die Datenbank vor dem Import erstellen und dafür sorgen, dass die alten Daten gelöscht sind. Außerdem ist er für den korrekten Export verantwortlich.

Snapshots und Copy-On-Write

Beim Namen „Snapshot“ denkt man oft an Virtualisierungslösungen wie VMware. Dabei wird zu einem bestimmten Zeitpunkt ein Snapshot erstellt, um bei Bedarf das Gastsystem auf diesen Snapshot zurückzusetzen. VMware verwendet dafür ein „Copy On Write“, es werden also für eine bestehende virtuelle Disk (zum Beispiel „disk1.vmdk“) eine neue Version (zum Beispiel „disk1-0001.vmdk“) angelegt und alle Änderungen in dieser Datei gespeichert. Jetzt kann der Gast jederzeit auf den alten Stand zurückgesetzt werden oder der alte Stand wird durch das Löschen des Snapshots auf den aktuellen Stand vorgerollt.

Ähnlich verhält es sich mit Oracle ZFS und „btrfs“-Filesystemen. Allerdings steht dabei nicht die Möglichkeit des Zurücksetzens im Vordergrund, sondern es geht darum, eine gemeinsame Basis der Daten zu haben, auf der beliebig viele Snapshots aufsetzen können. Es gibt also eine Quelle (das Original), auf die lesend zugegriffen wird (entsprechend „disk1.vmdk“ bei VMware), und mehrere unabhängig existierende Dateien, in die die Änderungen geschrieben werden. Auf die Oracle-Datenbank übertragen würde das bedeuten, dass die Datafiles in einen lesenden sowie einen schreibenden Teil aufgeteilt sind und das Filesystem die I/Os entsprechend verteilt. Es gibt unterschiedliche Methoden, wie ein solches Filesystem aufgebaut sein kann, wobei es sich immer um eine Art Indizierung der Blöcke (beispielsweise Filesystem- oder Oracle-Block) handelt *(siehe Abbildung 1)*.

Es wird ähnlich einem „B*Tree“-Index zunächst eine Struktur angelegt, in der die Blöcke referenziert sind. Alle Anwendungen lesen, ausgehend vom „Root“-Block über die Verzweigungen (L1 … L3) die Blätter (a … i), in denen die Zeiger zu den tatsächlichen Datenblöcken stehen. Wenn man jetzt Datenblöcke ändert, entsteht ein neuer Stamm („Root“-Verzweigungen), der auf diese sowie die entsprechenden nicht geänderten Blöcke verweist. Eine Anwendung, die also einen neueren Zustand der Blöcke (Snapshot) benötigt, greift jetzt auf „Root“‘ zu und über L1‘, L2 und L3 auf a, b‘, c‘, d … i.

Copy-On-Write für Datenbanken

Um dieses Verfahren für die Oracle-Datenbank zu nutzen, wird man zunächst sicherlich nicht die Produktionsdatenbank als Quelle nutzen; als Erstes wird also eine konsistente Kopie der Datenbank benötigt. Es liegt nahe, mit RMAN ein Backup

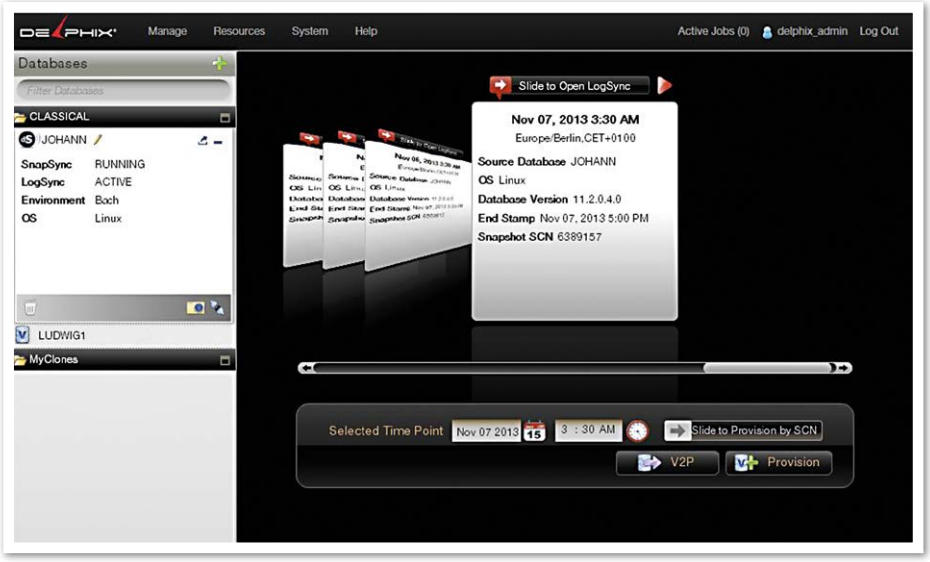


Abbildung 2: Delphix in Aktion

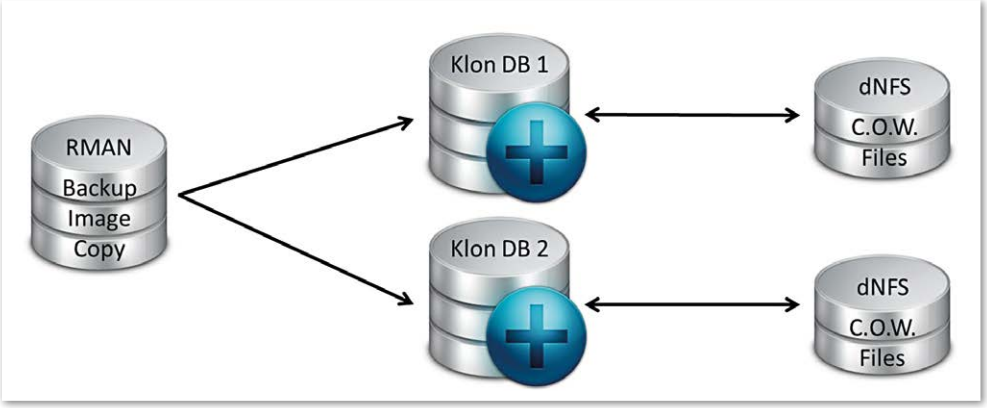


Abbildung 3: Das Prinzip von CloneDB

durchzuführen. Aus diesem konnte man bisher auch mit dem Befehl „DUPLICATE DATABASE" beliebig viele Datenbanken erstellen – aber das war nicht das Ziel: Das Backup selbst sollte die Datenbank sein. Denn nur dann lassen sich der Speicherverbrauch reduzieren und neue Datenbanken schnell aufbauen.

Die Frage ist also: „Wie stelle ich das Backup gleichzeitig mehreren Datenbanken beziehungsweise Servern zur Verfügung?" Egal ob Unix oder Linux, die Lösung ist bekannt und heißt „NFS". In einem derzeit noch andauernden Proof of Concept hatte der Autor die Möglichkeit, zwei Lösungen näher unter die Lupe zu nehmen: Delphix und CloneDB. Die Multitenant-Option von Oracle 12*c* bietet noch eine weitere Lösung, die nicht auf RMAN basiert. Dazu später mehr.

Delphix

In den USA hat sich seit einigen Jahren das Unternehmen „Delphix" mit der gleichnamigen Software einen Namen gemacht *(siehe Abbildung 2)*. Das Unternehmen bietet Copy-On-Write unter anderem auch für Oracle-Datenbanken an. Als Quelle beziehungsweise Startpunkt dienen ein normales RMAN-Backup sowie die archivierten Redolog-Dateien. Für die Clones steht ein spezielles Filesystem (DxFS) zur Verfügung, in dem die Daten-, Redolog-, Temp- und archivierten Redolog-Dateien liegen. Beim Starten der Datenbank-Instanz werden diese Filesysteme eingebunden und die Dateien als normale Datenbank-Dateien sichtbar. Das Copy-On-Write wird intern geregelt, man bekommt also von der ganzen Snapshot-Technologie nichts mit.

Die Technologie erweist sich als ausgereift, und wenn man die Delphix-Appliance einmal installiert hat, ist der Aufbau ausgesprochen einfach. Über eine grafische Oberfläche lassen sich sehr schnell auch von Nicht-DBAs (fast) beliebig viele Klone erstellen, die dann automatisch als Oracle-Datenbanken gestartet werden. Vorteil von Delphix ist, dass zusätzlich zum RMAN-Backup in Intervallen Snapshots der Quelldatenbank erstellt werden, sodass es möglich ist, einen Klon zu jedem beliebigen Zeitpunkt der Quelldatenbank aufzubauen und dann vor- oder zurückzurollen.

CloneDB

Oracle hat mit der Version 11.2.0.2 relativ unauffällig das Feature „CloneDB" auf den Markt gebracht. Auch dieses basiert auf einem RMAN-Backup (in diesem Fall allerdings als „Datafile Copy") und erlaubt es, eine beliebige Anzahl von Datenbanken auf dieser einen Kopie aufzubauen. Die geänderten Daten werden ebenfalls in entsprechende Datafiles geschrieben. Einzige Voraussetzung ist die Nutzung von Direct NFS (dNFS) für die „Client"-Datenbanken *(siehe Abbildung 3)*.

Zwar war die Dokumentation zu Beginn des Tests (Ende 2013) noch sehr übersichtlich; wenn man sich allerdings einmal intensiver mit dem Aufbau von Direct NFS beschäftigt hat, lassen sich Snapshots innerhalb weniger Minuten erstellen. Im Gegensatz zu Delphix muss man sich hierbei aber um das Starten beziehungsweise die Einstellung der Snapshot-Datenbank selbst kümmern.

Oracle 12*c*

Natürlich bleibt Oracle an dieser Stelle nicht stehen. Die Multitenant-Option der Datenbank 12*c* bietet jetzt zusätzlich die Möglichkeit eines „Snapshot Clone" für Pluggable Databases. Anstelle der RMAN-Kopie fungiert hier eine bereits existierende Pluggable Database als Quelle beziehungsweise lesende Version der Datenbank und dementsprechend können bis zu 251 weitere Snapshot-Klone angelegt werden. Voraussetzung hierbei ist die Verwendung eines Filesystems, das Snapshot Copy unterstützt (entweder vom Storage-Hersteller, etwa ZFS, oder ACFS). Diese Methode ist sicherlich von den vorgestellten Verfahren die unkomplizierteste, da man sich durch die Verwendung der Multitenant-Option nicht um die Parametrierung beziehungsweise das Starten der Datenbank-Instanz kümmern muss.

Compliance

Das hört sich alles ganz gut an – und funktioniert übrigens auch ganz hervorragend. Allerdings stellt sich die Frage, ob man solche Kopien überhaupt anlegen darf. Es sollte selbstverständlich sein, dass man Daten der Produktion nicht einfach kopieren und schon gar nicht für Entwicklungssysteme nutzen kann. Insofern ist die ursprüngliche Definition von Klonen als „identische Kopie" gar nicht das, was wir haben wollen. Wichtig ist, dass es ein Data-Masking-Verfahren für die Kopie gibt. Eine Herausforderung, die gar nicht so leicht zu lösen ist, denn damit wird ja schreibend auf den Klon zugegriffen und die ganze Technik ad absurdum geführt.

Delphix bietet daher eine Möglichkeit an, die Daten auf dem Weg zum Klon zu maskieren. Bei CloneDB und Snapshot Clone sind dem Autor solche Verfahren nicht bekannt. Als Alternative bietet sich hier an, eine Kopie der Datenbank zu erstellen, die mithilfe der Oracle-Data-Masking-Option gesäubert und dann als Quelle für die Klone genutzt wird.

Fazit

Generell können die genannten Verfahren den Storage-Bedarf von Datenbank-Kopien drastisch reduzieren. Delphix spricht davon, dass nur noch 5 Prozent der ursprünglichen Ressourcen erforderlich sind. Das ist sicherlich in einigen Umgebungen ein durchaus realistischer Wert. Allerdings eignen sich die hier vorgestellten Verfahren nicht für Performance-Tests, da der zusätzliche I/O die Messungen verfälschen würde.

Die Multitenant-Option eröffnet sicherlich noch weitere Möglichkeiten des Cloning, allerdings beschränkt sich dieses Verfahren auf Datenbanken, die auf einem gemeinsamen Server (natürlich auch als RAC) betrieben werden. Die anderen vorgestellten Lösungen sind mit den aktuellen Oracle-Versionen einsetzbar und können in Zukunft helfen, schneller, einfacher und vor allen Dingen in größerem Maße Datenbanken für Test und Entwicklung zur Verfügung zu stellen.

Johannes Ahrends
johannes.ahrends@carajandb.com

Schnelles Datenbank-Cloning mit Snapshots – ein Überblick

Manuel Hoßfeld, ORACLE Deutschland B.V. & Co. KG

Inhaltlich identische Kopien („Klone“) von Datenbanken sind für verschiedene Zwecke praktisch oder sogar notwendig – sei es als Kopie einer Produktionsdatenbank für Entwicklungszwecke oder als Basis von Tests, die man nicht an einer Produktionsdatenbank durchführen kann oder will.

Normalerweise erfordert das Anlegen eines Datenbank-Klons das vollständige Kopieren aller zum Original gehörenden Datenbank-Dateien. Es gibt inzwischen jedoch verschiedene alternative Möglichkeiten, Datenbanken auch über Snapshot-Technologien zu duplizieren, ohne dass tatsächlich ein physischer Kopiervorgang anfällt. Dieser Artikel zeigt, warum dies sinnvoll ist, und gibt einen Überblick über einige der möglichen Varianten. Da es sich hier nur um eine Übersicht handelt, erhebt er keinen Anspruch auf Vollständigkeit oder eine technisch erschöpfende Darstellung. Für tiefergehende Informationen empfiehlt sich daher der Blick in weiterführende Artikel oder in die jeweilige Dokumentation.

```
create pluggable database
testklon-db from prod-db snapshot copy;
```

Listing 1

Das vollständige Kopieren einer Datenbank mit traditionellen Methoden wird gelegentlich auch als Erzeugen einer „full copy“ oder „fat copy“ bezeichnet. Je nach Größe der Datenbank und Geschwindigkeit des zugrunde liegenden Storage-Systems kann dieser Vorgang sehr zeitraubend sein. Auch ist das Kopieren einer kompletten Datenbank in der Regel nicht sehr effizient, wenn man bedenkt, dass für die meisten Anwendungsfälle nur sehr wenige Änderungen oder neue Daten in der geklonten Datenbank anfallen werden.

Viel sinnvoller erscheint es daher, einen vollständigen Kopiervorgang zu vermeiden, indem man sich die Eigenheiten von Storage-Snapshots zunutze macht. Man könnte dabei von einer logischen Kopie (auch „thin copy“ oder „thin clone“) sprechen. Vorteile dabei sind vor allem der

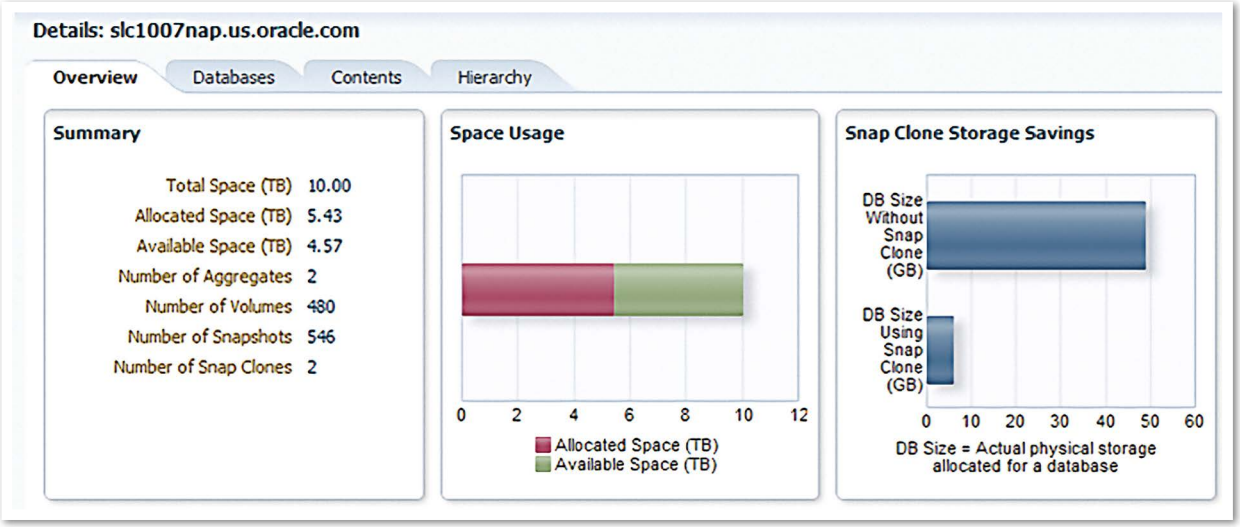


Abbildung 1: Ansicht eines für Snap Clones registrierten Storage-Systems in Enterprise Manager 12c

wesentlich geringere Zeitbedarf und auch eine größere Storage-Effizienz. Die Ersparnis an Zeit und Storage-Platz steigt mit der Größe der jeweiligen Datenbank beträchtlich an. Andersherum ausgedrückt: Je größer eine zu klonende Datenbank ist, desto lohnender ist es, sich mit den Techniken in diesem Artikel zu befassen. Konkret wurden zum Beispiel Zeiteinsparungen im Bereich von 80 bis 90 Prozent sowie Storage-Einsparungen von bis zu 99,9 Prozent beobachtet *[1]*. Dieser Artikel berücksichtigt folgende Verfahren zum Erstellen von Snapshot-basierten Datenbank-Klonen:

- Manuell/mit Skripten
- Snap-Management-Utility
- Snap Clone in Enterprise Manager Cloud Control
- Snapshot-Copy-Klausel beim Klonen von Pluggable Databases

Am Ende des Artikels sind diese Verfahren in einem tabellarischen Fazit mit ihren jeweiligen Vor- und Nachteilen gegenübergestellt.

Technische Grundlagen und Voraussetzungen

Grundlage aller in diesem Artikel genannten Methoden sind die von modernen Storage- und/oder File-Systemen bereitgestellten „Copy on Write"-Snapshots (CoW). Bei diesem Verfahren werden beim Anlegen eines Snapshot die entsprechenden Daten nicht wirklich kopiert (also physisch verdoppelt), sondern es wird, vereinfacht gesagt, lediglich ein neuer Eintrag angelegt, der auf die gleichen Daten verweist. Erst wenn an dem Snapshot Daten geändert werden oder neue hinzukommen, werden die entsprechenden Datenblöcke im Snapshot gesichert beziehungsweise die neuen geschrieben. Da nennenswerte I/O-Operationen erst beim Schreiben oder Ändern von Daten anfallen, erfolgt das Anlegen eines solchen Snapshot extrem schnell.

CoW-Implementationen sind auf unterschiedlichsten Ebenen zu finden: Sie können im Filesystem, im Volume Manager oder auf Storage-Ebene angelegt sein und für ein ganzes Volume, Filesystem oder auch nur einzelne Dateien zur Verfügung stehen. Filesysteme, die entsprechende Features anbieten, sind zum Beispiel ZFS, btrfs, ACFS und OCFS2. Entsprechende Storage-Systeme gibt es zum Beispiel von Oracle oder NetApp.

Hinweis: Je nach konkreter Implementierung ist entweder der Snapshot als solches bereits beschreibbar oder aber es muss auf Basis dessen zunächst ein Klon des Snapshot im Storage beziehungsweise Filesystem erzeugt werden, um darin enthaltene Daten ändern zu können oder neu zu schreiben. Die Nomenklatur ist daher bei der Verwendung der Begriffe „Snapshot" und „Clone" nicht immer eindeutig. Der Einfachheit halber wird im Folgenden auf diese Unterscheidung verzichtet und grundsätzlich von beschreibbaren Snapshots/Clones ausgegangen.

Klonen mit selbst erstellten Skripten

Schon seit geraumer Zeit ist es möglich, sich komplett von Hand einen Mechanismus (zum Beispiel ein Skript) zu erstellen, der die (physikalische) Kopie eines konventionellen Datenbank-Klons durch einen Storage-Snapshot der Datenbank ersetzt. Es sind dazu natürlich alle notwendigen Schritte zu berücksichtigen. Konkret handelt es sich im Wesentlichen um:

- Herunterfahren oder zumindest „Anhalten" der Datenbank (mittels „...begin backup")
- Erzeugen des Snapshot durch Aufruf entsprechender Storage-Befehle
- Anlegen beziehungsweise Kopieren der notwendigen Datenbank-Metadaten und -Konfiguration
- Konfigurations-Anpassung (wie Umbenennungen zur Vermeidung von Namenskollisionen)

Da die Einrichtung und Pflege eines solchen Verfahrens relativ mühsam und zeitraubend sein kann, wird es an dieser Stelle auch nicht näher betrachtet. Allen Lesern, die dennoch an einer individuellen, skriptbasierten Lösung interessiert sind, sei an dieser Stelle eine Variante empfohlen, die die mit Oracle 11.2.0.2 als Bestandteil von Direct NFS (dNFS) eingeführte „clonedb"-Funktionalität nutzt. Hierbei implementiert der dNFS-Stack der Datenbank selbst einen CoW-Mechanismus auf Basis einer

Verfahren	Unterstützte Storage- & Filesysteme	Unterstützte Datenbank-Versionen	Benötigtes technisches Know-how	Grafische Oberfläche	Zusätzliche Software-Installation erforderlich	Produkt/ supportete Lösung
Eigene Skripte (z.B. mittels dNFS-Clone)	Nahezu beliebig (btrfs, OCFS2, ZFS, ...)	ab 11.2.0.2	hoch	nein	nein	nein
SMU	Nur ZFSSA	(10*g**), 11*g*	gering**	ja	ja	ja
Snap Clone	ZFSSA, NetApp, generischer ZFS-Storage	(10g*), 11*g*, 12*c*	gering**	ja	ja	ja
Snapshot-Copy für PDBs	ZFSSA, NetApp, ACFS	Nur 12c mit Multitenant Option	mittel	Je nach Aufruf/ Nutzung von SQL	nein	ja

Tabelle 1

** Oracle 10g würde prinzipiell/technisch auch unterstützt, ist aber nicht mehr im aktiven Support*
*** Initiale Einrichtung kann Hilfe von Storage-Administrator erfordern*

zuvor entweder physisch kopierten oder als Snapshot (zum Beispiel auf btrfs) erzeugten Datenbank-Kopie. Beschrieben ist dies unter anderem in einigen Blog-Postings und Artikeln, so zum Beispiel in der DBA Community *[2]* sowie in einem auf der letztjährigen DOAG-Konferenz gehaltenen Vortrag *[3]*.

Snap-Management-Utility für Oracle ZS3/ZFSSA

Beim sogenannten „Snap Management Utility" (SMU) handelt es sich um ein von Oracle angebotenes Software-Werkzeug, das die Erstellung von Snapshot-basierten Datenbank-Klonen auf Storage-Systemen der ZS3-Reihe (auch bekannt als „ZFS Storage Appliance") bietet. Es kapselt die Möglichkeiten dieser Systeme zum Erstellen von Snapshot-basierten Datenbank-Klonen in einer einfach zu bedienenden grafischen Management-Konsole. Ziel ist es, dass DBAs dadurch auch ohne genaue Kenntnisse der zugrunde liegenden Storage-Mechanismen schnell und einfach DB-Klone erstellen können. Nähere Informationen zum SMU finden sich im Artikel in dieser Ausgabe auf Seite 18 sowie unter *[4]* und *[5]*.

„Snap Clone"-Feature in Enterprise Manager 12*c* Cloud Control

Enterprise Manager 12c Cloud Control bietet im Rahmen des „Cloud Management Packs for Oracle Databases" die Möglichkeit, neue, aber bereits mit Inhalten versehene Datenbanken durch einen User im Self-Service anzufordern. Die Datenbanken können hierbei nicht nur konventionell erzeugt werden (also durch physisch kopierte Datenbank-Dateien), sondern auch auf Basis von Storage-Snapshots. Dieses Feature wird als „Snap Clone" bezeichnet. Konkret unterstützt werden dafür zurzeit:

- Systeme der ZS3-Serie
- NetApp Filer
- Generische Storage-Systeme (SAN/NAS), die als ZFS-Dateisystem von einem Oracle Solaris Server (ab Solaris 11.1) bereitgestellt sind

Als Grundvoraussetzung für diese Funktionalität muss der jeweilige Storage einmalig in Enterprise Manager registriert sein. Dazu ist zuvor sowohl das Plug-in für das „Storage Management Framework" sowie das Plug-in für das jeweilige Speichersystem zu installieren. Damit dies auch funktioniert, sind natürlich auf dem Storage-System noch ein paar Handgriffe durch dessen Administrator erforderlich, um die entsprechenden Storage-API-Funktionen auch einzurichten beziehungsweise freizuschalten. Die genauen Schritte sind in der Enterprise-Manager-Dokumentation beschrieben *[6]*.

Sobald die genannten Voraussetzungen erfüllt sind, kann man im „Database as a Service"-Bereich von Cloud Control an zwei Stellen von Snap Clones profitieren. Zum einen ist es möglich, auf Basis von Storage-Snapshots sogenannte „Database Profiles" anzulegen, die eine bereits existierende Datenbank repräsentieren, um diese später sehr schnell als Klon bereitzustellen. Wenn ein Cloud -Administrator also bei der Definition eines „Service Templates" für eine Datenbank ein solches Snapshot-basiertes Profil ausgewählt hat, erfolgt später für einen Nutzer, der das entsprechende Template im Self-Service-Portal auswählt, kein Kopiervorgang mehr. Stattdessen wird die Datenbank als „Snap Clone" provisioniert, was unabhängig von der Größe des Originals immer gleich schnell erfolgt. Änderungsdaten beziehungsweise neue Daten in dieser Datenbank sind für den Nutzer auf eine zuvor vom Cloud-Administrator definierte Größe limitiert („writeable space") *[7]*. Die dadurch erzielten Storage-Einsparungen sind für den Administrator jederzeit in Oracle Enterprise Manager ersichtlich, wie *Abbildung 1* zeigt.

Zum anderen kann ein Cloud-Nutzer im Self-Service-Portal Backups als logische Kopien seiner von ihm provisionierten Datenbank anlegen, etwa um nach einem Testvorgang später wieder auf einen definierten Zustand aufsetzen zu können. Normalerweise werden diese Backups mithilfe klassischer Methoden durch archivierte Redologs und ein RMAN „point in time restore" zum späteren Zeitpunkt durchgeführt. Alternativ kann der Nutzer aber, sofern es der Administrator zuvor eingestellt hat, auch hier die „Snap Clone"-Funktionalität nutzen. Ein Zurücksetzen der Datenbank erfolgt dann nicht mittels RMAN, sondern durch Wiederherstellen eines früheren Snapshot, was bei größeren Datenbanken natürlich wesentlich schneller geht. Es sei an dieser Stelle allerdings die Anmerkung angeführt, dass ein Snapshot kein richtiges Backup,

sondern lediglich ein logisches Backup ist. Eine korrupte Datenbank kann mit einem solchen Backup nicht wiederhergestellt werden, da korrupte Blöcke dann auch im Snapshot korrupt sind.

„Snapshot Copies" mit PDB-Cloning bei 12*c*-Multitenant

Ein wesentlicher Bestandteil der mit der Datenbank 12*c* neu eingeführten Multitenant-Architektur ist die Tatsache, dass sich eine Pluggable-Datenbank (PDB) nicht nur leer, sondern auch als Klon einer bestehenden PDB erzeugen lässt. Diese Cloning-Möglichkeit ist bereits direkt in den entsprechenden SQL-Befehl eingebaut, sodass im einfachsten Fall ein „create pluggable database <name-des-klons> from <original-pdb>" bereits zum Anlegen einer geklonten PDB ausreicht. Hierbei wird zunächst jedoch nicht auf Snapshot-Mechanismen zurückgegriffen, sondern ein normaler Kopiervorgang der zugrunde liegenden Dateien durchgeführt.

Durch ein einfaches Anhängen der Klausel „Snapshot Copy" lässt sich dieses Verhalten jedoch ändern, um völlig transparent auch von einer vorhandenen „Storage-Intelligenz" zu profitieren. Ein kompletter Aufruf könnte also wie in *Listing 1* aussehen.

Konkret unterstützte Storage-Systeme, die diese „Snapshot Copy" anbieten sind:

1. Oracle ZS3/ZFSSA
2. NetApp Filer
3. Oracle ACFS (ASM Cluster File System, auch als Cloud-Filesystem bekannt)

Voraussetzung für die Nutzung der ersten beiden Storage-Systeme ist das einmalige Hinterlegen der jeweiligen Storage-Credentials im Wallet der Container-Datenbank, da sonst der DBA ohne Zutun eines Storage-Administrators die Snapshot-Copys nicht anlegen könnte. Wie auch bei normalem PDB-Cloning muss die Quell-PDB in den „Read only"-Modus versetzt sein, bevor man den Klon tatsächlich anlegen kann. Aus naheliegenden Gründen kann eine PDB außerdem nicht mehr gelöscht werden, solange Klone von ihr existieren. Weitere Informationen dazu stehen zum Beispiel im Abschnitt „Thin Provisioning and Instantaneous Cloning: The Oracle ZFS Storage Appliance Advantage" auf OTN *[8]*.

Fazit

Wie dieser Artikel zeigt, haben alle hier vorgestellten Verfahren zum Datenbank-Cloning ihre entsprechenden Eigenarten und Voraussetzungen. Einige erfordern zum Beispiel den Einsatz bestimmter Datenbank-Versionen oder -Optionen, andere funktionieren vielleicht gerade mit dem eigenen Lieblings-Storage nicht. Welche der vorgestellten Varianten am besten passt, sollte daher je nach vorhandenem Know-how sowie der jeweils vorhandenen IT-Infrastruktur entschieden werden. *Tabelle 1* liefert dabei eine kleine Entscheidungshilfe.

Weiterführende Links

[1] http://www.oracle.com/technetwork/articles/servers-storage-admin/multitenant-on-sparc-solaris-2016889.html

[2] Sebastian Solbach, „Direct NFS Clone – Klonen der Oracle Datenbank zum Testen": https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_TIPP?P_ID=1121

[3] Robert Marz, „Da fliegt die Kuh – Rasante Datenbank-Klone durch cow (copy on write)", abrufbar für Teilnehmer der DOAG 2013 Konferenz im Mitgliederbereich der DOAG-Website

[4] http://www.oracle.com/us/products/servers-storage/storage/nas/snap/overview/index.html

[5] http://www.oracle.com/us/products/servers-storage/storage/nas/smu-bus-wp-final-1857065.pdf

[6] Oracle Enterprise Manager Cloud Administration Guide 12c Release 3, „Registering and Managing Storage Servers": http://docs.oracle.com/cd/E24628_01/doc.121/e28814/cloud_db_setup.htm#BABGDDDB

[7] Oracle Enterprise Manager Cloud Administration Guide 12c Release 3, „DBaaS Using Snap Clone Based Database Provisioning Profile": http://docs.oracle.com/cd/E24628_01/doc.121/e28814/cloud_db_portal.htm#CHDBCIIG

[8] „Oracle Multitenant on SPARC Servers and Oracle Solaris": http://www.oracle.com/technetwork/articles/servers-storage-admin/multitenant-on-sparc-solaris-2016889.html

Manuel Hoßfeld
manuel.hossfeld@oracle.com

Datenbank-Cloning mit Oracle ZFS Storage Appliances

Franz Haberhauer, ORACLE Deutschland B.V. & Co. KG

Mit ZFS verfügt Oracle über eine erprobte Speichertechnologie mit einem höchst effizienten „Copy on Write"-Mechanismus, der ein schnelles und platzeffizientes Cloning von Datenbanken erlaubt.

ZFS kann nicht nur unter Oracle Solaris direkt für Datenbanken auf ZFS-Dateisystemen genutzt werden, es dient in den ZFS Storage Appliances auch als internes Dateisystem. Die neuen Cloning-Features in der Oracle-Software – „DBaaS Snap Clone" in Enterprise Manager 12*c* und das Cloning von Pluggable Databases in der Oracle Datenbank 12*c* – unterstützen direkt ein „Thin Cloning" für Datenbanken, die auf ZFS Storage Appliances liegen. Darüber hinaus bietet Oracle mit dem Snap Management Utility eine Lösung für das Sichern, Wiederherstellen, Provisionieren und insbesondere das Cloning von Oracle-10*g*- und -11*g*-Datenbanken auf Basis der Snapshot-/Clone-Funktionalität der ZFS Storage Appliances. Diese können im Storage Area Network (SAN) genutzt werden, im Vordergrund steht aber die Nutzung als Network Attached Storage (NAS) über (d)NFS (Files) oder iSCSI (Block-Storage unter ASM).

Copy on Write mit Solaris ZFS

ZFS ist ein modernes lokales Dateisystem, das sich seit fast zehn Jahren in Solaris bewährt hat und in Solaris 11 als Root-Filesystem vorgegeben ist – nicht zuletzt wegen seiner Snapshot- und Cloning-Funktionalitäten. Diese motivieren, darauf auch Datenbanken zu betreiben. Best Practices für Datenbanken auf ZFS hat der Autor in der DOAG News vom Juli 2013 vorgestellt *[1]*.

ZFS überschreibt Daten grundsätzlich nicht unmittelbar, sondern folgt dem Prinzip „Copy on Write" (CoW). Dabei werden geänderte Blöcke wie neue Blöcke an eine neue Stelle auf der Platte geschrieben, was wahlfreie Schreibzugriffe in effiziente, sequenzielle Schreiboperationen umwandelt und damit den Zugriffs-Charakteristika moderner Platten entgegenkommt. Dateien werden in Baumstrukturen verwaltet, wobei sich Änderungen erst mit der Änderung des Wurzelknotens („Überblock") materialisieren. Wird dabei der vorige Wurzelknoten nicht

freigegeben, kann er als Wurzel eines Snapshot dienen, der eine Sicht auf die bisherige Baumstruktur – und damit den bisherigen Datei-Inhalt – liefert.

In ZFS sind Snapshots „read only". Sobald sie schreibbar sind, spricht man von Clones. Der große Vorteil dieser Technologie besteht darin, dass für Clones gegenüber dem Ausgangszustand lediglich das tatsächlich geänderte Volumen als zusätzliche Speicherkapazität benötigt wird („Thin Cloning"). Zudem sind beim Anlegen nur minimal Zeit und Ressourcen erforderlich und die Zahl der Snapshots und Clones ist nicht begrenzt.

Snapshots können an jeder Stelle in der ZFS-Datei-Hierarchie angelegt werden, beispielsweise für das Verzeichnis „/mypool/mydataset/mydir" im Pool „mypool" mit dem Kommando „# zfs snapshot mypool/mydataset/mydir@201404081400". Dabei identifiziert der String nach dem „@" den Snapshot. Über „# zfs clone /mypool/mydataset/mydir@201404081400 mypool/mydataset/cloneddir" wird daraus ein Clone unter dem Pfad „/mypool/mydataset/cloneddir". Sind weitere ZFS-Datasets im Pfad eingebunden, lassen sich über die Option „-r" synchronisiert rekursive Snapshots darüber hinweg erstellen.

ZFS Storage Appliance

Mit Solaris verfügt Oracle über ein Betriebssystem, das alle Anforderungen an ein Speichersystem abdeckt: ZFS als leistungsstarkes lokales Dateisystem; COMSTAR, eine SCSI-Target-Funktionalität für die Bereitstellung von Block-Storage über Fibre Channel (FC) im SAN oder iSCSI im IP-Netzwerk und über iSER und SRP insbesondere auch via InfiniBand, sowie NFS als Netzwerk-Dateisystem, das ja einst bei Sun entwickelt worden war. Auch CIFS ist performant direkt im Solaris-Kern implementiert.

Während früher für NAS-Filer im Hinblick auf die verfügbare Hardware dedizierte Betriebssysteme entwickelt wurden, ist heute aufgrund der Leistungsexplosion bei der Hardware der Einsatz von General-Purpose-Betriebssystemen naheliegend, um von den Entwicklungen für andere Einsatzbereiche zu profitieren – insbesondere was Hardware-Unterstützung, Treiber, Skalierbarkeit, Sicherheit etc. angeht – und Entwicklungsbudgets besser in höherwertige Funktionen investieren zu können. Die ZFS Storage Appliances *[2]* bringen eine Vielzahl an Funktionalitäten mit wie Kompression oder Deduplizierung, die nicht separat lizenziert werden müssen. Außerdem ist auf diesen Speichersystemen für Oracle-Datenbanken „Hybrid Columnar Compression" einsetzbar.

