Nr. 1 | Februar 2019 | ISSN 2366-7915 | www.doag.org

# Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP


## Keine Ende in Sicht

40 Jahre
Oracle Forms

## Im Interview

Dr. Axel Pols, Geschäftsführer
Bitkom Research

## DevOps und SaaS

Datenbank-Application-
Container

MUNIQSOFT
CONSULTING

IT-Consulting
& Support

# Mit **IQ** schaffen wir ein solides Fundament für Ihre Prozesse

## Mehr Power für Ihre Oracle Lösungen!

- Real Application Clusters 11.2 - 18c
- Datenbanken in der Oracle Cloud
- DB-Security in Zeiten der DSGVO (VPD, TDE, ASO)
- Upgrade auf Oracle 12.2 und 18c
- DB-Healthcheck / Performance-Tuning
- Oracle Enterprise Manager Cloud Control 13c
- Oracle Linux / Oracle VM
- Oracle Database Appliance (ODA)

Oracle IT-Consulting

Softwareentwicklung APEX und PL/SQL

Remote DBA Support (deutschsprachig)

Oracle Lizenz-management

Jetzt Beratungstermin vereinbaren:
**+49 89 6228 6789-21**

ORACLE Gold Partner
Specialized
Oracle Database

## NEUER NAME - BEWÄHRTER SERVICE

NEU

Die **Muniqsoft Consulting GmbH** konzentriert sich künftig auf Services für IT-Consulting und Support rund um Ihre Oracle Datenbank Lösungen.

**www.muniqsoft-consulting.de**

*Wolfgang Taschner*
*Chefredakteur Red Stack Magazin*

# Liebe Mitglieder, liebe Leserinnen und Leser,

im März 2019 gehe ich nach einem erfüllten Berufsleben in Rente. Meine Nachfolge als Chefredakteur des Red Stack Magazin übernimmt Martin Meyer aus dem DOAG-Kommunikationsteam.

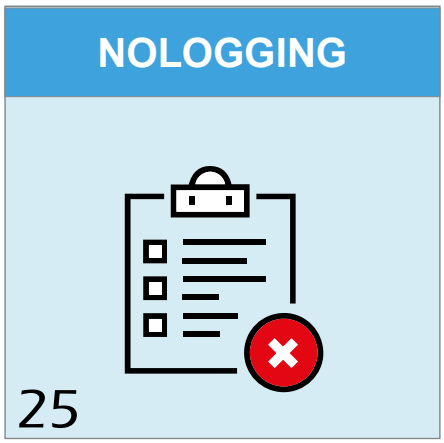
Als ich meine IT-Ausbildung begann, besaßen die Computer noch einen Magnetkernspeicher und wurden mit Lochkarten gefüttert. Lange ist es her … Schon damals habe ich erkannt, dass es hilfreich ist, sich mit seinen Kollegen auszutauschen, um eigenes Wissen weiterzugeben und von deren Erfahrungen zu profitieren. Dies war auch meine Motivation, nach dem Informatikstudium zunächst wichtige Berufserfahrungen als Software-Entwickler zu machen und dann in den Journalismus zu wechseln. Dort begann ich als Redakteur beim Computer-Magazin „Chip", hatte später eine eigene Zeitschrift „HC – Mein Home-Computer" und wechselte dann in die Selbstständigkeit, um Unternehmen wie Apple, Hewlett-Packard, NEC und Olivetti bei der PR-Arbeit zu unterstützen. Daneben war ich erfolgreicher Autor umfangreicher Shareware-Bücher beim Zweitausendeins-Verlag sowie zahlreicher Wander-, Mountainbike- und Reiseführer.

Im Jahr 2000 lernte ich Fried Saacke kennen, den damaligen Vorstandsvorsitzenden der DOAG. In freundschaftlicher Zusammenarbeit mit ihm war ich maßgeblich an der Entwicklung der DOAG-Webseiten und des DOAG-Newsletters beteiligt. Darüber hinaus habe ich die Pressearbeit der DOAG aufgebaut und zahlreiche Flyer sowie den jährlichen Geschäftsbericht verfasst und war als Chefredakteur für das Red Stack Magazin, die DOAG Business News sowie die Java aktuell verantwortlich.

Für ihre Geduld und Inspiration bedanke ich mich bei meiner Frau und unseren beiden Söhnen. Auch wenn das Leben manchmal schwer war, sind sie mir nie von der Seite gewichen.

Ihr


*Insbesondere bei der Installation der Datenbank 18c gibt es Neuerungen*




*Nologging Operations und Objects begegnen jedem Datenbank-Administrator*


*Das wichtigste Tool der Oracle-Datenbank ist auch heute noch viel im Einsatz*

## Einleitung



## Datenbank 18c



## Aktuell



## Datenbank



## Entwicklung



## Intern

# Timeline

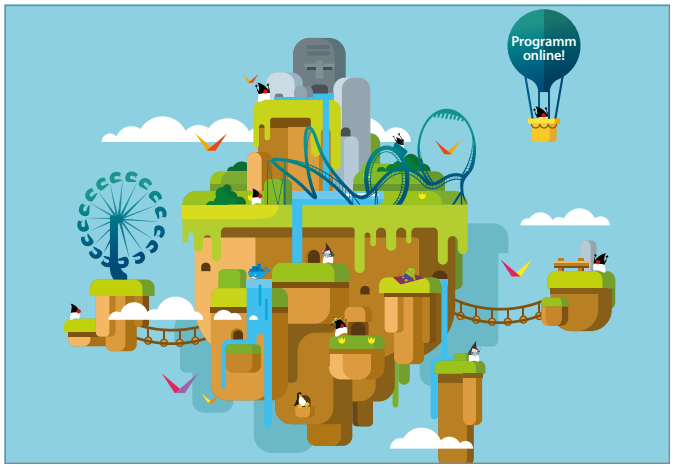
## 30. Oktober 2018

*Im schweizerischen St. Gallen fällt der Startschuss für das länderübergreifende Oracle-Beer-Meetup der Region Bodensee. Mit insgesamt sieben Teilnehmern aus der Schweiz und Liechtenstein, jedoch leider (noch) ohne Teilnehmer aus Deutschland und Österreich, wird mehr als vier Stunden über die Oracle OpenWorld diskutiert. Neben den technischen Neuigkeiten zu den Themen „Cloud", „Datenbanken", „Software Development" und „Engineered Systems" stehen diese Fragen im Raum: Wie ist die Stimmung in der Oracle-Community? Sieht man sich für die Herausforderungen der Zukunft gerüstet? Wird Oracle mit dem Cloud-Business überleben und wie sieht die Zukunft insgesamt aus – für Oracle, für Datenbanken, für die Cloud? Welche Rolle spielen Open Source und IBM (gab gerade den Red-Hat-Deal bekannt) in der Zukunft? Allgemeiner Konsens am Ende des Tages ist: Es gibt viele spannende Themen mit und um Oracle, sodass sich alle Teilnehmer schon auf das nächste Oracle-Beer-Meetup freuen – hoffentlich dann auch mit Teilnehmern aus Deutschland und Österreich.*

*Die Gründungsveranstaltung der Region Bodensee*

## 2. November 2018

*Das Organisationsteam der JavaLand 2019 gleicht in einer Abstimmungs-Telko den Status für die Veranstaltung fest. Schon jetzt zeichnet sich bei den eingehenden Anmeldungen ein deutliches Plus bei der Teilnehmerzahl ab. Man macht sich deshalb Gedanken, wie die erwarteten Besucherströme geschickt zu steuern sind.*




*Die Java-Community trifft sich vom 19. bis 21. März 2019 im Phantasialand in Brühl*

## 9. November 2019

*Rainier Kaczmarczyk, Senior Solution Architect bei Opitz Consulting, hält ein Webinar zum Thema „Dbvisit Standby – die hybride Lösung". Er demonstriert live das einfache Setup und stellt die Vor- und Nachteile der im Markt etablierten Hochverfügbarkeitslösung für Oracle vor.*

## 16. November 2019

*Das Organisationsteam der DOAG Dienstleistungen GmbH trifft im Berliner Büro die letzten Vorbereitungen zur DOAG 2018 Konferenz + Ausstellung. Die Kisten sind gepackt und für den Transport nach Nürnberg vorbereitet.*

## 19. November 2018

*Am Vortag der DOAG 2018 Konferenz + Ausstellung treffen sich in Nürnberg 27 Teilnehmer zum EOUC Leader Summit. Die Vertreter der internationalen User Groups diskutieren die zukünftige Zusammenarbeit in Europa. Wichtige Punkte sind die in Deutschland bekannten Probleme mit den Oracle-Lizenzierungspraktiken, insbesondere in virtuellen Umgebungen, sowie die Qualität des Supports. Dabei wird klar, dass sich die Probleme europaweit ähnlich darstellen und die User Groups hier mehr geschlossen agieren sollten. Darüber hinaus werden die Fortsetzung des gemeinsamen Magazins ORAWORLD besprochen sowie die Zusammenarbeit im nächsten Jahr. Neben einem Treffen im Frühjahr, das in Riga stattfinden könnte, soll es wieder ein Treffen auf der DOAG Konferenz geben. Es besteht der Wunsch der EMEA User Groups, auf der DOAG Konferenz + Ausstellung 2019 einen eigenen Stream gestalten zu dürfen. Die Vertreter der DOAG sind nicht abgeneigt und nehmen das Thema mit.*


*Gute Stimmung beim EOUC Leader Summit*

## 20. November 2018

*Bei seiner Eröffnungsrede zur DOAG 2018 Konferenz + Ausstellung stellt der DOAG-Vorstandsvorsitzende Stefan Kinnen die Ergebnisse der DOAG-Erhebung zur Nutzung von Cloud Computing vor. Während Oracle mit der Cloud-Offensive 2016 einen transformativen Schritt eingeleitet hat, der vom Unternehmen als nahezu imperativer Meilenstein auf dem Weg mittelständischer Unter-*

*nehmen präsentiert wird, scheint die Bereitschaft zum Einsatz von Cloud Computing bei vielen Anwendern nicht so ausgeprägt wie zunächst erhofft. Diese Skepsis spiegelt sich nun auch an vielen Stellen der DOAG-Umfrage wider. Laut Stefan Kinnen ergibt sich aus den Ergebnissen der Cloud-Umfrage deutliches Verbesserungspotenzial für den Hersteller: „Die DOAG appelliert an Oracle, sich der sicherheitsrelevanten Kriterien am Standort Deutschland anzunehmen und die Kommunikation darüber sowie zu Preisen und Services mit den Kunden zu verbessern. So sind die Sicherheitsüberlegungen der alles entscheidende Faktor, mit dem die Entscheidung der meisten Nutzer für oder gegen Cloud Computing fällt. Ohne diesem Faktor oberste Priorität einzuräumen, wird es Oracle kaum möglich sein, Cloud Computing zeitnah breit zu etablieren – ganz gleich, wie stark die Unabwendbarkeit oder gar Omnipräsenz des Phänomens Cloud beschworen wird."*


*Der DOAG-Vorstandsvorsitzende Stefan Kinnen eröffnet die DOAG 2018 Konferenz + Ausstellung*

## 20. November 2018

*In der vom ehemalige DOAG-Vorstandsvorsitzenden und Ehrenmitglied Dr. Dietmar Neugebauer moderierten Podiumsdiskussion mit dem DOAG Legal Council auf der DOAG 2018 Konferenz + Ausstellung sprechen die Rechtsanwälte Dr. Thomas Thalhofer, Dr. Jan Bohnstedt, Carsten J. Diercks und Dr. Ivo Rungg mit Michael Paege, DOAG Competence Center Lizenzierung, über Lizenzierung und Business Practices, Verträge, allgemeine Vertragsbedingungen, Audit, Datenschutz und Lizenzmitgabe bei Ausgründungen. Intensiv wird die Tatsache diskutiert, dass die öffentlichen Updates für das Oracle-Java-JDK ab Ende Januar 2019 nur noch kostenpflichtig zur Verfügung stehen. Dabei beschäftigt die Zuhörer die Frage, inwieweit dies Probleme in den Firmen schaffen kann. Das DOAG Legal Council diskutiert über rechtliche Möglichkeiten, Oracle in dieser Angelegenheit zu begegnen, und äußert die Ansicht, dass es hier keine allgemeingültige Aussage gäbe und es auf den konkreten Einzelfall ankäme. Wer sich als DOAG-Mitglied für das vom Legal Council angefertigte Gutachten zu dem ausführlich besprochenen Thema „Lizenzierung der Oracle-Produkte in virtuellen Umgebungen" interessiert, kann dieses gegen eine Schutzgebühr von 250 Euro über „office@doag.org" bestellen. In zwei weitere Podiumsdiskussionen kommen die Themen „Vor- und Nachteile, Datenschutz, Pricing, Strategie, Regionalität etc. in der Cloud" und „Virtualisierung unter VMware" zur Sprache.*


*Das DOAG Legal Council auf der DOAG 2018 Konferenz + Ausstellung*

## 20. November 2018

*Im Rahmen der DOAG-Konferenz lädt die Austrian Oracle User Group (AOUG) auch dieses Jahr wieder alle österreichischen Teilnehmer zum traditionellen Österreicher Abend ein. Nach dem Sektempfang im Untergeschoss des Kongresszentrums durch den österreichischen Präsidenten und einigen Vorstandsmitgliedern der User Group geht es für alle mit einem angemieteten Reisebus in die Innenstadt von Nürnberg zum Restaurant „Alte Küch'n & Im Keller". Das Lokal hat sich für alle österreichischen Teilnehmer in den letzten Jahren zum urigen Stammlokal entwickelt. Bei Bier und gutem Essen tauscht man sich zu allgemeinen Themen sowie über aktuelle Dinge auf der DOAG-Konferenz aus, etwa die bevorstehenden Änderungen bei der Oracle-Java-Lizenzierung.*

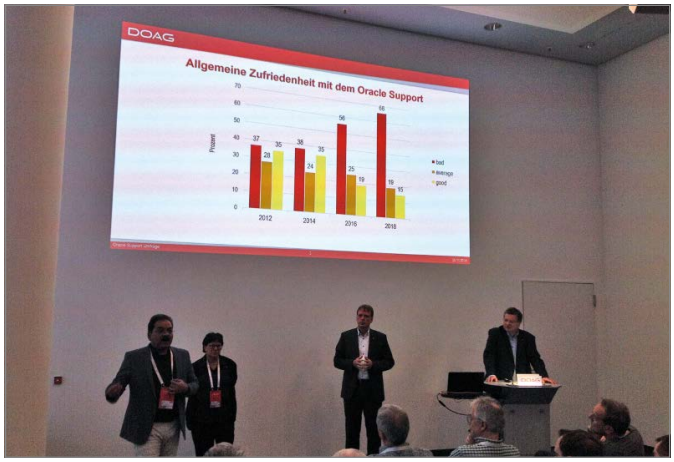
## 20. November 2018

*Auch auf der diesjährigen DOAG-Konferenz in Nürnberg findet sich ein kleiner, aber feiner Kreis von Teilnehmern aus der Schweiz zum traditionellen „Schweizer-Abend" zusammen und genießt beim gemütlichen Beisammensein fern der Heimat lokale Nürnberger Spezialitäten in Burg-Atmosphäre. Man lässt die Eindrücke des Tages auf sich wirken und tauscht sich über die informativen Sessions der DOAG aus. Bevor es mit dem Bus Richtung Restaurant losgeht, gibt es noch eine unerwartete Ergänzung im Programm. Die Überraschung der Teilnehmer ist groß, als sie beim Versammlungs-Apéro zum ersten Mal aktiv ihren Input zu den Themen der nächsten SOUG-Veranstaltungen bunt auf ein Flipchart malen dürfen. Der Swiss Oracle User Group (SOUG) ist es wichtig, die Bedürfnisse und Anliegen ihrer Anwender noch besser zu kennen, um so entsprechende Themen aufgreifen zu können und einen noch stärkeren Mehrwert für die User zu schaffen. Aufgrund des positiven Feedbacks der Teilnehmer darf in Zukunft öfter mit solchen Input-Sessions gerechnet werden.*

*Die Teilnehmer aus der Schweiz speisen in der Nürnberger Burg*

## 21. November 2018

*Im Rahmen der DOAG 2018 Konferenz + Ausstellung präsentieren Christian Trieb, Leiter des DOAG Competence Center Support, sowie der DOAG Vorstandsvorsitzender Stefan Kinnen die Ergebnisse der Oracle-Support-Umfrage 2018 und diskutierten das Ergebnis mit den anwesenden Oracle-Vertretern Clarissa Rohrmann (Customer Success Manager) und Pravin Anil Sheth (Director Database Supporting) sowie dem Publikum. Die wichtigsten Ergebnisse: Die Akzeptanz der Oracle-Proactive-Support-Tools steigt deutlich, während die allgemeine Zufriedenheit beim Support sinkt und die Cloud-Nutzung weiterhin gering ist (Details siehe Seite 42). DOAG-Vorstand und Geschäftsführer Fried Saacke resümiert, dass beim Oracle-Support zu sehr am Personal gespart wird, und stellt die Frage, ob Oracle dazu bereit sei, die Support-Mitarbeiter besser zu qualifizieren und dafür mehr Geld auszugeben. Er mahnt Oracle eindringlich, die Kritik am Support ernst zu nehmen. Clarissa Rohrmann von Oracle kündigt an, die präsentierten Ergebnisse der DOAG-Support-Umfrage dem Board of Directors vorzulegen und über eine Reaktion bis Ende des Jahres 2018 zu informieren.*




*Von links: Pravin Anil Sheth, Clarissa Rohrmann, Stefan Kinnen, Christian Trieb*

## 21. November 2018

*Im Anschluss an die Keynote des Buchautors Lars Vollmer zum „Zurück an die Arbeit – Wie aus Business-Theatern wieder echte Unternehmen werden“, der zeigt, was in den Unternehmen falsch läuft und warum, bietet der traditionelle Community-Abend mit Themen-Buffets und Live-Musik viele Gelegenheiten für interessante Gespräche.*

*Lasershow am Ende des Abendevents*

## 23. November 2018

*Die DOAG 2018 Konferenz, größte Oracle-Konferenz in Europa, geht mit dem Schulungstag zu Ende. Fried Saacke, Vorstandsmitglied und Leiter der DOAG-Geschäftsstelle, zieht ein positives Fazit: „Die Signalwirkung der Veranstaltung reicht über Europa hinaus bis in die Oracle-Headquarters.“*

## 29. November 2018

*Das Organisationsteam der DOAG-Geschäftsstelle hält in Berlin eine Feedback-Runde zur DOAG 2018 Konferenz + Ausstellung. Alle eingegangenen Kommentare und Vorschläge werden bewertet und daraus bereits die Weichen für eine erfolgreiche Jahreskonferenz 2019 gestellt.*

## 1. Dezember 2018

*Die zwölfte Ausgabe des EOUC E-Magazine ORAWORLD erscheint. Sie widmet sich dem Thema „Pair Programming“ und beschäftigt sich mit der Oracle-Datenbank 18c On-Premises und in der Cloud (Teil 2). Neben dem Oracle-Cost-Based-Optimizer (CBO) stehen auch Roboter-Vielfalt sowie viele weitere spannende Geschichten im Fokus. Der unter der Regie der DOAG von den European Oracle User Groups herausgegebene, englischsprachige Newsletter wird von Tausenden Anwendern in Europa und in den USA gelesen.*


*Das E-Magazine ORAWORLD, Ausgabe 12/2018*

## 6. Dezember 2018

*Der DOAG-Vorstand trifft sich vor seiner Sitzung mit den Mitarbeiterinnen und Mitarbeitern der DOAG-Geschäftsstelle sowie den wichtigsten Geschäftspartnern zum Jahresabschluss. Fried Saacke, DOAG-Vorstand und Geschäftsführer, betont das Potenzial von mehr Gemeinsamkeit zwischen Vorstand und Geschäftsstelle und appelliert, neben der täglichen Arbeit vor allem auch strategische Fragen noch mehr gemeinsam zu gestalten. Er und der DOAG-Vorstandsvorsitzende Stefan Kinnen danken allen für die tolle Arbeit im Jahr 2018.*

## 7. Dezember 2018

*Der DOAG-Vorstand kommt in Berlin zu seiner letzten Sitzung im Jahr 2018 zusammen und plant die Aktivitäten für das Jahr 2019.*

## 7. Dezember 2018

*Bei einem Technischen Frühstück der Austrian Oracle User Group (AOUG) in Wien geht es um „Java Lizenz-Update – Könnten Ihre Java-Installationen kostenpflichtig sein?". Durch die Änderung der Lizenzgewährung durch Oracle ab Januar 2019 und das Abkündigen der öffentlich verfügbaren Updates für Java 8 ist eine Änderung der bisherigen Java-Strategie der kommerziellen Nutzer erforderlich. Bernhard Halbetel von SAM CoOp referiert deshalb zu den möglicherweise bestehenden Compliance-Risiken aufgrund des automatisierten Ausrollens von Patches und Updates sowie der Nutzung von Features, die nur unter „Subscription" gewährt werden. Auch eine künftige Strategie, die den möglichen den Einsatz von OpenJDK unter der GPL in Erwägung zieht, wird diskutiert. Zudem werden die Einschränkungsmöglichkeiten der verschiedenen Lizenzdecks und der virtuellen Umgebung besprochen.*

## 21. Dezember 2018

*Die DOAG-Geschäftsstelle beendet das Jahr 2018 und schließt das Berliner Büro über die Feiertage. Fried Saacke, DOAG-Vorstand und Geschäftsführer, blickt mit seinem Team auf ein sehr erfolgreiches Jahr zurück.*

## 2. Januar 2019

*Die DOAG wünscht allen Mitgliedern und Interessenten ein gesundes und erfolgreiches neues Jahr!*

*Günther Stürner*
*dbms publishing*

## Aus der Ferne betrachtet: DOAG und Oracle – gemeinsam unschlagbar

Als die DOAG vor mehr als dreißig Jahren gegründet wurde, war es für die Oracle-Anwender – damals richtige Pioniere – wichtig, mit Gleichgesinnten ins Gespräch zu kommen und sich auszutauschen. Dass Oracle diese Gesprächskreise gefördert und auf die Gründung der DOAG gedrungen hat, war der Erkenntnis geschuldet, dass gut informierte Kunden dem Geschäft nicht abträglich sein können und dass das Wissen und die Erfahrung der Kunden auch für Oracle nützlich sind. Das war sehr clever und weitsichtig.

Wenn die DOAG nach mehr als dreißig Jahren noch immer vital und aktiv daherkommt, ist dies nicht weniger als ein Segen für Oracle-Anwender, Oracle-Partner und vor allem für Oracle selbst.

Die DOAG hat über die Jahre den Know-how-Transfer in einer Art kultiviert, dass man nur den Hut ziehen kann. Es ist für mich immer wieder erstaunlich, wie viele Konferenzen, Videos, Interviews, Expertengespräche etc. angeboten werden – Angebote auf sehr hohem Niveau, durchgeführt von exzellenten Kenner(n)(innen) der Materie, mit viel Praxisbezug. Keine Marketing-Schaumschlägereien, sondern fundierte Informationen, die man unmittelbar nutzen kann.

Oracle sollte eigentlich jeden Abend ein Stoßgebet aufsagen und darum bitten, dass die DOAG auch weiterhin so fleißig an den unzähligen Veranstaltungen und Publikationen arbeitet. Das ist Marketing und Vertriebsunterstützung in der allerbesten Form. Frei Haus. Bei minimalen Kosten für Oracle.

Die technischen Oracle-Kollegen aus Deutschland sowie die Entwickler und Produkt-Manager aus Redwood Shores haben das längst erkannt. Die Kommunikation ist hier eng und sehr vertrauensvoll. Ein Auftritt bei einer DOAG-Konferenz wird als Ehre empfunden. Diskussionen mit Kunden werden als wichtiger Input verarbeitet und nicht als lästig betrachtet, auch wenn es manchmal richtig zur Sache geht. Das Geben und Nehmen ist ausgeglichen, so wie es sein muss, und trotzdem oder gerade deshalb kommen auch kritische Punkte auf den Tisch, die diskutiert und einer Lösung zugeführt werden.

Diese Art der Zusammenarbeit ist eine Blaupause für alle Gruppen und alle Führungsebenen von Oracle, denn auch die nicht-technischen Problemfelder müssen adäquat, vertrauensvoll und auf Augenhöhe diskutiert werden. Ein Aussitzen oder das Ignorieren von Punkten, die Kunden auf den Nägeln brennen, ist alles andere als verkaufsfördernd. Besser wäre ein ständiger Dialog zwischen der DOAG, als Vertretung der Kunden, und Oracle. Jeden Monat sollte man sich treffen und ernsthaft die wichtigsten Kunden-Themen besprechen. Nein, kein Schmusekurs, sondern echte und harte Arbeit, die beiden Seiten zugutekommt, den Kunden und Oracle.

Klingt das banal? Vielleicht. Man sollte es trotzdem versuchen. Denn wie sagte Michael Jordan treffend: „100 Prozent der Würfe, die ich nicht mache, gehen daneben!" Na dann, hier ist der Ball.

*Stefan Kinnen (rechts) im Gespräch mit Dr. Axel Pols*

# „Die Wachstumsrate liegt momentan bei fünfundzwanzig bis dreißig Prozent …“

Das Thema „Cloud Computing“ ist in aller Munde. Stefan Kinnen, Vorstandsvorsitzender der DOAG, und Wolfgang Taschner, Chefredakteur des Red Stack Magazin, sprachen darüber mit Dr. Axel Pols, Geschäftsführer Bitkom Research GmbH.

*Was sind die Aufgaben und Ziele des Bitkom?*

**Dr. Axel Pols:** Bitkom ist der Digitalverband in Deutschland und hat die Aufgabe, Deutschland als Digitalstandort zukunftssicher zu machen. Einerseits kümmern wir uns um die Belange der Anbieter digitaler Lösungen und Technologien, andererseits beschäftigen wir uns mit der Nutzung dieser Technologien. Daraus entsteht ein großes Spektrum an Themen und Aufgaben.

*Aus welchem Grund hat der Bitkom einen umfangreichen Cloud-Monitor erhoben?*

**Dr. Axel Pols:** Der Bitkom befasst sich bereits seit Jahren mit dem Cloud Computing, sei es in Arbeitsgruppen, auf Veranstaltungen und auch in der Marktforschung. Wir möchten wissen, wo Deutschland bei den wichtigen Technologien steht, in diesem Fall beim Cloud Computing. Der Cloud-Monitor existiert seit dem Jahr 2011 in Zusammenarbeit mit der KPMG. Dadurch wir in der Lage, eine große, repräsentative Unternehmensbefragung durchzuführen und entsprechend zu kommunizieren.

*Wie viele Unternehmen haben daran teilgenommen und wie ist deren Struktur?*

**Dr. Axel Pols:** Wir haben seit Jahren eine stabile Stichprobengröße von rund fünfhundert Unternehmen mit mindestens zwanzig Mitarbeitern in Deutschland. Zielpersonen sind in der

**Über den Bitkom und Bitkom Research**

Bitkom ist der Digitalverband Deutschlands. Im Jahr 1999 als Zusammenschluss einzelner Branchenverbände in Berlin gegründet, werden mehr als 2.600 Unternehmen der digitalen Wirtschaft vertreten, unter ihnen mehr als 1.000 Mittelständler, über 500 Startups und nahezu alle Global Player. Diese Mitglieder bieten Software, IT-Services, Telekommunikations- oder Internetdienste an, stellen Hardware oder Consumer Electronics her, sind im Bereich der digitalen Medien oder der Netzwirtschaft tätig oder in anderer Weise Teil der sich digitalisierenden Wirtschaft.

Bitkom setzt sich insbesondere für eine innovative Wirtschaftspolitik, eine Modernisierung des Bildungssystems und eine zukunftsorientierte Netzpolitik ein. Im Jahr 2018 startet Bitkom eine neue Digitaloffensive. Im Mittelpunkt steht dabei zum einen der beschleunigte Ausbau von Gigabitnetzen und digitalen Infrastrukturen für Energie und Verkehr. Zum zweiten geht es um die breite Digitalisierung von Wirtschaft, Gesellschaft und Verwaltung, Bildung und Arbeit sowie Datenschutz und Sicherheit.

Die Bitkom Research GmbH ist ein Tochterunternehmen des Bitkom, auf die die Marktforschung rund um die digitale Wirtschaft spezialisiert. Bitkom Research versteht sich als Full-Service-Dienstleister von der Studienkonzipierung über die Datenerhebung und -analyse bis zur öffentlichkeitswirksamen Vermarktung der Ergebnisse. Darüber hinaus managt Bitkom Research das europäische Marktforschungsprojekt „European Information Technology Observatory" (EITO), das seit fünfundzwanzig Jahren über die Entwicklung der internationalen Technologiemärkte informiert.

Für den Bitkom werden regelmäßig Statistiken sowie Studien zur Entwicklung der digitalen Branche in Deutschland erstellt und dabei eng mit dem Experten-Netzwerk des Verbands zusammengearbeitet. Das Angebot an Publikationen und individuellen Marktforschungsdienstleistungen richtet sich an alle Unternehmen unabhängig von einer Bitkom-Mitgliedschaft. Das Team der Bitkom Research verfügt über mehr als zehn Jahre Erfahrung im Bereich der Marktforschung und Unternehmensberatung.

Regel die IT-Verantwortlichen. Die Ergebnisse werden gewichtet, sodass sich ein repräsentatives Gesamtbild für die deutsche Wirtschaft ergibt.

*Inwieweit nutzen diese Unternehmen bereits Cloud Computing beziehungsweise planen den Einsatz?*

**Dr. Axel Pols:** Zwei Drittel der Unternehmen in Deutschland nutzen bereit Cloud Computing in der einen oder anderen Form. Rund zwanzig Prozent der Unternehmen planen oder diskutieren derzeit den Einsatz, die verbleibenden dreizehn Prozent sehen momentan keinen Bedarf.

*Wie ist hier der Trend im Vergleich zu früheren Umfragen?*

**Dr. Axel Pols:** Deutschland ist beim Cloud Computing sehr spät gestartet und liegt beim Einsatz im Vergleich zu anderen Ländern wie Skandinavien, Japan oder den USA rund zwei Jahre hinterher. Wir beobachten allerdings in den letzten Jahren einen deutlichen Schub nach vorne.

*Welchen Einfluss hatte die Datenschutz-Grundverordnung auf diesen Trend?*

**Dr. Axel Pols:** Die Konformität mit der Datenschutz-Grundverordnung ist für die Cloud-Nutzer eines der Top-Kriterien bei der Auswahl der Cloud-Provider.

*Wie groß ist der Cloud-Computing-Markt in Deutschland?*

**Dr. Axel Pols:** In diesem Jahr werden hierzulande rund sieben Milliarden Euro mit Cloud-Lösungen umgesetzt; die Wachstumsrate liegt momentan bei fünfundzwanzig bis dreißig Prozent. Der Gesamtmarkt für IT-Lösungen beträgt in Deutschland rund einhundertfünfundsechzig Milliarden Euro und wächst um weniger als zwei Prozent. Die Cloud ist somit einer der Wachstumstreiber.

*Welche Vorteile versprechen sich die Unternehmen vom Cloud Computing?*

**Dr. Axel Pols:** Die klassischen Vorteile sind die Reduzierung der IT-Kosten sowie der Administrations-Aufwände, der bessere Abruf Zugriff auf IT-Ressourcen und die leichtere Skalierbarkeit. Hinzu kommt die schnelle und kostengünstige Erschließung zukünftiger Technologien wie beispielsweise Künstliche Intelligenz oder das Internet der Dinge.

*Wie hoch ist die Bereitschaft, auch geschäftskritische Anwendungen in die Cloud zu verlagern?*

**Dr. Axel Pols:** Dieser Punkt ist sehr wichtig, weil damit das Potenzial der Cloud-Lösung noch einmal deutlich ansteigt. Im Vergleich zum Vorjahr gehen große Unternehmen bereits dazu über, auch kritische Anwendungen in die Cloud zu migrieren. Ein Trend in diese Richtung ist also vorhanden.

*Welche Kriterien und Leistungen sind bei der Auswahl eines Cloud-Providers für ein Unternehmen wichtig?*

**Dr. Axel Pols:** Es gibt einige Klassiker, die seit Jahren unter den Top-Fünf-Argumenten genannt werden, darunter der Wunsch der Unternehmen, dass der Anbieter sein Rechenzentrum innerhalb der EU oder in Deutschland betreibt. Hinzu kommt die Anforderung nach einer transparenten Sicherheitsarchitektur beim Provider; ferner muss sich die Cloud-Lösung unkompliziert in die Unternehmens-IT integrieren lassen. Auch die lizenztechnischen und rechtlichen Fragen spielen natürlich eine wichtige Rolle. Aktuell steht, wie erwähnt, zudem die DSGVO-Konformität im Mittelpunkt.

*Welche Probleme treten bei der Integration der Cloud-Lösungen in die bestehende IT-Infrastruktur des Unternehmens auf?*

**Dr. Axel Pols:** Etwa zwei Drittel der befragten Unternehmen haben keine Schwierigkeiten mit der Integration. Beim restlichen Drittel gibt es Probleme mit den Compliance- und Sicherheitsanforderungen, insbesondere bei kritischen Anwendungen. Da ist oft auch externe Beratung notwendig.

*Ist es schwieriger, Daten in der Public Cloud umfassend zu schützen als im unternehmensinternen IT-Netzwerk?*

**Dr. Axel Pols:** Momentan herrscht bei den Unternehmen noch

die Auffassung vor, dass die Daten in der eigenen IT sicherer aufgehoben sind. Im Vergleich zu den Vorjahren gibt es allerdings einen Trend zu mehr Vertrauen in die Sicherheit in der Cloud; die großen Unternehmen gehen auch hier voran.

*Welche Art von Daten speichern Unternehmen in der Public Cloud?*
**Dr. Axel Pols:** Überwiegend beginnen die Unternehmen mit unkritischen Daten, sind allerdings zunehmend bereit, auch Kundendaten oder kritische Business-Informationen in die Cloud zu verlagern.

*Besitzen Unternehmen spezifische Sicherheitskonzepte, in denen Sicherheitsanforderungen und Sicherheitsmaßnahmen für den jeweiligen Anwendungsfall definiert sind?*
**Dr. Axel Pols:** Der Bedarf für solche Sicherheitskonzepte ist in jedem Fall vorhanden und deren Umsetzung ist jedem Unternehmen empfohlen. Aktuell haben sieben von zehn Cloud-Nutzer entsprechende Konzepte vorliegen, die allerdings oft nicht alle Szenarien abdecken. Es gibt also noch jede Menge Verbesserungspotenzial.

*Welche Geschäftsbereiche eines Unternehmens sind eher motiviert, ihre Anwendungen in die Cloud zu verlagern?*
**Dr. Axel Pols:** Es sind oftmals die Fachabteilungen, die mit einem akuten Bedarf eine Cloud-Lösung anstreben; das ist beispielsweise im Marketing der Fall und zunehmend auch in der Produktion. Mit Blick auf das Internet der Dinge bietet es sich geradezu an, Daten in der Cloud zusammenzuführen. In den Finanz- und Controlling-Abteilungen sowie im Personal-Ressort gibt es oft noch eine gewisse Zurückhaltung.

*Wie groß ist der Aufwand, um Mitarbeiter vom Einsatz der Cloud-Lösungen zu überzeugen?*
**Dr. Axel Pols:** Generell darf dieser Aufwand nicht unterschätzt werden. Es bedarf rechtzeitiger Informationsveranstaltungen, Workshops und Schulungen. Innerhalb der Unternehmen ist die Führungsebene generell positiver gegenüber der Cloud eingestellt, während die Anwender eher skeptisch sind, vielleicht auch aufgrund der Sorge, dass sich ihr Arbeitsfeld verändern wird.

*Wie wird sich die Cloud-Nutzung in Zukunft ändern?*
**Dr. Axel Pols:** Ich glaube, die Cloud- Nutzung wird immer mehr zum Standard. Bereits heute machen viele Anwender keinen Unterschied mehr zwischen Cloud- und On-Premises-Anwendungen. Die Rechenleistung aus der Steckdose wird in absehbarer Zeit ähnlich selbstverständlich sein wie heute der Bezug elektrischen Stroms.

*Welchen persönlichen Rat geben Sie Unternehmen mit auf dem Weg in die Cloud?*
**Dr. Axel Pols:** Wichtig ist, das Thema strategisch anzugehen und sich die Zeit zu nehmen, alle Seiten ausführlich zu beleuchten, bei Bedarf auch durch Hinzuziehen externer Beratung. Die sorgfältige Auswahl des richtigen Cloud-Providers ist ein weiterer wichtiger Schritt.

*Was würden Sie einem Cloud-Anbieter wie Oracle empfehlen?*
**Dr. Axel Pols:** Entscheidend ist, mit dem Thema Sicherheit zu punkten. Jeder Anbieter, der sich hier glaubwürdig und transparent zeigt, wird beim Kunden Gehör finden.

*Wie sollte eine Anwendergruppe wie die DOAG mit dem Cloud-Thema umgehen?*
**Dr. Axel Pols:** Die DOAG hat mit ihren Veranstaltungen und über die Mitgliederkommunikation gute Möglichkeiten, die Anwender zu informieren und den Erfahrungsaustausch zu fördern. Die Chancen der Cloud-Nutzung sollten dabei sicher auch ein Thema sein.

*Welche Nachteile gibt es für Unternehmen, die sich der Cloud verweigern?*
**Dr. Axel Pols:** Die Frage ist, ob sich ein Unternehmen im Wettbewerb behaupten kann, ohne die eingangs genannten Vorteile der Cloud zu nutzen.

## Zur Person: **Dr. Axel Pols**

Dr. Axel Pols ist seit dem Jahr 2012 Geschäftsführer der Bitkom Research GmbH, einem Tochterunternehmen des Bitkom e.V. Er ist zudem Chairman der internationalen Expertengruppe des europäischen Marktforschungsprojekts „European Information Technology Observatory" (EITO). Er kam im Jahr 2001 zum Bitkom und baute dort die Arbeitsbereiche Marktforschung und Außenwirtschaft auf. Zuvor war er im VDMA Fachverband Informationstechnik beschäftigt.

In seiner aktuellen Funktion als Geschäftsführer von Bitkom Research hat Dr. Axel Pols das Unternehmen erfolgreich als Anbieter von Marktforschungsdienstleistungen am Markt etabliert. Der Schwerpunkt liegt dabei auf empirischen Untersuchungen zur Nutzung digitaler Technologien durch Unternehmen und Verbraucher sowie Fragestellungen rund um die Digitalisierung. Er ist promovierter Wirtschaftswissenschaftler mit Abschlüssen der Universität Göttingen und der University of East Anglia (UK).


# Oracle Database 18c: Installation

Tobias Deml, ORACLE Deutschland B.V. & Co. KG

Die im Januar 2018 veröffentlichte Oracle Database 18c beinhaltet Verbesserungen in vielen Bereichen. Insbesondere bei der Installation gab es einige Neuerungen, die den Veränderungen der vorherrschenden Anforderungen im Datenbank-Umfeld Folge tragen. Der Artikel stellt die verschiedenen Installationsmöglichkeiten der Oracle-Datenbank Version 18c vor.

In den letzten Jahren gewann die Methodik der agilen Software-Entwicklung mehr und mehr an Zulauf. Mit dieser Bewegung im Entwicklungsumfeld haben sich auch die Anforderungen an den Datenbank-Sektor geändert. Themen wie „Automatisierung" und „Instrumentalisierung" haben deutlich an Bedeutung gewonnen, da durch eine Skript-basierte Installation beziehungsweise Administration viel Zeit eingespart werden kann. Folgende Änderungen wurden mit dieser Version eingeführt:

- Image-basierte Installation und Konfiguration
- Paket-basierte Installation (RPM)
- Oracle Database Docker-Images

## Image-basierte Installation

Nach der Oracle Grid Infrastructure 12.2 wurde nun mit der Version 18c auch bei der Datenbank die Image-basierte Installation eingeführt. Dies beschleunigt und vereinfacht den Installationsprozess der Datenbank-Software enorm. Grund dafür ist, dass im ausgelieferten Archiv der Daten-

```
cd $ORACLE_HOME
unzip LINUX.X64_180000_db_home.
zip
./runInstaller
```

*Listing 1: Entpacken und Installation der Datenbank-Software*

```
[root@18c-test2 ~]# whoami
root

[root@18c-test2 ~]# yum install -y oracle-database-preinstall-18c
...

[root@db18c-test install]# ls oracle-database-ee-18c-1.0-1.x86_64.rpm
oracle-database-ee-18c-1.0-1.x86_64.rpm

[root@db18c-test install]# yum -y localinstall oracle-database-ee-
18c-1.0-1.x86_64.rpm
...
```

*Listing 2: Paket-basierte Installation der Datenbank-Software*

bank-Software bereits die Binaries in ihrer schlussendlichen Form paketiert sind. Während des Installationsprozesses werden lediglich einige Konfigurationen vorgenommen und die Binaries „re-linked".

Nachfolgend ist der Installationsprozess mit den notwenigen Befehlen genau beschrieben: Initial ist das ZIP-Archiv der Datenbank-Software („LINUX.X64_180000_db_home.zip") herunterzuladen. Anschließend kopiert und entpackt man dieses im endgültigen „ORACLE_HOME" der Datenbank-Umgebung. Anschließend wird der Installationsprozess wie gewohnt mit „runInstaller" gestartet *(siehe Listing 1)*.

Wie bereits erwähnt sind für den Installationsvorgang weniger Schritte notwendig, als es im Vergleich zur vorherigen Datenbank-Version 12.2 der Fall war. Außerdem sind beim Start des „runInstaller" weniger Angaben im Installationsassistenten beziehungsweise beim Silent-Aufruf notwendig, da sich die Software bereits am endgültigen Installationsort befindet. Hinsichtlich der Erstellung und Konfiguration der Oracle-Datenbank kann man wie gewohnt den DBCA oder das präferierte Tooling verwenden.

## Paket-basierte Installation (RPM)

Mit Veröffentlichung der Datenbank-Version 18c besteht nun die Möglichkeit, eine Single-Instance-Datenbank mit einem RPM-Paket zu installieren. Die Benutzung dieser Funktionalität ist aufgrund der Verfügbarkeit des RPM auf UNIX-artige Betriebssysteme beschränkt.

Im ersten Schritt ist das RPM-Paket herunterzuladen und auf den ausgewählten Server zu kopieren. Hinsichtlich der Bezugsquellen stehen zwei Möglichkeiten zur Auswahl, das „Oracle Technology Network" und das „Unbreakable Linux Network". Wenn das RPM-Paket auf den

Server kopiert ist, werden alle anschließenden Tätigkeiten mit dem „root“ fortgesetzt.

Um den Server und das darauf befindliche Betriebssystem für die bevorstehende Installation vorzubereiten, wird empfohlen, das ebenfalls zur Verfügung gestellte „oracle-database-preinstall-18c“-RPM zu installieren. Damit werden diverse notwendige Software des Betriebssystems nachinstalliert und einige Anpassungen hinsichtlich Parametrisierung und Benutzerverwaltung vorgenommen.

Sobald die nötigen Vorbereitungen abgeschlossen sind, kann nun die Paketbasierte Installation der eigentlichen Datenbank-Software erfolgen. Um den Vorgang zu starten, muss man im Ordner des RPM-Pakets sein und dieses mit „yum localinstall“ installieren. *Listing 2* zeigt den Prozess auf einem Server mit Oracle Enterprise Linux.

