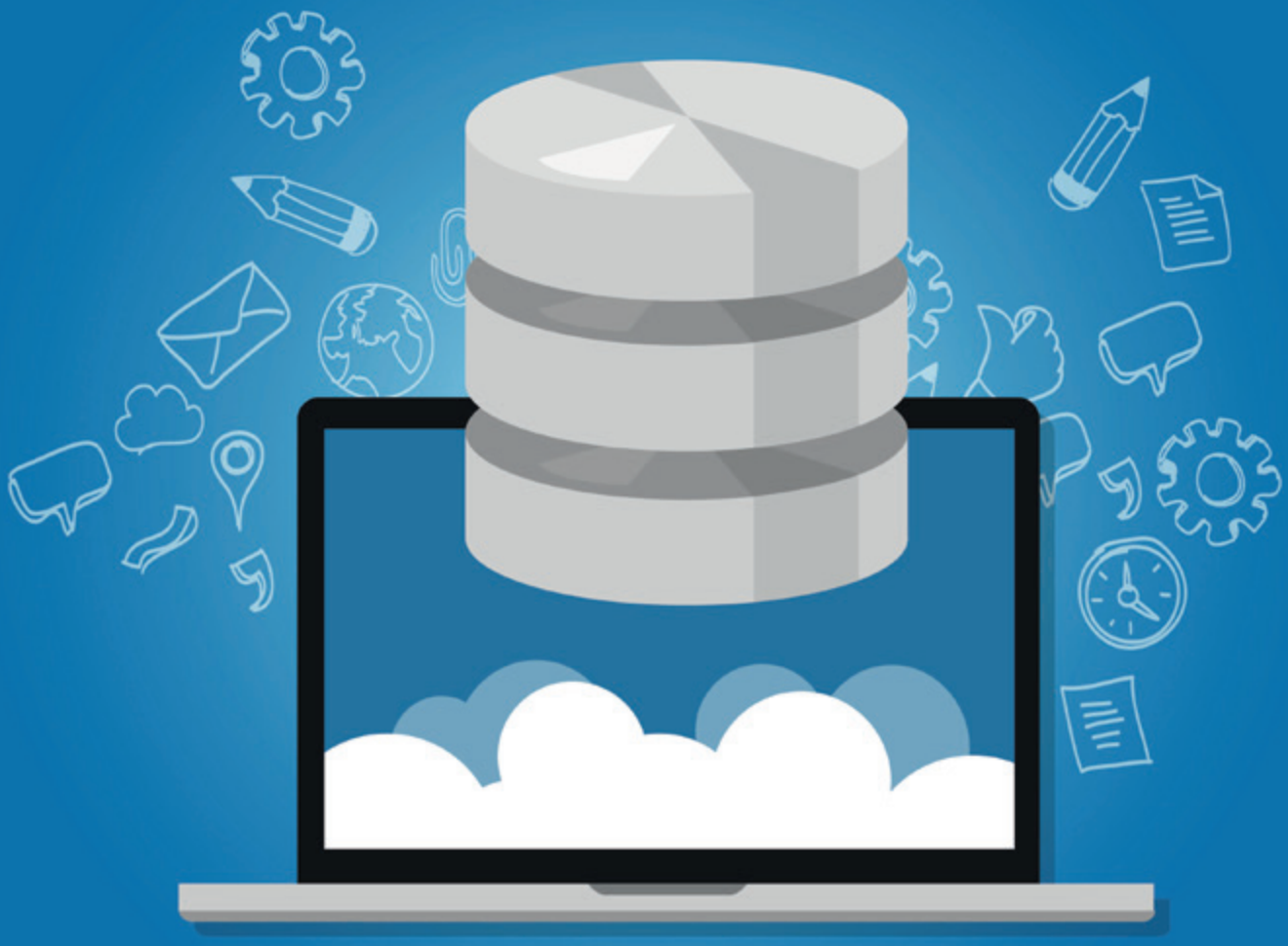
Nr. 4 | September 2017 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG Swiss Oracle User Group AOUG AUSTRIAN ORACLE USER GROUP

Neu: Oracle 12.2



12.2-New-Feature

Migration ohne Downtime

Im Interview

Vier Datenbank-Urgesteine über die neue Version

Performance

Oracle Database in der Cloud

ORACLE OPEN WORLD

German Business Network

October 1–5, 2017
San Francisco

Neue Impulse, neue Perspektiven, neue Erlebnisse

Kommen Sie auf die Oracle OpenWorld nach San Francisco –
und nutzen Sie gleichzeitig die Möglichkeit, sich mehr als je zuvor
in der deutschen Community auszutauschen!

Melden Sie sich gleich heute für d**as German Business Network**
während der Oracle OpenWorld 2017 an!

http://bit.ly/2toeFv0

Sunday October 1
Daily Touchpoint
German Business Network
in der Roof Top Bar

Monday October 2
Daily Touchpoint
German Business Network
in der Roof Top Bar

Tuesday October 3
Sunset Tour
Exklusiver Abend auf der Yacht
– by invitation only

Wednesday October 4
Oracle OpenWord 2017
Appreciation Evening
im Rahmen der Oracle OpenWorld

Bei Rückfragen zu unserem interessanten Abendprogramm
der German Business Network Community
wenden Sie sich bitte an michaela.seika@oracle.com.

ORACLE

Christian Trieb
DOAG-Vorstand und Leiter Datenbank Community

Liebe Mitglieder, liebe Leserinnen und Leser,

endlich ist die Oracle Datenbank 12.2 verfügbar. Es hat lange gedauert. Einige (Cloud-Nutzer) konnten schon eher mit dieser Version arbeiten, andere erst später. Wahrscheinlich ist das der neue Trend bei Oracle – womit man sich wohl in Zukunft abfinden muss. Die DOAG Datenbank Community wird dies weiterverfolgen und in Gesprächen mit Oracle-Repräsentanten thematisieren.

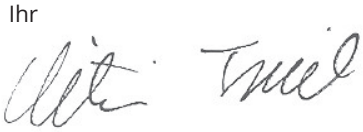
Diese Ausgabe stellt die Neuigkeiten vor; erste Erfahrungsberichte runden die Eindrücke ab. Die Datenbank 12.2 bietet einige interessante neue Funktionalitäten, die es wert sind, näher betrachtet zu werden. Nehmen Sie sich die Zeit, motiviert die eine oder andere in dieser Ausgabe vorgestellte Funktionalität auszuprobieren und eigene Erfahrungen zu sammeln.

Schon im Vorfeld hat mir das Treffen mit den Datenbank-Urgesteinen Dr. Günter Unbescheid, Dierk Lenz und Dr. Dietmar Neugebauer sehr viel Spaß gemacht. Die Datenbank 12.2 stand im Mittelpunkt, aber auch andere Themen kamen nicht zu kurz. Das Ergebnis können Sie auf Seite 8 lesen.

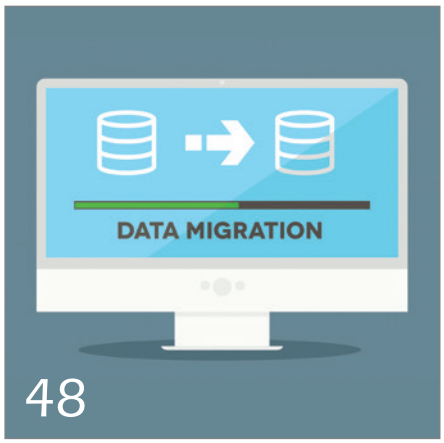
Inzwischen steht auch das Programm der DOAG 2017 Konferenz + Ausstellung fest. Auch hier wird es viele Informationen rund um die Datenbank 12.2 geben.

Ich würde mich freuen, Sie in Nürnberg zu treffen, und wünsche Ihnen viel Spaß beim Lesen dieser Ausgabe.

Ihr



Wichtige Funktionen für Datenbank-Entwickler in der neuen Version



Pluggable Databases online kopieren beziehungsweise migrieren

Testergebnisse. wie konsistent die Plattform eine bestimmte Performance liefert

Einleitung

Datenbank 12.2

Cloud

Tipps und Tricks

Intern

Timeline

11. April 2017

Das Organisationsteam der JavaLand 2017 trifft sich zur Feedbackrunde in Berlin. Nach dem Rückblick auf die Veranstaltungen beginnen bereits die Planungen für die JavaLand 2018. Um den steigenden Besucherzahlen gerecht zu werden, sind vor allem die zur Verfügung stehen Räume zu optimieren.

Auf dem Weg zum nächsten Vortrag auf der JavaLand 2017

12. April 2017

Der DOAG-Vorstandsvorsitzende Stefan Kinnen, DOAG-Vorstand und Geschäftsführer Fried Saacke, DOAG-Vorstand und Leiter der Data Analytics Community Rolf Scheuch sowie der DOAG-Vorstand und Leiter der Next Generation Community Ingo Sobik diskutieren im Berliner Büro über die PR-Strategie der DOAG. Das Spannungsfeld mit Oracle in der globalen Cloud-Welt macht es immer schwieriger, die Interessen der Anwender zu vertreten. Doch die DOAG ist in den Medien gut vernetzt, um auch auf diesem Weg die anstehenden Probleme für die Öffentlichkeit transparent zu machen.

18. April 2017

Fried Saacke, DOAG-Vorstand und Geschäftsführer, telefoniert mit Dirk Thomas Wagner, Business Development Manager ERP Cloud bei Oracle Deutschland. Oracle will den Cloud Day nicht mehr im Rahmen der DOAG Konferenz + Ausstellung veranstalten, weil das Unternehmen jetzt eine andere Strategie verfolgt. Oracle wird aber weiterhin mit einem eigenen Stand und einem Mitarbeiter-Team in Nürnberg vertreten sein.

Der Oracle-Stand auf der DOAG 2016 Konferenz + Ausstellung

28. April 2017

Das Leitungskräfteforum in Berlin bringt alle DOAG-Leitungskräfte aus den Communities und Competence Centern sowie Delegierte zusammen. Gesprächsthemen sind unter anderem Aktivitäten und Strategien für die Jahre 2017 und 2018 zur Modernisierung des Vereins.

29. April 2017

Die Delegiertenversammlung der DOAG wählt ihren Vorstand für die nächsten drei Jahre neu und setzt auf Kontinuität: Neun der Vorstände werden wiedergewählt. Kasi Färcher-Haag übernimmt als stellvertretender Vorsitzender den Finanz-Posten. Die Delegierten einigen sich zudem auf die Umbenennung der BI Community, die in Zukunft „Data Analytics Community" heißen soll. Weitere Änderungen betreffen die beiden Communities „Next Generation" und „Java", bei denen jeweils ein eigenes Logo einen Markenauftritt gewährleisten soll. Als ein allgemeines Sonderprojekt für die Website des Vereins wird beschlossen, verstärkt auf Suchmaschinenoptimierung zu setzen, um Dokumente besser auffindbar zu machen. Wolfgang Scherrer wird als Kassenprüfer wiedergewählt und bekommt mit Axel vom Stein einen zweiten Prüfer zur Seite gestellt, um mehr Flexibilität und Qualitätssicherung zu ermöglichen. Die Delegierten beschließen zudem die Anpassung des Budgets für das Jahr 2017 und verabschieden den Budgetplan 2018. Nachdem die Mitgliedsbeiträge jahrelang stabil geblieben waren, entscheidet die Versammlung eine moderate Erhöhung der Beiträge ab dem Jahr 2018.

12. Mai 2017

Die Vorstandssitzung der DOAG findet in Düsseldorf statt. Es gilt, die vielen und umfangreichen Inputs aus dem Leitungskräfteforum in Arbeitspakete zu schnüren, Zuständigkeiten festzulegen und entsprechend auf den Weg zu geben.

16. Mai 2017

Der DOAG-Anwender-Beirat trifft sich bei der IKB Bank in Düsseldorf. Stefan Kinnen, DOAG-Vorstandsvorsitzender, begrüßt als Gast Paul Wehner von Oracle, um mit ihm erste Wünsche und Fragestellungen aus diesem Gremium zu besprechen. Erfreulicherweise kündigt er Neuerungen in Produkten an, die den Erwartungen des Beirats entgegenkommen. Details dazu wird die DOAG nach Freigabe kommunizieren.

17. Mai 2017

In Olten findet der zweite SOUG Training Day zum Thema „Hochverfügbarkeit" statt. Die Teilnehmer erfahren von den kompetenten Referenten Robert Bialek, Johannes Ahrends und Herve Schweitzer, wie eine Applikation hochverfügbar gemacht wird

(„Oracle Client Failover"), wie sich die neue Multitenant-Architektur in einer hochverfügbaren Umgebung betreiben lässt und wie man mithilfe eines Rolling-Upgrades mit „near zero downtime" ein Oracle-Upgrade durchführen kann. Die Teilnehmer gehen am Abend mit wertvollen Informationen nach Hause.

Neues auf dem SOUG Training Day entdecken

22. Mai 2017

Karl-Heinz Land, digitaler Darwinist und Evangelist, wird die Keynote auf der DOAG 2017 Konferenz + Ausstellung in Nürnberg halten. Er bestätigt in einem Telefonat mit DOAG-Vorstand und Geschäftsführer Fried Saacke die Einladung aus der DOAG-Vorstandssitzung.

Karl-Heinz Land, digitaler Darwinist und Evangelist

23. Mai 2017

Carsten Wiesbaum leitet das Berliner Expertenseminar. Der Principal Oracle Consultant unterstützt seine Kunden als Architekt bei der Konzeption und Umsetzung innovativer IT-Lösungen, basierend auf dem Oracle-Software-Stack. Es geht zwei Tage lang um Microservices, einen Software-Architekturansatz, durch den sich große und komplexe Systeme in eine Anzahl von kleinen und autonomen Service-Applikationen zerlegen lassen. Die Teilnehmer nehmen viele neue Ideen mit in ihre Projekte.

30. Mai 2017

In seiner Keynote auf der DOAG 2017 Datenbank kündigt Andrew Mendelsohn, Oracle Executive Vice President für Datenbankserver-Technologien, einen neuen Release-Zyklus für die Oracle-Datenbank an. Zukünftig will Oracle jährlich ein neues sogenanntes „Feature Release" herausbringen. Was bereits im Jahr 2016 mit den Patch-Set-Updates begann, soll sich nunmehr auch in der Namensgebung der Datenbank-Releases fortsetzen: Die Patch-Nummer entspricht dem Tag, an dem das Patchset freigegeben wurde. Doch die neue Release-Planung wirft auch ein paar Fragen auf: Interessant dürfte unter anderem die Auswirkung dieser Nummerierung auf den Compatibility-Parameter werden. Eine weitere Frage kommt aus dem Auditorium: „Wie sieht eine zukünftige Zertifizierung als OCA, OCP etc. aus? Eine jährliche Zertifizierung wird wohl kaum das Ziel sein", kommentiert ein Besucher. Leider scheint sich auch in Zukunft herauszukristallisieren, dass neue Releases zunächst einmal nur für die Cloud zur Verfügung stehen und erst einige Monate später On-Premise freigegeben werden. Einen kleinen Hinweis zur XE-Version gibt es dann noch am Rande der Veranstaltung: Mit der künftigen Datenbank-Version wird auch endlich die lang erwartete Oracle XE Database aktualisiert. Es bleibt also spannend.

31. Mai 2017

Johannes Ahrends, DOAG-Themenverantwortlicher für die Datenbankadministration, und Markus Flechtner, Mitglied der DOAG-Delegiertenversammlung, stellen in ihrem Vortrag auf der DOAG 2017 Datenbank eine zukunftsträchtige Frage: „Wer hat Angst, irgendwann seinen Job als DBA zu verlieren?". Damit ist die Session „Quo vadis, DBA?" eher eine interaktive Meinungsrunde, was sich auch in dem bis auf den letzten Platz belegten Saal zeigt. Diskutiert wird mit vielen Beiträgen aus dem Publikum über Befürchtungen um das zukünftige Jobprofil des DBA. Flechtner spricht zur Beruhigung vieler Anwesenden aus, was wohl die meisten hören wollen: „Keine Panik, das DBA-Leben geht weiter." Der Job des DBA werde sich – wie auch in der Vergangenheit – selbstverständlich ändern. Doch trotz der Oracle-Cloud-Strategie ist der DBA auch in Zukunft unverzichtbar. Ein weiteres Highlight ist der durchgängige und an beiden Konferenztagen laufende Workshop. Hier steht der „Oracle Database Cloud Service" im Mittelpunkt. Das Besondere: Der Einstieg in den Workshop ist zu jeder Zeit möglich, entweder um nur kurz reinzuschnuppern oder um gleich länger teilzunehmen. Christian Trieb, Leiter der DOAG Datenbank Community, zieht nach zwei ereignisreichen Tagen ein durchweg positives Fazit: „Viele Informationen und Neuigkeiten direkt aus dem Oracle-Produktmanagement, ein intensiver Erfahrungsaustausch und ein ausgewogenes Vortragsprogramm machen die Konferenz wieder einmal zu dem Treffpunkt der Oracle-Datenbank-Community."