Für die Administration gibt es zum einen eine webbasierte, intuitive grafische Oberfläche. Eine Besonderheit ist dabei Analytics, eine Komponente, die eine sehr fein granulierte Analyse von IO-Mustern ermöglicht, bis hin zur Visualisierung des Spektrums der Latenz einzelner Typen von E/A-Operationen oder der Offsets in Dateien *[3]*. Hierdurch wird die Diagnose von I/O-Problemen signifikant vereinfacht – und in einigen Fällen überhaupt erst möglich. Zum anderen ist eine Integration sowohl mit dem Enterprise Manager Ops Center *[4]* wie auch – zumindest was das Monitoring angeht – über ein Storage-Plug-in direkt mit dem Enterprise Manager möglich *[5]*. Speziell für das Cloning von Datenbanken gibt es eine weitergehende Werkzeugunterstützung, die nachfolgend vorgestellt wird.

Zunächst noch kurz zur Hardware der ZFS Storage Appliances *[6]*: Die aktuellen Systeme der ZS3-Familie (ZS3-2, siehe *Abbildung 1*, beziehungsweise ZS3-4) verfügen jeweils über einen einzelnen Controller oder über einen HA-Cluster aus zweien (aktiv-aktiv). Auf diesen Controllern oder Köpfen läuft die Betriebssoftware. Zur Speicherung kommen Disk Shelves mit bis zu 24 Platten zum Einsatz. In Engineered Systems werden ZFS Storage Appliances intern in der Exalogic, Exalytics oder dem SPARC SuperCluster verbaut. Sie sind aber auch als Backup-Plattform insbesondere für die Exadata populär – nicht zuletzt, da sie eine direkte InfiniBand-Anbindung erlauben *[7]*. Hierbei wurden bis zu 26 TB/Stunde beim Backup und 17 TB/Stunde beim Restore gemessen. Seit Anfang März 2014 ist für diesen Einsatzbereich vorkonfiguriert die ZS3 Backup Appliance ZS3-BA verfügbar *[8]* und seit April 2014 die Version 2.0 des Oracle Engineered Systems Backup Utility for Oracle ZFS Storage Appliance, über das in wenigen Schritten (etwa generierte RMAN-Skripte) das Backup für Exadata, ODA und SuperCluster auf ZFS Storage Appliances aufgesetzt werden kann *[9]*.

Sind Oracle-12c-Datenbanken über dNFS auf den ZFS Storage Appliances abgelegt, kommt das Oracle Intelligent Storage

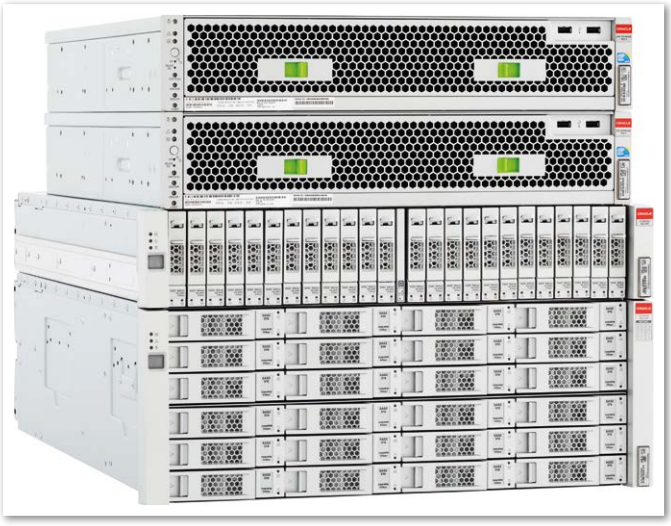


Abbildung 1: ZFS Storage Appliance ZS3-2 mit zwei Clustered Controller Heads, einem DE-24P Disk Shelf mit 24 2.5"-10.000rpm-900GB- und einem DE-24C Disk Shelf mit 24 3.5"-7200rpm-4TB-Laufwerken

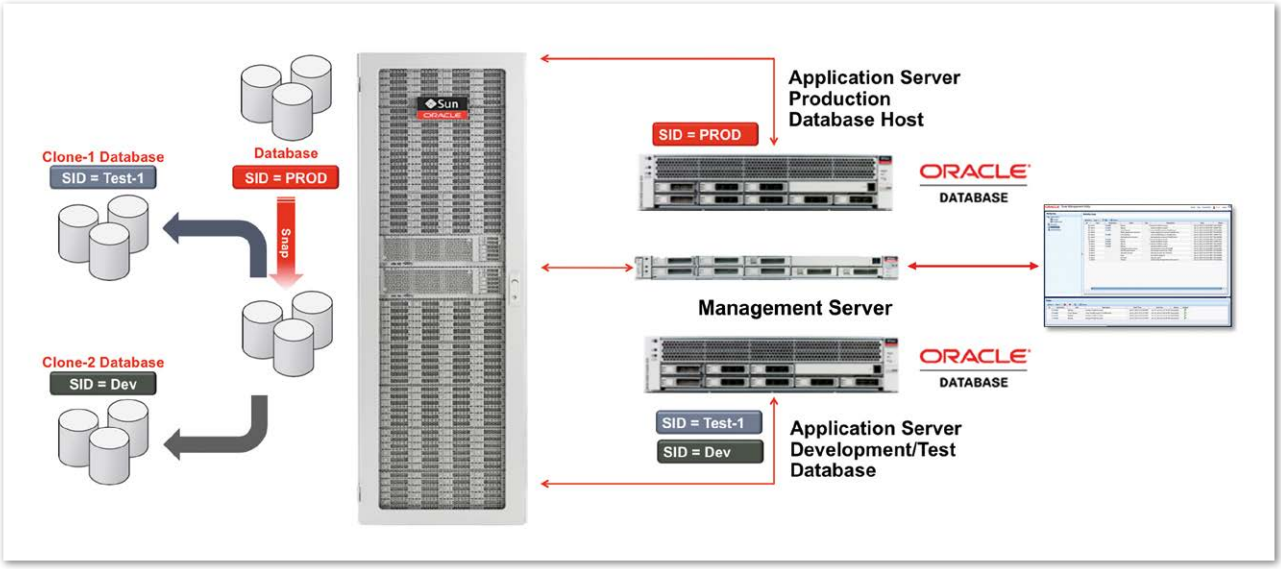


Abbildung 2: Snap Management Utility for Oracle Database: Datenbank-Clones und Management-Server können auf verschiedene Server verteilt sein

Protocol (OISP) zum Tragen, über das die Datenbank exklusiv den ZFS-SA Meta-Informationen über die Last bereitstellt, die diese für automatisches Tuning nutzen *[10]*.

Datenbank-Cloning mit der ZFS Storage Appliance

Wie gesagt, unterstützen die neuen Cloning-Features in der Oracle-Software – „DBaaS Snap Clone" im Enterprise Manager 12*c* oder das Cloning von Pluggable Databases in der Oracle Datenbank 12*c* – direkt ein „Thin Cloning" für Datenbanken, die auf ZFS Storage Appliances liegen. Um eine ZFS Storage Appliance im Enterprise Manager als Storage-Server zu nutzen, werden administrative Nutzer beziehungsweise Rollen mit den entsprechenden Privilegien angelegt und die Appliance im EM registriert *[11]*. Danach erfolgt das Management aus dem Enterprise Manager, der mit der Appliance über „ssh"-Verbindungen kommuniziert. Auch für das Cloning von Pluggable Databases werden Username und Passwort für einen entsprechend autorisierten Nutzer (hier in einem Oracle Database TDE Keystore) hinterlegt *[12]* und das Cloning auf der Appliance über eine SQL-Klausel getriggert.

Während die meisten Software-Features in den ZFS Storage Appliances einschließlich Snapshots frei nutzbar sind, erfordern das Cloning von Daten auf einem System sowie die Replikation zwischen Systemen zusätzliche Lizenzen. Beim Enterprise Manager Snap Clone, ebenfalls ein Feature des Oracle Cloud Management Pack for Oracle Database, ist eine „Restricted Use License" für das ZFS-Storage-Appliance-Cloning enthalten. Auch beim nachfolgend vorgestellten Snap Management Utility ist das Cloning von Oracle-Datenbanken über das Tool bereits mit lizenziert.

Snap Management Utility for Database

Während die oben skizzierten neuen Funktionalitäten Enterprise Manager 12*c* beziehungsweise die Datenbank 12*c* voraussetzen, bietet das Oracle Database Snap Management Utility (SMU, siehe *Abbildung 2*) eine Lösung für das Sichern, Wiederherstellen, Provisionieren und insbesondere das Cloning von 10*g*- und 11*g*-Datenbanken (auch RAC) auf der Basis der Snapshot-/Clone-Funktionalität der ZFS-Storage-Appliances-Utility (SMU) *[13,14]*. Gegenüber manuellen Vorgehensweisen *[15]* bietet das SMU den Vorteil einer unterstützten Software, die gepflegt und weiterentwickelt wird und nicht zuletzt aufgrund der enthaltenen Cloning-Lizenz auch preislich attraktiv ist. Zudem verfügt sie nicht nur über ein Command-Line-Interface, sondern auch über eine Browser-basierte Nutzerschnittstelle. SMU kann für Datenbanken unter Oracle Solaris, Linux oder Windows genutzt werden, wobei Direct NFS und ASM auf iSCSI 11*g* erfordern.

Das Snap Management Utility ermöglicht Cloning einerseits auf Basis eines Snap-Backups einer Datenbank und andererseits auf Basis eines RMAN-Backups *(siehe Abbildung 3)*. Beim ersterem wird auf der ZFS StorageAppliance ein Snapshot einer Datenbank angelegt – als Offline- (cold) oder als Online-Backup (hot, nur bei Datenbanken auf (d)NFS). Daraus wird ein Clone erstellt als Datenbank auf demselben Host wie die Ausgangsdatenbank oder auf einem anderen Host, der dieselbe Architektur und Datenbank-Software wie der Host der Ausgangsdatenbank haben muss. Eine Konversion zwischen einer Single-Instance- und einer RAC-Umgebung ist möglich.

Für das Cloning auf Basis eines RMAN-Backups muss dieses im „Image Copy"-Format vorliegen. Quell- und Zielhost des Datenbank-Cloning können bei dieser Variante nicht identisch sein, müssen aber denselben Softwarestand haben. Dabei wird vom RMAN-Backup ein ZFS-Clone erzeugt, der dann auf dem Zielhost eingebunden wird. SMU startet auf dem Zielhost eine temporäre Datenbank-Instanz, über die verschiedene Informationen aus dem Backup-Controlfile ermittelt werden – insbesondere der Recovery-Point (maximale SCN) und die Größe der Flash Recovery Area. Die temporäre Instanz wird dann wieder heruntergefahren, ein Parameterfile für die neue Clone-Datenbank erzeugt, diese damit gestartet und ein Recovery durchgeführt.

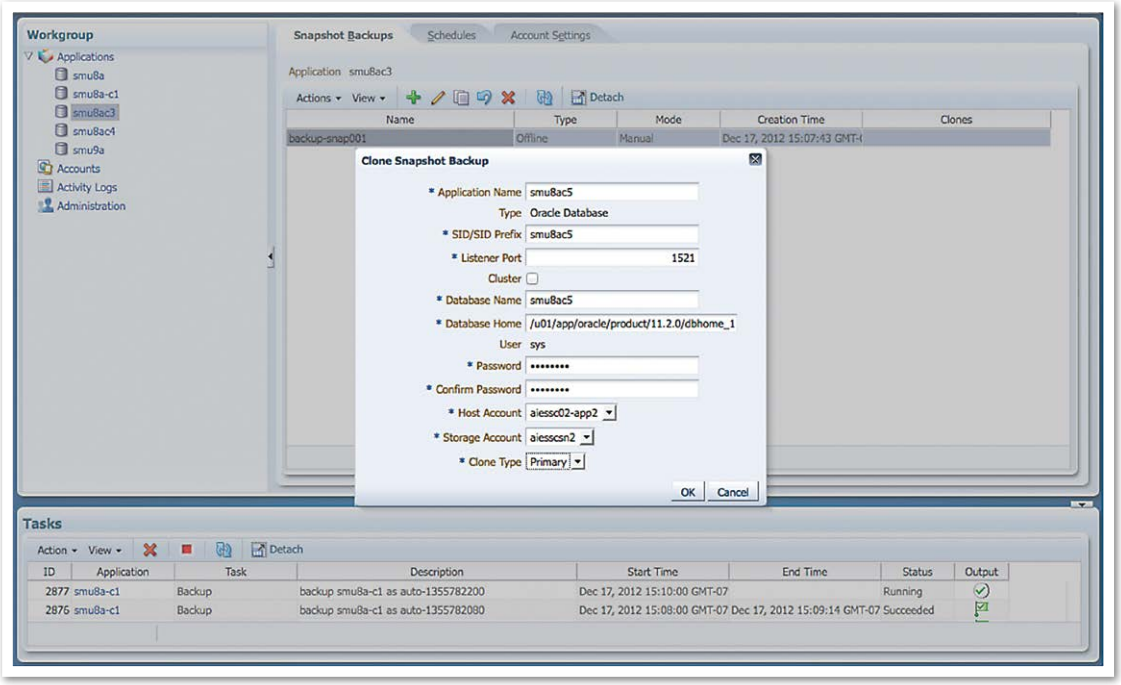


Abbildung 3: Snap Management Utility: Clone-Snapshot-Backup-Dialog

Fazit

Mit den ZFS Storage Appliances hat Oracle Speichersysteme im Produktportfolio, die aufgrund der zugrunde liegenden „Copy on Write"-Technologie hervorragend für Thin Cloning geeignet sind. Das Cloning von Datenbanken erfordert ein Zusammenspiel von Datenbank- und Storage-Administration. Mit den neuen Cloning-Funktionalitäten in der Datenbank 12*c* sowie im Enterprise Manager 12*c* lassen sich die Cloning-Features nun transparent direkt aus der Datenbank beziehungsweise dem Enterprise Manager heraus nutzen.

Mit dem Snap Management Utility steht zudem ein Werkzeug zur Verfügung, das Thin Cloning auch für ältere Datenbank-Versionen einfach nutzbar macht –demonstriert in einem YouTube, wobei zunächst Analytics für Datenbanken gezeigt wird *[16]*. ZFS Storage Appliances haben sich übrigens auch in der Oracle-IT und den Cloud-Datacentern durchgesetzt. Hier ist inzwischen eine Gesamtkapazität von weit über 200 Petabyte installiert *[17]*.

Literaturhinweise

[1] Franz Haberhauer, Best Practices für Datenbanken auf ZFS: DOAG News, Ausgabe 03/2013, S. 18-21

[2] Architectural Overview of the Oracle ZFS Storage Appliance, An Oracle White Paper, Januar 2014: http://www.oracle.com/technetwork/server-storage/sun-unified-storage/documentation/o14-001-architecture-overview-zfsa-2099942.pdf

[3] Oracle ZFS Storage Appliance Analytics Guide: http://docs.oracle.com/cd/E27998_01/html/E48490/index.html

[4] How to Prepare a Sun ZFS Storage Appliance to Serve as a Storage Device with Oracle Enterprise Manager Ops Center 12c: http://www.oracle.com/technetwork/articles/servers-storage-admin/howto-zfs-appl-ops12c-1840019.html

[5] Oracle Enterprise Manager System Monitoring Plug-in for Oracle ZFS Storage Appliance: http://www.oracle.com/technetwork/oem/grid-control/downloads/zfs-storage-plugin-487867.html

[6] ZFS Storage Appliance Site u.a. mit 3D-Modellen der Hardware: http://www.oracle.com/technetwork/server-storage/sun-unified-storage/overview

[7] Oracle ZFS Storage Appliance: http://www.oracle.com/us/products/servers-storage/storage/nas/overview und http://www.oracle.com/us/products/servers-storage/storage/nas/zs3-ds-2008069.pdf

[8] Backup and Recovery for Oracle Engineered Systems with Oracle ZFS Storage ZS3-BA, Oracle Data Sheet, March 2014: http://www.oracle.com/us/products/storage/engineered-systems-backup/ds-zs3-ba-140228-2162876.pdf; Oracle ZFS Storage ZS3-BA: http://www.oracle.com/us/products/servers-storage/storage/nas/zs3-ba

[9] Oracle Engineered Systems Backup Utility for Oracle ZFS Storage Appliance: http://www.oracle.com/technetwork/server-storage/sun-unified-storage/downloads/zfssa-plugins-1489830.html

[10] Oracle Intelligent Storage Protocol, Data Sheet: http://www.oracle.com/us/products/servers-storage/storage/nas/oracle-isp-ds-final-2139042.pdf

[11] Registering and Managing Storage Servers: http://docs.oracle.com/cd/E24628_01/doc.121/e28814/cloud_db_setup.htm#autoId7

[12] Ritu Kamboj, Oracle Multitenant on SPARC Servers and Oracle Solaris, September 2013: http://www.oracle.com/technetwork/articles/servers-storage-admin/multitenant-on-sparc-solaris-2016889.html

[13] Snap Management Utility for Oracle Database: http://www.oracle.com/us/products/servers-storage/storage/nas/snap

[14] Oracle Snap Management Utility for Oracle Database Documentation: http://docs.oracle.com/cd/E39520_01

[15] Database Cloning using Oracle Sun ZFS Storage Appliance and Oracle Data Guard, Oracle White Paper, December 2011: http://www.oracle.com/technetwork/database/features/availability/maa-db-clone-szfssa-172997.pdf

[16] Art Licht, Simplifying Database-to-Storage Management, YouTube, March 2013: https://www.youtube.com/watch?v=w4yYhPiHP88

[17] Sun ZFS Storage Appliance and Oracle IT, Use Cases and Benefits, Oracle White Paper, September 2012: http://www.oracle.com/us/products/servers-storage/storage/nas/resources/zfs-saoracle-it-whitepaper-100812gc-1875031.pdf

Franz Haberhauer
franz.haberhauer@oracle.com

Development und Operation macht DevOps

Mylène Diacquenod, DOAG Online

Der Begriff DevOps zirkuliert bereits seit 2009 auf Veranstaltungen, in den Blogs und Fachzeitschriften dieser Welt und doch herrscht immer noch keine komplette Einigkeit über seine Bedeutung. Das Silbenwort, das sich aus Development und Operations zusammensetzt, steht oft genug unter dem Verdacht, ein weiteres Buzz-Wort unter vielen zu sein. Im Gegensatz zur agilen Softwareentwicklung gibt es kein offizielles schriftliches Manifest. Manch einer vermutet darunter ein Tool oder einen Standard – nichts dergleichen. DevOps ist eine Kultur, eine Ansammlung von Ideen, die an sich kein Novum sind. Aber ein paar Gedanken an DevOps zu verlieren, kann angesichts mindestens zweier Grundprinzipien lohnend sein: Agilität im Betrieb und Zusammenarbeit zwischen Entwicklung und Betrieb.

Geburtsstunde von DevOps

Seine offizielle Geburtsstunde hat DevOps im Jahre 2009 im belgischen Gent erlebt. Auf Anregung von Patrick Dubois fand die erste gleichnamige Zweitageskonferenz statt: die DevOpsDays. Die Ideen der Bewegung waren zu diesem Zeitpunkt nicht neu. DevOps fungierte sozusagen nur als Etikett, als Katalysator von Strömungen und schlug ein wie eine Bombe. Seitdem gibt es DevOpsDays fast rund um den Globus.

Agile Infrastruktur

Wahrscheinlich haben die großen Internetdienste die DevOps-Bewegung hervorgerufen, als Administratoren anfingen, die Prinzipien der agilen Softwareentwicklung auf den Betrieb zu übertragen. Es entstanden Begriffe wie „Agile Systemadministration" oder „Agile Operation" – eine grundlegende Änderung der Arbeitsweise im Betrieb.

Dev und Ops arbeiten zusammen

Dass es zwischen Development und Betrieb zu Kollisionen kommt, ist wohlbekannt. Viel zu oft verbringen die Abteilungen dann mehr Zeit mit Schuldzuweisungen als mit Problemlösungen.

Noch vor den ersten DevOpsDays teilten John Allspaw und Paul Hammond ihre Erfahrungen beim Fotodienst Flickr in einem Vortrag namens 10+ Deploys per Day – Dev and Ops Cooperation at Flickr und riefen Begeisterungen hervor. Sie zeigten die Vorteile einer Zusammenarbeit zwischen Entwicklung und Betrieb in allen Phasen des Zyklus.

Dev und Ops sind als generische Begriffe zu verstehen. Unter Ops verstecken sich nicht nur der Systemadministrator, sondern natürlich auch der Datenbankadministrator sowie weitere Teildisziplinen, die zum stabilen Betrieb einer Software beitragen. Ähnlich verhält sich Dev: Vom Entwickler bis hin zum Produktmanagement über die Qualitätssicherung sind alle Mitarbeiter gemeint, die in irgendeiner Weise an der Erstellung einer Software beteiligt sind.

Definition

DevOps wirft einen ganzheitlichen Blick auf den Softwarebereitstellungsprozess entlang des gesamten Zyklus. Es ist eine Verlängerung der agilen Prinzipien über die Grenzen des Codes hinweg auf die gesamten IT-Services – von der Produktion bis zum Betrieb. DevOps richtet Ziele, Prozesse und Werkzeuge von Entwicklung und Betrieb zueinander aus und verbessert damit die Zusammenarbeit.

Die vier Säulen von DevOps

Kultur

Nichts funktioniert ohne Kultur. DevOps ist vor allem das – eine Kultur, eine Intension, ein gemeinsamer Nenner, der Entwicklung und Betrieb verbindet und Informationssilos vermeidet. Gemeinsame Werte sind die Grundlage einer guten Zusammenarbeit, die auf Verständnis, Vertrauen und Respekt basiert. Denn mit den wohlbekannten Schuldzuweisungen „es ist dein Code" oder „es ist deine Maschine" ist man bislang nicht unbedingt weitergekommen. Teil dieser Kultur ist es auch, Fehler zu erlauben. Denn daraus kann man lernen, wenn das Finger-Pointing nicht ablenkt.

Automatisierung

Bei immer wiederkehrenden Tätigkeiten ist eine Automatisierung sinnvoll. So werden menschliche Fehler vermieden. Durch die gewonnene Transparenz gewinnt auch die Qualitätskontrolle an Qualität und die Mitarbeiter können sich auf wichtigere Aufgaben konzentrieren. DevOps-Anhänger bezeichnen die Automatisierung als Infrastructure as a Code.

Ein netter Nebeneffekt: Automatisierung sorgt auch für ein gegenseitiges Vertrauen im Projekt. Die Automatisierung zwingt das Projektteam gewissermaßen vorauszuschauen und den Betrieb frühzeitig einbinden. Für Entwicklung und Betrieb herrschen demnach die gleichen Deployment- und Konfigurationsregeln.

Messung

Werte entscheiden über die Qualität eines Produkts, nicht das Bauchgefühl. Das Treffen von Entscheidungen anhand von Messwerten gehört zu einem wichtigen Aspekt von DevOps.

DevOps braucht allerdings neue Messkriterien, die für die gemeinsame Leistung von Betrieb und Entwicklung gelten.

Teilen

Menschen mit verschiedenen Hintergründen besitzen oftmals auch unterschiedliches Wissen und andere Fähigkeiten. Indem sie sich austauschen, lernen sie voneinander und lösen Probleme viel erfolgreicher. Die Voraussetzung dafür: die Bereitschaft zu teilen.

Myléne Diacquenod
mdi@doag.org

DevOps mit Cloud Control

Andreas Chatziantoniou, Foxglove-IT BV; Tobias Weih, CABP; Dirk Ruchatz, ITC-DR

Dieser Artikel zeigt, was DevOps für das Oracle-Umfeld beinhaltet und wie die Aktivitäten im Cloud Control abgebildet werden können.

Es ist geschäftskritisch, die Durchlaufzeit von Änderungen an der Software zu reduzieren, um eine neue Version so schnell wie möglich in ein funktionierendes System zu verwandeln. Während manche Unternehmen sich heute noch Zykluszeiten von mehreren Monaten erlauben, bringen andere schon mehrfach täglich eine neue Version in Produktion.

Zur Verkürzung der Zykluszeiten und zur Reduktion des Release-Risikos ist eine konsequente Vollautomatisierung der „Delivery und Quality Assurance" über alle Bereiche hinweg essenziell – und das sowohl für „Dev" als auch für „Ops". Wie lange zieht sich ein Change in der Organisation hin, wie lange dauert es, eine Änderung an einer einzigen Zeile Code in Produktion zu bringen? Was kostet das? Und wie komplex wird diese Aktivität, wenn die Änderung nicht nur Code, sondern auch Fusion-Middleware-Komponenten betrifft?

Continous Delivery

Jeder kennt und viele fürchten diese Wochenenden, an denen ein System-Upgrade ausgerollt werden soll. Was nutzt ein integrierter Oracle-Middleware-Stack, wenn die Entwicklungs-, Deploy- und Release-Prozesse nicht integriert sind? Wie stopft man das lang vergessene Loch zwischen Development und Operations? Wie bekommt man diese letzte Meile in den Griff? Eine funktionierende Continuous Integration (CI) ist dabei nur ein erster Schritt. Neben Code sind Konfiguration, Middleware-Stack, Daten und Systemkontext fragil, komplex und kritisch.

Continous Delivery beschreibt den Prozess für DevOps. Ziel ist die Integration der verschiedenen Aufgaben und Funktionen in einer Deployment-Pipeline, die Change-, Build-, Requirements-, Integrations-, Release- und Test-Management unter agiler Maßgabe implementiert. Dazu gehören:

- Continuous Integration und Deployment
- Data Management
- Configuration Management
- Environment Management
- Automated Testing
- Release Management

Cloud Control bietet hier verschiedene Formen der Unterstützung – teilweise durch kostenpflichtige Module.

Agile Delivery

Der zunehmende Einsatz komplexer Middleware verschiebt maßgebliche Aufwände aus der Software-Entwicklung („Dev") in den Betrieb („Ops"). Auch die Middleware und damit Middleware-Ops ist zunehmend im Sog dynamischer und sich entwickelnder Anforderungen. Dahinter stehen wandelbare, sich im Zeitablauf entwickelnde Anforderungen an Middleware und deren effiziente und qualitativ hochwertige Umsetzung.

System Engineering ist im Betrieb hingegen noch immer weitgehend einem statischen Idealbild verhaftet. Die notwendigen Schritte hin zu einer anforderungsgetriebenen, zunehmend mächtigen und komplexen Middleware stehen aus. Jede Änderung ist ein maßgebliches Risiko, das die Stabilität gefährdet. Dazu kommt eine Fragmentierung der eingesetzten Tools, was zwangsläufig zu einer niedrigen Nachvollziehbarkeit von Änderungen führt.

Cloud Control als integraler Bestandteil

Um ein Oracle-Fusion-Middleware-System im Ops-Bereich überhaupt kontrollieren und betreiben zu können, kommt Cloud Control zum Einsatz. Das Betreiben einer solchen Umgebung mit Selbstbau-Skripten und Command-Line-Akrobatik wird im Enterprise-Umfeld schnell zu einer unhaltbaren Situation für den Betrieb führen. Da Cloud Control schon vorhanden ist, sollten in Hinsicht auf DevOps-Aktivitäten intensiv die Möglichkeiten des Cloud Control genutzt werden. Da der klassische Betrieb schon mit Cloud Control vertraut ist, dürfte die Ausbreitung der Monitoring-Kapazitäten des Cloud Control mit den Provisioning-Möglichkeiten ein logischer Schritt sein.

Die Lösungsansätze für die Bereitstellung der Software benötigen natürlich auch eine Plattform, auf der die Software eingesetzt werden kann. Hierbei stellen sich oft die folgenden Probleme: Bereitstellung von Hardware, Betriebssystem, Storage, Netzwerk-Komponenten etc. erfordert Expertenwissen. Dies in Kombination mit Oracle-Software zu einem lauffähigen System zusammenzubringen, ist im Enterprise-Umfeld oft ein langer und mühsamer Weg. In einfachen Umgebungen wird dann regelmäßig eine Kopie eines existierenden Systems benutzt. Dies ist allerdings in komplexen Fusion-Middleware- und Fusion-Apps-Umgebungen nicht mehr machbar, da es Hunderte von Punkten gibt, bei denen Feinheiten aufeinander abgestimmt werden müssen.

Der Lösungsweg von Venisolvere geht dahin, (abhängig vom Einsatzgebiet) mit vorkonfigurierten VM- beziehungsweise Scripted Oracle/OS-Installationen Standardbausteine anzubieten. Bei häufig gefragten Komponenten (WLS, SOA Suite) ist ein VM-Template der bessere Weg, bei anderen Komponenten bieten die „Ready To Build"-Skripte einen Ausweg. Diese

Bausteine werden dann in Cloud Control eingebunden. Wichtig ist hierbei ein sogenanntes „Late-Binding“ von Parametern, um die Bausteine genau so fertigzustellen („finishing touch“), wie sie in der zu bauenden Umgebung benutzt werden sollen. Die Kombination aus IaaS und PaaS bietet die Flexibilität, die zurzeit oft benötigt wird, aber selten geliefert werden kann.

Damit sind die Übergänge zwischen den Entwicklern und DevOps besser definiert. Der Fokus der Entwickler bleibt auf dem Code, während das DevOps-Team mit einem Katalog der Angebote eine passende und gewünschte Ziel-Landschaft zur Verfügung stellt. Gerade beim DevOps-Team geht dann der Weg vom reaktiven („muss so schnell wie möglich fertig sein“) zum automatisierten Bereitstellen der Umgebungen und zur Unterstützung (Lernen, Verbessern, Wiederholen) der Entwickler („Extra-Umgebung für Fehlersuche“).

Am sinnvollsten ist der Einsatz von Venisolvere bei Engineered Systems (Exa*), aber auch in konventionellen Systemen. Das liegt am Einsatz der Provisionierungsmöglichkeiten des Cloud Control. Da schon beim Start Cloud Control eingebunden ist, gibt es im Laufe des Projekts deutliche Vorgehensweisen, wenn es um die Unterstützung des Lebenszyklus geht. Ob es um die Bereitstellung von Umgebungen, das Management von Artefakten und Konfigurationsversion, die Performance-Analyse oder den normalen Betrieb geht, die Werkzeuge und Übersicht sind schon vorhanden.

Das Ziel durch diese Baustein-Sicht liegt darin, ein Self-Service der Plattform zu ermöglichen. Ein Projekt entscheidet dann selbst, wann es welche Ressourcen benötigt. Durch Charge-Back kann dann mit den Abnehmern entschieden werden, ob Ressourcen noch erforderlich sind.

Die Plattform – das Ziel?

Obwohl wir alle davon überzeugt sind, dass nur funktionsfähige Software im Produktiv-Einsatz dem Anbieter und dem Kunden einen echten Mehrwert bietet, müssen wir akzeptieren, dass dies nur dann stattfindet, wenn es eine Plattform gibt, auf der diese Software lauffähig ist. Wer sich heutzutage mit Umgebungen in Projekten auseinandersetzt, stellt fest: Die Tage der statischen Umgebung sind vor langer Zeit auf dem Friedhof der IT gelandet.

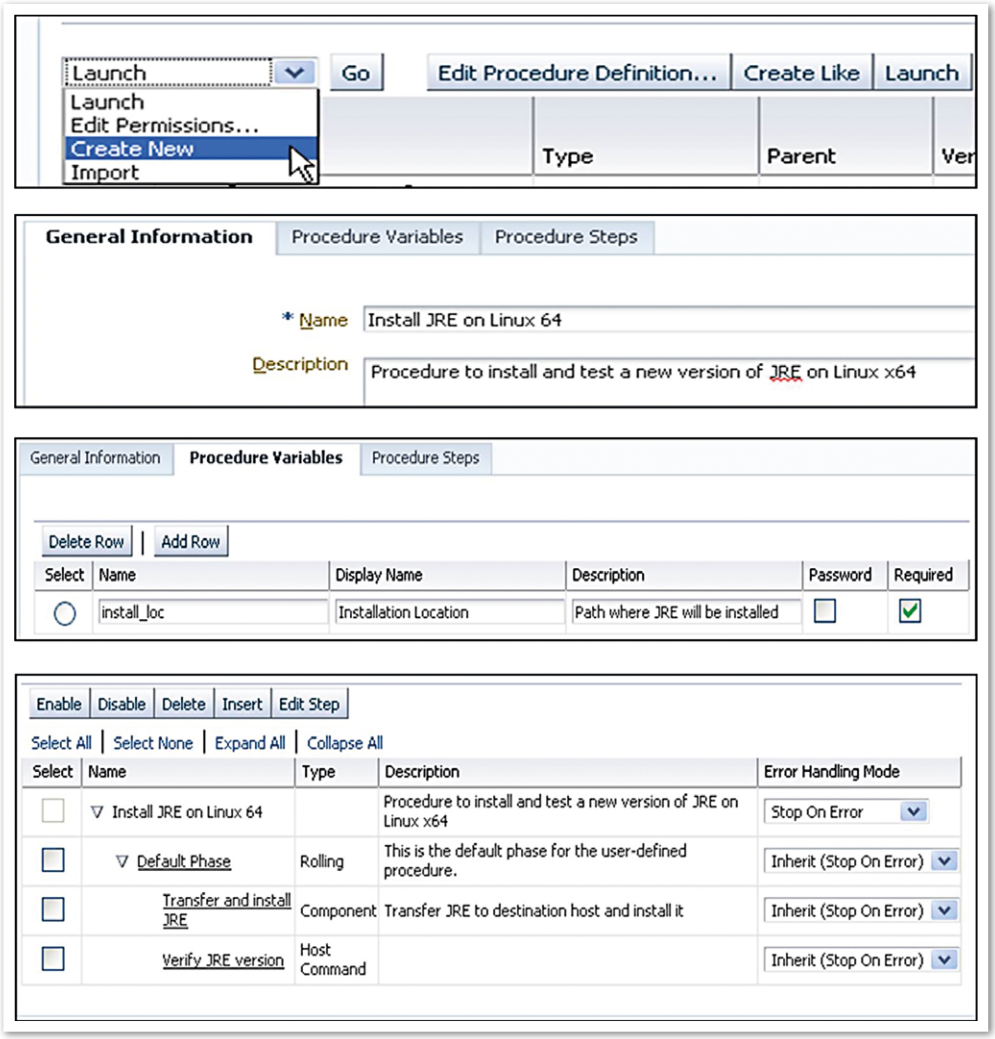


Abbildung 1: Die Schritte der Deployment Procedure

Das Merkmal einer IT-Umgebung ist heute die schnelle Anpassung an Veränderungen, egal ob dies ein Patch von Oracle ist, eine Änderung im Deployment oder eine Anpassung von Netzwerk-Komponenten. In der guten alten Zeit gab es noch die Prozesse, in denen sich alle Beteiligten an einen großen Tisch setzten und Änderungen besprachen, das nächste Wartungsfenster abwarteten und dann eine Konfiguration angepasst wurde (es scheint sogar noch heute Organisationen zu geben, die so arbeiten).

Abgesehen vom Zeitaufwand und der Belastung des Personals (Wochenendarbeit) ist bei der Komplexität der heutigen Umgebungen diese Arbeitsweise nicht mehr zu vertreten. Der Grund hierfür liegt in der Tatsache, dass ein Oracle-Fusion-Middleware-System schnell mehr als zwanzig verschiedene Komponenten beinhaltet und diese auch noch in den verschiedenen Lebenszyklen des Projekts beziehungsweise Produkts auftreten. Nehmen wir das Beispiel einer WebCenter Umgebung: Datenbanken, JRockit, WLS, WebCenter Content, WebCenter Portal, SES, OIM, OWSM, OSB etc.

Jede Komponente durchläuft den Zyklus „Installation, Konfiguration, Deployment von Software auf der Komponente, Anpassungen des Deployment“, um schließlich mit Monitoring und Performance-Tuning in der Produktion aktiv zu sein. Schon bei der Installation einer Komponente, egal ob Datenbank, WLS oder FMW, gibt es eine Vielzahl von Einstellungen, die sowohl die Funktionsweise selbst

als auch den Betrieb beeinflussen. Nun durchläuft jedes Projekt mit allen Komponenten einen Lebenszyklus. Wie garantiere ich als DevOps den identischen Aufbau meiner Umgebungen? Wie gehe ich mit dem Problem um, dass meine Umgebungen keine logische Äquivalenz – also eine unterschiedliche Topologie – haben?

Hinzu kommt, gerade bei großen Enterprise-Umgebungen, dass die Ziele „Hochverfügbarkeit", „Security" und „Performance" oft abhängig von richtig konfigurierten Komponenten sind. Im Sinne von DevOps kann diese Komplexität der Bereitstellung und der zu liefernden Qualität nur dann zufriedenstellend berücksichtigt werden, wenn eine uniforme, jedoch leicht anzupassende Methode eingesetzt wird. Cloud Control bietet dies durch verschiedene Werkzeuge. Im Zyklus „Planung, Entwicklung, Test (Continuous Integration)" kommen mit „Bereitstellung, Deployment und Operate" die Bereiche „Continuous Delivery" und „DevOps" hinzu. Durch den Einsatz von Cloud Control können gerade diese Aktivitäten in einer guten Art und Weise bedient werden.