Mit Einrichten des RPM-Pakets wird die Software der Oracle-Datenbank 18c unter dem Pfad „/opt/oracle/product/18c/dbhome_1“ installiert. Um nun eine entsprechende Datenbank zu erstellen, wurde ebenfalls ein neues Utility entwickelt, um diesen Vorgang mit Ausführen nur eines Befehls abhandeln zu können. Im Verzeichnis „/etc/init.d“ befindet sich ein Executable, um diesen Vorgang zu starten. Das Beispiel im *Listing 3* zeigt die Erstellung einer Single-Instance-Datenbank mit dem zuvor erwähnten Werkzeug. Nach Fertigstellung dieses Vorgangs wurde eine neue Oracle-Multitenant-Datenbank „ORCLCDB“ erstellt, deren Prozesse dem üblichen „oracle“-Benutzer zugeordnet wurden.

```
[root@18c-test2 opc]# /etc/init.d/oracledb_ORCLCDB-18c configure
Configuring Oracle Database ORCLCDB.
Prepare for db operation
8% complete
Copying database files
31% complete
Creating and starting Oracle instance
32% complete
36% complete
40% complete
43% complete
46% complete
Completing Database Creation
51% complete
54% complete
Creating Pluggable Databases
58% complete
77% complete
Executing Post Configuration Actions
100% complete
Database creation complete. For details check the logfiles at:
 /opt/oracle/cfgtoollogs/dbca/ORCLCDB.
Database Information:
Global Database Name:ORCLCDB
System Identifier(SID):ORCLCDB
Look at the log file "/opt/oracle/cfgtoollogs/dbca/ORCLCDB/ORCLCDB.log"
for further details.

Database configuration completed successfully. The passwords were auto
generated,
you must change them by connecting to the database using 'sqlplus / as
sysdba' as the oracle user.
```

*Listing 3: Vereinfachte Erstellung einer Oracle-Datenbank-Instanz*

## Oracle Database Docker-Images

Bereits mit der Version 12.2 hat Oracle erstmalig ein offizielles Docker-Image der Datenbank zur Verfügung gestellt. Grund für die Erwähnung in diesem Artikel ist die Veröffentlichung einiger Neuerungen in Bezug auf die Version 18c in diesem Umfeld. Am 20. Oktober 2018 kam eine neue Version der Oracle Database XE heraus. Neben den üblichen Installationsmöglichkeiten wurde diese Edition ebenfalls als Docker-Image offiziell bereitgestellt.

Alle verfügbaren Docker-Images sind mit den notwendigen Skripten auf dem Oracle-GitHub-Repository *(siehe „https://github.com/oracle“)* verfügbar. Dort liegen neben den Docker-Images auch Repositories diverser Oracle-Open-Source-Projekte wie GraalVM und Helidon.

## Fazit

In den vergangenen Monaten und besonders mit dem Release 18c wurden diverse Installationsmöglichkeiten hinzugefügt. Diese Neuerungen eröffnen der Oracle-Datenbank viele neue Möglichkeiten hinsichtlich verschiedenster Plattformen sowie Installationsmöglichkeiten.

## Weitere Informationen und Links

- Deutschsprachiger Oracle-Blog, Thema Oracle Database 18c: *https://blogs.oracle.com/coretec/datenbank-18c*
- Dokumentation Oracle Database 18c: *https://docs.oracle.com/en/database/oracle/oracle-database/18/index.html*
- Dokumentation Oracle Database 18c, Installation Guide: *https://docs.oracle.com/en/database/oracle/oracle-database/18/ladbi/index.html*
- Oracle GitHub Repository, Direktlink zu den Datenbank-Images: *https://github.com/oracle/docker-images/tree/master/OracleDatabase*

Tobias Deml
tobias.deml@oracle.com

# Oracle 18 XE

Johannes Ahrends, CarajanDB GmbH

Am 19. Oktober hatte das Warten ein Ende: Oracle hat endlich eine neue Version der kostenlosen Oracle-Datenbank freigegeben. „Oracle Database 18c Express Edition" oder kurz „Oracle 18 XE" basiert auf der Oracle Enterprise Edition 18c mit dem Release-Update vom Oktober 2018. Natürlich ist auch diese Version wieder limitiert, doch sie enthält einige Features, die man so nicht erwartet hätte.

Nachdem die letzte Oracle-XE-Version 11g Release 2 in die Jahre gekommen ist (immerhin gibt es sie schon seit mehr als vier Jahren), war es jetzt Zeit für ein Update. Bereits vor einem Jahr hat Oracle angekündigt, die Release-Policy der Datenbank zu ändern. Beginnend mit der Version 18 wird es jedes Jahr ein neues Release geben. Dies gilt laut Oracle auch für die XE-Datenbank. Wir können uns also schon darauf freuen, dass es im Jahr 2019 eine „Oracle 19 XE" geben wird. Allerdings gilt auch weiterhin die Einschränkung, dass es keinen Support beziehungsweise keine Patches für die XE-Datenbank geben wird. Das Oracle-Community-Forum ist also die einzige offizielle Anlaufstelle für Probleme beim Betrieb von XE. Wer mag, darf sich natürlich auch gerne an die DOAG oder an den Autor wenden.

## Limitierungen

Wie bereits erwähnt – und nicht anders zu erwarten – gibt es harte Grenzen für den Einsatz von XE. Die technischen Voraussetzungen beziehungsweise Restriktionen sind:

- Maximal zwei CPU-Threads
- Maximal zwei GByte Memory (SGA + PGA)
- Maximal zwölf GByte Benutzerdaten
- Eine Installation pro Umgebung

Wer die Dokumentation intensiv liest, wird eine weitere Limitierung feststellen: maximal drei Pluggable Databases (PDB). Mit dieser Einschränkung wird jedoch klar, dass die XE-Datenbank die Multitenant-Database-Option unterstützt. Zwar sind drei PDBs auch für die Standard beziehungsweise die Enterprise Edition ohne Multitenant-Option erlaubt, dort allerdings nur, wenn es sich um eine normale PDB, einen Proxy und einen Application Container handelt. Bei der XE-Datenbank sind hingegen gleichartige PDBs möglich. Somit steht dem Test dieser Option nichts mehr im Wege. Neben der Multitenant-Database-Option enthält XE folgende Optionen:

- Advanced Compression
- Advanced Security
- Partitioning
- Label Security
- Database Vault
- Advanced Analytics
- Database In-Memory

Zwar sind nicht alle Optionen in voller Funktionsbreite verfügbar und man kann geteilter Meinung sein, ob sich die In-Memory-Option bei maximal zwei GByte Hauptspeicher

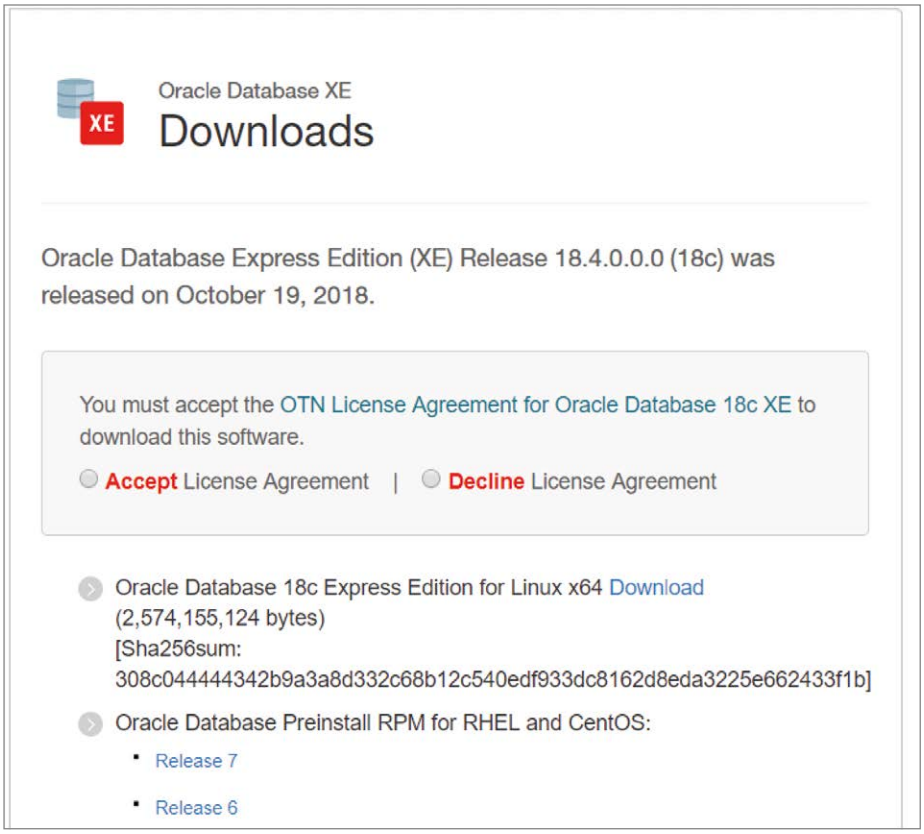



*Abbildung 1: Download Oracle Database XE*

```
cd $ORACLE_HOME/rdbms/admin
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -b catnojav -d $ORACLE_HOME/
rdbms/admin catnojav.sql
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -b rmxml -d $ORACLE_HOME/xdk/
admin rmxml.sql
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -b rmjvm -d $ORACLE_HOME/ja-
vavm/install rmjvm.sql
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -b rmctx -d $ORACLE_HOME/ctx/
admin catnoctx.sql
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -b imrem -d $ORACLE_HOME/ord/
im/admin imremdo.sql
$ORACLE_HOME/perl/bin/perl catcon.pl -n 1 -e -b utlrp -d '''.''' utlrp.
sql
```

*Listing 1*

```
SQL> ALTER SYSTEM SET sga_target=1900M SCOPE=spfile;
SQL> ALTER SYSTEM SET pga_aggregate_target=100M;
```

*Listing 2*

```
ORA-56752: Oracle Database Express Edition (XE) memory parameter inva-
lid or not specified
ORA-01078: failure in processing system parameters
```

*Listing 3*

```
SQL> ALTER PLUGGABLE DATABASE xepdb1 CLOSE IMMEDIATE;
SQL> DROP PLUGGABLE DATABASE xepdb1 INCLUDING DATAFILES;
```

*Listing 4*

```
SQL> ALTER SYSTEM SET db_create_file_dest='/u02/oradata';
```

*Listing 5*

```
SQL> CREATE PLUGGABLE DATABASE johannes
ADMIN USER pdbadmin IDENTIFIED BY MANAGER;
```

*Listing 6*

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size=50G;
SQL> ALTER SYSTEM SET db_recovery_file_dest='/u03/orabackup';
```

*Listing 7*

lohnt, aber versuchen kann man es ja mal. Der Appetit kommt bekanntlich beim Essen. Es scheint, als habe Oracle einen Hintergedanken … also Vorsicht, es könnte sein, dass die Entwickler oder DBAs auf Ideen gebracht werden, die viele zusätzliche Lizenzen für die Enterprise Edition nach sich ziehen.

## Installation und Konfiguration

Die Installation der Software und die Konfiguration der Datenbank sind denkbar einfach. Neben dem nur ca. 2,5 GByte großen Download *(siehe Abbildung 1)* ist nur noch das Pre-Install RPM für RHEL beziehungsweise CentOS herunterzuladen, dazu gehört natürlich auch Oracle Enterprise Linux.

Das bedeutet umgekehrt, dass es momentan keine Möglichkeit gibt, die XE-Datenbank auf SuSE oder einer anderen Linux-Distribution zu installieren. Auf die Verwendung unter Microsoft Windows kann nicht eingegangen werden, da XE für Windows noch nicht verfügbar war, als der Artikel entstanden ist. Die Installation muss als User „root" erfolgen und verwendet folgende Verzeichnisse beziehungsweise Variablen:

- ORACLE_BASE=/opt/oracle
- ORACLE_HOME=/opt/oracle/product/18c/dbhomeXE

Ausgepackt und installiert wird, wie für RPM-Pakete mittlerweile üblich, über „yum localinstall oracle-database-xe-18c-1.0-1.x86_64.rpm" – und fertig. Auf die weitere Beschreibung der Installation wird hier verzichtet, die Dokumentation ist ausführlich und der Autor hat noch niemanden gesehen, der bei der Installation Probleme hatte.

Löblicherweise hat Oracle sowohl die Skripte zum automatischen Starten und Stoppen der Datenbank als auch zum Löschen der Datenbank sowie zum Löschen der Software mitgeliefert. Solche Skripte wären auch für die Standard und die Enterprise Edition ganz nett.

## Optionen löschen

Die Datenbank heißt nach der Installation „XE" und wird mit den genannten „Optionen" erstellt. Einzig Apex und der SQL Developer sind nicht mitinstalliert, weil Oracle endlich eingesehen hat, dass es nicht sinnvoll ist, irgendeine veraltete Version der Tools mit auszuliefern. Stattdessen sollte man sich dann lieber die neueste Version laden und installieren.

Da es keine Patches für XE gibt, empfiehlt es sich, zumindest die Oracle Java Virtual Machine zu deinstallieren. Grund dafür ist, dass OJVM bei den Security Alerts immer wieder hohe Risk Scores (9.x) hat. Was man nicht installiert hat, muss man auch nicht patchen, so der Grundsatz des Autors. *Listing 1* zeigt eine mögliche Vorgehensweise. Damit werden die OJVM, Multimedia und XMLDB deinstalliert. Die Deinstallation von Optionen erklärt auch Mike Dietrich in mehreren Blogs, so unter *„https://mikedietrichde.com/2017/08/07/javavm-xml-clean-oracle-database-11-2-12-2".*

## Memory

Wie erwähnt, ist die maximale Größe von SGA und PGA auf zwei GByte begrenzt.

Um diesen Speicher möglichst optimal auszunutzen, empfiehlt es sich, die Parameter entsprechend anzupassen. Die PGA ist standardmäßig auf 512 MB eingestellt, was der Autor für diese Konfiguration als viel zu viel erachtet. 100 MB sollten bei einer XE-Datenbank ausreichen – somit gewinnt man 400 MB für den Buffer Cache oder auch für die In-Memory-Option *(siehe Listing 2)*. Bei der Konfiguration gilt es vorsichtig zu sein, denn ein zu hoher Wert führt beim Restart der Instanz und erst dann zu einem Fehler *(siehe Listing 3)*.

## Multitenant Database

Wer sich mit der Multitenant-Database-Option vertraut machen will, dem seien Oracle Managed Files empfohlen. Leider wird die XE-Datenbank mit benannten Dateien erstellt, was die Syntax für den Befehl „CREATE PLUGGABLE DATABASE“ unnötig erschwert. Daher sollten folgende Einstellungen vorgenommen werden: Löschen der mitinstallierten PDB *(siehe Listing 4)*, Hinzufügen der OMF-Parameter *(siehe Listing 5)* und Erstellen der PDB *(siehe Listing 6)*. Außerdem ist es ganz hilfreich, auch gleich noch die Fast Recovery Area zu definieren, dann steht auch einem Standard-Backup mit RMAN nichts mehr im Wege *(siehe Listing 7)*.

## Neue Datenbank erstellen

Wie bereits erwähnt, kann es nur eine Installation pro Umgebung geben, dabei ist es unerheblich, ob es sich um einen physikalischen Server oder eine virtuelle Umgebung wie VMware oder Container handelt. Die Software-Installation kann nur im Verzeichnis „opt/oracle/product/18c/dbhomeXE“ erfolgen, insofern verbietet sich damit eine zweite Installation – zumindest bis Version 19. Es lässt sich jedoch jederzeit eine neue Datenbank anlegen. Der Database Configuration Assistant (DBCA) ist Bestandteil von XE und kann für die Erstellung weiterer Datenbanken mit unterschiedlichen Namen genutzt werden *(siehe Abbildung 2)*. Dabei ist es auch möglich, eine NON-CDB-Datenbank anzulegen; Oracle Managed Files können hier von Beginn an verwendet werden. Insofern ist sehr zu empfehlen, sich eine eigene XE-Datenbank zu bauen. Die einzige Voraussetzung beziehungsweise Beschränkung bei der selbst konfigurierten XE-Datenbank ist, dass ein vordefiniertes Template verwendet wird.

Allerdings kann von den erstellten Datenbanken immer nur eine aktiv sein. Ansonsten kommt die Fehlermeldung „ORA-00442: Oracle Database Express Edition (XE) single instance violation error“. Das bedeutet jedoch im Umkehrschluss, dass man sich im Entwicklungsumfeld oder zum Testen durchaus mehrere XE-Datenbanken erstellen kann.

## Fazit

Man darf eine XE-Datenbank produktiv einsetzen, wenn die erwähnten Limitierungen eingehalten werden. Der Autor kennt genügend Anwendungen, die mit zwölf GByte Benutzerdaten und zwei GByte SGA auskommen. Mit XE können diese Anwendungen im Gegensatz zur Standard Edition sogar Flashback Database und Transparent Data Encryption verwenden. Es ist jedoch zu bedenken, dass man keine Patches einspielen kann. Dennoch ist XE eine sehr gute Möglichkeit, sich mit den verschiedenen Oracle-Optionen vertraut zu machen, ohne Gefahr zu laufen, einen Lizenzverstoß zu begehen.

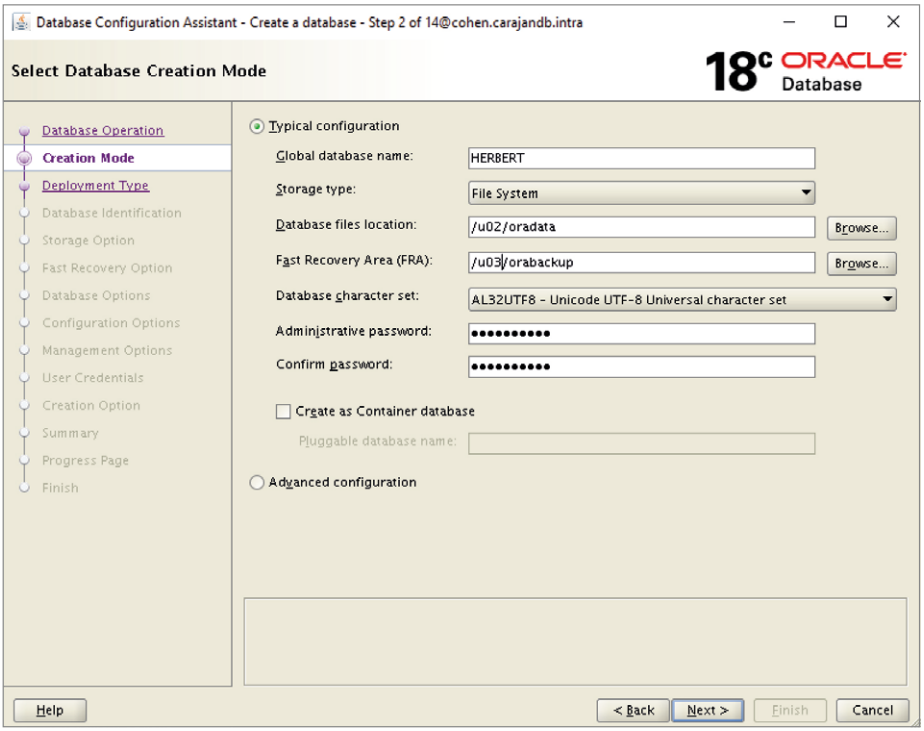
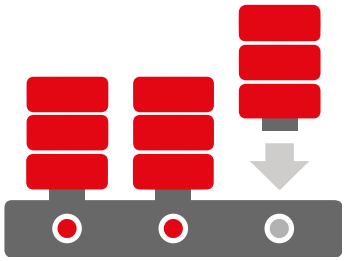



*Abbildung 2: Database Configuration Assistant*

Johannes Ahrends
johannes.ahrends@carajandb.com



# Von 12.1 NonCDBs zu 18c Multitenant – ein Erfahrungsbericht

Robert Ortel, Hypoport Systems GmbH

In einem Umfeld einer gewachsenen Infrastruktur entstand die Idee zur Vereinheitlichung, Vereinfachung und Konsolidierung der vorhandenen Datenbank-Infrastruktur. Die Saat dazu hatte die Multitenant-Option gesät. Der Artikel stellt die ehemalige Infrastruktur des Autors kurz vor und erläutert, welche Überlegungen ihn von Multitenant überzeugt haben. Er stellt insbesondere die Erfahrungen rund um die Upgrades und Migrationen sowie den Betrieb über die ersten Wochen vor und wie er auch ohne RAC eine möglichst hohe Verfügbarkeit erreichen will.

Über viele Jahre entstanden und gingen Datenbanken, bereicherten neue Server die Infrastruktur und wurden unterschiedliche Lizenzen für verschiedene Zwecke und Situationen eingesetzt. Auch wenn die Bedeutung der Datenbanken keinesfalls schrumpfte, standen die vorhandene Hardware und die Anforderungen an diese in keinem Verhältnis mehr.

Benötigter RAM, lokaler Plattenplatz auf den physischen Servern (der mit jedem Patch-Bundle anwächst) und die Limitierung, den gleichen Datenbank-Namen nicht mehrfach pro Server verwenden zu können, begrenzten die Konsolidierung. Ohnehin war Konsolidierung bisher kaum ein Thema für die Oracle-Datenbanken im Unternehmen des Autors. So kam ihm eine Konsolidierungsidee in den Sinn, mit der Multitenant-Option die Anzahl der Server zu reduzieren und gleichzeitig für eine deutlich bessere Ausnutzung der Lizenzen zu sorgen. Primär wird nachfolgend die technische Umsetzung dargestellt.

## Die alte Infrastruktur

Verteilt auf 16 physische Servern und zwei Standorte liefen 76 Datenbank-Instanzen. Aufgrund der Oracle-Lizenzpolitik setzte man nur physische Server exklusiv für die Oracle-Datenbank ein. All diese Datenbank-Instanzen der beiden Editionen liefen unter 12.1.0.2 als Single Instance und NonCDB. Weil diese sehr stabil ist und nach Erfahrung des Autors nur im Falle von Hardware-Problemen den Dienst versagt, genügte die Single Instance bisher den Anforderungen. Hardware-Probleme erfordern dann zwar ein manuelles Eingreifen, doch dank verfügbarer Ersatz-Hardware und schneller Mechanismen zu deren Inbetriebnahme toleriert das Unternehmen diese möglichen Ausfallzeiten. Eine Migration zu Singletenant war bisher wegen Problemen im Oracle-Migrations-Skript („noncdb_to_pdb.sql“) bei der Migration von 11.2.0.4 nach 12.1.0.2 vertagt worden.

Auf jedem der physischen Server läuft ASM und stellt die LUNs von Shared Storages jeweils als drei Diskgruppen pro Datenbank zur Verfügung. Diese speichern dann jeweils Datafiles, Logfiles und Dateien der Recovery Area. Auf produktiven Datenbanken sind diese Diskgruppen redundant über zwei Shared Storages ausgelegt („normal redundancy“). Auf den Test- und Entwicklungssystemen arbeitet man bisher ohne Redundanz seitens ASM („external redundancy“). Nur für die Dateien des Backups verwendet man reguläre Dateisysteme, entweder gemountet per NFS oder als LUN von einem Shared Storage mit einem Dateisystem. Durch den Einsatz von ASM auf jedem Server setzt man auch die Funktionalität von Oracle Restart überall ein.

## Optimierungsidee

Die meisten der sechzehn Server waren bei Weitem nicht ausgelastet. Ein großer Teil der Hardware war nur in Verwendung, weil neuer RAM, lokaler Plattenplatz und/oder die Limitierung der Datenbank-Namen das Verwenden eines neuen Servers nötig machte, der (vor allem für Test und Entwicklung) meist bereits existierte und nicht erst bestellt werden musste.

Mit der Idee hinter Multitenant wurde nach und nach klar, dass sich dies grundlegend ändern ließ. Die sechzehn Server könnte man auf zwei Server reduzieren (ohne Betrachtung der Ersatz-Hardware). Diese zwei Server würden jedoch deutlich mehr RAM benötigen. Zusätzlich ließe sich über die zwei Standorte eine Replikation betreiben, die neue Vorteile unter anderem für die Hochverfügbarkeit bietet. Nicht zuletzt bringt das Cloning von PDBs mit Multitenant neue Möglichkeiten für Entwicklung und Tests mit sich. Neben den Optimierungen rund um Platzbedarf im Server-Rack, Stromverbrauch und

Wartungsaufwand für die vielen Datenbank-Server, die Verbesserung der Hochverfügbarkeit und der neuen Features war auch eine Optimierung der Lizenzen möglich.

## Die neue Infrastruktur

Auf jedem der zwei neuen Server läuft nun allerdings nicht nur eine Container Database (CDB), sondern jeweils zwei. Nur eine CDB ist quasi aktiv, konsumiert einen großen Anteil an RAM für SGA/PGA und lässt Pluggable Databases (PDBs) laufen. Die andere CDB ist zwar geöffnet, aber nur im „restricted mode" und nur mit sehr wenig SGA/PGA gestartet.

Kommt es zum permanenten Ausfall einer CDB, kann die zweite CDB auf dem Server aushelfen und die PDBs nach einem Neustart (wegen Anpassung der SGA/PGA) aufnehmen, die zuvor in der ersten CDB liefen. Außerdem lassen sich so die Patching-Downtimes verkürzen, die meist zwei Mal im Jahr für Release-Updates durchgeführt werden. Die zweite CDB kann man bereits patchen und muss dann nur die PDBs umhängen. Die Ausführung von „datapatch" gelingt dann wieder online.

Auf jedem der zwei neuen Server läuft wieder ASM und damit Oracle Restart. Jede CDB erhält wieder ihre üblichen drei Diskgruppen für Datafiles, Logfiles und Recovery Area. Da sich alle PDBs dann den Platz für Logfiles und Recovery Area mit der CDB teilen, spart man zusätzlich Plattenplatz ein, da sich nun auch alle eine Art Hochlast-Reserve teilen und nicht mehr jede DB ihre eigene Reserve bekommt. Jede PDB erhält nur noch eine Diskgruppe für ihre Datafiles. In dieser neuen Infrastruktur werden alle Diskgruppen redundant („normal redundancy") betrieben und von zwei Shared Storages versorgt. So schützt die Redundanz vor Datenfehlern und dem Ausfall eines Storage.

An jedem der zwei Standorte wird einer der Server laufen und eine Ersatz-Hardware bereitstehen. Der aktive Server an jedem Standort wird per Data Guard eine Replikation auf den anderen aktiven Server am anderen Standort vollführen. Wie von Oracle vorgesehen, laufen die CDBs unter 18.3 mit lokalem Undo, also einem Undo-Tablespace in jeder PDB, um alle neuen Features von Multitenant (Flashback PDB, PDB Point-In-Time-Recovery etc.) nutzen zu können.

## Migrationspfad

Für den Migrationspfad von 12.1.0.2 NonCDBs zu 18.3.0.0 Multitenant bieten sich viele Möglichkeiten. In diesem Fall ist Ausführungssicherheit bei der Migration wichtiger als die Kürze der Downtime, da nächtliche Downtimes vergleichsweise einfach zu erhalten sind. Als einziger Oracle-DBA möchte der Autor die Methoden möglichst sicher beherrschen, was nur möglich ist, wenn dies nicht so viele sind. So fiel die Entscheidung auf die klassische Migration in zwei Phasen, analog zu dem, wie es auch Mike Dietrich *(siehe „https://mikedietrichde.com/2017/03/08/converting-an-12-1-non-cdb-and-plug-it-into-an-12-2-cdb")* für das Upgrade von 12.1 nach 12.2 empfiehlt: 12.1 NonCDB nach 18.3 NonCDB nach 18.3 PDB. Alternativ kommt für einige Sonderfälle die Migration mit klassischem Datapump zum Einsatz.

Die zwei-phasige Migration dauert für alle Datenbanken, egal welcher Größe, nahezu gleich lang, weil nicht der gesamte Datenbestand kopiert werden muss. Das ist ein großer Vorteil, denn es gilt auch Multi-Terabyte-Datenbanken zu migrieren. Die zwei Kernphasen dauern ca. 25 Minuten (12.1 NonCDB nach 18.3 NonCDB) und 12 Minuten (18.3 NonCDB nach 18.3 PDB). Zusammen mit allen manuellen Vorbereitungen (insbesondere Checks), Zwischenschritten (wie DST.-Update) und Nachbereitungen (Anpassungen jeder PDB) war pro Datenbank knapp eine Stunde erforderlich.

Der Pfad mit Datapump kann bei kleinen Datenbanken einen Geschwindigkeitsvorteil bieten und dient als Fallback für alle anderen Datenbanken. Nur bei den wenigen großen Datenbanken mit mehreren Terabytes steht diese Option dann quasi nicht zur Verfügung, weil sie viel länger brauchen würde.

## Checks vor der Migration

Im Zuge der Checks vor der Migration hielt sich der Autor an verschiedene Empfehlungen. Auch wenn es keine Checkliste für die manuelle Migration nach 18c NonCDB zu geben scheint, hält er sich zumindest an die Checkliste für die Migration nach 12.2 NonCDB: Doc ID 2173141.1. Es ist zwar Teil dieser Checkliste, er möchte dennoch die Ausführung von „dbupgdiag.sql" (siehe Doc ID 556610.1) hervorheben. Es liefert zum Zustand der Datenbank vor dem Upgrade eine gute Zusammenfassung, die insbesondere für den Support im Problemfall nötig ist.

Nicht Teil der Checkliste, dafür aber von Mike Dietrich *(siehe „https://mikedietrichde.com/2018/09/21/improved-preupgrade-jar-for-oracle-12-2-and-18c")* empfohlen und über Doc ID 884522.1 zu beschaffen, ist das Oracle-Database-Pre-Upgrade-Utility, das vor jedem Upgrade ausgeführt wurde. Sind all diese Checks erfolgreich, startet das Upgrade.

## Erfahrungen mit „DBMS_PDB.CHECK_PLUG_COMPATIBILITY"

Mit den ersten Tests für die Migration hielt man sich an den Vorschlag von Mike Dietrich: 12.1 NonCDB nach 18.3 NonCDB nach 18.3 PDB. Während das Upgrade nach 18.3 NonCDB von Beginn an tadellos funktionierte, gab es gleich beim ersten Versuch mit dem zweiten Schritt Probleme. Denn vor der Migration zu einer PDB steht der Check mittels „DBMS_PDB.CHECK_PLUG_COMPATIBILITY" an, ob diese kompatibel ist. Doch in der Release-Version von 18.3.0.0 kommt es dabei immer zu einem gravierenden Fehler: „ORA-07445 [__intel_ssse3_rep_memcpy() +6471]". Ein Ergebnis des Checks erhält man nicht.

Durch einen Service Request konnte dies als Bug 28502403 identifiziert werden, wofür es zwischenzeitlich einen Patch gibt. Dieser beseitigt das Problem erfolgreich und ist ebenfalls im Release-Update 18.4 enthalten. 18.4 konnte bisher allerdings noch nicht getestet werden.

Inzwischen hatte man sich einen Workaround überlegt. Die Migration lässt sich auch entlang eines anderen Pfads vollziehen: 12.1 NonCDB nach 12.1 PDB nach 18.3 PDB. In 12.1.0.2 arbeitet der Aufruf „CHECK_PLUG_COMPATIBILITY" tadellos und so ist dies eine Alternative. Da man jedoch Datenbanken mit beiden National Charactersets betreut (UTF16 und UTF8), sind in diesem Fall für den ersten Migra-

tionsschritt zwei verschiedene 12.1 CDBs bereitzuhalten, denn 12.1 unterstützt nicht beide National Charactersets in einer CDB (im Gegensatz zu 18c). So muss je nach National Characterset in der NonCDB die passende CDB für die Migration gewählt werden.

Mit diesem Vorgehen gab es einige erfolgreiche Migrationen, man stieß aber vereinzelt auch auf Probleme. In einem Fall gab es Probleme mit dem Patch-Stand bei „CHECK_PLUG_COMPATIBILITY". Es gab mehrere „ROLLBACK SUCCESS"- und „APPLY SUCCESS"-Meldungen in der View „DBA_REGISTRY_SQLPATCH" zu dem gleichen Bundle-Patch. Erst nach manuellem Löschen aus „DBA_REGISTRY_SQLPATCH" (nach Rücksprache mit dem Support), sodass es nur noch maximal einen Eintrag jeweils zu „APPLY" und „ROLLBACK" gibt, lief „CHECK_PLUG_COMPATIBILITY" erfolgreich durch.

Eine andere Migration zu einer 12.1 PDB verursachte, dass die View „SYS.USER_AUDIT_POLICIES" hinterher „invalid" war. Ein Neukompilieren beseitigte das Problem allerdings nicht. Weil die Verfügbarkeit des Patches zum Bug 28502403 kurz nach dem Auftreten dieses Problems gegeben war, konnte man diesen Migrationsweg wieder verwerfen, ohne dazu eine Lösung zu finden. Nach Verifikation des Patches gegen den „ORA-07445" beim „CHECK_PLUG_COMPATIBILITY" kehrte man zum ursprünglichen Migrationspfad (12.1 NonCDB nach 18.3 NonCDB nach 18.3 PDB) zurück und verwendet diesen bis heute.

## Erfahrungen mit Enterprise Manager Repository DB

Das Upgrade der Enterprise Manager Repository DB war das einzige Mal, dass die Phase 12.1 NonCDB nach 18.3 NonCDB an einem Problem scheiterte. In der letzten Phase (108) brach das Upgrade mit Perl-Fehlern ab und scheiterte laut Upgrade-Logfile an Listing 1.

```
SELECT TO_NUMBER('NONUPGRADED_TABLEDATA') FROM SYS.V$INSTANCE
*
ERROR at line 1:
ORA-01722: invalid number
```

*Listing 1*

Zu diesem Zeitpunkt versuchte man auch das Upgrade über die 12.1 PDB erneut, das ebenfalls im Zusammenhang mit „NONUPGRADED_TABLEDATA" scheiterte. In MOS fand sich dann Doc ID 2279497.1, die anweist, dass in diesem Fehlerfall vor dem Upgrade zwei Skripte auszuführen sind, um dieses Problem zu beseitigen. Nach einem Rückfall auf einen Stand der DB vor dem Upgrade und der Ausführung dieser Skripte lief das Upgrade 12.1 NonCDB nach 18.3 NonCDB erfolgreich durch.

## Erfahrungen mit dem Platz auf der Diskgruppe

Nach dem Upgrade der Datenbank von 12.1 NonCDB auf 18.3 NonCDB steht die Migration zu einer PDB auf dem Plan. An dieser Stelle scheiterte man auch mal, weil nicht beachtet wurde, dass im Zuge dieser Migration der Platzbedarf leicht steigt. In diesem Fall war die Diskgruppe für die Datafiles der jeweiligen DB bereits nahezu voll und so kam es direkt beim „CREATE PLUGGABLE DATABASE" zu einem „ORA-15041: diskgroup „..." space exhausted". Als Folge dessen gilt die erzeugte PDB als inaktiv und kann nicht gestartet werden. Doch da bereits Änderungen an den Datafiles stattgefunden haben, lässt sich auch die alte NonCDB nicht mehr starten. So blieb mir in diesen Fällen nur ein Restore auf einen Stand vor dieser Migration.

Im Moment des „CREATE PLUGGABLE DATABASE", das vor „noncdb_to_pdb.sql" aufzurufen ist, entsteht ein neues Tempfile unter neuem Pfad auf der Diskgruppe, während das alte Tempfile der NonCDB liegen bleibt. Dafür wird ein weiteres GB Platz benötigt. Ist dafür nicht genug Platz vorhanden, kommt es zu diesem Problem.

## Erfahrungen mit Database Services unter 18c

In der Umgebung mit Oracle Restart kommen natürlich Database-Services zum Einsatz. Das soll auch mit 18c weiterhin so sein. Doch beim ersten Versuch, nach dem Upgrade unter 18c einen Service per „srvctl add service" anzulegen, kam es zum Fehler „CRS-2918: Authorization failure: operating system group 'oper' does not exist on server '...'". Gemäß Dokumentation *(siehe „https://docs.oracle.com/en/database/oracle/oracle-database/18/cwlin/standard-oracle-database-groups-for-database-administrators.html#GUID-0A789F28-169A-43D6-9E48-AAE20D7B0C44")* sei die „OSOPER"-Rolle nur optional festzulegen, woraus man schloss, dass dies ebenfalls für die Existenz der Betriebssystem-Gruppe „oper" gilt. Ein Service Request kam ebenfalls zu dem Schluss, dass dies ein Fehler in der Dokumentation sei. Das reine Anlegen der Gruppe „oper" genügte, um den Fehler aus der Welt zu schaffen,

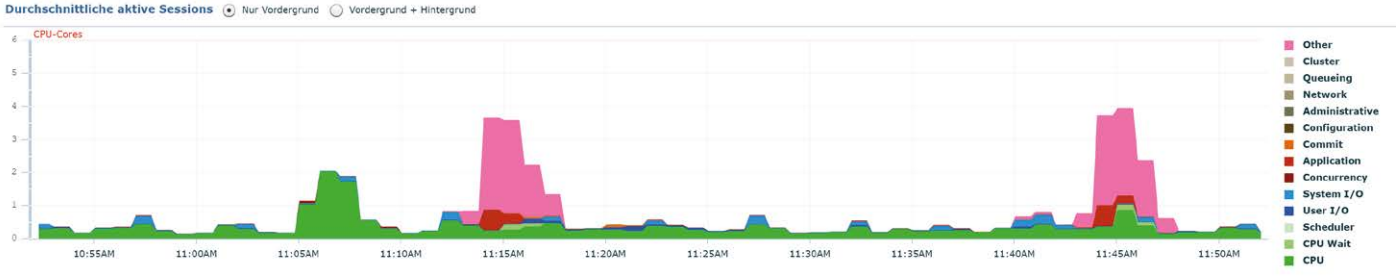



*Abbildung 1: Wait-Spitzen alle dreißig Minuten*

TRIOLOGY { Agility for your business.


# Oracle APEX- und PL/SQL-Entwickler (m/w/d)

## TRIOLOGY GmbH

Agile Entwicklungsmethoden, exzellente Code-Qualität und aktuelle Technologien – dafür steht TRIOLOGY.
Gestaltungskraft, Expertise und Leidenschaft zeichnen unsere 100 Mitarbeiter aus. Sie auch? Dann wachsen Sie mit uns und den Herausforderungen, nutzen Sie Ihren Freiraum und werden Sie Teil unseres engagierten Expertenteams! We need your talent.

## Unser Angebot

Bei TRIOLOGY erwartet Sie eine herausfordernde und vielseitige Tätigkeit in einem engagiertem Team mit attraktiven Gehaltskonditionen, flachen Hierarchien und einem Miteinander mit „Wir“-Gefühl. Regelmäßige Weiterbildungsmöglichkeiten und Zusatzangebote wie z.B. Physiotherapie, Englischunterricht sowie Events schaffen das ideale Umfeld für Ihren persönlichen Erfolg.

## Ihre Herausforderung

- Selbstständiges Betreuen und Beraten von Projekten und Kunden
- Analysieren, Konzipieren, Entwickeln und Testen von modernen Webanwendungen und Datenbanklösungen mit Oracle Application Express (APEX) und PL/SQL in agilen und klassischen Projekten
- Modellieren und integrieren von Datenbanken und Webanwendungen in Enterprise-Architekturen
- Mitwirken bei der Gestaltung und Weiterentwicklung unseres Kompetenzbereichs

## Das bringen Sie mit

- Studium oder Ausbildung mit IT-Bezug sowie Projekterfahrung Erfahrung in der Entwicklung komplexer Webanwendungen mit Oracle Application Express (APEX) und PL/SQL
- Sehr gute Kenntnisse in Internettechnologien, insbesondere HTML, CSS und JavaScript
- Gute Kenntnisse im Bereich SQL und Datenmodellierung Lösungs- und kundenorientierte Arbeitsweise sowie sicheres Auftreten im Projekt
- Sehr gute Deutschkenntnisse sind ein Muss
- Reisebereitschaft innerhalb Deutschlands

Senden Sie uns Ihre Bewerbung gern über das Online-Bewerbungsformular oder **jobs@triology.de**. Für Fragen steht Ihnen unser Team Human Resources auch gern telefonisch unter **+49 531 23528-52** zur Verfügung. Besuchen Sie uns online unter **triology.de** oder bei

auch ohne dieser Gruppe einen User zuzuweisen oder sie mit der Rolle „OSOPER" zu verknüpfen.

## Erfahrungen mit wiederkehrenden Wait-Spitzen

Nach kurzer Zeit des Betriebs einiger PDBs in einer CDB fiel auf, dass es alle dreißig Minuten zu Wait-Spitzen von „Sync ASM Rebalance" und „enq: RC - Result Cache: Contention" kommt *(siehe Abbildung 1)*. Diese werden vom Enterprise Manager Agent (hier Version 13.3.0.0.0) verursacht, der mit mehreren Queries die Tablespace-Usage ermitteln möchte und dazu unter anderem „cdb_tablespace_usage _metrics" abfragt. Auch ein manuelles „select * …" auf dieser View verursacht eine kleinere Wait-Spitze für diese Waits.

Die PDBs, die bisher migriert wurden und auf dieser CDB laufen, scheinen darunter für den Moment nicht zu leiden. Jedoch verursachen sie kaum signifikante Last und so ist es denkbar, dass diese Einschätzung bei höherer Last und komplexeren PDBs nicht mehr gilt. Für den Moment ist in dem Service Request dazu noch kein Bug geöffnet worden.

Nach einiger Zeit kam der Support zu dem Schluss, dass der Underscore-Parameter „_use_cached_asm_free_space" auf „TRUE" zu setzen ist, um das Problem zu beseitigen. Es handelt sich um eine Wiederkehr von Bug 22243217, der zwar mit 12.1 korrigiert wurde, mit 12.2 aber zurückkehrt. Unter 18.3 beseitigen das Setzen dieses Parameters und ein Datenbank-Neustart das Problem.

## Erfahrungen mit Core Dump bei Diskgroup-Dependency-Anpassung

Im Zuge der Migration der NonCDBs in PDBs auf einer CDB findet in diesem Fall bei jeder Migration einiges an Wartungsarbeit rund um die ASM-Diskgruppen statt (zwecks Erhöhung der Redundanz). Als Nebenwirkung wurde die Liste der Diskgruppen, von der die CDB-Ressource in Oracle Restart abhängig zu sein scheint, um viele Diskgruppen erweitert, zu denen nun keinerlei Abhängigkeit mehr besteht. Sofort war klar, dass dies bei einem Neustart der CDB Probleme verursacht, weil eben die Diskgruppen ohne echte Abhängigkeit auch nicht mehr mountbar sind.