Viele interessante Neuigkeiten auf der DOAG 2017 Datenbank

6. Juni 2017

Das EMEA-Usergroup-Meeting startet im spanischen Valencia mit mehr als achtzig Repräsentanten von Oracle-, MySQL- und Java-Anwendergruppen aus ganz Europa. Stefan Kinnen und Ralf Kölling vertreten die DOAG im Oracle-Umfeld, Christian Trieb nimmt für den MySQL-Bereich und Fried Saacke für Java teil. Im Mittelpunkt steht die Zusammenarbeit der Usergroups untereinander und mit Oracle.

7. Juni 2017

Im Rahmen einer Breakout-Session auf dem EMEA-Meeting beschließen die Vertreter der Oracle-Usergroups, ihr selbst organisiertes Treffen Anfang September auf der Anwenderkonferenz der UK Usergroup abzuhalten. Das unabhängig von Oracle organisierte Treffen ist von zunehmender Bedeutung für die teilnehmenden Anwendergruppen, weil dort auch die kritischen Themen auf die Tagesordnung kommen.

8. Juni 2017

Beim EMEA-Usergroup-Meeting steht heute ORAWORLD, das gemeinsame E-Magazin der Usergroups, ganz oben auf der Agenda. Die bisherige Redaktionsarbeit von Dr. Dietmar Neugebauer von der DOAG und Ami Aharonovich von der ilOUG (Israel) wird künftig durch Ann-Sofie Vikstrom Often von der Oracle User Group Norway, Andrejs Vorobjovs von der Oracle User Group Latvia und Jean-Jacques Camps von der Oracle User Group France verstärkt. Die Einbindung aller Usergroups soll die Verbreitung nachhaltig ausweiten. ORAWORLD ist das erste weltweit verfügbare Organ der Anwendergruppen (siehe "www.oraworld.org").

Das E-Magazin ORAWORLD, Ausgabe 5

15. Juni 2017

Die Mitarbeiterinnen und Mitarbeiter der DOAG-Geschäftsstelle treffen sich zum diesjährigen Team-Workshop. Diesmal geht es um Projektmanagement, insbesondere um die Planung und Steuerung von Veranstaltungen.

19. Juni 2017

Am Vortag zur Anwenderkonferenz der Austrian Oracle User Group findet im Konferenzhotel Austria Trend Hotel Savoyen Vienna der „AOUG Training Day 2017 – Learn from the Best!" statt. In mehreren parallelen Halbtagsworkshops können sich die Besucher über aktuelle Themen wie „Oracle Lizenzierung", „Datenschutz und Privacy" und „Execution Plan und Datenmanagement" informieren. Die Workshops werden unter anderen von den Experten Christian Antognini und Gerald Venzl gehalten. Am späten Nachmittag geht es dann mit den Sprechern und zahlreichen Teilnehmern der Anwenderkonferenz in einer Nostalgie-Straßenbahn („Bim") zu einer kleinen Sightseeing-Tour rund um die Wiener Ringstraße bis zum Riesenrad. Bei schönster Abenddämmerung gibt es anschließend für alle im Stadtgasthaus Eisvogel kulinarische Köstlichkeiten wie Wiener Schnitzel, Sachertorte und einen guten Wein.

Speaker und Teilnehmer der AOUG-Anwenderkonferenz vor dem Wiener Riesenrad

20. Juni 2017

Im Konferenzhotel Austria Trend Hotel Savoyen Vienna findet die Anwenderkonferenz 2017 der Österreichischen User Group statt. 170 Teilnehmer informieren sich einen Tag lang zu zahlreichen Themen. Die Keynote wird in diesem Jahr in einer unterhaltsamen Doppel-Conference von Maria Colgan und Gerald Venzl zum Thema „Oracle12c und DevOps" gehalten. Beide zeigen im spielerischen Dialog, wie sich Developer und DBAs mit aktuellen 12c-Features gegenseitig unterstützen und sich damit das Leben leichter machen können. Im Anschluss gibt es 28 Fachvorträge mit Sprechern aus insgesamt elf Nationen in vier parallelen Tracks. Nächstes Jahr feiert die AOUG als eine der ältesten Oracle User Groups ihren 30sten Geburtstag.

Der AOUG-Vorstand ist guter Dinge

Von links: Dr. Dietmar Neugebauer, Dierk Lenz, Christian Trieb, Dr. Günter Unbescheid und Wolfgang Taschner

„Jedes neue Release hat seine eigene Marketingstrategie …“

Oracle hat die neue Datenbank-Version 12.2 herausgebracht. Dr. Dietmar Neugebauer, ehemaliger Vorstandsvorsitzender der DOAG, und Wolfgang Taschner, Chefredakteur des Red Stack Magazin, sprachen darüber mit Christian Trieb, DOAG-Vorstand und Leiter der Datenbank Community, Oracle ACE Director, Paragon Data GmbH, Dierk Lenz von Herrmann & Lenz Services und DOAG-Botschafter 2012 sowie mit Dr. Günter Unbescheid von Database Consult und DOAG-Botschafter 2010.

Wie bist du mit der neuen Version in Kontakt gekommen und was war dein erster Eindruck?

Dierk Lenz: Ich war im Beta-Programm und habe bereits die frühen Versionen sehen können. Mein erster Gedanke war, ob es überhaupt einen Unterschied zur Version 12.1 gibt.

Dr. Günter Unbescheid: Ich hatte die Informationen über den Start des Beta-Programms erhalten, wollte allerdings diesmal nicht teilnehmen und habe auf das offizielle Release gewartet. Anfangs war ich ziemlich enttäuscht, dass dieses nur in der Cloud zur Verfügung stand. Ich habe mir zunächst die Liste der neuen Features angeschaut und dort viele interessante neue Sachen entdeckt. Ein wichtiges Thema ist natürlich die neue Multitenant-Architektur. Beim Upgrade auf 12.1 drehte sich vieles um die neuen Adaptive-Features, was bei den Kunden etliche Probleme verursacht hat. Auch das ist mit 12.2 jetzt verbessert worden.

Christian Trieb: Das Beta-Programm dauerte dieses Mal ja fast eineinhalb Jahre. Das erste, was ich damals gehört habe, war, dass sich Oracle anderen Datenbanken angleicht, Stichwort „Sharding“. Dieser Feature ist aufwändig umsetzbar und daher sicher nur für eine kleine Zielgruppe interessant.

Dierk Lenz: Man hört mittlerweile von Oracle, dass Multitenant die neue Architektur wird. Beim Sharding darf es hingegen keine Multitenant-Datenbank sein. Hier kann man nur auf ein späteres Release hoffen.

Christian Trieb: Dennoch halte ich Multitenant für eine gute zukünftige Architektur.

Gibt es denn schon erste Reaktionen von Kunden auf die neue Version?

Dr. Günter Unbescheid: Ich kenne Kunden, die von den neuen Features begeistert sind und sofort migrieren möchten. Es gibt allerdings noch Bedenken wegen des großen Aufwands, um auf

die Multitenant-Architektur zu gehen.
Christian Trieb: Interesse an der neuen Version ist da, der Umstieg ist jedoch mit der Diskussion darüber verbunden, ob die Multitenant-Architektur bereits jetzt eingeführt werden kann, wenn damit noch nicht alle Features möglich sind.

Welche neuen Features gefallen dir bei der Version 12.2 besonders gut?
Dierk Lenz: Ein Feature, aufgrund dessen einer meiner Kunden möglichst schnell migrieren möchte, ist die Erhöhung der maximalen Länge von Bezeichnern von 30 auf 128 Bytes. Gut ist auch die Möglichkeit, einzelnen Feldern die Sortier- und Vergleichseigenschaften mitgeben zu können, was bisher nur auf Sessionebene möglich war. Ein Wermutstropfen hingegen ist, dass diese Funktion in PL/SQL noch nicht vollständig implementiert ist.
Dr. Günter Unbescheid: Die neuen Features, die mir am besten gefallen, sind natürlich die, die den Kunden am meisten nützen. Hier gibt es einiges im Partitioning-Umfeld wie erweiterte Online-Funktionen, die für bestimmte Projekte relevant sind, sowie auch bei Data Guard. Im Bereich „Security" hat sich hingegen nicht viel getan.

Auf welche Features hätte Oracle verzichten können?
Dr. Günter Unbescheid: In meinem Projekt-Umfeld gab es Irritationen zuhauf, weil die Adaptive-Features aus der Version 12.1 bereits im nächsten Release wieder völlig umgestellt worden sind. Auch die Heatmaps haben in 12.1 bei großen Datenbanken Probleme bereitet.
Dierk Lenz: Mein Eindruck ist, dass Oracle manche Features nicht in „real life"-Umgebungen umfassend testet. Von daher schalte ich die Adaptive-Features standardmäßig ab.
Dr. Dietmar Neugebauer: Mir fehlt hier eine eindeutige Strategie bei Oracle, solche Mängel auch zu kommunizieren, sobald sie bekannt sind. Damit könnte man verhindern, dass manche Kunden die Probleme erst dann erkennen, wenn das System produktiv geht.

Was sind die Vorteile, die Datenbank in der Oracle-Cloud zu betrieben?
Dierk Lenz: Ich habe bei einigen Kunden Erfahrungen mit Test-Datenbanken in der Oracle-Cloud sammeln können. Da kam es öfter mal vor, dass das System wegen Wartungsarbeiten für einige Stunden stillstand. Aus diesem Grund ergibt es keinen Sinn, mit größeren Anwendungen in die Oracle-Cloud zu gehen. Hier sind definitiv noch Nachbesserungen seitens Oracle erforderlich.
Christian Trieb: Wir haben kürzlich eine Datenbank in die Oracle-Cloud migriert und als Primärseite genutzt. Technisch hat das mit etwas Eingewöhnungsaufwand auch gut funktioniert, aber wie schon gesagt, solange Oracle kein Service-Level-Agreement eingeht und bestimmte Leistungen vertraglich zusichert, ist es schwierig, ein Produktivsystem in der Oracle-Cloud betreiben. Um hingegen mal ein neues Feature zu testen, ist es durchaus sinnvoll, das in der Cloud zu erledigen.

Dr. Günter Unbescheid


Dierk Lenz

Dierk Lenz: Ich kenne allerdings noch mittelständische Unternehmen, die über keine Internetanbindung mit einer hohen Bandbreite verfügen, sodass ein Cloud-Projekt schon aus diesem Grund nicht machbar ist.
Dr. Günter Unbescheid: Ich könnte mir auf der anderen Seite auch eine Trendwende vorstellen, indem beispielsweise ein kleineres Unternehmen die Komplexität seiner Systeme und Daten reduziert und dann gar nicht auf die Cloud angewiesen ist.
Dierk Lenz: Ein anderer Punkt ist die Problematik bei der Oracle-Lizenzierung. Die ist in der Cloud derzeit einfacher und eindeutiger.

Wie sind eure Erfahrungen bei anderen Cloud-Anbietern?
Dierk Lenz: Wir betreiben ein eigenes Produkt in der europäischen Amazon-Cloud. Mit deren Verfügbarkeit und Leitungsfähigkeit gibt es keine Probleme.

Wird sich euer Arbeitsfeld ändern, wenn die Datenbanken künftig in der Cloud betrieben werden?
Christian Trieb: Das Arbeitsfeld des Administrators wird sich ändern. Er muss in der Lage sein, eine Verbindung in die Cloud aufzubauen und entsprechende Firewalls einzurichten. Auf der anderen Seite sichert Oracle zu, einen Teil der Datenbank-Administration in der Cloud zu übernehmen.
Dr. Günter Unbescheid: Nachdem die Administration einer Datenbank immer komplexer und aufwändiger wird, ist natürlich

die Versuchung groß, diese in die Cloud zu verlagern und Oracle dort beispielsweise das Patchen zu überlassen.

Kann der Datenbank-Administrator eine Datenbank in der Cloud überhaupt noch groß beeinflussen, beispielsweise zum Performance-Tuning?

Dr. Günter Unbescheid: Er kann natürlich beim Design der Datenbank vieles beeinflussen, beispielsweise bei der Partitionierung. Beim Instanz-Tuning hingegen bestehen in der Cloud weniger Möglichkeiten.

Christian Trieb: Nach meiner Einschätzung lassen sich rund achtzig Prozent der Tuning-Maßnahmen beim Design und in der Applikation durchführen; daran ändert dann auch die Cloud nichts.

Dierk Lenz: Es ist natürlich richtig, dass Probleme mit der Datenbank-Performance zum größten Teil vom Design und von der Art und Weise der Programmierung kommen, aber es stellt sich schon die Frage, ob ein Provider Informationen über seine Infrastruktur herausgibt beziehungsweise ob man diese überhaupt beeinflussen kann.

Christian Trieb

Habt ihr Bedenken hinsichtlich Datensicherheit in der Cloud?

Christian Trieb: Bei Test- und Entwicklungssystemen sehe ich hier kein Problem. Finanz- und andere sensible Daten würde ich natürlich ohne intensive vorherige Sicherheitsprüfung nicht in die Cloud geben.

Dierk Lenz: Insbesondere, wenn die Cloud in den USA gehostet wird, habe ich große Bedenken. Dort sind gerade Erlasse unterschrieben worden, dass die Daten eingesehen werden dürfen.

Dr. Günter Unbescheid: Hinzu kommt, dass die amerikanischen Geheimdienste selbst auf Daten amerikanischer Unternehmen zugreifen dürfen soll, wenn diese außerhalb des Landes gehostet sind.

Oracle hat die neue Version zunächst nur in der Cloud zur Verfügung gestellt. Wird es die Version 13 nur noch in der Cloud geben?

Dr. Günter Unbescheid: Ich kann mir nicht vorstellen, dass es so kommen wird.

Dierk Lenz: Das würde insbesondere hier in Deutschland zu großen Problemen führen.

Mit welcher Datenbank-Version hast du zuerst gearbeitet?

Christian Trieb: Ich habe mit Oracle 5 im Versicherungsumfeld begonnen.

Dierk Lenz: Als ich bei Oracle in Düsseldorf eingestellt wurde, war die Version 5 aktuell. Einen Monat später begannen die Schulungen für Oracle 6.

Dr. Günter Unbescheid: Ich erinnere mich noch ganz genau an die Version 5.1.22 und MS/DOS.

Wie beurteilst du seitdem die Entwicklung der Oracle-Datenbank?

Dr. Günter Unbescheid: Die Releases 6 bis 8 hatten immer neue Features, auf die neunzig Prozent der Kunden bereits gewartet haben. Seit den letzten Releases ist bei mir der Eindruck entstanden, dass zwar immer noch neue Features hinzukommen, diese aber nur noch für wenige Kunden von großer Bedeutung sind. Hier greift für die Mehrheit dann der Druck zur Migration nur noch durch das Ende des Supports für die vorherige Version, obwohl die Anwendungen auch unter einem früheren Release weiterhin zufriedenstellend funktionieren würden.

Dr. Dietmar Neugebauer: Oracle begleitet ja immer jedes neue Release mit einer eigenen Marketingstrategie, um die Kunden zur Migration zu motivieren. Nach meiner Meinung fehlt aber immer noch eine Datenbank, die man patchen oder migrieren kann, ohne dass es überhaupt ein Anwender mitbekommt.

Christian Trieb: Eine der größten Änderungen war für mich die Einführung der Multitenant-Architektur. Bezüglich der Features ist die Zeit der großen Neuerungen für die Breite der Masse vorbei.