Cloud Control und Continuous Delivery

Um mit der Tür ins Haus zu fallen: Cloud Control ist kein DevOps-Tool. Tools spielen bei DevOps eine wichtige Rolle, aber DevOps ist kein Tool. Andererseits wird gesagt, jedes Tool sei ein DevOps-Tool. Demzufolge ist Cloud Control sicherlich in der Lage, als Werkzeug in DevOps eine Rolle zu spielen.

Sehen wir uns den Vorgang im Detail an: Wie eine Umgebung aufgebaut wird, lässt sich anhand von Manuals und Best Practices beschreiben. Jedoch kommen hier schon bei wenig komplexen Umgebungen (wie RAC) verschiedene Tools zum Einsatz. So müssen sich Benutzer mit SSH auf anderen Hosts einloggen können, Netzwerke müssen definiert und konfiguriert sowie ASM, CRS und andere Systembestandteile aufgebaut werden, bevor man überhaupt seine Datenbank erstellt. Nun wird ein DevOps-Verantwortlicher diese Umgebung mehrmals beziehungsweise sehr oft aufbauen müssen. Also ist eine Abbildung in Tools offensichtlich; Skripte spielen hier eine wichtige Rolle. Cloud Control bietet jedoch mehr, da es auch im Bereich der Bereitstellung sogenannte „benutzerdefinierte Deployment Procedures" anbietet *(siehe Abbildung 1)*.

In einer solchen Deployment Procedure kann man:

- Eigene Prozeduren erstellen
- Inhalte der Software Library nutzen
- Parametrisierung durch globale Variablen durchführen
- Verschiedene Aktionen aufrufen:
  - Host-Kommandos
  - Skripts
  - Dateitransfer

Dies bietet die Chance, seine Umgebung in einer flexiblen Art aufzubauen mit dem Wissen, dass die Wiederholung der Ausführung dasselbe Resultat liefert, durch Parameter gesteuert und im Falle von Veränderungen angepasst werden kann. Venisolvere hat dies erkannt und bietet eine Reihe dieser Bausteine als benutzerdefinierte Deployment Procedure an. Gerade im Engineered System wird mithilfe des Virtual Assembly Builders ein komplexes (Teil-)System aufgebaut *(siehe Abbildung 2)*.

Im DevOps-Sinne werden natürlich auch diese Bausteine (benutzerdefinierte Deployment Procedures, Packaged Assemblies) als Produkt gesehen und dementsprechend verwaltet. Cloud Control bietet hier als Bordmittel die Unterstützung zum Protokollieren von Versionen und Ausführungszeiten, um nachvollziehen zu können, wer wann welche Umgebung aufgebaut hat.

Cloud Control geht weiter

Die Bereiche „Konfigurations-Management", „Provisioning- und Patch-Management" sowie „Change-Management" wurden stark erweitert und derart miteinander kombiniert, dass diese drei Management-Packs nun im Rahmen des Gesamtpacks als „Lifecycle Management Pack" angeboten werden. So lassen sich nun auch Konfigurationseinstellungen, die in entsprechen-

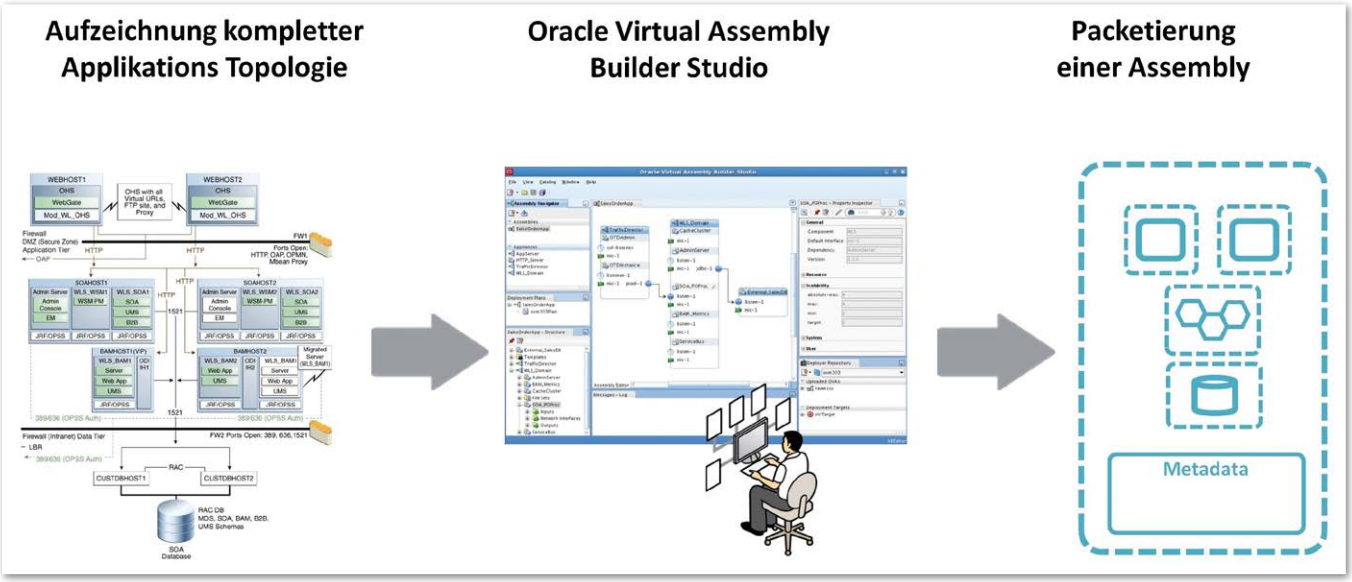


Abbildung 2: Schritte des Virtual Assembly Builders

den Dateien (wie „listener.ora“) getroffen werden, überwachen und aufzeichnen.

Auch die Analysemöglichkeiten sind stark erweitert. Mit grafischen Visualisierungen bekommt man zum Beispiel schneller einen Überblick über die eingesetzten Versionen einer Software. Eine Trend-Analyse zeigt anschaulich, wie sich die Versionsnutzung im Unternehmen entwickelt. Zudem ist die Suchfunktionalität in den aufgezeichneten Konfigurationsdaten verbessert und man kann seine Suchdefinitionen für eine regelmäßige Nutzung speichern.

DevOps und Patches

Die Provisionierung von Software (und dazu gehören auch Patches) ist mit Cloud Control noch einfacher und strukturierter realisierbar. Dabei kann die Zuständigkeit zur Vorbereitung und Durchführung einer Provisionierung sauber getrennt werden. Im Rahmen einer Vorbereitung wird eine Deployment Procedure vorbereitet, mit der wir ja schon durch die Bereitstellung vertraut sind.

Neue Provisioning Profiles legen fest, wie ein Software-Image zu provisionieren ist. Bei der Definition einer Deployment Procedure lassen sich Parameter festlegen, die später bei der Nutzung der Prozedur nicht mehr verändert werden dürfen (wie Verzeichnisstruktur). Damit können Standards noch besser durchgesetzt werden. Die eigentliche Provisionierung durch das Starten der Deployment-Prozedur wird von einem Operator durchgeführt, der für diese das Ausführungsrecht hat. Auch dieses Ausführungsrecht für jede einzelne Prozedur (und nicht als Gesamtrecht) ist neu. Der Operator kann nur freigegebene Parameter eingeben. Fehlgeschlagene Ausführungen werden per Benachrichtigung an die entsprechenden Benutzer weitergeleitet.

Im Rahmen des Patch-Managements führt Cloud Control auch „Out of Place“-Upgrades durch. Dabei wird das aktuelle Oracle-Home der zu patchenden Datenbank kopiert (Cloning) und die neue

Venisolvere

Venisolvere ist eine Initiative von Tobias Weih, Dirk Ruchatz und Andreas Chatziantoniou. Alle drei sind Experten in IT Architektur, Cloud Control und Fusion Middleware mit zusammen mehr als 50 Jahren IT-Erfahrung bei verschiedenen Kunden in diversen Branchen. Die Kernkompetenz liegt in der Umsetzung von großen Architekturen in komplexen Oracle-Fusion-Middleware-Umgebungen über alle Phasen des Software-Lifecycles. Venisolvere bietet diese Erfahrung (CI und CD) als im Cloud Control paketierte Lösung beziehungsweise Pre-build-Umgebung an.

Software zunächst auf diese Kopie eingespielt. Während dieses Vorgangs läuft die Datenbank ohne Downtime. Ist die neue Software jetzt installiert, wird die Datenbank herunter- und mit der neuen Software wieder hochgefahren. Je nach Patch werden dann noch SQL-Skripte gestartet. Diese Patch-Methode findet auch in den neuen Patchsets der Datenbank Anwendung.

Fazit

Das Cloud Control ist notwendig, um komplexe Umgebungen zu überwachen, und zeigt seinen Wert erst Recht im Einsatz des Provisioning. Mit den vorgefertigten Bausteinen von Venisolvere wird der Übergang zu DevOps im Oracle-Fusion-Middleware-Umfeld leichter.

Andreas Chatziantoniou
andreas@foxglove-it.nl

Tobias Weih
obias.weih@cabp.de

Dirk Ruchatz
itc-dr@gmx.de

Production Redeployment von ADF-Anwendungen

Ulrich Gerkmann-Bartels und Andreas Koop, enpit

Durch die Einführung von agilen Vorgehensweisen sind nachweislich Ergebnisse aus der Software-Entwicklung früher sichtbar und auch nutzbar. In vielen Fällen, in denen „Feature Driven Development“ und „Continuous Integration“ erfolgreich im Einsatz sind, zeigen sich jedoch manchmal neue Hindernisse aus Architektur- und Betriebssicht.

Zwei häufig auftauchende Herausforderungen sind der Anwendungsschnitt einer Applikation und die Bereitstellung von neuen Features oder Modulen mit minimaler Unterbrechung der im Einsatz befindlichen Anwendung. Der Artikel beschreibt, welche Voraussetzungen und Vorgehensweisen im Deployment von ADF-Anwendungen eingeführt werden können, um ein Production Redeployment dieser Anwendungen und Bibliotheken einzuführen.

Ausgangslage

Möchte man Flexibilität, Granularität und Änderungshäufigkeit in der Software-Entwicklung mit der Bereitstellung und Hochverfügbarkeit einer Anwendung im Betrieb kombinieren, treffen zwei unterschiedliche Zielvorstellungen aufeinander. In der Software-Entwicklung möchte man nach agilem Prinzip oftmals wie folgt verfahren: „Das eine Feature bringe ich noch rein“, frei übersetzt nach dem Ausspruch „Den Bug behebe ich noch“, um die Fachlichkeit laufend mit Funktionalität zu erfreuen. Demgegenüber stehen betriebliche Aspekte, die vielleicht nach dem Motto „Never Change a running System!“ ihre bestehenden Systeme mit Sicherheit zu schützen wissen. Neben der methodischen Vorgehensweise, „DevOps“ in Unternehmen einzuführen, gilt es zu erörtern, welche Möglichkeiten der Oracle-Fusion-Middleware-Stack bietet, insbesondere WebLogic Server und

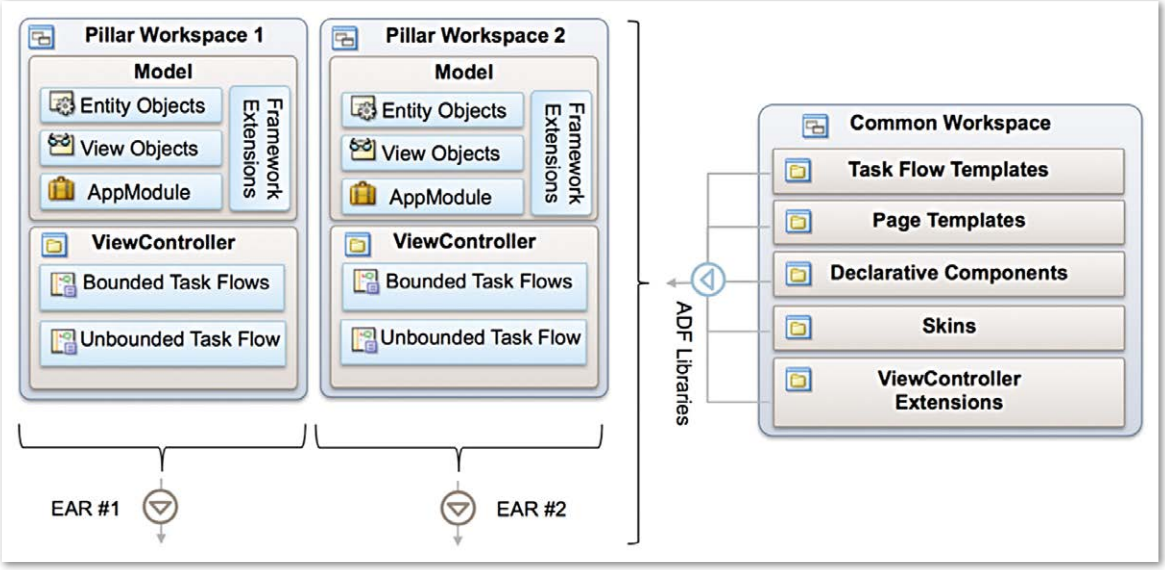


Abbildung 1: Pillar-Architektur „ADFArchitecture Fundamentals" (siehe „http://www.youtube.com/watch?v=toEuQvp73h8", Chris Muir)

Oracle ADF, um eine sichere, modulare und im Zweifel zurücksetzbare Anwendungsbereitstellung umzusetzen.

Architektur

Möchte man Flexibilität, Granularität und Änderungshäufigkeit in der Anwendungsentwicklung nutzen, ist es hilfreich, sich grundsätzlich einige Gedanken darüber zu machen, wie die Applikation und die beinhalteten Module aus dem Blickwinkel der Informations-Architektur und aus Sicht der Strukturierung der Workspaces/ Projekte geschnitten sind:

- Was lässt sich auf welcher Ebene modularisieren?
- Inwieweit beherrschen technische Aspekte den fachlichen Schnitt?
- Bewegt sich die Software-Architektur hinsichtlich der bereitzustellenden Artefakte zu einem Riesenmonolithen in der Ausprägung eines 300 MB Enterprise Archiv Files (EAR)?
- Wie können andere funktionale Bausteine in Projekten wiederverwendet werden und gegebenenfalls ihren eigenen Lebenszyklus haben?

Eine Architektur, die bezüglich dieser Aspekte grundsätzlich eine Antwort gibt, ist die sogenannte Pillar-Architektur *(siehe Abbildung 1)*.

Dieser Entwurf basiert auf dem Ansatz, dass fachliche Module zu einem Enterprise Archiv (EAR) gebündelt und durch eine entsprechende ADF-Library mit allgemeinen, nicht fachlichen Komponenten versorgt werden. Die jeweiligen Enterprise Archive sind unabhängig voneinander bereitstellbar und unterliegen ihrem eigenen Lebenszyklus. Ein gemeinsamer Kontext zwischen den Modulen kann gegebenenfalls durch einen „ContextStore" in der Datenbank oder durch einen Coherence-Cluster abgebildet werden *[1]*. Ausgehend von dieser Architektur stellen sich für das Deployment der Anwendung folgende Fragen:

- *Ist es möglich, ein neues fachliches Modul in Form eines EAR im laufenden Betrieb bereitzustellen?*
  Ja, durch die Option „Production Redeployment"
- *Ist es möglich, die nicht fachlichen Komponenten einer ADF Library für alle Applikationen zentralisiert zur Verfügung zu stellen, um bei neuen Features und Fehlerbehebungen nicht alle Anwendungen neu zu bauen und bereitzustellen?*
  Ja, durch den Einsatz von „WebLogic Shared Library"
- *Ist es möglich, verschiedene Versionen einer WebLogic Shared Library bereitzustellen?*
  Ja, durch den expliziten Verweis auf die genutzte Version in der entsprechenden Applikation

Production Redeployment

Production Redeployment ermöglicht es, im WebLogic Server zwei Versionen einer Anwendung gleichzeitig durch die Angabe unterschiedlicher und aufsteigender Versionsnummern auszuführen. Um die Versionierung von Applikationen im WebLogic Server zu aktivieren, muss dem Archiv

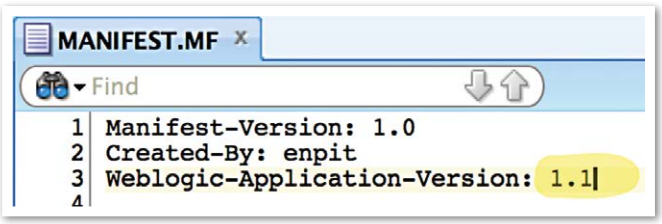


Abbildung 2: Manifest mit WebLogic-Application-Version

☐	enpit-common-war-lib(1.0,1.0.5)	Active		Library
☐	enpit-common-war-lib(1.0,1.0.6)	Active		Library
☐	enpittestcommons-reflib.war (1.0)	Retired		Web Application
☐	enpittestcommons-reflib.war (1.1)	Active	OK	Web Application

Abbildung 3: Retired Web Application nach Timeout der letzten Sessions

ein entsprechendes Manifest unter „/src/META-INF/MANIFEST.MF“ mit den notwendigen Properties hinzugefügt werden *(siehe Abbildung 2)*.

Vor dem Deployment muss sichergestellt sein, dass eine Applikation mit derselben Versionsnummer nicht existiert. Sollte dies der Fall sein, lässt sich die Anwendung nicht bereitstellen. Durch zusätzliche Parameter kann die Version auch beim Deployment beeinflusst werden. Aktive Anwendungssitzungen werden bei der Bereitstellung mit der vorherigen Version weitergeführt. Es ist kein Neustart der gesamten Anwendung notwendig *(siehe Abbildung 3)*.

Nach Ablauf aller Anwendungssitzungen der Anwendung in Version 1.0 wird der Status der Applikation in den Modus „Retired“ versetzt. In diesem Zustand nimmt die Anwendung keine neuen Sessions mehr an. Diese werden ausschließlich von der Anwendung in Version 1.1 bedient *(siehe Abbildung 4)*.

Um im nächsten Release-Zyklus eine neue Version (1.2) zur Verfügung zu stellen, muss die Anwendung mit dem Status „Retired“ entfernt werden. Dies kann leicht vor dem Deployment automatisiert durch ein WLST-Script erfolgen, das über alle Applikationen mit dem gewünschten Applikationsnamen iteriert und sich die Version der entsprechenden Retired-Applikation merkt. Dies ist unbedingt notwendig, da zur Entfernung einer versionierten Anwendung die entsprechende Versionsnummer beim „Undeployment“ notwendig ist.

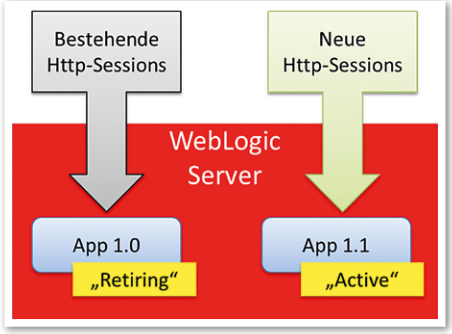


Abbildung 4: Production Redeployment – oder auch Side-by-Side-Deployment

WebLogic Shared Library

Neben versionierten Deployments von Applikationen erlaubt der WebLogic Server auch die Versionierung sogenannter „Shared Libraries“. Wie der Name schon verrät, lassen sich damit Bibliotheken in einer WebLogic-Domain zentral zur Verwendung durch mehrere Applikationen bereitstellen. Im Kontext von ADF-Anwendungen lassen sich damit einzelne fachliche Module oder auch Module mit Querschnittsfunktionalitäten als Shared-ADF-Library bereitstellen. Ähnlich den versionierten Deployments von EARs beziehungsweise WARs können auch die Bibliotheken versioniert bereitgestellt sein. Um eine Shared Library bereitzustellen, gehe man wie folgt vor:

- Erstellung einer „/META-INF/MANIFEST.MF“ mit folgenden Eigenschaften: „Extension-Name: enpit-common-war-lib“, „Specification-Version: 1.0“ und „Implementation-Version: 1.0.5“
- Paketierung des gewünschten Bibliotheksinhalts als WAR. Dies kann beispielsweise über ein JDeveloper-Deploymentprofil erfolgen oder auch mit jedem beliebigen anderen Tool. Entscheidend ist, dass die unter Punkt 1 genannten Metadaten enthalten sind. Die Best Practice bei ADF-Libraries ist, dass das resultierende JAR im Shared-Library-WAR unter „WEB-INF/lib“ gebündelt ist. Die Shared-WAR-Library kann in diesem Zusammenhang auch als „Proxy-Library“ für die ADF-Library bezeichnet werden.
- Deployment des WAR als Bibliothek im WebLogic Server. Je nach eingesetzter Technik muss beim Deployment angegeben werden, dass es sich um eine Library handelt. Am Beispiel des „weblogic.Deployer“-Kommandozeilen-Werkzeugs sieht es dann wie folgt aus: „java weblogic.Deployer -adminurl t3://eden.local:7001 -username weblogic -password welcome1 -upload -library -targets AdminServer -deploy -source enpit-shared-lib-war.war“.

Als Ergebnis erhält man eine Shared Library, deren Version und Verfügbarkeit in der Admin Console verifiziert werden kann *(siehe Abbildung 5)*.

Zur Verwendung einer Shared Library muss die entsprechende Applikation eine Library-Reference im WebLogic-spezifischen Deployment-Deskriptor spezifiziert sein („weblogic.xml“, siehe *Listing 1*).

Optional können Spezifikations- und Implementierungs-Versionsnummern mitgegeben werden. Fehlen diese, bedeutet dies, dass die Applikation stets die Bibliothek mit der höchsten Versionsnummer verwenden soll. Um neue Versionen bereitzustellen, muss die Version in der „MANIFEST.MF“ geändert und das anschließend paketierte WAR mit dem Standard-Library-Deployment-Befehl bereitgestellt sein. Man kann auf diese Weise beliebig viele Versionen einer Shared Library installieren (beim Application Production Redeployment können maximal zwei Versionen aktiv sein). Zu beachten sind für Shared Libraries folgende Bedingungen/Einschränkungen:

- Eine Bibliotheksversion kann nicht entfernt werden, solange mindestens eine Applikation darauf verweist
- Alle referenzierten Bibliotheken müssen auf denselben Servern bereitgestellt sein, auf denen die Applikation installiert ist
- Eine neue Bibliotheksversion wird für eine Anwendung, die die Bibliothek ohne Versionsnummern referenziert, erst maßgeblich, sobald die Anwendung erneut gestartet wird

Bibliotheken, die ein ADF-Library-JAR in „/WEB-INF/lib“ enthalten, müssen unbedingt per „library-ref“ in der „weblogic.

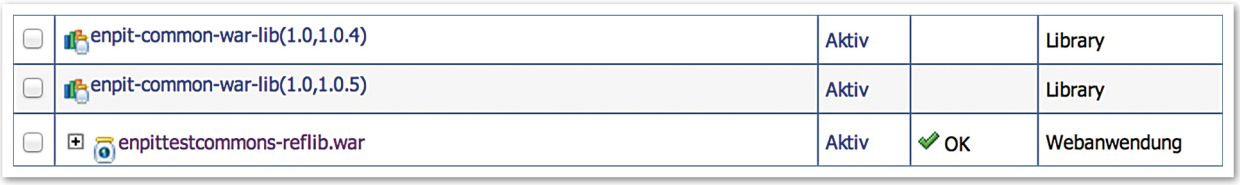


Abbildung 5: Shared-Library-Übersicht in der Admin-Console

```
<?xml version='1.0' encoding='UTF-8'?>
<weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/weblogic-web-app.
xsd" xmlns="http://www.bea.com/ns/weblogic/weblogic-web-app">
  <library-ref>
       <library-name>enpit-common-war-lib</library-name>
  </library-ref>
</weblogic-web-app>
```

Listing 1

xml" und nicht in der „weblogic-application.xml" referenziert sein. Andernfalls gibt es eine Exception zum Zeitpunkt des Deployments *[2]*. Weitere Einzelheiten und Beispiele zu Shared Libraries im WebLogic finden sich im Blog-Beitrag unter *[3]*.

Möglichkeiten

Kombiniert man jetzt alle Möglichkeiten, so erlaubt diese Methode auch die Realisierung fachlicher Module, die Anwendungen von außen integrieren. Die Pillar-Architektur kommt an dieser Stelle an ihre Grenzen. Realisiert mit ADF-Task-Flows und verpackt in einer ADF-Library, die durch eine WebLogic Shared Library der konsumierenden Anwendung zur Verfügung gestellt wird, ist dies möglich. Die entsprechende WebLogic Shared Library dient als Proxy für die nutzende Anwendung. Ein abgestimmter Build- und Deployment-Prozess führt eine modulare und flexible Bereitstellung durch. Dass das alles nicht nur blanke Theorie ist, zeigt die Entwicklung eigener ADF-Task-Flows für WebCenter Portal ab der Version 11.1.1.8, siehe *[4]*.

Fazit

Wie bei allen Dingen gibt es natürlich auch bei Nutzung der skizzierten Vorgehensweisen Grenzen, die man nicht unterschätzen sollte. Insbesondere die Herausforderung, zwei gleiche Versionen einer Software für einen bestimmten Zeitraum zur Verfügung zu stellen, oder gegebenenfalls das Zurückfahren auf eine vorherige Version kann erhebliche Komplexität beinhalten. Die Grenzen beginnen natürlich bei der Integration und hier sicherlich als Erstes im Backend. Insbesondere, wenn es sich dabei um Anwendungen handelt, die die Datenbank weitreichend für Logik und Verarbeitung nutzen.

Basieren die neuen Versionen der einzelnen Bausteine und Applikationen auf entsprechenden Datenbank-Schemata-Änderungen im Backend, so sind diese entsprechend in eine Rollback-Strategie mit einzuplanen. Unter Umständen kann eine gleichzeitige Nutzung verschiedener Versionen bei einer Änderung im Backend auch ausgeschlossen sein. In diesem Fall bleibt immer noch die Möglichkeit, eine entsprechende Bereitstellung in einem Wartungsfenster durchzuführen. Der Vorteil, die Artefakte durch die eingeführte Versionsnummer identifizieren zu können, bleibt bestehen.

Bestehen Abhängigkeiten zu Web-Services- oder Rest-Schnittstellen, sind diese ebenso zu berücksichtigen, können aber mit entsprechenden Konzepten bezüglich der Versionierung und des Lifecycle durch eine vorhandene Governance geregelt werden und ermöglichen die Verfügbarkeit verschiedener Versionen zur Nutzung in den jeweiligen Modulen.

Nicht zuletzt sollte man sich gut überlegen, wieweit man die Modularisierung der einzelnen Module in eigene WebLogic Shared Libraries treibt, sonst hat man sich eines Tages einen eigenen DLL-Hell *[5]* erschaffen.

Referenzen und weitere Hinweise

[1] Continuous Delivery in ADF-Projekten: http://de.slideshare.net/ugb/20130619-continuousdeliveryinadfprojekten (2013)

[2] Production Redeployment with ADF Shared WebLogic Libraries: http://multikoop.blogspot.de/2013/06/production-redeployment-with-adf-shared.html

[3] WebLogic Application redeployment using shared libraries - without downtime: http://multikoop.blogspot.de/2013/06/weblogic-application-redeployment-using.html

[4] S. 1340 ff., Developing Portals with Oracle WebCenter Portal and Oracle JDeveloper, 11gR1 (11.1.1.8), E27739-01, Juli 2013

[5] DLL-Hell: http://en.wikipedia.org/wiki/DLL_Hell

[6] Deploying Applications to Oracle WebLogic Server 11g Release 1 (10.3.6): http://docs.oracle.com/cd/E23943_01/web.1111/e13702/redeploy.htm

[7] WebLogic Tutorial for Production Redeployment: http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/14-Redeployment--4465/redeploy.htm

Ulrich Gerkmann-Bartels
ulrich.gerkmann-bartels@enpit.de

Dipl.-Inf. Andreas Koop
andreas.koop@enpit.de

Neu: Oracle NoSQL DB 3.0

Carsten Czarski, ORACLE Deutschland B.V. & Co KG

Seit April 2014 steht die neue Version 3.0 zum Download bereit. Neue Funktionen wie ein „Tabellen-API“ lassen aufhorchen: Tabellen in einer NoSQL DB? Dieser Artikel gibt einen Überblick über die neuen Funktionen – neben dem Tabellen-API gibt es neue Security-Funktionen und Erweiterungen im Cluster-Management.

Die neue Version führt eine Tabellen-Schnittstelle in der NoSQL DB ein. Das bedeutet aber nicht, dass aus der NoSQL- nun eine SQL-Datenbank wird, vielmehr bekommt der Entwickler eine Schnittstelle zur Definition von Tabellenstrukturen – also zum Lesen und Schreiben von Daten in Form von Zeilen und Spalten. Alle Vorgänge werden dabei auf die Key-Value-Technologie abgebildet; die ab Version 2 unterstützten „Avro Schemata“ können ebenfalls zur Definition von Tabellen verwendet werden. *Abbildung 1* zeigt die Zusammenhänge zwischen Tabellenmodell, AVRO-Schemata und Key-Value-Paaren.

Die Definition von Tabellen erfolgt mithilfe des Command Line Interface (CLI) der NoSQL DB, mit der (wie in den vorherigen Versionen) die NoSQL-Datenbank initial aufgesetzt wird. Wie dieser Setup im Detail aussieht, soll hier nicht vertieft werden, die Dokumentation (siehe „Weitere Informationen“) gibt hierzu Auskunft. *Listing 1* zeigt, wie eine Tabelle mit dem CLI erstellt wird.

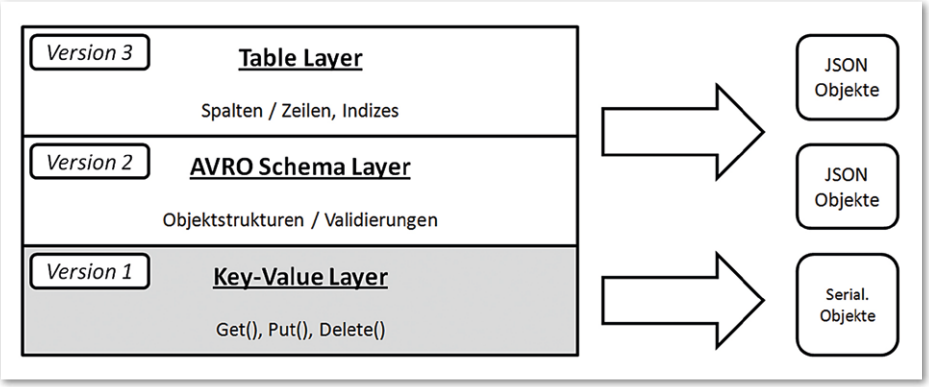


Abbildung 1: Auch das neue Tabellenmodell speichert die Daten tatsächlich als Key-Value-Paare ab

```
kv-> table create -name EMP
EMP-> add-field -type INTEGER -name empno
EMP-> add-field -type STRING  -name ename
EMP-> add-field -type DOUBLE  -name sal
EMP-> add-field -type STRING  -name job
EMP-> primary-key -field empno
EMP-> exit
Table EMP built.

kv-> plan add-table -name EMP -wait
Plan 19 ended successfully
```

Listing 1: Definition einer Tabelle „EMP“ in der NoSQL DB

Es können numerische, alphanumerische und strukturierte Datentypen in Form von Records oder Arrays verwendet werden – „DATE“ oder „TIMESTAMP“ stehen jedoch noch nicht zur Verfügung. Der Primärschlüssel ist Pflicht, er wird logischerweise für den Key der als Key-Value-Paare gespeicherten Daten verwendet. Tabellen können auch in einer Parent-Child-Beziehung miteinander verknüpft sein. Das wird so abgebildet, dass der Primärschlüssel der Parent-Tabelle in die Child-Tabelle(n) übernommen wird. Sobald die Tabelle erstellt wurde, können Zeilen abgelegt werden – dies kann (wie immer bei der NoSQL DB) per Java-Programm geschehen *(siehe Listing 2)*. Alternativ kann auch mit dem CLI gearbeitet werden *(siehe Listing 3)* – die Tabellenzeile wird dann mit dem Befehl „put table“ im JSON-Format übergeben.

Analog dazu kann man mit dem Kommando „get table“ Zeilen lesen. Der Zugriff erfolgt hier allerdings allein über den Primärschlüssel oder (wie sich noch zeigen wird) über per „Secondary Index“ indizierte Spalten. Die freie Selektion, wie bei SQL-Datenbanken üblich, ist in der NoSQL-Datenbank nach wie vor nicht möglich.

Wie *Listing 4* zeigt, ist der Zugriff auf eine konkrete Tabellenzeile zunächst nur mit dem Primary Key möglich – man erkennt sofort, dass der „Key Value Store“ der NoSQL DB die technische Grundlage ist. Allerdings ist das noch nicht alles: Mit den neu eingeführten „Secondary Indexes“ wurde eine Möglichkeit geschaffen, auch über andere Spalten als den „Primary Key“ zu suchen. *Listing 5* zeigt, wie ein „Secondary Index“ erzeugt und genutzt wird.

```
import oracle.kv.*;
import oracle.kv.table.*;

public class write {
   public static void main(String args[]) {
       // Verbindung zur NoSQL DB öffnen
       KVStore store= KVStoreFactory.getStore (
           new KVStoreConfig(
               "workshopstore"
               new String[] {"sccloud031:10100", "sccloud031:10200"}
           )
       );
       // Handle zur definierten Tabelle EMP holen
       TableAPI tableH = store.getTableAPI();
       Table myTable = tableH.getTable("EMP");
       Row row = myTable.createRow();
       row.put("empno", 7839);
       row.put("ename", "MILLER");
       row.put("sal", 3000d);
       row.put("job", "SALESMAN");
       // Tabellenzeile speichern
       tableH.put(row, null, null);
       store.close();
   }
}
```

Listing 2: Das Java-Programm speichert eine Tabellenzeile in die NoSQL DB

```
kv-> connect store -host localhost -port 10100 -name workshopstore
Connected to workshopstore
Connected to workshopstore at localhost:10100.
kv-> put table
 -name EMP
 -json ,{"empno":815, "ename":"JONES", "sal": 3500, "job":"MANAGER"}`

Operation successful, row inserted.
```

Listing 3: Tabellenzeilen mit dem CLI in die NoSQL DB speichern

```
kv-> get table -name EMP
{"empno":7844,"ename":"FORD",  "sal":2500.0,"job":"CLERK"}
{"empno":815, "ename":"JONES", "sal":3500.0,"job":"MANAGER"}
{"empno":7839,"ename":"MILLER","sal":3000.0,"job":"SALESMAN"}

3 rows returned.

kv-> get table -name EMP -field "empno" -value 7844
{"empno":7844,"ename":"FORD","sal":2500.0,"job":"CLERK"}

kv-> get table -name EMP -field "ename" -value "SMITH"
Error handling command get table -name EMP -field "ename" -value
"SMITH": Field is not part of PrimaryKey: ename
```

Listing 4: Tabellenzeilen mit dem CLI aus der NoSQL DB abrufen

Die technische Grundlage eines „Secondary Index" ist einfach: Beim Erstellen generiert die NoSQL DB zusätzliche Key-Value-Paare, mit denen der Index modelliert wird. Dabei dienen die Inhalte der indizierten Spalte als Key – der „Primary Key" der indizierten Zeile ist dagegen der „Value". Beim Abrufen von Zeilen wird folgerichtig zunächst mithilfe dieses Index der Primärschlüssel und damit erst die konkreten Daten abgerufen. Da die NoSQL DB keinen Query-Optimizer besitzt, muss der „Index Lookup" von der Anwendung explizit angefordert werden – in *Listing 4* ist das am Parameter „-index" erkennbar.

Die in den Listings 4 und 5 vorgestellten Möglichkeiten lassen sich natürlich auch per API aus Java-Programmen heraus nutzen. Auf die bereits erwähnten „Child Tables" oder Array-Datentypen soll hier aus Platzgründen nicht mehr im Detail eingegangen werden – die Dokumentation der NoSQL DB enthält dazu weitere Informationen.

Security-Funktionen

Die bisherigen Versionen der Oracle NoSQL DB enthielten keinerlei Security-Funktionen; weder war ein Login zur Arbeit mit der NoSQL DB vorgesehen, noch wurden die Daten auf dem Transportweg in irgendeiner Art und Weise verschlüsselt.