```
[oracle@sol044 ~]$ srvctl modify database -d CDB1SOL044 -diskgroup DG_
DATA_[…],[…]_MIS_DS
#
# A fatal error has been detected by the Java Runtime Environment:
#
# SIGSEGV (0xb) at pc=0x00007f5396ca29c8, pid=18219,
tid=0x00007f53d2a83700
[…]
# Problematic frame:
# C [libhasgen18.so+0x1119c8] clsrteAddListArg+0x270
[…]
#
```

*Listing 2*

Zusätzlich gibt es in der Umgebung Diskgruppen mit identischen Namen, die auf verschiedenen Servern aktiv sind. Auf dem einen Server mit den CDBs kam es so zu Namenskonflikten mit Diskgruppen in der Abhängigkeitsliste und Diskgruppen von NonCDBs, die zu migrieren sind, deren Störungspotenzial nicht bekannt war. Die Liste musste also gekürzt werden. Ein Aufruf von „srvctl modify database" zur Anpassung/Kürzung der Diskgruppen-Liste endete jedoch wegen eines „SIGSEGV" in „libhasgen18.so" in einem Core Dump *(siehe Listing 2)*.

Wie sich in einem Service Request zeigte, verträgt „srvctl modify database" für die Liste mit Diskgruppen keine sonderlich lange Liste: Die Grenze liegt zwischen acht Diskgruppen mit 150 Zeichen und neun Diskgruppen und 167 Zeichen, ab der es zu diesem Fehler kommt. Die Bugs zu diesem Problem lauten 24590494, 28642469 und 27588725. Bisher (Stand 14. 11. 2018) ist keiner davon korrigiert. Als Workaround half, dass man die Diskgruppen ohne echte Abhängigkeit zumindest stoppen/unmounten konnte, ohne dass die CDB sich davon stören lässt. Dann kann man im Hintergrund die LUNs tauschen/ändern und eine andere Diskgruppe mit dem gleichen Namen wieder mounten.

Die einzige Möglichkeit zum erfolgreichen Kürzen dieser Abhängigkeitsliste ist, sie stärker zu kürzen als nötig. Laut Support soll das sogar im Betrieb möglich sein. Es wurde jedoch nur durchgeführt, als die CDB offline war. Dazu hat man per „srvctl modify database" die Liste auf drei Diskgruppen reduziert. Im Zuge des Restart der CDB und ihrer PDBs haben sich die wirklich gültigen Abhängigkeiten von selbst regeneriert. Alternativ hätte man mit dem „-nodiskgroup"-Parameter die Liste auch gänzlich zurücksetzen können. Später setzte der Support dann noch nach, dass Manipulationen an dieser Diskgruppen-Liste nicht empfohlen sind. Einzig das gänzliche Zurücksetzen, zusammen mit einem Datenbank-Neustart, sei bei Bedarf empfohlen, da so alle echten Abhängkeiten neu gesetzt werden.

## Erfahrungen mit „SYSAUX"-Tablespace-Wachstum

Nach etwas Laufzeit der CDB mit immer mehr PDBs fiel ein sehr ungleichmäßiges Wachstum der „SYSAUX"-Tablespaces in den PDBs auf *(siehe Abbildung 2)*. Manche kommen mit rund 400 MB aus, während andere bereits mehr als das Zehnfache davon benötigen. Die Suche nach den Ursachen brachte zwei Ergebnisse: Vor allem der Optimizer Statistics Advisor, der mit 12.2 neu eingeführt wurde, ist für hohen Platzverbrauch im „SYSAUX"-Tablespace bekannt und kann bei Bedarf in die Schranken verwiesen werden (Doc ID 2305512.1). In meinem Fall geht ein Teil des hohen Platzverbrauches auch auf dessen Konto, weil er vereinzelt viele Findings zu falsch ermittelten Statistiken hat und sich diese durch tägliche Prüfung auf Dauer anhäufen.

Die andere Quelle für den Platzverbrauch ist der Verlauf von Optimizer-Statistiken über die letzten 31 Tage (Standardwert). Datenbanken mit vielen Objekten haben viele Statistiken und damit auch einen größeren Platzbedarf für deren Verlauf. Hier kann man bei Bedarf die Aufbewahrungsdauer reduzieren (Doc ID 329984.1 & Doc ID 452011.1). Das muss vor dem Upgrade beziehungsweise der Migration schon so gewesen sein, denn dieses Feature existiert seit 10g.

| CON_ID | TBS_NAME | USED_MB |
|---|---|---|
| 14 | SYSAUX | 6,481 |
| 6 | SYSAUX | 4,553 |
| 1 | SYSAUX | 3,069 |
| 28 | SYSAUX | 2,526 |
| 10 | SYSAUX | 2,290 |
| 16 | SYSAUX | 2,165 |
| 30 | SYSAUX | 1,846 |
| 9 | SYSAUX | 1,707 |
| 4 | SYSAUX | 1,700 |
| 12 | SYSAUX | 1,627 |
| 11 | SYSAUX | 1,557 |
| 18 | SYSAUX | 1,501 |
| 22 | SYSAUX | 1,468 |
| 13 | SYSAUX | 1,431 |
| 21 | SYSAUX | 1,327 |
| 7 | SYSAUX | 1,327 |
| 25 | SYSAUX | 1,304 |
| 27 | SYSAUX | 1,185 |
| 8 | SYSAUX | 1,021 |
| 3 | SYSAUX | 806 |
| 24 | SYSAUX | 752 |
| 19 | SYSAUX | 533 |
| 31 | SYSAUX | 470 |
| 32 | SYSAUX | 458 |
| 20 | SYSAUX | 456 |
| 26 | SYSAUX | 448 |
| 15 | SYSAUX | 429 |
| 23 | SYSAUX | 429 |
| 29 | SYSAUX | 425 |
| 5 | SYSAUX | 398 |
| 17 | SYSAUX | 378 |

*Abbildung 2: Die Größe unterschiedlicher „SYSAUX"-Tablespaces in einer CDB*

Ich kann keine Empfehlung zur Konfiguration der beiden Mechanismen abgeben. Für den Moment habe ich beide auf den Standardeinstellungen belassen und akzeptiere den Platzverbrauch.

## Grundsätzliche Beobachtungen

Bevor 18c für On-Premises veröffentlicht wurde, hat der Autor bereits erste Erfahrungen mit 12.2 gesammelt. Einige Beobachtungen davon gelten auch weiterhin unter 18c: Bei Upgrades früherer Versionen (etwa 11.2.0.4 nach 12.1.0.2) war es bereits nötig, eigene, in die DB JVM geladene JARs zu droppen. Erst nach dem Upgrade konnte man die JARs wieder laden. Das gilt auch weiterhin. Bis 12.1.0.2 wurden fünf JAR-Dateien aus dem Home geladen („jaxrpc-api.jar", „ejb.jar", „dms.jar", „jssl-1_1.jar" und „soap.jar"). Diese Dateien gehören jedoch seit 12.2 nicht mehr zum Lieferumfang. Wie sich zeigte, werden diese aber gar nicht mehr benötigt.

Beim Testen mit Multitenant unter 12.2 fiel auf, das die Views „[dba|cdb]_tablespace_ usage_metrics" die Undo-Tablespaces nicht mehr anzeigen, obwohl sie sehr wohl unter „[dba|cdb]_tablespaces" auftauchen. In einem Service Request wurde das als Bug 24361167 identifiziert. Die Korrektur sei komplex, weshalb ein Fix erst für 19.1 angekündigt wurde. In der Tat besteht das Problem unter 18.3 weiterhin.

Ebenfalls beim Testen mit Multitenant (noch unter 12.2) erschien die PDB-Cloning-Funktionalität von RMAN sinnvoll für den Fall, dass man für Diagnose- oder Datenrettungszwecke eine PDB mit einem früheren Stand in einer anderen CDB wiederherstellt, während jedoch das Original weiterläuft. Dafür schien „DUPLICATE DATABASE" zusammen mit „UNTIL TIME" geeignet *(siehe Listing 3)*.

Doch das schlägt immer mit einer merkwürdigen Fehlermeldung fehl: „RMAN-06019: could not translate tablespace name „pdb1:undotbs1"". Es spielt keine Rolle, ob man dabei mit Shared Undo oder Local Undo arbeitet. Ein Service Request dazu ergab, dass dieses Feature in 12.2 schlicht nicht unterstützt wird. Auch für 18.3 gilt das weiterhin. Ein Enhancement Request dafür läuft über Bug 27891119.

Die einzigen Alternativen sind der In-Place-PITR („point in time recovery") auf der Original-CDB, doch dann kann das Original nicht weiterlaufen. Weiterhin beschreibt Doc ID 2142675.1 genau diesen PDB-PITR auf einem anderen Server (und so in einer anderen CDB), doch er basiert auf dem PITR der gesamten CDB. Dazu kann man beim Restore noch eine Liste der gewünschten PDBs mitgeben, während jedoch der Recovery eine Negativ-Liste aller Tablespaces aller PDBs benötigt, die nicht wiederherzustellen sind. Das funktioniert zwar, ist aber extrem unhandlich und fehleranfällig. Entsprechend bleibt nur zu hoffen, dass über den Enhancement Request der „DUPLICATE für PDBs" künftig dazulernt.

Mit dem ersten Release-Update für 18c (18.4) gab es anfangs Probleme, dieses erfolgreich auf meine Grid-Infrastructure-Installationen anzuwenden. Die Installation von 18.4 schlug fehl, weil „opatchauto" die Rollback-Skripte von 18.3 nicht finden konnte. Dies war jedoch ein eigener Fehler: Bei dem neuen Installationsmodus von 18c enthält das Installations-Archiv bereits ein fertiges Home, dass an den Zielort zu kopieren oder dort zu entpacken ist. Hier gilt es sicherzustellen, dass dabei auch die versteckten Ordner „.patch_storage" und „.opatchauto_storage" am Zielort ankommen. Hier war das nicht der Fall und so war dieser Fehler vorprogrammiert.

## Fazit

Nach kleineren Anfangsschwierigkeiten laufen die Migrationen für den Moment so reibungslos, wie man gehofft hatte.

```
rman auxiliary sys/xxx
Duplicate database to 'auxcdb' pluggable database pdb1 noopen backup
location '/backup_aux' undo tablespace "pdb1:undotbs1" UNTIL TIME "TO_
DATE ('2018-03-08_10:26:32', 'YYYY-MM-DD_HH24:MI:SS')";
```

*Listing 3*

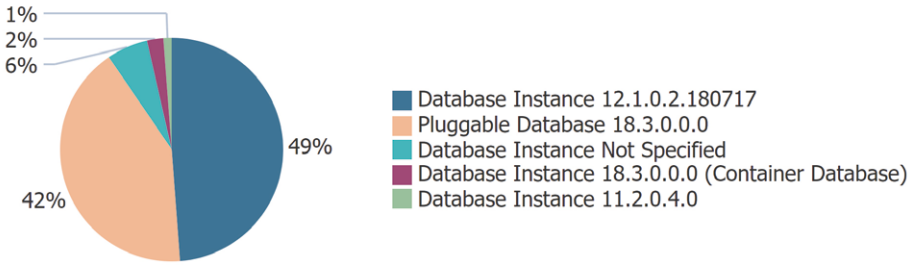



*Abbildung 3: Übersicht aus dem Enterprise Manager zum Migrationsfortschritt*

Von den 76 DB-Instanzen sind bisher 33 nach 18.3 PDB migriert *(siehe Abbildung 3)*. Natürlich gab es auch Probleme, doch dafür gab es immer zeitnah eine Lösung oder einen passablen Workaround. Für die Zukunft sieht man nur ein Problem am Horizont: Die Migration von Datenbanken mit Usern, deren Passwort noch auf 10g basiert. Es ist bekannt, dass einige Datenbanken davon betroffen sind. Ohne manuelle Änderungen des Passworts sind diese User nach der Migration ausgesperrt.

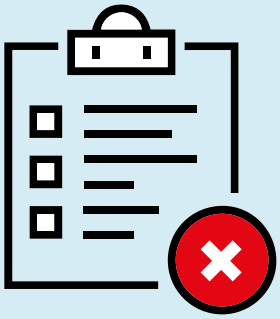
Im Betrieb zeigen sich noch ein paar Probleme, doch nichts davon ist ein Showstopper. Die meisten davon dürften zu lösen sein oder sind aktuell nur von theoretischer Natur. Während der Autor anfangs die Exklusivität von 18c nur für die Cloud noch als Problem empfand, scheint sich dies aber für die Stabilität und die Beseitigung von Kinderkrankheiten gelohnt zu haben. Er ist sehr optimistisch, dass sich die Investition in die Multitenant-Option sowie die damit verbundene Migration und Konsolidierung nach 18c auch langfristig als Erfolg darstellen wird.

Robert Ortel
robert.ortel@hypoport-systems.de

# DOAG Botschafter für Technologie 2018: Niels de Bruijn

Für sein großes Engagement innerhalb der DOAG wurde Niels de Bruijn, Themenverantwortlicher Oracle Application Express (Apex), auf der DOAG 2018 Konferenz + Ausstellung zum Botschafter für Technologie gekürt. Diese Auszeichnung, die ihm Robert Szilinski, DOAG-Vorstand und Leiter der Development Community überreichte, hat er sich verdient – da stimmte auch das Publikum zu, das den Preisträger mit einer La-Ola-Welle feierte, die mittlerweile zum Wahrzeichen der Apex-Community geworden ist.

Seine Leidenschaft für die Entwicklungsumgebung kommt nicht von ungefähr; auch beruflich ist Niels de Bruijn als Fachbereichsleiter für Apex bei der MT AG in Ratingen tätig. „Wer mit Apex gearbeitet hat, weiß, warum ich mich dafür einsetze: es funktioniert einfach. Aus meiner Sicht ist die Apex-Community einzigartig und ich bin stolz darauf, ein Teil davon zu sein.“, so der Konferenz- und Programmleiter der Apex Connect Konferenz, die er selbst ins Leben gerufen hat. Mit großem Erfolg: Im Jahr 2018 konnte die Veranstaltung ein neues Rekordhoch von 350 Teilnehmern aus 16 verschiedenen Ländern verzeichnen. De Bruijns Ziel ist es, dazu beizutragen, die Software-Entwicklungsumgebung auch in Deutschland noch populärer zu machen. De Bruijn freut sich über die Auszeichnung, die „für mich eine Bestätigung für meine, anscheinend gute, Arbeit in der Community ist“, so der frisch ernannte Botschafter.



# Nologging Objects von 11g bis 18c

Timo Giese, Fiducia & GAD IT AG

Nologging Operations und Objects begegnen früher oder später jedem Datenbank-Administrator. Dies äußert sich dann, wenn die Oracle-Fehlermeldung „ORA-26040: Data block was loaded using the NOLOGGING option“ im Alert-Log und/oder in den Applikationslogs auftaucht; meist in Kombination mit einem Anruf der Applikations-Administratoren dahingehend, dass die Anwendung ein Problem hat. Dieser Artikel zeigt, was es mit Nologging Objects und -Operationen auf sich hat, und stellt Methoden zur Anzeige, Diagnose, Reparatur und Verhinderung vor. Dabei wird auch auf Neuerungen und Verbesserungen im Umgang mit Nologging Objects bis hin zu 18c eingegangen.

Nologging-Operationen sind wenig protokollierte Transaktionen in der Datenbank. Dabei werden nur minimale Informationen über die vonstattengehende Transaktion in den Redologs protokolliert. Dadurch ist es möglich, die Daten schneller in die Tablespaces beziehungsweise Datenfiles einzufügen. Dieses Verfahren hat auch seine Kehrseite und kann unter Umständen zu Problemen beim Recovery einer Datenbank führen. Dies ist dadurch bedingt, dass keine vollständigen Redolog-Informationen vorhanden sind.

Eine weitere Problematik wird erst sichtbar, wenn die Applikation nach dem Recovery wieder online ist: Das Recovery wurde erfolgreich (ohne Fehler im RMAN-Log) durchgeführt. Die Datenfiles, die mit Nologging-Operation geladen wurden, sind im Zustand „unrecoverable“ und aus Sicht der Datenbank und Applikation korrupt. Ist dies der Fall, merkt das die Applikation sehr schnell: Ein einfacher „select“ auf die Daten produziert die Fehlermeldung „ORA-26040: Data block was loaded using the NOLOGGING option“.

## Einsatzbereiche

Trotz der im vorigen Abschnitt geschilderten Problematik gibt es gute Gründe, aus Sicht der Applikation auf diese Methode zu setzen. Nologging-Operationen werden überwiegend bei großen zu ladenden Datenmengen eingesetzt. Dies wiederum ist bedingt durch eine Zeitrestriktion, wann die Daten geladen beziehungsweise wann die Daten verarbeitet sein müssen und ein bestimmtes Ergebnis den Anwendern der Applikation zur Verfügung stehen muss. Überwiegend findet diese Methode Einsatz in Data-Warehouse- und Entscheidungs-Systemen (DSS). Darüber hinaus sie auch in Mixed-Workload-Systemen anzutreffen, die anteilig aus OLTP und Batch bestehen. Zu den Nologging Objects gehören:

- Tabellen
- Partitionen
- Indizes
- LOBs

Nicht jede Datenbank-Aktion führt zu Nologging-Objekten. Grundbedingung ist zuerst, dass das gewünschte Objekt als „Nologging Object“ angelegt wurde (etwa mit „alter table test nologging“). Alternativ ist die auszuführende Operation explizit mit dem Zusatz „nologging“ zu versehen. Folgende Operationen können „nologging“ ausgeführt werden:

- insert /*+ append */ into
- insert into <table> nologging
- create table <table_copy> as select
- alter table <table> move|split partition
- alter index <index> rebuild
- alter index <index> split|rebuild partition
- SQL-Loader direct load (sqlldr direct=true und “unrecoverable option”)

## Nologging-Objekte erstellen

Operationen werden „nologging“ ausgeführt, wenn die zugrunde liegenden Objekte als „nologging“ definiert sind (wie Tabelle). Eine Variante ist die Festlegung auf Tablespace-Ebene. Dies kann entweder beim Erstellen des Tablespace oder nachträglich festgelegt werden. Wichtig ist dabei, dass alle Objekte innerhalb des Tablespace das „nologging“-Flag bekommen, sofern nicht anderweitig beim Erstellen der Objekte explizit angegeben *(siehe Listing 1)*.

Die kleinste Stufe sind Objektebene-Tabelle, LOB-Segment, Index etc. Hier gilt, wie beim Tablespace, dass eine Änderung auf „nologging“ auch nach der Erstellung möglich ist *(siehe Listing 2)*. Für Indizes gilt dies nur bei Neuanlage, Split und Rebuild.

## Nologging-Operationen verhindern

Nologging-Operationen können in bestimmten Umfeldern nicht gewünscht sein, etwa in einem Data-Guard-Umfeld, bei dem sonst zusätzlich auf der Standby-Seite Inkonsistenzen auftreten. Im weiteren Verlauf dieses Artikels wird auf Neuerungen in der Oracle-Datenbank eingegangen, die eine Verbesserung dieser Problematik aufzeigen. Ein weiterer Grund kann eine Unternehmensrichtlinie sein, die Datenkonsistenz und Wiederherstellbarkeit über die Laufzeit zur schnellen Bereitstellung von Daten vorgibt.

In der Oracle-Datenbank gibt es Möglichkeiten, Nologging-Operationen zu unterbinden, sodass diese auch nicht unabsichtlich durch Entwickler oder Applikations-Administratoren verursacht werden können. Dies lässt sich global aktivieren; für die gesamte Datenbank werden mit „alter database force logging;“ alle Nologging-Operationen in Logging-Operationen mit vollständigem Redo-Record umgeschrieben. Zu beachten ist, dass Objekt-Definitionen mit dem Zusatz „nologging“ weiterhin angelegt werden können, dies jedoch beim Einfügen von Daten ignoriert wird.

Sollen Nologging-Operationen nur für bestimmte Bereiche in der Datenbank erlaubt sein (etwa Staging Area), kann dies auch auf Tablespace-Ebene festgelegt sein. Dazu werden alle Tablespaces, für die keine Nologging-Operationen erlaubt sind, mit „alter tablespace <tbs> force logging;“ in den „force logging“-Modus gesetzt.

Ist die Datenbank global auf „force logging“ gesetzt und man möchte dies später wieder rückgängig machen, ist dies ohne

```
Tablespace nologging erstellen:
   create tablespace staging_tbs nologging;

Tablespace anpassen:
   alter tablespace staging_tbs nologging;
```

*Listing 1*

```
Tabelle erstellen:
   create table tbl_nologging (id number) nologging;

Tabelle anpassen:
   alter table staging_table nologging;
```

*Listing 2*

```
select force_logging from v$database;
   FORCE_LOGGING
   -------------
   YES

select tablespace_name,force_logging from dba_tablespaces;
   TABLESPACE_NAME       FORCE_LOGGING
   --------------------  -------------
   TBS1                  NO
   TBS2                  YES
   TBS3                  NO
```

*Listing 3*

```
RMAN> validate check logical datafile 10;
…
List of Datafiles
==================
File  Status  Marked Corrupt  Empty Blocks  Blocks Examined  High SCN
----  ------  --------------  ------------  ---------------  ---------
10    OK      1308            9989          12800            18647556
   File Name: /u01/oradata/test.dbf
   Block Type Blocks Failing Blocks Processed
   ---------- -------------- ----------------
   Data       0              2621
   Index      0              0
   Other      0              190
```

*Listing 4*

```
select file#,block#,blocks,NONLOGGED_START_CHANGE#,NONLOGGED_END_CHANGE#
from v$nonlogged_block;

   FILE#   BLOCK#   BLOCKS  NONLOGGED_START_CHANGE#  NONLOGGED_END_CHANGE#
   -----   -------  ------  ------------------------  ---------------------
   10      267      13      18647507                  18647507
```

*Listing 5*

```
RMAN> report unrecoverable;
   File Type of Backup Required Name
   ---- ----------------------- -----------------------
   10   full or incremental     /u01/oradata/test.dbf
```

*Listing 6*

Weiteres mit dem SQL-Befehl „alter database no force logging“ möglich. Selbiges gilt für den Tablespace mit „alter tablespace <tbs> no force logging“.

Der Status, ob „force logging“ global oder auf Tablespace-Ebene aktiv ist, kann durch Abfrage der Views „v$database“ beziehungsweise „dba_tablespaces“ angezeigt werden *(siehe Listing 3)*. Für Tabellen und Indizes ist dies in der Spalte „LOGGING“ in den Views „dba_tables“ und „dba_indexes“ ersichtlich.

Ist es unverzichtbar, Nologging-Operationen durchzuführen, muss aus Gründen der Wiederherstellbarkeit gewährleistet sein, dass nach der Operation ein Backup der betroffenen Objekte durchgeführt wird. Dies ist die einzige Möglichkeit, eine erfolgreiche Wiederherstellung mit allen Daten zu garantieren.

## Nologging-Operationen feststellen

Wenn Objekte „nologging“ definiert sind, heißt das bei Weitem nicht, dass alle Operationen „nologging“ erfolgen. Folglich muss zuerst festgestellt werden, welche Bereiche der Datenbank von Nologging-Operationen betroffen sind. Oracle bietet in den unterschiedlichen Datenbank-Versionen verschiedene Möglichkeiten an, dies festzustellen.

Das Utility DB-Verify „dbv” kann mit „dbv file='<pfad_zu_datenfile>/data01.dbf' userid=sys/pw“ Nologging-Operationen auf Datenfile-Ebene in Versionen vor 10g bis hin zu 18c (nur NON-CDB) feststellen. Jeder gefundene Nologging Change wird mit den Meldungen „DBV-00200“ und „DBV-00201“ quittiert.

Ab 10.2.0.5 werden Nologging-Objekte mit RMAN und dem Befehl „validate check logical database“ festgestellt. Dabei werden alle Tablespaces und Datenfiles überprüft. Es ist ebenfalls möglich, dies auf einzelne Objekte zu beschränken, etwa auf ein Datenfile durch „validate check logical datafile 3“ *(siehe Listing 4)* oder aber auf Tablespaces. Dabei sind Informationen über korrupte Blöcke in 10g und 11c in der View „v$database_block_corruption“ in der Spalte „CORRUPTION_TYPE“ mit dem Flag „NOLOGGING“ versehen.

Ab der Version 12c gibt es aus Kompatibilitätsgründen diese View weiterhin, jedoch wird sie nicht mehr mit Informationen bezüglich Nologging-Operationen befüllt. Alle Informationen gehen in die View „v$nonlogged_block“ *(siehe Listing 5)*. Zusätzlich wird im Datenbank-Alert-Log ein Eintrag für jeden gefundenen Nonlogged-Block geschrieben (etwa „Finished Nonlogged Block Replacement recovery(validate) on file 5. 400 blocks found“).

Ab 11g steht ein weiterer RMAN-Befehl „report unrecoverable“ zur Verfügung, der korrupte Datenfiles anzeigt. Dieser bietet in seiner Auflistung den Hinweis zur Verhinderung einer späteren Recovery-Problematik: Ein volles oder inkrementelles Backup hilft, den ORA-26040-Fehler zu vermeiden *(siehe Listing 6)*. Datenbanken in Version 12.2 und neuer bieten mittels RMAN weitere Befehle zur Detektion: „validate database nonlogged block“ und „validate datafile nonlogged block“.

## Reparatur

Eine Reparatur der korrupten Daten ist generell nicht möglich, lediglich die Korruption in den Datenblöcken lässt sich beseitigen. Ist die Korruption in einem „nologging“-Index aufgetreten, kann dieser relativ einfach mit dem SQL-Befehl „alter index <idx> rebuild [online]“ neu erstellt werden.

Ob ein Index aufgrund einer Nologging-Operation korrupt ist, kann nicht aus den Views „dba_indexes“ beziehungsweise „dba_ind_partitions“ ermittelt werden, da der Status des jeweiligen Index weiterhin als „VALID“ angezeigt wird. Hier hilft die Abfrage der View „v$nonlogged_block“ (12c) beziehungsweise „v$data_block_corruption“ (11g) mit der zusätzlichen Ausgabe der Spalte „object_id“. Mit dieser Information lässt sich der betroffene Index mit der View „dba_objects“ ermitteln.

Falls Tabellen „nologging“ geladen und beim Recovery die zugrunde liegenden Datenfiles korrupt wurden, sind die geladenen Daten unweigerlich nicht wiederherstellbar. Ein erneutes Laden der Daten durch Applikationsprozesse ist die Folge. Zuvor müssen die korrupten Blöcke in der Datenbank repariert werden. Das Vorgehen gliedert sich in zwei Schritte: Zuerst wird mit dem Package „DBMS_REPAIR“ die Datenbank angewiesen, alle korrupten Blöcke des Tabellenobjekts zu ignorieren.

Im zweiten Schritt wird die Tabelle mit allen intakten Blöcken verschoben *(siehe Listing 7)*. Zu beachten ist, dass die ursprünglichen Blöcke weiterhin korrupt, jedoch nicht mehr ein Teil des Tabellen-Objekts sind und ab jetzt zu den freien Blöcken gehören. Diese Blöcke wiederum werden bei weiteren Inserts herangezogen, jedoch vor der Verwendung neu initialisiert; dadurch wird die Korruption vollständig beseitigt. Weitere Details zur Reparatur stehen in der MOS-Note 794505.1.

Sind LOB-Objekte im Spiel, ist die Prozedur zur Reparatur deutlich aufwendiger. Zuerst muss das betroffene LOB-Segment genau ermittelt werden. In einem zweiten Schritt wird mit einer PL/SQL-Routine (Skript und Verfahren sind in MOS-Note 293515.1 genau beschrieben) die Information, welche Zeilen in den LOBs korrupt sind, in eine Hilfstabelle geschrieben. Danach werden die korrupten Blöcke mithilfe der Information aus der Hilfstabelle geleert *(siehe Listing 8)*.

```
BEGIN
   DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
   SCHEMA_NAME => '&schema_name',
   OBJECT_NAME => '&table_name',
   OBJECT_TYPE => dbms_repair.table_object,
   FLAGS => dbms_repair.SKIP_FLAG);
   END;
   /

   alter table &table_name move [online];
```

*Listing 7*

```
 update &table_owner.&table_with_lob
      set &lob_column = empty_blob()
      where rowid in (select row_id from bad_rows);
```

*Listing 8*

Im Anschluss daran sollte das LOB-Segment in einen anderen Tablespace verschoben werden. Im Zuge der vorherigen Korrektur wird ausschließlich der Pointer auf die fehlerhaften Blöcke umgebogen. Ohne Verschieben des LOB-Segments können bei einem erneuten Insert von Daten die korrupten Blöcke wieder herangezogen werden, was zu einem erneuten Abbruch der Aktion führt.

Ein besonderes Augenmerk ist auch beim Einsatz von Data Guard geboten. Eine Reparatur erfolgt durch ein Incremental- oder Full-Backup von der Primärseite (MOS-Note 958181.1). Ist die Datenbank-Version 12c oder neuer, gibt es die Möglichkeit, die Standby-Datenbank mittels RMAN direkt über den Service-Namen der Primary zu korrigieren (MOS-Note 1987763.1). Dies vereinfacht einen Teil der Schritte, ist jedoch weiterhin aufwendig. Aus diesen Gründen ist es nicht empfehlenswert, Nologging in Data-Guard-Umfeldern einzusetzen, zumindest bis Version 12.1.

Eine Active-Data-Guard-Lizenz vorausgesetzt, gibt es in Version 12.2 ein Feature, um die Korruption zu vermeiden. Dabei werden die Informationen aus der View „v$nonlogged_block" der Primärseite in das Controlfile der Standby-Datenbank übertragen. Ist die Nologging-Operation erfolgreich auf der Primärdatenbank durchgeführt, kann mit dem RMAN-Befehl „recover database nonlogged block;" auf der Standby-Datenbank die Konsistenz wiederhergestellt werden. Dies stellt einen Ansatz dar, um Nologging auch in Data-Guard-Umfeldern einzusetzen. Wichtig ist dabei, die Aktion in den Prozess des Datenladens mit einzubinden.

## 18c-Verbesserungen

In 18c gibt es Verbesserungen der Nologging-Funktionalität für Data Guard auf Engineered Systems (Exadata/Oracle Database Appliance) und in der Oracle-Cloud (ab EE-ES). Zusätzlich zu diesen Restriktionen ist auch für diese die Active-Data-Guard-Lizenz erforderlich. Sind die Voraussetzungen erfüllt, sind zwei neue Modi für Data Guard möglich:

- Standby Nologging for Data Availability
- Standby Nologging for Load Performance

In der ersten Variante liegt der Fokus auf Verfügbarkeit und Wiederherstellbarkeit. Dabei werden für den Data-Guard-Transport alle Nologging-Operationen in der Primär-Instanz wie gehabt „nologging", beim Transport zur Standby-Datenbank als „logging" ausgeführt. Dafür ist der Commit verzögert, bis alle Standby-Datenbanken den Empfang quittiert haben. Aktivieren lässt sich dieser Modus mit „alter database set standby nologging for data availability".

Im zweiten Fall ist der Sachverhalt etwas anders. Es wird versucht, den Datenstrom so schnell wie möglich zur Standby-Datenbank zu transferieren. Ist dies aufgrund von Netzwerk-Engpässen nicht möglich, wird der Transfer gestoppt und der Standby-Datenbank fehlen wie in vorigen Versionen die Daten der Nologging-Operationen. Tritt diese Situation ein, werden im Zuge des Managed Recovery die Daten nachgezogen. Dieser Modus lässt sich mit dem Befehl „alter database set standby nologging for load performance" aktivieren.

Wer den Mittelweg aus Nologging und Einsatz von Data Guard sucht, hat diesen mit dem Modus „Data Availability" gefunden. Der zweite Modus „Load Performance" verspricht die gleiche Performance in Data-Guard-Systemen wie auf einer Nicht-Data-Guard-Datenbank. Der Best-Effort-Ansatz birgt jedoch einige Fallstricke im täglichen Betrieb: Monitoring und Diagnose können unter Umständen schwierig werden, wenn besagte Netzwerk-Engpässe auftreten. Deshalb ist es auch weiterhin unverzichtbar, das Backup der Datenbank mit den Datenlade-Prozessen der Applikation zu koppeln.

## Flashback Database und Nologging

Nologging-Ladevorgänge lassen sich mit den Flashback-Mechanismen der Datenbank absichern. Es bietet sich an, einen Restore Point vor der Ladeaktion zu setzen. Bricht der Ladevorgang ab oder kommt es zu einem Crash der Datenbank oder des Servers, können die Nologging-Ladevorgänge schnell mit dem Zurückgehen auf den Restore Point zurückgesetzt werden. Im Anschluss lassen sich die Daten erneut laden. Am Ende des Ladeprozesses ist es weiterhin erforderlich, ein Backup zu erstellen, um eine Korruption der Datenbestände bei einem späteren Recovery zu vermeiden.

## Datapump und Nologging

Werden Daten mit Datapump in die Datenbank geladen, besteht ab Version 12.1 die Möglichkeit, den Ladevorgang im Nologging Mode durchzuführen. Dies wird mit dem Parameter „TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y" für alle Objekte oder mit „TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y:TABLE" für bestimmte Objekte aktiviert. Für die zu importierenden Objekte werden vor dem Laden die Objekte auf „nologging" gesetzt und nach erfolgreichem Laden wieder auf „logging". Auch bei diesem Modus ist unbedingt im Anschluss ein Backup der Datenbank beziehungsweise der betroffenen Objekte durchzuführen.

## Fazit

Oracle hat einiges für das Handling von Nologging-Operationen über die Versionen hinweg getan. Weiterhin gilt die Devise: Wenn nicht unbedingt notwendig, sollte Nologging vermieden werden. Ist dies nicht möglich, muss dies für alle am Betrieb der Applikation beteiligten Personen transparent sein, um einen Datenverlust zu vermeiden. Essenziell dabei ist auch die Verknüpfung der Applikationsprozesse mit den Backup-Routinen der Datenbank.

Den Verbesserungen in Data-Guard-Systemen bis hin zu den Features in 18c zum Trotz bleibt es weiterhin schwierig, diese Umgebungen vollumfänglich effizient zu managen. Die Nichtverfügbarkeit der Features außerhalb von Engineered Systems beziehungsweise der Oracle Cloud schränken den Nutzen für die Allgemeinheit weiter ein.

Wer ohnehin Konsistenz als oberste Maxime hat, sollte seine Datenbanken auf Force Logging stellen, damit auch unbeabsichtigte und ungeplante Nologging-Operationen nicht stattfinden können.

Timo Giese
timo.giese@fiduciagad.de

# Polymorphe Tabellen-Funktionen in Oracle 18c

Andrej Pashchenko, Trivadis GmbH

Mit der Version 18c stellt Oracle seine Implementierung für die im „ANSI SQL 2016"-Standard definierten polymorphen Tabellen-Funktionen vor. Die Funktionalität ist dafür gedacht, generische benutzerdefinierte Erweiterungen zu entwickeln, die – auf die einfachste Weise in SQL aufrufbar – beliebige Input-Tabellen verarbeiten können. Dieser Artikel gibt Einblicke in die völlig neue Art des Zusammenspiels von SQL und PL/SQL in der Datenbank.

Tabellen-Funktionen sind in der Oracle-Datenbank schon seit der Version 8i bekannt. Aus diesen kann man in der „FROM"-Klausel einer SQL-Abfrage wie aus einer Tabelle selektieren. Eine wesentliche Einschränkung für richtig generische Anwendungsfälle ist es allerdings, dass der Rückgabe-Datentyp bereits bei der Definition der Funktion angegeben werden muss. Auch die Übergabe der Daten in die Funktion mithilfe von Cursor-Parametern ist nicht wirklich einfach und flexibel.

Polymorphe Tabellen-Funktionen (PTF) haben diesen Nachteil nicht. Sie heißen nicht umsonst „polymorph", weil sie unterschiedliche Eingabe- und Ausgabe-Datenstrukturen unterstützen. Dabei wird die Ausgabestruktur erst zur Laufzeit abhängig von Input und Parametern festgelegt.

Oracle stellt dem PTF-Entwickler eine Infrastruktur zur Verfügung. Diese besteht aus einem neuen Package „DBMS_TF" mit benötigten Datentypen und Hilfsprozeduren sowie einem neuen Pseudo-SQL-Operator „COLUMNS()" für die Übergabe der Spaltenlisten.

Damit die Vorstellung der technischen Details nicht zu trocken wirkt, versuchen wir den Einstieg mit einer konkreten Beispielaufgabe. Für eine beliebige Tabelle sollen in jedem Datensatz alle Spalten – bis auf eine definierte Spaltenliste – zusammengeführt und durch Semikolon getrennt in einem neuen Feld dargestellt werden *(siehe Abbildung 1 und Listing 1)*.

## Row- und Table-Semantic PTF

Wenn genau ein aktuell zu verarbeitender Input-Datensatz ausreichend ist, um einen oder je nach Aufgabe auch mehrere entsprechende Ergebnis-Datensätze vollständig zu produzieren, spricht man von einer „Row-Semantic PTF". Diese Art eignet sich

gut für das Hinzufügen berechneter Spalten, Umformatieren des Datensatzes oder Ausgabe in bestimmten Formaten, Datensatzreplikation, Transponieren der Spalten etc.

Kann man das Ergebnis nur aus der Betrachtung mehrerer zusammenhängender Datensätze ableiten, hat man es mit „Table-Semantic PTF" zu tun. In diesem Fall lassen sich die Input-Daten optional partitionieren und sortieren. Mit Table-Semantic PTF lassen sich benutzerdefinierte Aggregat-/Fensterfunktionen entwickeln. Für unsere Aufgabe ist eine Row-Semantic PTF die richtige Wahl.


*Abbildung 1: PTF stellt mehrere Felder als CSV dar*

```
CREATE TABLE  t (A NUMBER, B NUMBER, C NUMBER, V VARCHAR2(10));
INSERT INTO t (A,B,C,V) VALUES (1, 2, 3, 'ONE');
INSERT INTO t (A,B,C,V) VALUES (4, 5, 6, 'TWO');
INSERT INTO t (A,B,C,V) VALUES (7, 8, 9, 'THREE');
COMMIT;
```

*Listing 1: Definition der Testtabelle*

## Interface-Methoden

Jede PTF benötigt ein Implementierungs-Package mit definierten Interface-Methoden: „DESCRIBE", „FETCH_ROWS", „OPEN" und „CLOSE". Nur die „DESCRIBE"-Methode ist Pflicht. Diese wird beim Parsen der SQL-Abfrage implizit aufgerufen; ihre Aufgabe besteht darin, das Format der Rückgabe-Tabelle zu definieren und Meta-Informationen der Spalten zu vergeben, die den Fluss der Daten in der PTF steuern. „FETCH_ROWS" ist die Methode, die die Input-Datensätze verarbeitet und das Ergebnis produziert. „OPEN" und „CLOSE" ermöglichen beispielsweise die Initialisierung zusätzlich benötigter Zustandsvariablen vor der Ausführung und die Ressourcen-Freigabe nach der Ausführung und sind oft gar nicht erforderlich.

*Listing 2* zeigt die Package-Spezifikation sowie die Definition der PTF für unsere Aufgabe. Es ist möglich, die Funktion sowohl auf Schema-Ebene, wie in unserem Beispiel, als auch im Package selbst zu definieren. Abweichend zum ANSI-Standard SQL 2016, der auch PTF ohne (Leaf-PTF) oder mit mehreren Tabellen-Parametern

```
CREATE OR REPLACE PACKAGE my_ptf_package AS
FUNCTION describe (tab IN OUT dbms_tf.table_t
                 , cols2stay IN dbms_tf.columns_t
                 , new_col_name IN DBMS_ID DEFAULT 'AGG_COL')
         RETURN dbms_tf.describe_t;

PROCEDURE fetch_rows (new_col_name IN DBMS_ID EFAULT 'AGG_COL');
END my_ptf_package;
/
CREATE OR REPLACE  FUNCTION my_ptf
( tab         TABLE
, cols2stay   COLUMNS
, new_col_name IN DBMS_ID DEFAULT 'AGG_COL' )
RETURN TABLE PIPELINED ROW POLYMORPHIC USING my_ptf_package;
/
```

*Listing 2: Package-Spezifikation und Definition der PTF*

erlaubt, ist in Oracle genau ein Parameter vom Typ „TABLE" Pflicht. Beim Parsen werden die Meta-Informationen über die angesprochene Tabelle zum PL/SQL-Datentyp „DBMS_TF.TABLE_T" umgewandelt und an die „DESCRIBE"-Methode übergeben.

Um Spaltenlisten als Parameter in die Funktion zu übergeben, dienen Parameter vom Typ „COLUMNS". Beim Aufruf der PTF steht hier ein neu in 18c eingeführter Pseudo-Operator „COLUMNS()" zur Verfügung. Auch hier wird die Spaltenliste umgewandelt und an „DESCRIBE" als „DBMS_TF.COLUMNS_T" übergeben.

Wir werden einen „COLUMNS"-Parameter namens „COLS2STAY" verwenden, um die Liste der Spalten anzugeben, die nicht zusammengeführt werden sollen. Alle anderen skalaren Parameter wie der Name für die neue Spalte in unserem Beispiel sind sowohl für „DESCRIBE" als auch für „FETCH_ROWS" anzugeben.

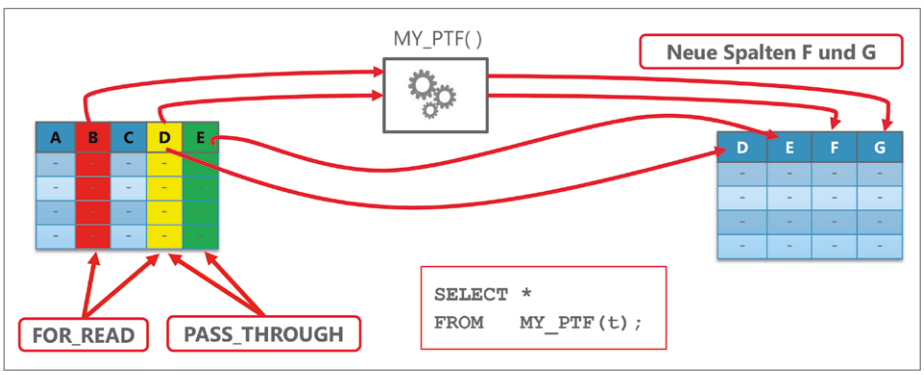



*Abbildung 2: „PASS_THROUGH"- und „FOR-READ"-Spalten*

## PASS-THROUGH- und FOR-READ-Spalten

Bevor wir uns mit der Implementierung auseinandersetzen, ist das Verständnis der Spalten-Eigenschaften „PASS-THROUGH" und „FOR-READ" sehr wichtig. Die Metadaten über die in der Abfrage referenzierte Tabelle inklusive Spaltenliste bekommen wir in der Methode „DESCRIBE" über den „IN/OUT"-Parameter vom Typ „DBMS_TF.TABLE_T". Hier können wir für jede Spalte angeben, ob diese ohne Änderung in das Ergebnis der Funktion

```
CREATE OR REPLACE PACKAGE BODY my_ptf_package AS
FUNCTION describe (tab IN OUT dbms_tf.table_t
                 , cols2stay IN dbms_tf.columns_t
                 , new_col_name IN DBMS_ID DEFAULT 'AGG_COL')
         RETURN dbms_tf.describe_t IS
BEGIN
   FOR I IN 1 .. tab.COLUMN.COUNT LOOP
       IF NOT tab.COLUMN(i).description.name MEMBER OF cols2stay THEN
           tab.column(i).pass_through := false;
           tab.column(i).for_read := true;
       END IF;
   END LOOP;
   RETURN dbms_tf.describe_t(
           new_columns =>
               dbms_tf.columns_new_t(
                   1 => dbms_tf.column_metadata_t(
                       name => new_col_name,
                       TYPE => dbms_tf.type_varchar2)));
END describe;
…
```

*Listing 3: Implementierung DESCRIBE*

weitergereicht werden soll. Die Eigenschaft „PASS_THROUGH“ ist dann „TRUE“.

Die Spalten, deren Inhalt wir zum Produzieren von neuen Spalten im Ergebnis brauchen, markieren wir mit „FOR_READ=TRUE“. Nur diese Spalten werden in der Methode „FETCH_ROWS“ sichtbar. Die beiden Eigenschaften schließen sich gegenseitig nicht aus: Das Zusammenspiel und der Datenfluss sind in *Abbildung 2* dargestellt.

Nun sind wir bereit, mit der Implementierung anzufangen. *Listing 3* zeigt die Funktion „DESCRIBE“ für unsere Aufgabe. Als Erstes gehen wir in der Schleife über alle Spalten der in der Abfrage verwendeten Tabelle (Parameter „TAB“) und prüfen, ob die jeweilige Spalte nicht auf der Liste der beizubehaltenden Spalten ist (Parameter „COLS2STAY“). Solche Spalten möchten wir zusammenführen. Daher lassen wir sie nicht durch („PASS_THROUGH=FALSE“) und markieren sie mit „FOR_READ=TRUE“.

Für die restlichen Spalten greifen die Defaults „PASS_THROUGH = TRUE“ und „FOR_READ = FALSE“. Diese Informationen ändern wir direkt an den Record-Strukturen des „IN/OUT“-Tabellen-Parameters und geben sie dann über diesen Parameter an die Datenbank zurück.

Wenn eine PTF neue Spalten ins Ergebnis einfügen soll, muss die „DESCRIBE“-Methode eine Liste davon als Tabellen-Datentyp „DBMS_TF.DESCRIBE_T“ zurückliefern. Wir verwenden dabei eine 18c-Neuerung, die sogenannten „qualified expressions“: Es wird eine neue Spalte vom Typ „VARCHAR2“ und mit dem über Parameter übergebenen Namen definiert.

## FETCH_ROWS

Zur Laufzeit findet die ganze Verarbeitung innerhalb der Methode „FETCH_ROWS“ statt. Diese wird – möglicherweise mehrfach und parallel – von der Datenbank aufgerufen. Als Entwickler bekommt man innerhalb dieser Prozedur nur einen Rowset zu sehen. Wie groß der ist und wie viele es gibt, entscheidet die Datenbank. Man arbeitet immer mit einem aktiven Rowset zu einem Zeitpunkt. Die Struktur eines Rowset (Datentyp „DBMS_TF.ROW_SET_T“) ist in *Abbildung 3* dargestellt.

Der Datentyp ist eine PL/SQL-Tabelle, deren Elemente Records vom Typ „COLUMN_DATA_T“ sind, einmal für jede Spalte. Dieser Record beinhaltet wiederum einige Meta-Informationen zu der Spalte (Name, Datentyp) und sechzehn Collec-

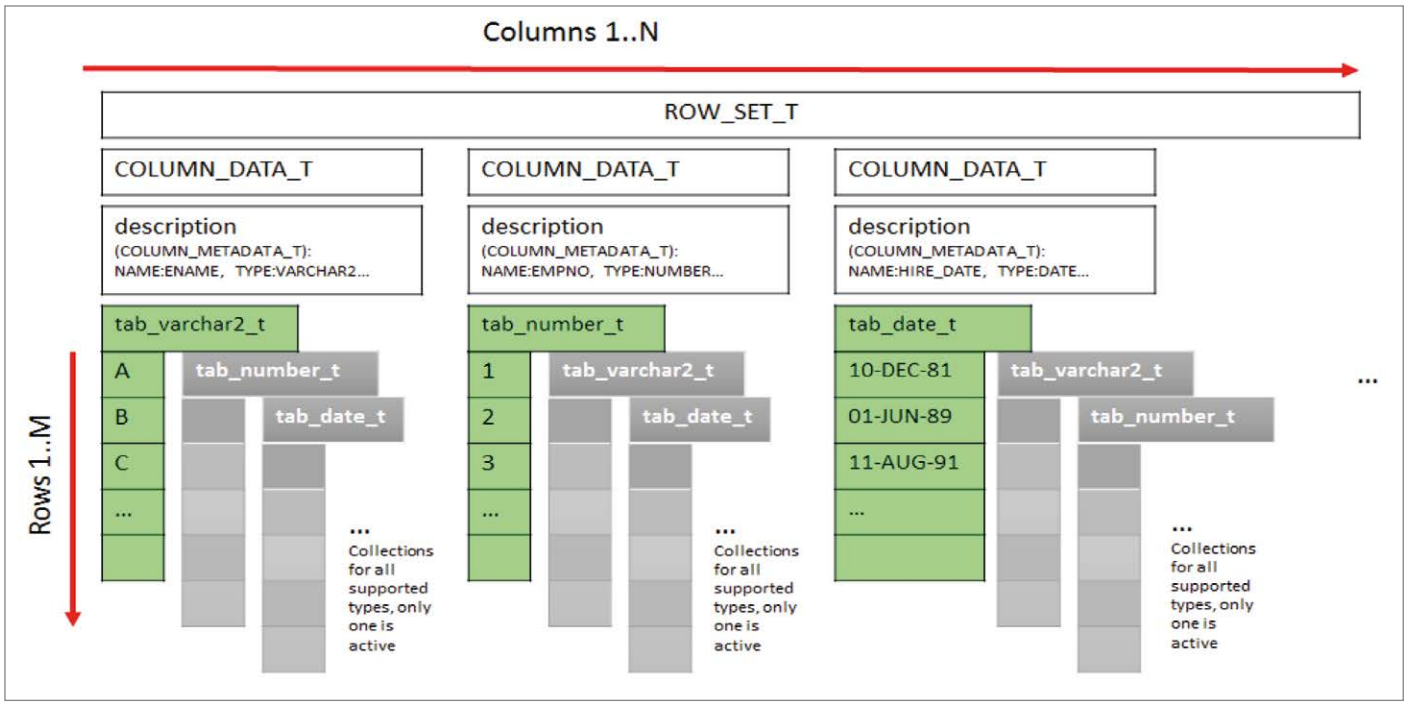



*Abbildung 3: Die Struktur vom Rowset*