Dierk Lenz: Für mich ist die Entwicklung des Optimizers über die ganze Zeitspanne sehr spannend. Bis Version 10 gab es die Vorstellung, dass, wenn man Statistiken nur genau genug berechnen könnte, sich daraus der perfekte Plan ableiten lassen würde. Bei 11*g* ist daraus fast eine Art künstliche Intelligenz entstanden, um Abläufe zu optimieren. Nachdem dies in der Version 12.1 deutlich über das Ziel hinausgeschossen war, wird jetzt in 12.2 wieder zurückgerudert. Es ist immer schwieriger zu verstehen, wie der Optimizer arbeitet.

Was sind die Stärken der Oracle-Datenbank?

Dierk Lenz: Die Robustheit zählt zu den absoluten Stärken. Im Fehlerfall ist ein Wiederanlaufen recht einfach, vorausgesetzt, man hat eine vernünftige Backup-Strategie im Einsatz.

Christian Trieb: Ich schätze auch die Kompatibilität. Die Daten sind sowohl horizontal als auch vertikal über alle Versionen und Betriebssysteme migrierbar.

Wo liegen die Schwächen?

Dr. Günter Unbescheid: Die Datenbank ist ein sehr komplexes System geworden. Damit steigt auch die Fehleranfälligkeit. Dafür wäre dann ein guter Support erforderlich, den es leider nicht gibt. Oft dauert es Tage und Wochen, bis dort ein Problem überhaupt verstanden wird.

Christian Trieb: Obwohl jetzt die maximale Länge von Bezeichnern von 30 auf 128 Bytes erhöht wurde, ist der Datenbank-Name immer noch achtstellig geblieben. Das stört mich schon. Ein anderes Thema ist die Lizenzpolitik von Oracle. Bestimmte Features stehen nur in entsprechender Kombination zur Verfügung.

Dr. Dietmar Neugebauer

Dierk Lenz: Die hohe Konfigurierbarkeit macht die Datenbank vom Start weg sehr komplex. Gerade für kleinere und mittlere Anwendungen sind die Möglichkeiten zu wenig transparent. Hier wäre ein einfaches und schnelles Aufsetzen angebracht.

Wie sieht deine ideale Datenbank-Anwendung aus?

Dierk Lenz: Als langjähriger Berater kann ich immer nur gebetsmühlenartig wiederholen, nicht auf die gerade vom Oracle-Marketing angepriesenen neuen Features zu schauen, sondern in erster Linie ein klares und gut strukturiertes Datenbank-Design zu entwerfen.

Dr. Günter Unbescheid: Hier sind auch die Java-Entwickler gefragt, bei der Programmierung die Datenbank nicht einfach nur als Blackbox zu betrachten, sondern auch auf deren Eigenheiten zu schauen und diese entsprechend zu nutzen.

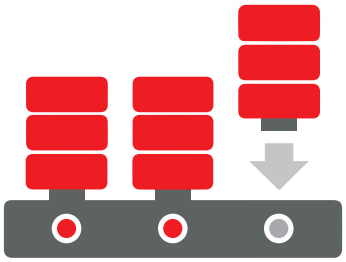
In welche Richtung sollte sich die Oracle-Datenbank weiterentwickeln?

Dr. Günter Unbescheid: Es wäre für mich sehr schwer, hier Vorgaben aufzustellen. Aus meinen Projekterfahrungen ist die Robustheit neuer Features ein ganz wichtiger Punkt. Wenn also etwas Neues herauskommt, sollte es auch gleich richtig einsetzbar sein. Ansonsten habe ich eher kleine Punkte wie die Erweiterung der Online-Fähigkeiten oder der Liste der Multitenant-Features.

Dierk Lenz: Ich wünsche mir vor allem eine lizenztechnische Stabilität. Oracle sollte hier die kleineren und mittleren Unternehmen nicht aus den Augen verlieren und für diese Kunden berechenbar bleiben.

Christian Trieb: Wichtig ist einmal, die Features stabiler zu machen, und zum anderen, den Support nicht zu vergessen. Hier gibt es noch viel Verbesserungspotenzial.

Dr. Dietmar Neugebauer: Nachdem zukünftig immer mehr unstrukturierte Daten anfallen, sollte die Datenbank auch dafür eingerichtet sein oder die Voraussetzung dafür schaffen, andere transaktionale Systeme ankoppeln zu können.



Oracle 12.2 Multitenant – der nächsten Level

Ralf Durben, ORACLE Deutschland B.V. & Co. KG

Die Multitenant-Option der Oracle-Datenbank wird mit dem Release 12.2 durch viele neue Funktionalitäten auf ein neues Niveau gehoben. Beginnend mit der Reduzierung von Downtime bei Cloning- oder Transport-Aktionen sowie der besseren Kontrolle über Ressourcen-Nutzung der Pluggable Datenbanken bis hin zur Optimierung des Betriebs von Anwendungsfarmen ist alles auf den Betrieb von Umgebungen jeglicher Größe ausgelegt – On-Premise und in der Cloud. Damit wird Multitenant als Konsolidierungslösung für Oracle-Datenbanken noch attraktiver.

Die mit der Version 12*c* eingeführte Multitenant-Architektur kann als eine Konsolidierungs- und Virtualisierungs-Plattform betrachtet werden: Mehrere Datenbanken teilen sich gemeinsame Ressourcen, um mit diesen sparsamer umzugehen und damit Kosten einzusparen. Das zweite Release von 12*c* geht diesen Weg konsequent weiter. So können sich mehrere Datenbanken das gleiche Datenmodell und sogar statische Daten zu einer Anwendung teilen, um mandantenorientierte Datenbanken noch effizienter zu nutzen. Damit lassen sich SaaS-Lösungen für neue Kunden noch schneller aktivieren und viel effizienter warten. Viele Operationen, wie zum Beispiel das Verschieben und Klonen von Datenbanken, sind nun online möglich.

Grundgedanke von Oracle Multitenant

Die mit 12*c* Release 1 eingeführte Multitenant-Architektur grenzt sich von der bis zur Version11*g* verwendeten (jetzt Non-CDB genannten) Architektur dadurch ab, dass sich mehrere Datenbanken verschiedene Ressourcen teilen. Bei der Non-CDB-Architektur wird für jede Oracle-Datenbank mindestens eine sogenannte „Instanz", bestehend aus Hauptspeicher (SGA) und Hintergrundprozessen, gestartet. Diese Ressourcen sind auch dann in Gebrauch, wenn die Datenbank nicht viel arbeitet. So ist der Hauptspeicher einer Instanz also auch dann exklusiv in Verwendung, wenn zum Beispiel eine Test-Datenbank ohne große Last geöffnet ist. Dieser Hauptspeicher kann nicht anderweitig verwendet werden, etwa für andere Datenbanken auf dem gleichen Server. In vielen IT-Umgebungen werden heute pro Datenbank dedizierte virtuelle Maschinen betrieben, die ihrerseits Ressourcen belegen, auch wenn diese aufgrund der Last gar nicht erforderlich sind. Auf diese Weise werden ganze Server beziehungsweise Ressourcen dieser Server unnötig verwendet.

Oracle Multitenant stellt dem ein Konzept der dynamischen Zuweisung von Ressourcen gegenüber, nach dem sich mehrere Datenbanken (Pluggable Database, PDB) eine gemeinsame Instanz teilen. Diese Instanz wird von einer zentralen Datenbank (Container-Datenbank, CDB) betrieben. Damit lassen sich Hauptspeicher und Prozessorleistung eines Servers (ob virtuell oder nicht) auch wirklich sinnvoll nutzen.

Aber nicht nur die Ressourcen Hauptspeicher oder CPU werden gemeinsam verwendet: Auch Speicherbereiche in den Datendateien sollen möglichst ressourcensparend zum Einsatz kommen. So wird der Teil des Data Dictionary, der direkt nach der Erstellung einer Datenbank vorhanden und unabhängig vom Inhalt einer Datenbank ist, nur einmal und zentral in der CDB gespeichert. Auf dieser Basis werden nachfolgend die Neuerungen in Oracle 12*c* Release 2 beschrieben.

Anzahl und Voraussetzungen von PDBs in einer CDB

Die maximale Anzahl von PDBs in einer CDB wurde auf 4.096 angehoben, sofern diese in einer Exadata-Maschine oder in einem Oracle Database Cloud Service (Exadata oder Non-Exadata) betrieben wird. In allen anderen Fällen bleibt es bei der maximalen Anzahl von 252.

Die Zeichensätze der PDBs in einer CDB können jetzt unterschiedlich sein, vorausgesetzt, dass diese zum Zeichensatz der CDB kompatibel sind, also eine Teilmenge darstellen. Aus diesem Grund empfiehlt es sich, für die CDB den Zeichensatz „AL32UTF8" zu verwenden. In diesem Fall können dann zum Beispiel eine PDB mit dem Zeichensatz „WE8ISO8859P1" und eine weitere PDB mit dem Zeichensatz „KANJI" in dieser CDB betrieben werden.

PDBs und die Nutzung von SGA-Ressourcen

Alle PDBs, die in einer CDB betrieben werden, verwenden die gleiche SGA, wobei

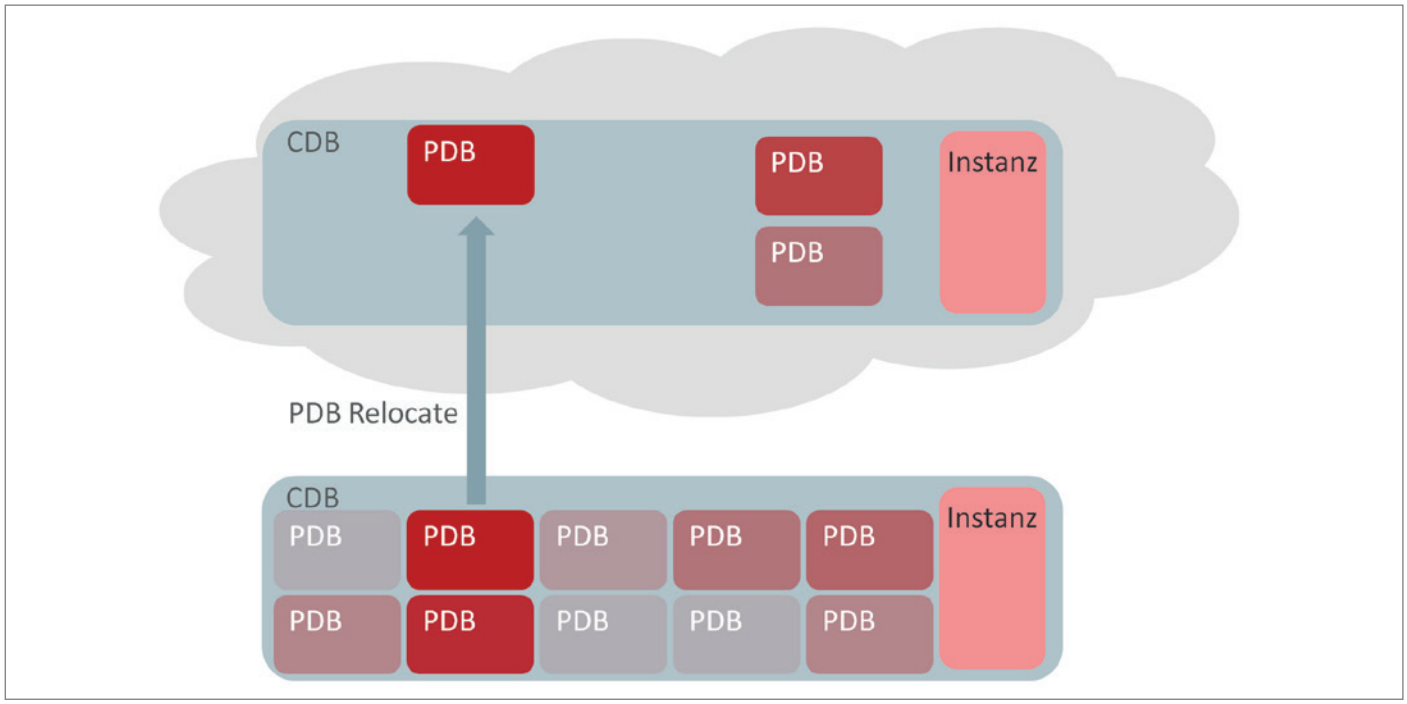


Abbildung 1: Mit PDB Relocate einfach in die Cloud

natürlich jede PDB nur mit den eigenen Daten arbeiten kann. Dabei findet automatisch eine Ressourcenzuteilung statt, die sich an der momentanen Arbeitslast orientiert. Ab Oracle 12*c* Release 2 können dabei Mindestwerte für die einzelnen Bereiche über folgende Initialisierungsparameter eingestellt werden:

- DB_CACHE_SIZE
- SHARED_POOL_SIZE
- PGA_AGGREGATE_LIMIT
- PGA_AGGREGATE_TARGET
- SGA_MIN_SIZE
- SGA_TARGET

Voraussetzung dafür ist, dass auf der CDB-Ebene die Einstellungen „NONCDB_COMPATIBLE=false“ und „MEMORY_TARGET=0“ oder nicht gesetzt getroffen wurden.

Erstellen von PDBs durch „Hot Cloning“

PDBs lassen sich durch das Klonen bereits bestehender PDBs erstellen. Bislang musste die Quell-PDB dazu in einen transaktionskonsistenten Zustand gebracht werden und „READ ONLY“ geöffnet sein. Ab 12*c* Release 2 ist dies nicht mehr notwendig und die Quell-PDB kann bei normaler Nutzung (auch schreibend) geklont werden. Voraussetzung dafür ist, dass die CDB der Quell-PDB im ARCHIVELOG-Modus betrieben wird, was für jede Produktiv-Datenbank der Fall sein sollte.

Während des Klonens wird die Quell-PDB zunächst kopiert; anschließend werden alle Datenänderungen, die während des Kopierens vorgenommen wurden, automatisch über die Redo-Log-Daten in die Klon-PDB eingespielt. Die Kommando-Syntax hat sich dabei gegenüber dem Release 1 nicht geändert und das „Hot Cloning“ funktioniert auch zwischen zwei unterschiedlichen CDBs.

Refreshable PDB-Klone

Man kann einen PDB-Klon auch so erstellen, dass alle Datenänderungen, die in der Quell-PDB vorgenommen werden, auch im Klon angewendet werden. Das ist zum Beispiel sinnvoll, wenn die Klone als Datenfarm für Lesezugriffe verwendet werden sollen und die Aktualisierung der Daten nicht sofort durchgeführt werden muss. Die Aktualisierung lässt sich manuell vornehmen oder man definiert ein Zeitintervall für automatische Aktualisierungen. Die Klon-PDB muss dabei im „read only“-Modus betrieben werden.

Quell- und Klon-PDB müssen in verschiedenen Container-Datenbanken liegen. Man kann jedoch in einem Szenario von zwei CDBs die Quell-PDB in der CDB1 und mehrere Refreshable-Klone in der CDB2 betreiben. Außerdem ist zu beachten, dass in der Ziel-CDB ein Datenbank-Link für den Zugriff auf die Quell-CDB erforderlich ist. Dieser baut eine Verbindung zur Quell-CDB mit einem Benutzer auf, der in beiden CDBs existieren muss. Man erstellt einen Refreshable-Klon zum Beispiel mit „CREATE PLUGGABLE DATABASE pdb2 FROM pdb1@db_link_to_pdb1 ... REFRESH MODE MANUAL;“ für manuelle Refreshs. Alternativ stehen „REFRESH MODE EVERY n MINUTES“ für automatische Aktualisierungen und „REFRESH

```
ALTER SESSION SET CONTAINER=pdb2;
ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE REFRESH;
ALTER PLUGGABLE DATABASE OPEN READ ONLY;
```

Listing 1

MODE NONE“, um die Aktualisierung abzuschalten, zur Verfügung.

Ein manuelles Refresh wird erzeugt, indem man sich mit der PDB verbindet, diese kurz schließt, ein REFRESH durchführt und dann die PDB wieder öffnet *(siehe Listing 1)*. Die Daten im Klon sind dann auf dem aktuellen Stand. Auf diese Weise lassen sich sehr leicht Kataloge zur Verfügung stellen, die zum Beispiel einmal pro Tag aktualisiert werden müssen.