Die vorliegende Version 3.0 ändert dies: Die Kommunikation über das Netzwerk kann nun mit SSL verschlüsselt und der Zugang zur NoSQL DB per Login geschützt werden. Eine bestehende NoSQL-DB-Installation lässt sich nachträglich sichern, indem per Admin-Werkzeug eine Security-Konfiguration hinzugefügt wird – das geschieht, grob skizziert, mit folgenden Schritten:

- Herunterfahren aller NoSQL-DB-Clusterknoten
- Erstellen einer neuen Security-Konfiguration auf einem Knoten per CLI
- Kopieren dieser Konfiguration (Verzeichnis „security") auf alle anderen Knoten
- Aktivieren der Konfiguration auf allen Knoten per CLI
- Starten aller Knoten
- Anonymer initialer Login
- Erstellen des ersten ADMIN-Users

Danach sind Zugriffe nur noch nach erfolgreichem Login möglich. Beim Erstellen der Sicherheitskonfiguration wird ein „External Password Store" festgelegt – für die Enterprise Edition kann und soll-

```
kv-> plan add-index -name idx_emp_ename -table EMP -field ename
Started plan 19. Use show plan -id 19 to check status.
      To wait for completion, use plan wait -id 19
kv-> plan wait -id 19
Plan 19 ended successfully

kv-> get table -name EMP -field ename -value FORD
Error handling command get table -name EMP -field ename -value FORD: Field is not part of PrimaryKey: ename
kv-> get table
  -name EMP -field ename -value FORD -index idx_emp_ename
{"empno":7844,"ename":"FORD","sal":2500.0,"job":"CLERK"}
```

Listing 5: Erstellung und Nutzung eines Secondary Index in der NoSQL DB

te ein Oracle-Wallet verwendet werden; die Community-Edition nutzt eine Passwortdatei. Wichtig ist, dass dort nicht die Passwörter der in der NoSQL DB eingerichteten User abgelegt sind, sondern vielmehr das Passwort zum Zugriff auf den Java-Keystore – der wiederum enthält die Schlüssel für die SSL-Kommunikation. Die User-Verwaltung, also das Anlegen und Verwalten von Nutzerkonten, findet mit dem Admin-CLI statt. *Listing 6* zeigt das Erzeugen eines neuen Users („SCOTT“).

Zugriffe auf die NoSQL DB erfordern nun ein Login – das Java-Programm in *Listing 2* würde nun so nicht mehr funktionieren. *Listing 7* zeigt die zur Verbindung auf eine gesicherte NoSQL DB erforderlichen Java-Codeabschnitte.

In der aktuellen Version 3.0 dienen Usernamen allein dem Login: Ein Rechtekonzept, nach dem unterschiedliche User unterschiedliche Dinge tun dürfen, gibt es (noch) nicht. Ein Login ist nun auch erforderlich, wenn Daten, wie schon in *Listing 3* dargestellt, mit dem CLI geschrieben oder gelesen werden sollen *(siehe Listing 8)*.

Weitere neue Funktionen

Neben dem Tabellen-API und den Security-Funktionen sind noch weitere neue Features im Release enthalten:

- Die Knoten eines NoSQL-DB-Clusters können in verschiedene Zonen unterteilt sein. Knoten in der „Primary Zone“ arbeiten wie bisher – sie können als Master und als Replika fungieren. Fällt ein Master aus, so wählen die Replikas einen neuen Master. Knoten in der „Secondary Zone“ fungieren dagegen nur

```
$ java -jar /opt/oracle/nosqldb/sw/lib/kvstore.jar runadmin
   -host localhost -port 10100
   -security KVROOT1/security/client.security

Could not login as anonymous: Authentication failed (12.1.3.0.5)
Login as:root
root's password: ******
Logged in admin as root
Redirecting to master at rmi://sccloud031:10300

kv-> plan create-user -name SCOTT -password tiger -wait
Executed plan 21, waiting for completion...
Plan 21 ended successfully
```

Listing 6: Einrichtung eines neuen Nutzerkontos in der NoSQL DB

```
:
KVStoreConfig conf = new KVStoreConfig(
 "workshopstore",
 new String[] {"sccloud031:10100", "sccloud031:10200"}
);

// SSL-Kommunikation einschalten
Properties secProps = new Properties();
secProps.setProperty(
 KVSecurityConstants.TRANSPORT_PROPERTY,
 KVSecurityConstants.SSL_TRANSPORT_NAME
);
secProps.setProperty(
 KVSecurityConstants.SSL_TRUSTSTORE_FILE_PROPERTY,
 "/opt/oracle/nosqldb/ex/client.trust"
);
conf.setSecurityProperties(secProps);

// konkrete Verbindung mit Authentifizierung
KVStore store= KVStoreFactory.getStore (
 conf,
 new PasswordCredentials("SCOTT", new String("tiger").toCharArray()),
 null
);
:
```

Listing 7: Java-Verbindung zur NoSQL DB: Passwort-Login und SSL-Kommunikation

```
kv-> connect store -host localhost -port 10100 -name workshopstore
-security KVROOT1/security/client.security
Login as:SCOTT
SCOTT's password:******
Connected to workshopstore
Connected to workshopstore at localhost:10100.

kv-> get table -name EMP
{"empno":8000,"ename":"MILLER","sal":3000.0,"job":"SALESMAN"}
{"empno":7844,"ename":"FORD","sal":2500.0,"job":"CLERK"}
:

4 rows returned.
```

Listing 8: Arbeit mit dem CLI auf einer gesicherten NoSQL-DB-Umgebung

als Replika – auch wenn ein Master ausfällt, nehmen sie nicht an der Wahl des neuen Masters teil.

- Die neue Lesekonsistenz-Stufe „NONE_REQUIRED_NO_MASTER“ erzwingt das Lesen von einer Replika. Das ist sinnvoll, wenn die Ressourcen der Master-Knoten (obwohl sie zu einem Zeitpunkt theoretisch frei wären) dennoch (etwa für andere Zugriffe) geschützt werden sollen.

Fazit

Das neue Release 3.0 bringt die Oracle NoSQL DB einen großen Schritt nach vorn. Das Tabellen-API erlaubt das Hinterlegen von Datenstrukturen – die Praxis hat in vielen Fällen schon gezeigt, dass es ohne diese nicht geht – auch nicht in der Welt der NoSQL-Datenbanken. Damit können sich Entwickler auf eine Tabellenstruktur einigen, diese kontrolliert pflegen und weiterentwickeln. „Secondary Indexes“ erlauben einfache Zugriffe über ein Attribut, das nicht als Schlüssel definiert wurde – diese fertige Funktionalität spart große eigene Programmieraufwände. Dennoch bleibt die Datenbank eine NoSQL-Datenbank – ein Query Optimizer existiert nicht und das freie Formulieren von Abfragen ist nicht möglich.

Security-Funktionen sind im Unternehmenseinsatz ein Muss – das schließt sowohl die verschlüsselte Kommunikation als auch den Schutz der Installation durch Authentifizierung ein. Beides wird mit dem vorliegenden Release eingeführt.

Wer NoSQL-Technologien einsetzt oder sich damit beschäftigt, sollte sich die neue Oracle NoSQL DB auf jeden Fall näher ansehen – dem Produkt ist deutlich anzumerken, dass es für den Unternehmenseinsatz im Rechenzentrum gewappnet ist.

Weitere Informationen

[1] Download Oracle NoSQL DB
http://www.oracle.com/technetwork/products/nosqldb/downloads/index.html

[2] Oracle NoSQL DB Dokumentation
http://docs.oracle.com/cd/NOSQL/html/index.html

[3] Oracle NoSQL DB 3.0: Complete List of Changes
http://docs.oracle.com/cd/NOSQL/html/changelog.html

[4] Oracle NoSQL DB Frequently Asked Questions
http://www.oracle.com/technetwork/database/nosqldb/overview/nosqldb-faq-518364.html

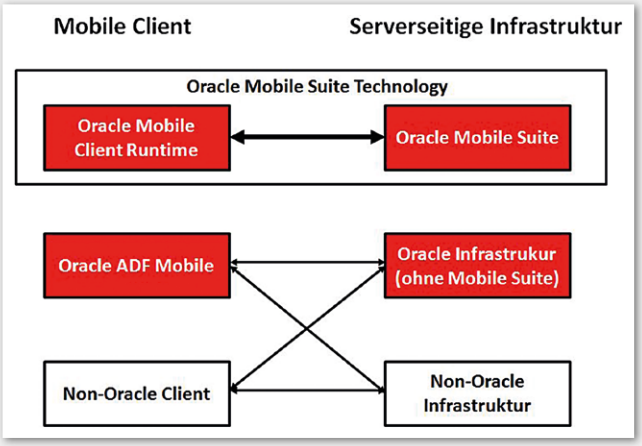
Carsten Czarski
Carsten.Czarski@oracle.com
http://twitter.com/cczarski
http://sql-plsql-de.blogspot.com

Artikel „Neue Oracle Product Suites für mobile Anwendungen“ von Dr. Jürgen Menge, ORACLE Deutschland B.V. & Co. KG, in der letzten Ausgabe auf Seite 22

Leider haben sich in dem Artikel Fehler eingeschlichen, für die sich der Autor entschuldigt.

Die Oracle Mobile Suite besteht aus folgenden Komponenten:

- Oracle Service Bus
- Oracle Applications Adapter
- Technologie-Adapter



Die Abbildung zeigt die korrekten Zusammenhänge zwischen den Produkten.

Datenbanken konsolidiert in der Cloud

Heiko Eitner, Landesbetrieb Daten und Information Rheinland-Pfalz, und Marco Mischke, Robotron Datenbank Software GmbH

In der heutigen Zeit steigt die Zahl der Datenbanken ebenso wie die Anforderungen an deren Verfügbarkeit. Gleichzeitig sollen die Betriebskosten aber oftmals noch sinken. Diese konträren Zielsetzungen lassen sich durch den Einsatz eines Policy-managed RAC-Clusters erreichen. Die Autoren zeigen den beschrittenen Weg dorthin mit all seinen Fehlschlägen und Problemen.

Am Anfang des Weges stand ein vorhandener Fünf-Knoten-RAC-Cluster, der lediglich für eine einzige Datenbank reserviert war. Die Last auf dem Gesamtsystem war praktisch vernachlässigbar. Grund für den Einsatz von RAC war die Absicherung gegen Ausfälle einzelner Server. Daneben existierten noch eine ganze Reihe weiterer produktiver Datenbanken, die alle auf jeweils einem eigenen dedizierten Server liefen. Die Ausgangslage war also folgende:

- Fünf Server-RAC, eine Datenbank
- Neun Single Server, neun Datenbanken

Diese 14 Server mit ihren zehn Datenbanken stellen die Grundlage für den täglichen Betrieb dar. Darum gliederten sich noch entsprechende Test-, Entwicklungs- und Integrationssysteme, diese bleiben hier aber außen vor. Die produktiven Datenbanken können in drei verschiedene Klassen entsprechend ihrer Bedeutung für den täglichen Betrieb eingeteilt werden. Daraus resultiert diese Klassifizierung der Datenbanken:

- *High*
  Drei Datenbanken
- *Medium*
  Vier Datenbanken
- *Low*
  Drei Datenbanken

Die Rahmenbedingungen

Für die Konsolidierung der Datenbank-Landschaft wurde eine Reihe von Zielen definiert, die es möglichst effizient umzusetzen galt:

- *Hohe Flexibilität*
  Änderungen an der Landschaft wie neue Datenbanken, andere Priorisierung oder steigende Arbeitslasten sollen einfach handhabbar sein.
- *Optimale Auslastung der Hardware*
  Die Last soll möglichst gleichmäßig über alle Server verteilt werden können, sodass weder ruhende noch überlastete Server existieren.
- *Minimierung der finanziellen Aufwände*
  Kosten für Lizenzen, Anschaffungskosten der Server, Kosten für Energie und Kühlung sowie Personalkosten sollen soweit als möglich gesenkt werden.
- *Priorisierung entsprechend definierter SLAs*
  In Fehlerfällen müssen die Datenbanken mit dem höchsten SLA am längsten überleben beziehungsweise im Fall eines Desasters am schnellsten wieder verfügbar sein.
- *Optimierung der Administration*
  Einrichtung, Betrieb und Maintenance sollen standardisiert werden, um Aufwände und Fehler zu minimieren. Die nötigen Handlungsabläufe sollen entsprechend dokumentiert sein. Dies soll ebenfalls Vertretungsregelungen vereinfachen.
- *Verwendung verschiedener Releases*
  Die eingesetzten Applikationen stellen verschiedene Anforderungen und Bedingungen an die verwendete Datenbank-Version. Dies soll innerhalb der neuen Umgebung umsetzbar sein.

Die neue Lösung musste alle genannten Punkte abdecken. Ist ein Punkt nicht oder unzureichend umgesetzt, so wird die Lösung entsprechend verworfen.

Die Theorie

Um ein besseres Verständnis der Thematik zu bekommen, ist es wichtig, die Begriffe „Server“, „Serverpool“ und „Service“ zu verstehen. Ein Server repräsentiert einen physikalischen Rechner, der Teil eines Clusters ist. Serverpools fassen mehrere Server zu einer Gruppe zusammen. Dabei können minimale und maximale Anzahl der Server im Pool sowie die Priorität des Serverpools festgelegt werden. Diese Attribute beeinflussen die Startreihenfolge wesentlich.

Zuerst werden alle definierten Serverpools in der durch die Priorität festgelegten Reihenfolge bis zum Erreichen des Minimums mit Servern bestückt. Ist die minimale Anzahl von Servern in allen Serverpools erreicht, so wird der Serverpool mit der höchsten Priorität bis zum Erreichen der maximalen Anzahl an Servern befüllt. Danach folgen alle anderen Serverpools in der Reihenfolge ihrer Priorität.

Services sind ein Instrument, um Datenbanken anzusprechen. Man kann hier festlegen, welche Datenbanken welchen Service zur Verfügung stellen und auf welchem Serverpool der Service laufen soll. Weiterhin werden im Service entsprechende Attribute definiert, die das Verhalten im Fehlerfall bestimmen.

Erste Idee: Child Serverpools

Von Anfang an war klar, dass die neue Clusterlösung unter Verwendung von Serverpools realisiert werden muss. Dabei stießen die Autoren auf das Problem, dass ein Datenbank-Service nur genau ei-

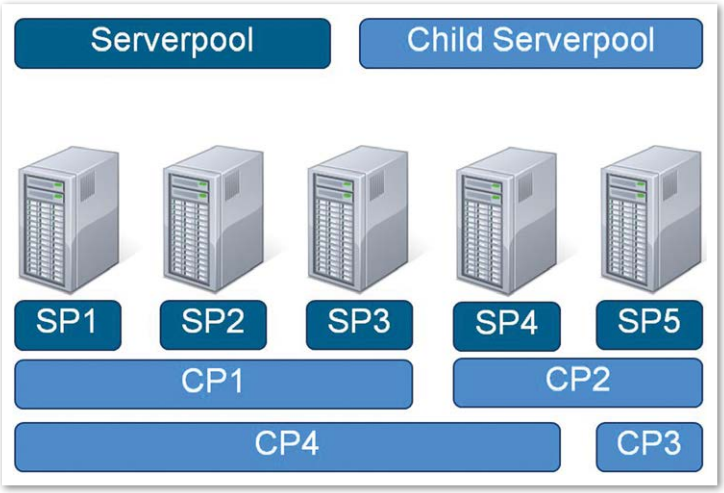


Abbildung 1: Cluster-Aufbau mit Child Serverpools

nem Serverpool zugeordnet werden kann. Es ist mit mehreren Serverpools also nie möglich, einen Service auf allen Servern im Cluster laufen zu lassen. Beim Studium der Clusterware-Dokumentation entdeckten sie dann die Möglichkeit, Child Serverpools einzurichten. Diese können mehrere Serverpools als Parent besitzen und es dürfen auch mehrere Child Serverpools auf einem Server laufen.

Die Idee war nun, einen Child Serverpool pro Service einzurichten. Dazu wurden fünf Serverpools mit absteigender Priorität und der Eigenschaft „min=max=1" erstellt. Das bedeutet, dass beim Starten des Clusters zuerst der Serverpool mit der höchsten Priorität auf dem ersten hochgefahrenen Server gestartet wird. Der zweite Serverpool wird dann entsprechend auf dem zweiten hochgefahrenen Server gestartet etc. Damit ist die Startreihenfolge der Serverpools unabhängig von der Startreihenfolge der Server.

Nun werden entsprechend Child Serverpools für jede Datenbank eingerichtet und diesen die gewünschten Serverpools entsprechend ihrer Einteilung zugewiesen. Dabei kann die Anzahl der Server im Child Serverpool sowie die Startreihenfolge und somit die Priorisierung frei gewählt werden. *Abbildung 1* zeigt den geplanten Aufbau im Cluster.

Diesem Aufbau entsprechend wurde nun versucht, die entsprechenden Serverpools anzulegen. Dazu musste das „crsctl"-Tool verwendet werden, da nur mit diesem das Festlegen eines Parent Serverpools möglich ist. Der erste Versuch scheiterte jedoch, da die so eingerichteten Child Serverpools nicht per „srvctl" einer Datenbank zugeordnet werden können. Die Clusterware erwartet offenbar intern noch den Prefix „ora." für diese Serverpools. *Listing 1* zeigt dies.

Daraufhin wurden die Child Serverpools entsprechend neu angelegt und nun jeweils der Präfix „ora." für die Namensgebung verwendet. Auch dieser Versuch scheiterte, wie in *Listing 2* zu sehen ist.

Wie sich herausstellte, darf nur „srvctl" zur Einrichtung von Serverpools für Policy-managed Datenbanken verwendet werden. Auch Tools wie „dbca" bieten die mit „crsctl" angelegten Serverpools beim Einrichten von Datenbanken gar nicht erst an.

So wurde das Serverpool-Konzept erneut mit „srvctl" eingerichtet. Nachdem die Serverpools entsprechend eingerichtet und eine Datenbank in einen zukünftigen Child Serverpool gelegt war, musste nur noch das „PARENT_POOL"-Attribut dieses Serverpools per „crsctl" modifiziert werden, da „srvctl" keine solche Option besitzt. Allerdings wechselte die Datenbank dadurch plötzlich von „policy managed" in „administrator managed" *(siehe Listing 3)*.

Ein Service Request stellte klar, dass die Benutzung von Child Serverpools für Datenbanken nicht unterstützt wird. Der zugehörige Bug 16369884 „ADDING SUBPOOL CHANGES POLICY MANAGED TO ADMINISTRATOR MANAGED DB" wurde entsprechend mit „Status: 32 – Not a Bug. To Filer" geschlossen.

Finale Idee: Policy-managed Databases

Nach diesen Fehlschlägen waren die Autoren gezwungen, ihr Konzept zu überdenken. Letztendlich haben sie für jede der drei eingangs definierten Gruppen (high, medium, low) einen Serverpool mit entsprechender Kardinalität und Priorität definiert. Die Services der Datenbanken wurden von ihnen auf die passenden Serverpools verteilt. Eine Datenbank sollte jedoch per Kundenanforderung auf allen Servern im Cluster laufen, für diese Datenbank haben sie dementsprechend Services in allen Serverpools eingerichtet. *Abbildung 2* zeigt die neue Architektur. *Listing 4* verdeutlicht noch einmal das Anlegen der einzelnen Serverpools sowie die Zuordnung der Services zu den entsprechenden Serverpools.

Aus diesem Aufbau ergab sich nun das Problem, eine Datenbank über mehrere

```
crsctl add serverpool S1 -attr "IMPORTANCE=9,MIN_SIZE=1,MAX_SIZE=1"
crsctl add serverpool S2 -attr "IMPORTANCE=8,MIN_SIZE=1,MAX_SIZE=1"
crsctl add serverpool S3 -attr "IMPORTANCE=7,MIN_SIZE=1,MAX_SIZE=1"

crsctl add serverpool CP1   -attr \
   "IMPORTANCE=9,MIN_SIZE=1,MAX_SIZE=5,PARENT_POOLS=S1 S2 S3"
crsctl add serverpool CP2   -attr \
   "IMPORTANCE=9,MIN_SIZE=1,MAX_SIZE=2,PARENT_POOLS=S1 S2"
crsctl add serverpool CP3   -attr \
   "IMPORTANCE=8,MIN_SIZE=1,MAX_SIZE=2,PARENT_POOLS=S1"

srvctl add database -d DB1 -g CP1 -o $ORACLE_HOME
PRCR-1039 : Server pool ora.CP1 does not exist
```

Listing 1

```
crsctl add serverpool ora.CP1 -attr \
   "IMPORTANCE=9,MIN_SIZE=1,MAX_SIZE=5,PARENT_POOLS=S1 S2 S3"
crsctl add serverpool ora.CP2 -attr \
   "IMPORTANCE=9,MIN_SIZE=1,MAX_SIZE=2,PARENT_POOLS=S1 S2"
crsctl add serverpool ora.CP3 -attr \
   "IMPORTANCE=8,MIN_SIZE=1,MAX_SIZE=2,PARENT_POOLS=S1"

srvctl add database -d DB1 -g CP1 -o $ORACLE_HOME
PRKO-3150 : The server pool(s) CP1 specified with the -g cannot be used
to add an administrator-managed database
```

Listing 2

```
srvctl add serverpool -g S1  -i 10 -l 1 -u 1
srvctl add serverpool -g CP1 -i 10 -l 1 -u 5
srvctl add database -d DB1 -g CP1

srvctl status database -d DB1
Database is policy managed

crsctl modify serverpool ora.CP1 -attr „PARENT_POOL=ora.S1"

srvctl status database -d DB1
Database is administrator managed
```

Listing 3

Services transparent vom Client aus anzusprechen. Wie also muss ein Eintrag in der „tnsnames.ora" aussehen, um mehrere Services einer Datenbank in einem Alias zusammenzufassen? Dazu kam eine „DESCRIPTION_LIST" zum Einsatz *(siehe Listing 5)*.

Tests haben gezeigt, dass die Client-Verbindungen entsprechend über alle Services verteilt werden. Damit benutzen die Clients auch alle Server im Cluster. Mit dieser Lösung sind nun alle Anforderungen an das neue System erfüllt.

Erfahrungen aus dem Alltag

Im täglichen Betrieb dieser neuen Umgebung wurden bereits einige Erfahrungen gesammelt. Die Bereiche erstrecken sich von Backup über Monitoring bis hin zu eingesetzten Versionen und Patching. Für die Backups der Datenbanken im Cluster wird RMAN verwendet. Dieser verbindet sich über den Standard-Service zur jeweiligen Datenbank. Auf die Einrichtung eines separaten Service wurde zugunsten der Übersichtlichkeit verzichtet, da dies keinen nennenswerten Mehrwert bedeutet.

Das Monitoring des Systems erfolgt mit Cloud Control Release 3. Dieses ist ohnehin im Einsatz und scheint am besten geeignet, eine solche Umgebung effektiv zu überwachen und zu warten. Trotzdem gibt es einige Fallstricke bei der Einrichtung der Umgebung im Cloud Control. In älteren Releases wurden die Scan Listener im Cluster als „Down" gemeldet, wenn diese aus irgendeinem Grund auf einen anderen Server geschwenkt waren. Cloud Control versuchte dann nach wie vor, die Scan Listener auf dem ursprünglichen Server zu überwachen. Dazu existiert eine My Oracle Support Note 1493823.1; das zugrunde liegende Problem ist in Release 3 behoben. Trotzdem findet ein erneutes Discovery die vorhandenen Scan Listener als neue Listener. Dies kann und sollte man einfach ignorieren.

Ein größeres Problem stellt das Einrichten der Datenbanken dar. Cloud Control findet nicht immer alle Instanzen zu einer Datenbank, sondern teilt die Instanzen auf mehrere Datenbanken auf. Das Cloud Control prüft die Verfügbarkeit der Instanzen immer anhand des Servers, der zum Zeitpunkt des Discovery aktuell war. Werden Instanzen im Cluster umverteilt, hinzugefügt oder entfernt, weil Änderungen an der Kardinalität der Serverpools vorgenommen werden oder einzelne Server offline gehen müssen, so verliert Cloud Control praktisch völlig den Überblick und es bleibt nur ein Rediscovery als Weg, die Umgebung wieder sauber einzurichten. Das Problem ist bei Oracle als Bug in Bearbeitung.

Das Release- und Patch-Management erfordert ebenfalls einige Vorüberlegungen. Da die Version der Grid Infrastructure immer mindestens der Datenbank-Version entsprechen muss, haben sich die Autoren entschieden, immer die neueste Version der Grid Infrastructure einzuset-

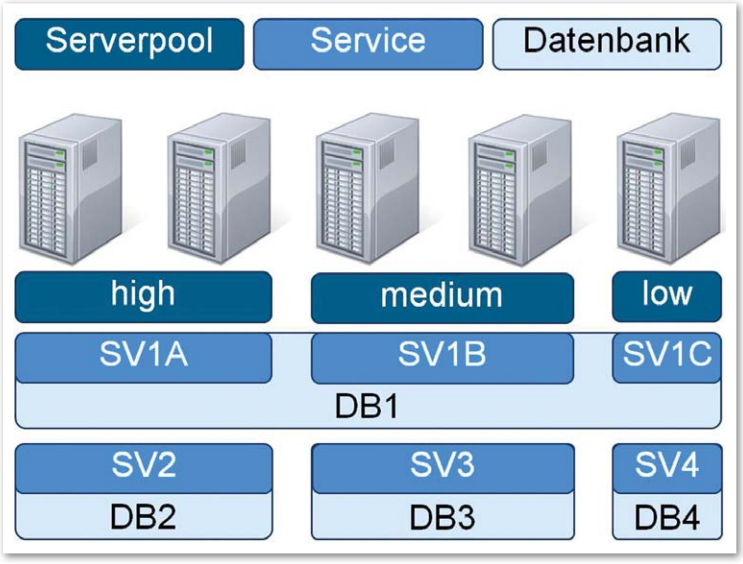


Abbildung 2: Clusteraufbau mit Serverpools und Services

zen. Die Upgrades der Grid Infrastructure können „rolling“ erfolgen. Das bedeutet, es entstehen keine Ausfallzeiten für die Applikationen. Datenbanken werden immer per „out of place“-Upgrade aktualisiert. Das wird nicht nur für Releases und Patchsets praktiziert, sondern auch für das Einspielen von Patches oder Patch Bundles. Man installiert in solchen Fällen immer ein neues Oracle-Home, das nach Bedarf gepatcht wird. Danach erfolgt die Umstellung der Datenbanken auf das neue Oracle Home. Dadurch verringert sich die nötige Downtime auf ein Minimum und es entstehen keinerlei Beeinträchtigungen für andere, nicht betroffene Datenbanken.

Fazit

Die neue Cloud-Lösung deckt alle im Vorfeld gesteckten Rahmenbedingungen vollständig ab. Es entstand eine standardisierte Umgebung für alle produktiven Datenbanken, die Verteilung der Instanzen erfolgt automatisch über die den Services zugeordneten Serverpools entsprechend den definierten SLAs und ermöglicht einen kontinuierlichen, unterbrechungsfreien Betrieb für alle Datenbanken. Zuvor war dies nur für eine Datenbank der Fall, die als RAC lief. Dabei war keine einzige zusätzliche Lizenz notwendig, da der Fünf-Knoten-Cluster bereits vorhanden war. Man konnte sogar Lizenzen einsparen, da eine ganze Reihe Datenbanken statt auf dedizierten Servern nun mit in diesem Cluster laufen.

Ausblick

Mit der hier vorgestellten Umgebung werden Features benutzt, die schon seit Längerem in 11*g* R2 etabliert sind und ausgereift erscheinen. Eine interessante Möglichkeit stellt die mit Database 12*c* neu eingeführte Multitenant-Option dar. Damit ließe sich eine noch bessere Flexibilität erzielen. Allerdings gibt es derzeit noch einige Einschränkungen beim Klonen von Datenbanken; die Quell-Datenbank ist während des Klonens für die Anwendung nicht verfügbar. Diese Tatsache, zusammen mit den zusätzlichen Lizenzkosten, ist der Grund, warum diese Option aktuell nicht verwendet wird. Die Autoren verfolgen die Entwicklung der Multitenant-Option aber mit Spannung.

```
srvctl add srvpool -g high   -i 10 -l 1 -u 2
srvctl add srvpool -g medium -i 8  -l 1 -u 2
srvctl add srvpool -g low    -i 6  -l 1 -u 1

srvctl add service -d DB1   -s SV1A    -g high   -c uniform
srvctl add service -d DB1   -s SV1B    -g medium -c uniform
srvctl add service -d DB1   -s SV1C    -g low    -c uniform

srvctl add service -d DB2   -s SV2     -g high   -c uniform
srvctl add service -d DB3   -s SV3     -g medium -c uniform
srvctl add service -d DB4   -s SV4     -g low    -c uniform
```

Listing 4

```
ALIAS=
(DESCRIPTION_LIST=
   (DESCRIPTION=
       (ADDRESS=(PROTOCOL=TCP)(HOST=cloud-scan)(PORT=1521))
       (CONNECT_DATA=
           (SERVICE_NAME= SV1A))
   )
   (DESCRIPTION=
       (ADDRESS=(PROTOCOL=TCP)(HOST=cloud-scan)(PORT=1521))
       (CONNECT_DATA=
           (SERVICE_NAME= SV1B))
   )
   (DESCRIPTION=
       (ADDRESS=(PROTOCOL=TCP)(HOST=cloud-scan)(PORT=1521))
       (CONNECT_DATA=
           (SERVICE_NAME= SV1C))
   )
)
```

Listing 5

Daneben bietet die Clusterware in der Version 12*c* die Möglichkeit, Serverpools dynamisch über Serverkategorien nach bestimmten Hardware-Attributen zusammenzusetzen. Es lassen sich außerdem verschiedene Konfigurationen der Serverpools mittels Policy verwalten und nach Bedarf aktivieren. Diese Features werden mittelfristig zum Einsatz kommen, um den Betrieb der Umgebung weiter zu vereinfachen und zu automatisieren.

Heiko Eitner
heiko.eitner@ldi.rlp.de

Marco Mischke
marco.mischke@robotron.de

Forms2ADF mal anders: Wie aus einer Oracle-Vision Praxis wird

Markus Klenke, TEAM GmbH; Marvin Grieger, s-lab Universität Paderborn; Wolf G. Beckmann, TEAM GmbH

Der von Oracle aufgezeigte Weg, Forms-Applikationen zu ADF zu migrieren, wurde schon in diversen White-Papers und Präsentationen erläutert. Dennoch wird in der Praxis mit einfachen „1:1"-Migrationstools und viel Handarbeit migriert. Wie ein modellgetriebener Migrationsansatz in der Praxis aussieht und welche Vorteile er gegenüber den konventionellen Migrationsstrategien bietet, wurde von TEAM in einem ZIM-geförderten Projekt in Zusammenarbeit mit der Universität Paderborn ermittelt. Das Ergebnis ist ein Werkzeug zur semi-automatisierten Migration, mit dem native ADF-Applikationen aus Forms-Anwendungen erstellt werden.

Beginnen wir mit einem kleinen Beispiel. Es besteht eine Suchmaske mit einigen Eingabefeldern und einem Anzeige-Button. Bei dessen Anklicken wird eine unter der Suchmaske befindliche Tabelle gefüllt. Die einfachste Art, diese Funktionalität zu realisieren, besteht darin, jedes Eingabefeld als Einschränkung („Bind Variable") für die Datenbank-Abfrage am Ausgabeblock zu verwenden.

Dieser Dialog soll nun migriert werden. Eine „1:1"-Migration würde vorsehen, für die Datenbank-Abfrage die benötigten Business-Service-Elemente zu erstellen, für die Eingabefelder äquivalente Input-Elemente zu erzeugen und über eine Aktion die Abfrage ausführen zu lassen. Dies ist ein valider Ansatz, doch muss man sich die Frage stellen: Ist das eine optimale Überführungsstrategie? Vielleicht, wenn ausschließlich der Technologiewechsel im Vordergrund steht. Wenn jedoch eine echte Modernisierung erfolgen soll, hätten wir unser Ziel verfehlt.

ADF bietet durch seine moderne Technologie insbesondere eine dynamisch generierte Suchkomponente, die sich für diese Funktionalität viel besser eignen würde, da sich zusätzlich zur Such-Funktionalität unter anderem vom Benutzer eingegebene Suchen speichern lassen. Somit haben wir einen viel benutzerorientierteren Dialog geschaffen, die Funktionalität im Kern aber beibehalten. Wir haben den Dialog modernisiert, nicht nur migriert.

Auch diesen Schritt könnte man automatisieren. Jedoch ist an dieser Stelle das explizite Wissen vonnöten, dass es sich um einen Suchdialog handelt, denn wir wollen schließlich nicht alle Dialoge mit Eingabefeldern auf Suchmasken abbilden. Ist damit die Idee einer Migration mit Modernisierung von Forms zu ADF gestorben? Die semiautomatisierte Migration beziehungsweise Modernisierung befasst sich im Kern mit genau dieser Fragestellung und liefert aufgrund der manuellen Interaktionsmöglichkeiten eine Antwort auf diese Problemstellung: Nein, es ergibt durchaus Sinn, eine Modernisierung von Forms mit ADF durchzuführen, sofern die Migrationsgründe es zulassen *(siehe Abbildung 1)*.

Warum werden in der Praxis trotz dieser Problemstellung „1:1"-Migrationen

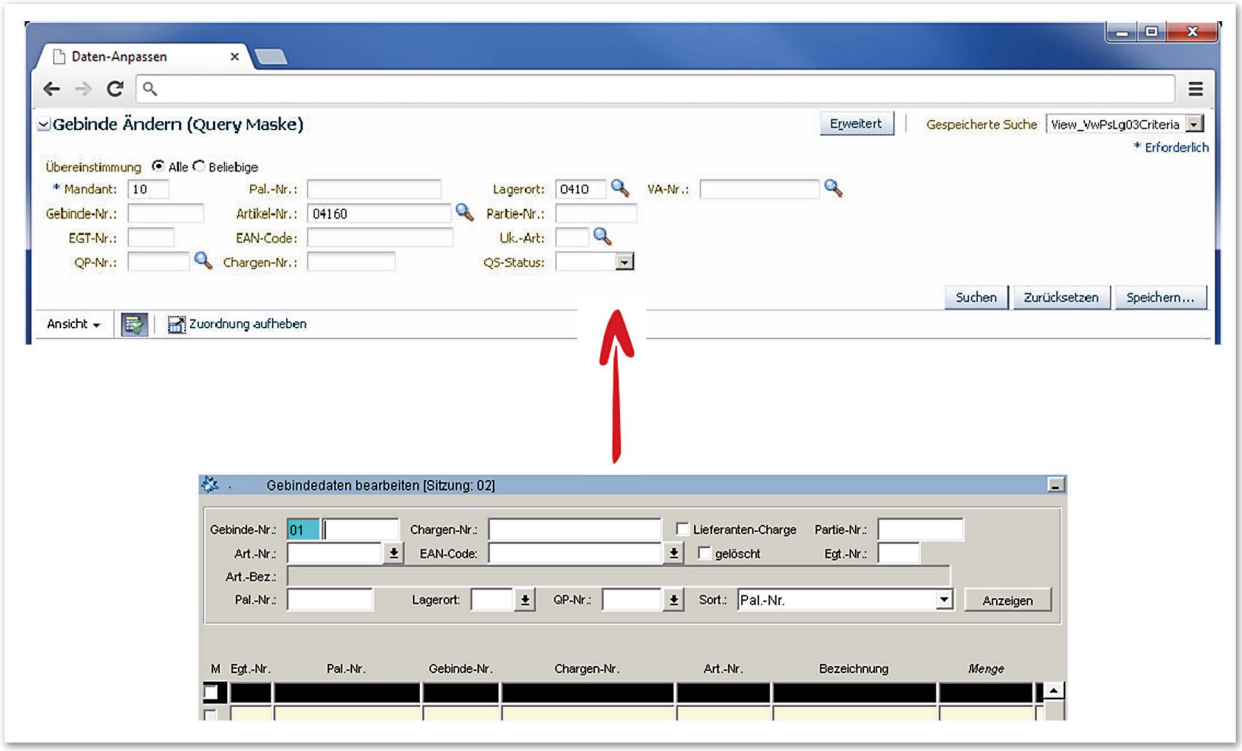


Abbildung 1: Einzelne Forms-Items zur Suche (unten) werden zur individualisierbaren ADF-Suchkomponente (oben)

durchgeführt? Weil Programme zur automatisierten „1:1"-Migration mit deutlich weniger Aufwand entwickelt werden können als Tools für die modellgetriebene Migration von Forms nach ADF. Bei einer „1:1"-Migration wird für jedes Element aus dem Quellsystem eine Abbildung auf ein Element in der neuen Umgebung definiert. Es scheint also, dass eine automatisierte Migration von Forms nach ADF relativ einfach realisierbar ist.