```
…
PROCEDURE fetch_rows IS
   rowset dbms_tf.row_set_t;
   colcnt PLS_INTEGER;
   rowcnt PLS_INTEGER;
   agg_col  dbms_tf.tab_varchar2_t;
BEGIN
      dbms_tf.get_row_set(rowset, rowcnt, colcnt);
      FOR r IN 1..rowcnt LOOP
          agg_col(r) := '';
          FOR c IN 1..colcnt LOOP
          agg_col(r) := agg_col(r)||';'
                 ||DBMS_TF.COL_TO_CHAR(rowset(c), r);
          END LOOP;
          agg_col(r) := ltrim (agg_col(r) ,';');
      END LOOP;
      dbms_tf.put_col(1, agg_col);
END fetch_rows;
END my_ptf_package;
/
```

*Listing 4: Implementierung von „FETCH_ROWS“*

```
SQL> SELECT * FROM my_ptf(t, COLUMNS(A));

         A AGG_COL
---------- --------------------------------------------------
         1 2;3;"ONE"
         4 5;6;"TWO"
         7 8;9;"THREE"

SQL> SELECT * FROM my_ptf(t, COLUMNS(A,B));

         A          B AGG_COL
---------- ---------- ----------------------------------------
         1          2 3;"ONE"
         4          5 6;"TWO"
         7          8 9;"THREE"

SQL> SELECT * FROM my_ptf(scott.emp, COLUMNS(empno));

     EMPNO AGG_COL
---------- --------------------------------------------------
      7369 "SMITH";"CLERK";7902;"17-DEC-80";800;;20
      7499 "ALLEN";"SALESMAN";7698;"20-FEB-81";1600;300;30
      7521 "WARD";"SALESMAN";7698;"22-FEB-81";1250;500;30
```

*Abbildung 4: Polymorphismus in der Praxis*

```
…
PROCEDURE fetch_rows IS
   rowset dbms_tf.row_set_t;
   rowcnt PLS_INTEGER;
   json_col  dbms_tf.tab_varchar2_t;
BEGIN
      dbms_tf.get_row_set(rowset, rowcnt);
      FOR r IN 1..rowcnt LOOP
          json_col(r) := DBMS_TF.ROW_TO_CHAR(rowset, r, DBMS_TF.FORMAT_
JSON);
      END LOOP;
      dbms_tf.put_col(1, json_col);
END;
…
```

*Listing 5: Darstellen vom Rowset als JSON*

tions für jeden unterstützten Datentyp. Nur eine Collection vom richtigen Datentyp ist für jedes Rowset-Element jeweils aktiv. Diese Collections sind assoziative Arrays, auch als „Index-By-Tabellen“ bekannt.

*Listing 4* zeigt die Prozedur „FETCH_ROWS“ für unser Beispiel. Als Erstes holen wir den aktuellen Rowset. Zur Erinnerung: Es werden nur die „FOR-READ“-Spalten gelesen. Der Aufruf von „DBMS_TF.GET_ROW_SET“ gibt außerdem die Anzahl von Spalten und Datensätzen im Rowset zurück. Nun befüllen wir in der Schleife über die Datensätze und Spalten die deklarierte Collection „AGG_COL“ vom Typ „DBMS_TF.VARCHAR2_T“ mit Werten der konkatenierenden Spalten. Wir verwenden dafür die Funktion „DBMS_TF.COL_TO_CHAR“, die eine String-Repräsentation vom jeweiligen Spaltenwert liefert. Wichtig: Für jeden Datensatz muss das Collection-Element einmal zugewiesen werden (die Collection darf nicht „sparse“ sein).

Die Prozedur „DBMS_TF.PUT_COL“ übergibt die gefüllte Collection zurück an die Query. Der erste Parameter ist die Spalten-ID, einfach die fortlaufende Nummer in der Reihenfolge, wie wir die neuen Spalten in der Methode „DESCRIBE“ definiert haben. Ein Referenzieren mit dem Spaltennamen ist nicht vorgesehen.

Nun können wir unsere erste PTF aufrufen und sehen, was der Polymorphismus bedeutet *(siehe Abbildung 4)*. Die Struktur des Ergebnisses wird durch Parameter beeinflusst: Beim ersten Aufruf zeigen wir die Spalte „A“ separat an und führen alle anderen zusammen; beim zweiten Aufruf werden nur die Spalten „C“ und „V“ konkateniert. Auf eine ganz andere Tabelle zuzugreifen geht auch: Mit derselben Funktion konkatenieren wir auch den Datensatz aus der bekannten Tabelle „SCOTT.EMP“.

## JSON

Wenn wir schon die Darstellung relationaler Inhalte in Form eines Dokumentes (CSV) als Beispiel genommen haben, wie wäre es mit JSON statt CSV als Ausgabeformat? Das wird die Aufgabe noch um einiges vereinfachen. „DBMS_TF“ bietet die Funktion „ROW_TO_CHAR“ an, die genau das macht: Alle Spalten im Rowset als JSON darstellen *(siehe Listing 5 und Abbildung 5)*.

## Replikation

Bis jetzt haben wir für einen Input-Datensatz immer genau einen Datensatz im Ergebnis produziert. Es geht aber auch anders: Mithilfe von Row-Replikation ist eine PTF in der Lage, auch mehrere oder gar keine Ergebniszeilen pro Input-Datensatz zu erzeugen. Der Weg dorthin führt über eine überlagerte Prozedur „DBMS_TF.ROW_REPLICATION“, die entweder einen festen Replikationsfaktor für alle Datensätze im Rowset annimmt oder eine Tabelle mit je einem eigenen Faktor für jeden Datensatz.

Als Beispiel können wir die Funktionalität des in 11g eingeführten SQL-Operators „UNPIVOT“ betrachten: Eine definierte Liste der Spalten bleibt im Datensatz in der ursprünglichen Form stehen und alle anderen Spalten sollen in Form von Key-Value-Pärchen im eigenen Datensatz dargestellt werden *(siehe Listing 6 und Abbildung 6)*.

Die Implementierung der „DESCRIBE“-Funktion sieht wieder sehr ähnlich aus. Wichtig ist hier das Flag „ROW_REPLICATION“ als Bestandteil des Records „DESCRIBE_T“, das wir auf „TRUE“ setzen (Default-Wert ist „FALSE“). Dies erlaubt uns, das Replikations-Interface in „FETCH_ROWS“ zu nutzen, andernfalls würde ein „ORA-62574“-Fehler geworfen werden.

In der „FETCH_ROWS“-Prozedur wählen wir die flexiblere Variante – obwohl hier nicht zwingend nötig – und setzen die Replikations-Faktoren pro Datensatz. Dafür verwenden wir die Tabellenvariable „repfac“ vom Typ „DBMS_TF.TAB_NATURALN_T“ und erhöhen ihren Wert um eins für jede zu transponierende Spalte. Vor dem Übergeben der Ergebnisse an die Query zurück rufen wir „DBMS_TF.ROW_REPLICATION“ auf, um die Replikation mit den ausgerechneten Faktoren aktiv zu setzen.

## Table-Semantic PTF

Um die Besonderheiten bei den Table-Semantic PTF zu beleuchten, stellen wir uns eine neue Aufgabe: Für den Output der ersten Aufgabe *(siehe Abbildung 4)* muss eine fortlaufende Summe der Längen der Ergebnis-Spalte „AGG_COL“ ausgegeben werden. Wir nehmen es als einfaches Beispiel, wohl wissend, dass die Funktionalität in SQL mit analytischen Funktionen bereits zur Verfügung steht: „SUM(LENGTH(agg_col)) OVER (...)“.

```
SQL> SELECT * FROM my_ptf(t, COLUMNS(A), 'JSON_COL');

         A JSON_COL
---------- --------------------------------------
         1 {"B":2, "C":3, "V":"ONE"}
         4 {"B":5, "C":6, "V":"TWO"}
         7 {"B":8, "C":9, "V":"THREE"}

SQL> SELECT * FROM my_ptf(t, COLUMNS(A,B), 'JSON_COL');

         A          B JSON_COL
---------- ---------- --------------------------
         1          2 {"C":3, "V":"ONE"}
         4          5 {"C":6, "V":"TWO"}
         7          8 {"C":9, "V":"THREE"}

SQL> SELECT * FROM my_ptf(scott.emp, COLUMNS(empno), 'JSON_COL');

     EMPNO JSON_COL
---------- ------------------------------------------------------------------------
      7369 {"ENAME":"SMITH", "JOB":"CLERK", "MGR":7902, "HIREDATE":"17-DEC-80", ...
      7499 {"ENAME":"ALLEN", "JOB":"SALESMAN", "MGR":7698, "HIREDATE":"20-FEB-81", ...
      7521 {"ENAME":"WARD", "JOB":"SALESMAN", "MGR":7698, "HIREDATE":"22-FEB-81", ...
```

*Abbildung 5: Darstellen vom Rowset als JSON*

```
CREATE OR REPLACE PACKAGE BODY tab2keyval_pkg AS
FUNCTION describe (tab IN OUT dbms_tf.table_t,
                   cols2stay IN dbms_tf.columns_t)
           RETURN dbms_tf.describe_t IS
BEGIN
   FOR I IN 1 .. tab.COLUMN.COUNT LOOP
       IF NOT tab.COLUMN(i).description.name MEMBER OF cols2stay THEN
           tab.column(i).pass_through := false;
           tab.column(i).for_read := true;
       END IF;
   END LOOP;
   RETURN dbms_tf.describe_t(new_columns =>
              dbms_tf.columns_new_t( 1 => dbms_tf.column_metadata_t(
                  name => 'KEY_NAME',
                  TYPE => dbms_tf.type_varchar2),
                  2 => dbms_tf.column_metadata_t(
                  name => 'KEY_VALUE',
                  TYPE => dbms_tf.type_varchar2)),
              row_replication => true);
END describe;
PROCEDURE fetch_rows IS
   rowset dbms_tf.row_set_t;
   repfac dbms_tf.tab_naturaln_t;
   rowcnt PLS_INTEGER;
   colcnt PLS_INTEGER;
   name_col  dbms_tf.tab_varchar2_t;
   val_col  dbms_tf.tab_varchar2_t;
BEGIN
      dbms_tf.get_row_set(rowset, rowcnt, colcnt);
      FOR r IN 1..rowcnt LOOP
          repfac(r) := 0;
          FOR c IN 1..colcnt LOOP
              repfac(r) := repfac(r) + 1;
              name_col(nvl(name_col.last+1,1)) :=
                  regexp_replace(rowset(c).description.name, '^"|"$');
              val_col(nvl(val_col.last+1,1)) :=
                  DBMS_TF.COL_TO_CHAR(rowset(c), r);
          END LOOP;
      END LOOP;
      dbms_tf.row_replication(replication_factor => repfac);
      dbms_tf.put_col(1, name_col);
      dbms_tf.put_col(2, val_col);
END fetch_rows;
END tab2keyval_pkg;
/

CREATE OR REPLACE  FUNCTION tab2keyval (tab TABLE, cols2stay COLUMNS )
   RETURN TABLE PIPELINED ROW POLYMORPHIC USING tab2keyval_pkg;
/
```

*Listing 6: Implementierung der Funktion „TAB2KEYVAL“*

Wie eingangs erwähnt, unterscheiden sich die Table-Semantic PTF dadurch, dass nicht nur der aktuelle Datensatz, sondern auch ein aggregierter Stand aus den zuvor verarbeiteten Datensätzen für das Ergebnis relevant ist. Dabei ist es wichtig, nochmal zu betonen, dass man es nicht selbst in der Hand hat, wie oft „FETCH_ROWS" von der Datenbank aufgerufen wird und wie groß die Rowsets werden. Somit muss man grundsätzlich damit rechnen, den aggregierten Zwischenstand von Ausführung zu Ausführung übergeben zu können. Dies kann man mithilfe eigener, im Package definierter Datenstrukturen erreichen. Falls nötig, erfolgt deren Initialisierung über die Methode „OPEN" und die Ressourcen-Freigabe über „CLOSE". Um die Zustände der eventuell mehrfachen Ausführungen in einer Session voneinander trennen zu können, sollte man diese Strukturen mithilfe der sogenannten „Unique Execution ID" (XID) indizieren. Sie ist über die Funktion „DBMS_TF.GET_XID" abrufbar. Für einfache Anwendungsfälle, wie in unserem Beispiel *(siehe Listing 7)*, genügt auch ein über „DBMS_TF" zur Verfügung stehender Key-Value-Store („XSTORE") sowie entsprechende Get- und Set-Methoden.

Die gezeigte einfache Implementierung kann nur mit einer zu summierenden Spalte klarkommen. Obwohl man eine Liste der Spalten über den „COLUMNS"-Parameter übergeben kann, wird nur die erste Spalte beachtet. *Abbildung 7* zeigt die Verwendung der entwickelten Funktion. Zum einen erkennt man, dass die PTF nicht direkt verschachtelt werden dürfen. Diese Einschränkung kann man aber mit Verwendung der Subquery-Factoring-Klausel („WITH"-Klausel) sehr einfach umgehen. Zum anderen besteht bei Table-Semantic PTF die Möglichkeit, die Input-Daten zu partitionieren und/oder zu sortieren, ähnlich wie es mit der analytischen Funktion „SUM" auch der Fall wäre. So berechnen wir in unserem Bespiel die laufende Summe pro Abteilung.

Aber Achtung: Wird die Abfrage durch die Datenbank parallelisiert, so müssen die Daten der ganzen logischen Partition zwingend vom selben Parallel-Slave-Prozess verarbeitet werden. Der Grund dafür ist offensichtlich: Die Zwischenstände in Package-Variablen werden pro Parallel-Slave unterschiedlich sein. Gibt man in der Abfrage keine Partitionierung an, so ist die PTF ein einziger Serialization Point. Ganz anders sieht es mit Row-Semantic PTF aus: Hier darf die Datenbank beliebig parallelisieren.

```
SQL> SELECT * FROM tab2keyval(t, COLUMNS(A,B));

         A          B KEY_NAME   KEY_VALUE
---------- ---------- ---------- --------------------
         1          2 C          3
         1          2 V          "ONE"
         4          5 C          6
         4          5 V          "TWO"
         7          8 C          9
         7          8 V          "THREE"

SQL> SELECT * FROM tab2keyval(scott.emp, COLUMNS(empno));

     EMPNO KEY_NAME   KEY_VALUE
---------- ---------- --------------------
      7369 Ename      "SMITH"
      7369 Job        "CLERK"
      7369 Mgr        7902
      7369 Hiredate   "17-DEC-80"
      7369 Sal        800
...
```

*Abbildung 6: Ergebnis der Funktion „TAB2KEYVAL"*

```
CREATE OR REPLACE PACKAGE BODY sumlen_package AS
FUNCTION describe (tab IN OUT dbms_tf.table_t
                 , col2sum IN dbms_tf.columns_t )
      RETURN dbms_tf.describe_t IS
BEGIN
   FOR I IN 1..tab.COLUMN.COUNT LOOP
      IF tab.COLUMN(i).description.name = col2sum(1) THEN
         tab.column(i).for_read := true;
      END IF;
   END LOOP;
   RETURN dbms_tf.describe_t(
          new_columns => dbms_tf.columns_new_t(
              1 => dbms_tf.column_metadata_t(
                  name => 'SUM_LEN', TYPE => dbms_tf.type_number)));
END describe;PROCEDURE fetch_rows IS
   rowset dbms_tf.row_set_t;
   colcnt PLS_INTEGER;
   rowcnt PLS_INTEGER;
   len_curr_col dbms_tf.tab_number_t;
   v_len pls_integer;
   v_currlen pls_integer := 0;
BEGIN
      dbms_tf.get_row_set(rowset, rowcnt, colcnt);
      dbms_tf.xstore_get('len', v_len);
      v_currlen := nvl(v_len, 0);
      FOR r IN 1..rowcnt LOOP
         v_currlen := v_currlen + length (rowset(1).tab_varchar2(r));
         len_curr_col(r) := v_currlen ;
      END LOOP;
      dbms_tf.xstore_set('len', v_len+v_currlen);
      dbms_tf.put_col(1, len_curr_col);
END fetch_rows;
END sumlen_package;
/

CREATE OR REPLACE  FUNCTION sumlen_ptf (tab TABLE, col2sum COLUMNS )
      RETURN TABLE PIPELINED TABLE POLYMORPHIC USING sumlen_package;
/
```

*Listing 7: Implementierung „SUMLEN_PTF"*

## Fazit

SQL wird immer mächtiger und flexibler. Mit Polymorphic Table Functions wird die Arbeit derjenigen einfacher, die generische Funktionalitäten in ihre täglichen SQL-Abfragen einbinden. Verlagert sich die Komplexität in den Aufgabenbereich der PTF-Entwickler,

```
SQL> select *
  2  from sumlen_ptf(my_ptf(scott.emp, COLUMNS(deptno)), columns(agg_col));

ORA-62569: nested polymorphic table function is disallowed

SQL> with agg as (SELECT * FROM my_ptf(scott.emp, COLUMNS(deptno)))
  2  select * from sumlen_ptf(agg partition by deptno, columns(agg_col));