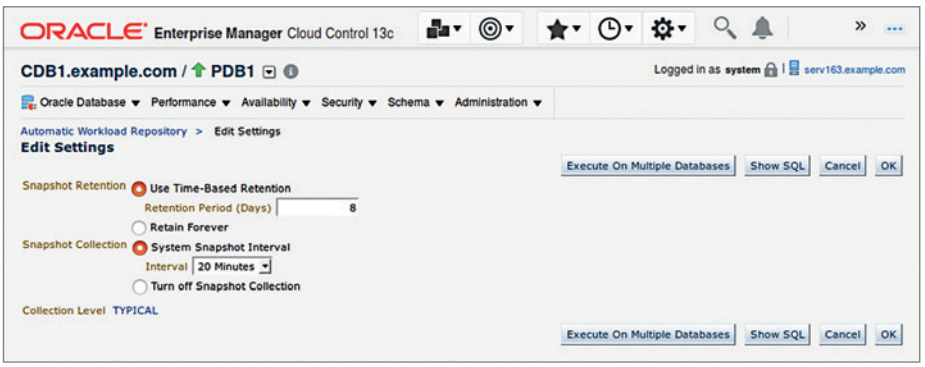


Abbildung 2: AWR-Einstellungen für PDBs in Cloud Control

PDB Relocate

Pluggable Datenbanken lassen sich von einer CDB zu einer anderen verschieben. Dabei stehen zwei Verfahren zur Verfügung:

- Seit 12.1: Aushängen (Unplug) aus der alten CDB, Kopieren der Datendateien inklusive einer XML-Datei, die die Struktur der PDB beschreibt, und Einhängen (Plug) der PDB in die neue CDB.
- Neu mit 12.2: Relocate-Operation, wobei der Umzug komplett automatisch und mit minimaler Downtime erfolgt.

```
CREATE PLUGGABLE DATABASE pdb1 FROM pdb1@db_link_to_pdb1
FILE_NAME_CONVERT = …
RELOCATE AVAILABILITY MAX;
```

Listing 2

Die in 12.1 vorhandene Methode zur Verschiebung einer PDB mittels Unplug/Plug ist ab dem Zeitpunkt des Unplug mit Downtime verbunden. Um diese zu minimieren, gibt es jetzt eine neue Methode: PDB Relocate. Damit lässt sich eine PDB fast online von einer CDB in eine andere CDB transportieren. Dazu muss es einen Datenbank-Link von der Ziel- zur Quell-CDB geben, wie schon zu den Refreshable Clones beschrieben, wobei der im Datenbank-Link verwendete Benutzer neben dem Recht „CREATE PLUGGABLE DATABASE“ auch das Recht „SYSOPER“ oder „SYSDBA“ besitzen muss, und das mit der Option „CONTAINER=ALL“.

Grundsätzlich kann durch PDB Relocate eine PDB auch zwischen CDBs verschoben werden, die auf unterschiedlichen Plattformen laufen, jedoch müssen diese die gleiche Filesystem-„Endianness“ besitzen. Zudem muss die Ziel-CDB natürlich auch alle Datenbank-Optionen wie die Quell-CDB installiert haben *(siehe Abbildung 1)*.

Die Quell-PDB muss dabei entweder im „ARCHIVELOG“ betrieben werden (vorgegeben durch die Quell-CDB) oder im „read only“-Modus sein. *Listing 2* zeigt das Vorgehen für das PDB Relocate. Man kann während der „RELOCATING“-Operation immer noch mit der Quell-PDB arbeiten und auch Daten ändern. Das Ergebnis ist eine Kopie der PDB im Status „MOUNTED“, die mit „ALTER PLUGGABLE DATABASE pdb1 OPEN;“ geöffnet wird.

Während des Öffnens der PDB wird die transportierte PDB auf den neuesten Stand gebracht. Dabei ist die PDB weder in der Quell-CDB noch in der Ziel-CDB verwendbar. Jetzt ist also eine Downtime zu verzeichnen. Die PDB in der Ziel-CDB wird auch automatisch als Service eingetragen.

Je nachdem, ob beide CDBs mit dem gleichen Listener arbeiten oder nicht, verwendet man die Option „AVAILABILITY“. Falls es unterschiedliche Listener sind, nimmt man „AVAILABILITY MAX“ und der Quell-Listener wird so konfiguriert, dass er die Verbindungsaufrufe korrekt an die neue Adresse weiterleitet. Aus Sicht der Clients ändert sich also nichts. Um die PDB, die jetzt ihren Standort gewechselt hat, später per Recovery wiederherstellen zu können, sollte ein Backup der PDB erstellt werden.

PDB Lockdown Profiles

Durch die Konsolidierung mehrerer Pluggable Datenbanken in Container-Datenbanken stellen sich verschiedene Fragen hinsichtlich des Umgangs mit Sicherheitsanforderungen in deren Betrieb. Inwieweit ein Administrator einer Pluggable Datenbank privilegierte Kommandos ausführen kann oder welche Datenbank-Optionen durch eine PDB verwendet werden dürfen, kann nun vorgegeben werden. Gerade Letzteres kann in einer konsolidierten Umgebung für ein Lizenzmanagement von großem Interesse sein.

Auch der Zugriff auf das Netzwerk mithilfe der Packages „UTL_TCP“, „HTL_HTTP“ und anderer sowie der Zugriff auf das Betriebssystem beispielsweise per „UTL_FILE“ sollte bei Sicherheitsüberlegungen mit einbezogen werden. Zu diesem Zweck sind sogenannte „Lockdown Profiles“ definierbar. Diese erlauben oder verbieten folgende Nutzung:

- *Datenbank-Optionen*
  Zurzeit sind die Optionen „Partitioning“ und „Database Queuing“ unterstützt
- *Datenbank-Features*
  Es wird eine Reihe von Features unterstützt, die im Handbuch beschrieben sind, darunter „AWR_ACCESS“, „COMMON_SCHEMA_ACCESS“ und „NETWORK_ACCESS, OS_ACCESS“
- *SQL-Statements*
  Die Nutzung der SQL-Statements „ALTER DATABASE“, „ALTER PLUGGABLE DATABASE“, „ALTER SESSION“ und „ALTER SYSTEM“ ist steuerbar

Die Lockdown Profiles werden auf der Ebene der CDB mit einem Namen erstellt

verbunden. Auf der Ebene der PDB kann maximal ein Lockdown Profile, das in der CDB existieren muss, per Instanz-Parameter aktiviert sein. Das Lockdown Profile lässt sich in einer CDB recht einfach mit „CREATE LOCKDOWN PROFILE wdg;" anlegen. Dazu ist das Privileg „CREATE LOCKDOWN PROFILE" erforderlich. Es entsteht ein leeres Profil, das sich durch „ALTER LOCKDOWN PROFILE" mit Inhalt füllen lässt. Dabei werden Features, Optionen oder Statements ein- beziehungsweise abgeschaltet. Dazu einige Beispiele:

- *Abschalten der Datenbank-Option „Partitioning"*
  ALTER LOCKDOWN PROFILE wdg DISABLE OPTION = ('Partitioning');

- *Abschalten des Datenbank-Features „XDB"*
  ALTER LOCKDOWN PROFILE wdg DISABLE FEATURE = ('XDB_PROTOCOLS');

- *Abschalten des Statements „ALTER SYSTEM"*
  ALTER LOCKDOWN PROFILE wdg DISABLE STATEMENT = ('ALTER SYSTEM');

Pro PDB lässt sich maximal ein Lockdown Profile aktivieren, indem dort in einer Datenbanksitzung mit „ALTER SYSTEM SET PDB_LOCKDOWN = wdg;" der Instanz-Parameter „PDB_LOCKDOWN" gesetzt wird, der aber nur für die betreffende PDB wirkt. Dazu ist das „ALTER SYSTEM"-Privileg in der PDB erforderlich.

Lokales AWR

Das Automatic Workload Repository (AWR) wurde mit der Version 10g eingeführt und muss über das Diagnostics Pack lizenziert sein. Technisch werden dabei in der Datenbank regelmäßige Zugriffe direkt auf die System Global Area (SGA) der Datenbank durchgeführt, um sehr performant auf Nutzungsstatistiken der Datenbank-Sitzungen zuzugreifen. Die Daten sind in Tabellen im Tablespace „SYSAUX" gespeichert.

Das AWR bietet umfangreiche Daten für eine Analyse von Performance-Engpässen und sonstigen Problemen, die von den Anwendungen verursacht werden können, und ist mittlerweile für viele eine unverzichtbare Basis für die Verwaltung von Oracle-Datenbanken geworden. In 12*c* Release 1 wird das AWR ausschließlich auf der Ebene der CDB gespeichert und konfiguriert. Bei einem Transport einer PDB von einer CDB zur anderen CDB (Unplug und Plug) werden die AWR-Daten nicht mitgenommen.

Um Multitenant als Konsolidierungsplattform noch weiter zu verbessern, können ab 12.2 die AWR-Daten auch lokal (also zusätzlich) in einer PDB gespeichert sein. Das gilt natürlich nur für die AWR-Daten, die für diese PDB relevant sind. Damit können sie bei einem Umzug einer PDB von einer CDB zu einer anderen mit-

genommen werden. Zusätzlich sind die Applikations-Owner in der Lage, eigene AWR-Snapshots durchzuführen, unabhängig von den Einstellungen der CDB.

Per Default ist die automatische Erstellung von Snapshots für PDBs deaktiviert; die Aktivierung erfolgt in der jeweiligen PDB per „ALTER SYSTEM SET AWR_PDB_AUTOFLUSH_ENABLED=TRUE;". Danach setzt man eine Zeit für das Erfassungsintervall (in Minuten), also zum Beispiel „EXECUTE DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS(interval => 60,retention => 20160)". Dieser Schritt ist notwendig, da sonst PDB-seitig keine AWR-Snapshots erstellt werden, der Default für das Intervall also 0 ist.

Man kann diese Einstellungen auch in Oracle Enterprise Manager Cloud Control vornehmen, sobald das Plug-in der Version 13.2.2 für Oracle-Datenbanken installiert ist. AWR-Snapshots lassen sich auch manuell (wie gewohnt mit dem PL/SQL-Package „DBMS_WORKLOAD_REPOSITORY") erstellen, wobei die Vorgehensweise sich auf CDB- oder PDB-Ebene nicht unterscheidet. Einzig entscheidend ist, an welcher Datenbank man angemeldet ist.

Jede PDB führt eigene sogenannte „Snapshots" aus, entweder manuell oder über ein regelmäßiges Intervall. Die Kommandos dazu unterscheiden sich nicht von dem bislang gewohnten Umgang mit dem AWR, nur dass sie in einer Datenbank-Sitzung in einer PDB ausgeführt werden. Bei einem Transport einer PDB über Unplug und Plug beziehungsweise PDB Relocate werden die AWR-Daten also mitgenommen. Die Einstellungen lassen sich auch in EM Cloud Control 13.2 vornehmen *(siehe Abbildung 2)*.

Über PDB Lockdown Profiles lässt sich mit „ALTER LOCKDOWN PROFILE profile_name DISABLE FEATURE=('AWR_ACCESS');" die Erstellung von AWR-Snapshots auf PDB-Ebene verhindern. Die Nutzung der lokalen AWR-Snapshots unterscheidet sich nicht von denen auf CDB-Ebene.

Lokaler Undo Tablespace

Undo-Daten, die zum Beispiel für ein Rollback einer nicht beendeten Transaktion verwendet werden, sind in einem Undo Tablespace gespeichert. Für Multitenant bedeutet das mit 12*c* Release 1, dass dieser auf der Ebene der CDB existiert, „Shared Undo Mode" genannt. Diese Daten werden bei einem Transport einer PDB von einer CDB zu einer anderen nicht mitgenommen. Da die Undo-Daten auch für die Funktionalität des Flashbacks erforderlich sind, ist ein Flashback nach einem Transport also nicht möglich.

12*c* Release 2 bietet als Alternative den „Local Undo Mode". Dabei bekommt jede PDB einen eigenen Undo Tablespace. Dadurch werden auch die Undo-Daten bei einem Transport mitgenommen und alle Funktionalitäten, die auf diesen Undo-Daten beruhen, können weiterhin verwendet werden. Auch in 12.2 ist der „Shared Undo Mode" Standard. Die Einstellung des „Local Undo Mode" erfolgt auf CDB-Ebene, entweder beim Anlegen einer CDB mit „CREATE DATABASE newcdb ... LOCAL UNDO ON" oder durch nachträgliches Einschalten des Modus *(siehe Listing 3)*.

Danach lässt sich noch der Undo Tablespace in der Seed-Datenbank manuell erstellen, um die Größe dieses Tablespace für die PDBs vorzugeben, die über die Seed-Datenbank erstellt werden.

```
SHUTDOWN
STARTUP UPGRADE
ALTER DATABASE LOCAL UNDO ON;
SHUTDOWN
STARTUP
```

Listing 3

Applikationscontainer

Wie beschrieben, geht es bei Multitenant unter anderem darum, dass Ressourcen gemeinsam genutzt werden. Das beginnt beim Data Dictionary, dessen Standardbereich zentral in der CDB gespeichert ist. Das Data Dictionary ist das Datenmodell der Oracle-Datenbank und man kann es mit einem Datenmodell von Anwendungen gut vergleichen. Also kann man dieses Konzept der einmaligen, zentralen Speicherung auch auf der Ebene der Anwendung anwenden. Es gibt für jede Anwendung:

- Das Datenmodell, also die Definition von Datenbank-Objekten
- Statische Daten wie zum Beispiel einen Katalog von Postleitzahlen
- Dynamische Daten wie zum Beispiel Buchungsdaten

Werden für eine Anwendung mehrere Datenbanken betrieben, sei es dass diese für verschiedene Mandanten oder für verschiedene Zustände im Lebenszyklus

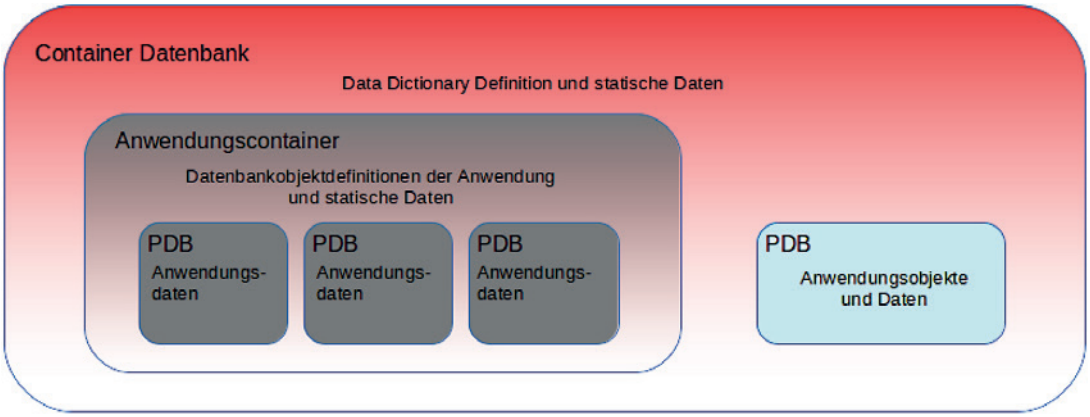


Abbildung 3: Der Applikationscontainer

```
ALTER PLUGGABLE DATABASE APPLICATION mapp BEGIN INSTALL '1.0';
CREATE USER wdg IDENTIFIED BY wdg CONTAINER=ALL;
GRANT CREATE SESSION, DBA TO wdg;
CONNECT wdg/wdg@APPCONTAINER
CREATE TABLE wdg.plz_verzeichnis SHARING=DATA
      (PLZ number,ort varchar2(100));
INSERT INTO wdg.plz_verzeichnis VALUES (59556 , 'Lippstadt');
COMMIT;
ALTER PLUGGABLE DATABASE APPLICATION mapp END INSTALL '1.0';
```

Listing 4

```
ALTER PLUGGABLE DATABASE APPLICATION mapp
  BEGIN UPGRADE '1.0' to '1.1';
…
ALTER PLUGGABLE DATABASE APPLICATION mapp
  END UPGRADE '1.0' to '1.1';
```

Listing 5

(Entwicklung, Test, Produktion) erforderlich sind, stellt sich schnell die Frage, ob man nicht wie beim Data Dictionary eine einmalige, zentrale Speicherung all dessen durchführen kann, was in allen PDBs gleich genutzt wird. Genau für dieses Ziel gibt es den Applikationscontainer. *Abbildung 3* zeigt diesen Zusammenhang.