Warum tun sich so viele Projekte schwer mit einer Migration? Einen Effekt erkennt man schon an dem einfachen Beispiel aus der Einleitung: Syntaktisch gleiche Elemente sind auf funktional unterschiedliche Komponenten abzubilden. Für direkte Transformationen stellt dieses Unterfangen schon eine sehr komplexe Herausforderung dar. Sollen hingegen Konzepte überführt werden, die in der Quell-Plattform gänzlich anders realisiert sind (Navigation, Templates) oder gar nicht zur Verfügung gestanden haben (hierarchische Applikationsstruktur), so können diese Anforderungen schlicht und einfach nicht mit einer „1:1"-Migration durchgeführt werden, da eine Abbildungsvorschrift fehlt.

Diese fehlenden Vorschriften gehen aus dem starken architektonischen Unterschied zwischen Forms und ADF hervor und stellen daher einen – gerade bei großen Applikationen – nicht unerheblichen Teil der Migration dar. Die Qualität der Überführung dieser Konzepte steht in direkter Korrelation mit der Qualität der resultierenden ADF-Applikation. Aus diesem Grund werden in der Praxis entweder wenige Quelldateien automatisiert überführt oder es ist eine erhebliche manuelle Nachbearbeitung erforderlich.

Ein Ei gleicht nicht dem anderen

Wie erwähnt, sind Forms und ADF architektonisch grundverschieden. Während Forms monolithisch geprägt ist, also eine sehr enge Verzahnung zwischen Oberfläche und Datenbank anstrebt, ist die Reali-

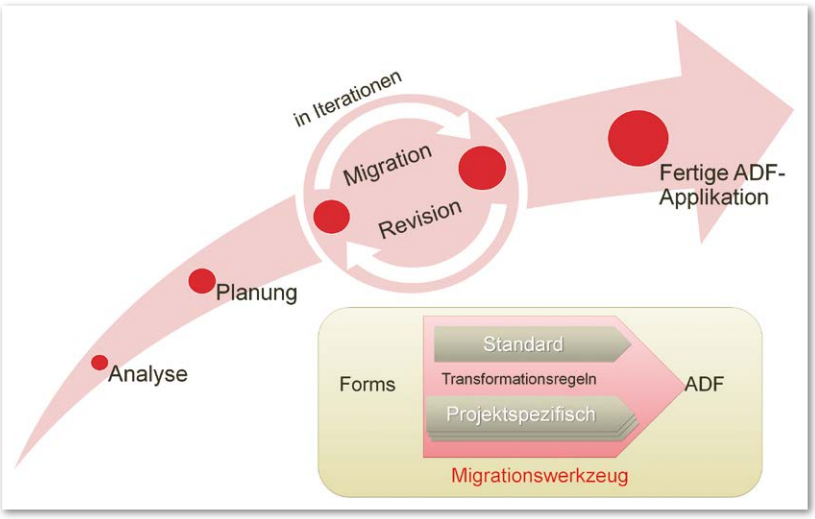


Abbildung 2: Migrationsprozess eines semi-automatisierten Ansatzes

sierung in ADF eine gegenläufige: Die Oberfläche einer Applikation soll nur zur Anzeige und Interaktionsmöglichkeit des Benutzers dienen, ist also in der Theorie strikt von jeglicher Logik zu trennen. Auch die Business-Service-Schichten der beiden Plattformen unterscheiden sich vehement voneinander.

Noch schwerwiegender fällt allerdings folgender Punkt ins Gewicht: Die Denkweise, in der auf beiden Systemen entwickelt wird, ist unterschiedlich. Während Forms-Applikationen dialoglastig und kompakt entwickelt werden, ist ADF prozessorientiert und jede Applikation kann als wiederverwendbare Komponente in einem höher liegenden Prozess gesehen werden.

Zudem bietet ADF zahlreiche Schnittstellen und Möglichkeiten zur Individualisierung und Erweiterbarkeit. Alle generierten Komponenten sollten das Potenzial der Wiederverwendung, die sie durch die Plattform mitbringen, auch ausnutzen. Schließlich ist das der enorme Vorteil von ADF gegenüber Forms; Daten-Anbindungen werden zentral als Basiseinheit gebildet, Labels von Texten werden in Ressourcen ausgelagert und Oberflächen sind benutzerspezifisch anpassbar.

Zusätzlich zur architektonischen Differenz der Plattformen kommen die unterschiedlichen Entwicklungsmuster, die sich in den vielen Jahren der Forms-Entwicklung in den Quelldateien niedergeschrieben haben. Kaum eine Forms-Applikation verhält sich wie eine äquivalente Applikation in einem anderen Projektumfeld. Daher ist es für die Migration immens wichtig, ein Verständnis für die Funktionalitäten und Fokussierungen der Altanwendung aufzubauen. Insbesondere ist es wichtig, die Applikation nicht einfach nach „Schema F" auf die neue Plattform zu überführen, sondern angepasste Transformationsregeln darüber aufzustellen, wie die Migration einzelner Komponenten durchgeführt werden soll. Dazu bietet es sich oft an, nicht nur ein Objekt einzeln zu behandeln, sondern mehrere Objekte zu einer konzeptionellen Komponente zusammenzufassen und gemeinsam zu untersuchen oder Objekte vom gleichen Typ als unterschiedliche Stereotype zu betrachten (etwa unterschiedliche Typen von Items). Kehren wir noch einmal zum Beispiel zurück, so erkennen wir dieses Schema selbst an den einfachsten Anwendungsfällen: Mehrere Eingabefelder und ein Aktionselement werden zu einer Suchkomponente zusammengeführt, da sie ihre Funktionalität nur im Zusammenspiel ausführen können.

Wie lautet also der Plan?

Der Schlüssel zum Erfolg bei einer erfolgreichen Software-Migration (automatisiert oder manuell) ist ein gut geplanter und durchgeführter Migrationsprozess. Eine Automatisierung ohne Verständnis ist genauso wenig erstrebenswert wie eine vollständige manuelle Re-Implementierung. Wie können nun also diese beiden Extrema miteinander verschmolzen und somit das Beste der beiden Wege extrahiert werden? Ein sehr effektiver Ansatz ist der Migrationsprozess, der in *Abbildung 2* beschrieben ist.

Das Migrationsprojekt beginnt mit einer Analyse-Phase. Dabei bewerten Migrationsexperten die Architektur und die Migrationskomplexität der Alt-Anwendung sowohl mithilfe von Analyse-Werkzeugen, mit den Entwicklern der Applikation, als auch manuell. Unter Beachtung der zusätzlich erfassten Projektziele und des Projektkontexts wird als Ergebnis eine vorläufige Migrationsstrategie für die Applikation erstellt.

In der anschließenden Planungs-Phase erfolgen tiefergehende Analysen der Alt-Anwendung nicht nur bezüglich der Architektur, sondern auch bezüglich der vorhandenen Entwicklungsmuster. Basierend auf diesen Erkenntnissen wird durch ADF-Experten eine detaillierte Ziel-Architektur entworfen und ein Migrationspfad, also Abbildungen zwischen Alt- und Ziel-Applikation, festgelegt.

Aufgrund des semi-automatisierten, modellgetriebenen Vorgehens kann die Quell-Applikation sehr stark umstrukturiert und trotzdem zu einem hohen Grad automatisiert migriert werden. Anschließend werden Migrationspakete gebildet, um eine iterative Überführung zu ermöglichen, und eine Umstellungsstrategie (wie Parallelbetrieb oder schrittweise Ablösung) festgelegt, um eine Unterbrechung des laufenden Betriebs der Applikation zu vermeiden. Das Ergebnis dieser Phase ist folglich eine angepasste und verfeinerte Migrationsstrategie. Parallel dazu werden die eigentliche Migration vorbereitet, also die Infrastruktur auf-

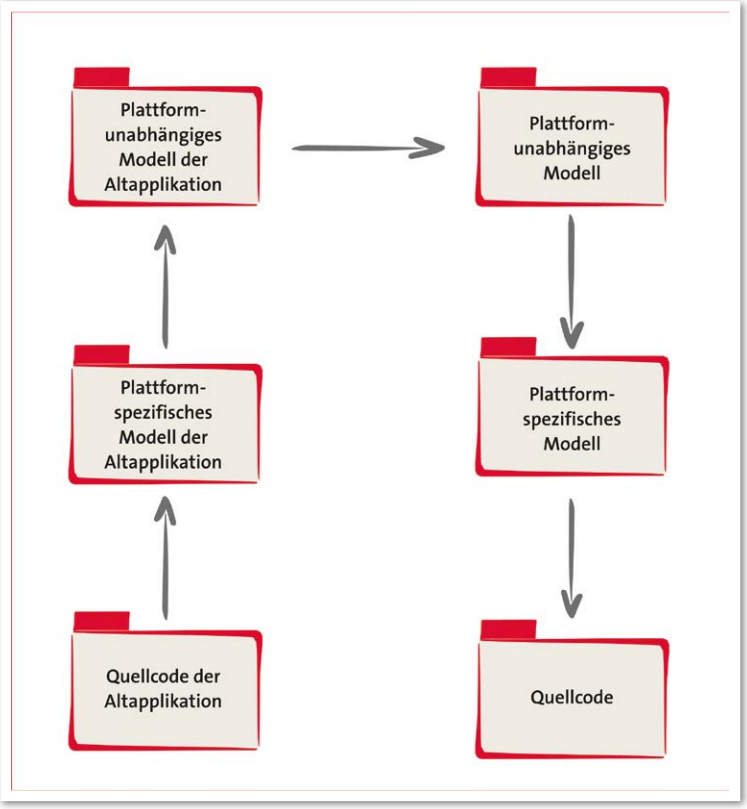


Abbildung 3:Technische Darstellung des Migrationsansatzes über Abstraktion

gebaut, und gegebenenfalls die ehemaligen Forms-Entwickler in ADF geschult.

Jetzt beginnt die eigentliche Migration in Iterationen. Diese läuft immer in vier Schritten ab:

1. Implementierung von projektspezifischen Regeln für die Erkennung und das Umsetzen von Mustern aus der Altanwendung nach ADF
2. Automatisches Migrieren eines der geplanten Module von Forms nach ADF
3. Manuelles Fertigstellen des Moduls
4. Revision der Fertigstellungsphase, um zu ermitteln, welche weiteren Muster zusätzlich automatisiert umgesetzt werden können. Diese werden dann im ersten Schritt der nächsten Iteration in das Migrationswerkzeug eingearbeitet.

Gerade bei der ersten Migrationsiteration werden noch viele weitere Muster gefunden, wodurch von Iteration zu Iteration immer weniger manuell fertiggestellt werden muss. Durch das iterative Vorgehen werden dabei sehr früh Teilergebnisse sichtbar. Diese Vorgehensweise hat noch einen Seiteneffekt: Da typischerweise die ehemaligen Forms-Entwickler die einzelnen Module fertigstellen, werden sie sanft an die neue Entwicklungsumgebung herangeführt.

Modellgetriebene, semi-automatisierte Migration in Softwareform

Aufgrund der oben genannten Unterschiedlichkeit von Applikationen ist es für eine Migration einer Software unerlässlich, Anpassungsmöglichkeiten und Erweiterungsmechanismen für das Migrationswerkzeug bereitzustellen. Die von TEAM entwickelte Implementierung eines modellgetriebenen, semi-automatisierten Migrationsmodells charakterisiert sich besonders durch den hohen Abstraktionslevel und die damit verbundenen Restrukturierungsmöglichkeiten *(siehe Abbildung 3)*.

Entgegen der bekannten „1:1"-Migrationsstrategien liegt der Fokus nicht auf der Transformation der einzelnen Elemente wie Eingabefelder oder Buttons, sondern auf dem Verständnis der Funktionalität der Alt-Applikation. Sind Strukturen der Applikation verstanden worden, die eine komplexere Funktionalität realisieren (beispielsweise das Suchfeld), werden diese extrahiert und durch ein Plattformunabhängiges Modell repräsentiert. Es dient zunächst als Zwischenrepräsentation der Alt-Applikation und wird im Laufe des Migrationszyklus durch Restrukturierungen und Funktionsanreicherungen zur gewünschten Ziel-Applikation transformiert. Diese wird nun mithilfe projektspezifischer Transformationsregeln zu einer ADF-Applikation generiert, die zum einen die Best-Practices von Oracle implementiert, zum anderen die während des Migrationsprojekts entstandenen Richtlinien einhält und Komponenten beinhaltet. Somit entsteht am Ende eines durchgeführten Migrationszyklus eine Applikation, die den projektspezifischen Migrationsanforderungen genügt.

Ein Blick hinter die Fassade

Im Folgenden sind die einzelnen Schritte aus *Abbildung 3* aus technischer Sicht beleuchtet. Als technologische Grundlage für die genutzten Modelle bietet sich mit EMF ein flexibles und erweiterbares Framework an, das den Anforderungen und dem notwendigen Individualismus eines Migrationsprojekts gewachsen ist. Um die Modelle zu persistieren und eine performante Schnittstelle zwischen Code und Modell herzustellen, werden die Modelle innerhalb eines CDO-Repository gespeichert, dessen Grundlage wiederum eine relationale Datenbank ist. Somit ist es möglich, mit bekannten SQL-Queries gezielt auf Modell-Elemente aus dem Repository zuzugreifen.

Zunächst werden alle notwendigen Quelldateien (FMB, PLL, MMB, OLB) sowie externe Komponenten (Datenbank, Forms-Plattform) in ein eigens erstelltes Modell geladen. Dieses wurde auf Basis

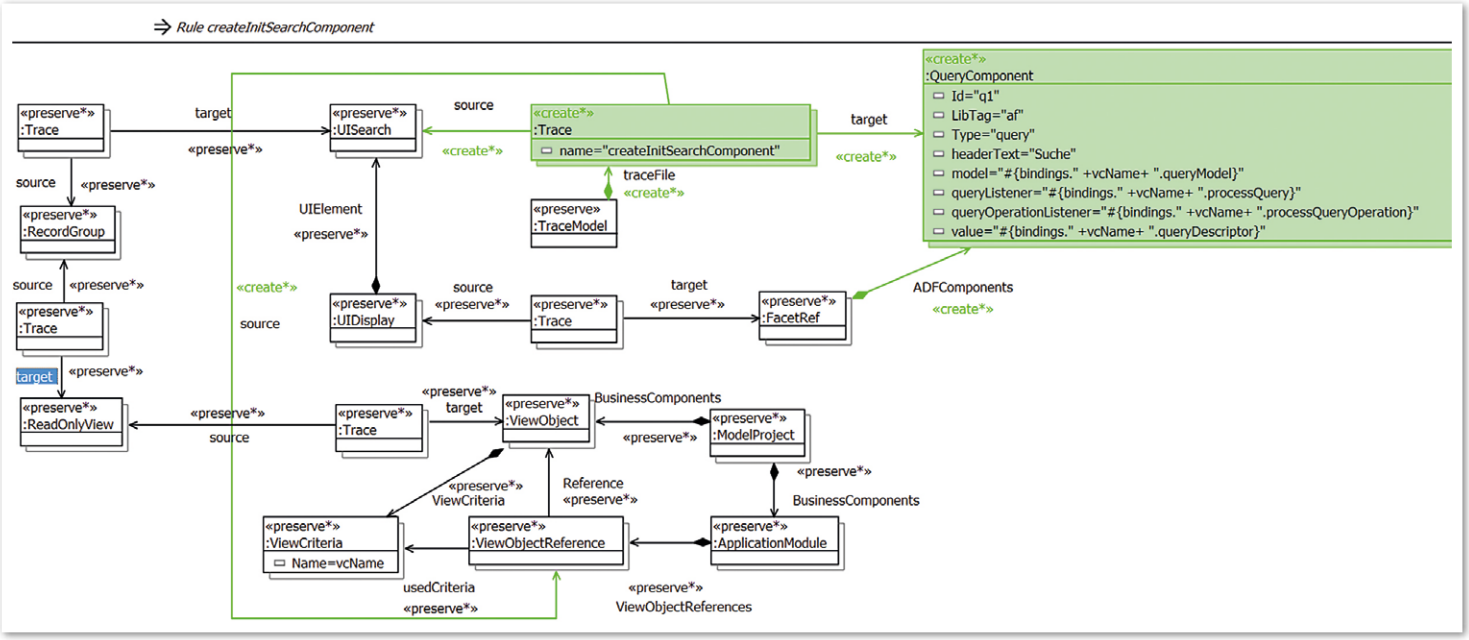


Abbildung 4: Transformationsregeln zur Erstellung einer Suchmaske

der „jdapi“-Schnittstelle des Forms Builder sowie aus den Definitionen der Sprache SQL sowie PL/SQL und des Oracle Data Dictionary erstellt. Basis des Plattform-unabhängigen Modells ist eine Erweiterung des OMG-Standards „Knowledge Discovery Metamodel“. Es erlaubt, eine Applikation in einem standardisierten Format Plattform-unabhängig zu repräsentieren. Da das Meta-Modell eine Beschreibungssprache für allgemeine Software-Applikationen ist, ist das Werkzeug flexibel genug, um auf Wünsche und Zielplattform-Einschränkungen beziehungsweise -Erweiterungen zu reagieren und diese in die Migration mit einfließen zu lassen.

Die Beschreibung der ADF-Modelle wird durch ein vollständig selbstentwickeltes ADF-Meta-Modell realisiert, das wiederum auf Best-Practices beruht. Der Fokus dieses Meta-Modells liegt auf Templates und wiederverwendbaren Komponenten, entspricht dabei allerdings immer den ADF-Standards. Somit können sich ADF-Neulinge sowie erfahrene ADF-Entwickler sofort in die neue Applikation einfinden.

Um aus dem ADF-Modell Quellcode für die ADF-Applikation zu generieren, kommt das Acceleo-Framework zum Einsatz. Es bietet eine einfache Möglichkeit, mithilfe von textuellen Templates (ähnlich wie bei Velocity und Texen) Quellcode zu generieren. Der Vorteil dieses Frameworks liegt allerdings in der direkten Implementierung einer EMF-Schnittstelle, durch die beim Erstellen der Templates eine direkte Einbindung des Modells möglich ist und daher bei fehlerhaften beziehungsweise nicht validen Eingaben ein zugehöriger Editor vor unerlaubten Aktionen warnt.

Um Modelle von einem Zustand in einen anderen zu transformieren, wird die Modell-Transformationssprache „Henshin“ eingesetzt. Diese ist im Kontext eines Eclipse-Incubation-Programms entstanden und bietet dem begleitenden Migrationsexperten die Möglichkeit, Transformationsregeln deklarativ und visuell zu erstellen. Die direkte Anbindung an die genutzten Meta-Modelle schränkt ihn sinnvoll ein, sodass valide Modelle immer auf valide Modelle abgebildet und nur durch das Migrationsprojekt erlaubte Elemente erstellt werden. Da die Transformationsregeln durch Henshin aus technischer Sicht ebenfalls Modelle sind, können diese zusätzlich in das Modell-Repository gespeichert werden. Somit wird kein Migrationsprojekt bei null gestartet, sondern kann bei Bedarf und Wunsch auf zuvor erstellten Transformationsregeln basieren *(siehe Abbildung 4)*.

Fazit

Was ist letztendlich der Vorteil, der durch eine semi-automatisierte Migration entsteht? Zunächst einmal scheint es durch die komplexen und zeitaufwändigen Vorbereitungs- und Verständnis-Schritte einen enormen Mehraufwand im Vergleich zu einer „1:1“-Migration zu geben. Dieser initiale Mehraufwand wandelt sich allerdings im Laufe des Migrationsprojekts in eine Beschleunigung der Arbeitspaket-Bearbeitung. Je präziser und detaillierter die Lösung für das erste Migrationspaket ist, desto größer wird der Gewinn für die folgenden Migrationspakete ausfallen. Darüber hinaus werden durch die zyklische Arbeitsweise die späteren Migrationspakte ebenfalls durch neue Features oder weitere Automatismen geschmälert, was letztendlich zu einer drastischen Kürzung des Migrationsaufwands führt.

Nicht nur das: Im Gegensatz zu einer vollautomatisierten „1:1“-Migration, die den Fokus auf die unmittelbare Ausführung des generierten Quellcodes richtet, wobei allerdings die Individualisierung auf der Strecke bleibt, erhält man zum Abschluss eines semi-automatisierten Migrationsprojekts genau die Ausprägungen an ADF-Software, die für die Ausführung der Funktionalitäten der Alt-Applikation innerhalb der neuen Umgebung sinnvoll sind; Templates, Basis-Komponenten und Applikationsrahmen inklusive.

Durch die vielen manuellen Interaktionsmöglichkeiten innerhalb des Migrationszyklus lässt sich der Weg zu einer Applikation, die man als ADF-Entwickler erwartet, sehr gut ebnen. Jeder Entwickler, der sich in der Zielumgebung auskennt, wird für die Entscheidung eines semi-automatisierten Prozesses dankbar sein. Auch der Einstieg für Entwickler des Quellsystems ist durch die Konzeption nahe an einer Referenz-Architektur einfacher.

Durch die diversen manuellen Interaktionsmöglichkeiten kann bereits in den ersten Phasen der Migration an wiederverwendbaren Komponenten, möglichen Benutzerprozessen oder Code-Libraries gearbeitet werden, die später in der Generation nutzbar sind. Genau diese Features sorgen dafür, dass am Ende einer semi-automatisierten Migration wirklich eine Applikation erzeugt wird, die von einer manuell erstellten ADF-Applikation kaum mehr zu unterscheiden ist. Das sollte schließlich das Ziel eines jeden Automatismus sein.

Wolf G. Beckmann
wb@team-pb.de

Markus Klenke
mke@team-pb.de

Marvin Grieger
mgrieger@s-lab.upb.de

Edition Based Redefinition: Versionsverwaltung für Datenbankobjekte

Daniel Horwedel, merlin.zwo InfoDesign GmbH & Co. KG

Entwickler haben häufig die Anforderung, dass die Aktualisierung von Anwendungen ohne große Wartungsfenster erfolgen soll. Darüber hinaus kann es sinnvoll sein, neue Versionen von Datenbankobjekten innerhalb eines bestehenden Schemas zu testen und die Auswirkungen auf abhängige Objekte zu analysieren.

Oracle bietet zur Lösung dieser Problematik auf der Datenbank-Ebene ein einfach nutzbares Werkzeug an: Edition Based Redefinition (EBR). Damit lassen sich mehrere Versionen eines Datenbankobjekts im Schema installieren und Session-abhängig verwenden, wodurch ein Update auf eine neue Version der Datenbankobjekte oder ein Test der neuen Version ohne Beeinträchtigung des laufenden Betriebs erfolgen kann. Den Anwendern steht in ihrer Session die alte Version der Datenbankobjekte zur Verfügung, während die neue installiert und getestet werden kann. Mit einem einfachen „ALTER SESSION"-Befehl kann anschließend auf die neue Version umgestellt werden.

Szenario

Die Aktualisierung bestehender Datenbank-Anwendungen im laufenden Betrieb ist mit einigen Gefahren verbunden. Zunächst einmal muss eine Downtime eingeplant werden, um die aktualisierten Datenbank-Objekte zu installieren und zu kompilieren.

Je nach Art der Anwendung ist eine Unterbrechung der Verfügbarkeit nicht möglich, sodass bislang aufwändige Lösungen benötigt wurden, um ein Anwendungs-Upgrade ohne Unterbrechung des laufenden Betriebs durchführen zu können. Zudem kann auch nach intensivem Testen der Anwendung nicht in jedem Falle ausgeschlossen werden, dass auf dem Produktiv-System unvorhergesehene Probleme, Seiteneffekte oder datenbezogene Fehler auftreten.

Eine gängige Möglichkeit zum Testen umfangreicher Änderungen stellt die Verwendung einer Schema-Kopie dar, die auf Basis von Produktivdaten die Auswirkungen einer Änderung der Datenstruktur ermittelt. Dabei entsteht allerdings durch das Vorhalten mehrerer Testumgebungen ein gewaltiger Aufwand, wodurch im Endeffekt oftmals veraltete Testdaten zur Verfügung stehen oder die Testschemata unterschiedliche Versionsstände bei den enthaltenen Datenbankobjekten aufweisen, wodurch kein verlässlicher finaler Abnahmetest möglich ist.

Einen effizienten Lösungsweg für diese Problematik bietet das Online Application Upgrade, dessen Zielsetzung die Durchführung einer Aktualisierung von Datenbankobjekten möglichst ohne Downtime ist. Sein Grundprinzip zeichnet sich dadurch aus, dass der Benutzer zunächst auf der bestehenden Programmversion weiterarbeiten kann, bis die (parallel zu installierende) neue Version fertig installiert, getestet und freigegeben ist. Sobald dies geschehen ist, werden die Benutzer quasi per Knopfdruck auf die neue Anwendungsversion umgestellt.

Oracle bietet zur Durchführung dieser Online Application Upgrades seit der Version 11*g* R2 über Edition Based Redefinition ein umfangreiches Werkzeug an, das mit Erscheinen der Version 12*c* erheblich erweitert und verbessert wurde. Es lässt sich mit allen Editionen der Datenbank kostenfrei nutzen, sodass diese Funktionalität nicht nur den Anwendern der Enterprise Edition vorbehalten, sondern auch schon ab der Express Edition einsetzbar ist.

Versionierung der Datenbankobjekte als Lösungsweg

Zur Reduzierung des Aufwands beim Betrieb mehrerer Test- und Entwicklungsumgebungen sowie der Durchführung von Online Application Upgrades, also der Bereitstellung neuer Anwendungsversionen ohne Downtime, bietet sich eine „Versionierung" der Datenbankobjekte mittels Edition Based Redefinition an. Dabei werden mehrere Versionen eines Datenbankobjekts in Kombination mit der Information, zu welcher Edition das jeweilige Objekt gehört, innerhalb eines Schemas vorgehalten. Zwischen den einzelnen Editionen kann nun innerhalb des Session-Kontextes gewechselt werden. Dies reduziert den Aufwand für die Pflege und Bereitstellung von Testumgebungen deutlich, da für die Durchführung von Tests mit Produktivdaten nicht mehr zwingend eine separate, auf dem aktuellsten Datenstand gehaltene Testumgebung bereitgestellt werden muss.

Den größten Nutzen bietet die Editionierung der Datenbankobjekte im Bereich des Online Application Upgrades. Bei der Aktualisierung der Anwendung können die Benutzer weiterhin in ihrer bisherigen Edition weiterarbeiten, bis die aktualisierten Objekte in der neuen Edition fertig installiert und freigegeben sind. Anschließend wird die neue Edition als Default-Edition gesetzt, sodass die Nutzer ab der nächsten Anmeldung standardmäßig

mit den Objekten dieser Edition arbeiten. Alternativ kann auch innerhalb einer aktiven Session die zu verwendende Edition manuell gesetzt werden. Edition Based Redefinition unterstützt folgende Objekttypen sowohl in der Datenbank-Version 11*g* R2 als auch in der Version 12*c* R1:

- Views
- Functions/Procedures/Packages
- Trigger
- Types
- Libraries
- Private Synonyme

In der Version 12*c* ist die Edition-Based-Redefinition-Funktionalität stark erweitert.

Grenzen der Edition Based Redefinition

Leider wird der durch Edition Based Redefinition (EBR) gebotene Komfort durch einige Einschränkungen reduziert:

- *Public Synonyms*
  Im Gegensatz zu privaten Synonymen ist eine Editionierung von Public Synonyms nicht möglich, da nicht jede Edition in jedem Schema verwendet werden muss. Bei einem Public Synonym handelt es sich allerdings um ein Objekt, das Schema-übergreifend existiert, wodurch für die Datenbank nicht feststellbar ist, welcher Edition es zugeordnet werden soll. Innerhalb eines editionierten Schemas sind Public Synonyms allerdings möglich, solange diese nicht auf ein editioniertes Objekt verweisen.
- *Benutzerdefinierte Datentypen*
  Die Verwendung benutzerdefinierter Datentypen in Tabellen, durch die ein Verweis auf ein editioniertes Objekt erfolgt, ist ebenfalls nicht möglich, ebenso darf ein nicht-editioniertes Subprogram keine statische Referenz auf ein Subprogram haben, dessen Eigentümer-Schema editioniert ist.
- *Function Based Indizes*
  Eine Editionierung von Function Based Indizes, die auf einem editionierten Objekt basieren, ist nicht möglich.
- *Materialized Views (11g R2)*
  In der Version 11*g* R2 können keine Materialized Views erstellt werden, die auf editionierten Objekten basieren, da für die Materialized View nicht ersichtlich ist, welche Edition des referenzierten Objekts verwendet werden soll. In Version 12*c* entfällt diese Einschränkung, da hier nun angegeben werden kann, welche Edition des referenzierten Objekts verwendet werden soll.
- *Virtual Column (11g R2)*
  Die Verwendung von virtuellen Spalten, die auf editionierte Objekte verweisen, ist in der Version 11*g* R2 aufgrund derselben Problematik wie bei Materialized Views nicht möglich. Mit der Version 12*c* wurde hier ebenfalls eine Möglichkeit eingeführt, die zu verwendende Edition explizit anzugeben.
- *Statische Referenz auf editioniertes Subprogram*
  Eine statische Referenz auf ein editioniertes Subprogram durch ein nicht-editioniertes Subprogram ist ebenfalls nicht möglich, da nicht festgelegt werden kann, auf welche Edition die Referenz erfolgen soll.
- *Tabellen und darin enthaltene Daten*
  Ein großes Manko beim Einsatz von Edition Based Redefinition ist, dass die Editionierung von Tabellen nicht ohne Weiteres beziehungsweise ohne manuellen Eingriff möglich ist. Zur Versionierung von Tabellen und der darin enthaltenen Daten existieren zwei unterschiedliche Konzepte, die im Abschnitt „Editionierung von Tabellen" näher beschrieben sind.

Die Aktivierung

Zunächst wird überprüft, ob EBR für das betreffende Schema bereits aktiviert wurde *(siehe Listing 1)*. Für die Erzeugung einer neuen Edition ist das „CREATE ANY EDITION"-Recht erforderlich, für das Löschen bestehender Editionen das „DROP ANY EDITION"-Recht. Diese Rechte werden dem Schema durch „GRANT CREATE ANY EDITION, DROP ANY EDITION TO meinschema;" zugewiesen.

```
SELECT editions_enabled
  FROM dba_users
 WHERE username = 'MEINSCHEMA';
```

Listing 1

```
SELECT u.name Schema,
       o1.name Objektname
  FROM obj$ o1,
       obj$ o2,
       dependency$ dep,
       user$ u
 WHERE o1.obj# = dep.d_obj#
   AND o2.obj# = dep.p_obj#
   AND o1.remoteowner is null
   AND o2.owner# = (   SELECT user_id
                        FROM sys.dba_users
                       WHERE username = 'MEINSCHEMA'
                   )
   AND o1.owner# = u.user#
   AND o2.type# in (4,5,7,8,9,10,11,12,13,14,22,87)
   AND (
          (      u.type# <> 2
             AND bitand(u.spare1, 16) = 0
             AND u.user# <> o2.owner#
          )
        OR
          (
             O1.type# NOT IN (4,5,7,8,9,10,11,12,13,14,22,87)
          )
       )
;
```

Listing 2

Bevor nun für das jeweilige Schema die Editionierung aktiviert wird, sollte zunächst überprüft werden, ob in diesem Schema Objekte vorhanden sind, die selbst nicht editionierbar sind, aber gleichzeitig von editionierbaren Objekten abhängen, wodurch eine Aktivierung der Editionierung zu Problemen führen kann. Mithilfe des als SYS-Benutzer auszuführenden SELECT-Statements kann (für die Version 11*g* R2) überprüft werden, ob solche Objekte im Schema vorhanden sind *(siehe Listing 2)*.

Sollte dieser „SELECT" keine Daten zurückliefern, stellt die Aktivierung von EBR kein Problem dar. In der Version 11*g* sind oftmals Materialized Views, die auf editionierbare Views zugreifen, ein Problem, da Materialized Views in dieser Datenbankversion nicht editioniert werden können. In der Version 12*c* muss hierbei lediglich angegeben werden, welche Edition der zugrunde liegenden Objekte verwendet werden soll. Anschließend wird EBR mittels „ALTER USER meinschema ENABLE EDITIONS;" für das angegebene Schema aktiviert.

Erzeugen und Löschen von Editionen

Nach der Aktivierung der Editionierung ist standardmäßig die sogenannte Root-Edition „ora$base" vorhanden. Auf dieser basieren alle nachfolgend angelegten Editionen. Sie kann daher nicht gelöscht werden. Sobald eine neue Edition erstellt wird, wird diese durch die Datenbank als Child-Edition unterhalb der Root-Edition angelegt.

Eine neue Edition wird immer als Schema-unabhängiges Objekt angelegt, zur Erstellung dient der SQL-Befehl „CREATE EDITION edition1 [AS CHILD OF ora$base];". Eine neue Edition erbt zum Zeitpunkt ihrer Erstellung immer die editionierten Objekte ihrer Parent-Edition – diese werden dazu in die neue Edition hineinkopiert.

Zum Löschen einer Edition – mitsamt ihrer editionierten Objekte – genügt ein simpler DROP-Befehl: „DROP EDITION edition1;". Objekte, die nicht editioniert sind, werden mit diesem Befehl nicht gelöscht.

Wechsel zwischen den Editionen

Bei einem Wechsel der Edition innerhalb der Session wird zunächst die aktuell in der Session aktivierte Edition durch die Funktion „SYS_CONTEXT" ermittelt: „SELECT sys_context('userenv', 'session_edition_name') FROM dual;". Analog dazu ist die aktuell gültige bzw. neueste Edition festzustellen, dies geschieht über den Systemkontext „current_edition_name".

Die Auswahl der zu nutzenden Edition erfolgt auf Session-Ebene und kann dementsprechend komfortabel per „ALTER SESSION"-Statement stattfinden: „ALTER SESSION SET EDITION = edition1;". Das Recht zum Zugriff auf die einzelnen Editionen muss dem Nutzer allerdings im Voraus mittels „GRANT USE ON EDITION edition1 TO meinschema;" zugewiesen werden. Soll die Verwendung einer Edition für alle Nutzer möglich sein, wird durch das Grant-Statement dieses Recht einfach dem Schema PUBLIC zugeordnet. Die standardmäßig zu verwendende Edition wird durch „ALTER DATABASE DEFAULT

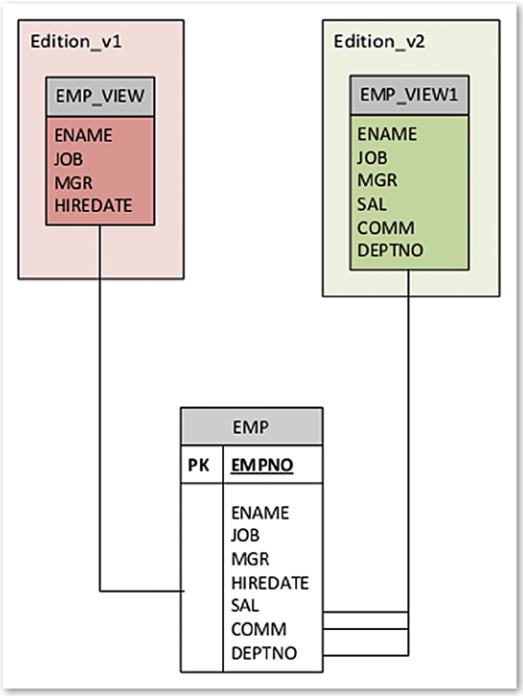


Abbildung 1: Tabellen-Redefinition mittels „Editioning Views"

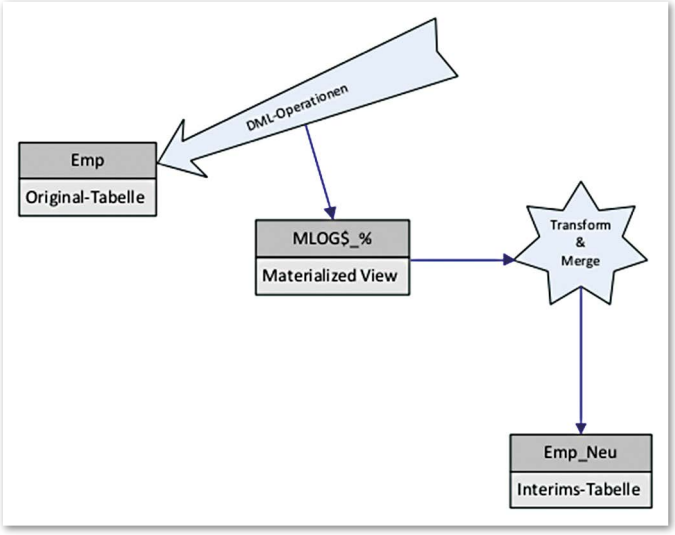


Abbildung 2: Tabellen-Migration mit „DBMS_REDEFINITION"

EDITION = edition1" datenbankweit festgelegt.