    DEPTNO AGG_COL                                                SUM_LEN
---------- ---------------------------------------------------- ----------
        10 7782;"CLARK";"MANAGER";7839;"09-JUN-81";2450;              45
        10 7839;"KING";"PRESIDENT";;"17-NOV-81";5000;                 87
        10 7934;"MILLER";"CLERK";7782;"23-JAN-82";1300;              131
        20 7566;"JONES";"MANAGER";7839;"02-APR-81";2975;               45
        20 7902;"FORD";"ANALYST";7566;"03-DEC-81";3000;                89
        20 7876;"ADAMS";"CLERK";7788;"23-MAY-87";1100;               132
        20 7369;"SMITH";"CLERK";7902;"17-DEC-80";800;                174
        20 7788;"SCOTT";"ANALYST";7566;"19-APR-87";3000;             219
        30 7521;"WARD";"SALESMAN";7698;"22-FEB-81";1250;500           48
        30 7844;"TURNER";"SALESMAN";7698;"08-SEP-81";1500;0           96
        30 7499;"ALLEN";"SALESMAN";7698;"20-FEB-81";1600;300         145
        30 7900;"JAMES";"CLERK";7698;"03-DEC-81";950;                187
        30 7698;"BLAKE";"MANAGER";7839;"01-MAY-81";2850;             232
        30 7654;"MARTIN";"SALESMAN";7698;"28-SEP-81";1250;140        283
```

*Abbildung 7: Das Ergebnis der „SUMLEN_PTF“*

die die PTF erst entwickeln müssen? Ja und nein. Auch wenn die Neuerungen auf den ersten Blick schwer beherrschbar scheinen, kann man relativ schnell das Verständnis aufbauen und sich gewisse Automatismen aneignen. Immerhin übernimmt die Datenbank den Großteil der technischen Aufgaben. Und was die reine Geschäftslogik angeht – vor allem wenn diese komplex ist –, ist es besser, diese einmal generisch für viele Anwendungsfälle zu implementieren, als sie immer wieder neu zu erfinden. Die PTF ergänzt die bestehenden Kapselungsmöglichkeiten via PL/SQL oder Views gut.

Auch wenn ein produktiver Einsatz der neuen Technologie in der allerersten Version in der Regel nicht zu empfehlen ist, ist jetzt die beste Zeit – während der Datenbank-Hersteller an den Verbesserungen und der Stabilität weiterarbeitet –, passende Anwendungsfälle zu identifizieren und auf Herz und Nieren zu prüfen, auch unter dem Gesichtspunkt der Performance.

## Weitere Informationen

- *https://docs.oracle.com/en/database/oracle/oracle-database/18/arpls/DBMS_TF.html*
- *http://blog.sqlora.com/en/tag/ptf*

Andrej Pashchenko
andrej.pashchenko@trivadis.com

# Anwendungskonsolidierung mit der Oracle Cloud Infrastructure

Thomas Rein, dbi services

Mit der Oracle Cloud Infrastructure (OCI) stehen eine Plattform und Werkzeuge zur Verfügung, mit denen Anwendungen und Datenbank-Systeme einfach und schnell konsolidiert werden können. Die unterstützten Oracle-Releases reichen von 11.2.0.4 bis 18.2. Als Editions stehen die Standard und die Enterprise Edition zur Verfügung. Die Enterprise Edition gibt es in drei unterschiedlichen Ausprägungen: normale Enterprise Edition, High Performance Edition und Extreme Performance Edition. Der Artikel zeigt die Einsatzmöglichkeiten der OCI im Rahmen eines Kundenprojekts.

Auf der Open World 2018 hat Larry Ellison in seiner Keynote die neue Generation der Oracle Cloud (OCI) vorgestellt *(siehe „https://blog.dbi-services.com/oracle-open-world-2018-cloudgen2“)*. Natürlich standen die autonomen Datenbanken (Warehouse- und OLTP-Systeme) wie auch die neuen Sicherheits-Features im Vordergrund, aber für den Autor viel wichtiger: Die OCI entwickelt sich zunehmend zu einer echten Cloud-Infrastruktur, mit der sich komplette Anwendungen – Datenbank und Applikation – in die Cloud konsolidieren lassen. Wir besitzen jetzt einen Baukasten, um komplette Datacenter in der Cloud zu realisieren. Ein Kundenprojekt zeigt, wie so eine Migration in die Cloud durchgeführt werden kann.

## Die Ausgangslage

Der Kunde betreibt ein Content Management System (CMS) auf Basis von Censhare für seine Mandanten. Die Anwendung besitzt eine klassische Dreischicht-Architektur. Die Zugriffe auf den Content erfolgen über den Browser („http“/„https“). Die Anwendung wird über Jetty-Applikationsserver bereitgestellt, die Metadaten sind in einer Oracle-Datenbank gespeichert. Der Content selbst liegt im Filesystem.

Zusätzlich gibt es einen oder mehrere Satelliten-Server, die ebenfalls mit einem Jetty-Server realisiert sind. Die Satelliten kommunizieren nicht mit der Datenbank, sondern nur mit dem Applikationsserver. Alle Systeme teilen sich für die Bearbeitung des Contents einen Adobe-In-Design-Server auf Windows 2012 R2. Als Datenbank kommt eine 11.2.0.4 Standard Edition zum Einsatz. Die Datenbanken selbst sind mit bis zu 20 GB eher klein. Die Größe des Contents variiert je nach Mandanten und reicht bis zu 8 TB.

Insgesamt werden Systeme für sechs Mandanten betrieben, jeweils Produktion und Test, insgesamt haben wir es mit 24 Servern zu tun. Die Server sind als virtuelle Maschinen realisiert, die Datenbank-Server als Bare-Metal-Systeme. Die komplette Infrastruktur wird durch einen Provider gehostet *(siehe Abbildung 1)*. Daraus ergeben sich die folgenden Problemstellungen:

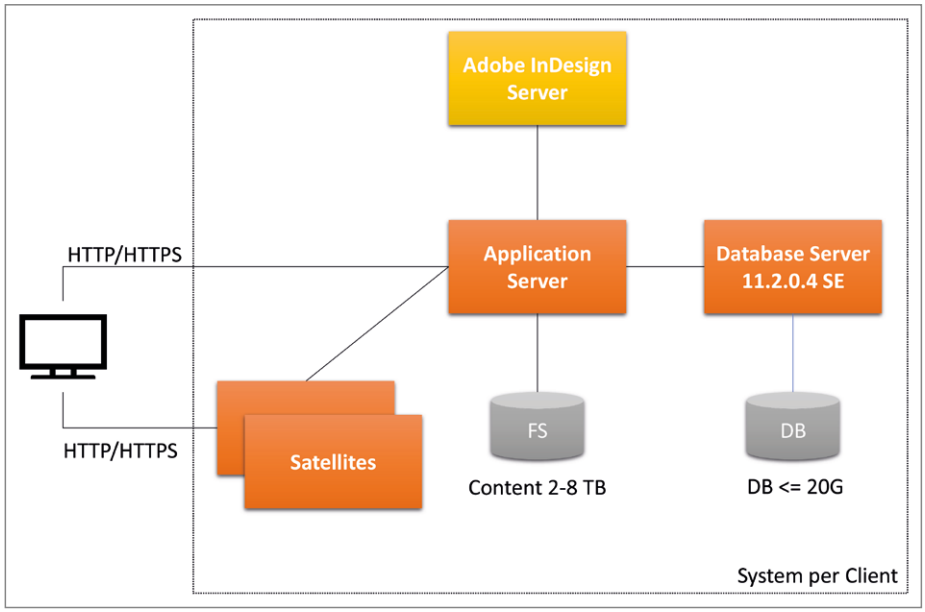



*Abbildung 1: Die Ausgangslage*

| Bereich | Anforderung | Lösung |
|---|---|---|
| Datenbank | Reduktion der Instanzen; bisher besitzt jeder Mandant seine eigene Datenbank | Pluggable Databases; jeder Mandant erhält je eine PDB für Produktion und Test |
| | Verfügbarkeit der Datenbank-Systeme; bisher Single Instance Standard Edition | Einsatz von Oracle Enterprise Edition und Data Guard |
| | Recovery einzelner Mandanten | PDB mit Local Undo; damit ist ein Recovery einzelner PDBs möglich |
| | Zurückrollen fehlerhafter Installationen | Restore Points; mit Local Undo Flashback einzelner PDBs möglich |
| | Migration einzelner Mandanten auf neue RDBMS-Releases | Unplug/Plug von PDBs |
| Applikationsserver und Satelliten | Schnelles und einfaches Provisionieren von VMs | Customized Boot Image erstellen, ab dem die VMs geklont werden |
| | Scale-up des Contents Storage | Pro VM eigener Block Storage für den Content; kann in der OCI problemlos vergrößert werden |
| | Update der Applikation | Eigener Block Storage, auf dem die Applikation installiert wird, Update der Applikation mit umount/mount |
| | Auslastung/Überlastung von VMs, Hochverfügbarkeit der Applikationsserver | Einsatz von Loadbalancern |
| Sicherheit | Zugang zu den Systemen nur über HTTP/HTTPS möglich | Loadbalancer bekommen Public-IP-Adresse; Network Segregation |
| | Anbindung der VMs | SSH-Keys, Bastion Host, direkter Zugriff via VPN aus dem Kunden-Netzwerk |

*Tabelle 1*

- Die Bereitstellung neuer Server ist aufwendig, jede VM muss neu installiert werden (zusätzliche Pakete, Konfiguration der Applikationsserver)
- Das Ausrollen neuer Bare-Metal-Systeme benötigt etwa einen Tag (Hardware-Bereitstellung, Linux-Installation, Oracle-Installation, Datenbank anlegen)
- Keine oder schwierige Skalierung der Systeme
- Extended Support für Oracle RDBMS 11.2.0.4 ab dem Jahr 2019
- Einschränkungen bei Upgrade auf Oracle RDBMS 12c SE
- Deutlich höhere Kosten durch Wechsel auf Oracle EE
- Fehlende DR-Lösungen (Datenbank und Applikationsserver)
- Änderungen am Content Storage sind kompliziert
- Keine Desaster-Lösung vorhanden, Crashs bedeuten Restore der Systeme

## Migrations- und Konsolidierungsziele

Die wichtigsten Ziele des Kunden waren vor allem, mehr Flexibilität im Deployment und in der Konfiguration der Systeme zu erreichen. Darüber hinaus waren Ausfallsicherheit und Upgrade der Datenbank auf 12c, 18c oder 19c wichtige Kriterien. Die Auswahl fiel von daher schnell auf die OCI, da sie folgende Kriterien erfüllt:

- Schnelle und flexible Bereitstellung von VMs möglich
- Unterstützung von Linux und Windows Guests
- Einfache Skalierung von VMs und Storage
- Oracle-Lizenzierung eingeschlossen
- Moderate Kosten für die Enterprise Edition
- Hochverfügbarkeit der Infrastruktur

Die gesamte Migration sollte innerhalb von zwei Monaten abgeschlossen sein, eine Downtime zum Umschaltzeitpunkt war möglich. Aus den Kundenanforderungen und -zielen wurde die Architektur der künftigen Systeme wie in *Tabelle 1* entwickelt.

## Die Netzwerk-Architektur

Ein wesentlicher Schritt besteht in der Definition der Netzwerk-Strukturen. Hier ist einerseits zu berücksichtigen, dass für die Availability Domains jeweils ein Subnetz gebraucht wird, andererseits möchte man die VMs nicht mehr als notwendig im Internet exponieren. Es wurden daher für jede der aktiven Availability Domains drei Subnetze definiert *(siehe Abbildung 2)*:

- *Frontend-Netzwerk*
  Hier werden später die Loadbalancer und Bastion Hosts lokalisiert
- *Middleware-Netzwerk*
  Alle Applikationsserver, Satelliten und der InDesign-Server befinden sich in diesem Netzwerk
- *Backend*
  für die Datenbank-VMs und gegebenenfalls andere Systeme, die nicht über das Internet erreichbar sein sollen

## Security Lists und Routing Tables

Im nächsten Schritt werden die Zugänge zum VCN definiert. Die Anbindung des Internets erfolgt über ein Internet-Gateway. Die Integration des Kunden-Netzwerks wird über VPN und ein Dynamic Routing Gateway (DRG) realisiert. Die Konfiguration und Anbindung des DRG ist hier der komplizierteste Vorgang, da die Konfiguration jeweils von der Infrastruktur des Kunden abhängt *(siehe Abbildung 3)*.

Nachdem die Zugänge realisiert sind, werden Routing- und Security-Listen definiert. Eine Security-Liste ist nichts anderes als eine Reihe von Firewall-Regeln. Hierbei gilt: Jeder Netzwerk-Verkehr, der nicht erlaubt ist, wird weggefiltert. Es gibt eingehende („Ingress“) und ausgehende („Egress“) Regeln. Eine typische Security-Liste für das Backend-Netz (Datenbank) sieht wie in *Tabelle 2* aus.

Jeder darf sich also via SSH an die Systeme verbinden. Verbindungen an die Datenbank sind aus allen Netzen 10.0.0.0/16 erlaubt. Innerhalb des Backend-Netz-

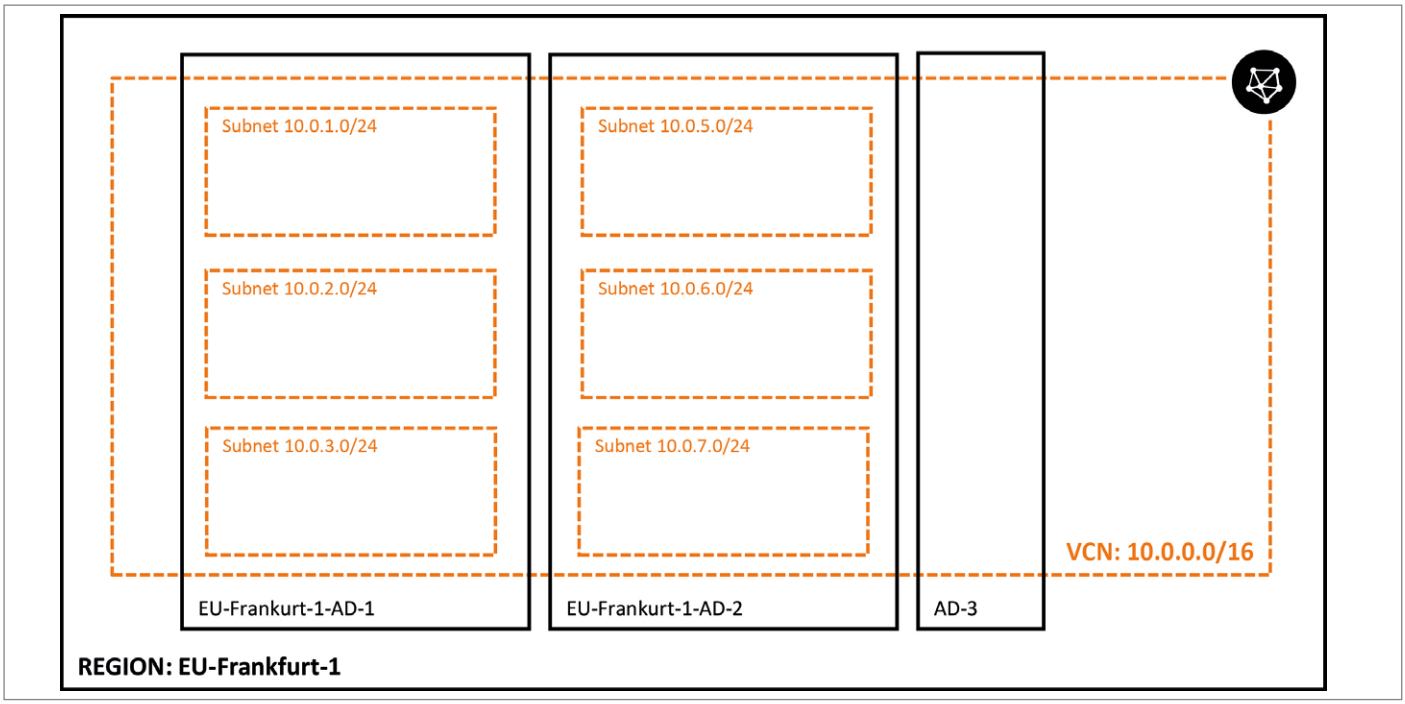



*Abbildung 2: Die Subnetze*

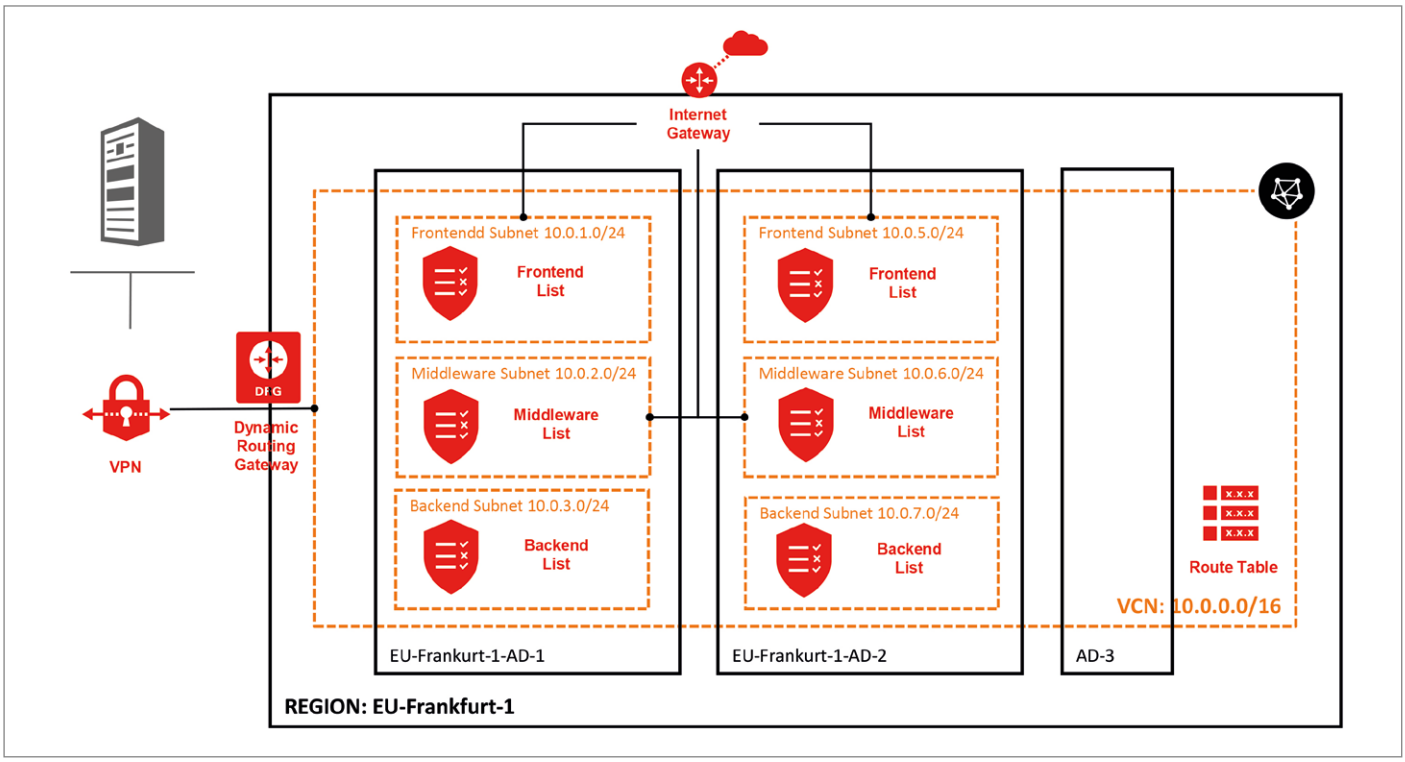



*Abbildung 3: Security und Routing Lists*

| Ingress Rules | | | | |
|---|---|---|---|---|
| Source: 0.0.0.0/0 | IP Protocol: TCP | Source Port Range: All | Destination Port Range: 22 | Allows: TCP traffic for ports: 22 SSH Remote Login Protocol |
| Source: 10.0.0.0/16 | IP Protocol: TCP | Source Port Range: All | Destination Port Range: 1521 | Allows: TCP traffic for ports: 1521 |
| Source: 10.0.3.0/24 | IP Protocol: TCP | Source Port Range: All | Destination Port Range: All | Allows: TCP traffic for ports: all |
| **Egress Rules** | | | | |
| Destination: 0.0.0.0/0 | IP Protocol: TCP | Source Port Range: All | Destination Port Range: All | Allows: TCP traffic for ports: all |

*Tabelle 2*

werks gibt es keine Beschränkungen zwischen den Systemen. Alle VMs dürfen eine Verbindung nach außen aufbauen. Die Routing-Tabellen legen anschließend fest, wie der Datenverkehr von und zum DGR beziehungsweise zum Internet-Gateway erfolgt. Hier kann entweder mit einer globalen Tabelle gearbeitet werden oder jedes Subnetz bekommt seine eigene Routing-Tabelle. Die Definition der Security-Listen sollte mit Sorgfalt erfolgen, da sie meistens die Ursache dafür sind, dass ein System nicht erreicht werden kann, dass Verbindungen zwischen den Systemen nicht möglich sind oder ein System offen im Internet exponiert ist.

## Datenbank-Setup und Data Guard

Nach Abschluss der Netzwerk-Konfiguration kann das Datenbank-System aufgesetzt werden. In der OCI stehen folgende Editionen zur Auswahl:

- Standard Edition; entweder als Standard Edition One (11g) oder Standard Edition Two (12c, 18c)
- Enterprise Edition (keine zusätzlichen Optionen aktiviert)
- Enterprise Edition High Performance; alle Optionen außer In-Memory, Active Data Guard und RAC dürfen benutzt werden
- Enterprise Edition Extreme Performance mit allen Optionen

Da ein Data-Guard-System aufgesetzt werden sollte, fiel die Entscheidung auf die High Performance Edition. Als Version wurde 12.2.0.1 gewählt, da einerseits das Release noch bis Ende 2020 auf der OCI unterstützt wird und der Kunde dann direkt auf 19c migrieren möchte, andererseits die 12.2 mit Local Undo die notwendigen Features mitbringt, um ein Recovery/Flashback von einzelnen PDBs durchzuführen.

Die Version 18c (19c) ist interessant, wenn mehr Mandanten in unterschiedlichen Applikations-Versionen auf das System gebracht werden müssen (Application Container), allerdings muss der

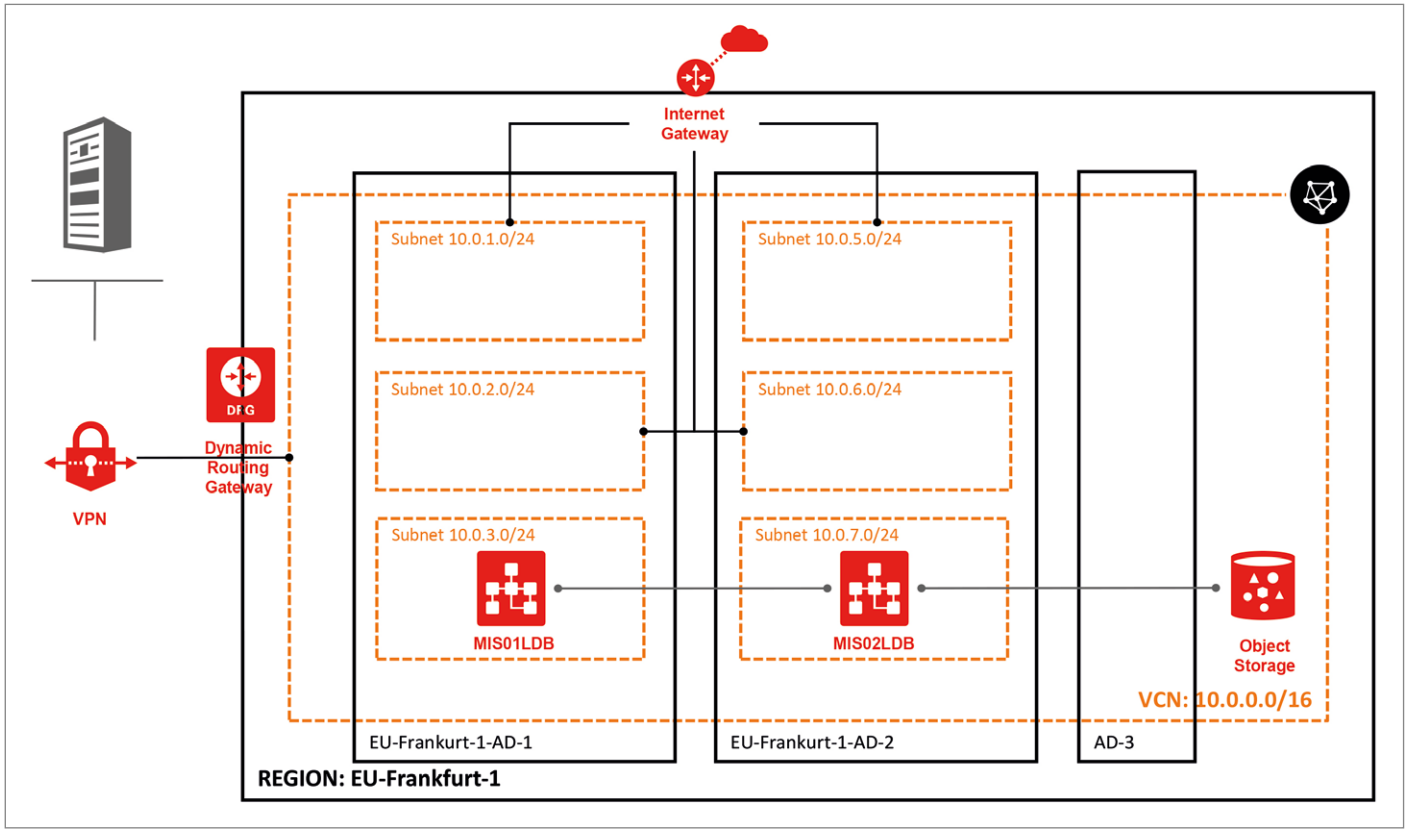



*Abbildung 4: Das Datenbank-Setup*

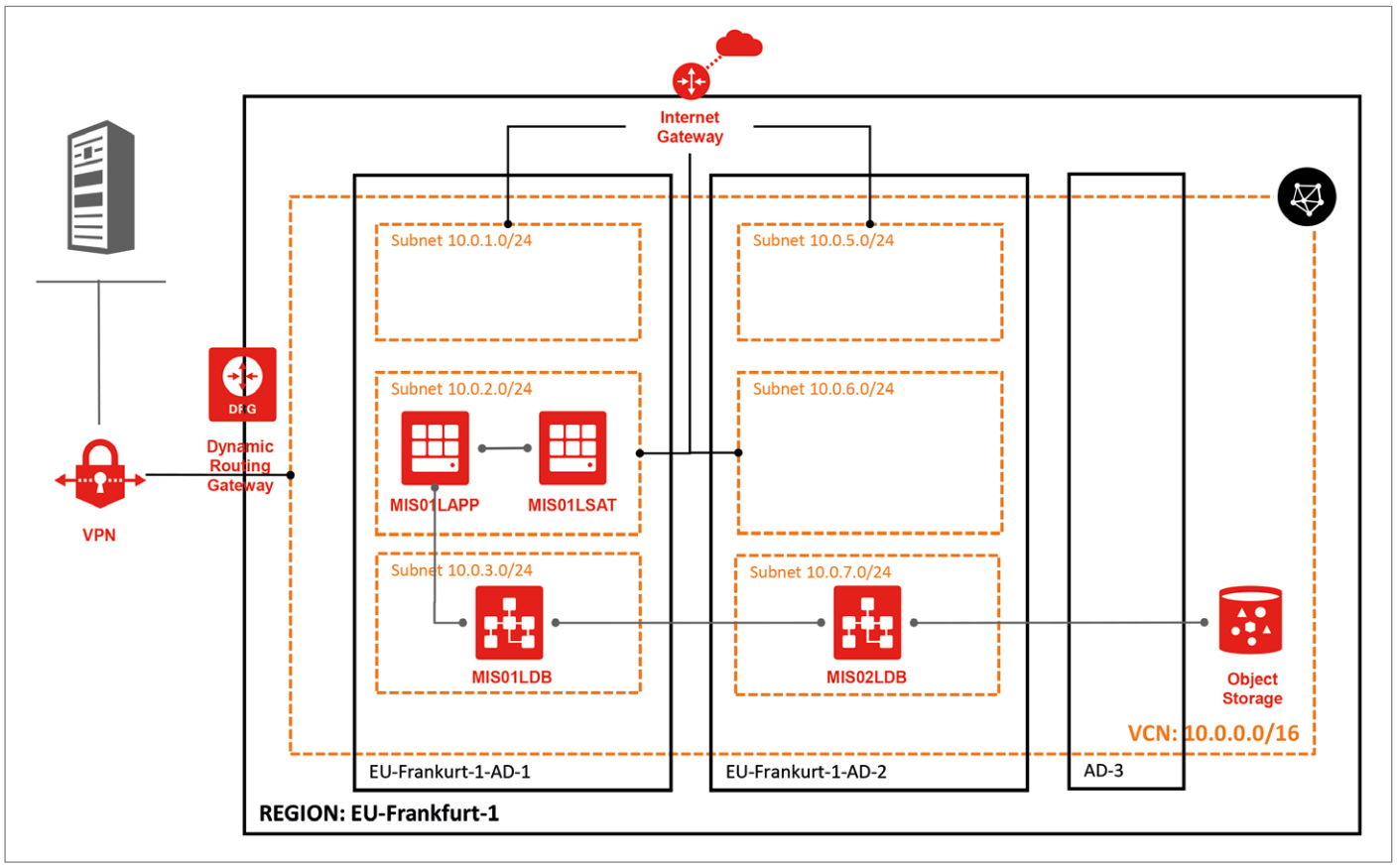



*Abbildung 5: Der Applikationsserver*

Software-Hersteller dafür noch eine Freigabe erteilen. Da die Datenbanken der einzelnen Mandanten eher klein ausfallen, wurde als Datenbank-Shape eine Konfiguration mit zwei OCPUs und 1-TB-NVMe-Storage ausgewählt. Sollten mehr OCPUs notwendig sein, wird einfach ein weiteres Datenbank-System bereitgestellt und die PDB auf dieses umgezogen. Zusätzlicher Storage kann in den bestehenden Systemen bereitgestellt werden.

Als Backup-Variante hat man das automatische Cloud-Backup ausgewählt (wöchentlicher Level-0-Backup, täglicher Level-1-Backup). Will man diese Backup-Variante nutzen, ist ein sogenannter „Object Storage" erforderlich, der eigens angelegt und in das System über einen Routing-List-Eintrag integriert werden muss. Vorsicht ist geboten beim Block-Change-Tracking auf der Standby-Datenbank, da dies unter Umständen zu Active Data Guard führen kann. Die Data-Guard-Konfiguration muss bei diesem Shape manuell über DGMGRL durchgeführt werden. Eine Konfiguration über die Web-Konsole ist nur bei Bare-Metal-Systemen möglich *(siehe Abbildung 4)*.

## Setup der Applikationsserver

Für die Applikationsserver wurde zunächst eine Template-VM unter Oracle Linux 7.5 aufgesetzt. Als Shape wurde eine VM.Standard2.2 mit zwei OCPU, 30 GB Memory und 50 GB Boot Volume gewählt. Für die Anwendung und den Content wurden zwei zusätzliche Disks über iSCSI integriert. Nachdem die Anwendung und zusätzliche Pakete installiert waren, wurde aus dem Boot Volume ein Custom Image erstellt, das als Grundlage für die Erstellung neuer VMs dient.

Nach dem Klonen muss dann nur noch eine Konfigurationsdatei für die Anwendung und die TNS-Einträge für die PDB angepasst werden. Die Satelliten setzen auf dem gleichen Shape auf, haben aber keinen Content Storage und auch keine direkte Verbindung zur Datenbank. Auch hier empfiehlt es sich, die Bereitstellung von VMs über Custom Images zu vereinfachen *(siehe Abbildung 5)*.

Die Migration auf die neuen Systeme erfolgt in zwei Schritten. Vorab wird der Content regelmäßig inkrementell mit „rsync" auf die neuen Systeme übertragen. Zum Umschaltzeitpunkt erfolgen eine letzte Synchronisation und ein Export/Import der Datenbank. Sollte ein Upscale des Content Storage notwendig sein, lässt sich das entsprechende Volume direkt vergrößern. Wird mehr CPU benötigt, stellt man die neue VM zur Verfügung und verbindet diese mit den Volumes der alten VM.

## Hochverfügbarkeit durch Loadbalancer

Um die Verfügbarkeit der Systeme zu gewährleisten, ist abschließend noch ein Loadbalancer mit einer Public-IP-Adresse erforderlich. Die zugehörigen Backend-VMs werden dann in unterschiedlichen Availability-Domains platziert. Der Zugang von außen auf einzelnen VMs wird über einen Bastion-Host kontrolliert, der ebenfalls eine Public-IP erhält. Der Zugang auf die Backendsysteme erfolgt dann über „ssh -o ProxyCommand='ssh -i .ssh/id_rsa opc@<PublicBastionIP> -W %h:%p %r' opc@<BackendIP>" *(siehe Abbildung 6)*. Bei Bedarf kann danach in der Availability-Domain AD-3 noch ein Observer installiert und Fast Start Failover im Data Guard konfiguriert werden.

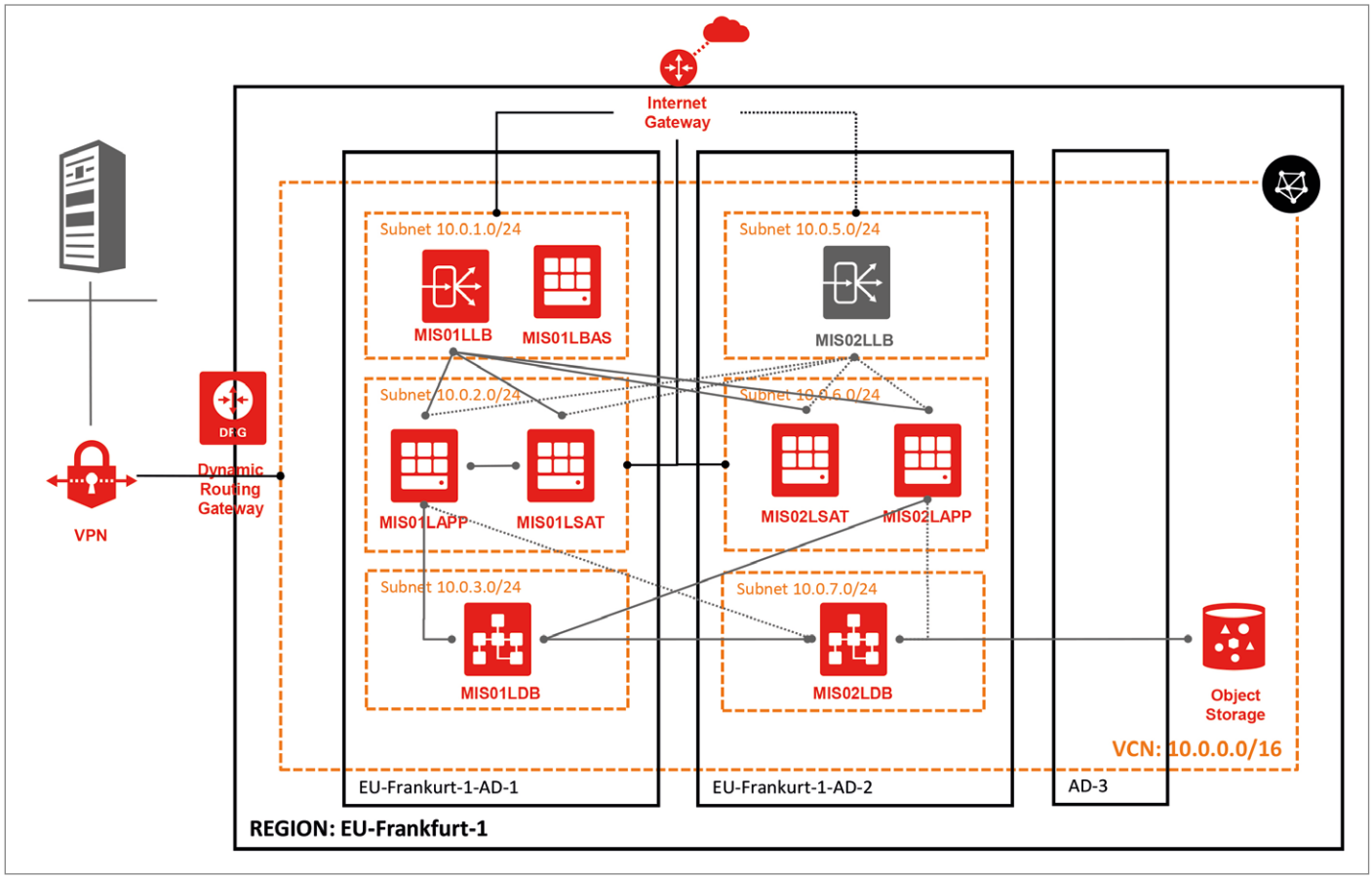



*Abbildung 6: Loadbalancer und Hochverfügbarkeit*

## VM Cloning und Custom Images

Wie bereits erwähnt, lässt sich das Provisioning der VM vereinfachen, indem Custom Images aus den Boot Volumes erstellt werden, die dann beim Erzeugen von neuen VMs als Grundlage verwendet werden. Die Installation zusätzlicher Pakete entfällt, Netzwerk-Konfiguration etc. wird bei der Erstellung von der OCI automatisch angepasst. Ebenso ist es möglich, ein Block-Volume zu klonen und damit für eine andere VM bereitzustellen. Das Provisioning eines neuen Applikationsservers ist damit in kürzester Zeit erledigt – anzupassen ist nur noch die Konfiguration des Applikationsservers selbst (TNS, Jetty Config), und auch dies kann recht einfach über Shell-Skripte automatisiert werden *(siehe Abbildung 7)*.

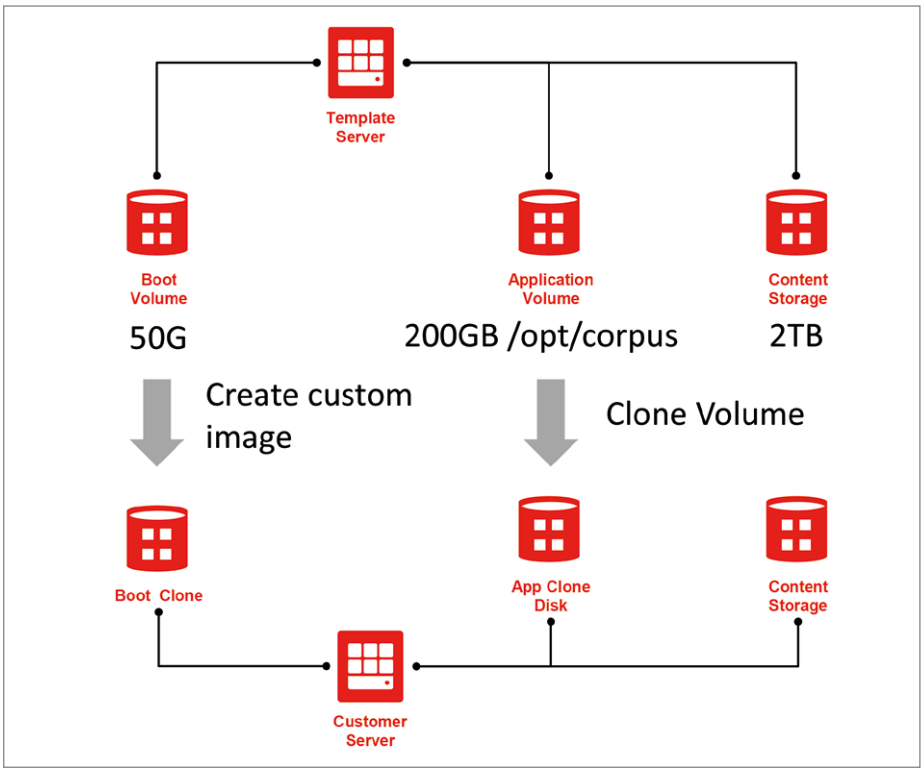



*Abbildung 7: VM Cloning*

## Fazit

Hat man die Nomenklatur und Verwendung der einzelnen Komponenten erst einmal verstanden (VCN, Subnetze, Security Lists etc.), lassen sich Infrastrukturen sehr schnell erstellen. Im konkreten Fall dauerte die Bereitstellung der Umgebung zwei Tage, für den Windows-InDesign-Ser-

ver wurde ein Tag benötigt. Bereits nach zwei Wochen war der erste Mandant in der neuen Umgebung produktiv: Mittlerweile ist die Migration abgeschlossen.

Trotzdem gibt es natürlich immer Verbesserungsbedarf, speziell beim Handling von Pluggable Databases. Diese werden im Moment noch nicht über die Web-Konsole oder das CLI unterstützt. Es ist zwar möglich, beim Anlegen einer Container-Datenbank eine PDB mit anzulegen, jede weitere muss aber mit „CREATE PLUGGABLE DATABASE“ manuell erzeugt werden. Eine Stolperfalle ist hierbei die Transparent Data Encryption (TDE), die automatisch in der OCI aktiviert ist. Wird eine neue PDB angelegt, so muss der Master Key über „dbcli update-tdekey“ für die PDP aktiviert werden, ansonsten funktioniert weder der Backup, noch können zusätzliche Tablespaces in der PDB angelegt werden.

Auch beim Restore und Recovery der Datenbank gibt es Nachholbedarf. Zwar ist es möglich, über die Konsole oder das CLI ein Recovery der Container-Datenbank durchzuführen; will man jedoch eine PDB recovern, muss dies manuell durchgeführt werden. Hier wäre eine Unterstützung über das CLI oder die Konsole wünschenswert. Trotzdem war der Autor angenehm überrascht, wie einfach und schnell sich Kunden-Umgebungen in der OCI bereitstellen lassen.

Thomas Rein
thomas.rein@dbi-services.com

# Oracle-Support-Umfrage 2018

Christian Trieb, DOAG Competence Center Support

Die Nutzung und Akzeptanz der Oracle Proactive Support Tools steigt deutlich, die allgemeine Zufriedenheit beim Support sinkt und die Cloud-Nutzung ist weiterhin gering.

Im zweijährigen Abstand befragt die DOAG ihre Mitglieder zur Qualität und zur Zufriedenheit mit dem Oracle-Support. Diesmal standen sowohl der klassische Support als auch erstmalig der Support für Cloud-Lösungen im Fokus. Insgesamt beteiligten sich dieses Jahr mit 277 Teilnehmern 63 DOAG-Mitglieder mehr an der Support-Umfrage als im Jahre 2016.

Christian Trieb, Leiter des DOAG Competence Center Support, und DOAG-Vorstandsvorsitzender Stefan Kinnen stellten auf der DOAG 2018 Konferenz + Ausstellung die Ergebnisse der Oracle-Support-Umfrage 2018 vor und diskutierten mit den anwesenden Oracle-Vertretern Clarissa Rohrmann (Customer Success Manager) und Pravin Anil Sheth (Director Database Supporting) sowie dem Publikum.

Die Oracle-Support-Kunden aller Unternehmensgrößen und Berufsbilder vergaben zu 15 Prozent positive Bewertungen hinsichtlich der allgemeinen Zufriedenheit. Die allgemeine Zufriedenheit ist seit dem Jahr 2016 somit nochmals um 4 Prozentpunkte gesunken. Rund ein

Fünftel bewertete den Support mit „Befriedigend“. Insgesamt 66 Prozent bezeichneten sich als weniger oder gar nicht zufrieden. Dies sind 10 Prozentpunkte mehr als 2016 *(siehe Abbildung 1)*.

Mit dem Online-Portal My Oracle Support sind die Nutzer zu 42 Prozent ziemlich zufrieden, 10 Prozent sogar sehr zufrieden. Damit stieg die Zufriedenheit um weitere 4 Prozentpunkte auf 52 Prozent im Vergleich zum Jahr 2016. Der Anteil der sehr zufriedenen Nutzer steigerte sich von 7 auf 10 Prozent.

Im Detail ist mehr als die Hälfte der Teilnehmer insbesondere mit der Dauer bis zur Lösung ihres Problems (77 Prozent), den Abläufen/Prozessen (65 Prozent), der Qualität der angebotenen Problemlösung (58 Prozent) und dem Verständnis für das spezifische Problem (68 Prozent) unzufrieden *(siehe Abbildung 2)*.

Da nur 9 Prozent der erstmals zum Thema „Cloud“ Befragten Oracle-Cloud-Produkte nutzen, gab es nur eine geringe Ausgangsbasis von 25 Stimmen für die qualitative Beurteilung und die Häufigkeit der Nutzung des Oracle Cloud Support. Von diesen Cloud-Nutzern setzen 20 den Cloud Support zu insgesamt 45 Prozent regelmäßig und intensiv.

Weiterhin noch Steigerungspotenzial ist hinsichtlich der Bekanntheit und Nutzung einzelner Support-Tools festzustellen. So wird beispielsweise der Oracle Proactive Support nur von 7 Prozent der Befragten genutzt. 41 Prozent kennen den Service nicht. Die Proactive-Support-Tools werden überwiegend positiv beurteilt. Die Zufriedenheit nahm im Vergleich deutlich zu, denn 2016 gab es nur insgesamt 8 Prozent zufriedene Nutzer.

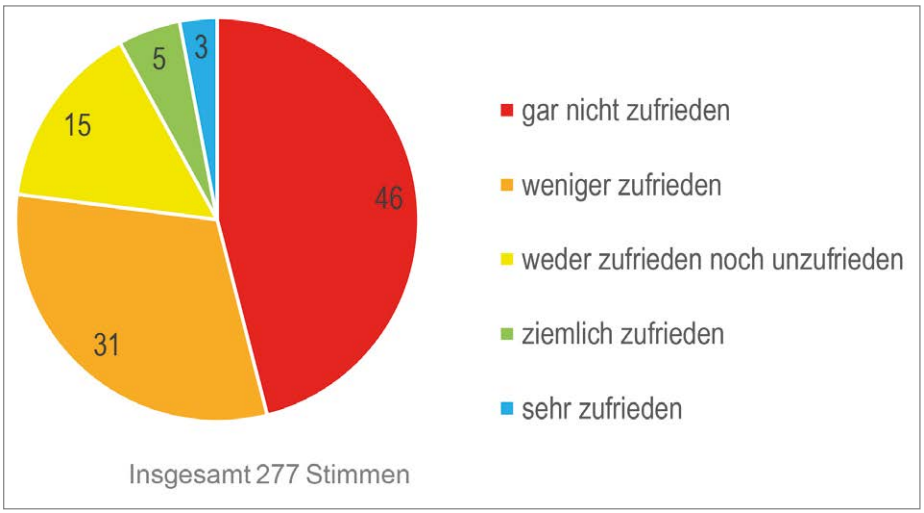



*Abbildung 1: Die Zufriedenheit sinkt*

Hinsichtlich der Beurteilung der qualitativen Entwicklung des Oracle Support im letzten Jahr zeichnete sich eine Trendwende ab. Konstatieren im Jahr 2016 noch 64 Prozent eine Verschlechterung, so sehen dies 2018 nur noch 49 Prozent. Für 36 Prozent ist die Qualität gleich geblieben.

Obgleich die von anwesenden Besuchern geschilderten negativen Support-Erfahrungen einen wichtigen Einblick in die aktuellen Ungereimtheiten im Bereich des Oracle Support boten, lag der Fokus doch auf der Feststellung, dass es sich nicht um die Summe bedauernswerter Einzelfälle handelt. Aufgrund der langjährig zu beobachtenden Tendenzen liege im Gegenteil ein grundlegendes strukturelles Problem vor, das von Oracle gelöst werden müsse, wozu die DOAG das Unternehmen dringend aufrufe.

Vorstand und Geschäftsführer der DOAG Fried Saacke meinte, dass beim Oracle Support zu sehr am Personal gespart wird, und fragte, ob Oracle dazu bereit sei, die Support-Mitarbeiter besser zu qualifizieren und dafür mehr Geld auszugeben. Stefan Kinnen und Fried Saacke mahnten Oracle eindringlich, die Kritik am Support ernst zu nehmen. Clarissa Rohrmann von Oracle kündigte an, die präsentierten Ergebnisse der DOAG-Support-Umfrage dem Board of Directors vorzulegen und nach genauerer Nachfrage eine Reaktion bis Ende des Jahres 2018 zu präsentieren.

Die gesamte Support-Umfrage steht zum Download unter *„https://backoffice.doag.org/formes/pubfiles/10833356/docs/Presse/2018/Support-Umfrage_2018.pdf“* bereit.

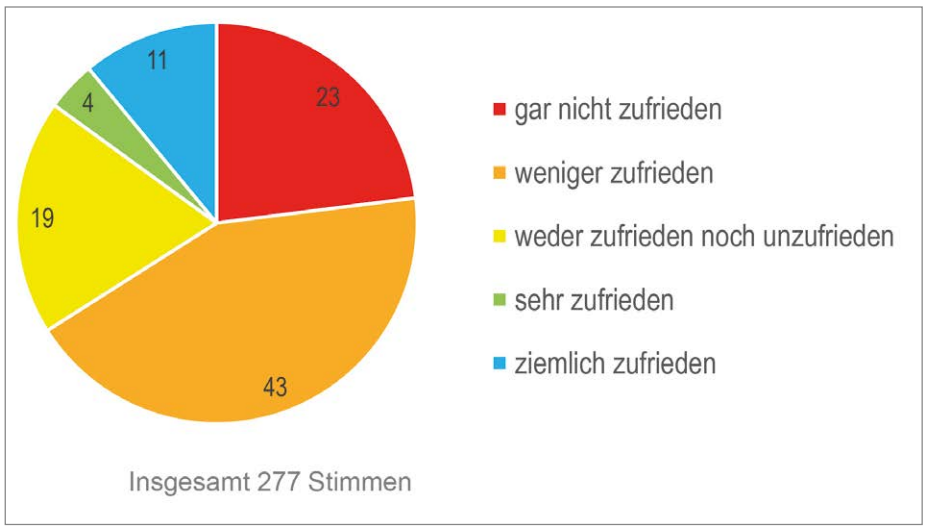



*Abbildung 2: Wenig Zufriedenheit mit der Dauer bis zur Lösung*

Christian Trieb
christian.trieb@doag.org

# 40 Jahre Oracle Forms

Frank Hoffmann, Cologne Data GmbH


Oracle feiert in diesem Jahr ein beeindruckendes Doppel-Jubiläum: Vierzig Jahre Datenbanken und Forms/Reports. Es war Bill Friend, der nach Bruce Scott fünfte Oracle-Mitarbeiter (#1004), den Larry Ellison im Jahr 1979 mit der Entwicklung des berühmten Software-Tools beauftragte. Im selben Jahr brachte Oracle dann noch vor Weihnachten neben der ersten kommerziellen Datenbank (Version 2.3) auch die erste Forms- und Reports-Version unter der Bezeichnung „IAF/RPT" heraus.

Mercedes wirbt für seine G-Klasse aus dem Jahre 1979 mit dem Claim „Stronger than time" und das ist auch für Oracle Forms passend. Die Software-Evolution läuft bereits seit vierzig Jahren; der Modulcode von ASCII-Text bis XML und der Support für Java-Versionen von 1 bis 11, der Datenbank-Support für die Versionen 2.3 bis 19, der Windows-Support von 3.1 bis 2016 sowie der Linux-, Solaris- und AIX-Support ebenfalls. Immer ist das Tool mitgegangen, ohne dass der Anwender seinen Code umschreiben musste – ein starkes Plus. Darüber hinaus schaffen 4GL-Standardfunktionen wie Row Locking, Binding sowie die PL/SQL-Nutzung der aktuellen Datenbank enorme Vorteile gegenüber allen anderen PL/SQL-Werkzeugen *(siehe Abbildung 1)*.

Es ist an der Zeit, einmal zu den Forms-Ursprüngen zu gehen, um die Gründe zur Entstehung dieses Tools verstehen zu können. Über Hinweise von Mike Ferrante und soziale Medien ist der Autor auf Bill Friend gestoßen, den Entwickler der Ursprungsversion, und durfte ihn intensiv dazu befragen. Mit seinen sechzig Jahren ist er immer noch sehr vital und kann sich an viele Details aus den Anfängen von Forms erinnern. Die folgende Geschichte und das Interview stammen direkt aus den Informationen von Bill Friend und einem Gespräch Ende Dezember 2018. Er hat damit nach vierzig Jahren sein erstes Interview zu Oracle Forms gegeben.

## Forms erblickt das Licht der Welt

Das Jahr 1979 hat viele wirklich interessante und spannende Neuerscheinungen hervorgebracht. Quincy Jones verhalf mit „Off the Wall" Michael Jackson zu einem Durchbruch und Mercedes entwickelte das erste Modell der G-Klasse. Namco brachte in Japan mit „Galaxy" die erste farbige Arcade-Fassung heraus sowie danach die berühmten Nachfolger „Galaga" und „Pac Man" – noch lange bevor Heimcomputer ihren Siegeslauf antraten. Aber auch wichtige technische Neuerungen fielen in dieses Jahr wie zum Beispiel „TCP/IP", die Programmiersprache „C", der Verkauf der Prozessoren Intel 8088 und Motorola 68000, alles Voraussetzungen für zukünftige Entwicklungen von Microsoft, Apple und Oracle sowie der Verbreitung des Internets.

In dieser Zeit suchte ein junger Entwickler einen neuen Job. Sein Name ist Bill Friend und er machte das so, wie man in dieser Zeit üblicherweise einen neuen Job suchte: Man nahm die Gelben Seiten zur Hand und suchte unter „Data Processing" nach geeigneten Firmen, um dann unter „S" auf „Software Development Labs" zu stoßen, einer Firma mit drei Gründern und einem talentierten angestellten Entwickler namens „Bruce Scott". Beim ersten Anruf ging Bob Miner an das Telefon. Bei der Frage nach einem Job kam folgende Antwort: „We're creating the first commercial relational database management system, we're doing it on DEC PDP minicomputers in assembly language, but we want to rewrite it in C for portability. We need someone who can write some tools for the thing." Sie suchten also nach einem Entwickler, der für die neue relationale Datenbank Werkzeuge in der neuen Programmiersprache C entwickeln konnte.

Bill fuhr zum Vorstellungsgespräch nach Sand Hill Circle in Menlo Park und wurde zum Mittagessen mit Bob Minor und Larry Ellison eingeladen. Nachdem Bob Minor mit UFI (User Friendly Interface, einem SQL-Plus-Vorläufer) ein paar

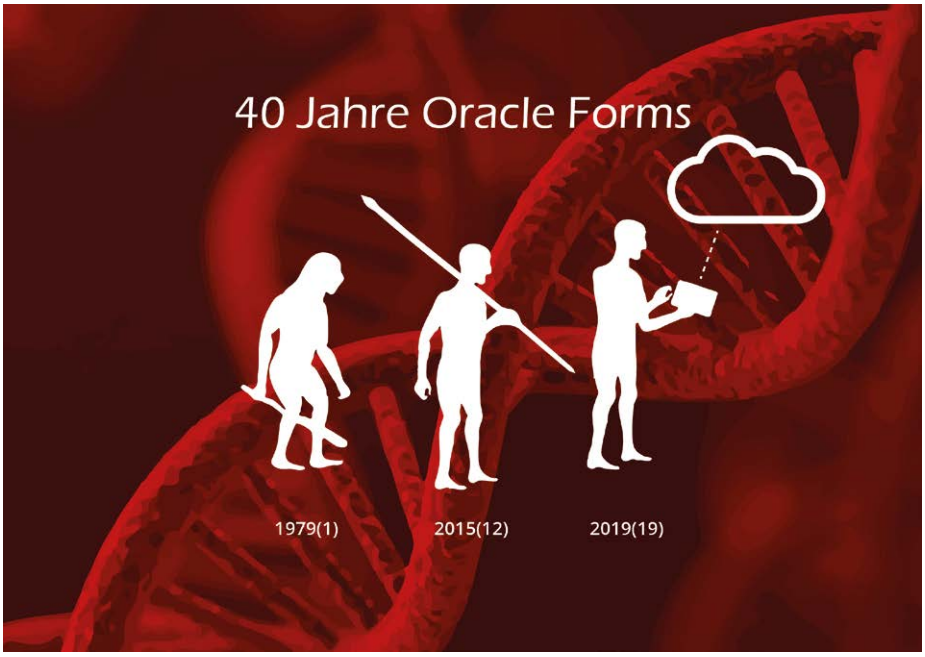



*Abbildung 1: 40 Jahre Oracle Forms*

Datenbank-Abfragen präsentiert und Bill über seine Programmier-Erfahrungen gesprochen hatte, wurde man sich schnell einig. Dass Bill weder einen Universitätsabschluss vorweisen noch jemals zuvor mit C entwickelt hatte, störte dabei nicht – das könne man ja noch lernen. Für Larry und Bob schien Bill genau der Richtige zu sein, also smart genug, um die Aufgabe erledigen zu können. Der Instinkt sollte sie nicht trügen. So wurde Bill Friend die Nummer fünf bei Oracle, die zweite Person, die man einstellte, und eine starke Unterstützung für den Aufstieg von fünf auf fünfhundert Mitarbeiter in den nächsten Jahren. Er kaufte sich sofort das Buch „The C Programming Language" von Brian Kernighan and Dennis Ritchie (K&R book), das 1978 erschienen war, lernte autodidaktisch und mit großer Freude C und hatte genau den Job gefunden, den er immer gesucht hatte.

## Ein einfacher Auftrag von Larry Ellison brachte Oracle Forms

Neben der Einarbeitung in C beschäftigte sich Bill Friend mit dem neuen relationalen Datenbank-System und arbeitete eng mit Bruce Scott zusammen. Nach ein paar Monaten beherrschte er C und hatte auch einen guten Überblick über die Datenbank-Technologie sowie deren APIs. Dann musste er selbst herausfinden, womit er sich genau beschäftigen sollte. In dieser Zeit wurde niemand überwacht, bevormundet oder kontrolliert – jeder machte das, was er am besten konnte, mit großer Leidenschaft.

Bill fehlte ein konkretes Projekt und so ging er mit dieser Bitte zu Larry Ellison. Der sagte: „Wir brauchen einen komfortableren Weg, um Daten in die Datenbank zu schreiben, als über einen „INSERT"-Befehl. Vielleicht zeigst du für jede Datenbank-Spalte („column") ein „Prompt" an" – damit ging es los. Aus diesen sehr wagen Vorgaben erstellte Bill Friend ein detailliertes Konzept, das deutlich mehr bot, als nur den Wunsch von Larry zu erfüllen. Das Ziel von Bill bestand darin, das Leben für den Masken-Entwickler und -Anwender zu erleichtern und eine hohe Produktivität bei der Masken-Entwicklung zu schaffen. Passend zur 4GL-Theorie schreibt Wikipedia dazu: „Als fourth generation language oder kurz „4GL" bezeichnet man Programmiersprachen beziehungsweise Umgebungen der vierten Generation. Diese sind darauf ausgerichtet, rasch – mit möglichst wenigen Codezeilen – für einen bestimmten Anwendungsbereich Funktionen oder komplette Anwendungen schreiben zu können." Analog dazu die Ziele, die sich Bill Friend stellte:

- Der Design-Prozess einer Eingabemaske sollte interaktiv und einfach änderbar erfolgen.
- Der Anwender sollte zur Laufzeit dynamisch durch die Eingabemaske geführt werden.
- Die Runtime-Engine soll die Transaktion zur Datenbank dynamisch erzeugen. Der Entwickler der Maske sollte nicht irgendetwas programmieren müssen, um Standard Insert-, Update-,-Locking-, Delete- und Query-Transaktionen durchführen zu müssen, und ein Rollback sollte bei Fehlern möglich sein. Alle SQL-Befehle sollten dazu noch in einer korrekten Reihenfolge ablaufen.
- Der End User sollte in der Lage sein, auch komplexe Abfragen selbst erstellen zu können, mit relationalen Operatoren wie „<", „>", „LIKE" und in Verbindungen mit anderen Variablen.
- Das relationale Modell sollte logisch mit Transaktionsintegrität umsetzbar sein.
- Dateneingabe von Beziehungen zu anderen Tabellen sollten auch einfach über Wertelisten „„List of Values") möglich sein.
- Für automatisierte SQL-Transaktionen wurden Pre- und Post-Trigger entwickelt, um erweiterte SQL-Logik in einer prozeduralen Abwicklung zu ermöglichen.
- Das Beste aus zwei Welten sollte vereint werden. Dynamisches SQL mit Transaktions-Management und Möglichkeiten der erweiterten Logik mit Laufzeit-Triggern. Diese „SYSTEM-R"-Trigger waren eine Oracle-Vision und noch nicht in der Datenbank implementiert, zuerst in Forms.
- Das Tool sollte auf allen Betriebssystemen lauffähig sein.
- Mit einer „CRT"-Maske sollte das System auf verschiedene Bildschirme anpassbar sein (zum Beispiel 24x80 character grids auf einer „cathode-ray-tube").
- Die Eingabe musste auch ohne Bildschirm via „teletypes" möglich sein, weil nicht jeder Arbeitsplatz zu dieser Zeit einen Bildschirm hatte.

Mit allen Möglichkeiten, die die Programmiersprache C bot, hatte Bill Friend nach etwas mehr als vier Wochen die erste Version von Oracle Forms entwickelt. Larry Ellison nannte das Produkt „IAF" (Interactive Application Facility). Später änderte sich der Name noch ein paar Mal von „Fastforms" über „SQL-Forms", bis schließlich „Oracle Forms und Reports" daraus wurde (ab Version 3).

Weil Larry einem Kunden ein Reporting-Tool versprochen hatte, wurde Bill gefragt, ob er nicht noch schnell eines erstellen könne. Wenn er dafür seinen Urlaub verschieben würde, wäre ein Upgrade (Paris oder Hawaii) möglich. So nahm Bill eine ältere Dokumentations-Lösung von Bruce Scott (FMT) und erweiterte das System um ähnliche Funktionen wie in Forms. Dazu kam die Spezifikation von zusätzlichen Variablen, Select-Statements mit Variablen, Schleifen und IF-Statements. Nach etwa zwei Wochen war RPT fertig und wurde gemeinsam mit der Datenbank ausgeliefert. Bill konnte daraufhin einen schönen Urlaub mit seiner Freundin auf Hawaii machen.

## Die ersten Erfolge

IAF und RPT waren ein Volltreffer für Larry Ellison und halfen, die Kundenliste zu vergrößern. Jeder, von der CIA bis zur Bank of America, große Öl- und Gas-Unternehmen, aber auch kleine IT-Consulting-Firmen, entwickelten nun „Forms und Reports".

In dieser Zeit waren Kunden noch komplett an den Datenbank-Hersteller gebunden; sie konnten selbst keine Software-Applikationen erstellen. Dateneingaben mit „UFI" (später SQL*PLUS) oder über das neue C-API waren nicht benutzerfreundlich.

Nun konnten sie eine eigene Datenbank modellieren, diese physisch mit UFI erstellen und dazu noch komfortable Eingabemasken und Berichte mit IAF/RPT entwickeln. Mit IMP/EXP kamen weitere Tools von Bill Friend dazu, um Datenbanken zu exportieren und migrieren zu können. Die Datentypen „DATE", „TIME" und „TIMESTAMP" wurden von Bill auch noch in die Datenbank integriert, weil Zeitangaben bei der Datenerfassung oft sehr wichtig sind. Langsam wurde das Ganze rund.

## Forms heute

Nach Jahren der Migrations-Abwerbeversuche in andere Welten bildete sich in der DOAG eine solide Community unter der Leitung von Dr. Jürgen Menge, die auf der DOAG Konferenz in Nürnburg und einmal im Jahr (dieses Jahr zur Erscheinung dieser Ausgabe am 21. Februar 2019) zu einem „Forms-Day" zusammenkommen. Waren es im Jahr 2015 drei Oracle-Forms-Vorträge auf der DOAG Konferenz in Nürnberg, so gab es letztes Jahr schon sechzehn Vorträge. Forms lebt und wird weiterentwickelt – nicht so schnell und agil wie andere Werkzeuge, aber stetig und kontinuierlich. Zudem wächst auch die LinkedIn-Gruppe „Forms und Reports Developer" wieder. Nach knapp 9.600 Mitgliedern im letzten Jahr sind es aktuell schon 10.051 Mitglieder in dieser Gruppe – Tendenz steigend.

Viele Forms-Kunden sind inzwischen auf die aktuelle Plattform 12.2.1.3 gewechselt, beginnen mit nativen Modernisierungen und schauen auf die bereits angekündigten neuen Versionen Forms 19 und Forms 20. Mit der Version 19 wird primär ein Maintenance-Release erscheinen. In der Version 20 sollen weitere Änderungen eingeführt werden. Wie bei Oracle-Datenbanken heute üblich, orientiert sich die neue Versionsnummer ab sofort an der Jahreszahl der Produktveröffentlichung.

Sehr wahrscheinlich ist nach aktuellem Stand auch, dass die Version 19 noch einmal mit Reports ausgeliefert wird. Das wird allen Kunden noch etwas mehr Zeit geben, Alternativen zu evaluieren.

Laufzeiten von zwanzig oder dreißig Jahren sind bei Oracle-Forms-Projekten durchaus üblich. Kunden planen aktuell, wie Oracle auch mit seiner Enterprise Business Suite (EBS), die in Teilen immer noch auf Oracle Forms basiert, Support und Weiterentwicklung ihrer Applikationen bis zum Jahr 2030. Das sind Laufzeiten, wie sie mit anderen Entwicklungswerkzeugen eher unüblich sind. Mike Ferrante, Produktchef von Oracle Forms, hat folgende Features für die weiteren Versionen (19/20) in Aussicht gestellt:

- REST-Aufruf-Funktionen für externe Services
- Support für SSO with FSAL
- Support mit dem Identity Cloud Service
- Support mit OAuth
- UI-Verbesserungen (Rahmen, Farben, Custom-Colorschema)
- Konfigurierbare Java-Versionen for FSAL
- Support für Java11 FSAL (also Java 11+)
- Forms-Builder-Integration mit FSAL (aktuell nur HTTP-Plug-in)

Zusammen mit den bereits in Forms12 publizierten Neuerungen haben Forms-Entwickler mit Forms19 dann eine Reihe von Features zur Verfügung, um den Masken ein Facelift zu geben. Forms kann und soll auch im Layout professioneller werden, ohne Grautöne, mit hohen Kontrasten und Farbpaletten, die den Anwendern zugutekommen. Eine weitere neue Option wird die Auslagerung der Forms-Entwicklung und des Deployments in die Oracle-Cloud sein. Ganz aktuell stellt Mike Ferrante in einem Whitepaper von November 2018 die neuen Forms-Entwicklungsmöglichkeiten (DevOps) in der Cloud vor. Zur Oracle Cloud wird es auch einen Vortrag des Autors auf dem nächsten Forms Day geben. Bereiten wir uns schon langsam auf das fünfzigjährige Forms-Jubiläum im Jahr 2029 vor.

## Weitere Informationen

- Interessante Versions-Tabelle mit Erscheinungsdaten der Oracle-Datenbanken: *https://en.wikipedia.org/wiki/Oracle_Database*
- Whitepaper DevOps für Oracle Forms in der Oracle Cloud (11.2018): *https://www.oracle.com/technetwork/developer-tools/forms/documentation/oracleforms-in-dcs-5216022.pdf*

Frank Christian Hoffmann
fch@cologne-data.de

## Aktuelles Interview mit Bill Friend

*Was sind aus Ihrer Sicht die Stärken von Oracle Forms?*

**Bill Friend:** Ich würde sagen, die Automatisierung der Transaktionen, also das dynamische SQL und der korrekte Ablauf von Triggern bei der Änderung von Formular-Daten durch den Benutzer; eine Benutzerführung gesteuert durch Transaktionen und Trigger. Das Herzstück ist jedoch die Produktivität, die viele Funktionen umfasst. Das schien etwas zu sein, das unsere Konkurrenz nicht erreichen konnte. Es war ein Ansatz von 4GL, und wo 4GL aufhörte, ließ sich die Logik prozedural erweitern.

*Forms wird dieses Jahr vierzig Jahre alt und erfreut sich wachsender Beliebtheit. Konnten Sie den Erfolg von Forms vorhersehen?*

**Bill Friend:** Es ist für mich eine große Ehre, dass Oracle Forms, von einigen kleinen Modifikationen abgesehen, immer noch im Stil der anfänglichen Version in Betrieb ist. Ich habe nicht gedacht, dass es so lange existieren würde, und dachte eher, Oracle würde es selber einmal komplett ablösen. Ich wollte auch einmal eine neue Architektur einführen, um einige Restriktionen zu beseitigen. Ich denke, es gibt nur sehr wenige Beispiele eines Systems oder Produkts, das diese Laufzeit in einer Industrie aufweist, die sich so stark geändert hat. Zum ersten Mal hatte ich im Jahr 1984 das Gefühl, etwas Bedeutendes entwickelt zu haben, als ich die vielen Job-Angebote für Forms- und Reports-Entwickler gesehen habe.

*Jemand sagte einmal, dass die Datenbank und Forms die einzigen Produkte gewesen wären, die Oracle selber entwickelt hat. Stimmt das?*

**Bill Friend:** In der Zeit, als wir mit Oracle starteten, war Computer Associates so wie heute Oracle – sie kauften alles auf, um zu wachsen. Heute scheint Oracle eine ähnliche Position eingenommen zu haben. Jedoch sind Tools wie Oracle Designer, Oracle Applications (die originale Version) und viele Data-Warehouse-Tools so wie die, die ich von 1998 bis 2000 entwickelt habe, bei Oracle selbst entstanden.

*Wie hat Larry Ellison den Entwicklungsprozess von Forms geprägt?*

**Bill Friend:** Larry ist der smarteste Typ, den ich je kennengelernt habe, und ein Visionär – nicht nur in Technologie-Fragen, sondern auch darin, wie er eine Firma groß machen kann. Er hat auch einen feinen, sich selbst infrage stellenden Humor, der ihn nahbar macht. Ich mag Larry sehr, aber es kann auch für manche sehr frustrierend sein, mit ihm zusammenzuarbeiten. Genauso mochte ich Bob Minor, er hatte großen Einfluss auf mich. Es gab bei Oracle in den Anfangstagen bei allen einen starken Willen, intelligent, direkt und, wenn nötig, auch sehr scharf zu sein, wenn es um kritische Fragen zu Software-Produkten ging. Jeder war in seinem Bereich sehr gegenwärtig und stand ständig mit anderen in Konkurrenz, um bei dem, was er tat, die besten Ergebnisse zu erzielen. Das ging sogar bis hin zu Hobbies wie Tennis, Poker oder anderen Freizeitbeschäftigungen. Larry stand immer mit allen im Wettbewerb und wollte ständig der Beste sein, auch die besten Ergebnisse bei unserer Arbeit erzielen. Das hatte Einfluss auf unsere entwickelten Produkte.

*Was bedeutet ein Wechsel von Forms (4GL) zu einer 3GL-Sprache?*

**Bill Friend:** In meiner Zeit gab es eine Bewegung, die 4GL-Forms-Lösungen mit Java ablösen wollte. Ich bin mir nicht sicher, ob dieser Weg der richtige war. Bei dem Wechsel zu einer 3GL-Sprache geht viel von dem Funktionsumfang einer 4GL-Applikation verloren, etwa die Fähigkeit, automatisch, zuverlässig und für den Entwickler unsichtbar SQL-Transaktionen abzuwickeln. Das muss in 3GL-Sprachen komplett selbst übernommen werden. Der Vorteil einer nicht prozeduralen 4GL-Sprache besteht darin, dass sie viele komplexe Vorgänge sehr einfach beschreiben kann. Daher lieben Entwickler 4GL-Werkzeuge, nehmen allerdings dabei in Kauf, dass Vorgaben, die mit den vorgegebenen Funktionen in 4GL nicht umsetzbar sind, auch nicht ohne Weiteres möglich sind. Dann wird das 4GL-Werkzeug oft infrage gestellt. Eine prozedurale Sprache bietet dagegen unbegrenzte Flexibilität, was Entwickler lieben, aber dann ohne das ganze Spektrum der 4GL-Funktionen. Forms kann die Vorteile beider Welten vereinen – ein automatisches Transaktions-Management sowie die Flexibilität und Einbindung von prozeduralen Triggern und Funktionen, um den Ablauf programmtechnisch zu beeinflussen.

*Wie viele Entwickler hatte Forms in den ersten Versionen?*

**Bill Friend:** Einen, und am Ende meiner Entwicklungszeit vielleicht noch mit Sohaib Abbasi einen zweiten dazu, der mich dann ablöste. Ich meine, er ist später auch ein Senior Vice President der Tool-Devision geworden. Vielleicht hätte ich in den ersten Jahren noch mehr Mitarbeiter für die Entwicklung von Forms anfordern sollen – aber es machte mir einfach so viel Spaß, alles selbst zu machen.

*Warum macht Oracle so wenig Werbung für Forms?*

**Bill Friend:** Ich würde sagen, es frustriert Einige bei Oracle, dass Forms noch lebt (und anscheinend gedeiht). Die Menschen möchten neue Produkte vorantreiben, von denen sie glauben, dass sie etwas so Altes wie Forms ersetzen könnten. Wie ich in meiner Biografie schon gesagt habe, ist Oracle ein sehr wettbewerbsliebender Ort. Wenn aber die neuen Produkte nicht das Wesentliche dessen ausmachen, was Forms so leistungsfähig und hochproduktiv macht, sowohl für Programmierer als auch für Anwender, wird Forms wahrscheinlich fortfahren und gedeihen. Ich bedaure es manchmal, dass ich nicht selbst eine neue Forms-Version erstellt habe, weil ich glaube, dass ich aus meinen Erfahrungen mit Applications und Data Warehousing einige sehr gute Ideen habe, die, auf den Kernfunktionen von Forms aufbauend, eine Menge der Einschränkungen, die Forms heute hat, beseitigen könnte.

*Eine letzte Frage: Wären Sie bereit, beim nächsten Forms Day per Videokonferenz live dabei zu sein, um die deutsche DOAG-Forms-Gruppe kennenzulernen?*

**Bill Friend:** Sehr gerne.

# Wann PL/SQL nicht verwendet werden sollte

Jürgen Sieben, ConDeS GmbH & Co. KG

Diese Folge erscheint dem Autor unter dem Eindruck mehrerer Veröffentlichungen, auch im Red Stack Magazin, sinnvoll: Im Red Stack Magazin zum Beispiel fand er vor einigen Ausgaben einen Artikel über die Vermeidung von Umgebungswechseln zwischen SQL und PL/SQL. So lesenswert und richtig der Artikel auch ist, so leidet das dort verwendete Beispiel jedoch an einem verbreiteten Problem, das in dieser Kolumne verdeutlicht werden soll.

In einem Kochbuch für PL/SQL-Code, also einem Buch, das Musterlösungen für verbreitete Programmierprobleme anbietet, fand sich folgende, sehr schöne PL/SQL-Prozedur für das Problem, doppelte Einträge in einer Tabelle zu finden *(siehe Listing 1)*. Wunderbar! Bitte versuchen Sie zunächst, den Code zu verstehen, und fragen sich, was daran wohl falsch sein könnte.

## Das Problem

Die Tabelle „EMPLOYEES“ gehört zum Schema „HR“ und enthält 107 Zeilen. Das ist nichts. Daher erstellen wir eine realistischere Simulation und verwenden die Daten der Tabelle „ALL_OBJECTS“ mit etwa 100.000 Zeilen für unsere Beispiele. Die Datenbank-Objekte des Benutzers „SH“ werden anschließend doppelt importiert, damit Verstöße gegen einen zu schaffenden Primärschlüssel auftreten *(siehe Listing 2)*. Dann portieren wir das Rezept auf unsere Daten *(siehe Listing 3)*. Wer eine Pause braucht, ruft die Prozedur nun auf *(siehe Listing 4)*.

Dieser Code hat offensichtlich eine ganze Menge Probleme; eines davon ist die Laufzeit, ein anderes die Tatsache, dass jede Dublette doppelt ausgegeben wird. Zeit zu analysieren, auf welche Weise die Dubletten in der Tabelle aufgespürt werden. Wir iterieren über alle Zeilen der Tabelle „TEST_TABLE“ und öffnen für jede ermittelte Zeile einen weiteren Cursor