Da eine CDB (je nach Art der Nutzung) bis zu 4.096 PDBs betreiben kann, werden in aller Regel auch mehrere verschiedene Anwendungen in mehreren PDBs betrieben. Daher sind mehrere Applikationscontainer in einer CDB möglich. Jeder Applikationscontainer ist sowohl eine PDB (in der Haupt-CDB) als auch CDB (für die Anwendungs-PDBs). Auch für den Applikationscontainer kann eine Seed-Datenbank vorbereitet werden, um die Erstellung neuer PDBs zu beschleunigen.

Von daher soll eine Anwendung in Datenbanken betrieben werden, die nur die Anwendungsdaten enthält, jedoch keine Anwendungsobjekte wie zum Beispiel Tabellen-Definitionen, PL/SQL-Prozeduren (-Funktionen, -Packages) etc. Auch statische Daten wie zum Beispiel Kataloge von Postleitzahlen oder Vorwahlnummern sollen nicht in den einzelnen Anwendungsdatenbanken gespeichert sein, sondern nur und einmalig in dem Applikationscontainer, in dem die Anwendungsdatenbanken betrieben werden.

Im Applikationscontainer werden die zentral gespeicherten Definitionen der Anwendungsobjekte (Tabellen, Indizes, Sequences etc.) versioniert erstellt, sodass die Anwendungsdatenbanken zentral gesteuert mit einer neuen Version versorgt werden können. Dabei kann zunächst eine neue Version erstellt werden; erst wenn diese in der Anwendungsdatenbank verwendet werden soll, wird die neue Version aktiviert.

Ein Applikationscontainer wird mit „“CREATE PLUGGABLE DATABASE myapp_ac AS APPLICATION CONTAINER…;” erstellt. Angemeldet an dem neuen Applikationscontainer, werden dann sowohl die Seed-Datenbank also auch alle Anwendungsdatenbanken so erstellt, wie man es auch in einer normalen CDB ausführen würde. In diesem Fall verhält sich der Applikationscontainer also wie eine CDB. Um nun die Anwendungsobjekte im Applikationscontainer zu erstellen, muss dies in einem speziellen Rahmen erfolgen *(siehe Listing 4)*.

Auf diese Weise ist ein Datenmodell entstanden, das zentral im Applikationscontainer gespeichert ist und nun für die Anwendungs-PDBs verfügbar gemacht wird, indem mit „ALTER PLUGGABLE DATABASE APPLICATION mapp SYNC;” ein SYNC-Vorgang gestartet wird. Wenn nun das Datenmodell verändert werden soll, wird eine neue Version erstellt *(siehe Listing 5)*. Mit einem erneuten SYNC bekommen die Anwendungs-PDBs die neue Version der Anwendung.

Neben der einmaligen Speicherung der Daten ist bei der Nutzung von Anwendungscontainern das Deployment von Anwendungsdatenbanken viel einfacher, da einfach nur eine neue PDB im Applikationscontainer erstellt werden muss. Alle Datenbank-Objekte, die für die Anwendung gebraucht werden, stehen sofort zur Verfügung. Auch das Einspielen neuer Versionen muss nur einmalig vorgenommen werden. Mit einem einzigen SYNC-Kommando kann die neue Version der Anwendungsobjekte auf die Anwendungsdatenbanken übertragen werden.

Fazit

Mit 12*c* Release 2 wird die neue Multitenant-Architektur erheblich erweitert, um noch besser als Konsolidierungs- beziehungsweise Virtualisierungsplattform genutzt werden zu können. Durch die flexible Nutzung von Ressourcen können bestehende Server noch besser ausgelastet werden.

Der Transport von PDBs zwischen verschiedenen Containern ist noch einfacher und mit minimaler Downtime möglich. Damit lassen sich Oracle-Datenbanken leicht in einer flexiblen, hybriden Cloud-Landschaft betreiben.

Ralf Durben
ralf.durben@oracle.com

Mehr **Sicherheit** bei weniger Arbeit

Norman Sibbing, ORACLE Deutschland B.V. & Co. KG

Oracle führt die Weiterentwicklung der Sicherheitsfeatures mit der neuen Datenbank 12*c* Release 2 kontinuierlich fort. Ein paar große, aber viele kleine interessante Neuerungen und Vereinfachungen sind hinzugekommen, darunter die komplette Verschlüsselung der Oracle-Datenbank, Erleichterungen beim Verschlüsseln bestehender Datenbanken, vereinfachte TDE-Schlüssel-Verwaltung sowie neue Rollen. Noch nie war die Oracle-Datenbank-Sicherheit so einfach.

Fully Encrypted Database

Bisher war es lediglich möglich, benutzerdefinierte Tablespaces verschlüsselt anzulegen. Oracle-definierte Tablespaces wie „SYSTEM", „SYSAUX", „TEMP" oder das „UNDO" konnten nicht verschlüsselt werden. Mit der Version 12*c* Release 2 lässt sich nun auch die Datenbank komplett verschlüsseln („Fully Encrypted Database"). Die Konsequenz ist, dass hochprivilegierte Benutzer wie ein Betriebssystem- oder ein Storage-Administrator im SYSTEM-Tablespace beziehungsweise in dessen Datenbank-Dateien durch eine einfache Strings-Suche (wie in *Listing 1* gezeigt) nach Data-Dictionary-Informationen suchen können, ohne sich an der Datenbank anmelden zu müssen.

Die Möglichkeit, jetzt alle Tablespaces inklusive der Oracle-definierten Tablespaces zu verschlüsseln, ist eine hervorragende Neuerung, wenn man bedenkt, dass im Data-Dictionary neben allen Benutzernamen auch die Hashwerte der Kennwörter zu finden sind. Die Verschlüsselung des temporären und des Undo-Tablespace kann als optional betrachtet werden, da dort die Datenblöcke, die aus bereits verschlüsselten Tablespaces stammen, weiterhin verschlüsselt bleiben. Es stellt sich nun die Frage, wie eine bestehende Datenbank nachträglich verschlüsselt werden kann. Hier bietet die Version 12*c* Release 2 mit TDE Live Conversion die perfekte Lösung.

TDE Live Conversion

Mit TDE Live Conversion stehen zwei Möglichkeiten (Online und Offline Conversion) zum nachträglichen Verschlüsseln der Datenbank zur Verfügung. Damit lassen sich auch bereits bestehende Datenbanken einfacher nachverschlüsseln. Bisher war es nur durch das Umkopieren von Daten aus einem nicht verschlüsselten in einen verschlüsselten Tablespace möglich, diese nachträglich zu verschlüsseln, was entweder durch Online Redefinition oder Data Pump Import/Export durchgeführt werden konnte. Für beide Varianten musste neben einer partiellen Datenbank-Downtime der doppelte Plattenplatz vorgehalten werden.

Online Conversion

Um einen Tablespace beziehungsweise dessen Datenbank-Datei(en) während des Betriebs, also ohne Ausfallzeiten, nachträglich zu verschlüsseln (Online Conversion), reicht jetzt ein einziger Datenbank-Befehl aus. Voraussetzung ist, dass der Datenbank-Parameter „COMPATIBLE" auf „12.2.0.0" eingestellt ist. Bei der Online Conversion wird intern zu jeder dem Tablespace zugeordneten Datenbank-Datei eine korrespondierende, neue verschlüsselte Datenbank-Datei angelegt. Danach werden jeweils die Inhalte der originalen Datenbank-Dateien im laufenden Betrieb in die verschlüsselten Datenbank-Dateien übertragen. Wenn alles erfolgreich war, werden die originalen Datenbank-Dateien automatisch gelöscht und somit der Platz wieder freigegeben. Der Tablespace wird nur mit einem Kommando verschlüsselt: „ALTER TABLESPACE DATA_TS ENCRYPTION ONLINE USING 'AES192' ENCRYPT FILE_NAME_CONVERT = ('data_ts.dbf', 'data_ts_enc.dbf')".

Offline Conversion

Alternativ dazu gibt es mit der Offline Conversion eine platzsparende Variante. Sie ermöglicht das Konvertieren eines nicht verschlüsselten in einen verschlüsselten Tablespace, ohne zusätzlichen Plattenplatz zu benötigen. Eine partielle Downtime der Applikation muss hierbei eingeplant werden, da die zu konvertierenden Tablespaces OFFLINE sein müssen. Die Konvertierung selber geschieht auf Datenbank-File-Ebene *(siehe Listing 2)*.

Für die Offline Conversion wurde jeweils ein Patch für die Datenbank-Versionen 11.2.0.4 und 12.1.0.2. zur Verfügung gestellt. Bei Verwendung des Bundle-Patches vom Januar 2017 ist diese Funktionalität bereits enthalten. Die Offline Conversion arbeitet momentan ausschließlich mit dem Advanced-Encryption-Standard-Algorithmus AES128. *Tabelle 1* gibt Antworten auf oft gestellte Fragen:

Neue Verschlüsselungs-Algorithmen

Das Release 2 der Datenbank 12*c* stellt drei neue Verschlüsselungs-Algorithmen zur Verfügung:

- SEED (Korea Information Security Agency) für Südkorea
- ARIA (Academia, Research Institute, and Agency) für Südkorea
- GOST (GOsudarstvennyy STandart) für Russland

```
strings system01.dbf | grep -E -o "S:\w+;T:\w+"

S:073B974663C7733814745CABBDCFE597721B722F34EE5803D9C2734B75BC;T:9793EE
38331120623CC830D95364EF9F48D65921744BA528123DB7BC29EA68B7054B7DE21B72A
9E455404F504AABBE28125FB8EE905EDA785D8FC7B2099C5E66195DE880E88984776778
A2A8A8F303C5
```

Listing 1

```
ALTER TABLESPACE users OFFLINE NORMAL;
ALTER DATABASE DATAFILE '…/users01.dbf' ENCRYPT;
ALTER TABLESPACE users ONLINE;
```

Listing 1

Funktionalität	Offline Conversion	Online Conversion
Wann kann ich konvertieren?	Offline-Tablespaces oder bei Datenbanken im Mount-Status	Online-Tablespaces und bei geöffneter Datenbank im „read write mode"
Muss ich eine Ausfallzeit planen?	Vorübergehendes TS-Offline-Setzen erforderlich Alternative: Konvertierung mit Data Guard (Doc ID 2148746.1)	Nein, verschlüsselt Tablespace im Hintergrund ohne Ausfallzeiten
Brauche ich zusätzlichen Plattenplatz?	Nein	Ja, Speicher-Overhead beträgt das Zweifache der größten Tablespace-Datei
Kann ich dies parallel ausführen?	Ja, gleichzeitige Verschlüsselung mehrerer Datendateien möglich	Ja, auf Tablespace-Ebene mit mehreren Sessions
Kann ich dabei ein Re-Keying durchführen?	Nein	Ja, unterstützt Live-Re-Keying von Tablespaces
Backport für vorherige Datenbank-Versionen?	Releases 12.1.0.2 und 11.2.0.4	Nein (nur Datenbank 12c Release 2)

Tabelle 1:Online- vs. Offline-Tablespace-Konvertierung

Damit stehen mit SEED, ARIA, GOST, AES und DES nun fünf Verschlüsselungs-Algorithmen zur Verfügung.

ENCRYPT_NEW_TABLESPACES

Damit beim Anlegen eines neuen Tablespace nicht vergessen wird, die Verschlüsselung zu nutzen, gibt es einen neuen Datenbank-Parameter „ENCRYPT_NEW_TABLESPACES". Er sorgt dafür, dass − bei entsprechender Einstellung − alle neuen Tablespaces ohne zusätzliche Storage-Klausel automatisch verschlüsselt angelegt sind. Es bestehen drei Einstellungsmöglichkeiten:

- Bei „CLOUD_ONLY" erkennt die Datenbank automatisch, dass sie sich in der Oracle Public Cloud befindet, und verschlüsselt neue Tablespaces mit „AES 128" ohne explizite Encryption-Storage-Klausel.
- „ALWAYS" macht da keine Unterschiede. Hier ist es egal, ob sich die Datenbank in der Oracle Public Cloud befindet oder nicht. Neue Tablespaces werden mit „AES 128" verschlüsselt.
- „DDL" stellt das bisherige Verhalten sicher, die explizite Angabe einer Encryption-Storage-Klausel ist also notwendig, wenn verschlüsselt werden soll.

KEYSTORE IDENTIFIED BY

Um eine verschlüsselte Pluggable Database zu klonen, musste bisher der zugehörige TDE Master Key aus dem Keystore der Quell-Container-Datenbank in eine PKCS12-Datei exportiert werden, um diesen dann wieder vorab in den Keystore der Ziel-Container-Datenbank zu importieren. Dieser doch eher umständliche Weg ist ab dem Release 2 nicht mehr notwendig. Das Kommando „CREATE PLUGGABLE DATABASE" wurde durch die Klausel „KEYSTORE IDENTIFIED BY "Keystore Kennwort"" erweitert. Sie ermöglicht das automatische Exportieren und Importieren des TDE Master Key aus dem Quell- in den Ziel-Keystore, ohne weitere Zwischenschritte *(siehe Listing 3)*.

EXTERNAL STORE

Bisher musste der Datenbank-Administrator das Kennwort des Keystores für alle Schlüssel-Operationen kennen. Dies ist zumindest aus Sicherheitsgründen ungünstig, manchmal aber auch einfach nur lästig. Abhilfe schafft eine neue Funktionalität namens „EXTERNAL STORE", ein separater PKCS12-Keystore, in dem das Kennwort der Datenbank-Keystores gespeichert wird − ähnlich wie bei dem seit langer Zeit bekannten und bei Kunden häufig eingesetzten Oracle Secure External Password Store (SEPS). Dadurch wird aus „KEYSTORE IDENTIFIED BY "Kennwort"" ein „KEYSTORE IDENTIFIED BY EXTERNAL STORE". Voraussetzung ist die Verwendung eines neuen Datenbank-Parameters namens „EXTERNAL_KEYSTORE_CREDENTIAL_LOCATION". Nur drei Schritte sind dazu notwendig:

- Verzeichnis für den External-Store anlegen

```
create pluggable database pdb1_clone from pdb1@dblink_cdb1_pdb1 file_
name_convert=('…/cdb1/pdb1', '…/cdb2/pdb1_clone') KEYSTORE IDENTIFIED
BY "welcome1";
```

Listing 3

```
ALTER SYSTEM SET external_keystore_credential_location='external store
file-location' SCOPE=SPFILE;
```

Listing 4

```
ADMINISTER KEY MANAGEMENT ADD SECRET 'Kennwort'
FOR CLIENT 'TDE_WALLET' TO LOCAL AUTO_LOGIN
KEYSTORE 'external store file-location';
```

Listing 5

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
IDENTIFIED BY EXTERNAL STORE;
ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'cdb1:root:v.02'
IDENTIFIED BY EXTERNAL STORE WITH BACKUP USING 'backup:v.01'
CONTAINER = CURRENT;
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE
IDENTIFIED BY EXTERNAL STORE;
```

Listing 6

```
ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'cdb1:root:v.03' FORCE KEY-
STORE IDENTIFIED BY EXTERNAL STORE
WITH BACKUP CONTAINER = CURRENT;
```

Listing 7