Das Erzeugen, Bearbeiten und Löschen von Datenbank-Objekten erfolgt wie gewohnt, es ist lediglich zu beachten, dass in der Datenbanksession die jeweils gewünschte Edition aktiviert ist – es werden ausschließlich die in der für die jeweilige Session aktiven Edition verfügbaren Objekte bearbeitet, erzeugt oder gelöscht.

Die Editionierung erfolgt vollständig transparent, sodass zum Zeitpunkt der Installation der Datenbankobjekte (bis auf die Auswahl der Edition innerhalb der Session) keinerlei Besonderheiten beachtet werden müssen.

Editionierung von Tabellen

Mittels EBR lassen sich Tabellen leider nicht in Editionen verwalten, da eine automatisierte Veränderung und Anpassung der Daten konzeptionell nicht gewünscht ist. Zur Lösung dieser Problematik stehen zwei verschiedene Lösungswege zur Verfügung. Mithilfe von „Editioning Views" kann eine Redefinition der Tabellen in allen Editionen der Oracle-Datenbank durchgeführt werden, der Aufwand hierfür ist unter Umständen aber etwas höher als bei der Synchronisierung mittels „DBMS_REDEFINITION", die leider nur in der Enterprise Edition zur Verfügung steht.

Bei der Tabellen-Redefinition per „Editioning Views" wird der Zugriff auf die Tabellen mit einem View-Layer abstrahiert, durch den die jeweilige „Edition" der Tabelle abgebildet wird *(siehe Abbildung 1)*. Alle Editionen der View greifen auf dieselbe Tabelle zu, stellen der Anwendung aber nur die für die jeweilige Edition benötigten Daten zur Verfügung. Wird nun der Tabelle eine neue Spalte hinzugefügt, so wird diese an die zugrunde liegende Tabelle angehängt und in der View der Edition, mit der die Spalte zur Verfügung stehen soll, ebenfalls hinzugefügt. Beim Entfernen von Spalten werden diese nicht aus der Tabelle entfernt, sondern lediglich in der View der entsprechenden Edition nicht mehr mit ausgegeben.

Durch diese Lösung lassen sich Änderungen am Datenmodell relativ leicht abbilden, ohne die zugrunde liegenden Daten ändern zu müssen, allerdings lässt sich nicht jeder Fall einwandfrei abbilden.

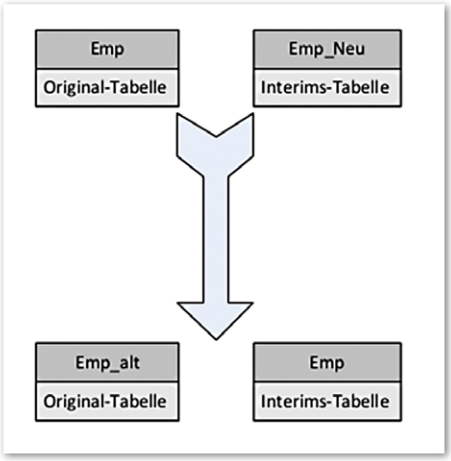


Abbildung 3: Ersetzen der Ursprungstabelle durch die neue Tabellenversion

Probleme können zum Beispiel beim Ändern von Datentypen auftreten, außerdem können die zugrunde liegenden Tabellen schnell sehr viele nicht mehr benötigte Spalten enthalten.

Bei der Tabellen-Redefinition mit „DBMS_REDEFINITION" werden nicht mehrere Editionen parallel vorgehalten, sondern nur die Migration von Tabellenobjekten in eine neue Version unterstützt. Vereinfacht formuliert handelt es sich hierbei um die Erzeugung eines Klons der zu verändernden Tabellen, der nun bearbeitet werden kann, bei Abschluss der Redefinition die Daten der ursprünglichen Tabelle enthält und diese Tabelle ersetzt *(siehe Abbildung 2)*.

Vor der Tabellen-Redefinition muss zunächst überprüft werden, ob eine Redefinition grundsätzlich möglich ist. Hierzu stellt Oracle eine Überprüfungsfunktion zur Verfügung: „dbms_redefinition.can_redef_table('meinschema', 'meine_original_tabelle')". Anschließend wird eine Tabelle mit der neuen Struktur erzeugt, die später die ursprüngliche Tabelle ersetzen wird. Hierfür wird ein normales „CREATE TABLE"-Statement ohne jegliche Besonderheiten verwendet. Es ist nur zu beachten, dass genügend Speicherplatz zur Erzeugung einer Kopie der Ursprungstabelle zur Verfügung steht.

Mittels „dbms_redefinition" werden nun die Redefinition gestartet und die Inhalte der Ursprungstabelle in die neue Tabelle kopiert: „dbms_redefinition.start_redef_table('meinschema', 'meine_original_tabelle', 'meine_neue_tabelle')". Seit der Datenbank-Version 10*g* werden die den Tabellen zugehörigen Objekte wie Constraints, Trigger sowie Indizes automatisch mithilfe der Prozedur „COPY_TABLE_DEPENDENTS" mitkopiert und bei Bedarf auch kompiliert bzw. aktiviert.

Aus Performance-Gründen sollte vor dem Abschluss der Redefinition noch eine Synchronisation der Tabellen durch die Prozedur „dbms_redefinition.SYNC_INTERIM_TABLE" erfolgen. Mittels „FINISH_REDEF_TABLE" wird anschließend die Redefinition abgeschlossen und die neue Tabelle ersetzt die Ursprungstabelle *(siehe Abbildung 3)*.

Fazit

Mit Edition Based Redefinition stellt Oracle eine komfortable Möglichkeit zur transparenten Versionierung von Datenbankobjekten zur Verfügung. Allerdings lässt sich diese Technik nicht durchgängig für alle Objekttypen anwenden, wodurch sich der Einsatzbereich im Alltag in der Regel auf die Editionierung von Stored Procedures und Views beschränken wird. Mit einigen Abstrichen ist auch die Migration von Tabellen auf eine neue DDL-Version möglich – allerdings kann hierbei nicht ohne Weiteres innerhalb der Session zwischen verschiedenen Versionen gewechselt werden.

Insbesondere für den Test neuer Versionen von Stored Procedures und deren Auswirkungen auf andere Datenbankobjekte sowie das Online Application Upgrade von Stored Procedures stellt die Verwendung von Edition Based Redefinition ein praktisches Feature dar, das sich insbesondere in der Datenbank-Version 12*c* für die regelmäßige Verwendung eignet.

Daniel Horwedel
daniel.horwedel@merlin-zwo.de

Intelligentes Monitoring: Implementierung von Overflow- und Runtime-Prognosen

Dr. Thomas Petrik, Sphinx IT Consulting

Ziel eines intelligenten Monitorings ist es, einige wenige KPIs zu definieren, die einerseits leicht zu interpretieren sind und andererseits die für den Betrieb einer Datenbank wesentlichen Fragen beantworten: Ist die Instanz respektive die Datenbank verfügbar? Ist eine stabile Response-Zeit gewährleistet? Welche Vorsorge ist zu treffen, um Storage-Engpässe zu vermeiden?

Der Schlüssel zum intelligenten Monitoring liegt zum einen in der Betrachtung zeitlicher Entwicklungen und zum anderen in der statistisch korrekten Interpretation der gesammelten Daten. Die weit verbreitete alleinige Verwendung statischer Metriken im Monitoring wie Füllgrade von Tablespaces oder die Auslastung der Flash-Recovery-Area ist definitiv als unzureichend zu betrachten und wird durch zeitabhängige Methoden nahezu vollständig ersetzt.

Statistische Grundlagen für Overflow-Prognosen

Hat man sich erst einmal eine historische Datensammlung angelegt (wie den Platzverbrauch pro Tablespace, pro Filesystem, pro Maschine oder auch für ein ganzes Storage-Tier im zentralen Storage-Pool), so lassen sich diese Messpunkte in einem Koordinatensystem auf der y-Achse gegen die Zeit auftragen. Die Optik eines derartigen Diagramms suggeriert meist bereits einen funktionalen Zusammenhang zwischen Messwert und Zeit, den es schließlich auch zu finden gilt. Im Idealfall hat man es mit annähernd linearem Verhalten zu tun, aber auch andere Abhängigkeiten kommen vor und müssen ebenso behandelt werden. Beim Betrachten von *Abbildung 1* ergeben sich zwei mögliche Fragestellungen:

1. Wann wird ein vorgegebener Maximalwert erreicht (oder anders gefragt: Wann geht der Tablespace über)?
2. Welcher y-Wert wird zu einem bestimmten Zeitpunkt erreicht sein (oder: Wie hoch ist der Platzbedarf in einigen Monaten)?

Während die erste Form der Fragestellung klassischerweise die eines DBAs ist, der einen Overflow verhindern muss, betrifft der Fokus der zweiten Fragestellung die Planung.

Um zu diesen Aussagen zu gelangen, wendet man die Methoden der Regression und Extrapolation an und beurteilt den gefundenen funktionalen Zusammenhang durch Korrelations-Kennzahlen. Wie spätere Betrachtungen zeigen werden, kommt man mit dem Spezialfall der linearen Regression aus – selbst dann, wenn ein System kein durchgängig lineares Verhalten zeigt.

Lineare Regression

Um also eine geeignete Regressionsfunktion zu finden, betrachten wir die Summe

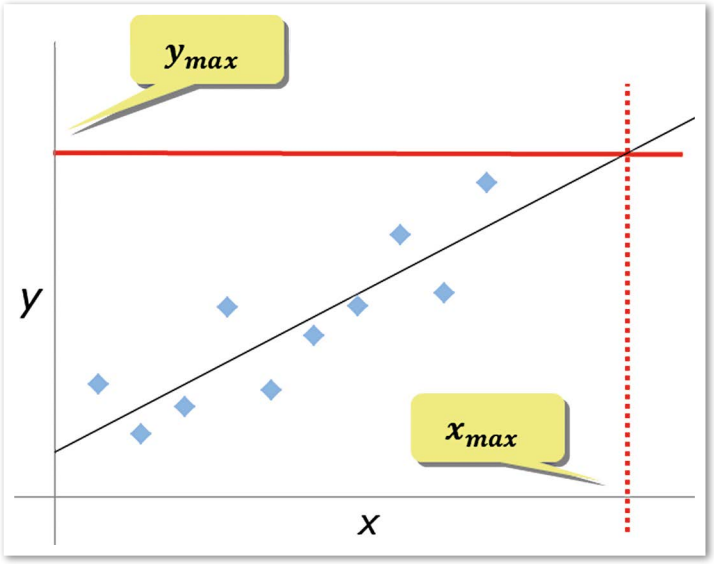


Abbildung 1: Beispiel für annähernd lineare Entwicklung

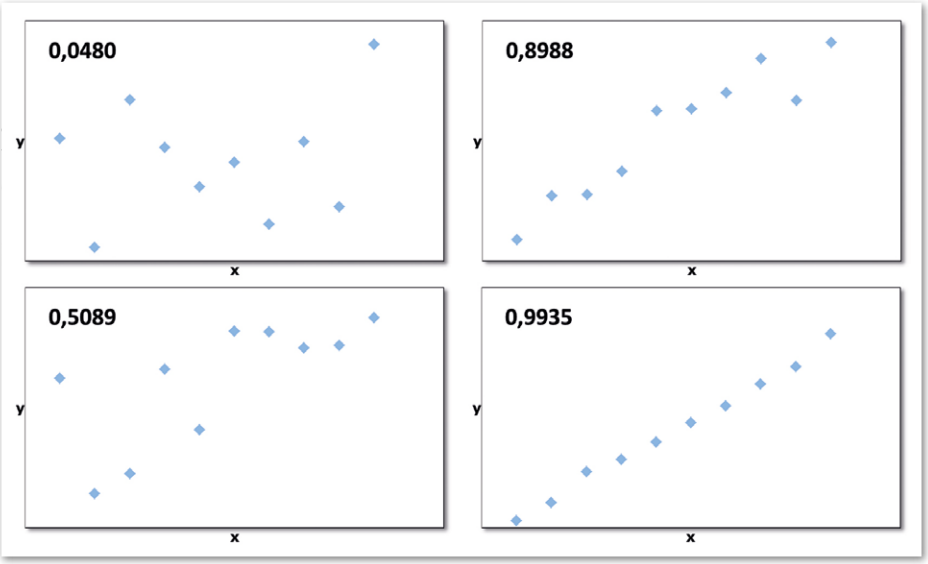


Abbildung 2: r^2 für verschiedene Verteilungen

Diese Aufgabe ist durch einfache Rechnung leicht lösbar (interessierte Leser mögen die genaue Ableitung der Standardliteratur entnehmen) und führt zu den Formeln für Ordinatenabschnitt (Intercept) und Steigung (Slope). Bemerkenswert an diesen Formeln ist die Tatsache, dass die ganze Berechnung lediglich mit Mittelwertbildungen auskommt. Im Grunde ist also die AVG-Funktion, die in allen SQL-Dialekten anzutreffen ist, bereits ausreichend, um derartige Berechnungen via SQL durchzuführen.

Oracle (und mittlerweile ebenso die meisten anderen Datenbanken) stellt für die lineare Regression ein eigenes Set an Funktionen mit dem Präfix „REGR_" zur Verfügung. „REGR_SLOPE(y,x)" berechnet die Steigung, „REGR_INTERCEPT(y,x)" den Ordinatenabschnitt (die erste Expression ist stets der abhängige Wert, im vorliegenden Beispiel also y).

Verteilung und Korrelation

Für eine automatisierte Beurteilung, ob die Grundannahme des linearen Verhaltens gerechtfertigt ist, empfiehlt sich die Berechnung des Korrelations-Koeffizienten r beziehungsweise von r^2, dem sogenannten Bestimmtheitsmaß.

Der Korrelationskoeffizient lässt sich somit auf die Varianzen der x- und y-Werte s_x^2 und s_y^2 beziehungsweise auf die Kovarianz s_{xy} zurückführen. Die dafür vorgesehenen SQL-Funktionen sind „VAR_POP" und „COVAR_POP", aber vor allem „CORR(x,y)" beziehungsweise „REGR_R2(x,y)" für die direkte Berechnung von r^2.

r^2 kann Werte zwischen 0 und 1 annehmen, wobei bei einem Wert von 1 eine perfekte Gerade vorliegen würde und bei einem Wert von 0 hingegen keinerlei Korrelation bemerkbar wäre. Für die Werte zwischen 0 und 1 gilt, dass die Korrelation besser ist, je näher der Wert bei 1 liegt. Empirische Untersuchungen im Bereich der Storage Forecasts haben gezeigt, dass man ab einem Wert von 0,8 berechtigterweise von einem linearen Zusammenhang ausgehen kann; Regressionen mit kleineren Korrelationskoeffizienten sind zu verwerfen. Es muss allerdings betont werden, dass es sich hierbei um einen langjährigen Erfahrungswert handelt, der allenfalls in der Praxis nachjustiert werden sollte.

Abbildung 2 zeigt den Zusammenhang zwischen verschiedenen Verteilungen und dem Bestimmtheitsmaß. In der Literatur sind immer wieder Beispiele für extreme Verteilungen anzutreffen, die zu irreführenden (zu hohen) Werten für r^2 führen, für die gegenständlichen Betrachtungen allerdings nur eine untergeordnete Rolle spielen.

Nicht-lineares Verhalten

Storage-Wachstum erfolgt selten streng linear, aber dennoch ist es nicht erforderlich, auf nicht-lineare (polynomische, exponentielle, logarithmische etc.) Regressionen auszuweichen. Es genügt, die Zeitbasis in mehrere Intervalle (ausgehend vom aktuellen Datum) zu unterteilen und für jedes dieser Intervalle eine eigene Regressionsgerade zu rechnen (mehrfache lineare Regression). Beispielsweise bewähren sich für das Tablespace-Monitoring Berechnungen auf der Basis von 3, 9, 27 und 81 Tagen.

Abbildung 3 zeigt ein typisches Verhalten eines Tablespace, in dem periodisch Löschungen vorgenommen werden. Trotzdem lässt sich daraus ein klares Kurzzeit- (Basis drei Tage) und ein Langzeit-Verhalten (alle Tage) ableiten. Prognosen auf einer Basis zwischen diesen beiden Voraussagen mussten im konkreten Fall verworfen werden aufgrund der viel zu schlechten Korrelation.

Extrapolation

Mithilfe der Regressionsgeraden und des gegebenen oberen Limits für die y-Werte (zum Beispiel die maximale Größe des Tablespace) ist es nun ein Leichtes, den

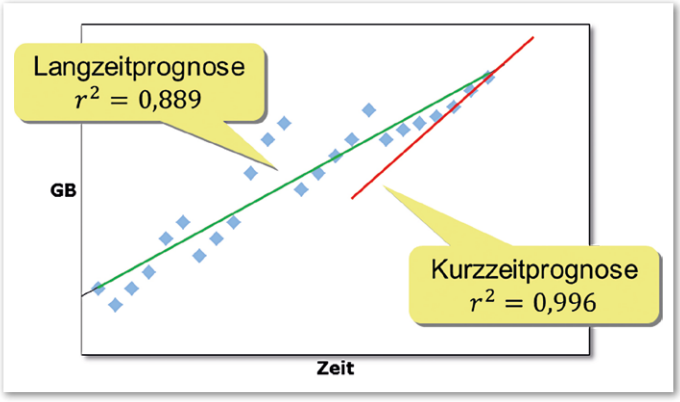


Abbildung 3: Mehrfache lineare Regression

Overflow-Zeitpunkt zu berechnen. Ausgehend von *Abbildung 1* ergibt sich durch einfache Rechnung

$$x_{max} = \frac{y_{max} - a}{b}$$

Steht also in einer Tabelle „REGR1" der Wertevorrat für x und y zur Verfügung und ist der maximale y-Wert beispielsweise mit „10" vorgegeben, so erhält man das dazugehörige x_{max} durch das SQL-Statement in *Listing 1*.

Tablespace Monitoring

Für das Tablespace Monitoring muss zunächst der verbrauchte Platz pro Tablespace periodisch aus „dba_free_space" ermittelt werden. Dies geschieht sinnvollerweise täglich − bei Bedarf auch in kürzeren Intervallen, was allerdings die Datenbank-Last erhöht −, wobei darauf zu achten ist, dass der absolute Verbrauch und nicht der Füllgrad in Prozent zur Auswertung herangezogen wird, da der absolute Space-Usage-Wert invariant gegen Tablespace-Vergrößerungen ist. Mit einer einfachen History-Tabelle wie in *Listing 2* könnte das erforderliche Insert/Select-Statement wie in *Listing 3* aussehen.

Listing 4 zeigt, wie man mit einem einzigen Select-Statement auf ein konkretes Overflow-Datum kommt − in diesem Beispiel auf der Basis von drei Tagen, wobei die aktuelle Tablespace-Größe aus „dba_data_files" ausgelesen wird. Da die Zeitachse in Form von Datums-Zeitpunkten gespeichert wurde, müssen diese Werte zunächst mithilfe eines beliebigen Bezugspunkts („sysdate") in eine Zeitdauer umgewandelt werden. Durch die Addition von „sysdate" im Output ergibt sich dann wiederum ein Overflow-Datum.

Ein tägliches Reporting sollte dem DBA alle jene Tablespaces auflisten, die aufgrund einer der Prognosen (Langzeit bis Kurzzeit) einen Overflow innerhalb einer definierten Frist vermuten lassen. Dank der ebenfalls aufgelisteten Prognosewerte (mit zugehöriger Basis) kann ein mögliches Fehlverhalten von Applikationen rasch erkannt werden, denn exponentielles Wachstum spiegelt sich beispielsweise in zunehmend kurzfristigeren Overflow-Prognosen bei abnehmender Zahl an Basistagen wider. Für andere Applikationen kann eine vorausschauende Planung durchgeführt werden, die auch kurzfristige Spitzen berücksichtigt.

Die Auswertung des momentanen Füllgrads erweist sich als überflüssig.

Systeme, die in keinem Intervall ausreichendes lineares Verhalten zeigen, entziehen sich naturgemäß jeglicher Prognose. Dies ist im Speziellen bei Entwicklungs-Systemen zu beobachten. Produktions-Systeme, die ein derartiges Verhalten zeigen, sollten dringend hinterfragt werden. Ebenfalls nicht detektierbar mit dieser Methode sind Ad-hoc-Zuwächse (durch Beladungen, Importe, Upgrades) innerhalb des Beobachtungs-Intervalls.

Overflow-Prognose für die Flash Recovery Area

Mit Einführung der Flash Recovery Area (FRA) hat Oracle die automatische Selbstverwaltung der Datenbank weiter vorangetrieben. Archivelogs (und allenfalls Flashbacklogs) müssen nicht manuell oder beim Backup gelöscht werden, sondern die Datenbank kümmert sich selbst darum. Leider gibt es bis heute keinen Automatismus in der Datenbank selbst, der ein RMAN-Archivelog-Backup anstoßen würde, wenn dies aufgrund des Füllgrads der FRA nötig wäre. Daher ist es erforderlich, die FRA analog zu den Tablespaces zu monitoren und bei Bedarf ein Backup-Skript zu starten.

Im Unterschied zur Tablespace-Prognose interessiert man sich in diesem Fall in der Regel nur für das Kurzzeit-Verhalten, um einem drohenden Overflow rasch durch den Start des Backups begegnen zu können. Daraus ergeben sich ein kurzes Sampling-Intervall von beispielsweise fünf Minuten und eine Regressionsbasis von etwa einer Stunde. Auf eine mehrfache li-

```
SELECT 10 - REGR_INTERCEPT(y,x) /
REGR_SLOPE(y,x) as xmax
   FROM regr1;
```

Listing 1

```
CREATE TABLE space_history
(
      log_date       DATE
   ,tablespace_name  VARCHAR2(30)
   ,bytes_used       NUMBER
);
```

Listing 2

```
INSERT INTO space_history
   SELECT sysdate, tablespace_name, SUM (bytes) AS bytes
      FROM dba_free_space
GROUP BY tablespace_name;
```

Listing 3

```
   SELECT tablespace_name
          ,COUNT(1) AS base
          ,SYSDATE + (max_bytes - REGR_INTERCEPT(bytes_used,days)) /
          REGR_SLOPE(bytes_used,days) AS overflow_date
          ,REGR_R2 (bytes_used, days) AS r2
      FROM (SELECT SYSDATE - a.log_date AS days
              ,a.bytes_used
              ,b.max_bytes
              ,a.tablespace_name
          FROM space_history a
              ,( SELECT tablespace_name, SUM (bytes) AS max_bytes
                  FROM dba_data_files
              GROUP BY tablespace_name) b
          WHERE a.tablespace_name = b.tablespace_name)
   WHERE days <= 3
GROUP BY tablespace_name, max_bytes;
```

Listing 4

neare Regression kann in diesem Fall gut verzichtet werden.

Die Daten für das Monitoring stellt Oracle über die View „v$flash_recovery_area_usage“ zur Verfügung, wobei zwei Prozent-Werte für die Datensammlung zu ermitteln sind:

3. % Available Space = 100 – percent_space_used – percent_space_reclaimable
4. Anteil der noch nicht gesicherten Archivelogs aus percent_space_used – percent_space_reclaimable

Es ist in diesem Fall zulässig, mit Prozentwerten zu agieren, da man im Beobachtungsintervall in der Regel nicht mit einer Größenveränderung der FRA rechnen muss. Will man dennoch mit absoluten Werten rechnen, ist die Gesamtgröße der FRA aus dem Parameter „db_recovery_file_dest_size“ heranzuziehen.

Die Extrapolation auf 100 Prozent ergibt eine Abschätzung für den Overflow-Zeitpunkt, der Füllgrad kann zusätzlich als statische Metrik verwendet werden, um eine bestimmte Reserve an freiem Platz nicht zu unterschreiten. Das RMAN-Skript selbst kann über einen „DBMS_SCHEDULER“-Job gestartet werden. Wenn der Anteil an noch ungesicherten Archivelogs zehn Prozent unterschreitet, ist ein Backup kaum mehr sinnvoll. In diesem Fall muss es zu einem Alert kommen, weil die FRA offenbar zu klein dimensioniert ist.

Runtime Forecasts (RTFC)

Bei der Voraussage von Laufzeiten von Batch-Prozessen geht es nicht nur darum zu wissen, wann der Prozess fertig sein wird. Mindestens ebenso wichtig ist das rasche Erkennen abnormen Verhaltens, nicht reproduzierbarer Laufzeiten und Ausreißern (vor allem nach oben).

Die Klassifikation der Daten im Sinne einer Auftrennung nach zu erwartender Laufzeit ist eine Grundvoraussetzung. *Abbildung 4* zeigt ein typisches Verhalten eines Prozesses, der an bestimmten Tagen aufgrund einer kalenderabhängigen Verarbeitungslogik (Ultimoverarbeitung, Quartalsabschluss, etc.) eine andere Laufzeit aufweist als normalerweise. In diesem Fall ergeben sich also zwei RTFCs für unterschiedliche Tagestypen. Der gezeigte Ausreißer muss in jedem Fall ignoriert werden.

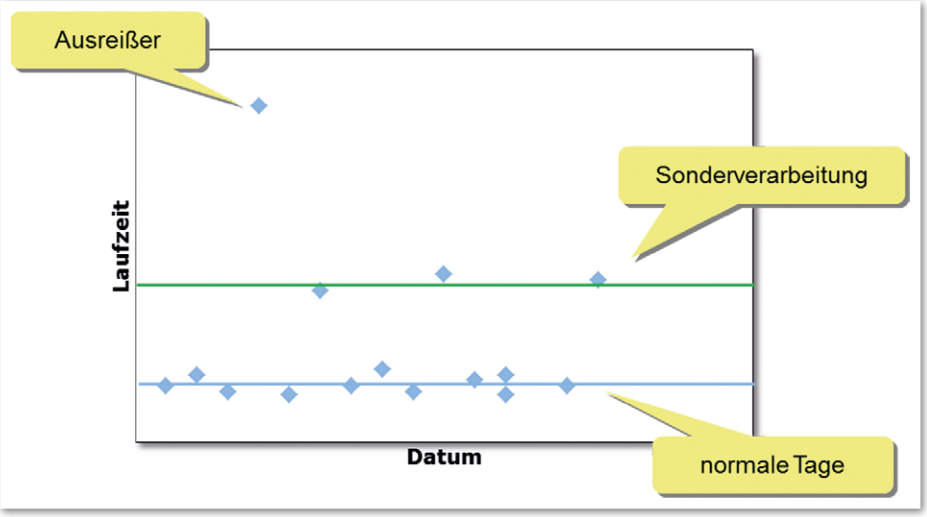


Abbildung 4: Typisches Verhalten eines Batch-Prozesses

Die einfache Mittelwertbildung ist in jedem Fall zu wenig, um zu einer Voraussage zu kommen. Nach erfolgter Klassifizierung (beispielsweise Trennung) der Daten muss der zunächst pro Tagestyp errechnete Mittelwert von Ausreißern befreit und korrigiert werden. Die Statistik sieht für Ausreißer-Tests mannigfache Methoden vor. Für die hier diskutierten Anwendungen hat sich ein (manchmal auch nur angenäherter) t-Test bewährt.

Da man zu Recht davon ausgehen kann, dass die Messwert-Schwankungen einer Normalverteilung unterliegen, berechnet man zunächst den Mittelwert der einzelnen Messpunkte (in diesem Anwendungsfall sind das die Laufzeiten) sowie die Varianz s^2 der Stichprobe. In SQL stehen dafür die Funktionen „AVG“ und „VAR_SAMP“ beziehungsweise „STDDEV_SAMP“ (oder nur „STDDEV“) zur Verfügung, wobei auch hier wiederum aus der Formel ersichtlich ist, dass man ausschließlich mit Mittelwert-Funktionen hinkommen würde.

In *Abbildung 5* ist die Bedeutung der Standardabweichung σ einer unendlich großen Grundgesamtheit in Bezug auf den Mittelwert μ dargestellt. In einem bestimmten Bereich (der als Vielfaches von σ angegeben wird) um den Mittelwert herum findet man alle Einzelwerte mit einer bestimmten Wahrscheinlichkeit. Diese Abweichung

$$\Delta y = \pm \sigma \cdot z_\alpha$$

wird als Vertrauensbereich oder Konfidenzintervall zum Signifikanzniveau α bezeichnet.

So kann man also beispielsweise bei einem Signifikanz-Niveau von $\alpha = 0{,}05$ beziehungsweise $1 - \alpha = 0{,}95$ davon ausgehen, dass 95 Prozent aller Messwerte in einem Bereich von $\pm 2\sigma$ um den Mittelwert liegen ($z_{0{,}05}=2$). Aufgrund der Tatsache, dass man es in der Realität mit einer endlichen Zahl von Datenpunkten zu tun hat (im Fall von monatlichen Batchprozessen können das tatsächlich mitunter auch nur 3 oder 4 sein), ergibt sich eine größere Unsicherheit (also eine breitere Verteilung), was in der Statistik durch die Verwendung der Student- oder t-Verteilung berücksichtigt wird:

$$\Delta y = \sigma \cdot t_\alpha$$

Der Student-Faktor t_α ist für ein bestimmtes Signifikanz-Niveau aus Tabellen in Abhängigkeit von der Zahl der Freiheitsgrade (das ist in diesem Fall die Zahl der Messwerte minus 1) abzulesen, geht aber für große Datenmengen gegen die z-Werte der Normalverteilung.

Als Ausreißer werden nun jene Punkte klassifiziert, die außerhalb des gewählten Vertrauensbereichs zu liegen kommen. Verzichtet man auf die Korrektur für kleine Stichproben, so kann man näherungsweise davon ausgehen, dass alle Werte außerhalb des 3σ-Bereichs (also bei An-

wendung einer Wahrscheinlichkeit von 99,7%) als Ausreißer zu betrachten sind. Diese 3σ-Regel hat sich in der Praxis als brauchbare Näherung erwiesen und liefert in der Regel sehr gute Ergebnisse.

Praktische Durchführung von Ausreißertests

Unter Verwendung der beschriebenen 3σ-Regel ergibt sich folgende Vorgehensweise:

1. Auffinden des Wertes mit der größten Abweichung vom Mittelwert

$$max|y_i - \bar{y}|$$

2. Berechnung des Schätzwertes für z:

$$\hat{z} = \frac{max|y_i - \bar{y}|}{s}$$

3. Ist $\hat{z} > 3$, handelt es sich um einen Ausreißer
4. Der Ausreißer wird eliminiert und Mittelwert und Standardabweichung werden neu berechnet

Dieses Prozedere wird so lange wiederholt, bis kein Ausreißer mehr gefunden wird. Für die Umsetzung in PL/SQL bedeutet dies die Programmierung einer rekursiven Routine. Soll die geringe Datenmenge Berücksichtigung finden, ist anstelle von $z = 3$ der entsprechende t-Wert aus der Tabelle für einseitige t-Tests zu entnehmen. *Abbildung 6* stellt den rekursiven Kreisprozess nochmals dar.

RTFC Alerts

Mit der nun bekannten Baseline (dem eigentlichen Runtime Forecast) lässt sich leicht beurteilen, ob die Laufzeit des nächsten Batchprozesses im erwarteten Rahmen liegt oder nicht. Grob gesagt, ist der nächste Wert innerhalb des berechneten Konfidenzintervalls (also beispielsweise innerhalb des 3σ-Bereichs) zu erwarten. Es liegt daher nahe, mit dem neuen Wert einen Ausreißertest auf Basis der Werte der vorgegebenen Baseline durchzuführen. Bei größeren Datenmengen kann auch in diesem Fall auf eine Endlichkeitskorrektur verzichtet und die Normalverteilung selbst verwendet werden.

Ein praxisorientiertes Alert-Schema könnte also (zumindest für größere Datenmengen) so aussehen, dass bei Überschreitung des Konfidenz-Intervalls (3σ-Bereich) ein Warning-Alert gesendet wird, bei Überschreitung des 6α-Bereichs hingegen ein Critical Alert ausgelöst wird. Speziell für Batch-Prozesse ist in jedem Fall ein hartes unteres Limit zu setzen (etwa eine Minute), unter dem keinerlei Alerting erfolgt.

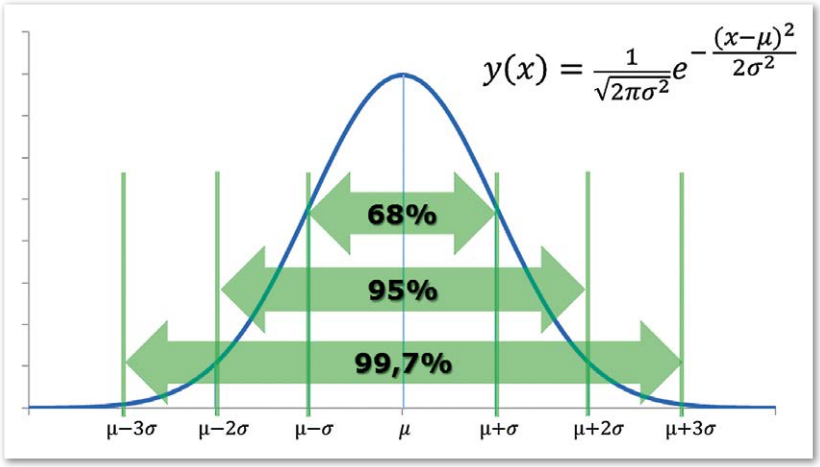


Abbildung 5: Gauß'sche Normalverteilung

Instabile Prozesse lassen naturgemäß keine Aussagen zu, können aber an der Zahl der Ausreißer schnell erkannt werden. Auch im Rahmen von RTFCs sollten lang- und kurzfristige Regressionen gerechnet werden, um einen Anstieg von Laufzeiten zu detektieren. Speziell im Data-Warehouse-Umfeld treten solche Fälle häufig durch wachsende Datenmengen bei gleichzeitig schlecht geplantem oder gar fehlendem Partitioning auf.

Availability Monitoring

Erfolgreiches Datenbank-Monitoring muss in erster Linie folgende zwei Fragen sofort und zu jeder Zeit zweifelsfrei beantworten können:

1. Ist eine bestimmte Instanz verfügbar?
2. Ist die Response-Zeit akzeptabel?

Dass diese Grundfragen allein mit Oracle-Datenbankmitteln und ohne großen Aufwand beantwortet werden können, zeigt exemplarisch der folgende Abschnitt. Zunächst benötigt man eine zentrale, dedizierte Monitoring-Datenbank, die über ein Repository aller Oracle-Instanzen mit Hosts, Listener-Ports und SIDs verfügen muss. Idealerweise wird diese Datenbank gleichzeitig als CMDB genutzt und dient auch als Kollektor für andere Monitoring-Ergebnisse (etwa Tablespace-Prognosen). Mithilfe der Repository-Information werden durch einen Startup-Prozess Database-Links zu allen Instanzen angelegt, wobei darauf zu achten ist, dass nur IP-Adressen

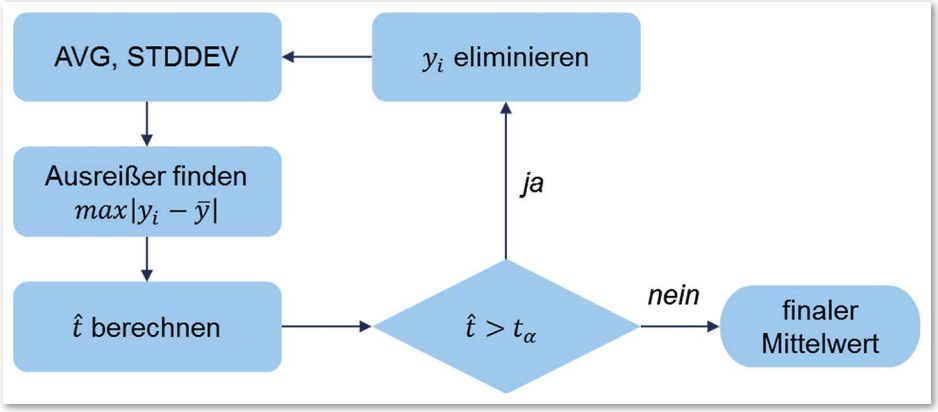


Abbildung 6: Schema eines Ausreißer-Tests

```
create database link db1 con-
nect to … identified by … using '(de
scription=(address=(protocol=tcp)
(host=192.168.0.61)(port=1521))

 (connect_data=(SID=DB1)
(server=dedicated))';
```

Listing 5

```
v_t0 := systimestamp;
select null into v_dummy from dual;
v_t1 := systimestamp;
select null into v_dummy from dual;
v_t2 := systimestamp;
v_r1 := v_t1-v_t0;  -- logon included
v_r2 := v_t2-v_t1;
```

Listing 6

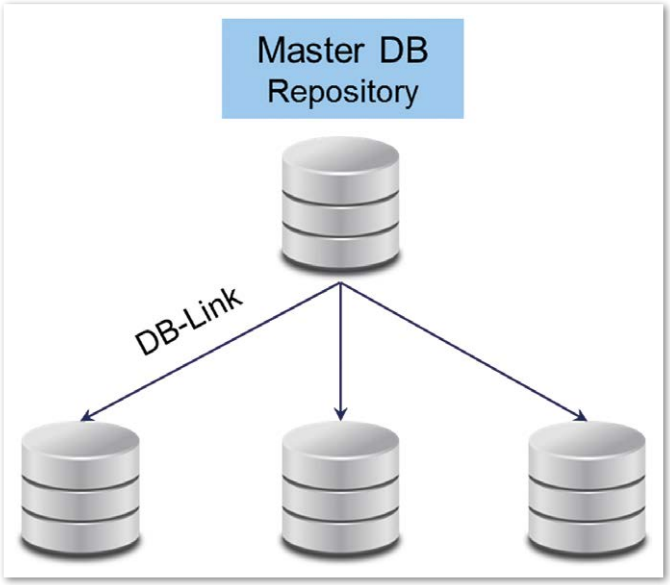


Abbildung 7: Setup für Availability-Monitoring

anstelle von Hostnamen verwendet werden, um bei der späteren Messung des Logon-Prozesses nicht implizit den DNS-Server mitzumessen. Zudem ist auf eine Dedicated-Server-Verbindung zu achten. *Abbildung 7* zeigt schematisch das Setup.