```
declare
   cursor emp_cur is
       select *
           from employees
       order by employee_id;
   emp_count number := 0;
   total_count number := 0;
begin
   for emp in emp_cur loop
       select count(*)
           into emp_count
           from employees
           where employee_id = emp.employee_id;
       if emp_count > 1 then
           total_count := total_count + 1;
           -- do_something
       end if;
   end loop;
end;
```

*Listing 1*

```
SQL> create table test_table(
   2       owner varchar2(30),
   3       object_id number,
   4       object_name varchar2(30),
   5       object_type varchar2(30));

Tabelle wurde erstellt.

SQL> insert into test_table(owner, object_id, object_name, object_type)
   2       select owner, object_id, object_name, object_type
   3       from all_objects
   4       union all
   5       select owner, object_id, object_name, object_type
   6       from all_objects
   7       where owner = 'SH';

92553 Zeilen erstellt.
```

*Listing 2*

darauf, um die Anzahl der Zeilen zu zählen, die für diese ID vorhanden sind. Um allem die Krone aufzusetzen, ist die Tabelle „TEST_TABLE“ des äußeren Cursors zudem noch sortiert, obwohl nicht ganz klar ist, wozu das dienen soll. Der äußere Cursor wird also etwa 95.000 Zeilen lesen und im Arbeitsspeicher des Session-Kontextes sortieren, um anschließend für jede Zeile eine Auswertung auf die gleiche Tabelle durchzuführen. Sollte sich dann herausstellen, dass eine Primärschlüssel-Information doppelt vorhanden ist, wurde eine Dublette gefunden und ausgegeben.

Eines der Probleme des Codes besteht darin, dass nicht sichergestellt ist, dass die weiteren „select“-Abfragen in einem lesekonsistenten Zustand zur äußeren Abfrage beantwortet werden. Eine „select“-Abfrage sperrt keine Daten und solange der Session-Isolation-Level nicht auf „SERIALIZABLE“ gestellt wurde, wird jede „select“-Abfrage die Daten der Tabelle so sehen, wie sie zum Zeitpunkt der Abfrage sichtbar waren. Die äußere Abfrage wird in über drei Minuten ausgeführt, das heißt, dass die letzten „select“-Abfragen die gleiche Tabelle über drei Minuten später sehen als die ersten Abfragen der Schleife. Zudem werden immerhin etwa 100.000 „select“-Abfragen abgearbeitet.

Dieser Ansatz atmet prozeduralen Geist, was nicht per se schlecht, in Datenbanken aber beinahe immer falsch ist. Das Mantra lautet:

- Wenn du ein Problem hast, löse es in SQL
- Wenn das gar nicht geht, löse es in PL/SQL
- Wenn das gar nicht geht, löse es in Java
- Wenn das gar nicht geht, löse es in C
- Wenn das gar nicht geht – überlege, was für ein Problem du da hast

Das Wichtige an diesem Mantra ist die Reihenfolge der einzusetzenden Programmiersprachen. Das Problem ist, dass die vorgestellte Prozedur hier einen entscheidenden Fehler macht.

## Lösungsansätze

Versuchen wir, den Code zu retten, indem wir das Problem der Lese-Konsistenz und der vielen „select“-Anweisungen angehen. Inhaltlich sollen doch wohl die

```
SQL> create or replace procedure report_duplicates
  2  as
  3      cursor object_cur is
  4          select *
  5              from test_table
  6          order by object_id;
  7      l_obj_count number := 0;
  8      l_total_count number := 0;
  9  begin
 10      for obj in object_cur loop
 11          select count(*)
 12              into l_obj_count
 13              from test_table
 14              where object_id = obj.object_id;
 15          if l_obj_count > 1 then
 16              dbms_output.put_line(
 17                  obj.object_type || ' ' ||
 18                  obj.object_name || ' existiert doppelt.');
 19          end if;
 20      end loop;
 21  end report_duplicates;
 22  /

Prozedur wurde erstellt.
```

*Listing 3*

```
SQL> set serveroutput on;
SQL> call report_duplicates();

TABLE COSTS existiert doppelt.
TABLE COSTS existiert doppelt.
TABLE PARTITION COSTS existiert doppelt.
TABLE PARTITION COSTS existiert doppelt.
TABLE PARTITION COSTS existiert doppelt.
TABLE PARTITION COSTS existiert doppelt.
TABLE PARTITION COSTS existiert doppelt.
...

Aufruf wurde abgeschlossen.
Abgelaufen: 00:03:14:51
```

*Listing 4*

```
SQL> select object_id, object_type, object_name, count(*) anzahl
  2   from test_table
  3   group by object_id, object_type, object_name
  4   having count(*) > 1;

OBJECT_ID OBJECT_TYPE              OBJECT_NAME               ANZAHL
--------- ------------------------ ------------------------  ----------
   92618  TABLE PARTITION          COSTS                     2
   92613 TABLE PARTITION           COSTS                     2
   92660 TABLE PARTITION           SALES                     2
   92694 TABLE                     TIMES                     2
   92850 INDEX                     CHANNELS_PK               2
   92891 INDEX PARTITION           COSTS_TIME_BIX            2
...
310 Zeilen ausgewählt.

Abgelaufen: 00:00:00.48
```

*Listing 5*

Daten, die nicht als Dublette vorhanden sind, ignoriert werden. Wir wollen also lediglich die Daten sehen, die doppelt auftauchen. Das klingt sehr nach der „select"-Anweisung wie in *Listing 5*.

Eine erste Verbesserung, das ist klar; man achte auf die Zeit. Nun könnte man den Eindruck haben, die Abfrage würde Dubletten doppelt ausgeben (die ersten beiden Zeilen), doch wir sehen an der Objekt-ID, dass dies nicht der Fall ist, sondern hier tatsächlich zwei Partitionen unter dem Namen der Tabelle vorliegen. Diese Abfrage würde sich gut für die Verwendung im Cursor der Prozedur eignen, zumal sich dadurch die innere Abfrage komplett erledigt hätte *(siehe Listing 6)*.

Doch die Frage ist, was wir eigentlich tun möchten. Wollten wir nur das Ergebnis auf den Bildschirm schreiben? Dann hätten wir die Text-Konkatenation in „dbms_output.put_line()" auch direkt in der „select"-Abfrage schreiben können. Wahrscheinlicher ist jedoch, dass wir die Dubletten nur einfach loswerden möchten. Welche Optionen hier zur Verfügung stehen, hängt vom Szenario ab. Spielen wir einmal einige durch.

```
SQL> create or replace procedure report_duplicates
  2  as
  3    cursor object_cur is
  4      select object_id, object_type, object_name, count(*) anzahl
  5        from test_table
  6       group by object_id, object_type, object_name
  7      having count(*) > 1;
  8  begin
  9    for obj in object_cur loop
 10      dbms_output.put_line(
 11        obj.object_type || ' ' ||
 12        obj.object_name || ' existiert doppelt.');
 13    end loop;
 14  end report_duplicates;
 15  /

Prozedur wurde erstellt.

SQL> call remove_duplicates();
TABLE PARTITION COSTS existiert doppelt.
TABLE PARTITION COSTS existiert doppelt.
TABLE PARTITION SALES existiert doppelt.
TABLE TIMES existiert doppelt.
...
Aufruf wurde abgeschlossen.

Abgelaufen: 00:00:00.45
```

*Listing 6*

## Daten aus einer CSV-Datei importieren

Das wahrscheinliche Szenario ist, dass wir die guten Daten in eine Ziel-Tabelle kopieren und die schlechten verwerfen möchten. Das ist ohne PL/SQL machbar *(siehe Listing 7)*.

Die „insert"-Anweisung ignoriert die Dubletten beim Import. Die Lösung nutzt die analytische Funktion „rank()", die eine Reihenfolge über ein Gruppierungskriterium nach einem Sortierkriterium bildet. Da die doppelten Zeilen gleich sind, benötigen wir ein Kriterium, um uns zwischen den beiden zu entscheiden. Hier bietet sich zum Beispiel die „ROWID" an. In der äußeren Abfrage werden anschließend alle Zeilen gefiltert, die keine Dubletten sind, denn diese haben einen „Rang = 1". Die Liste der „ROWID" der verbleibenden Zeilen wird dann zum Identifizieren der Zeilen verwendet, die eingefügt werden sollen. Es ist zu beachten, in diesem Fall keine „row limiting"-Klausel ab Version 12c einzusetzen, da sich diese nicht (nach der Objekt-ID) partitionieren lässt.

```
SQL> create table target_table
  2   as select *
  3   from test_table
  4   where null is not null;

Tabelle wurde erstellt.

SQL> alter table target_table add constraint pk_target_table
  2   primary key(object_id);

Tabelle wurde geändert.

SQL> insert into test_table(owner, object_id, object_name, object_type)
  2   select owner, object_id, object_name, object_type
  3       from test_table
  4   where rowid in (
  5           select row_id
  6               from (select rowid row_id,
  7                      rank() over (
  8                          partition by object_id order by rowid) rang
  9                  from test_table)
  10              where rang = 1);

92243 Zeilen erstellt.
Abgelaufen: 00:00:00.17
```

*Listing 7*

## Daten bereinigen, um einen Primärschlüssel einzurichten

Sind die Daten bereits in der Tabelle enthalten und sollen die Dubletten gelöscht werden, ist auch dies ohne PL/SQL möglich *(siehe Listing 8)*.

Auch hier gilt es, auf die Ausführungszeit zu achten. Die Lösung nutzt wieder die analytische Funktion „rank()“. In der äußeren Abfrage werden nur Dubletten gefiltert, denn diese haben einen „Rang > 1“. Die Liste der „ROWID“ dieser Zeilen wird anschließend zum Bereinigen der Tabelle verwendet.

## Die guten ins Töpfchen, die schlechten ins Kröpfchen

Ist das Ziel die Separierung der beiden Datenmengen, stehen ebenfalls mehrere Wege zur Verfügung. Einerseits könnte man eine „insert“-Anweisung in die Ziel-Tabelle mit einer „log errors“-Klausel ausstatten, die bei einem Primärschlüssel-Verstoß die verstoßende Zeile in eine Fehler-Tabelle schreiben wird *(siehe Listing 9)*.

Alternativ könnte der Weg über eine „multi-table-insert“-Anweisung laufen. Bei dieser Variante würden die Daten durch eine entsprechende „select“-Anweisung mit einem Rang ausgestattet und beim Einfügen der Daten eine Fallunterscheidung nach dem Rang ausgeführt *(siehe Listing 10)*. Die Aufbereitung der Daten ist bekannt, hier entfällt nur die Filterung über die Dubletten, da diese ja für die Fallunterscheidung benötigt werden.

```
SQL> delete from test_table
  2   where rowid in (
  3       select row_id
  4           from (select rowid row_id,
  5               rank() over (
  6                   partition by object_id order by rowid) rang
  7               from test_table)
  8           where rang > 1);

310 Zeilen gelöscht.
Abgelaufen: 00:00:00.06
```

*Listing 8*

```
SQL> call dbms_errlog.create_error_log('TARGET_TABLE', 'TARGET_ERR');

Aufruf wurde abgeschlossen.

SQL> insert into target_table
  2   select *
  3       from test_table
  4           log errors into target_err
  5   reject limit unlimited;

92243 Zeilen erstellt.

Abgelaufen: 00:00:00.23

SQL> select count(*)
  2   from target_err;

  COUNT(*)
----------
       310
```

*Listing 9*

```
SQL> insert first
  2       when rang = 1 then
  3       into target_table(owner, object_id, object_name, object_type)
  4           values(owner, object_id, object_name, object_type)
  5       else into target_err(owner, object_id, object_name, object_type)
  6           values(owner, object_id, object_name, object_type)
  7   select t.*, rank() over (partition by object_id order by rowid) rang
  8       from test_table t;

92553 Zeilen erstellt.

Abgelaufen: 00:00:00.14
```

*Listing 10*

## Fazit

Das Hauptproblem der Programmierung von Datenbanken ist, der Datenbank möglichst wenig im Weg zu stehen. Genau das passiert allerdings gerade routinierten Anwendungsprogrammierern häufig. Zu sehr ist man den prozeduralen Ansatz gewohnt, als dass man sich auf die Suche nach einem mengenorientierten Ansatz macht. SQL hat viele Anstrengungen unternommen, prozedurale Programmierung unnötig zu machen und Standardprobleme direkt aus SQL heraus zu lösen.

Sollten die eingesetzten Techniken unbekannt gewesen sein („log errors“-Klausel, „multi-table-insert“, „analytische Funktionen“ etc.), kann man das doch bitte als Einladung verstehen, sich wieder einmal mit SQL auseinanderzusetzen. Die Lösungen sind im Regelfall kürzer, schneller, besser zu warten und benötigen keine weitreichenden Unit-Tests. Genug Argumente also, PL/SQL auch einmal im Schrank zu lassen.

Jürgen Sieben
j.sieben@condes.de

# Angst vor dem Datengau? Tipps und Tricks für einen besseren Schlaf

Thomas Nau, Kommunikations- und Informationszentrum (kiz) der Universität Ulm

Ransomware – auch als Verschlüsselungs-Trojaner bekannt – ist heute in aller Munde, wenn über die Sicherheit beziehungsweise Integrität von Daten diskutiert wird. Längst ist dieses Problem nicht nur auf private Systeme beschränkt, sondern es werden auch immer wieder Fälle in Behörden, Unternehmen etc. weltweit bekannt, wobei die Dunkelziffer als hoch einzuschätzen ist.

Mindestens ebenso problematisch wie diese offensichtliche und reale Bedrohung sind jedoch Defizite im eigenen Daten-Management. Diese Defizite sind dabei zwar oft der enormen Zunahme von Datenbeständen geschuldet, aber insbesondere auch der fehlenden Anpassung der eigenen IT-Umgebung an die schnell fortschreitende technische Entwicklung. Darüber hinaus halten sich manche Mythen seit Jahren, auch im professionellen Umfeld, und bremsen damit die interne Anpassung von Prozessen.

Wikipedia definiert den Begriff „Ransomware“ wie folgt *(siehe „https://de.wikipedia.org/wiki/Ransomware“)*: „Ransomware (von englisch „ransom“ für „Lösegeld“), auch Erpressungs-Trojaner, Erpressungs-Software, Krypto-Trojaner oder Verschlüsselungs-Trojaner, sind Schadprogramme, mit deren Hilfe ein Eindringling den Zugriff des Computer-Inhabers auf Daten, deren Nutzung oder auf das ganze Computer-System verhindern kann. Dabei werden private Daten auf dem fremden Computer verschlüsselt oder der Zugriff auf sie verhindert, um für die Entschlüsselung oder Freigabe ein Lösegeld zu fordern.“ Derartige Software existiert bereits seit den 1980er-Jahren, erlangte aber erst in den letzten Jahren traurige Berühmtheit als Massen-Phänomen.

Sieht man von gezielten Angriffen gegen Personen oder Unternehmen ab, denen oft spezielle Angriffsmethoden zugrunde liegen, so werden die meisten Betroffenen eher zufällig Opfer. Die Einfallstore sind mannigfaltig; E-Mail-Anhänge und Drive-by-Downloads, ausgelöst von aktiven Inhalten und mit automatischer Weiterleitung auf

entsprechende Web-Seiten, sind nur zwei mögliche Wege. Andererseits ist auch eine wurmartige Verbreitung über unbekannte oder „zero day"-Lücken wie im Fall von WannaCry *(siehe „https://www.heise.de/security/meldung/Alles-was-wir-bisher-ueber-den-Petya-NotPetya-Ausbruch-wissen-3757607.html?artikelseite=all")* möglich ebenso wie eine Kombination mehrerer Verbreitungsmethoden.

Klar ist, dass häufig propagierte Schutzmaßnahmen wie der Einsatz aktueller Virenscanner und Betriebssysteme allenfalls das Ausmaß begrenzen. Diese Aussage wird durch die rasante Verbreitung von WannaCry und Petya auch im Unternehmensbereich gestützt, zumindest sofern man dort einen professionellen IT-Betrieb zugrunde legt. Daraus lässt sich schließen, dass neben Schutz- und Gegenmaßnahmen vor allem auch organisatorische Maßnahmen dringend angeraten sind. Essenziell sind im allgemeinen Bedrohungsszenario zumindest folgende Punkte:

- Erkennung eines Angriffs
- Existenz einer unabhängigen, schreibgeschützten Datenkopie, im Idealfall inklusive Versionierung
- Kenntnis, Dokumentation und Beschränkung von Zugriffsrechten
- Organisatorische Abwägung und Entscheidung: „ease of use" vs. Sicherheit

Die Dauer, für die schreibgeschützte Kopien der essenziellen Datenbestände vorzuhalten sind, wird maßgeblich durch die Zeit bestimmt, die erforderlich ist, um einen Angriff im schlechtesten Fall erkennen zu können. Diese Erkennung gestaltet sich im Allgemeinen als schwierig. Zwar existieren mittlerweile Lösungen, die das Problem auf Basis von Verhaltens-Analysen adressieren, jedoch ist das Grundproblem der Erkennung dem der Spam-Erkennung ähnlich. Während „brute force"-Angriffe klare Muster liefern und oft sogar der Nutzer direkt zur Kasse gebeten wird, sind subtile Angriffe nur äußerst aufwendig erkennbar; etwa wenn nur wenige Dateien pro Zeit-Intervall verschlüsselt sind und diese darüber hinaus spezifisch nach Typ und Alter ausgewählt werden.

Es darf nicht unerwähnt bleiben, dass die Kenntnis, Dokumentation und Beschränkung von Zugriffsrechten oft stiefmütterlich behandelt werden. Als Beispiel seien Netz-Laufwerke genannt, die oft ganzen Abteilungen als für alle beschreibbare Datenablage dienen und damit auch ein entsprechend höheres Schadenspotenzial mit sich bringen.

```
svc:/application/time-slider:default
svc:/system/filesystem/zfs/auto-snapshot:*
```

*Listing 1*

```
INSTANCE=nfs
HOUR=${INSTANCE}-hourly
DAY=${INSTANCE}-daily

svccfg -s svc:/system/filesystem/zfs/auto-snapshot add ${HOUR}
svccfg -s auto-snapshot:${HOUR} addpg zfs application
svccfg -s auto-snapshot:${HOUR} setprop zfs/interval = hours
svccfg -s auto-snapshot:${HOUR} setprop zfs/keep = 23
svccfg -s auto-snapshot:${HOUR} setprop zfs/period = astring: 1
svccfg -s auto-snapshot:${HOUR} addpg general framework
svccfg -s auto-snapshot:${HOUR} setprop general/complete = \
          astring: \"\"
svcadm refresh auto-snapshot:${HOUR}

svccfg -s svc:/system/filesystem/zfs/auto-snapshot add ${DAY}
svccfg -s auto-snapshot:${DAY} addpg zfs application
svccfg -s auto-snapshot:${DAY} setprop zfs/interval = days
svccfg -s auto-snapshot:${DAY} setprop zfs/period = astring: 1
svccfg -s auto-snapshot:${DAY} setprop zfs/keep = 28
svccfg -s auto-snapshot:${DAY} addpg general framework
svccfg -s auto-snapshot:${DAY} setprop general/complete = \
          astring: \"\"
svcadm refresh auto-snapshot:${DAY}
```

*Listing 2*

```
svcadm refresh time-slider:default
svcadm enable  time-slider:default
svcadm enable  auto-snapshot:${HOUR}
svcadm enable  auto-snapshot:${DAY}
```

*Listing 3*

```
pool1/home/kiz@zfs-auto-snap_nfs-hourly-2017-09-24-16h09
pool1/home/kiz@zfs-auto-snap_nfs-daily-2017-09-24-16h58
pool1/home/kiz@zfs-auto-snap_nfs-hourly-2017-09-24-17h09
pool1/home/kiz@zfs-auto-snap_nfs-hourly-2017-09-24-18h09
```

*Listing 4*

## Hilfe in Form von Solaris-Bordmitteln

Solaris bietet bereits seit Jahren Hilfsmittel an, um die Integrität und Sicherheit von Daten in hohem Maß zu unterstützen. Darren Moffat schreibt darüber bereits im Jahr 2008 in seinem Blog *(siehe „https://blogs.oracle.com/darren/making-files-on-zfs-immutable-even-by-root")* den Artikel „Making files on ZFS Immutable (even by root!)". Grundlage ist dabei ZFS. Durch dessen Verfügbarkeit in weiteren Betriebssystemen, sehr früh schon in FreeBSD und nun stark zunehmend auch in Linux, gilt das nachfolgend Gesagte weitestgehend auch für den Einsatz dieser Betriebssysteme.

Der Einsatz dieser Techniken auf dem Fileserver kann im Falle der Bedrohung durch Ransomware dazu genutzt werden, die dort gespeicherten Daten zu schützen beziehungsweise den Verlust drastisch zu

minimieren. Generell lassen sich die skizzierten Lösungen auch für Datenbank-, Web- oder Mail-Server einsetzen. Endanwender sollten dort jedoch keinen direkten Zugriff auf die Filesysteme erhalten. Damit steht dann die Schnittstelle, die die heutige Ransomware verwendet, nicht zur Verfügung.

```
# ls -l
total 49
-rw-r--r--   1 root  root 15205 Sep 24 19:38 etc.txt
-rw-r--r--   1 root  root  8694 Sep 24 19:37 var.tmp.txt

# for s in *.txt; do
      t=$(echo $s | encrypt -a aes -k /tmp/MyKey |
          base64 -w 0 | tr '/' '@')
      encrypt -a aes -k /tmp/MyKey -i "$s" -o "$t"
      rm "$s"
done

# ls -l
total 50
-rw-r--r--   1 root  root  8744 Sep 24 19:40 AAAAAQAAA+h07s…
-rw-r--r--   1 root  root 15256 Sep 24 19:40 AAAAAQAAA+jUjV4I…
```

*Listing 5*

```
# zfs diff -o user -o oldname -o name rpool/test@original
M  root    -   /test/
-  root    -   /test/var.tmp.txt
-  root    -   /test/etc.txt
+  root    -   /test/AAAAAQAAA+jUjV4IEU…
+  root    -   /test/AAAAAQAAA+h07s8rfI…
```

*Listing 6*

```
# zfs list -o space -r mail
NAME  AVAIL  USED   USEDSNAP   USEDDS  USEDREFRESERV    USEDCHILD
mail  50.0T  10.9T  5.35T      5.57T   0                0
```

*Listing 7*

```
# zfs list -o space -r cifs
NAME  AVAIL  USED   USEDSNAP   USEDDS  USEDREFRESERV    USEDCHILD
cifs  27.2T  6.25T  1019G      5.26T   0                0
```

*Listing 8*

```
nau@alderaan:/test> ls -/v logfile
-rw-r-----   1 nau      kizinfra   1808 Sep 25 10:45 logfile
      {archive,nohidden,noreadonly,nosystem,noappendonly,
      nonodump,noimmutable,av_modified,noav_quarantined,
      nonounlink,nooffline,nosparse,nosensitive}
```

*Listing 9*

```
nau@alderaan:/test>  chmod S+vappendonly,vnounlink logfile
chmod: ERROR: cannot set the following attributes on logfile: not privi-
leged
      {appendonly,nounlink}
```

*Listing 10*

## Solaris Hilfsmittel #1: ZFS-Snapshots

Hinweis: Um die Lesbarkeit zu verbessern, wurde die Ausgabe von Programmen oder deren Quellcode im Folgenden teilweise gekürzt oder umformatiert.

Ergänzend zu lokalen Backup-Strategien bietet ZFS – in der Zwischenzeit auch einige andere Filesysteme – mit seiner Snapshot-Funktionalität eine sehr einfache, Ressourcen-schonende und schnelle Möglichkeit, die genannte Forderung nach schreibgeschützten Kopien von Daten zu bedienen. ZFS nutzt hierzu seine Basis-Technologie „copy-on-write" aus. Die einer Änderung als Basis dienenden Blöcke werden dabei nicht in die Freiliste des Filesystems überführt, sondern bleiben den jeweiligen Snapshots, jetzt nicht mehr veränderbar, zugeordnet. Die Solaris-Service-Management-Facility-Services *(SMF, siehe „http://www.oracle.com/technetwork/articles/servers-storage-admin/intro-smf-basics-s11-1729181.html")* übernehmen die notwendige Steuerung von Erzeugung und Löschung der Snapshots nach vordefinierten Regeln *(siehe Listing 1)*.

In der täglichen Praxis hat es sich bewährt, nicht die Default-Instanzen zu verwenden, sondern für jede Aufgabe eigene Instanzen zu erzeugen. Dies gestaltet sich einfach, wie *Listing 2* zeigt. Die beiden exemplarisch definierten, neuen Services erzeugen stündliche und tägliche Snapshots. Letztere stehen für 28 Tage zur Verfügung, während die stündlichen jeweils die letzten 24 Stunden abdecken.

Nach diesem Rezept lassen sich auch grob- oder feingranularere Snapshots erzeugen. Anfangs sind Debug-Meldungen hilfreich. Sie werden über das Setzen einer SMF-Property eingeschaltet „svccfg -s time-slider setprop daemon/verbose = true" und finden sich dann in der Datei „/var/svc/log/application-time-slider:default.log". Jetzt müssen die Dienste nur noch aktiviert werden *(siehe Listing 3)*. Die dadurch entstehenden Snapshots lassen sich auch aufgrund ihres Namens einfach von anderen, gegebenenfalls manuell erzeugten, unterscheiden *(siehe Listing 4)*.

Um zu verstehen, wie Snapshots gegen typische Ransomware-Probleme helfen, muss man sich vor Augen führen, welche File-Operationen im Zuge der Verschlüsselung auf die Dateien angewendet werden. Vereinfacht werden folgende Schritte pro Datei durchlaufen:

- Öffnen der Quelldatei „S“
- Anlegen und Öffnen einer Zieldatei „T“, deren Namen aus dem verschlüsselten Dateinamen von „S“ besteht. Hier ist im Allgemeinen ein Encoding notwendig, um spezielle Zeichen wie „:“ zu eliminieren
- Verschlüsselung der Daten aus „S“ nach „T“
- Löschen der Quelldatei „S“

Die Bash-Kommandos in *Listing 5* illustrieren dies. Mithilfe des Kommandos „zfs diff“ lassen sich Änderungen des aktuellen Datasets bezüglich eines Snapshots anzeigen. Für das Beispiel in *Listing 6* wurde vor dem Angriff ein Snapshot mit dem Namen „original“ angelegt.

Erkennbar sind die Änderung des Verzeichnisses „/test“ (M) sowie die Löschung (-) und Erzeugung (+) zweier Dateien des Besitzers „root“. Mit diesen Informationen sowie dem bei Bedarf zusätzlich einblendbaren Zeitstempel lassen sich die betroffenen Dateien sehr gut eingrenzen. Anschließend können die Originaldaten aus dem für jede Datei jeweils neuesten geeigneten Snapshot zurückkopiert werden. Je nach zeitlicher Granularität der Snapshots sowie dem Startzeitpunkt des Angriffs sind dessen Auswirkungen erheblich reduziert oder gar vernachlässigbar.

Bedenken hinsichtlich des Plattenplatzes, der für die Snapshots zusätzlich bereitzustellen ist, lassen sich durch die beiden folgenden Beispiele leicht zerstreuen. Im ersten Fall handelt es sich um einen Mailserver. Die letzten 24 Stunden sind mit stündlichen, die vergangenen drei Monate mit täglichen Snapshots abgedeckt *(siehe Listing 7)*.

Die Snapshots schlagen hier mit rund 50 Prozent des gesamten Bedarfs zu Buche. Dabei ist aber zu bedenken, dass gelöschte E-Mails, darunter auch SPAM, erst nach 90 Tagen wirklich aus dem System verschwinden. Ein Solaris CIFS Server, der rund 200 Arbeitsplätze im administrativen Bereich bedient, ist ein weiteres Beispiel. Die nahezu identische Konfiguration deckt hierbei nur den letzten Monat mit täglichen Snapshots ab, wird jedoch durch ein herkömmliches Backup-System ergänzt *(siehe Listing 8)*. In diesem Fall beträgt der zusätzliche Bedarf rund 20 Prozent, ein geringer Aufschlag für den gewonnenen zusätzlichen Schutz.

```
nau@alderaan:/test> ppriv -f +D -e chmod S+vappendonly,vnounlink logfile
chmod[1289]: missing privilege "file_flag_set"
(euid = 3201, syscall = "write") for "/test/logfile"
at zfs_setattr+0xc90
chmod: ERROR: cannot set the following attributes on logfile: not privi-
leged
      {appendonly,nounlink}

nau@alderaan:/test> ppriv -lv file_flag_set
file_flag_set
  Allows a process to set immutable, nounlink or appendonly
  file attributes.
```

*Listing 11*

```
root# ppriv -f +D -e chmod S+vappendonly,vnounlink logfile
root# ls -/v logfile
-rw-r-----   1 nau      kizinfra   1808 Sep 25 10:45 logfile
      {archive,nohidden,noreadonly,nosystem,appendonly,
      nonodump,noimmutable,av_modified,noav_quarantined,
      nounlink,nooffline,nosparse,nosensitive}
```

*Listing 12*

Tools wie ZnapZend *(siehe „http://www.znapzend.org“)* ersetzen nicht nur die von Solaris bereitgestellte Time-Slider-Funktionalität, sondern erweitern diese zum Teil erheblich. Unter anderem erlaubt es ZnapZend, die Snapshots mithilfe der ZFS-Mechanismen „send/receive“ auf andere Systeme zu übertragen. Unterschiedliche Retention-Zeiten für die Snapshots pro System sind ebenfalls unterstützt.

## Solaris Hilfsmittel #2: ZFS-Datei-Attribute

Vorab ein Hinweis für alle, die die Beispiele nachvollziehen möchten. Die mit Solaris ausgelieferten GNU-Utilities unterstützen die zusätzlichen Funktionalitäten nicht. Daher muss „/usr/bin“ zwingend im Pfad vor „/usr/gnu/bin“ stehen. Außerdem sind gegebenenfalls Shell-Aliasse für Kommandos zu prüfen, falls das Ergebnis nicht dem Erwarteten entspricht; Analoges gilt für MANPATH.

ZFS-Datei-Attribute sind im Falle von Ransomware weniger effektiv, da sie im Allgemeinen auch die tägliche Arbeit zu sehr einschränken. In speziellen Fällen, etwa bei überwiegend statischen Dateien, sind sie jedoch extrem hilfreich. So lässt sich mit ihrer Hilfe die Manipulation von Web-Server-Konfigurationen, PHP-Installationen sowie der zugehörigen Log-Dateien zuverlässig unterbinden, sofern diese Dienste nicht in Immutable-Zones *(siehe „https://docs.oracle.com/cd/E53394_01/html/E54762/glglv.html“)* laufen können. Einige der Attribute sind auch in der Lage, „root“ vom Zugriff auszuschließen, zumindest ohne vorherigen Eingriff. Daher ist ein Test der Gesamt-Funktionalität nach Aktivierung dieser Schutzmaßnahmen dringend angeraten. Um mögliche Verschleierungen zu verhindern, sollten Log-Dateien im Allgemeinen nur im Sinne eines Anhängens neuer Daten veränderbar sein *(siehe Listing 9)*. Für den geplanten Einsatzzweck werden die Attribute „appendonly“ und „nounlink“ gesetzt, die dann gemeinsam sowohl das Löschen als auch das Überschreiben der Datei verhindern *(siehe Listing 10)*.

Offensichtlich erlaubt Solaris es einem normalen Account nicht, diese Attribute zu verändern. Warum? Um dies herauszufinden, kommen die Debugging-Möglichkeiten des Least-Priviledge-Systems *(siehe „https://docs.oracle.com/cd/E53394_01/html/E54830/prbac-2.html#scrolltoc“)* zum Einsatz *(siehe Listing 11)*.

Das Problem lässt sich auf zwei Arten lösen. Zum einen könnten dem Account die notwendigen zusätzlichen Privilegien zugewiesen werden, was aber mit einer Schwächung des Schutzes einhergeht. Daher ist es sinnvoller, die Attribute durch einen anderen – privilegierten – Account passend zu setzen *(siehe Listing 12)*. Die Wirkung ist die erwünschte, selbst mit Root-Rechten *(siehe Listing 13)*.

Im Falle von Software-Installationen und insbesondere von Konfigurationsdateien, hier beispielsweise die eines Apache-Webservers, ist das Setzen des „immutable"-Attributs hilfreich. Auch dazu sind passende Privilegien notwendig *(siehe Listing 14)*. Um Anpassungen an der Konfiguration vornehmen zu können, muss die Datei also zuerst entsperrt sein *(siehe Listing 15)*.

## Daten-Management

Neben technischen Aspekten sind auch die organisatorischen Belange des Daten-Managements von größter Bedeutung. In diesem Umfeld existieren jedoch auch einige Legenden, die über die letzten Jahrzehnte mehr oder weniger gepflegt wurden, ohne sie jemals auf den Prüfstand aktueller Technologien zu stellen.

Nach Erfahrung des Autors sind im Datenbank-Umfeld viel häufiger Sicherungs- und auch Recovery-Szenarien implementiert und dokumentiert, als dies im Bereich von File-, Web- oder Mail-Servern der Fall ist. Regelmäßige realitätsnahe Tests und damit die Überprüfung, ob die Anforderungen an Ausfallzeiten beziehungsweise deren Folgen tragbar sind, finden jedoch in nur wenigen Fällen statt. Das bringt uns insbesondere hinsichtlich eingesetzter Backupstrategien zu einigen Mythen, die sicherlich nicht nur im Hochschulumfeld anzutreffen sind:

**Mythos #1: „**Nutzer kennen ihre Daten sowie die sich daraus ergebenden Anforderungen und folgen einem Daten-Management-Konzept." System-Verwaltern ist hier meist kein Vorwurf zu machen, vielmehr scheitert es oft an fehlenden organisatorischen Vorgaben und Entscheidungen. Zu den zu beantwortenden Fragen gehören unter anderem:

- *Welche Daten sind heiß/wichtig und wie oft wird diese Frage neu beantwortet?*
  Setzen von Prioritäten bei Wiederherstellung
- *Liegen Daten strukturiert/chaotisch vor (Spotlight-Problem)?*
  Einfache Auswahl und Zeitgewinn bei Wiederherstellung
- *Wie lange kann eine Gruppe/Einrichtung ohne Daten produktiv arbeiten?*
  Entscheidet über Technologie

**Mythos #2:** „Die Netzwerk-Bandbreite ist der begrenzende Faktor, Clients brauchen 10 GE!" Mag dies in Einzelfällen zutreffen – vor allem bei mobilen Endgeräten –, so beschränkt im täglichen Betrieb im Allgemeinen die Suche im zugrunde liegenden File- und Speicher-System die Geschwindigkeit des Backups. Nach Erfahrungen des Autors liegen die durchschnittlichen Bandbreiten-Anforderungen inkrementeller Backups deutlich unter dem, was ein 1-Gb/s-Interface zu liefern vermag. Ein wichtiger Rückschluss daraus ist, dass auch im Falle eines Full-Restore in den allermeisten Fällen das Speicher- und File-System der wesentliche zeitbestimmende Faktor ist.

**Mythos #3:** „Ein Archiv ist doch nur Backup mit langen Lagerzeiten." Neben einem grundlegend anderen Verständnis von Bibliothekaren hinsichtlich der Eigenschaften eines Archivs existieren auch im IT-Umfeld klare Abgrenzungen zu Backups: Sind Archivdaten mit Metadaten angereichert, vereinfacht das die Suche erheblich, und werden Archivdaten vom Quellsystem gelöscht, sind sie damit nach einiger Zeit auch im Backup gelöscht. Kurz gesagt wird oft von „Archivdaten" gesprochen, aber in Wirklichkeit sind „kalte Daten" gemeint *(siehe Mythos #1)*.

**Mythos #4:** „Es besteht ein Plan für den Fall der Fälle." Die Existenz eines derartigen Planes ist zwar eine notwendige, jedoch keine hinreichende Voraussetzung. Vielmehr bedarf jeder Plan regelmäßiger und vor allem realitätsnaher Tests, die sämtliche Schritte abdecken beziehungsweise simulieren:

- Zusammenstellung eines Teams mit allem notwendigen Wissen
- System (beschaffen und) wiederherstellen
- Daten wiederherstellen
- Anwendung mit restaurierten Daten bekannt machen, also Transaction-Logs einspielen etc.