```
AUDIT POLICY pci_dss BY USERS
WITH GRANTED ROLES modify_ccards, read_ccards;
```

Listing 8

- Datenbank-Parameter setzen und Datenbank durchstarten *(siehe Listing 4)*
- External-Store mit dem Schlüsselwort „TDE_WALLET“ und dem Keystore-Kennwort anlegen *(siehe Listing 5)*

Ab jetzt braucht der Datenbank-Administrator kein Kennwort zur Verwaltung des Master Key mehr zu kennen. Somit sind zum Beispiel vereinfachte Kommandos möglich *(siehe Listing 6)*.

FORCE KEYSTORE

Bisher konnten Operationen wie zum Beispiel das Erneuern eines Master Key nur durchgeführt werden, wenn kein Auto-Login-Wallet zum Einsatz kam. Die bisherige Vorgehensweise in diesem Fall war doch eher umständlich, wie folgendes Beispiel zeigt:

- Löschen des Auto-Login-Wallet „cwallet.sso“
- Schließen des Auto-Login-Wallet mit „ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE;“
- Öffnen des Standard-Wallet mit „ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN“ und "IDENTIFIED BY "Kennwort";"

Nach den Key-Management-Aktivitäten musste das Auto-Login-Wallet wieder erstellt werden. In der Version 12.2 wurde auch hieran gedacht. Die Klausel „FORCE KEYSTORE“ ermöglicht ein Key-Management, bei gleichzeitiger Nutzung des Auto-Login-Wallet *(siehe Listing 7)*. Dieses Beispiel zeigt ein Re-Keying ohne Angabe eines Kennworts bei gleichzeitiger Nutzung eines Auto-Login Wallet. Das Ganze funktioniert mit nur einem einzigen Befehl.

Durch all diese Neuigkeiten aus dem Bereich der Datenbank-Verschlüsselung mittels Oracle Transparent Data Encryption (TDE) wird die Verwaltung einer verschlüsselten Oracle-Datenbank nicht nur einfacher, sondern auch wesentlich sicherer.

Neue Rolle „SYSRAC“

Lange Zeit war die Verwendung der „SYSDBA“-Rolle für viele Datenbank-Operationen und -Funktionalitäten zwingend erforderlich. Eine strikte Trennung der Rollen gemäß „principle of least privilege“ (PoLP) wurde zwar auch für administrative Tätigkeiten gefordert, konnte aber bis zur Version 12*c* Release 1 für bestimmte Operationen und Funktionalitäten nicht umgesetzt werden. Neben den im Release 1 eingeführten Rollen zur Funktionstrennung für das Key-Management („SYSKM“), für das Backup („SYSBACKUP“) und für Data-Guard-Operationen („SYSDG“) gibt es nun auch eine auf die notwendigen Privilegien reduzierte Rolle „SYSRAC“. Diese kann nun für den Betrieb eines Real Application Cluster (RAC) genutzt werden. Sie stellt die Möglichkeit für Funktionstrennungen zur Verfügung. „SYSDBA“ darf allerdings noch alles, es sei denn, Database Vault wird eingesetzt.

Rollen als Auditing-Bedingung

Die in 12*c* Release 1 eingeführte Möglichkeit, Unified Audit Policies für bestimmte Benutzer zu aktivieren beziehungsweise zu deaktivieren, ist mit dem Release 2 um die Möglichkeit erweitert worden, als Auditing-Bedingung zusätzlich Datenbank-Rollen zu verwenden. Die Unified Audit Policy ist also nur bei Datenbank-Benutzern aktiviert, die entsprechende Rollen haben. Hiermit lassen sich wunderbar verschiedene Auditing-Gruppen, zum Beispiel je nach Applikations-Art, abbilden und zielgerichtet zuweisen *(siehe Listing 8)*.

Erfassung von VPD-Prädikaten im Unified Audit Trail

Bis zum Release 1 wurden die verwendeten Prädikate (SQL-Where-Bedingungen)

```
SELECT SQL_TEXT, RLS_PREDICATE FROM TABLE (DBMS_AUDIT_UTIL.DECODE_RLS_INFO_ATRAIL_UNI (CURSOR (SELECT * FROM
UNIFIED_AUDIT_TRAIL WHERE OBJECT_NAME='EMPLOYEES' )));

SQL_TEXT                       RLS_PREDICATE
-----------------------------  ------------------------------------------------------
select email from employees    SYS_CONTEXT('USERENV','IP_ADDRESS')!='10.200.110.205'
```

Listing 9

bei Nutzung der Virtual Private Database (VPD) nicht erfasst. Dies ließ demzufolge keine Aussage darüber zu, welche Bedingungen ein SQL-Statement letztendlich verwendete. Zu diesem Zweck wurde eine neue Spalte (RLS_INFO) im Unified Audit Trail hinzugefügt. Diese Spalte kann komfortabel mithilfe eines neuen Packages (DBMS_AUDIT_UTIL) ausgelesen werden *(siehe Listing 9)*.

Passwort-Profile

Inaktive Datenbank-Benutzer können jetzt nach Ablauf einer vorgegebenen Zeit durch das Setzen des Profile-Parameters „INACTIVE_ACCOUNT_TIME" im Passwort-Profil automatisch gesperrt werden. Somit lassen sich Datenbank-Benutzer-Leichen leichter identifizieren und unschädlich machen. Ebenfalls gelten die unten aufgeführten Einschränkungen im Passwort-Profil nun auch für SYSDBA & Co.:

- FAILED_LOGIN_COUNT
- PASSWORD_LOCK_TIME
- PASSWORD_GRACE_TIME
- PASSWORD_LIFE_TIME
- PASSWORD_VERIFY_FUNCTION-Klausel

Passwort-Datei

Die Passwort-Datei galt lange als Sicherheits-Risiko. Einmal erstellt, konnte ein Angreifer, solange er wollte, versuchen, sich als „SYSDBA" an der Datenbank anzumelden, ohne gesperrt zu werden. Zudem wurde die Komplexität des Passwortes nicht überprüft. Mit der Version 12.2 gehört das der Vergangenheit an. Wird die Passwort-Datei im Format „12.2" erstellt, gelten auch dort folgende Einschränkungen *(siehe Listing 10)*:

- Enthält mindestens acht Zeichen, davon mindestens ein numerisches und ein alphanumerisches Zeichen
- Ungleich Benutzername und ungleich Datenbankname
- Enthält mindestens ein Sonderzeichen
- Darf keine doppelten Anführungszeichen beinhalten

Folgende Passwort-Eigenschaften der Passwort-Datei werden aus dem „Default"-Passwort-Profil der Datenbank übernommen. Die Standardwerte sind:

- PASSWORD_LIFE_TIME: 180 Tage
- PASSWORD_GRACE_TIME: 7 Tage
- FAILED_LOGIN_ATTEMPTS: 10 Versuche

Nach zehn Fehlversuchen bei der Anmeldung erscheint dann auch die dem „SYSDBA" bekannte Meldung „ORA-28000: the account is locked" *(siehe Listing 11)*.

Weitere neue Eigenschaften der Passwort-Datei sind:

- Die Passwort-Datei bietet jetzt die Möglichkeit, anstelle einer Passwort-Authentifizierung eine SSL- beziehungsweise Kerberos-Authentifizierung zu nutzen *(siehe Listing 12)*
- Beim „ORAPWD" wurde das Argument „IGNORECASE" entfernt: Bei allen Passwörtern ist standardmäßig zwischen Groß- und Kleinschreibung zu unterscheiden
- Data Guard: Automatische Verteilung der Passwort-Datei nach Passwort-Änderung auf der Primär-Datenbank zu allen Standby-Datenbanken

```
orapwd FILE='orapwcdb3' force='y' password="welcome"
OPW-00029: Password complexity failed for SYS user : Password must contain at least 8 characters.

orapwd FILE='orapwcdb3' force='y' password="oracle_1"
OPW-00029: Password complexity failed for SYS user : Password must not be too simple.

orapwd FILE='orapwcdb3' force='y' password="welcome1"
OPW-00029: Password complexity failed for SYS user : Password must contain at least 1 special character.

orapwd FILE='orapwcdb3' force='y' password="welcome#A"
OPW-00029: Password complexity failed for SYS user : Password must contain at least 1 digit.
```

Listing 10

```
sqlplus sys as sysdba
SQL*Plus: Release 12.2.0.1.0 Production on Wed Apr 26 10:12:44 2017
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Enter password:
ERROR:
ORA-28000: the account is locked
```

Listing 11

```
orapwd FILE='+DATA/orcl/orapworcl' DBUNIQUENAME='orcl' FORMAT=12.2
sys=external('KerberosUserSYS@example.com') syskm=external('KerberosUserSYSKM@example.com')
```

Listing 12

Fazit

Die neue Datenbank-Version 12.2 bietet hervorragende Neuerungen bei der Datenbank-Sicherheit. Neben den beschriebenen existieren noch wesentlich mehr. Ein Gesamtüberblick ist in der Oracle Database 12.2 Dokumentation im Bereich „Oracle Database 12*c* Release 2 (12.2) New Features" zu finden.

Norman Sibbing
norman.sibbing@oracle.com

Neuigkeiten in der Datenbank-Entwicklung im Oracle Database 12c Release 2

Ulrike Schwinn, ORACLE Deutschland B.V. & Co. KG

Die Codierung von Datenbank-Anwendungen zu erleichtern, noch mehr Funktionalität zu bieten und dabei den Performance-Ansprüchen zu genügen, sind wichtige Ziele bei der Weiterentwicklung der Funktionen im Datenbank-Umfeld. Schlagworte wie „Case-insensitive Datenbank", „Verhalten im Fehlertoleranzfall" sowie das Tunen und Monitoren von neuem und existierendem SQL- und PL/SQL-Code spielen dabei eine wichtige Rolle. Eine der Stärken von Oracle innerhalb der Datenbank ist es, komplexe Datenstrukturen wie zum Beispiel XML, JSON oder Spatial effizient zugänglich zu machen. Die Weiterentwicklung der verfügbaren JSON-Funktionen stellt daher einen wichtigen Aspekt im Release 2 dar.

So viel vorweg: In vielen Bereichen ergänzt und vervollständigt Release 2 den Funktionsumfang von Release 1. Das ist auch typisch für das zweite Release einer Datenbank-Version. So profitieren die in Release 1 eingeführten Technologien wie Multitenant und In-Memory besonders von den Erweiterungen. Aber auch in der Datenbank-Entwicklung zeigt sich ein deutlicher Zuwachs an Features, die mehr Funktionalität, höhere Performance und eine hohe Fehlertoleranz versprechen.

Dieser Artikel zeigt punktuell Neuerungen an Beispielen. Da nicht alle neuen Funktionen erläutert werden können, finden sich im letzten Abschnitt einige Verweise auf interessante Publikationen, die dabei helfen, einen vollständigen Überblick über die Themen zu erhalten.

Basis-Funktionen

Zuerst eine wichtige Neuigkeit für alle Datenbank-Anwender: Die maximale Länge von Bezeichnern ist von 30 auf 128 Bytes erhöht worden. Namen für Tabellen, Objekte, Prozeduren, Variablen etc. können damit länger und somit sprechender sein; auch Migrationen von Fremd-Datenbanken werden damit leichter möglich.

Ein weiteres Ziel bei der Entwicklung von Features ist die Möglichkeit, mehr Kontrolle im Fehlerfall zu haben. Speziell bei den Konvertierungsfunktionen wie „TO_NUMBER", „TO_DATE", „TO_TIMESTAMP" etc. kann nun eine Fehlerausgabe vermieden und stattdessen ein Defaultwert ausgegeben werden. Ein einfaches Beispiel mit der „TO_NUMBER"-Funktion zeigt das neue Verhalten: Wird als Eingabe-Parameter eine „DATE"-Spalte mitge-

geben, erhält man vor Release 12.2 den Fehler „ORA-01722: invalid number“. Jetzt kann man diese Ausgabe mit einem Defaultwert umgehen *(siehe Listing 1)*.

Mit der neuen Funktion „VALIDATE_CONVERSION“ ist es darüber hinaus möglich zu überprüfen, ob ein Ausdruck in einen speziellen Datentyp konvertiert werden kann. Der Wert „1“ bedeutet, dass eine erfolgreiche Konvertierung möglich ist, „0“ hingegen, dass die Konvertierung fehlschlagen wird. Mögliche Datentypen sind dabei die numerischen und die Datums-Datentypen wie „DATE“, „NUMBER“, „TIMESTAMP“, „BINARY_DOUBLE“, „INTERVAL YEAR TO MONTH“ etc.

Einige weitere interessante Funktionserweiterungen finden sich im analytischen Window-Funktionen-Umfeld. „LISTAGG“, das eine String-Aggregation vornimmt, bietet nun eine Ausgabemöglichkeit bei Überschreitung der maximalen Ausgabelänge an. Statt eines „ORA-01489“-Fehlers kann nun eine Konstante ausgegeben werden. Dies ist auch mit einem Zähler kombinierbar, der die Anzahl der überzähligen Zeichen ausgibt *(siehe Listing 2)*.

Mit Oracle Database 12*c* Release 1 ist die Window-Funktion „APPROX_COUNT_DISTINCT“ eingeführt worden. Die Idee dabei war, die Ausführung von „COUNT (DISTINCT)“ zur Berechnung der Anzahl von verschiedenen Spalten-Ausprägungen zu beschleunigen. „APPROX_COUNT_DISTINCT“ führt ein Sampling durch, um die ungefähre Anzahl der verschiedenen Einträge einer Spalte zu berechnen, und ist in der Regel um ein Vielfaches schneller als „COUNT (DISTINCT)“. Der Wermutstropfen dabei war, dass man dazu die Funktionen im eigenen Code austauschen musste. Aus diesem Grunde sind nun im Release 2 zusätzliche Session-Parameter eingeführt worden. Beispielsweise führt das einfache Setzen des Session-Parameters „APPROX_FOR_COUNT_DISTINCT“ auf „TRUE“ im Hintergrund zu einem „APPROX_COUNT_DISTINCT“ ohne Änderung am Code.

Auch die Einführung neuer Strukturen, der sogenannten „analytischen Views“, liefert eine interessante Erweiterung. Diese werden wie reguläre Views angelegt und ermöglichen es, mit einfachen SQL-Abfragen komplexe analytische Operationen auf relationalen Strukturen durchzuführen. Sie können auf regulären Tabellen, External Tables sowie In-Memory-Spalten basieren und berücksichtigen Daten-Hierarchien, Berechnungen und Aggregationen. Damit werden nun Konzepte von Business-Intelligence-Applikationen in die Datenbank verlegt. Nutzt man SQL Developer ab der Version 4.2, kann man sogar von einer speziellen Unterstützung für analytische Views über Wizards und Templates profitieren.