Listing 5 zeigt, wie die Database-Links aufzubauen sind. Pro Instanz wird nun ein eigener minütlicher DBMS_SCHEDULER-Job gestartet, der zwei Selects absetzt, die Zeit misst, in einer Messwert-Tabelle ablegt und danach wieder terminiert. *Listing 6* zeigt den Kernteil der Implementation in PL/SQL. Das erste Select misst den Logon-Vorgang, während das zweite Select die reine Response-Zeit der Instanz widerspiegelt.

Auf diese Art und Weise entsteht pro Instanz eine große Anzahl von Messwerten, aus denen eine Baseline (also ein Referenzwert) mit hoher statistischer Genauigkeit gerechnet werden kann. Die Vorgehensweise ist identisch mit jener, die bereits für die Runtime Forecasts demonstriert wurde, also Berechnung von Mittelwert und Varianz, Eliminieren von Ausreißern, Hinterlegen der gefundenen Referenzwerte. Statistisch signifikante Abweichungen werden so problemlos erkannt und alertiert. Zusätzlich empfiehlt sich auch in diesem Fall eine Regressionsrechnung, um eine eventuell vorkommende Drift zu erkennen.

In Analogie zum Ausreißertest für Einzelwerte lässt sich auch ein Konfidenz-Intervall für den Mittelwert definieren, wobei näherungsweise der Student-Faktor wieder durch den z-Wert der Normalverteilung ersetzt werden könnte:

$$\Delta\bar{y} = \frac{s \cdot t_\alpha}{\sqrt{n}}$$

In den Prüfwert für den (zweiseitigen) t-Test gehen die Differenz der Mittelwerte und die mittlere Standardabweichung dieser Differenz ein:

$$\hat{t} = \frac{|\bar{y}_1 - \bar{y}_2|}{\sqrt{\frac{s_1}{n_1} + \frac{s_2}{n_2}}}$$

Wird eine signifikante Änderung der Baseline festgestellt, deutet dies auf eine Systemveränderung hin. Im Übrigen spricht aber nichts dagegen, die neue Baseline (ob signifikant verschieden oder nicht) als neue Referenz zu verwenden.

Fazit

Intelligentes Monitoring baut auf einfachen Prinzipien auf:

- Know your data
  - Was ist zu monitoren?
  - Welche Verteilungen und Verhaltensweisen liegen vor?
  - Daten kennenlernen
- Keep it simple – einfache und nur wenige KPIs festlegen
- Zeitabhängige Metriken sind unverzichtbar
- Ein zentrales Repository ist die beste Informationsquelle
- Intelligentes Alerting – automatische Schwellwerte definieren (setzt automatische Baselines voraus)

Für die automatisierte Analyse ist eine fundierte Kenntnis der erforderlichen statistischen Methoden unerlässlich, auch wenn sich mitunter in der Praxis Näherungsmethoden als ausreichend erweisen. Speziell bei der Overflow-Analyse von Tablespaces wird man allerdings um eine manuelle Endbeurteilung der Daten in vielen Fällen nicht herumkommen.

Dr. Thomas Petrik
thomas.petrik@sphinx.at

Continuous Integration für die Oracle-Datenbank und Apex

Peter Busch und Dominic Ketteltasche, MT AG

Immer kürzer werdende Intervalle von Releases und Hotfix-Bearbeitung machen eine Automatisierung für das Deployment von Datenbankobjekten und Anwendungen erforderlich. Gesicherte und geprüfte Abläufe sind dafür unerlässlich.

Um die entsprechenden Auslieferungen und Sicherheitsmechanismen effektiv nutzen zu können, werden unterschiedliche Werkzeuge benutzt. Continuous Integration (CI) ist der Oberbegriff, unter dem der gesamte Ablauf wiederkehrend abgehandelt wird. Eine Automatisierung der Prozesse führt zu einer Verkürzung der Zeitspanne zwischen den einzelnen Prozess-Schritten, aber auch zu Verbesserungen bei der Laufzeit der aufgerufenen Skripte. Außerdem liefert die Automatisierung der Prozesse eine einheitliche, projektweite Verfahrensweise für die unterschiedlichen Umgebungen.

Dieser Artikel stellt das bei der MT AG zum Einsatz kommende Continuous-Integration-Verfahren vor (siehe *Abbildung 1* in wiederkehrender Ausführung) – am Beispiel eines Projekts, in dem schon in der Entwicklungsphase fertige Teile an den Kunden geliefert wurden. Das Einspielen der Datenbankobjekte sowie der Applikation musste so einfach wie möglich erfolgen. Die Lösung dafür war CI. Anhand des

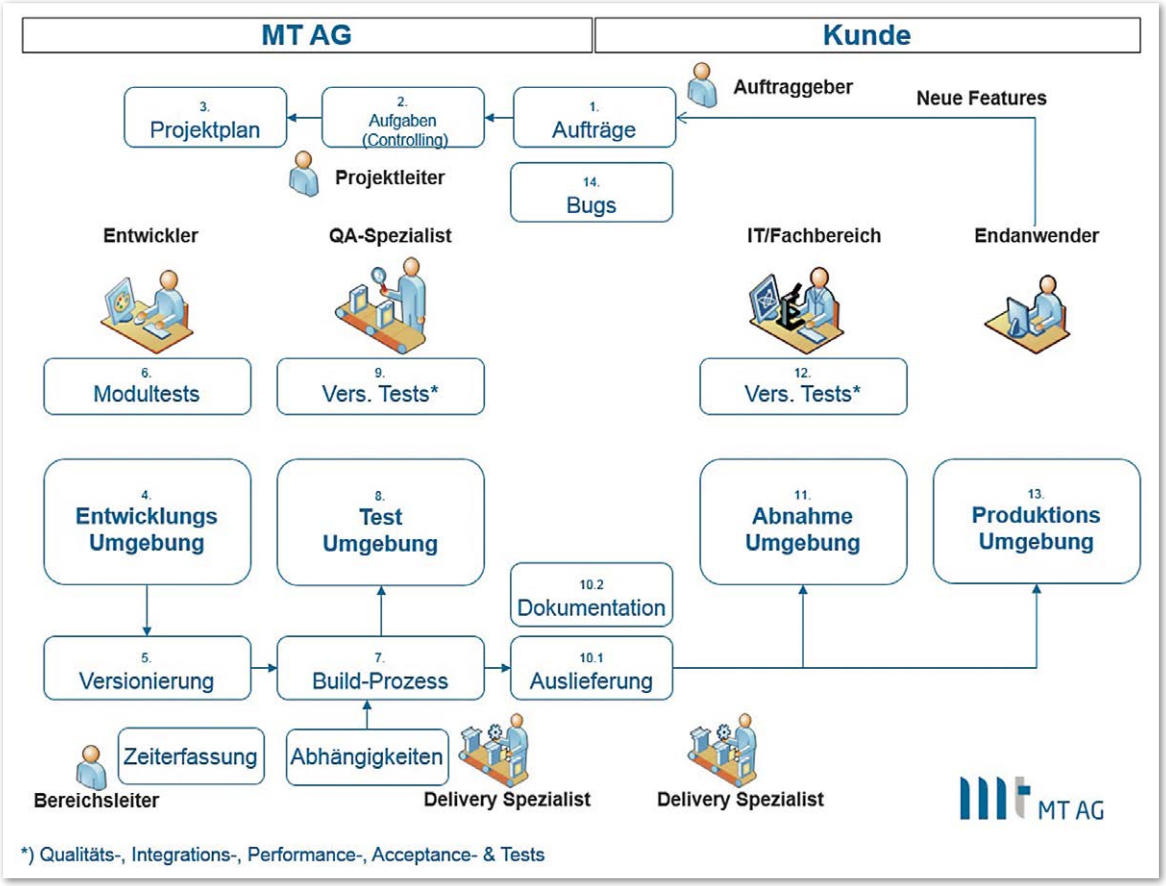


Abbildung 1: Referenz-Architektur für die Software-Entwicklung bei der MT AG

Projekts sind nachfolgend die verschiedenen Elemente aufgezeigt.

Continuous Integration in der Praxis

Als Tools sind TortoiseSVN für die Versionierung und Bug Tracker von Jira im Einsatz. Das Projekt basiert auf Windows-Server und Oracle-Datenbanken. Die Skriptverarbeitung erfolgt durch Batch-Prozesse, die das webbasierte System Hudson im Hause startet und überwacht. Das aktuelle Release wird dem Kunden jeweils als ZIP-Datei zur Verfügung gestellt. Darin sind alle Skripte enthalten, die für die Auslieferung notwendig sind.

In dieser kurzen Beschreibung stellt sich direkt eine Besonderheit und auch Herausforderung dieses Projekts dar. Die Entwicklung erfolgt in einer völlig anderen Umgebung (inhouse MT AG) als der Deployment-Umgebung (Kunde).

Die Architektur stellt sich wie folgt dar: Die Test- und Entwicklungsumgebung für die Release-Arbeiten und die Hotfix-Bearbeitung sind auf den hauseigenen Servern der MT AG installiert. Die Abnahme- und Produktionsumgebung ist beim Kunden vor Ort. Eine Umgebung, die der Produktion entspricht, ist bei der MT AG eingerichtet. Über diese kann dem Kunden eine aktuelle Version des Entwicklungsstands zur Verfügung gestellt werden.

Anforderungen des Kunden werden in dem Tracking-Tool Jira aufgenommen und an die entsprechenden Entwickler weitergereicht. Es wird der Stand der Arbeiten eingetragen und bei Bedarf der Status entsprechend gewechselt. Dort können auch die verbliebenen und verbrauchten Zeiten angezeigt werden.

Für jedes Datenbankobjekt wird ein Skript erzeugt. DML-Skripte und Strukturänderungen (DDL-Skripte) können fachlich und objektabhängig zusammengefasst sein. Die Skripte sind in eine feste Verzeichnisstruktur unterteilt. Alles, was erzeugt und ausgeliefert wird, unterliegt einer Versionierung. Die geänderten Skripte werden in ein Release-Verzeichnis eingebunden. Zusätzlich erfolgt noch die Aufnahme der Skriptnamen mit dem entsprechenden Release in eine Deployment-Anwendung. Über diese sind fachliche Abhängigkeiten eingebbar. Weitere Abhängigkeiten entwickeln sich durch die vorgegebene Verzeichnisstruktur *(siehe Abbildung 2)*.

Aufgrund dieser sich daraus ergebenden Reihenfolge sind die Skripte in sogenannten „Ausführungsskripten“ zusammengefasst. Im Installationsablauf wird bei einem Wechsel der Ausführungsskripte ein Re-Compile aller invaliden Objekte in den betroffenen Schemata ausgeführt.

Das fertige Release

Um die Vollständigkeit der Skripte mit größtmöglicher Sicherheit zu gewährleisten, erfolgt eine Vollständigkeitsprüfung über den Vergleich der eingetragenen Skripte der Deployment-Anwendung und der getaggten Skripte im Versionierungs-Tool. Um die umgesetzten Anforderungen vor einer endgültigen Auslieferung zu prüfen, werden die Skripte in ein Test-System eingespielt. Hier erfolgt ein zweiter Test durch QS-Spezialisten (Vier-Augen-Prinzip). Sollten wider Erwarten noch Skripte für die Auslieferung fehlen, würde es an dieser Stelle auffallen. Um sicherzustellen, dass eine Auslieferung fehlerfrei erfolgt, kann das Zielsystem in der Testumgebung auf eine erforderliche Vorversion geladen werden.

Sofern ein Release ansteht, werden zwei Phasen ausgeführt. Phase 1: Die Entwicklung eines Release ist abgeschlossen, alle Skripte sind erstellt und mit dem Versions-Label getaggt. Außerdem sind die Skripte in die Deployment-Anwendung eingetragen. Der angestoßene Job von Phase1 durchläuft folgende Prüfungen:

- Stimmen die Anzahl und das Format der übergebenen Parameter?
- Sind mit den Zugangsdaten alle erforderlichen Systeme erreichbar?
- Sind die vorgegebenen Verzeichnisse und die Batch-Skripte für das Deployment vorhanden?

Für den Fall, dass die Prüfungen erfolgreich sind, wird ein Ausführungsskript mit den Skriptnamen der eingetragenen Skripte der Deployment-Anwendung erstellt. Die aufgeführten Skripte müssen, wie bereits beschrieben, im Release-Verzeichnis enthalten sein und umgekehrt. Die Abhängigkeit der Skripte wird durch die Ordnerstruktur festgelegt – Datenbank-Schema, DDL, DML, PL/SQL, Apex-Seiten-Objekte (Views, Packages etc.). Innerhalb dieser Verzeichnisstruktur kann unabhängig von der Ordnerstruktur eine zusätzliche Abhängigkeit über die Eingabe innerhalb der Deployment-Umgebung gebildet werden.

Bevor das Release an den Kunden ausgeliefert wird, müssen die Änderungen im Testsystem noch Prüfungen durchlaufen. Das Einspielen in das Testsystem dient dabei ebenfalls als Testlauf für das Deployment beim Kunden. Anschließend wird der Job von Phase 2 angestoßen. Nach dem Aufruf werden analog zu Phase 1 die dort aufgelisteten Prüfungen durchlaufen. Ab hier wird der eigentliche Prozess ausgeführt.

- Der Start des Release wird in eine LOG-Tabelle eingetragen.
- Der derzeitige Release-Stand der Datenbank muss der Vorversion entsprechen.
- Die Skripte des Release werden aufgerufen.
- Die einzelnen Skripte werden in der LOG-Tabelle protokolliert.
- Es werden drei Re-Compiles durchgeführt, um fehlerhafte Datenbank-Objekte zu entfernen. Drei reichen der Erfahrung nach aus, um durch Abhängigkeiten entstandene invalide Objekte valide zu bekommen.

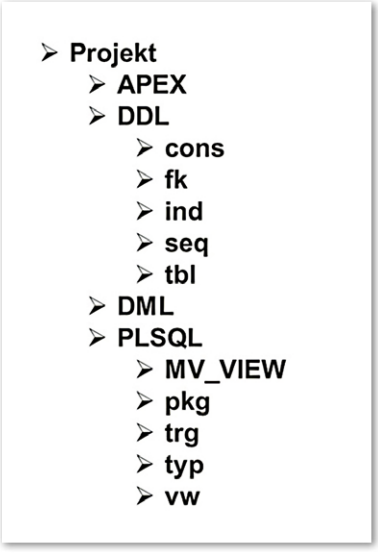


Abbildung 2: Verzeichnisstruktur in Subversion

- Es findet eine Abschlussprüfung statt.
- Der neue Release-Stand wird in die Steuerungstabelle eingetragen.
- Die LOG-Dateien werden in ein zentrales Verzeichnis abgestellt.

Jedes Batch-Skript der Phase 2 erzeugt eine LOG-Datei. Diese Dateien werden am Ende des Laufs auf Fehler durchsucht und die Fehlertexte sowie die Logfile-Namen gesammelt ausgegeben. Gegebenenfalls erfolgt auch ein Abbruch der Phase 2.

Automatisierung

Phase 1 und Phase 2 finden ohne Mitwirken des Kunden statt. Um den Deployment-Prozess gleichzuhalten, werden über Hudson immer nur Batch-Prozesse gestartet, so wie es der Kunde auch durchführen muss. Es ist allerdings möglich, über Parameter zum Beispiel die Release-Version oder die Zielumgebung vorzugeben.

Da die Versionierung unabhängig von der des Kunden ist, besteht die Möglichkeit, notwendige SVN-Arbeiten zu automatisieren. So werden die Tag-Verzeichnisse für die einzelnen Releases per Skript angelegt, vor dem Start der Phase 1 das Verzeichnis aktualisiert und die in Phase 2 erzeugten LOG-Dateien zentral ins SVN eingecheckt.

Aktuell gibt es noch Teile des Deployments, die bislang nicht in das CI mit aufgenommen wurden. In erster Linie sind es JavaScript-Dateien, die noch manuell auf die einzelnen Server zu übertragen sind. Im Prinzip betrifft es alle Objekte, die keine DB-Objekte sind oder nicht zur Apex-Anwendung gehören. Es gehört zu den zukünftigen Zielen, diese ebenfalls mit in den automatisierten Prozess aufzunehmen.

Nach dem Abschluss der Entwicklungsarbeiten wird das Release dem Kunden ausgeliefert. Hierfür fasst ein Batchprozess alle erforderlichen Skripte in einem ZIP-File zusammen, das dem Kunden zur Verfügung gestellt wird. Der Kunde entpackt das ZIP-File und führt den Prozess der Phase 2 aus. In dem ZIP-File sind alle notwendigen Account-Dateien für die Umgebungen des Kunden enthalten. Es ist somit nur noch die Phase 2 mit der entsprechenden Account-Datei zu starten. Aufgrund der Tatsache, dass nur Batch-Skripte verwendet werden, findet beim Kunden der gleiche Prozess wie bei uns im Hause statt. Potenziell entstehende Fehler beim Release können somit nicht auf den übergebenen Skripten beruhen.

Apex-Applikation

Nach Fertigstellung der Apex-Applikation auf der Entwicklungsumgebung erfolgt der Export der Applikation mithilfe des Java-API ApexExport, das in der Apex-Installation enthalten ist. Es stellt die Applikation als SQL-Skript in das File-System.

Dieser Prozess wird über ein Batchskript ausgeführt und wie bei den Datenbank-Objekten über Hudson angestoßen.

Für den Import der Applikation in das Testsystem werden die Parameter „WORKSPACE_ID“, „APPLICATION_ID“, „SCHEMA“, „APPLICATION_ALIAS“ und der „OFFSET“ entsprechend gesetzt. Über SQL*Plus erfolgt dann die Ausführung des SQL-Files der Applikation. Im Nachgang erfolgen das „UNAVAILABLE“-Setzen der veralteten Applikationen und das Setzen der aktuellen Version auf „AVAILABLE“. Somit ist ein Start der Applikation mit dem gleichen Link immer dann möglich, wenn die „APPLICATION_ID“ oder der „APPLICATION_ALIAS“ beibehalten wird.

Da das Export-Skript der Anwendung nicht verändert wird, ist ein Einspielen mit der Import-Funktion des Application Builder in andere Umgebungen möglich. Sobald die Applikation bereit zur Auslieferung ist, wird das SQL-File ins SVN eingecheckt, entsprechend der Version getaggt und in das ZIP-File der Datenbank-Skripte mit aufgenommen.

Die Vorteile

Durch die Anforderung des Kunden, während der laufenden Release-Arbeiten dringende Korrekturen (Hotfix) an die Produktion auszuliefern, sind im Laufe des Projekts zwei weitere Umgebungen bei der MT AG notwendig geworden. Es war ebenfalls erforderlich, die Datenbank-Schemata per Data Pump und die Skripte der Releases in die neuen Umgebungen einzuspielen. An dieser Stelle ist die Steuerung über Hudson sehr hilfreich gewesen. Über Variablen und Auswahlboxen können die einzelnen Umgebungen bedient werden. Ein zentrales Verzeichnis, das von allen Servern erreicht werden kann, dient als Drehscheibe für alle Daten, bei denen keine Versionierung notwendig ist.

Mittlerweile werden fünf Umgebungen für das Projekt vorgehalten, bei denen es mittels Hudson möglich ist, die Applikationen von jeder Umgebung aus zu exportieren und in jede zu importieren. Daraus ergeben sich die folgenden Vorteile. Innerhalb der eigenen Umgebungen sind die Autoren variabler und können schneller auf Kundenwünsche reagieren. Wenn zum Beispiel ein Hotfix neben den normalen Releases notwendig ist, kann ohne großen Aufwand eine aktuelle Umgebung aufgebaut werden, die auch dem Produktionsstand entspricht. Die Laufzeiten der Skripte haben sich durch den Aufruf über Hudson und durch die Optimierungen innerhalb des CI deutlich reduziert.

Da der Kunde für seine Installation die gleichen Skripte verwendet wie wir, profitiert er in Bezug auf Sicherheit von den laufenden Änderungen. So kommen ihm beispielsweise Fehlerroutinen, die im Hause der MT AG entwickelt werden, direkt zugute.

Im Bug Tracker Jira werden alle gemeldeten Änderungen eingetragen und mit der Release-Nummer gekennzeichnet. Über seine Auswertungsmöglichkeiten ist der Stand der Release-Arbeiten schnell abzulesen. Die eingetragenen Skripte in der Deployment-Anwendung sind mit den entsprechenden Jira-Nummern versehen. So ist zu erkennen, welche Skripte für einen Jira-Eintrag erstellt wurden und wer für diese Skripte verantwortlich ist.

Im Laufe des Projekts musste die Vollauslieferung auf ein inkrementelles Verfahren umgestellt werden. Dieses Verfahren stellte eine besondere Herausforderung dar, da die erzeugten Skripte von da ab Restart-fähig sein mussten. So war seitdem bei DDL-Skripten zu überprüfen, ob im Repository die Änderungen bereits enthalten sind. Auf diese Weise ließen sich ORA-Fehler vermeiden, die ein fehlerhaftes Ende der Auslieferung bewirken würden. Bei DML-Skripten kam die Prüfung hinzu, ob die Änderungen im Datenbankschema schon vorgenommen wurden. Für Sonderfälle musste ein Exception-Handling mit dem Abfangen des Fehlers eingebaut werden.

Fazit und Ausblick

Es ist festzuhalten, dass bereits viele Optimierungen geschehen sind, sich CI aber trotzdem immer noch im Wandel befindet, sei es durch eigene Ideen oder durch Anforderungen von Kunden. Für die Zukunft sind somit noch einige Erweiterungen denkbar beziehungsweise wünschenswert:

- Der Einsatz von Java-basierten Programmen wie beispielsweise ANT oder Maven, um Plattform-unabhängig zu sein
- Automatisches Einstellen der JavaScript-Dateien auf den Webserver
- Die Installation von Dumps beziehungsweise der Rücksprung auf eine beliebige Ausgangsversion generell oder speziell vielleicht sogar über Hudson
- die inkrementelle Auslieferung der Apex-Anwendung
- Allgemein ein höherer Grad der Automatisierung
- Umstieg auf Jenkins als CI-Server, da dort die Weiterentwicklung stärker vorangetrieben wird

Peter Busch
peter.busch@mt.ag.com

Dominic Ketteltasche
dominic.ketteltasche@mt-ag.com

Der Optimizer

Klaus Reimers, ORDIX AG

Der Artikel zeigt anhand von Beispielen die Grundzüge der SQL-Optimierung auf und behandelt Parsing, Statistiken („dbms_stats"), Initialisierungsparameter und die Fixierung von Ausführungsplänen so, dass auch Einsteiger dem Thema gut folgen können. Etwas tiefer wird der für viele undurchsichtige Bereich „MBRC" beschrieben.

Der Optimizer ist eine der zentralen Komponenten des gesamten Oracle-Systems. Er entscheidet, in welcher Form ein SQL-Statement intern abgearbeitet wird. Sowohl der Administrator als auch der Entwickler haben einige Möglichkeiten, auf den letztendlich entstehenden Ausführungsplan einzuwirken.

Der Anwender setzt im Client ein SQL-Statement ab, das dann in der SQL-Area abgelegt wird. Die SQL-Area ist dabei eine Komponente im Library Cache des Shared Pool innerhalb der SGA. Das SQL wird geparst, der Ausführungsplan wird entweder erstellt oder als Konserve aus der Datenbank genommen. Anschließend wird das Statement ausgeführt.

Das Parsen

Immer wenn ein SQL-Statement formuliert wird, muss es zunächst geparst werden. Das Parsen besteht immer aus drei Teilschritten:

- Syntaxcheck
- Semantikcheck
- Erstellung des Ausführungsplans

Zunächst erfolgt die Prüfung auf Korrektheit der Anweisung. Im semantischen Check wird das SQL-Statement dann in seine Einzelteile zerlegt. Hier fragt Oracle im Data Dictionary nach, ob die in der Anweisung angesprochenen Tabellen und Spalten real verfügbar sind. Diese Nachfrage erfolgt natürlich auf intern gebildete SQL-Abfragen, den sogenannten „rekursiven SQLs".

Die Hauptarbeit beim Parsen ist die Erstellung des Ausführungsplans. Hier errechnet der Optimizer den vermeintlich optimalen Ausführungsplan. Dazu wer-

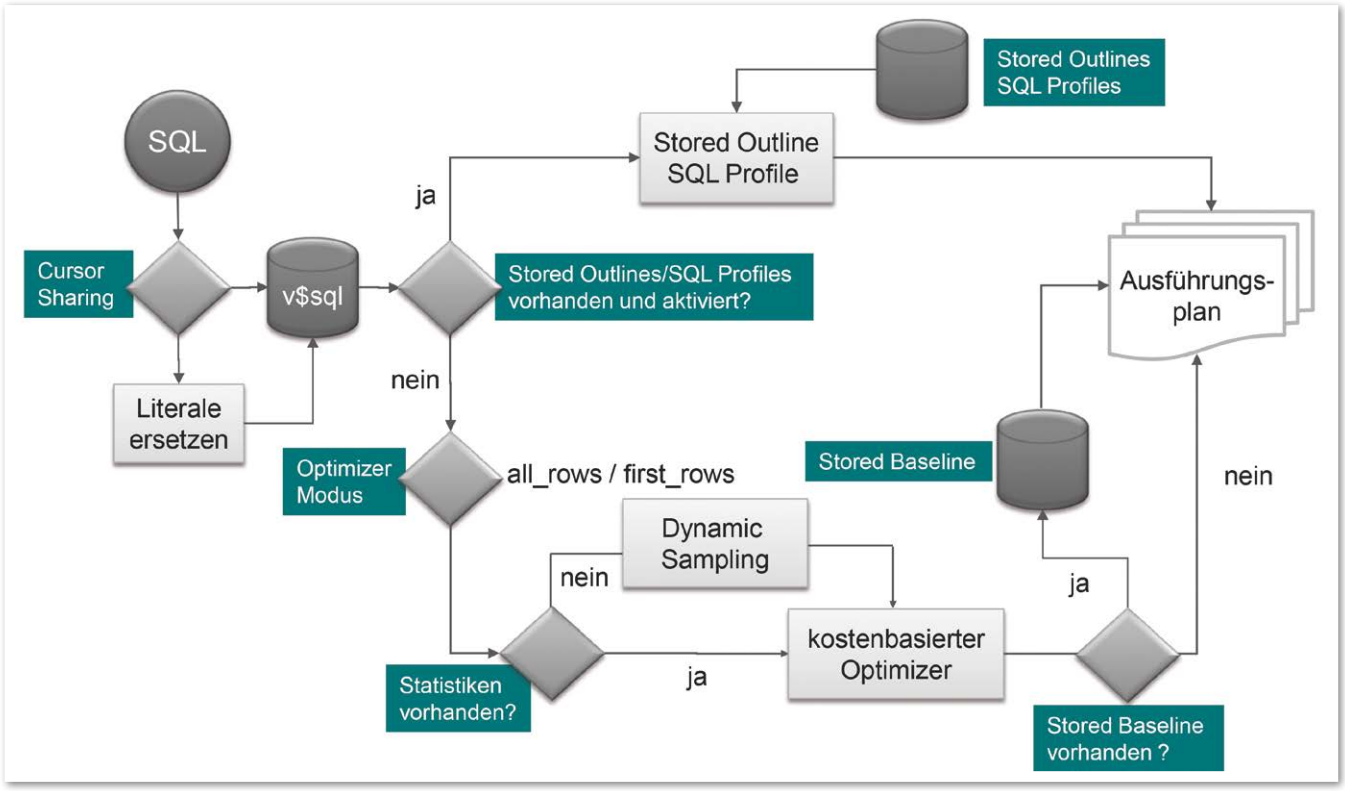


Abbildung 1: Erstellung von Ausführungsplänen - vom SQL zum Ausführungsplan

Statement in der SQL-Area vorliegt. Verständlicherweise ist ein Hard Parse viel aufwändiger als ein Soft Parse. Daher sollten möglichst viele Soft Parsings erfolgen. Dies erreicht man natürlich nur, wenn viele gleiche Statements formuliert werden. *Listing 1* verdeutlicht, dass diese Forderung sicher nicht bei Verwendung von Literalen, sondern nur von Bind-Variablen möglich ist: Daher sollten möglichst viele SQLs mit Bind-Variablen geschrieben werden. Ist dies nicht möglich oder nicht gemacht worden, so kann der Administrator immer noch das sogenannte „Cursor Sharing" aktivieren. Hierdurch werden alle Literale vor dem eigentlichen Parsen in Bind-Variablen verwandelt. Dieses Verfahren kann allerdings einige Nachteile (vor allem bei der Verwendung von Histogrammen) nach sich ziehen.

```
SELECT * FROM MITARBEITER WHERE MITARBEITERNR = 4711;
SELECT * FROM MITARBEITER WHERE MITARBEITERNR = 4712;
SELECT * FROM MITARBEITER WHERE MITARBEITERNR = :manr;
```

Listing 1

Der Optimizer

Der Optimizer generiert den eigentlichen Ausführungsplan. In früheren Versionen von Oracle hat man zwischen dem regelbasierten und dem statistischen Optimizer unterschieden. Der regelbasierte Optimizer (rule based) wird seit Version 10g nicht mehr unterstützt, kann aber nach wie vor intern (vor allem über Hints) verwendet werden. Der Entwickler hatte bei dieser Variante einen starken Einfluss auf die Güte des Plans. Dieser war stabil, da die Generierung auf Basis eines Regelwerks vollzogen wurde und bei gleichem SQL immer der gleiche Plan herauskam.

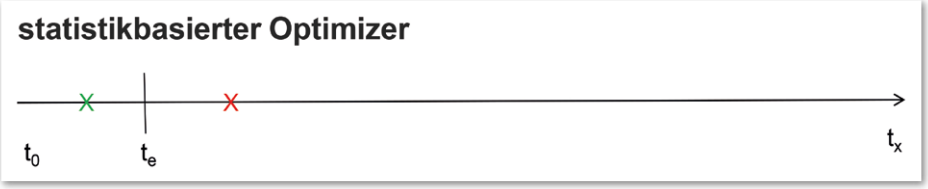


Abbildung 2: Statistikbasierter Optimizer – die Bedenkzeit des Optimizer

Der statistische Optimizer ist heute das Mittel der Wahl. Der Ausführungsplan wird auf Basis zweier Hauptkomponenten erstellt (Statistiken und Parameter). Dieser Optimizer kann in verschiedenen Varianten genutzt werden. Grundsätzlich unterscheidet man zwischen den Einstellungen „FIRST_ROWS" und „ALL_ROWS".

„ALL_ROWS" ist der Standard und optimiert auf eine möglichst schnelle Ausgabe der gesamten Ergebnismenge, während „FIRST_ROWS" einen Plan für eine möglichst schnelle Ausgabe der ersten Sätze erstellt. Grundsätzlich durchdenkt der Optimizer alle theoretisch möglichen Ausführungspläne und entscheidet sich dann für den vermeintlich besten.

Abbildung 2 stellt schematisch den „Denkprozess" des Optimizer dar. Alle theoretisch möglichen Ausführungspläne müssen zwischen „t0" und „tx" evaluiert sein. Jeder Ausführungsplan wird bewertet, das Ergebnis ist eine Zahl (die Kosten). Der Plan mit den niedrigsten Kosten wird verwendet. Es wird also angenommen, dass der günstigste Plan auch der beste ist (Discounter-Prinzip).

Je komplexer ein SQL-Statement ist, desto mehr theoretische Ausführungspläne kann es geben. Da die Parsing-Phase dadurch sehr lange laufen kann, wird nach einer kurzen Zeit („te") abgebrochen und der bis dahin beste (günstigste) Ausführungsplan verwendet. Grundsätzlich kann man sagen: Je komplexer ein SQL formuliert ist, desto höher wird die Gefahr schlechter oder kippender Ausführungspläne, da „te" fix ist. *Abbildung 3* zeigt an einem trivialen Beispiel eines Join über zwei Tabellen, wie viele Ausführungspläne möglich sein können.

statistikbasierter Optimizer (Denkprozess)

SELECT X.b, Y.b FROM X, Y WHERE X.a = Y.a;

- Annahme
  - beide Tabellen haben jeweils ein Index auf der Join-Spalte

Möglichkeit (m)	Anzahl (a)	Gesamt (m*a)
Reihenfolge	2	2
FTS vs Index	4	8
Methode	3	24

- Möglichkeiten bei einem Join über 10 Tabellen?

Abbildung 3: Möglichkeiten bei einem Join über zwei Tabellen

Statistiken

Das PL/SQL-Package „DBMS_STATS" generiert und verwaltet Statistiken für den kostenbasierenden Optimizer. Diese können sowohl im Data Dictionary als auch im Benutzerschema gespeichert sein. Mit diesem Paket können unter anderem folgende Funktionen ausgeführt werden:

- Parallele Generierung von Statistiken
- Erstellung individueller Statistiken
- Erstellen und Löschen von benutzerdefinierten Statistiktabellen
- Austausch von Statistiken zwischen dem Benutzerschema und dem Data Dictionary
- Austausch von Statistiken zwischen Datenbanken

- Sperren von Statistiken
- Wiederherstellung von Statistiken
- Parametrierung der Sammlung von Statistiken

Entscheidend ist hier die konzeptionelle Frage, wann und in welcher Tiefe die Statistiken aktualisiert werden sollen. Grundsätzlich findet man bei den Administratoren zwei verschiedene Vorgehensweisen zum Sammeln von Statistiken. Da neue Statistiken immer die Gefahr von kippenden Ausführungsplänen mit sich bringen, werden auf einigen Datenbanken keine neuen Statistiken erzeugt. Die Pläne sind dann stabil.

Allerdings birgt diese Methode die Gefahr der schleichenden Verschlechterung, da auf veränderte Bedingungen nicht reagiert werden kann. Die meisten Administratoren sammeln daher regelmäßig. Seit Version 9i bietet Oracle hier ein Verfahren an, in dem diejenigen Tabellen mit neuen Statistiken versorgt werden, bei denen mehr als 10 Prozent der Sätze verändert worden sind. Ab 10*g* ist dafür ein automatisierter Task hinterlegt *(siehe Abbildung 4)*.

Seit Version 11*g* kann man individuell je Tabelle die Anforderung an die Art der Statistikerstellung festlegen. Zu diesem Zweck wurden die sogenannten „Preferences" eingeführt:

- *cascade*
  Definiert, ob bei einer neuen Tabellenstatistik auch die an der Tabelle hängenden Indizes mit gepflegt werden sollen
- *degree*
  Grad der Parallelität
- *estimate_percent*
  Definition der Stichprobe in Prozent
- *method_opt*
  Umgang mit Histogrammen
- *no_invalidate*
  Definiert, ob offene Cursor invalidiert werden sollen
- *granularity*
  Umgang mit Partitionen
- *publish*
  Zeitpunkt der Veröffentlichung
- *incremental*
  Globale Statistiken auf partitionierten Tabellen
- *stale_percent*
  Definiert den Grad, wann eine Tabelle als „STALE" definiert wird

Jeder Administrator sollte für jede Datenbank ein klares Konzept haben, wie und in welchen Intervallen Statistiken erzeugt werden.