```
nau@alderaan:/test> > logfile
bash: logfile: Not owner

nau@alderaan:/test> rm logfile
rm: logfile not removed: Not owner

nau@alderaan:/test> mv logfile tmp123456
mv: cannot rename logfile to tmp123456: Not owner

nau@alderaan:/test> echo APPENDED_LINE >> logfile

root# rm logfile
rm: remove logfile (yes/no)? y
rm: logfile not removed: Not owner
```

*Listing 13*

```
root# chmod S+vimmutable httpd.conf

root# > httpd.conf
-bash: httpd.conf: Not owner

root# rm httpd.conf
rm: httpd.conf: override protection 644 (yes/no)? y
rm: httpd.conf not removed: Not owner
```

*Listing 14*

```
root# chmod S-vimmutable httpd.conf
root# vi httpd.conf
root# chmod S+vimmutable httpd.conf
```

*Listing 15*

Besonderes Augenmerk sollte unter anderem auf folgende kritische Punkte gelegt werden:

- Schlüssel-Management im Falle verschlüsselter Daten
- Priorisierung der Daten für die Wiederherstellung
- Die Zeiten, die für die Wiederherstellung einzelner Bereiche notwendig sind

Aus den aufgeworfenen Fragen und Kritikpunkten wird klar, dass im Falle von Datensicherung – aber viel mehr noch beim Daten-Management – organisatorische Entscheidungen von grundlegender Bedeutung sind, es sich also mitnichten um ein rein technisches Problem handelt.

Paradigmen und Aussagen kritisch zu hinterfragen, ist der Schlüssel zum Erfolg und hat erhebliche Auswirkungen auf die notwendigen Leistungsmerkmale künftiger Lösungen, da diese immer spezifischer in einzelne Bereiche passen müssen. Ein Beispiel ist die eingangs diskutierte Problematik von Ransomware oder anderen Angriffen. Diese erfordern spezielle und über die übliche Datensicherung hinausgehende Maßnahmen, stellen jedoch, wie das nachfolgende Beispiel zeigt, keinen Einzelfall dar.

## Wo ein herkömmliches Backup keinen Sinn ergibt

E-Mail-Systeme sind heute oft genug als zentrales Kommunikationsmittel kleiner und großer Organisationen eines der kritischsten IT-Elemente, bei denen selbst kurze Ausfälle wenig Toleranz finden. Je nach technischer Implementierung kann das System hundert Millionen Dateien oder mehr enthalten, deren initiale Sicherung mit herkömmlichen Backup-Methoden mehr als zwei Wochen dauern würde. Entscheidend dafür ist nicht die Summe der Größe der Daten – es sind die Random-IO-Zugriffe auf das Filesystem. Dieses liegt häufig, wie auch in diesem Fall, auf herkömmlichen SAN-Systemen.

Das Ergebnis legt den Schluss nahe, dass eine vollständige Wiederherstellung ebenfalls wenigstens einige Tage in Anspruch nehmen würde und in diesem Zeitraum eine Nutzung des Mail-Systems bestenfalls nur sehr eingeschränkt möglich wäre. Herkömmliche Sicherungskonzepte sind in diesem Fall also nicht zielführend. Eine vollständige Migration auf SSD-basierte Speicher kommt aus wirtschaftlichen Gründen nicht in Betracht. Nachfolgend die Skizze einer Lösung, wie sie aktuell an der Universität Ulm im Einsatz ist:

- *Wahl von ZFS als Filesystem*
  Vorhaltung von je 24 stündlichen und 90 täglichen Snapshots. Die feinere zeitliche Granularität von nur einer Stunde minimiert das Zeitfenster bezüglich der versehentlichen Löschung von Daten erheblich gegenüber einer normalen, täglichen Datensicherung.

- *Cyrus-IMAP-Server-Feature „delayed expunge“*
  E-Mails werden für 48 Stunden nur als gelöscht markiert und sind für den E-Mail-Nutzer nicht mehr sichtbar. Im Filesystem werden sie erst im Anschluss gelöscht und sind damit in jedem Fall in den ZFS-Snapshots enthalten.

- *Asynchrone IMAP-Replikation*
  Der IMAP-Server repliziert alle Änderungen am Mail-Store mit eigenen Mitteln asynchron auf ein System an einem anderen Standort. Die auftretenden typischen Verzögerungen liegen im Sekundenbereich. Dies vervollständigt den Schutz gegen Totalausfall des primären Standorts.

Überlegungen hinsichtlich der Dauer einer vollständigen Wiederherstellung erfolgen für alle kritischen Systeme. Dabei wird auch nicht außer Acht gelassen, dass die Verfügbarkeit der Daten nicht notwendiger Weise auch die Verfügbarkeit der Anwendung beziehungsweise Dienstleistung bedeutet.

## Fazit

Solaris bietet seit Jahren geeignete Hilfsmittel, um die Datensicherheit und -integrität erheblich zu verbessern. Neben dem Einsatz derartiger Mittel ist jedoch ein Daten-Managementkonzept sowie dessen kontinuierliche Fortschreibung und Hinterfragung ein Schlüssel zum Erfolg.

Thomas Nau
thomas.nau@uni-ulm.de

# Den ODA-Betrieb sicherstellen: Worst-Case-Szenario mit fünfzigprozentiger Überlebenschance bei RAC

Andrzej Rydzanicz, OPITZ CONSULTING Polska Sp.z.o.o

In diesem Artikel teilt der Autor seine langjährigen Erfahrungen im Umfeld der Oracle Database Appliance (ODA) und stellt die Prozedur vor, die für die reibungslose Wiederherstellung von nur einem ODA-Knoten erforderlich ist. Hinzu kommt ein Blick auf die Gründe für diese Prozedur und ihre Vorteile.

Im Oracle-Umfeld sind Restores an und für sich nicht ungewöhnlich. Verschiedene Probleme wie Block Corruption oder User-Fehler können dazu führen, dass die Daten zurückgespielt werden müssen. Zudem können Hardware-Probleme auftreten, die einen sauberen Start des Datenbank-Servers verhindern.

Die ODA unterscheidet sich in dieser Hinsicht nicht von anderen Maschinen. Im Falle eines Worst-Case-Szenarios bietet sie eine gewisse Unfallsicherheit, etwa mithilfe von RAC, weil sie aus zwei physikalischen Servern besteht. So weit gut und schön, aber was soll man machen, wenn die ODA nichtsdestotrotz Soft- oder Hardware-technisch nicht funktioniert?

Nun ja, vorausgesetzt es gibt ein brauchbares RMAN-Backup auf einem externen Speicher, lassen sich die Daten mit fast hundertprozentiger Sicherheit auf einem anderen Server oder auf dem ursprünglichen Server, also dort wo eine Neu-Installation stattgefunden hat, wiederherstellen.

Szenarios, bei denen die ganze ODA neu aufgesetzt werden muss, sind im Internet gut dokumentiert. Es gibt im Oracle-Support-Portal viele MOS-Einträge, die einen Bare-Metal-Restore der ODA detailliert beschreiben. Die MOS erläutern allerdings

immer einen Bare-Metal-Restore, der mit beiden ODA-Knoten zu tun hat. Was aber ist zu tun, wenn nur ein Knoten defekt ist?

Der Autor erinnert sich noch gut an zwei Fälle, in denen die End-User nicht sicher waren, dass einer der Datenbank-Knoten komplett defekt war. Da man es hier mit einem RAC zu tun hat, ist dieser Fall gar nicht so ungewöhnlich. Da fragt man sich, warum man den ganzen Cluster neu aufbauen soll, wenn man ebenso gut und transparent nur einen Knoten wiederherstellen und zum Cluster hinzufügen könnte. Das spart immerhin Geld und Zeit, und weil die End-User fast nichts davon merken, feiern sie einen als „Held des Tages" … Es lohnt sich also durchaus, eine solche Restore-Prozedur einmal genauer durchzuspielen. Die Restore-Prozedur für die ODA lässt sich in drei verschiedene Abschnitte unterteilen:

- Aufräumen der Konfiguration in Bezug auf Oracle Inventory und Cluster Registry vor dem Cluster-Aufbau
- Re-Imaging des defekten Knotens
- Konfiguration und Hinzufügen des neuaufgesetzten Knotens im Cluster

## Aufräumen der Konfiguration

Eine Aufgabe beim Aufräumen der Konfiguration ist das Löschen von Informationen zum defekten Knoten im Oracle Repository. Mit „oakcli" lässt es sich leicht herausfinden, welche Homeserver aktuell auf der Maschine installiert sind *(siehe Listing 1)*. Die Löschaufgaben müssen für jeden RDBMS Home durchgeführt werden, in diesem Fall für einen 12c RDBMS Home *(siehe Listing 2)*.

Wie schon erwähnt, sollte die Löschaktion für jeden Homeserver erfolgen, den das „oakcli show dbhomes"-Kommando ausgibt, um einen konsistenten Zustand zu erreichen. Die bis jetzt beschriebenen Schritte bezogen sich zwar lediglich auf das Oracle Repository. Sie wurden natürlich auch mit dem Oracle-User durchgeführt. Im Prinzip müssen wir auch die Cluster-Ressourcen des defekten Knotens als Grid aus der Konfiguration entfernen. Es handelt sich hierbei um die VIP-Komponenten. Dafür fragen wir zunächst die Konfiguration ab und speichern diese für die spätere Verwendung *(siehe Listing 3)*. Danach sind sie zu stoppen und zu löschen *(siehe Listing 4)*. Sobald der VIP gelöscht wurde, lässt sich der Knoten mit „/u01/app/12.1.0.2/grid/bin/crsctl delete node -n odanode1" sicher aus der Konfiguration entfernen.

Generell ist es wichtig, den Cluster vor dem Neuaufbau so zu gestalten, als wäre der defekte Knoten mit den dazugehörigen Ressourcen niemals dagewesen. Ebenso ist es wichtig, die SSH-Informationen des Oracle- und Grid-Users auf dem gebliebenen Knoten mithilfe entsprechender Schlüssel („SH Keys") zu löschen. Die Informationen über den Knoten sind

```
[root@odanode2 ~]# oakcli show dbhomes
Oracle Home Name     Oracle Home version              Home Location
----------------     -------------------              -------------
OraDb11203_home1     11.2.0.3.12(19121548,17592127)   /u01/app/oracle/
product/11.2.0.3/dbhome_1
OraDb12102_home1     12.1.0.2.1(19303936,19392604) /u01/app/oracle/prod-
uct/12.1.0.2/dbhome_1
```

*Listing 1*

```
export ORACLE_HOME=/u01/app/oracle/product/12.1.0.2/dbhome_1
cd $ORACLE_HOME/oui/bin
./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME "CLUSTER_
NODES=odanode2"
```

*Listing 2*

```
srvctl config vip -node odanode1
srvctl status vip -node odanode1
```

*Listing 3*

```
srvctl stop vip -vip odanode1-vip
srvctl remove vip -vip odanode1-vip
```

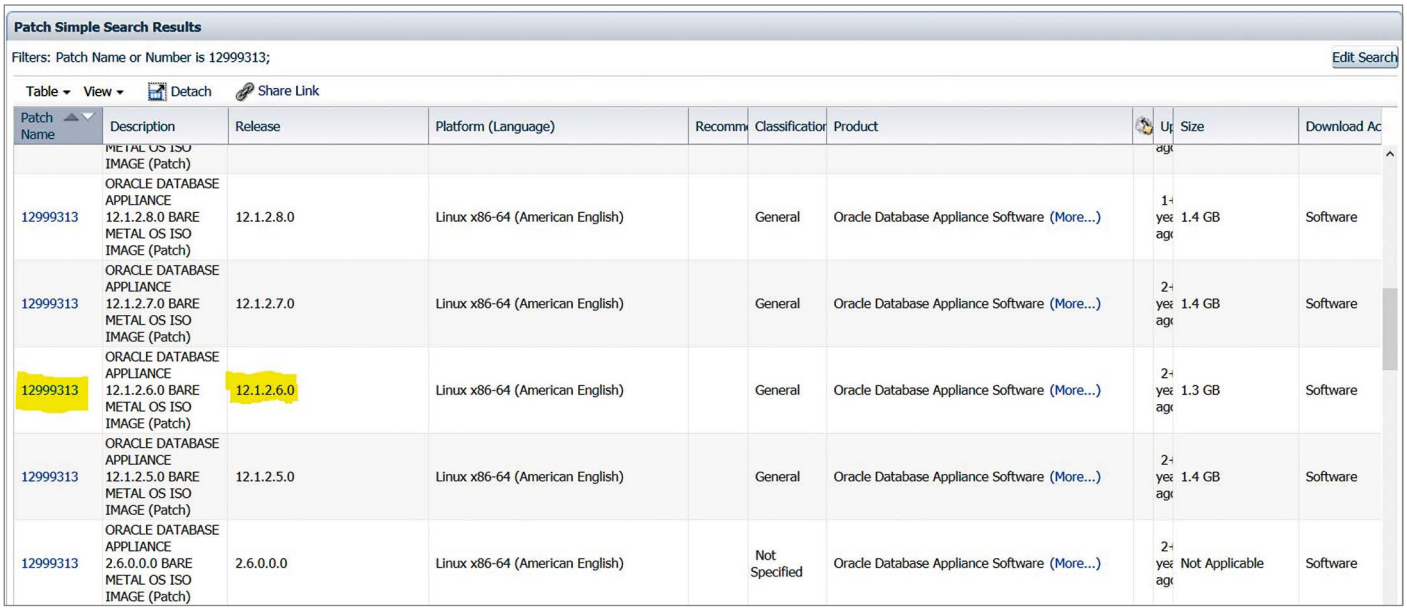
*Listing 4*

```
export ORACLE_HOME=/u01/app/12.1.0.2/grid cd $ORACLE_HOME/oui/bin
./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME „CLUSTER_
NODES=odanode2" CRS=TRUE
```

*Listing 5*

```
System Version  Component Name            Installed Version         Supported Version
--------------  --------------            -----------------         -----------------
12.1.2.6.0
                Controller_INT            4.230.40-3739             Up-to-date
                Controller_EXT            06.00.02.00               Up-to-date
                Expander                  0018                      Up-to-date
                SSD_SHARED {
                [ c2d16,c2d17,c2d18,      A122                      A29A
                c2d19 ]
                [ c2d20,c2d21,c2d22,      A122                      A29A
                c2d23 ]
                        }
                HDD_LOCAL                 A720                      Up-to-date
                HDD_SHARED                P554                      Up-to-date
                ILOM                      3.2.4.52 r101649          Up-to-date
                BIOS                      30050100                  Up-to-date
                IPMI                      1.8.12.4                  Up-to-date
                HMP                       2.3.4.0.1                 Up-to-date
                OAK                       12.1.2.6.0                Up-to-date
                OL                        6.7                       Up-to-date
                GI_HOME                   12.1.0.2.160119(2194      Up-to-date
                                          8354,21948344)
                DB_HOME                   12.1.0.2.160119(2194      Up-to-date
                                          8354,21948344)
```

*Abbildung 1: Auswahl des Patches (1)*



**Patch Simple Search Results**

Filters: Patch Name or Number is 12999313;

Edit Search

Table ▾ View ▾ Detach Share Link

| Patch Name | Description | Release | Platform (Language) | Recomm | Classificatio | Product | Up | Size | Download Ac |
|---|---|---|---|---|---|---|---|---|---|
| | METAL OS ISO IMAGE (Patch) | | | | | | | | |
| 12999313 | ORACLE DATABASE APPLIANCE 12.1.2.8.0 BARE METAL OS ISO IMAGE (Patch) | 12.1.2.8.0 | Linux x86-64 (American English) | | General | Oracle Database Appliance Software (More...) | 1- yea ago | 1.4 GB | Software |
| 12999313 | ORACLE DATABASE APPLIANCE 12.1.2.7.0 BARE METAL OS ISO IMAGE (Patch) | 12.1.2.7.0 | Linux x86-64 (American English) | | General | Oracle Database Appliance Software (More...) | 2- yea ago | 1.4 GB | Software |
| 12999313 | ORACLE DATABASE APPLIANCE 12.1.2.6.0 BARE METAL OS ISO IMAGE (Patch) | 12.1.2.6.0 | Linux x86-64 (American English) | | General | Oracle Database Appliance Software (More...) | 2- yea ago | 1.3 GB | Software |
| 12999313 | ORACLE DATABASE APPLIANCE 12.1.2.5.0 BARE METAL OS ISO IMAGE (Patch) | 12.1.2.5.0 | Linux x86-64 (American English) | | General | Oracle Database Appliance Software (More...) | 2- yea ago | 1.4 GB | Software |
| 12999313 | ORACLE DATABASE APPLIANCE 2.6.0.0.0 BARE METAL OS ISO IMAGE (Patch) | 2.6.0.0.0 | Linux x86-64 (American English) | | Not Specified | Oracle Database Appliance Software (More...) | 2- yea ago | Not Applicable | Software |

*Abbildung 2: Auswahl des Patches (2)*

in der Cluster Registry weiterhin vorhanden. Als Grid-User sollte man diese Einträge im Repository gänzlich entfernen *(siehe Listing 5)*. Nach diesem Schritt kann begonnen werden, den defekten Knoten neu aufzubauen (Re-Imaging).

## Re-Imaging des defekten Knotens

Bei einer Störung wie dieser muss man natürlich zunächst die Server-Hardware wieder zum Laufen bringen. Die defekten Hardware-Teile können vom Support ausgetauscht werden. Der Autor hat allerdings die Erfahrung gemacht, dass die Software-Probleme auf Betriebssystem-Ebene oft weitaus hartnäckiger sind. Dabei hilft letztendlich nur ein Bare-Metal-Restore.

Da wir es hier jedoch mit nur einem defekten Knoten zu tun haben, unterscheidet sich die Prozedur des Bare-Metal-Restore in diesem Fall von einem Restore der gesamten ODA. Die ersten Schritte sind allerdings gleich: Zuerst suchen wir den entsprechenden Patch mit dem OS-Image, um die Maschine zu installieren. Das richtige Image findet sich unter dem Patch 12999313. Dabei immer den Patch für die entsprechende OAK-Version des überlebten Knotens auswählen *(siehe Abbildungen 1 und 2)*.

Sobald der Patch heruntergeladen ist, beginnt die Installation des Knotens *(siehe Abbildungen 3 bis 5)*.

- Einloggen zum Ilom
- Remote Console im Ilom starten
- Einfügen des ISO-Image als Storage
- Der Server startet beim nächsten Boot-Vorgang vom angehängten Image

Nun zu den Unterschieden gegenüber einem Bare-Metal-Restore der ganzen ODA-Maschine: Sobald das Re-Imaging des Knotens abgeschlossen ist, beginnt die Konfiguration von User, SSH, Netzwerk und Cluster. Wichtig ist zu wissen, dass die Maschine nach dem Neuaufsetzen des Knotens über eine saubere Linux-Installation mit sämtlichen OS-Komponenten inklusive Appliance Manager verfügt. Nur OS-User, Netzwerk-Informationen und Multipath-Konfiguration feh-

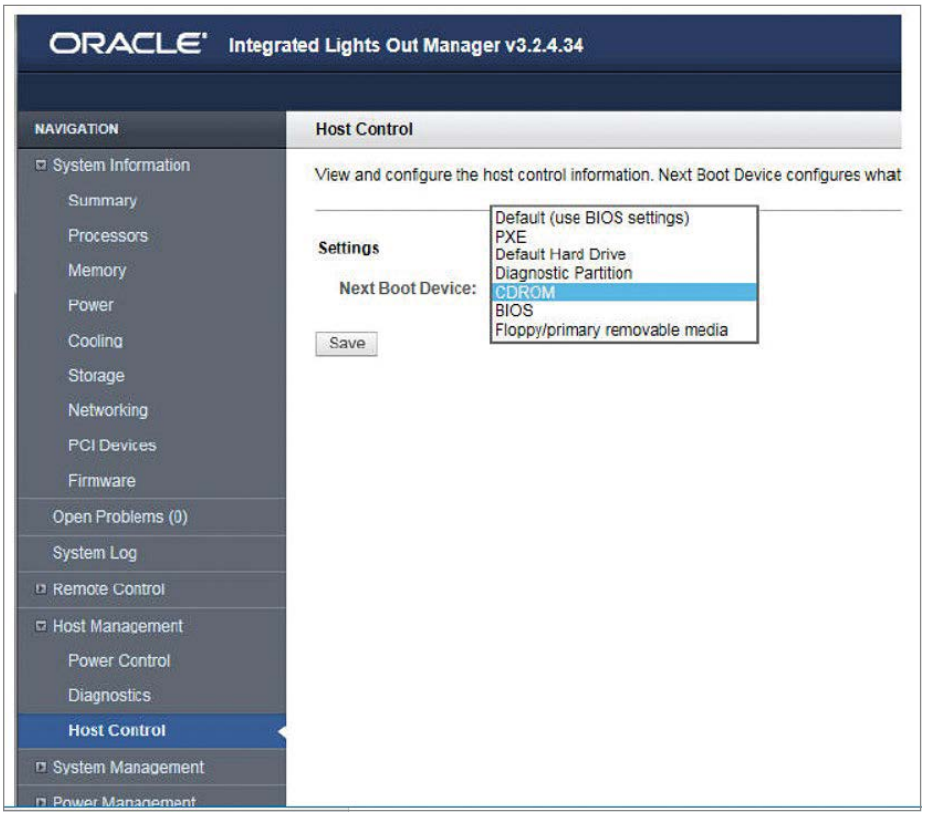



*Abbildung 3: Installation des Knotens (1)*

len noch und müssen erstellt werden, bevor man den Knoten zum Cluster hinzufügt.

## Konfiguration und Hinzufügen des neuaufgesetzten Knotens im Cluster

Sobald die Maschine neu aufgesetzt wurde, ist sicherzustellen, dass sie Netzwerktechnisch zur Verfügung steht, und dafür die Public-IP zu konfigurieren. Dazu dient wie üblich das Kommando „oakcli configure firstnet“. Fast alle Schritte, die vorzunehmen sind, um den Knoten zu konfigurieren, basieren auf einem ODA-Skript, das eine Vielzahl von Optionen bietet *(siehe Abbildung 6)*.

Nicht alle Optionen müssen unbedingt ausgeführt sein, aber im Prinzip sind dies die Optionen, die während des Deployments des App-Managers ausgeführt werden.

Um alle Konfigurationsdateien und Binaries kopieren zu können, muss die SSH-Verbindung zwischen beiden Knoten ohne Passwort funktionieren, beziehungsweise das Root-Passwort sollte auf dem funktionierenden Knoten auf „welcome1“ zurückgesetzt werden. Die Konfiguration von SSH ist als Punkt 3 in der Ausgabe-Liste der „onecommand“-Parameter-Datei dargestellt *(siehe Abbildung 7)*.

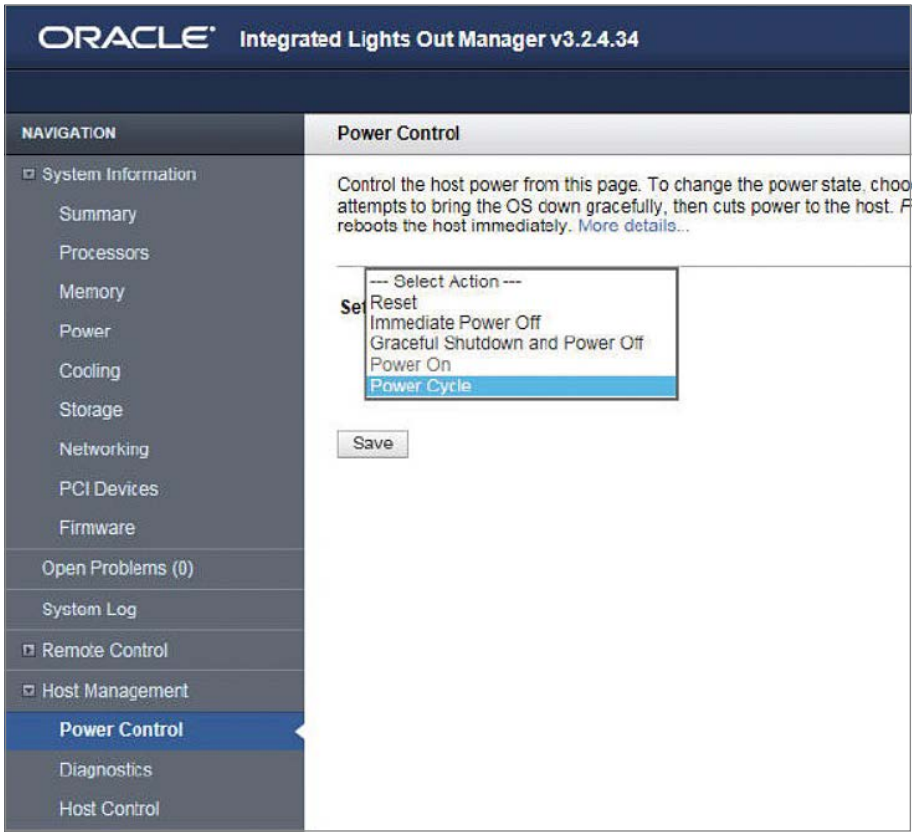



*Abbildung 4: Installation des Knotens (2)*

Natürlich gibt es auch Bereiche, in denen ein manueller Eingriff notwendig wird. Bei jedem ODA-Deployment generiert der App-Manager eine „onecommand“-Parameter-Datei, in der alle relevanten Informationen wie VIP-Adressen, Scans, Hostnamen oder privates Netzwerk über den Cluster gespeichert sind. Diese Datei muss auf den neu aufgesetzten Knoten kopiert und dort angepasst werden.

Ziel ist es, Einträge, die zu dem funktionierenden Knoten gehören, zu löschen. Die Parameterdatei sollte am Ende nur die relevanten Informationen für den neuen Knoten enthalten. Sobald die Datei fertiggestellt ist, beginnt man, das „GridInst.pl“-Skript mit der entsprechenden Schritt-Option auszuführen beziehungsweise bestimmte Schritte „in einem Rutsch“ abzuhandeln („-r“-Parameter, siehe *Listing 6*). Die Schritte kann man auch einzeln ausführen („-s“ Parameter): „./GridInst.pl –s 10“.

Damit sind alle wichtigen Tätigkeiten in Bezug auf Netzwerk und OS-User abgeschlossen. Die ursprüngliche Parameter-Datei kann wiederhergestellt werden; VIPs, Hostnamen etc. für beide Knoten sollten jetzt vorhanden sein.

Da zuvor nur das SSH-Protokoll für den Root-User konfiguriert wurde, ist jetzt die Zeit gekommen, alle anderen User (Grid, Oracle) auch mit SSH zu konfigurieren. Die Prompt-Ausgabe mit Hostnamen wird sich nicht ändern, bis die Maschine rebootet ist. Dieser Schritt ist erforder-

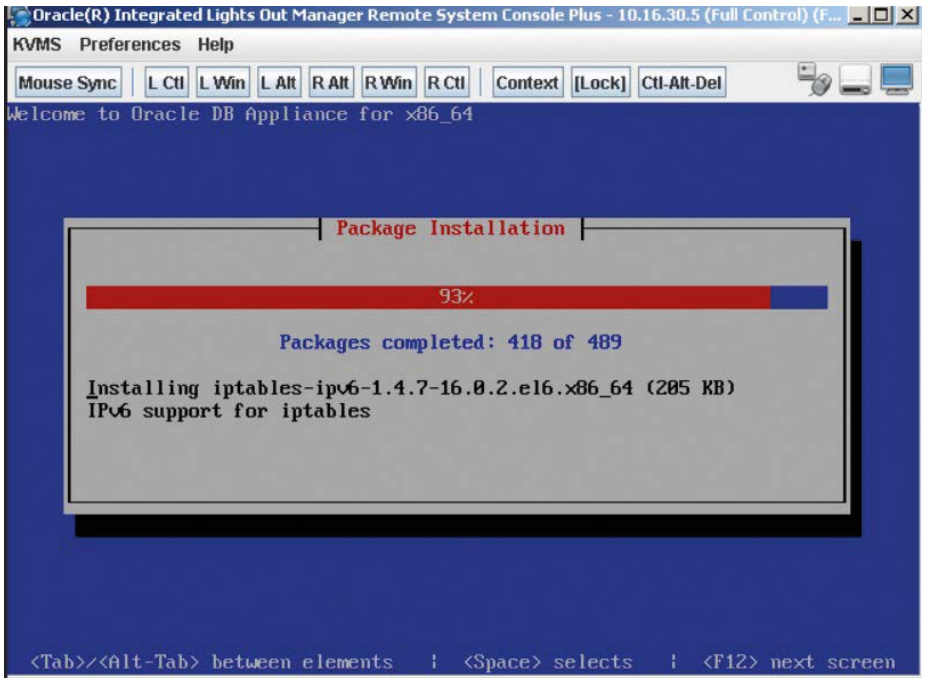



*Abbildung 5: Installation des Knotens (3)*

lich, um die notwendigen Konfigurationsaufgaben weiter durchführen zu können.

An dieser Stelle ist das System Netzwerk-technisch und OS-mäßig vorbereitet. Das einzige, das jetzt noch fehlt, sind die Storage-Anbindung und die Binaries für GI plus RDBMS. Die Multipath-Konfiguration ist sehr einfach und beschränkt sich auf das Kopieren von „multipath.conf" aus der fehlerfreien Maschine. Hier sind nur die WWIDs der lokalen Platten anzupassen und Multipath zu restarten, damit die Änderungen greifen. Wichtig ist dabei, dass auch die User-Rechte für die Disks angepasst werden *(siehe Listing 7)*.

Wenn spezifische Udev-Rules konfiguriert sind, müssen diese ebenfalls übernommen werden. Darüber hinaus sind unbedingt noch Mount Points für ACFS erforderlich. Hier sollte das Kommando „acfsutil info fs -o mountpoints" für die Ausführung auf dem fehlerfreien Knoten aushelfen.

Damit ist bereits die Basis für die GI- und RDBMS-Binaries geschaffen. Es bleiben nur noch zwei Schritte bis zum voll funktionstüchtigen Cluster. Das Hinzufügen des GI-Homeservers läuft fast vollständig automatisiert ab. Zunächst ist man gezwungen, mit „cluvfy comp peer -refnode odanode2 -n odanode1" ein Pre-Check auszuführen. Im Anschluss besteht die Möglichkeit, mit „cluvfy stage -pre nodeadd -n odanode1 –fixup" ein Fix-up-Skript zu generieren, das auf dem neuen Knoten abläuft. Es führt zur Installation des „cvuqdisk-1.0.9-1"-Pakets.

```
[root@ onecmd]# ./GridInst.pl -l
INFO   : Logging all actions in the file /dev/null and traces in the file /dev/null
INFO   : Loading the configuration file /opt/oracle/oak/onecmd/onecommand.params...
The steps in order are...
Step  0 = ValidateParamFile
Step  1 = SetupNetwork
Step  2 = WriteNodelists
Step  3 = SetupSSHroot
Step  4 = VerifyEnvVersion
Step  5 = SetupDNS
Step  6 = UpdateEtcHosts
Step  7 = SetTimezone
Step  8 = SetupNTP
Step  9 = SetupILOM
Step 10 = ValidateEnv
Step 11 = CreateUsers
Step 12 = SetupStorage
Step 13 = SetupSSHusers
Step 14 = InstallGIClone
Step 15 = RunGIClonePl
Step 16 = RunRootScripts
Step 17 = GIConfigAssists
Step 18 = CreateASMDiskgroups
Step 19 = InstallDBClone
Step 20 = RunDBClonePl
Step 21 = DbcaDB
Step 22 = SetupACFS
Step 23 = SetupASR
Step 24 = ResecureMachine
[root@ onecmd]#
```

*Abbildung 6: Die Optionen des ODA-Skripts*

Anschließend wird der GI-Homeserver hinzugefügt *(siehe Listing 8)*.

Am Ende muss noch das „root.sh"-Skript ausgeführt werden. Zudem ist es empfehlungswert, den „oakd"-Dämon durchzustarten, sobald der GI installiert ist. Nun ist die Zeit gekommen, endlich den RDBMS-Homeserver hinzuzufügen. Dieser Schritt ist immer als Oracle-User auszuführen *(siehe Listing 9)*.

Wichtig ist, ebenso viele Homeserver hinzuzufügen, wie es in der ursprünglichen Installation gab. Dabei kommt natürlich auch das „root.sh"-Skript zum Einsatz.

## Fazit

Im Großen und Ganzen ist die Wiederherstellung von nur einem ODA-Knoten dank RAC für den Endbenutzer schnell und transparent. Der Einsatz beschränkt sich nicht nur auf den Bare-Metal-Restore des Knotens und erfordert daher ein gewisses Know-how in Bezug auf Abhängig-

| **Original onecommand.params** | **onecommand.params File for Single node** |
|---|---|
| # DB HOST ARRAY INFO<br>HostArr=(odanode1 odanode1-priv0 odanode1-vip **odanode2 odanode2-priv0 odanode2-vip**)<br>NODE_NAMES=(odanode1 **odanode2**)<br>PRIV_NAMES=(odanode1-priv0 **odanode2-priv0**)<br>HA_NAMES=(odanode1-priv1 **odanode2-priv1**)<br>VIP_NAMES=(odanode1-vip **odanode2-vip**)<br><br># SCAN INFO<br>SCAN_NAME=odanode1-scan<br>SCAN_IPS=(100.24.48.56 100.24.48.57)<br>SCAN_PORT=1521<br>DNS_SERVERS=(192.13.82.180 192.13.82.188 14.20.190.73)<br>NTP_SERVERS=( )<br>NODE_IPS=(100.24.48.12 **100.24.48.13**)<br>VIP_IPS=(100.24.48.28 **100.24.48.29**)<br>PRIV_IPS=( 192.168.16.24 **192.168.16.25**)<br>HA_IPS= (192.168.17.24 **192.168.17.25**)<br><br># NETWORK INFO<br>NET0_IPS=(100.24.48.12 **100.24.48.13**)<br>NET0_NETMASK=255.255.248.0<br>NET0_GATEWAY=100.24.48.1<br>NET0_NAME=bond0<br>NET0_HOSTNAME=(odanode1 **odanode2)** | # DB HOST ARRAY INFO<br>HostArr=(odanode1 slcac451-priv0 odanode1-vip )<br>NODE_NAMES=(odanode1)<br>PRIV_NAMES=(odanode1-priv0 )<br>HA_NAMES=(odanode1-priv1 )<br>VIP_NAMES=(odanode1-vip )<br><br># SCAN INFO<br>SCAN_NAME=odanode1-scan<br>SCAN_IPS=(100.24.48.56 100.24.48.57)<br>SCAN_PORT=1521<br>DNS_SERVERS=(192.13.82.180 192.13.82.188 14.20.190.73)<br>NTP_SERVERS=( )<br>NODE_IPS=(100.24.48.12)<br>VIP_IPS=(100.24.48.28)<br>PRIV_IPS=( 192.168.16.24)<br>HA_IPS= (192.168.17.24)<br><br># NETWORK INFO<br>NET0_IPS=(100.24.48.12 )<br>NET0_NETMASK=255.255.248.0<br>NET0_GATEWAY=100.24.48.1<br>NET0_NAME=bond0<br>NET0_HOSTNAME=(odanode1) |

*Abbildung 7: Ausgabe-Liste der „onecommand"-Parameter-Datei*

```
cd /opt/oracle/oak/onecmd
./GridInst.pl -r 1-11
```

*Listing 6*

```
chown grid:asmadmin /dev/mapper/HDD*p*
chown grid:asmadmin /dev/mapper/SSD*p*
chmod 660 /dev/mapper/HDD*p*
chmod 660 /dev/mapper/SSD*p*
```

*Listing 7*

```
cd $ORACLE_HOME/addnode ./addnode.sh -silent CLUSTER_NEW_
NODES={odanode1} CLUSTER_NEW_VIRTUAL_HOSTNAMES={odanode1-vip}
```

*Listing 8*

```
[oracle@odanode2 ~]$ cd /u01/app/oracle/product/12.1.0.2/dbhome_1/
addnode/ [oracle@odanode2 addnode]$ ./addnode.sh -silent CLUSTER_NEW_
NODES={odanode1}
```

*Listing 9*

keiten und Installationsvorgänge, die auf der ODA automatisiert ablaufen.

Die ODA ist noch nicht reif genug, um mit einem Klick alles wiederherzustellen, der menschliche Faktor spielt daher bei Wartung und Betrieb immer noch eine sehr große Rolle. Ein gesundes Basiswissen zu dem, was während des ODA-Deployments unter der Haube passiert, ist also nicht zu unterschätzen und verschafft uns einen anderen Blickwinkel auf die bunte ODA-Welt mit Tools wie dem App-Manager.

## Quellen


[1] NOTE 1352884.1: „opatch auto“ or „root-crs.pl -unlock“ or „roothas.pl -unlock“ or „addNode.sh“, Hangs Due to Many Audit Files in „GRID_HOME“

[2] NOTE 1526405.1: „addNode.sh“-Fail PRCF-2023, The following contents are not transferred as they are non-readable

[3] NOTE 2027830.1: ODA HA (High Availability) Deployment Step Descriptions, A List of Deployment Version Specific Steps Used for Each ODA HA Version Using GridInst.pl

[4] NOTE 1373599.1: ODA Oracle Database Appliance Bare Metal Restore Procedure X5-2 and X6-2

[5] NOTE 888888.1: Oracle Database Appliance, 18.1, 12.X, and 2.X Supported ODA Versions & Known Issues (Doc ID 888888.1)



Andrzej Rydzanicz
andrzej.rydzanicz@opitz-consulting.com


# Continuous Integration in der Datenbank-Entwicklung


Dominic Weiser, ISTEC Industrielle Software-Technik GmbH



Betrachtet man die aktuellen Arbeitsweisen in der Datenbank-Entwicklung, stellt man sehr schnell fest, dass die agile Vorgehensweise schon ihren Einzug gehalten hat. Ständig wechselnde und sich an die Aufgaben anpassende Entwicklungsteams sind eine Ausprägung dieser Vorgehensweise.


In den seltensten Fällen sind Projekte und Aufgaben so simpel, dass sie von einem Entwickler zu jeder Zeit vollumfänglich überschaut werden können. An dieser Stelle hilft Continuous Integration dabei, negative Nebeneffekte auf fremdem oder eigenem Code frühzeitig aufzuspüren und zu beseitigen. Dabei ist Continuous Integration bestimmt kein Novum mehr, jedoch hat es den flächendeckenden Einzug in die Datenbank-Entwicklung noch nicht wirklich geschafft, obwohl bereits alle technologischen Hilfsmittel in Fülle am Markt verfügbar sind. Dieser Artikel beschreibt die Grundlagen von Continuous Integration und zeigt eine bei-

spielhafte Implementierung für den eigenen Entwicklungs-Workflow.

Wer sich mit Continuous Integration beschäftigt, wird sehr schnell auf unterschiedliche Definitionen und Herangehensweisen treffen. Um das Thema von Grund auf zu betrachten, befasst man sich zunächst mit der Definition nach Martin Fowler, wie er sie im April 2006 mit der aufkommenden Popularität von Extreme-Programming aufgestellt hat: „Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly" *[1]*.

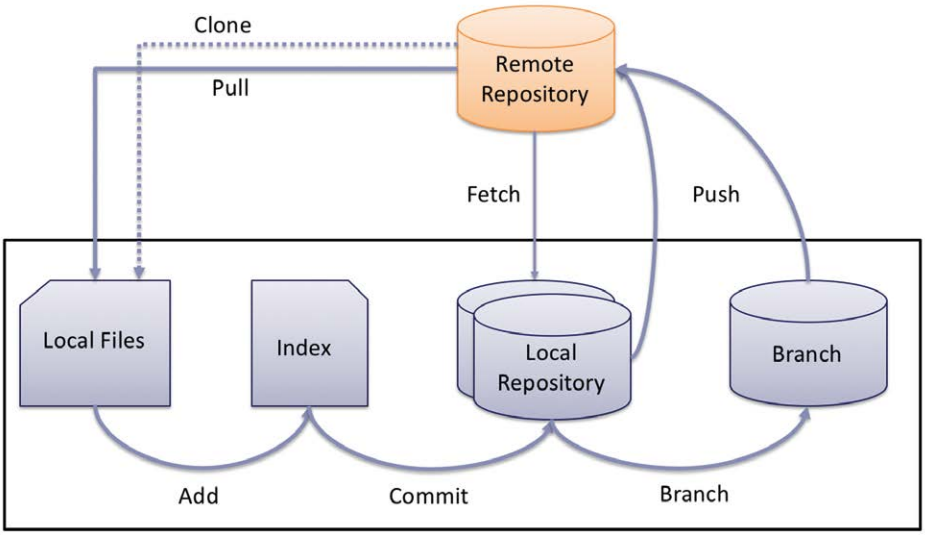



*Abbildung 1: Git-Übersicht*

## Technische Anforderungen

Aus dieser Definition von Continuous Integration lassen sich sehr einfach die genauen technologischen Anforderungen für die Realisierung ableiten:

- *Source-Verwaltungssystem („...each person integrates at least daily... ")*
  Jeder Entwickler hat seinen Entwicklungsstand mindestens täglich in das Verwaltungssystem zurückzuspielen. Zwar könnte hiermit auch das einfache Ablegen auf einem gemeinsamen Verzeichnis gemeint sein, jedoch bieten uns moderne Source-Verwaltungssysteme echte Mehrwerte wie Änderungsverfolgung oder Branching.

- *Build-System („...verified by an automated build... ")*
  Der zurückgespielte Code wird auf einer neutralen Umgebung automatisiert gebaut.

- *Automatisierte Tests („...including test ...")*
  Der zurückgespielte und gebaute Code muss anschließend automatisiert getestet werden.

## Source-Verwaltung: Git

Im Allgemeinen dienen Source-Verwaltungssysteme dazu, Änderungen an Dateien zu erfassen und für den Anwender nachvollziehbar zu machen. Wenn man an dieser Stelle in den Markt hört, gibt es unter Software-Entwicklern nur noch eine Wahl: Git *[2]*. Jeder spricht davon und jeder setzt es ein. Wirklich alle? Nein, denn es gibt noch andere weitverbreitete Source-Verwaltungssysteme wie CVS, Subversion (SVN) oder Mercury. Jedoch für welches System soll man sich entscheiden oder sollte das alte System vielleicht migriert werden?

Um diese Frage zu beantworten, muss man sich die Frage stellen, ob man eine zentrale oder eine verteilte Verwaltung anstrebt. Denn genau darin liegen die gravierenden Unterschiede. Bei CVS oder SVN liegen die zurückgespielten Dateien an einem Ort. Dies kann sehr hilfreich sein, wenn der entwickelte Code in genau einem Projekt oder Produkt zum Einsatz kommt. Alternativ dazu verwendet Git einen verteilten Ansatz und ist darauf ausgelegt, einfache Abspaltungen und Modularisierungen zu erzielen.

So ist es möglich, einzelne Programmteile aus unterschiedlichen Repositories lokal zusammenzuführen und weiterzuentwickeln. Anschließend lassen sich die lokalen Änderungen wieder in die verteilten Repositories zurückspielen. Git ist besonders stark darin, Konflikte in den Code-Versionen aufzuspüren und selbstständig zu beheben. Somit ist der Umgang mit Branches um einiges einfacher und handlicher *(siehe Abbildung 1)*.

Zu Beginn der Arbeit in einem neuen Projektteam muss der bestehende Source-Code heruntergeladen werden. Dazu wird der Befehl „git clone /pfad/zum/repository" ausgeführt. Er führt gleich zwei Aktionen durch; erstens werden die so geklonten Dateien im lokalen Dateisystem abgelegt und zweitens wird ein lokales Repository erstellt.

Das Zurückspielen der Änderungen des lokalen Repository ist die häufigste Fehlerquelle bei Umsteigern von CVS/SVN. Die benötigten Befehle lauten „git add <dateiname>" und „git commit -m „Commit-Nachricht"". Mit „add" werden die Änderungen dem lokalen Index hinzugefügt; ein sehr gutes Mittel, sich Zwischenstände für einen späteren Commit zu merken.

Gerade wenn man an großen Änderungen arbeitet, kann es sonst recht schnell dazu kommen, dass eine Source beim Zurückspielen übersehen wird. Commit schreibt die Änderungen in das lokale Repository. Hier liegt die Fehlerquelle: Dieser Code ist noch nicht für die Kollegen verfügbar. Mit „git push origin master" müssen die Änderungen vom lokalen Repository (origin) erst in das remote Repository (master) übertragen werden. Dies wäre der direkteste Weg, die eigenen Änderungen zu teilen. Ein Blick in die vielen Open-Source-Projekte zeigt, dass Änderungen in der Regel als eigener Branch zurückgespielt werden. So sorgt der Push nur dafür, dass eine Anfrage zum Zurückspielen gestellt wird. Ein Verwalter kann die Änderungen so einfacher nachvollziehen und den Push-Request gegebenenfalls sogar ablehnen.

## Build-System: Jenkins – Maven

Beim Blick ins Java-Umfeld stellen wir fest, dass Build-Systeme seit ca. 2008 ein fes-

ter Bestandteil des Entwicklungsprozesses sind. Der prominenteste Vertreter ist Jenkins *[3]*. Es handelt sich hierbei um einen Fork des Oracle-Produkts Hudson, dessen Weiterentwicklung zugunsten von Jenkins eingestellt wurde. Build-Systeme dienen dazu, den entwickelten Code auf einer neutralen Umgebung zu kompilieren. Somit lässt sich die Integrität des Codes unabhängig von der eigenen Entwicklungsumgebung gewährleisten. Dazu wird die einzelne Kompilierungsausführung in sogenannten Jobs definiert. Ein Job beschreibt immer die Ausführung einer Ausführungsroutine, die sich durch die modulare Struktur von Jenkins durch eine Vielzahl von Plug-ins erweitern lässt. Für jeden Job gibt es unterschiedliche Einstiegspunkte *(siehe Abbildung 2)*.

Die häufigste Wahl fällt dabei auf die Commit- beziehungsweise Zeit-gesteuerten Ausführungen. Die Commit-gesteuerte Ausführung eignet sich, um die Kompilierbarkeit des Codes zu überprüfen; Zeit-gesteuerte Ausführungen werden immer dann verwendet, wenn ganze Testpläne oder Deployments gefahren werden.

Zwar gibt es für Jenkins bereits einige Plug-ins, die in der Lage sind, PL/SQL-Code zu kompilieren, jedoch gibt es da noch weitere Alternativen. Eine Alternative ist Maven *[4] (siehe Abbildung 3)*.

Maven kommt ebenfalls aus dem Java-Umfeld und sollte eigentlich dazu dienen, abhängige Bibliotheken einheitlich in den Code zu laden. Daraus entwickelte sich mit der Zeit ein modulares System, das um etliche Plug-ins erweitert wurde. So gibt es verschiedene Plug-ins, mit denen sich PL/SQL-Code kompilieren lässt. Wer zusätzlich neben dem Datenbank-Code auch ein Frontend oder Middleware mit zu verwalten hat, bekommt mit Maven eine All-in-one-Lösung.

Zum Bauen von SQL mittels Maven eignet sich am besten das native Command Line Interface (CLI) der Datenbank. Daher kommt in *Listing 1* das CLI von Oracle zum Einsatz. Die wichtigsten Informationen sind: der Pfad zum SQL*Plus, der Pfad zur Datenbank und zu den Quelldateien inklusive Build- und Testausführung. Im Beispiel wird nur ein Ausführungsziel („goal“) einer Ausführungsphase des Maven-Builds dargestellt. In der Praxis bietet es sich an, die unterschiedlichen Ausführungen (Builds und Tests) in unterschiedlichen Zielen beziehungsweise Phasen ausführen zu lassen. Für sehr große Projekte kann die Trennung auch über Maven-Module durchgeführt werden.

Zurück zum Jenkins-Job: Dieser hat nun das Kompilieren über Maven und die Testausführung angestoßen. Die Ergebnisse sollen nun genauso automatisiert in den Entwicklungszyklus einfließen wie deren Ausführung. Dazu lässt sich Jenkins an moderne Kollaborationstools wie Slack oder Trello anbinden. Wer es dagegen klassisch haben möchte, lässt die verantwortlichen Entwickler per E-Mail benachrichtigen.

## Automatisierte Tests: utPLSQL

Im Zusammenhang mit Software-Tests gibt es viele unterschiedliche Herangehensweisen. Ein nativer Ansatz wäre, jede Methode einzeln zu testen. Dieser Versuch würde jedoch sehr schnell dazu führen, dass jede geschriebene Methode durch enorm viel mehr an Test-Methoden ergänzt werden muss. Das würde den Entwicklungsaufwand nur unnötig in die Höhe treiben. Aus diesem Grund haben sich die funktionalen Tests (Unit Tests) durchgesetzt. Es müssen also nur noch die Funktionalitäten getestet werden. Dies kann auch dazu führen, dass ein Test sich auf mehrere Schichten (Frontend, Middleware und Backend) auswirkt.

Betrachtet man nun eine Möglichkeit, Unit-Tests auf der Datenbank auszuführen, wird man sehr schnell auf das utPLSQL-Projekt *[5]* aufmerksam. Es wurde ursprünglich durch Steven Feuerstein entwickelt. Seit dem Jahr 2017 wird es durch ein Team auf GitHub als Open-Source-Projekt aktiv weiterentwickelt.

Auch bei Unit-Test gibt es unterschiedliche Ausführungszeitpunkte. Per se setzt sich ein Unit-Test aus mehreren Test-Methoden zusammen. Jede davon lässt sich durch eine „Before“- und eine „After“-Methode ergänzen, die immer davor und danach ausgeführt werden. Diese Ausführungen lassen sich noch weiter verfeinern, wie zum Beispiel vor jeder Methode, vor der Klasse oder vor dem Package. So lässt sich eine benötigte Ausgangssituation vorbereiten beziehungsweise wiederherstellen. Zum Installieren von utPLSQL bietet dieses eine vorgefertigte Installationsdatei (.sql), die zwingend als SYSDBA ausgeführt werden muss: „sqlplus sys/sys_pass@db as sysdba @@install_headless.sql“.

*Listing 2* zeigt, wie mithilfe von utPLSQL ein Testpaket erzeugt und gestartet wird. In der Testprozedur wird die zu testende Funktion „btwnstr“ aufgerufen. Diese soll als Ergebnis den Substring zwischen einem Start- und einem Endpunkt zurückgeben. Ist dies nicht der Fall, schlägt der Test fehl.

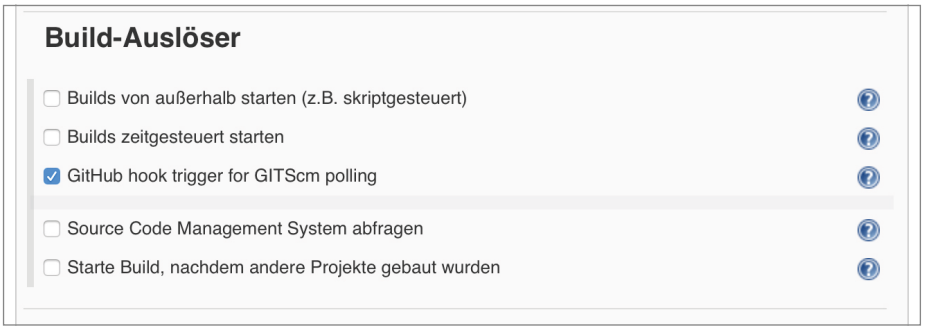



*Abbildung 2: OnCommit-Job*

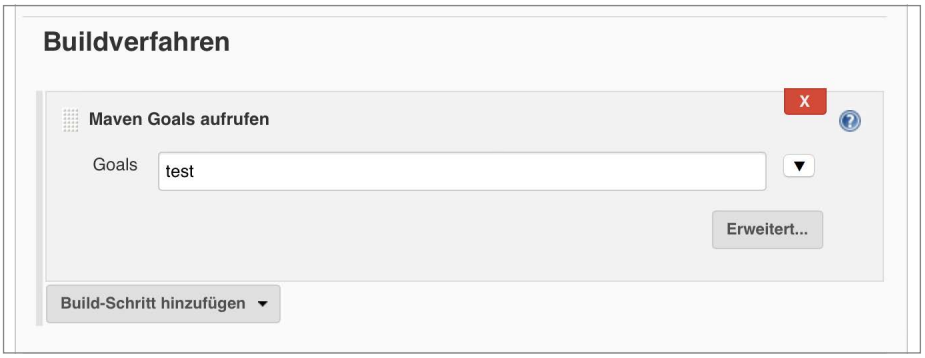



*Abbildung 3: Maven-Ziel*