```
SQL> select to_number(hiredate default 42 on conversion error) ausgabe
  from emp where rownum=1;

  AUSGABE
----------
      42
```

Listing 1

```
SQL> set pagesize 100000
SQL> select listagg(s, ';' on overflow truncate '...' with count)
  within group (order by r) ausgabe
  from (select rownum r , 'x' s from dual connect by level <20000);

AUSGABE
--------------------------------------------------------------------------------
x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x
;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;
x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x
;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;
x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;x;...(3631)
```

Listing 2

```
create table tab_ci
  (n number,
  wort varchar2(100) collate binary_ci)
```

Listing 3

```
SQL> select * from tab_ci where wort='Klein'
  N   WORT
-----------  ----------
  1   klein
  2   KLEIN
  3   Klein
```

Listing 4

```
select cust_city,
  json_objectagg('name' is cust_last_name) cust_per_city
from customers
where cust_city like 'Co%'
group by cust_city
```

Listing 5

```
create search index JSON_TAB_GUIDE on JSON_TAB (JSON_DOCUMENT) for json
parameters ('DATAGUIDE ON SYNC (ON COMMIT)')
```

Listing 6

Interessant für viele Datenbank-Entwickler ist sicherlich das Feature „Case Insensitivity by Default“. Damit besteht die Möglichkeit, eine Sortier-Reihenfolge (engl. „collation“) und Case-Insensitivität für Spalten beim „CREATE TABLE“ festzulegen – ähnlich wie bei der Festlegung auf einen Datentyp. Das Ganze geht so weit, dass ein normaler Index sogar zum passenden Function-Based Index umfunktioniert wird. Alles erfolgt automatisch, ohne dass man dies bei der Programmierung einer Applikation beachten muss. Ein Beispiel zeigt die einfache Vorgehensweise: Die Tabelle „TAB_CI“ besitzt eine Case-insensitive Spalte „WORT“. In 12.2 kann die Syntax dann wie in *Listing 3* aussehen. Zu beachten ist der Zusatz „collate binary_ci“ in der Spalten-Definition.

Jetzt fügen wir in die Spalte „WORT“ einige unterschiedlich geschriebene Werte ein wie „klein“, „Klein“ und „KLEIN“. Beim Selektieren muss man nun keinerlei Rücksicht auf die Groß- und Kleinschreibung nehmen, wie *Listing 4* zeigt.

Auf diese Weise kann man Groß- und Kleinschreibungen in Spalten einer Applikation vernachlässigen, ohne dass ein Datenbank-Entwickler explizit Funktionen wie „UPPER“ oder „LOWER“ hinzufügen muss. Diese Eigenschaft kann übrigens schon vorab in der Definition des Users mitgegeben werden, sodass eine Default-Einstellung beim Anlegen von Tabellen implementiert ist. Der User kann dann die Standard-Syntax beim Anlegen von Tabellen ohne Erweiterung verwenden und ohne Zutun die Case-Insensitivität automatisch nutzen.

Dieses Feature existiert natürlich nicht nur für die Groß- und Kleinschreibung, sondern auch für die Werte von „NLS_SORT“ wie „GERMAN“, „XGERMAN“, „XGERMAN_CI“, „FRENCH“, „FRENCH_M_AI“ etc. Bevor man startet, muss man allerdings wissen, dass das Feature nur in Zusammenhang mit dem Wert „EXTENDED“ bei „MAX_STRING_SIZE“ funktioniert; der Defaultwert „STANDARD“ ist nicht ausreichend.

Gerade beim Tunen und Analysieren von PL/SQL-Code stellt man sich auch häufig die Frage, ob und welche SQL-Statements im PL/SQL-Code verwendet wurden. Vor dem Release 2 war die Lösung dieser Fragestellung eher ein umständliches Unterfangen oder gar ein voneinander getrennter Vorgang. Jetzt ist es einfach und intuitiv möglich, verwendete SQL-Statements in den PL/SQL-Programmen zu finden. Der Hierarchical Profiler, der das Standard-Werkzeug für das Monitoren von langsam laufendem PL/SQL-Code darstellt, sowie auch PL/SCOPE beinhalten ab dem Release 2 automatisch Informationen über den SQL-Text beziehungsweise die SQL-ID. Damit ist es einfach möglich, ein umfassendes SQL Tuning durchzuführen.

JSON-Funktionalität in der Datenbank

Daten im JavaScript-Object-Notation-Format (JSON) werden häufig in NoSQL- oder in anderen speziellen Datenbanken gespeichert. Diese erlauben zwar die Speicherung und den Zugriff der Daten, weisen aber kein vergleichbares Konsistenz- beziehungsweise Transaktionsmodell oder andere Standard-Funktionalitäten von relationalen Datenbanken auf.

Ab Oracle Database 12*c* Release 1 (12.1.0.2) gab es auch die Möglichkeit, auf JSON-Daten mit Standard-Datenbankmitteln zuzugreifen. JSON in der Datenbank ist dabei ein integraler Bestandteil der Oracle-Datenbank und in allen Ausprägungen der Datenbank wie zum Beispiel RAC, Single Instanz, Non CDB, CDB etc. nutzbar. Die Schnittstellen sind ganz einfach zu bedienen; es sind keine besonderen Voraussetzungen oder Privilegien erforderlich, um JSON-Funktionen zu verwenden. Die Idee dahinter ist, nicht nur einen einfachen Textstring zu speichern und auf diesen zuzugreifen, was schon immer in jedem Release möglich war, sondern auch spezielle JSON-Pfad-Zugriffe oder JSON-Validierungen zu ermöglichen, um nur einige Features zu nennen.

Im Release 2 gibt es darüber hinaus auch Lösungen zu folgenden Fragestellungen: Wie kann man beispielsweise JSON mit Datenbankmitteln generieren? Wie lassen sich JSON-Inhalte in der Datenbank mit PL/SQL-Mitteln manipulieren? Wie kann man schnell und einfach relationale Views erzeugen? Wie lässt sich umfassend in JSON suchen? Die Einführung neuer Objekt-Typen und weiterer JSON/SQL-Funktionen sowie das neue Konstrukt „Data Guide“ helfen bei der Lösung dieser Aufgaben.

Ein Beispiel zeigt, wie man JSON aus bestehenden relationalen Daten generieren und sogar aggregieren kann. Es sollen

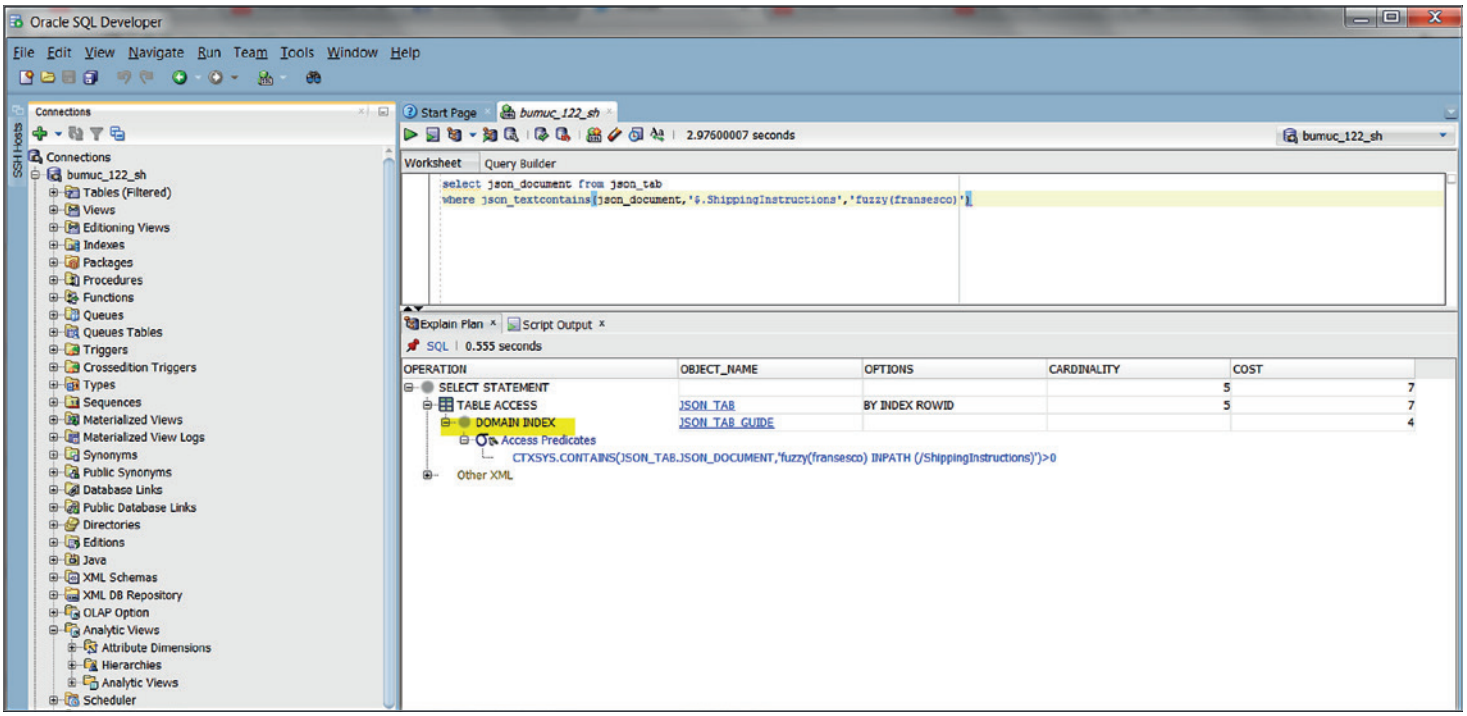


Abbildung 1: Ausführungsplan im SQL Developer

alle Kunden pro Stadt mit ihrem Nachnamen ausgegeben werden *(siehe Listing 5)*.

Hat man mit großen und unübersichtlichen JSON-Dokumenten zu tun, stellt sich schnell die Frage, wie man darin performant nach einem Wert oder Eintrag suchen kann. Die Basisfunktionen im Release 1 liefern schon erste Möglichkeiten, diese Anforderungen zu lösen. Allerdings ist die Handhabung etwas umständlich.

Eine Vereinfachung und weitere Funktionen verspricht ein neues Konzept im Release 2. Die Metadaten eines JSON-Dokuments können damit im Data Dictionary gespeichert werden. Optional wird ein zugehöriger Index zur Volltextsuche zur Verfügung gestellt. Die neue Struktur hat den treffenden Namen „Data Guide“. Damit ist es nun möglich, schnell und einfach mit einem Befehl relationale Views zu erzeugen, eine performante Suche im Text zu gestalten und sogar automatisch einer View virtuelle Spalten hinzuzufügen.

Das Beispiel in *Listing 6* zeigt das einfache Anlegen der Data-Guide-Struktur. Damit können beispielsweise folgende Filter-Kriterien in der „WHERE“-Klausel Verwendung finden und die zugehörigen Ad-hoc-Abfragen mit dem automatisch generierten Domain-Index beantwortet werden *(siehe Listing 7)*. Im SQL Developer lässt sich dann nicht nur das Ergebnis, sondern auch der Ausführungsplan überprüfen *(siehe Abbildung 1)*.

```
json_textcontains(json_document,'$.ShippingInstructions','fuzzy(franse
sco)')
json_exists(json_document,'$.ShippingInstructions.Address.zipCode')
json_textcontains(json_document, '$.LineItems', '$tie')
```

Listing 7

```
SQL> set markup CSV on delimiter ";"
SQL> select * from dept;