Parameter

Die Parameterdatei („init.ora"/„spfile") wird grundsätzlich beim Starten der Instanz eingelesen. In der Version 12*c* gibt es 366 offizielle Parameter, die fast alle modifizierbar sind. Allerdings ist bei 115 Parametern die Instanz neu durchzustarten. Etwa 25 bis 30 Parameter haben einen Einfluss auf den „Denkprozess" des Optimizer, *Tabelle 1* zeigt die wichtigsten. Anpassungen an diesen Parametern sollten nicht spontan vorgenommen werden, eine Veränderung muss jeweils gut getestet sein.

NAME	VALUE
optimizer_features_enable	12.1.0.1
optimizer_mode	ALL_ROWS
optimizer_index_cost_adj	100
optimizer_index_caching	0
optimizer_dynamic_sampling	2
optimizer_secure_view_merging	TRUE
optimizer_use_pending_statistics	FALSE
optimizer_capture_sql_plan_baselines	FALSE
optimizer_use_sql_plan_baselines	TRUE
optimizer_use_invisible_indexes	FALSE

Tabelle 1

MBRC

Bei vielen Administratoren gibt es eine Unsicherheit, inwiefern der Parameter „db_

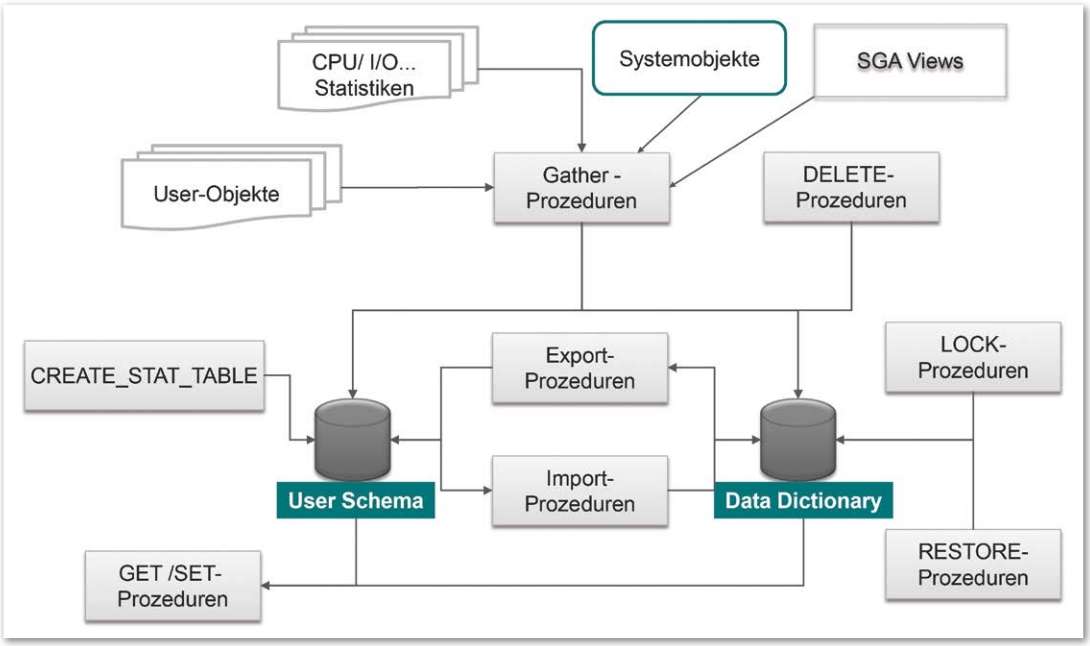


Abbildung 4: Überblick „dbms_stats"

file_multiblock_read_count“ noch wirkt, seit es in den Systemstatistiken den Parameter „MBRC“ gibt. Häufig höre ich die Meinung, dass der Parameter aus der Datei „spfile“ („db_file_multiblock_read_count“) obsolet ist und lediglich der interne Wert aus den Systemstatistiken greift. Hier liegt ein klares Missverständnis vor. Anhand einer kleinen Testreihe kann man die Wirkungsweise verdeutlichen. Grundsätzlich gilt: Je höher „MBRC“ gesetzt ist, desto günstiger wird ein Full Table Scan und desto weniger werden Indizes genutzt *(siehe Abbildung 5)*. Die Testreihe zeigt, dass der Parameter „MBRC“ aus den Systemstatistiken als Input für den Optimizer dient, während der Wert aus „db_file_multiblock_read_count“ bei der realen Durchführung zum Einsatz kommt.

dbfmrc	mbrc	Kosten	Durchführung
1		33924	1
64		6256	64
1	32	6696	1
64	32	6696	64

- Systemstatistiken (mbrc)
  - Kalkulation des Optimizers
- Parameter (db_file_multiblock_read_count)
  - Kalkulation des Optimizers (falls keine Systemstatistiken vorhanden)
  - reale Durchführung

Abbildung 5: Testreihe „MBRC“

Gespeicherte Ausführungspläne

Ist ein SQL-Statement sehr komplex oder ein Ausführungsplan nicht stabil, kann dieser in der Datenbank hart verdrahtet werden. Dafür bietet Oracle drei Möglichkeiten: Stored Outlines, SQL Profiles und SQL Baselines. Diese Varianten sind nicht in allen Oracle-Versionen verfügbar und unterstützt. Zudem ist die Nutzung teilweise kostenpflichtig.

Stored Outlines

Mithilfe der Stored Outlines hat man die Möglichkeit, Ausführungspläne in der Datenbank fest zu speichern. Damit ist gewährleistet, dass man unabhängig vom momentanen Stand der Statistiken immer mit dem gleichen Ausführungsplan arbeitet. Alle Outlines gehören zudem dem User „OUTLN“. Mit Version 10*g* sind die Outlines offiziell aus der Wartung genommen worden – aber intern noch nutzbar.

SQL Profile

Der Optimizer durchdenkt alle theoretisch möglichen Ausführungspläne und errechnet auf Basis der Input-Daten (Parameter und Statistiken) den vermeintlich optimalen Plan. Ist ein SQL nun sehr komplex, so geht die Anzahl der möglichen Ausführungspläne gegen unendlich. Damit die Parsing-Phase nicht unnötig lang wird, bricht der Optimizer den „Denkprozess“ nach einigen Millisekunden ab und es wird der bis zum Abbruch günstigste Plan verwendet. Dieser kann unter Umständen nicht der optimale Plan sein. Im erweiterten Modus (Tuning Mode) durchläuft der Optimizer folgende Tests:

- Statistik-Analyse
- SQL Profiling
- Access-Path-Analyse
- SQL-Structure-Analyse

Der Tuning-Optimizer prüft zunächst, ob Statistiken vorhanden und ob diese Statistiken auch aktuell sind. Stehen keine aktuellen Statistiken zur Verfügung, sammelt der Optimizer diese während der Optimierung. Dies kann dann natürlich länger dauern.

Beim Profiling führt der Optimizer zahlreiche Tests durch. Hierzu gehören beispielsweise die Analyse verschiedener Optimizer-Modi („all_rows“ vs. „first_rows“) sowie das Sampling von Daten. Weiterhin vergleicht der Optimizer unterschiedliche Ausführungspläne bezüglich der Kosten. Akzeptiert der Benutzer das vorgeschlagene Profil, so wird dieses in der Datenbank gespeichert und für spätere Ausführungen genutzt.

SQL Baselines

Bei Verwendung dieser Methode werden die Ausführungspläne in gute und schlechte Pläne unterteilt. Es wird somit garantiert, dass keine als schlecht markierten Pläne mehr ausgeführt werden. Darüber hinaus können Baselines als Fixum erstellt werden. Neue Ausführungspläne werden dann immer an dieser fixierten Basis gemessen und bewertet.

Fazit

Der Optimizer ist eine sehr komplexe Komponente, auf die ein DBA über viele Stellschrauben einwirken kann. Grundsätzlich sollten folgende Verhaltensregeln eingehalten werden:

- Es muss ein Konzept zur Statistikerzeugung vorliegen
- Statistiken müssen aktuell gehalten werden
- Parameter stabilisieren
- Anpassungen intensiv vor Veröffentlichung testen
- Statements nicht zu komplex formulieren

Zum Schluss soll noch eine Lanze für den Optimizer gebrochen werden. Es ist immer wieder erstaunlich, was ihm so alles vor die Füße geworfen wird und wie häufig er daraus einen guten Plan generiert. Die wenigen Fälle, in denen er daneben greift, führen dann aber zu Unmut bei den Anwendern. Der Optimizer ist von Version zu Version stabiler geworden, einzelne Bugs kommen jedoch immer mal vor.

Klaus Reimers
info@ordix.de

Tipps & Tricks aus Gerds Fundgrube

Best Practice Layout-Editor

Gerd Volberg, OPITZ CONSULTING Deutschland GmbH

In der guten alten Zeit, als man innerhalb des Layout-Editors noch im Character-Modus arbeitete, war es ein Leichtes, schnell und effizient Formulare zu erzeugen und zu designen.

Das Verschieben von Objekten war damals nur innerhalb eines „80*25"-Rasters möglich. Dieses grobe Raster sorgte dafür, dass Entwickler sehr schnell Objekte ausrichten konnten und nicht millimetergenau immer wieder prüfen mussten, ob alle Objekte zueinander korrekt gelayoutet waren. Der Nachteil lag auf der Hand und war auch der Grund dafür, dass Masken in Oracle Forms nicht immer gut aussahen, sondern eher praktikabel und effizient waren. Seit mehr als zehn Jahren gibt es nun auch den Real-Koordinaten-Modus, der gleich mit fünf Subtypen an den Start geht, die dem Entwickler die Speicherung der Objekt-Koordinaten in Pixel, Punkt, Dezimalzeichen, Zentimeter oder Zoll gestatten.

Jeder Subtyp hat seine eigenen Vor- und Nachteile. Die folgende Anleitung beschreibt, wie man im Real-Koordinaten-Modus wieder genauso effizient arbeiten kann wie im früheren Character-Modus. Zuerst setzt man auf Forms-Ebene das „Koordinatensystem" auf den Wert „Einheit" und die Property „Einheit" auf den Wert „Punkt" *(siehe Abbildung 1)*. Die Werte in den Feldern „Zeicheneinheit" sind „6" und „18". Im nächsten Schritt werden der Layout-Editor geöffnet und im Menüpunkt „Ansicht" unter „Lineal/Raster anpassen..." die Formatzeilen-Einstellungen gesetzt, wie in *Abbildung 2* zu sehen ist.

Mit diesen Voreinstellungen lässt sich nun im Koordinatensystem des Layout-Editors in einem Drei-Punkte-Grid arbeiten. Wenn man also einen Button anklickt und dann mit der Tastatur verschiebt, wirkt sich jede Verschiebung immer drei Punkte horizontal oder vertikal aus *(siehe Abbildung 3)*. Gleiches gilt für den Einsatz der Maus. Wenn der Menüpunkt „An Raster ausrichten" aktiviert ist, wird jedes Objekt an einer durch drei teilbaren X- oder Y-Koordinate ausgerichtet.

Diese Voreinstellungen sollte man in einem Forms-Template hinterlegen, das jeder Entwickler bei seinen Arbeiten benutzt. Auf diese Weise ist sichergestellt, dass das Layouten von Formularen maximal effizient ist. Links- oder rechtsbündige Ausrichtungen können dann direkt von Hand gemacht werden, ohne dass man eine aufwändige, pixelgenaue Nachbearbeitung vornehmen muss.

Gerd Volberg
gerd.volberg@opitz-consulting.com
talk2gerd.blogspot.com

Abbildung 1: Character-Modus vs. Real-Koordinaten-Modus

Abbildung 2: Character-Modus vs. Real-Koordinaten-Modus

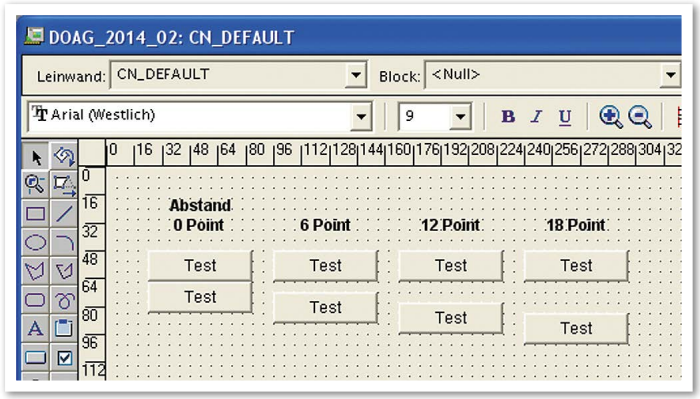
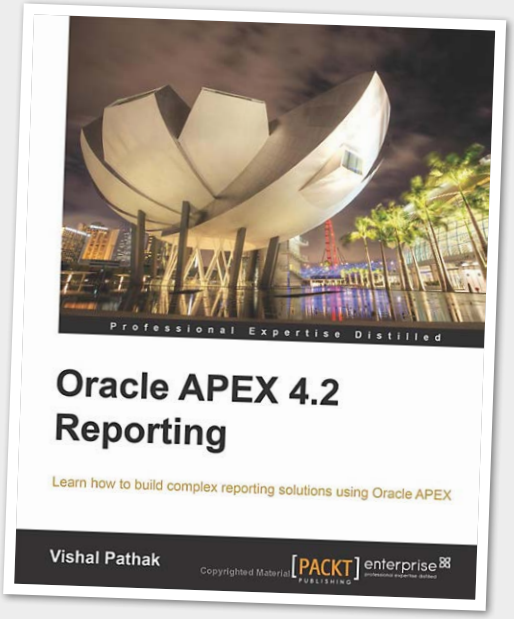


Abbildung 3: Abstände zwischen Objekten im 3-Punkte-Raster

Oracle Apex 4.2 Reporting

gelesen von Oliver Lemm



Innerhalb von Oracle Application Express stellt das Reporting einen großen Schwerpunkt dar und dazu hat Vishal Pathak das Buch „Oracle Apex 4.2 Report" geschrieben. Das Buch richtet sich an Apex-Architekten sowie -Entwickler, die schon Erfahrung besitzen. Der Einstieg in das Buch erfolgt über die Installation und Konfiguration der verschiedenen Apex-Gateways. Hier sind das Embedded PL/SQL Gateway (EPG), der Oracle HTTP Server (OHS) sowie der Apex Listener (mittlerweile als Oracle REST Data Services bekannt) beschrieben. Wer sich aufs Reporting konzentriert und auch die URL-Syntax innerhalb von Apex beherrscht, kann das Kapitel problemlos überspringen.

In den nachfolgenden Kapiteln sind nun verschiedene Ansätze, Methoden und unterstützende Tools im Bereich des Reportings erläutert. Kapitel 2 behandelt sehr viele unterschiedliche Themen, die sich vor allem auf die Darstellung des Reportings konzentrieren: Apex Standard Features, Anpassungen bezüglich CSS und Templates sowie die Nutzung von Dynamic Actions. Zusätzlich sind Environment-Variablen, Pivot-Funktion, Security-Aspekte und Up-/Download-Funktionalitäten in Apex erläutert. Die Themen werden im Einzelnen kurz vorgestellt, wobei das Kapitel insgesamt etwas unübersichtlich ist und auch die Themen inhaltlich nicht gut aufgeteilt wirken.

Das nächsten Kapitel zeigt die Interactive Reports, die als eine sehr mächtige Komponente in Apex schon „out of the Box" für viele Reporting-Funktionalitäten vom Endbenutzer angepasst werden können. Inhaltlich ist das Kapitel gut abgegrenzt und zeigt viele Möglichkeiten, wie man mit den Interactive Reports umgehen kann.

Die Kapitel 4 bis 6 heben verschiedenste Tools und Möglichkeiten hervor, die nicht Bestandteil von Apex sind beziehungsweise als externe Tools eingebunden werden können. Die drei Kapitel hätten an dieser Stelle etwas anders strukturiert werden sollen, da die Zusammenstellung etwas wahllos wirkt. Auch tauchen hin und wieder Apex-interne Komponenten wie Websheets auf, die innerhalb dieser Kapitel etwas fehl am Platz sind.

Das siebte Kapitel greift das Thema „Apex mit OBIEE" auf, was in Deutschland nicht so stark fokussiert wird, aber durchaus in anderen Regionen wie den USA ein zentrales Thema darstellt. Zuletzt wird dabei auch noch kurz auf den BI Publisher eingegangen.

Kapitel 8 konzentriert sich auf Integration sowie Webservices und hilft somit, wenn die Daten auch über Schnittstellen oder von weiteren Systemen verarbeitet werden sollen. Zuletzt wird innerhalb des neunten Kapitels die Performanz bzgl. Reports genauer erläutert. Die Tipps zu PL/SQL, SQL, HTML, aber auch zur Datenbank-Konfiguration sind durchaus hilfreich, wenn das Reporting läuft, jedoch zu langsam ist.

Fazit

Insgesamt ist auf den über 400 Seiten eine enorme Menge an Informationen vorhanden. Leider sind die Struktur im Buch sowie die Zusammenstellung der Kapitel nicht immer optimal ausgefallen. Es hätte dem Buch sicherlich gut getan, die Features, die Apex mitbringt, und die Features, die man durch Anpassungen innerhalb Apex, PL/SQL, SQL, HTML und CSS erreichen kann, von den zusätzlichen Produkten zu trennen. Wer ein Buch sucht, das zahlreiche Themen im Bereich „Reporting" abdeckt, kann es dennoch sehr gut nutzen, um gezielt einige Themen nachzulesen oder sich einen Gesamtüberblick zu verschaffen.

Titel:	Oracle APEX 4.2 Reporting – Learn how to build complex reporting solutions using Oracle APEX
Autor:	Vishal Pathak
Verlag:	Packt Publishing
Umfang:	409 Seiten + eBook
Preis:	42,99 Euro
ISBN:	978-1-84968-498-9
Website:	www.packtpub.com/oracle-apex-4-2-reporting/book

Oliver Lemm
oliver.lemm@mt-ag.com

Oracle Hidden Secrets:

Automatisch immer sauber lizenziert in Enterprise Manager Cloud Control

Manuel Hoßfeld, Oracle Deutschland B.V. & Co. KG

Viele Funktionen von Oracle Enterprise Manager 12*c* Cloud Control sind als sogenannte „Management Packs" separat zu lizenzieren, und zwar in Abhängigkeit von den jeweiligen Zielen. Wenn also zum Beispiel nicht für alle Datenbank-Targets Lizenzen für ein bestimmtes Pack vorliegen, ist es Aufgabe des Administrators, den Zugriff auf die entsprechenden Funktionen zu unterbinden. Hier geht es darum, wie man dies in Cloud Control einstellen und automatisieren kann.

Hinweis: Zur besseren Vergleichbarkeit mit der Dokumentation werden in diesem Artikel die englischsprachigen Bezeichnungen und Menüeinträge verwendet. Die Sprache der Cloud-Control-Oberfläche lässt sich jederzeit über die Spracheinstellung im eigenen Browser ändern.

Einigen Cloud-Control-Nutzern dürfte es bereits bekannt sein, dass über die Seite „Management Pack Access" (erreichbar über das Menü „Setup" -> „Management Packs") verschiedenen vorhandenen Enterprise-Manager-Targets gezielt der Zugriff auf die Management-Packs erteilt oder entzogen werden kann – und zwar entweder einzeln oder mit der Funktion „Pack based batch update" gleich für alle Targets eines bestimmten Typs.

Beide Möglichkeiten greifen aber nur für bereits bestehende Cloud-Control-Targets. Was passiert jedoch mit neuen Targets, die in Zukunft hinzugefügt werden? Genau diese Frage beantwortet das „Auto Licensing". Damit kann festgelegt werden, ob bestimmte Management-Packs für neue Enterprise-Manager-Ziele automatisch als lizenziert gelten oder nicht.

Auto Licensing einstellen

Aktiviert man auf der genannten „Management Pack Access"-Seite den Button „Auto Licensing", öffnen sich darunter zwei Auswahllisten: Die linke Liste zeigt die zur Verfügung stehenden Packs; von ihr aus können mithilfe der Knöpfe in der Mitte ein oder mehrere Packs in die rechte Liste befördert werden. Die Packs in der rechten Liste sind dann diejenigen, für die das „Auto Licensing" aktiviert ist, sobald man unten rechts den „Apply"-Knopf betätigt. Damit hierbei aber überhaupt etwas geschieht, ist es wichtig,

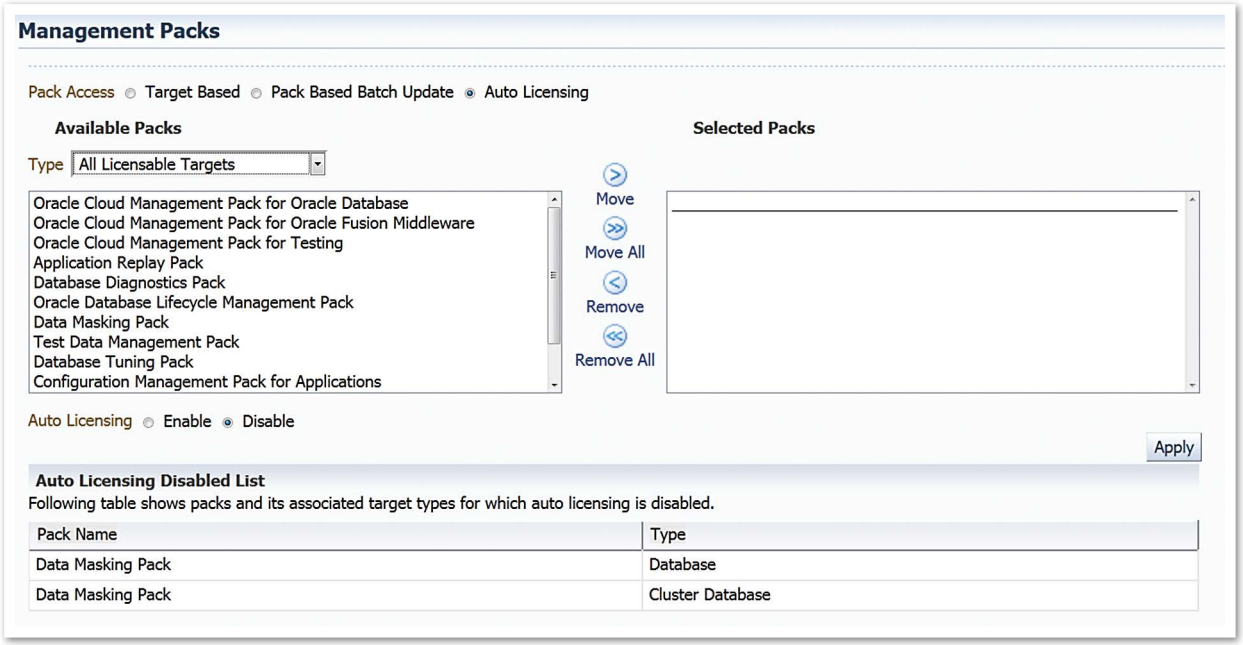


Abbildung 1: Auto-Licensing-Einstellungen in Enterprise Manager 12c Cloud Control

dass der Default auf „Auto Licensing: Enable“ steht.

Erst, wenn man unter der Auswahlliste vor dem Klick auf „Apply“ die entsprechende Auswahl auf „Disable“ ändert, wird die getroffene Auswahl auch tatsächlich auf die Liste der in Zukunft nicht mehr automatisch als lizenziert geltenden Packs gesetzt. Letztere lässt sich auch später jederzeit im unteren Bereich der Seite in der „Auto Licensing Disabled List“ einsehen. In der in *Abbildung 1* gezeigten Umgebung ist exemplarisch das Data-Masking-Pack auf diese Liste gesetzt.

Solange diese Einstellung bestehen bleibt, würde in Zukunft also für alle neuen Datenbank-Targets das Data-Masking-Pack immer automatisch als nicht lizenziert eingestellt werden – ohne dass der DBA dafür noch einmal Hand anlegen müsste.

Manuel Hoßfeld
manuel.hossfeld@oracle.com

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

Kai Liesenfeld
Tobias Höhn
Vishakha Bujone
Michael Seliger
Marcus Mönnig
Lars Wiedenhöft
Markus Paff
Waheeda Goerlitz
Alireza Bagherpour Tehrani
Christina Veit
Frank Thielen
Maryna Nazarova
Till Brügelmann
Lucas Prochnow
Serbülent Aki
Felix Michels

Firmenmitglieder

Marquardt Service GmbH, Yongzhen Ou
numetris AG, Björn Berg
Deutsche Edelstahlwerke GmbH, Isabella Drzazga
ZfP Emmendingen, Tobias Fiedler
Alte Leipziger Lebensversicherung, Gernot Grünsteidl

Aus dem Verein

Dr. Dietmar Neugebauer
Vorstandsvorsitzender der DOAG

Die DOAG auf der Collaborate 14

Anfang April fand in diesem Jahr wieder eine der größten Oracle-Anwenderkonferenzen der Welt statt. Organisiert wurde die Collaborate 14 von den großen amerikanischen Usergruppen Independent Oracle Users Group (IOUG), Oracle Applications Users Group (OAUG) und Quest International Users Group. In Las Vegas trafen sich mehr als 5.500 Teilnehmer, denen ein Konferenzprogramm mit rund 1.000 Vorträgen und eine große Ausstellung geboten wurden. Adam Savage, bekannt aus der Fernsehserie „MythBusters – Die Wissensjäger" und durch seine Mitwirkung bei den Filmeffekten in Star Wars I und II sowie Terminator 3, eröffnete die Veranstaltung mit einer Keynote. Er vermittelte mit diesem gelungenen Einstieg zur Konferenz der überwiegend technisch orientierten Zuhörerschaft, dass Kunst und Technik keinesfalls zwei sich ausschließende Themen sind, sondern zeigte, wie sich beide ergänzen und gegenseitig befruchten können.

Die DOAG war auf der Konferenz durch drei Vorstandsmitglieder vertreten: Dr. Dietmar Neugebauer, Dr. Frank Schönthaler und Fried Saacke. Trotz der hohen Hürde bei den Auswahlkriterien der angenommenen Vorträge gelang es auch drei DOAG-Mitgliedern, als Referenten eingeladen zu werden: Martin Klier und Johannes Michler wurden von der IOUG ausgewählt und Dr. Frank Schönthaler von der OAUG.

Die DOAG-Vertreter nutzten die Konferenz, um ihre bestehende Zusammenarbeit mit den internationalen Usergruppen durch persönliche Kontakte aufzufrischen. Gemeinsam mit den anwesenden Usergruppen und Vertretern von Oracle wurden die nächsten Aktivitäten der internationalen Zusammenarbeit besprochen. Darüber hinaus war es auch möglich, neue Kontakte zu Referenten zu knüpfen, um diese als Sprecher für die Veranstaltungen der DOAG anzuwerben. Dabei ist es immer wieder schön zu hören, welchen Bekanntheitsgrad die DOAG inzwischen auf der internationalen Ebene hat und dass eine Vielzahl von Referenten wieder gerne zu unseren Konferenzen kommen möchte.

Urban Lankes
Stellv. Vorstandsvorsitzender

IT-Nachwuchs fördern

Studenten leiden bekanntlich unter einem chronisch schmalen Budget. Um ihnen den Zugang zu Oracle-bezogenem Fachwissen zu erleichtern, hat die DOAG das bestehende Studentenprogramm der Jahreskonferenz um das „DOAG Student-Sponsorship-Programm" (S3) erweitert. Das Prinzip ist denkbar einfach: Unternehmen übernehmen zukünftig Reise- und Übernachtungskosten von Studierenden und fördern somit aktiv den IT-Nachwuchs im Oracle-Umfeld. Die Sponsoren werden ab jetzt gesucht.

80 Studenten haben im vergangenen Jahr am Studentenprogramm teilgenommen und in diesem Rahmen die DOAG 2013 Konferenz + Ausstellung besucht. Für die heranwachsenden Informatiker sind sowohl das dreitägige Konferenzticket als auch der anschließende Schulungstag kostenfrei. Im Gegenzug unterstützen sie die DOAG in der Konferenzwoche mit zwölf Stunden Arbeitszeit, indem sie beispielsweise das Konferenzmaterial konfektionieren, beim Aufbauen helfen, die Ticketkontrolle am Einlass übernehmen oder die Referenten betreuen.

Das Programm stößt seit Jahren auf sehr positive Resonanz: Viele Studenten haben sich bereits im Anschluss an die DOAG 2013 Konferenz + Ausstellung für die nächste Auflage angemeldet. Auch die Universitätsprofessoren bleiben mobilisiert. Einziger Wermutstropfen sind bisher die Reise- und Hotelkosten, für die die Studierenden selber aufkommen. Das nötige Geld für den Aufenthalt in Nürnberg aufzutreiben, gestalte sich laut Aussagen der Studenten, Studentinnen und Professoren oftmals als schwierig.

Das soll das „DOAG Student-Sponsorship-Programm" (S3) ändern: Die 80 Teilnehmer des Studentenprogramms sollen eine Finanzspritze von 200 Euro pro Person erhalten. Um diese Idee zu verwirklichen, sucht die DOAG ab jetzt Sponsoren. Wer sich angesprochen fühlt, kann sich schon per E-Mail an „hochschule@doag.org" wenden.

Auch für Firmen ist das S3 interessant: Mit diesem überschaubaren Obolus leisten sie nicht nur einen sozialen Beitrag; sie fördern auch den Nachwuchs, der möglicherweise im eigenen Unternehmen anfangen könnte. Als erste deutschlandweite Anlaufstelle im Oracle-Umfeld verbindet die DOAG Konferenz + Ausstellung mit dem Studentenprogramm und S3 mehr denn je Arbeitgeber und künftige Arbeitnehmer. Zum Programmumfang gehört deswegen ein Treffen zwischen Studenten und Sponsoren, das auf der DOAG-Konferenz stattfinden wird.

02.06.2014
Regionaltreffen NRW (Vorabend DB Fachkonf.)
Stefan Kinnen, Andreas Stephan
regio-nrw@doag.org

03.06.2014
DOAG 2014 Datenbank
Johannes Ahrends, Christian Trieb
sig-database@doag.org

03.06.2014
Regionaltreffen NRW (Vorabend Dev Fachkonf.)
Stefan Kinnen, Andreas Stephan
regio-nrw@doag.org

04.06.2014
Regionaltreffen Berlin/Brandenburg
Michel Keemers
regio-bb@doag.org

04.06.2014
DOAG 2014 Development
Andreas Badelt, Christian Schwitalla
sig-development@doag.org

05.06.2014
SIG Middleware
Jan-Peter Timmermann, Hajo Normann, Torsten Winterberg
sig-middleware@doag.org

05.06.2014
Regionaltreffen München/Südbayern
Franz Hüll, Andreas Ströbel
regio-muenchen@doag.org

05./06.06.2014
OUGF 2014 Harmony Conference
http://ougf-harmony.eventbrite.com/

10./11.06.2014
Berliner Expertenseminar mit Felix Krul zum Thema Komplexe Fragestellungen im Oracle-Data Warehouse
Cornel Albert
expertenseminare@doag.org

12.06.2014
Regionaltreffen Nürnberg/Franken
André Sept, Martin Klier
regio-franken@doag.org

12.06.2014
Regionaltreffen Bremen
Ralf Kölling
regio-bremen@doag.org

13.06.2014
DOAG Webinar: Maschinensizing
Johannes Ahrends, Christian Trieb
sig-database@doag.org

16./17.06.2014
Berliner Expertenseminar mit Jürgen Sieben zum PL/SQL-Performance-Tuning
Cornel Albert
expertenseminare@doag.org

16.06.2014
Regionaltreffen Halle/Leipzig
Matthias Reimann
regio-halle@doag.org

16.06.2014
Regionaltreffen Osnabrück/Bielefeld/Münster
Andreas Kother, Klaus Günther
regio-osnabrueck@doag.org

16.06.2014
Regionaltreffen Jena/Thüringen
Jörg Hildebrandt
regio-thueringen@doag.org

17.06.2014
AOUG Anwenderkonferenz 2014
http://www.aoug.at/Event/309

17.06.2014
Regionaltreffen Hamburg/Nord
Jan-Peter Timmermann
regio-nord@doag.org

24.06.2014
Regionaltreffen Rhein-Maim
Thomas Tretter
regio-rhein-main@doag.org

26.06.2014
Regionaltreffen Karlsruhe
Reiner Bünger
regio-karlsruhe@doag.org

26.06.2014
Regionaltreffen Dresden/Sachsen
Helmut Marten
regio-sachsen@doag.org

30.06.2014
SIG Oracle & SAP
Jörg Hildebrandt
sig-sap@doag.org

Weitere Termine und Informationen unter www.doag.org/termine/calendar.php

Impressum

Herausgeber:
DOAG Deutsche ORACLE-Anwendergruppe e.V.
Tempelhofer Weg 64, 12347 Berlin
Tel.: 0700 11 36 24 38
www.doag.org

Verlag:
DOAG Dienstleistungen GmbH
Fried Saacke, Geschäftsführer
info@doag-dienstleistungen.de

Chefredakteur (ViSdP):
Wolfgang Taschner, redaktion@doag.org

Redaktion:
Fried Saacke, Carmen Al-Youssef, Mylène Diacquenod, Dr. Frank Schönthaler, Dr. Dietmar Neugebauer, Urban Lankes, Christian Trieb

Titel, Gestaltung und Satz:
Alexander Kermas,
DOAG Dienstleistungen GmbH

Titelfoto: © Benis Arapovic / 123rf.com
Foto S. 9: © Eric Gevaert / 123rf.com
Foto S. 16: © ORACLE / Oracle.com
Foto S. 20: © bowie15 / 123rf.com
Foto S. 38: © lightwise / 123rf.com
Foto S. 61: © antonprado / 123rf.com
Foto S. 63: © ORACLE / Oracle.com

Anzeigen:
Simone Fischer, anzeigen@doag.org
DOAG Dienstleistungen GmbH
Mediadaten und Preise finden Sie unter: www.doag.org/go/mediadaten

Druck:
Druckerei Rindt GmbH & Co. KG
www.rindt-druck.de

Inserentenverzeichnis

ORACLE DATABASE 12c

EINLADUNG

Oracle European Launch Event

The Future of the Database Is Almost Here

17. Juni 2014, Frankfurt

Oracle Database In-Memory Option: Europa Launch in Frankfurt am 17. Juni

Sehr geehrte Leserin, sehr geehrter Leser,

seit mehr als 35 Jahren steht Oracle für Innovation im Bereich Datenbank. Dank unserer marktführenden Technologien sind Oracle Kunden in der Lage, ihrem Mitbewerb immer einen Schritt voraus zu sein. Jetzt können Sie ihren Vorsprung sogar noch schneller vergrößern.

Mit der neuen Oracle Database In-Memory Option profitieren Kunden von beschleunigter Datenbankleistung für Analytics, Data Warehousing, Reporting und Online Transaction Processing (OLTP).

Wäre es nicht revolutionär, wenn Sie Ihre Daten in Echtzeit abrufen könnten?

Wir laden Sie herzlich ein, am 17. Juni mit dabei zu sein, wenn Andy Mendelsohn, Head of Database Product Development, die neue, bahnbrechende Oracle Database In-Memory Option und weitere innovative Oracle Datenbank 12*c* Funktionen vorstellt.

Ihr Oracle Team

Sprecher

Andy Mendelsohn,
Head of Database Development,
Oracle

Registrieren Sie sich jetzt.
17. Juni 2014
Radisson Blu Hotel
Franklinstrasse 65
60486 Frankfurt am Main
www.oracle.com/goto/dbim/de

ORACLE

Gut zu wissen, dass es in der Firma läuft.

Gestalten Sie Ihr Leben sorgenfreier. Und Ihre IT leistungsfähiger. Denn wir haben das richtige Servicemodell für Sie. Von der Pflege und dem Support für Ihre Software und BI-Lösungen über den hochverfügbaren Betrieb Ihrer IT-Infrastruktur bis hin zu Outsourcing oder Cloud-Services. Immer effizient und innovativ. Trivadis ist führend bei der IT-Beratung, der Systemintegration, dem Solution-Engineering und bei den IT-Services mit Fokussierung auf Oracle- und Microsoft-Technologien im D-A-CH-Raum. Sprechen Sie mit uns. www.trivadis.com | info@trivadis.com

ZÜRICH ▪ BASEL ▪ BERN ▪ BRUGG ▪ GENF ▪ LAUSANNE ▪ DÜSSELDORF
FRANKFURT A.M. ▪ FREIBURG I.BR. ▪ HAMBURG ▪ MÜNCHEN ▪ STUTTGART ▪ WIEN

trivadis
makes IT easier.