```
<plugin>
 <groupId>org.codehaus.mojo</groupId>
 <artifactId>exec-maven-plugin</artifactId>
 <version>${maven_exec_plugin_version}</version>
 <executions>
  <execution>
  <phase>run-test</phase>
  <goals>
      <goal>test</goal>
  </goals>
  <configuration>
      <executable>pfadzuOracle/bin/sqlplus</executable>
      <workingDirectory>/pfad/zum/repository</workingDirectory>
      <environmentVariables>
          <ORACLE_HOME>PFADZUORACLE</ORACLE_HOME>
          <PATH>$PATH:$ORACLE_HOME/bin:$JAVA_HOME/bin</PATH>
      </environmentVariables>
      <arguments>
          <argument>user/pwd@db_tns</argument>
          <argument>doTest.sql</argument>
      </arguments>
  </configuration>
  </execution>
 </executions>
</plugin>
```

*Listing 1: Maven-Konfiguration*

```
create or replace package test_betwnstr as
   -- %suite(Between string function)
   -- %test(Returns substring from start position to end position)
   procedure basic_firstTest;
end;
create or replace package body test_betwnstr as
   procedure basic_firstTest is
   begin
       ut.expect( betwnstr( '1234567', 2, 5 ) ).to_equal('2345');
   end;
end;
begin
       ut.run('test_betwnstr');
end;
```

*Listing 2: „doTest.sql“*

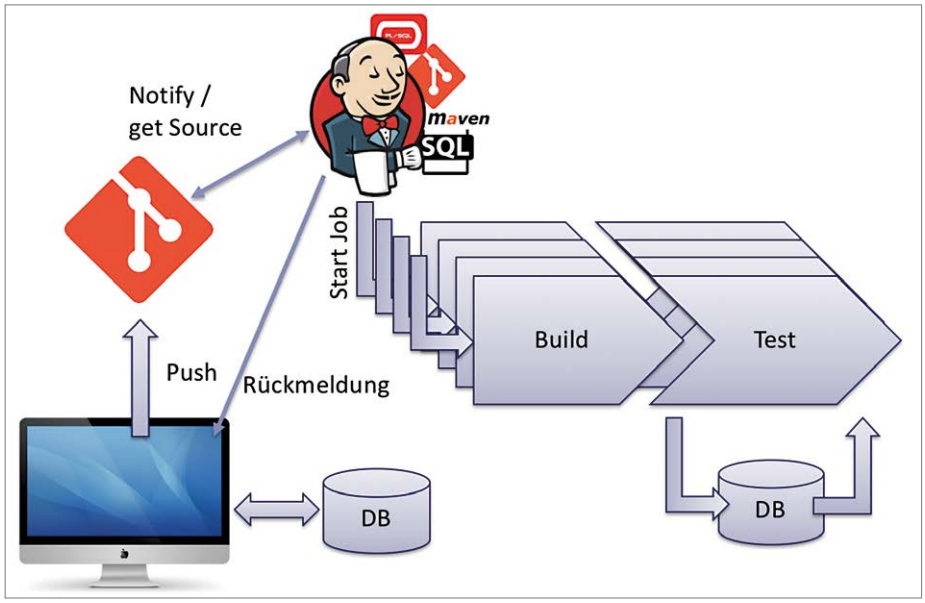



*Abbildung 4: CI-Infrastruktur*

## Fazit

Werden die unterschiedlichen Technologien so wie in diesem Artikel beschrieben aufgesetzt, erhält man die in *Abbildung 4* gezeigte Infrastruktur. Der Entwickler behält seine lokale Entwicklungsumgebung und sollte mit dieser autark arbeiten. Nachdem die Änderungen umgesetzt wurden, werden diese mit Git in das globale Repository zurückgespielt und mithilfe von Jenkins automatisiert gebaut und getestet. Der Entwickler erhält zeitnah Informationen über seine Änderungen und kann gegebenenfalls verbessernd eingreifen.

Die im Artikel dargestellte Vorgehensweise zeigt eine stark vereinfachte Anwendung von Continuous Integration. Durch den modularen Aufbau der Technologien und durch die möglichen Plugins lässt sich diese Umgebung einfach an die eigenen Bedürfnisse anpassen.

Für das Mehr an Wartungs- und Entwicklungsaufwand erhält man eine Reduzierung der Rückmeldezeiten und weniger unvorhergesehene Fehler bei der produktiven Installation. Der Autor ist überzeugt, dass das Thema „Continuous Integration“ mit seinen Ausprägungen „Continuous Deployment“ und „Continuous Delivery“ in Zukunft eine immer größere Verbreitung in der Datenbank-Entwicklung finden wird.

## Weitere Informationen


[1] *https://www.martinfowler.com/articles/continuousIntegration.html*
[2] *https://git-scm.com*
[3] *https://jenkins.io*
[4] *https://maven.apache.org*
[5] *http://utplsql.org*



Dominic Weiser
dominic.weiser@istec.de


# DevOps und SaaS mit Datenbank-Application-Containern

Dr.-Ing. Holger Friedrich, sumIT AG

Management, Entwicklung und Betrieb von Datenmodellen und datenbankbasierten Applikationen haben heute zwei Herausforderungen zu bewältigen: zum einen die Bewirtschaftung vieler Instanzen, sei es im Rahmen des Entwicklungs- und Testprozesses oder des Betriebs von Multi-Mandantensystemen (SaaS-Anwendungen), und zum anderen besteht die Notwendigkeit, agil und mit immer kürzeren Release-Zyklen hin zu Continuous Delivery zu kommen. Multi-Tenant-Applikations-Container sind ein neues Feature der Oracle-Datenbank, das bei der Bewältigung beider genannten Herausforderungen unterstützt.

Virtualisierung ist mittlerweile fast schon ein alter Hut. Zu Beginn waren es virtuelle Maschinen, die die Provisionierung einzelner Umgebungen und Systeme vereinfachten und die sichere, effiziente Nutzung geteilter Ressourcen ermöglichten. Mittlerweile zieht sich die Virtualisierung durch alle Aspekte eines Rechenzentrums. Virtualisiertes Netzwerk, virtueller Storage – der Entwicklung scheinen keine Grenzen gesetzt.

Während der letzten Jahre gab es allerdings Entwicklungen in der IT-Industrie, bei denen die Virtualisierung von Servern an ihre Grenzen stößt, ja schon zu ineffizient, zu altbacken geworden ist. Insbesondere sind dies die agile Software-Entwicklung sowie Software-as-a-Service/Cloud.

Agile Software-Entwicklung mit all ihren Bestandteilen, den kurzen Entwicklungs-Sprints, dem hohen Grad der Test-Automatisierung und schließlich dem Ziel des kontinuierlichen Deployments, erfordert den schnellen Auf- und Abbau zahlreicher Applikations-Entwicklungs- und Test-Umgebungen. Software-as-a-Service wiederum verlangt nach der schnellen Provisionierung von Umgebungen für neue Mandanten. In beiden Anwendungsfällen sollten Bereitstellung, Betrieb, aber auch Dekommissionierung möglichst hoch automatisierbar sein. Selbstbedienung ist erwünscht. Die Cloud lässt grüßen.

Gegenüber diesen hohen Ansprüchen hat die klassische Server-Virtualisierung mit virtuellen Maschinen deutliche Nachteile. Diese beruhen vor allem darauf, dass sie in jeder Instanz eine vollständige Betriebssystem-Installation beherbergen. Das macht VMs wartungsintensiv, groß und sehr Ressourcen-hungrig.

Zur Lösung dieses Problems wurde die Container-Technologie entwickelt. Hier beherbergt die virtuelle Maschine, also der Container, nur die Komponenten der Kunden-Applikation sowie einige Schnittstellen-Binaries und -Libraries. Diese kommunizieren mit der Server-Plattform, die eine Betriebssystem-Installation sowie die Container-Engine für alle gehosteten Container enthält. Die Applikationen kommunizieren über die Schnittstellen-Libraries in den

Containern mit der Container-Engine und dadurch mit der Betriebssystem-Installation. Aus dieser Architektur ergibt sich eine Reihe von Vorteilen von Containern gegenüber klassischen VMs *(siehe Tabelle 1)*.

Mittlerweile hat sich die Welt weitergedreht; selbst Container erscheinen bereits zu groß und zu unflexibel. Serverless-Architekturen sind der neueste Trend. Dabei ruft die eigene Applikation, natürlich als eine orchestrierte Menge von Micro-Services, nur noch APIs auf. Der Code lebt und läuft irgendwo in der Server-Landschaft eines Cloud-Providers, ohne dass Entwickler und Applikations-Owner sich um Provisionierung und Pflege kümmern müssen.

Alle diese Entwicklungen sind spannend und sehr gut für Betrieb und Entwicklung von Applikationen im Middle-Tier einsetzbar. Wie steht es jedoch mit dem Back-End von Applikationen, typischerweise relationalen Datenbank-Systemen? Auch diese müssen ihre Rolle in der agilen Software-Entwicklung und in Saas-Architekturen spielen und ebenfalls automatisiert provisioniert und betrieben werden können.

Ist es eine gute Idee, relationale Datenbank-Managementsysteme wie eine Oracle-Datenbank ebenfalls im Container zu installieren? Außer für sehr kleine Installationen wahrscheinlich eher nicht. Was dagegen spricht, ist die Tatsache, dass ein relationales Datenbank-Managementsystem nicht nur Applikationslogik enthält, sondern auch die Daten persistiert. Diese sind bei ernsthaften Anwendungen wie ERP-Systemen oder gar Data Warehouses umfangreich und wachsen mit der Zeit. Container-Technologie hingegen ist, wie gezeigt, für flexible, schlanke Installationen gedacht. Man kann natürlich die Datenbank-Software im Container installieren, die Server-Prozesse darin laufen lassen und den persistenten Storage, also die Datenbank-Dateien, außerhalb des Containers verwalten. Dies schafft allerdings Engpässe beim Zugriff und Abhängigkeiten, die das einfache Umherschieben und Provisionieren verhindern.

So lässt sich also das Fazit ziehen, dass Virtualisierung und Container-Technologie für Applikations-Entwicklung und -Betrieb hervorragend geeignet sind, sich für Datenbank-Installationen aber nur begrenzt eignen. Andererseits werden mehr Flexibilität, höherer Automatisierungsgrad und mehr Agilität auch im Datenbank-Umfeld benötigt.

| Virtuelle Maschinen | Container |
|---|---|
| Benötigt Hypervisor und eine vollständige OS-Installation in jeder VM | Spricht mit dem Host-Kernel (keine OS-Installation im Container) |
| Größerer Platzbedarf (RAM und nicht-flüchtiger Speicher) | Kleiner Footprint |
| In der Regel größer als 1 GB | Oftmals nur wenige MB |
| Startup in Minuten | Startup in Sekunden |
| komplexes Deployment (Netzwerk-Bandbreite etc.) | Einfaches Deployment |
| Langsamer | Schneller |
| Sicherheitsprobleme mit installiertem OS | Sicherheitsprobleme ausschließlich mit eigener Applikation |

*Tabelle 1: Klassische Virtualisierung und App-Container im Vergleich*

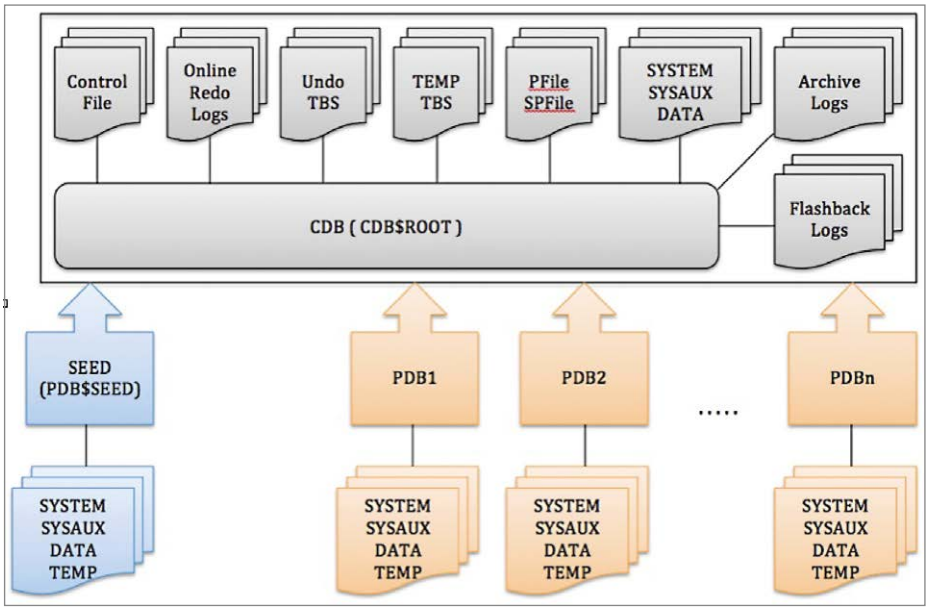



*Abbildung 1: Architektur der Oracle-Container- und Pluggable-Datenbanken*

## Oracle-Datenbank-Applikations-Container

Oracle hat den Bedarf erkannt und veröffentlichte mit der Datenbank-Version 12c eine Technologie, die Container für Datenbanken und Datenbank-Applikationen bereitstellt. In Release 12.1 war es die Container-Technologie für Datenbanken in Form von Container- und Pluggable-Datenbanken, in Release 12.2 kamen Datenbank-Applikations-Container hinzu. Die Architektur von Container- und Pluggable-Datenbanken beruht auf der Trennung allgemeiner Datenbank-Managementprozesse, -Metadaten und -Daten von applikationsspezifischen Datenstrukturen und Logik *(siehe Abbildung 1)*.

Die Container-Datenbank ist in Grau dargestellt. Sie stellt für alle angeflanschten Pluggable-Datenbanken die zentralen Management-Prozesse sowie dafür benötigte Strukturen und Daten zur Verfügung. Blau ist die Seed-Datenbank, aus der einfach neue Datenbanken geklont werden können. In orangener Farbe sind die Nutzer-Datenbanken angezeigt. Die Architektur erlaubt das Management vieler Datenbanken in einem Container, der CDB. Typische Wartungsprozesse, wie Backup und Recovery, Upgrades, Patching, aber auch die Provisionierung neuer Datenbanken und die Migration von Datenbanken auf andere Umgebungen, sind so stark vereinfacht. Immer mehr dieser Operationen sind auch online, also ohne Unterbrechung des Applikationsbetriebs, möglich.

Diese Architektur war für Tätigkeiten auf Datenbank-Administrationsebene bereits hervorragend. Bis Release 12.2 fehlte jedoch die Möglichkeit, Datenbank-Applikations-Instanzen, ihre Strukturen, Pro-

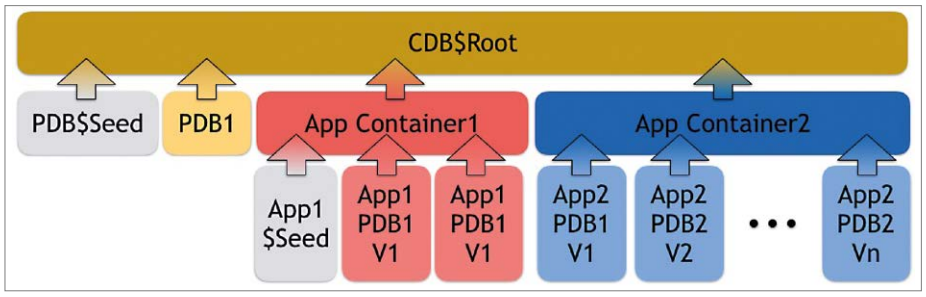



*Abbildung 2: Architektur der Datenbank-Applikations-Container*

gramm-Pakete und -Logik ähnlich komfortabel zu verwalten. Dafür hat Oracle das Konzept der Container-Datenbank um eine Ebene erweitert. Neben der Container-Datenbank und darunter angeflanschter Pluggable-Datenbanken gibt es nun auch Applikations-Container- und Applikations-Datenbanken *(siehe Abbildung 2)*.

Unter der Container-Datenbank (CDB) können außer der Seed-Datenbank und beliebiger Pluggable-Datenbanken (PDBs) nun auch Applikations-Container-Datenbanken eingehängt werden. App-Container sind nichts Weiteres als reguläre PDBs, die bei der Erstellung als App-Container gekennzeichnet und initialisiert werden. Mit dieser Initialisierung erhält die PDB die Funktionalität, Applikationen zu verwalten und sie in angeschlossenen Applikations-PDBs zu bewirtschaften.

In die App-Container können nun App-PDBs eingesteckt werden, analog zu den regulären PDBs, die direkt am „CDB$Root" hängen. App-PDBs können dabei aus einer App-Seed-PDB geklont, neu erstellt oder auch von beliebigen anderen PDBs geklont werden.

Über einen App-Container können Applikationen zentral installiert, upgegradet, gepatcht und deinstalliert werden. Wann Upgrades in angehängten App-PDBs wirksam werden, ist frei wählbar. Darüber hinaus kann der App-Container zentral Daten für alle angehängten App-PDBs verwalten. Auf diese Weise sind Daten allgemeinen Interesses schreibgeschützt allen provisionierten App-PDBs zugänglich.

Zwei Voraussetzungen müssen erfüllt sein, um DB-App-Container nutzen zu können. Die eine ist lizenzrechtlicher Natur, die andere betrifft die Konfiguration der Container-Root-Datenbank:

- Zur Nutzung des DB-App-Container-Features sind eine Enterprise-Edition-Datenbank-Lizenz und zusätzlich die Lizenzierung der Multi-Tenant-Option erforderlich.
- Der Datenbank-Root-Container, in dem DB-App-Container und App-DBs betrieben werden sollen, muss zur Anwendung von Oracle Managed Files (OMF) konfiguriert sein. Dies ist notwendig, da das DBMS im Verlaufe der Nutzung neue Datenbank-Files und PDBs erzeugen und dafür die Namen selbstständig generieren und verwalten können muss. OMF muss spätestens vor der Erstellung des ersten DB-App-Containers eingeschaltet sein.

## Einen Datenbank-Applikations-Container erstellen

Die Erstellung eines DB-App-Containers ist denkbar einfach. Sie besteht aus dem Einrichten einer PDB, erweitert um die Klausel „AS APPLICATION CONTAINER". Dafür müssen der Anwender die SYSDBA-Berechtigungen haben und der Kontext der Session auf dem „CDB$Root"-Container stehen. *Listing 1* löst die Erstellung aus.

Der Applikations-Container ist ausschließlich zur Verwaltung von Applikationen bestimmt. Um eine Applikation zu nutzen, ist mindestens eine Applikations-PDB erforderlich. Diese kann als Klon einer bestehenden PDB, als Klon einer „APP$Seed"-PDB oder als neue App-PDB erzeugt werden. Ein User mit SYSDBA-Rechten oder der Admin-User des Applikations-Containers löst die Erstellung der App-PDB aus. Dabei muss der Session-Kontext zunächst auf den DB-App-Container gesetzt werden *(siehe Listing 2)*. Nachdem die App-PDB erstellt und geöffnet wurde, können bereits im App-Container installierte Applikationen in ihrer aktuellen Version in die App-PDB synchronisiert werden *(siehe Listing 3)*.

Im Zuge der Synchronisation werden in der PDB Applikations-User, Datenbankdateien, Datenstrukturen und auch Inhalte angelegt beziehungsweise mit dem App-Container verlinkt. Dabei gibt es verschieden Möglichkeiten, um Strukturen und Inhalte zu organisieren und zwischen App-Container und App-PDBs zu (ver-)teilen.

## (Meta-)Daten teilen oder lokal halten

Sowohl Metadaten, also Struktur- und Logik-Definitionen, als auch Nutzdaten können zwischen App-Containern und ihren App-PDBs geteilt oder lokal in den App-PDBs gehalten werden. Dafür gibt es die neue Klau-

```
-- erstellen und öffnen des Application Containers
CREATE PLUGGABLE DATABASE appcon1 AS APPLICATION CONTAINER
   ADMIN USER appcon_admin IDENTIFIED BY ac1manager;
ALTER PLUGGABLE DATABASE appcon1 OPEN;
```

*Listing 1*

```
ALTER SESSION SET container = appcon1;
-- Erstelle Application PDB
CREATE PLUGGABLE DATABASE app1pdb1
ADMIN USER pdb_admin IDENTIFIED BY a1p1manager;
ALTER PLUGGABLE DATABASE app1pdb1 OPEN;
```

*Listing 2*

```
-- initialisere neue Application PDB
ALTER SESSION SET container = app1pdb1;
ALTER PLUGGABLE DATABASE APPLICATION ALL SYNC;
```

*Listing 3*

```
ALTER PLUGGABLE DATABASE APPLICATION
{ { app_name
    { BEGIN INSTALL 'app_version' [ COMMENT 'comment' ]
    | END INSTALL [ 'app_version' ]
    | BEGIN PATCH number [ MINIMUM VERSION 'app_version' ] [ COMMENT 'comment' ]
    | END PATCH [ number ]
    | BEGIN UPGRADE 'start_app_version' TO 'end_app_version' [ COMMENT 'comment' ]
    | END UPGRADE [ TO 'end_app_version' ]
    | BEGIN UNINSTALL
    | END UNINSTALL
    | SET PATCH number
    | SET VERSION 'app_version'
    | SET COMPATIBILITY VERSION { 'app_version' | CURRENT }
    | SYNC  }
  |
  { ALL SYNC }
}
```

*Abbildung 3: API zur Applikations-Erstellung und -Verwaltung*

sel „SHARING" für „Create DDL"-Statements. Damit wird festgelegt, wo Objekte und Daten lokalisiert sein sollen und welche Art des Zugriffs gestattet wird. Es gibt vier Arten der Definition und Berechtigungen.

- *Metadata*
  Dies ist die Default-Option. „Metadata-gesharte Objekte" bedeutet, dass die Definition im App-Container liegt und alle App-PDBs diese nutzen können, ohne sie allerdings verändern zu können. Die Definition einer Metadata-gesharten Tabelle ist in den App-PDBs unveränderlich. Jede App-PDB kann jedoch ihre eigenen Daten lokal darin verwalten.

- *Data*
  Sind Strukturen DATA-geshart, ist nicht nur die Definition im App-Container hinterlegt, sondern auch der Daten-Inhalt. App-PDBs können die Tabelle lesen, ihren Inhalt durch Constraints von lokalen Daten referenzieren, aber weder Struktur noch Inhalt verändern. Diese Option ist insbesondere für Stammdaten geeignet, die zentral gewartet werden sollten, etwa ein Postleitzahlen-Verzeichnis.

- *Extended Data*
  Diese Sharing-Option entspricht der „Data"-Option, mit der Erweiterung, dass App-PDBs eigene Daten in Form von Zeilen in die Struktur einbringen können. Die eigenen Daten sind nur innerhalb der betreffenden PDB sichtbar.

- *None*
  Die letzte Option ist für Strukturen gedacht, die lokal in jeder App-PDB erstellt werden sollen und von dieser danach in vollem Umfang verändert werden können.

## Management-Views

Zur Kontrolle des Installationszustands, der Version und des Patch-Levels von Applikationen stehen im App-Container und den App-PDBs verschiedene Data-Dictionary-Views zur Verfügung. Insbesondere sinnvoll sind diese:

- *dba_applications*
  Gibt Auskunft über installierte Applikationen und deren Version beziehungsweise Installationsstatus

- *dba_app_patches*
  Enthält Informationen über installierte Applikationspatches

Release 12.2 ist das initiale Release der DB-App-Container-Funktionalität. Das API zur Erstellung und Pflege von Applikationen ist sehr übersichtlich und einfach zu nutzen. *Abbildung 3* zeigt das vollständige API mit Stand 12.2.

Applikationen werden zunächst im App-Container installiert. Dabei wird im Kontext des App-Containers die Installation initiiert und ein Name vergeben. Anschließend können alle Operationen zur Erstellung der Applikation ausgeführt werden. Dies beinhaltet die Erstellung von Datenbank-Benutzern, Tablespaces, Strukturen, Programmpaketen etc. *Listing 4* zeigt die beispielhafte Installation einer Applikation. Danach kann die neue Applikation dann in App-PDBs synchronisiert werden *(siehe Listing 5)*.

## Upgrade von Applikationen

Applikations-Upgrades verlaufen prinzipiell gleich wie die Installation. Dazu wird das Upgrade im App-Container initialisiert. Dann kommen alle Upgrade-Skrip-

```
-- Beginn der Installation initiieren
ALTER PLUGGABLE DATABASE APPLICATION demoapp BEGIN INSTALL '1.0';
-- Ausführen aller Installtionsskripte
CREATE TABLE demoapp.bundesland SHARING=DATA (
   id             NUMBER,
   name           VARCHAR2(50),
   einwohnerzahl  NUMBER(22)
   CONSTRAINT bundesland_pk PRIMARY KEY (id));
ALTER TABLE demoapp.bundesland ADD CONSTRAINT bundesland_pk PRIMARY KEY
(id);
INSERT INTO demoapp.bundesland (id, name, einwohnerzahl)
   values (1, ,'Baden-Württemberg', 10879618);
INSERT INTO demoapp.bundesland (id, name, einwohnerzahl)
   values (2, 'Bayern', 12843514);
COMMIT;
-- Ende der Installation bestimmen
ALTER PLUGGABLE DATABASE APPLICATION demoapp END INSTALL '1.0';
```

*Listing 4*

te für die Applikation zur Ausführung, die die Entwickler bereitgestellt haben. Dabei können auch Strukturen, Benutzer etc. gelöscht werden. Zu guter Letzt wird wieder in die App-PDBs synchronisiert, wann immer es dafür Bedarf und ein Wartungsfenster gibt. Wie verhindert nun das Datenbank-Managementsystem, dass Änderungen im App-Container nicht sofort auf die App-PDBs durchschlagen?

Um dies zu bewerkstelligen, erzeugt das System im Hintergrund vollautomatisch und transparent einen Klon des App-Containers. Dieser enthält die bisherige Applikationsversion. Dann werden alle App-PDBs an diesen Schatten-Container umgehängt. All dies geschieht bei der Initiierung des Upgrades. Zum Ende des Upgrades ist der primäre App-Container im neuen Zustand, aber ohne eingehängte PDBs. Diese werden während des Synchronisierens upgegradet und vom Schatten-Container zum aktuellen Container umgehängt. *Abbildung 4* zeigt die drei Phasen des Upgrades.

```
ALTER SESSION SET container = app1pdb1;
ALTER PLUGGABLE DATABASE APPLICATION demoapp SYNC;
```

*Listing 5*

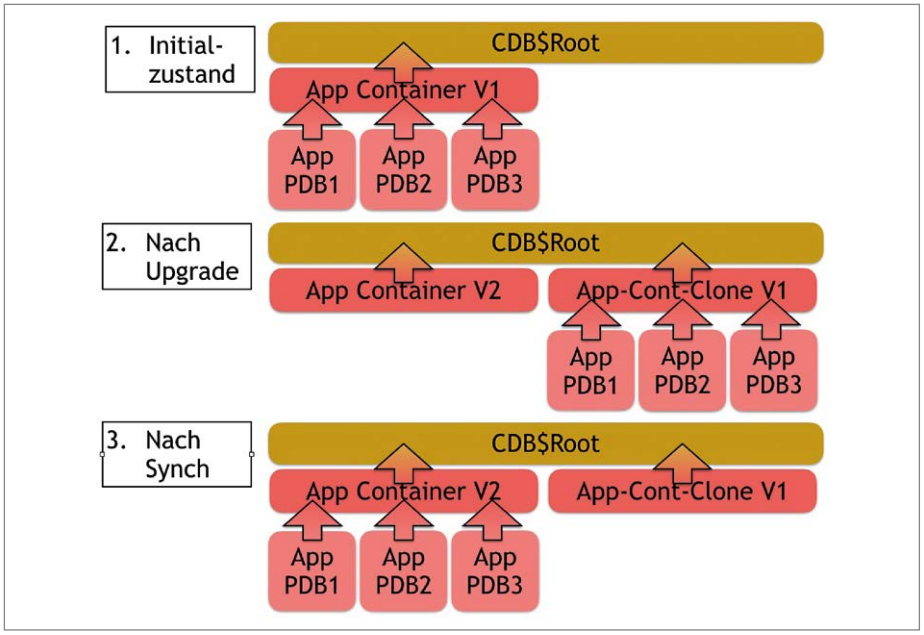



*Abbildung 4: Evolution der App-Container und -PDBs beim Upgrade*

## Patching

Der Vorgang beim Patching gleicht für den Applikations-Verantwortlichen dem des Upgrades. Im Hintergrund geschieht allerdings etwas anderes, weswegen es Einschränkungen gibt. Beim Patching wird eine Patchnummer statt einer Versionsnummer vergeben. Wichtiger ist jedoch, dass kein Schatten-Container erzeugt wird und die App-PDBs nicht umgehängt werden. Daraus resultieren Einschränkungen bei den möglichen Operationen innerhalb des Patching. Destruktive Operationen wie das Löschen von Spalten, Tabellen oder Usern sind nicht gestattet und führen zu Fehlermeldungen. Statt innerhalb eines Patches müssen sie in ein Upgrade verpackt und mit diesem Mechanismus ausgebracht werden.

Das Entfernen von Applikationen erfolgt nicht selbstständig durch die Datenbank. Auch hierfür muss der Applikations-Entwickler Skripte bereitstellen. Wie bei einer Installation werden die Deinstallations-Skripte zunächst im App-Container ausgeführt. In den App-PDBs ist die Applikation dann allerdings noch vorhanden. Hinter den Kulissen wird eine Deinstallation also wie ein Upgrade behandelt. Ein Schatten-Container entsteht zu Beginn, die App-PDBs werden an diesen gehängt und die Deinstallation innerhalb der App-PDBs erfolgt durch Aufruf der Synchronisation *(siehe Abbildung 4)*.

## Fazit

Datenbank-Applikations-Container können für das persistente Backend das sein, was Container-Technologie für den Middle-Tier Layer bereits sind, nämlich eine Methode, um schnell, ressourcensparend und kostengünstig viele Applikations-Instanzen zu erzeugen, zu verwalten und zu pflegen. Zusätzlich zu den beschriebenen Eigenschaften können auch geografisch verteilte Setups aufgebaut werden. Bei diesen sind mehrere DB-App-Container durch DB-Links miteinander verbunden. Mit solch einem Setup lassen sich alle angehängten App-PDBs mit einem Master-DB-App-Container synchronisieren. Dies ermöglicht einfaches Applikations-Management über Rechenzentrums- und Ländergrenzen hinweg.

Die Nutzung all dieser Eigenschaften ist im Rahmen agiler Software-Entwicklung, zur Unterstützung automatischen Testens und von Continuous Delivery ein großer Vorteil. In Software-as-a-Service-Anwendungs-Szenarien, bei denen neue Instanzen von Applikationen für Mandanten automatisch erzeugt und provisioniert werden müssen, können Datenbank-Applikations-Container ebenfalls eine Schlüsselrolle spielen.

Der Funktionsumfang in Release 12.2 ist bereits gut, um die Funktionalität gewinnbringend zu nutzen. Mit Spannung sind die weitere Entwicklung und zukünftige Erweiterungen in Release 18c zu erwarten.

Dr.-Ing. Holger Friedrich
holger.friedrich@sumit.ch

# Wir begrüßen unsere neuen Mitglieder

## Persönliche Mitglieder

- Michael Wiedemayer
- Vitali Fichtner
- Thorsten Becker
- Stephanie E. Vogelfaenger
- Oliver Löffler
- Lukas Horwath
- Haymo Jupé
- Alex Müller
- Jürgen Kamps

## Firmenmitglieder DOAG

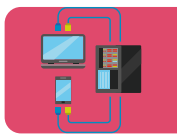
- Litreca AG, Stephanie E. Kliner
- Stiftung kirchl. Rechenzentrum Südwestdeutschland, Thorsten Ries
- Keycon Informations GmbH, David Brandt
- SUMIDA Components & Modules GmbH, Dieter Angerer

# Termine


*25.02.2019*
**Regionaltreffen Thüringen**
Jörg Hildebrandt
Jena

*27.02.2019*
**Primavera Community Day**
Sebastian Hunke & Sebastian Schweinle
Stuttgart

*28.02.2019*
**Regionaltreffen Würzburg**
Oliver Pyka




*08.03.2019*
**DOAG Datenbank Webinar: ORACLE und Docker**
online

*11.03.2019*
**Regionaltreffen München/Südbayern**
Andreas Ströbel

*11.03.2019*
**Regionaltreffen Osnabrück/Bielefeld/Münster**
Andreas Kother & Klaus Günther

*13.03.2019*
**Regionaltreffen Berlin/Brandenburg**
Michael Keemers & Mylène Diacquenod
Berlin/Brandenburg

*18.03.2019*
**NextGen-Programm zur Javaland 2019**
Brühl

*19.- 21.03.2019*
**Javaland 2019**
Brühl

*21.03.2019*
**Regionaltreffen Nürnberg**
Martin Klier & Thomas Köppel
Nürnberg

*26.03.2019*
**NextGen-Programm zur Data Analytics 2019**
Brühl

*26.03.2019*
**Data Analytics 2019**
Brühl


*08.04.2019*
**Regionaltreffen München/Südbayern**
Andreas Ströbel

*11.04.2019*
**Regionaltreffen Rhein-Neckar**
Frank Stöcker
Mannheim

*12.04.2019*
**DOAG Datenbank Webinar: National Language Support und Zeichensätze - Was gibt es zu beachten**
online

*18.04.2019*
**Regionaltreffen Nürnberg**
Martin Klier & Thomas Köppel
Nürnberg

## Impressum

Red Stack Magazin wird gemeinsam herausgegeben von den Oracle-Anwendergruppen DOAG Deutsche ORACLE-Anwendergruppe e.V. (Deutschland, Tempelhofer Weg 64, 12347 Berlin, www.doag.org), AOUG Austrian Oracle User Group (Österreich, Lassallestraße 7a, 1020 Wien, www.aoug.at) und SOUG Swiss Oracle User Group (Schweiz, Dornacherstraße 192, 4053 Basel, www.soug.ch).

Red Stack Magazin ist das User-Magazin rund um die Produkte der Oracle Corp., USA, im Raum Deutschland, Österreich und Schweiz. Es ist unabhängig von Oracle und vertritt weder direkt noch indirekt deren wirtschaftliche Interessen. Vielmehr vertritt es die Interessen der Anwender an den Themen rund um die Oracle-Produkte, fördert den Wissensaustausch zwischen den Lesern und informiert über neue Produkte und Technologien.

Red Stack Magazin wird verlegt von der DOAG Dienstleistungen GmbH,Tempelhofer Weg 64, 12347 Berlin, Deutschland, gesetzlich vertreten durch den Geschäftsführer Fried Saacke, deren Unternehmensgegenstand Vereinsmanagement, Veranstaltungsorganisation und Publishing ist.

Die DOAG Deutsche ORACLE-Anwendergruppe e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Stefan Kinnen. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führen einen konstruktiv-kritischen Dialog mit Oracle.

### Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
Chefredakteur (ViSdP): Wolfgang Taschner
Kontakt: redaktion@doag.org
Weitere Redakteure (in alphabetischer Reihenfolge): Lisa Damerow, Mylène Diacquenod, Marina Fischer, Klaus-Michael Hatzinger, Sanela Lukavica, Martin Meyer, Yann Neuhaus, Fried Saacke

### Titel, Gestaltung und Satz:
Alexander Kermas,
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

### Fotonachweis:
Titel: © Designed by vectorpouch/Freepik
S. 12: © rawpixel/123RF
S. 29: © Dmitry Kalinovsky/123RF
S. 36: © Dmitriy Shpilko/123RF
S. 42: © John Taka/123RF
S. 44: © inkdrop/123RF
S. 48: © alphaspirit/123RF
S. 53: © gstockstudio/123RF
S. 59: © photka/123RF
S. 68: © Binkski/fotolia.com
S. 73: Background: © Dzianis Kuryanovich/123RF

### Anzeigen:
Simone Fischer,
DOAG Dienstleistungen GmbH
(verantwortlich, Anschrift s.o.)
Kontakt: anzeigen@doag.org
Mediadaten und Preise unter:
*www.doag.org/go/mediadaten*

### Druck:
adame Advertising and Media GmbH,
*www.adame.de*



## Inserentenverzeichnis