"DEPTNO";"DNAME";"LOC"
10;"ACCOUNTING";"NEW YORK"
20;"RESEARCH";"DALLAS"
30;"SALES";"CHICAGO"
40;"OPERATIONS";"BOSTON"
```

Listing 8

Für hohe Performance-Anforderungen bei Zugriffen auf JSON-Dokumente ist im Release 2 zusätzlich zu den bestehenden In-Memory-Zugriffen ein spezielles optimiertes JSON-Dokument-Handling implementiert worden. Virtuelle Spalten, JSON-Daten im optimierten Format und Ausdrücke (Expressions) können somit im Column Store gehalten werden und sorgen für hohe Performance.

Neue Werkzeuge

Wie in jedem Release steht jetzt auch eine erweiterte Version von SQL*Plus zur Verfügung. Abgesehen von einigen neuen Features ist es wichtig zu wissen, dass es eine Änderung bei der Nutzung der user-definierten Profildatei „login.sql“ gibt: Aus Sicherheitsgründen findet diese im lokalen Verzeichnis keine Berücksichtigung mehr. Sie wird nur noch aus den Pfadverzeichnissen „ORACLE_PATH“ (Linux) bezie-

hungsweise „SQLPATH“ (Windows) gelesen. Diese Änderung ist bei der Migration der SQL*Plus-Skripte zu beachten.

Listet man die verschiedenen SQL*Plus-Optionen oder Systemvariablen mit „SQL*Plus –H“ oder mit „show all“ auf, kann man erkennen, dass einige neue Parameter und Optionen eingeführt worden sind. Zur Performance-Steigerung gibt es jetzt die neue Option „-F“, mit der SQL*Plus in einem speziellen Modus mit zusätzlichen Setup-Einstellungen gestartet wird. Damit werden beispielsweise verschiedene Caches aktiviert wie ein „Large Objects“-Prefetch „LOB-PREFETCH“ (Wert „16.384“), ein Zeilen-Prefetch „ROWPREFETCH“ (Wert „2“) und ein Statement-Caching „STATEMENTCACHE“ (Wert „20“). Diese Werte können natürlich auch unabhängig vom eingestellten Modus in der SQL*Plus-Session gesetzt beziehungsweise verändert werden.

Neu ist auch eine weitere Formatierungsmöglichkeit, nämlich die CSV-Formatierung. Das Beispiel in *Listing 8* zeigt die Verwendung: Die Tabelle „DEPT“ wird mit Semikolon-Delimiter im CSV-Format aufgelistet.

Außerdem kennt SQL*Plus jetzt endlich eine „history“-Funktion, die standardmäßig allerdings ausgeschaltet ist (Wert „off“). Damit lassen sich Kommandos aus der Historie anzeigen, editieren und löschen. Wie immer kann man sich eine detaillierte Funktionsbeschreibung mit „help“ ausgeben lassen.

An dieser Stelle soll nicht unerwähnt bleiben, dass die Werkzeuge SQL Developer und SQLcl natürlich ebenso mit der Software im Release 2 verfügbar sind. Da die Release-Stände dieser Werkzeuge allerdings unabhängig von der ausgelieferten Datenbank-Software sind, empfiehlt es sich stattdessen, die aktuellste Software von OTN zu laden und zu verwenden. So liefert die seit April verfügbare SQL-Developer-Version 4.2 interessante und wichtige Aspekte für das Monitoren und Tuning von Datenbank-Anwendungen und sollte unbedingt separat installiert werden.

Steht (noch) keine eigene Software-Installation zur Verfügung, kann man auch das Online-Werkzeug „Live SQL“ nutzen, das einen einfachen und kostenfreien Zugang auf die aktuelle Datenbank-Version 12*c* Release 2 ermöglicht. Damit lassen sich schnell und einfach Features der Oracle-Datenbank mit SQL und PL/SQL ausprobieren oder erlernen. Man benötigt lediglich einen Oracle-Technology-Network-Account (OTN) und einen Web-Browser.

Nach Eingabe der URL *„http://livesql.oracle.com“* steht ein SQL-Worksheet zur Eingabe von Kommandos zur Verfügung. Bestehende Oracle-Datenbank-Beispiel-Tabellen oder auch selbst erzeugte Tabellen sind beim Testen möglich. Eigene Skripte lassen sich sichern und einfach mit anderen Nutzern teilen. Um das Arbeiten und Erlernen von SQL- und PL/SQL-Features zu erleichtern, stehen zusätzlich kleine Tutorials – speziell auch für 12.2 − zur Verfügung. Sie geben einen Einblick in das Arbeiten mit SQL, PL/SQL, JSON, XML oder auch Oracle Text, um nur einige Beispiele zu nennen.

Fazit

Es gibt einige interessante Neuigkeiten − gerade auch für Datenbank-Entwickler − in Oracle Database 12*c* Release 2. Der Artikel konnte nur einen kleinen Einblick in die Vielzahl der Features geben. Allein im PL/SQL-Umfeld oder auch in den Bereichen „Oracle Text“, „Spatial“ und „XML“ finden sich noch viele weitere interessante Features. Wer daher mehr über die einzelnen Themenbereiche erfahren möchte, sollte die Handbücher und die zahlreichen Blog-Einträge zurate ziehen − oder das Ganze gleich mit Live SQL ausprobieren.

Weitere Informationen

- Documentation (12.2): *http://docs.oracle.com/database/122/nav/portal_booklist.htm*
- Dojos: *http://tinyurl.com/dojoonline*
- Download von SQL Developer: *http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html*
- Blog-Auswahl:
  - *https://blogs.oracle.com/coretec*
  - *http://oracle-text-de.blogspot.de*
  - *http://www.thatjeffsmith.com/sql-developer*
- Oracle Live SQL: *http://www.oracle.com/technetwork/database/application-development/livesql/index.html*
- White Paper, Artikel etc.:
  - Spatial and Graph Analytics with Oracle Database 12c Release 2
  - Oracle Database In-Memory with Oracle Database 12c Release 2
  - 12 Things Developers Will Love About Oracle Database 12c Release 2

Ulrike Schwinn
ulrike.schwinn@oracle.com

Oracle steigert Umsatz

Die aktuellen Quartalszahlen zeigen einen Wachstumskurs bei Oracle: Vor allem im Cloud-Geschäft konnte das Unternehmen zulegen und allein den Bereich „Software-as-a-Service“ um 75 Prozent auf etwa ein Milliarde Dollar erhöhen. Der Umsatz mit „Platform-as-a-Service“ verzeichnet ein Plus von 40 Prozent auf etwa 400 Millionen Dollar.

Das Cloud-Geschäft ist damit im vierten Quartal wieder einmal gestiegen, macht aber mit 1,4 Milliarden Dollar gemessen am gesamten Softwaregeschäft mit 8,9 Milliarden Dollar Umsatz nur einen kleinen Teil aus. Insgesamt kletterte der Konzernumsatz um drei Prozent auf 10,9 Milliarden Dollar und der betriebliche Gewinn auf 4,1 Milliarden Dollar.


| LOW | MEDIUM | HIGH
AVAILABILITY

Hochverfügbarkeit mit Oracle 12c Release 2

Sebastian Solbach, ORACLE Deutschland B.V. & Co. KG

Die Weiterentwicklungen der Hochverfügbarkeitsfunktionalitäten in der neuen Datenbank-Version reichen von kleinen Vereinfachungen im Data-Guard- und RAC-Umfeld bis hin zur besseren Unterstützung von unterschiedlichen Desaster-Recovery-Anforderungen.

Zum ersten Mal werden das Konstrukt eines Stretch- beziehungsweise Extended-Clusters direkt bei der Installation und während der Laufzeit berücksichtigt sowie das generelle Handling von Passwort-File-, Multitenant- und No-Logging-Operationen im Data-Guard-Umfeld vereinfacht. Aber auch grundlegende Veränderungen wie die Einführung eines Domain-Service-Clusters wird den Betrieb mehrerer RAC- und Data-Guard-Umgebungen in Zukunft prägen.

Die Maximum Availability Architecture von Oracle reicht von Backup & Recovery über Real Application Clusters bis hin zu Data Guard und der Client-Anbindung. In jedem dieser Bereiche gibt es kleinere und größere Neuerungen. Die meisten neuen Funktionen gibt es aber im Automatic-Storage-Management-, RAC- und Data-Guard-Umfeld, weshalb sich der Artikel darauf beschränkt.

Automatic Storage Management

Schon seit der Version 10*g* ist Automatic Storage Management (ASM) der bevorzugte Ablageort für Datenbank-Dateien. Nicht nur das „Stripe an Mirror Everything"-Konzept (SAME) wird mit ASM bereitgestellt, ASM bietet auch eine unschlagbare Datenbank-Performance und ist dank Cluster-Awareness die Storage-Grundlage für RAC schlechthin. ASM setzt dabei wie ein Volume Manager direkt auf Platten oder den bereitgestellten LUNs auf.

Diese LUNs beziehungsweise Platten sind normalerweise über Betriebssystemmittel als Devices bereitgestellt. Allerdings müssen diese bestimmte Rechte aufweisen, damit sie von ASM bearbeitet werden können. Dies kann über Betriebssystemmittel geschehen (Stichwort „UDEV") oder wurde früher unter Linux durch eine eigene Library, die sogenannte „ASMLIB", bewerkstelligt.

Mit dem neuen Release 2 wird diese Library durch den sogenannten „ASM Filter Driver" (kurz AFD) endgültig abgelöst. Zwar existierte AFD schon im Release 1, war aber erst nachträglich und recht umständlich zu konfigurieren. Dies fällt jetzt weg und AFD kann direkt bei der Installation aktiviert werden. Hierzu muss man zwar immer noch die Devices mit richtigen Rechten versehen, allerdings kann dies nun temporär mit einfachen Betriebssystem-Befehlen vor der Installation erfolgen und muss nicht persistiert werden. Das übernimmt AFD nach der Installation dann automatisch.

AFD löst ASMLIB nicht nur einfach ab, sondern ergänzt es auch um einige sinnvolle Funktionalitäten. Zu den wichtigsten Eigenschaften gehört dabei das Absichern der Devices vor unbefugtem Zugriff. So kann ein System-Administrator nicht mehr die ASM zugewiesenen Platten aus Versehen überschreiben. Darüber hinaus bietet AFD auch Unterstützung von Thin Provisioned Storage, sodass nach Freigabe von Platz in ASM dies auch dem Storage kommuniziert wird. Außerdem werden jetzt AFD-Platten mit einer Sektorgröße von 4K optimal unterstützt und sollten deswegen bei dieser Konstellation immer eingesetzt werden.

Ein weiterer Vorteil von AFD besteht darin, dass sich der Kernel-Treiber im IO-Pfad befindet und deshalb als einziger Prozess die Platten im Zugriff hat. Dies spart nicht nur Ressourcen, sondern kann auch bei Split-Brain-Situationen hilfreich sein, da es in dem Falle ausreichend sein kann, nur den Cluster-Stack anstelle des kompletten Knotens neu zu starten. AFD steht dabei, ganz im Gegensatz zu ASMLIB, nicht nur Linux zur Verfügung, sondern existiert auch auf Windows und Solaris.

Bei ASM selber ändert sich die Art des Storage-Managements: Bis einschließlich 12.1 stand beim Management, wie der Einstellung von Striping und Mirroring, immer die Diskgruppe im Vordergrund. Bei 12.2 hingegen ist die Definition viel fein granularer und daher wird dies nun auf Datenbankebene oder besser gesagt File-Gruppen-Ebene festgelegt. Standardmäßig entspricht eine File-Gruppe einer Pluggable Datenbank und die einzelnen PDB-Dateien werden dieser File-Gruppe zugeordnet.

Dies erlaubt neben einer unterschiedlichen Mirroring-Analogie pro Datenbanken auch das Festlegen einer Quote für die jeweiligen Datenbanken innerhalb der Diskgruppe. Ebenso ist damit eine Änderung der Redundanz einfach gegeben, da auch dies sich auf eine oder mehrere File-Gruppen bezieht und damit indirekt pro Datenbank. Damit man von dieser neuen Art des Managements profitieren kann, muss allerdings die neue sogenannte „FLEX-Diskgruppe" verwendet werden, wobei sich bestehende Normal-

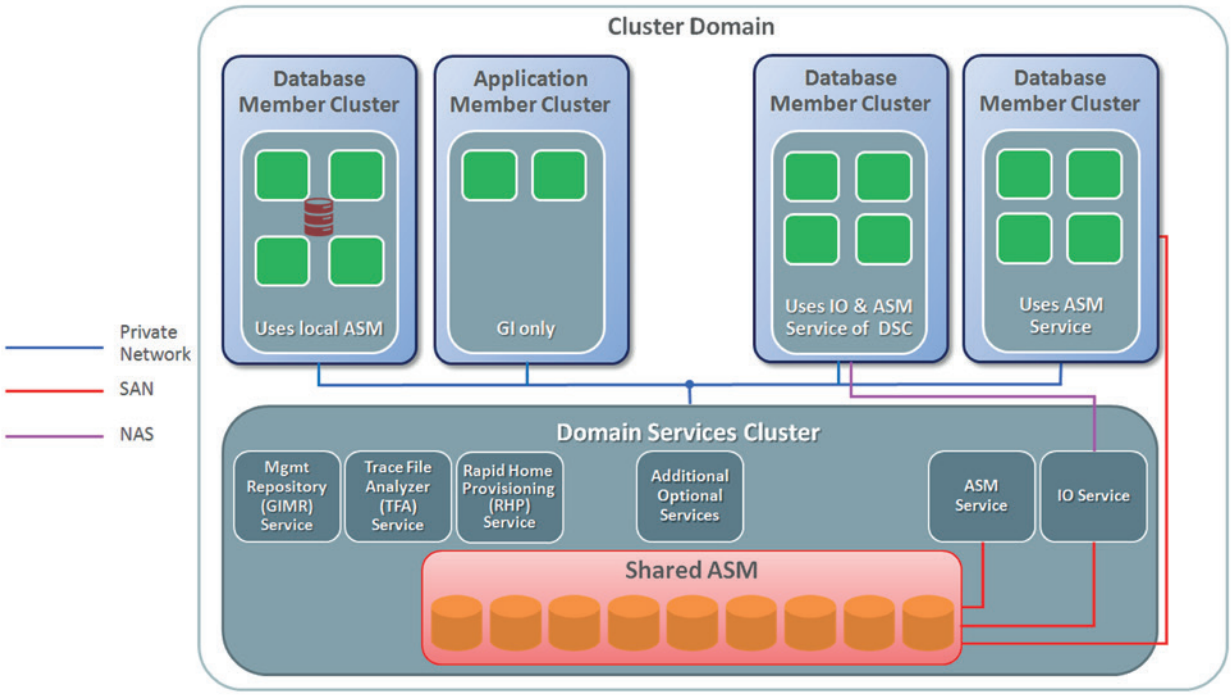


Abbildung 1: Domain Service Cluster

oder High-Redundancy-Diskgruppen bei der Migration nach 12.2 in eine solche FLEX-Diskgruppe überführen lassen.

Neu angelegte FLEX-Diskgruppen hingegen können auch vom Typ „Extended Redundancy" sein. Diese Art der Diskgruppe wird für Stretch- beziehungsweise Extended-Distance-Cluster benötigt, die nachfolgend erklärt werden.

Real Application Cluster

Eine der nach Meinung des Autors wichtigsten Änderungen im RAC Stack ist im neuen Release, dass der Oracle Cluster Stack direkt als Stretch-Cluster gekennzeichnet werden kann. Ein Stretch- beziehungsweise Extended-Cluster bezeichnet einen Cluster, bei dem die Daten auf mindestens zwei unterschiedlichen Storages liegen – im Normalfall über zwei Rechenzentren oder zwei Brandschutz-Abschnitte hinweg. Diese Information ist nicht nur für die Installation wichtig, sondern auch während der Laufzeit, da der Cluster die Informationen darüber nun aktiv in die Behebung von Split-Brain-Szenarien oder Storage-Problemen mit einbeziehen kann.

Hierzu werden sowohl die einzelnen Knoten den verschiedenen Sites zugeordnet als auch die einzelnen Failure-Gruppen der ASM-FLEX-Diskgruppen. Ebenfalls notwendig für dieses Setup ist das Angeben eines sogenannten „Quorum Device" und damit einer eigenen Quorum-Failgroup, die bei einem Split Brain zwischen den Sites bei der Entscheidung hilft, welche Seite aktiv bleibt und welche heruntergefahren werden muss.

Dies gab es auch schon bei vorhergehenden Clustern: Die einfache Mehrheitsregel besagte, dass ein Cluster-Knoten mindestens die einfache Mehrheit der Votes sehen muss, um im Cluster verbleiben zu dürfen. Jetzt erweitert das neue Release das Quorum-Konzept um zwei neue Besonderheiten: So gilt die einfache Mehrheitsregel nicht nur Cluster-übergreifend, sondern auch innerhalb eines Storage. Dies ermöglicht dem Cluster, zwischen Problemen in einem Storage und einem Komplett-Ausfall des Storage zu unterscheiden. Als Ergebnis daraus haben FLEX-Diskgruppen deshalb auch mindestens sieben Votes: drei pro Storage und ein übergreifendes Quorum. Die zweite Besonderheit ist, dass dies nicht nur für die Infrastruktur-Diskgruppe des Clusters gilt, sondern letztendlich für jede FLEX-Diskgruppe, auch wenn nur Datenbank-Dateien enthalten sein sollten.

Unabhängig von Stretch-Clustern ist in diesem Zusammenhang die Änderung des Cluster-Verhaltens interessant, falls es zu einem Split-Brain (dem Verlust des Interconnects zwischen den Cluster-Knoten) kommt. Wurde hier bei gleichartigen Clustern in der Vergangenheit immer zugunsten des Knotens mit der niedrigeren Knotennummer entschieden, fließt nun in die Entscheidung eine Gewichtung des Knotens ein. Dazu gehört neben der Verfügbarkeit des Public-Netzwerks auch die Anzahl der RAC-One-Node-Datenbanken und sogenannter „Singleton Services", also Ressourcen, die nur auf einem Knoten verfügbar sind. Zudem ist es möglich, das Verhalten als Administrator festzulegen, indem man einzelne Ressourcen oder ganze Server als kritisch („CSS_CRITICAL") einstuft und dementsprechend markiert.

Das eigentlich Failover bei RAC-Instanzen wurde ebenfalls beschleunigt. Dazu dient das sogenannte „Recovery Buddy Feature". Im Gegensatz zu früher werden jetzt die Instanzen via Round Robin festgelegt, die bei einem Fehlerfall federführend für den Ausfall eines Knotens zuständig sind. Dies beschleunigt das Recovery im Fehlerfall, allerdings wird hierzu etwas mehr I/O erzeugt.

Nicht nur Stretch-Cluster ändern die Installation, auch der Installationsprozess des Oracle-Cluster-Stacks wurde insgesamt verändert. So wird nun kein „runInstaller" mehr aufgerufen, sondern die Software einfach in das entsprechende Verzeichnis entpackt (bei Neuinstallation ist dies nur

auf dem ersten Knoten wichtig). Danach wird die Konfiguration des Clusters über das „gridSetup“-Skript gestartet. Die Konfiguration erlaubt nun die Konfiguration einer separaten Diskgruppe für das Grid Infrastructure Management Repository. Mit 12.2 ist ASM nun zwingend vorgeschrieben, auch wenn die Datendateien später auf einem NFS-Filer liegen sollten. Ebenfalls weiterentwickelt wurde das Cluster Health Framework, bestehend aus:

- *CLUVFY (Cluster Verification Utility)*
  Ein Tool zur kontinuierlichen Überwachung, ob der Cluster in Ordnung ist
- *ORACHK*
  Ein Tool zur manuellen Überprüfung auf Oracle Best Practices für Betriebssystem und Datenbank (verwendet implizit „CLUVFY“ mit)
- *Cluster Health Monitor (CHM)*
  Logging von Betriebssystem-Informationen zur Analyse bei Problemen wie Disk I/O, Netzwerk-Auslastung, CPU-Verbrauch
- *Trace File Analizer (TFA)*
  Sammlung aller relevanten Logs im Fehlerfalle, auch automatisch ohne Interaktion vom Administrator möglich
- *Hang Manager*
  Prozess bei RAC, um sogenannte „blockierende Sessions“ automatisch zu beenden, wenn Cluster-Probleme auftreten
- *Memory Guard*
  Überwachung der Memory-Auslastung eines Knotens und gegebenenfalls Einschreiten bei Memory-Engpässen, um die Stabilität des Clusters zu gewährleisten
- *Quality of Service Management (QoS)*
  Festlegung der SLAs im Cluster, um einzelnen Services/Datenbanken genügend Leistung zuzuweisen
- *Cluster Health Advisor*
  Neues 12.2-Cluster-weites Tool zur Analyse und Behebung von Performance-Engpässen bei RAC-Datenbanken

Die einzelnen Tools bieten alle kleinere Erweiterungen vom Baselining und ASM-Best-Practice-Check im „CLUVFY“ über CSV-Dateiausgaben und Erweiterbarkeit sowie Plug-ins im CHM bis hin zur manuellen Feinjustierung des Hang Manager. Erweitert wird der Funktionsumfang um den Cluster-Health-Advisor, der so wie ADDM in der Datenbank Cluster-übergreifend Performance-Tuning und Fehlerbehebungen vorschlagen kann. Die gestiegene Datenmenge erfordert dabei ein größeres Grid Infrastructure Management Repository (GIMR), sodass für Standard-Cluster nun etwa 33 GB dafür reserviert werden sollten.

Alternativ kann man sich entscheiden, diese GIMR-Daten aller Cluster auszulagern, was nur ein zentrales GIMR erfordert. Hierzu dient der sogenannte „Domain Service Cluster“. Er übernimmt nicht nur die zentrale Aufgabe des GIMR, sondern bietet auch andere zentrale Services. Dies geht von zentralen TFA über die Funktionalität des Rapid Home Provisioning (eine Lifecycle-Management-Lösung, um die Oracle-Software auf allen Datenbank- und Cluster-Servern im Unternehmen einfacher updaten und zentral verwalten zu können) bis hin zu sogenannten „I/O Services“. So kann ein Domain Service Cluster die komplette Storage-Funktionalität des ASM übernehmen, sodass sogenannte „Member Cluster“ zwar noch Zugriff auf die LUNs, aber keine zusätzlichen ASM-Ressourcen mehr benötigen *(siehe Abbildung 1)*. Der Oracle-Cluster (ohne RAC) selber bietet nun endlich sogenannte „Ressourcen-Gruppen“. Damit ist es möglich, mehrere Services zusammenzufassen und diese als einzelne Einheit zu betrachten. So könnten im Fehlerfall oder bei Wartungsarbeiten einfach alle Ressourcen einer solchen Gruppe umgezogen werden, ohne dass man sich über die Abhängigkeiten Gedanken machen müsste, wie in älteren Cluster-Versionen. Hierzu müssen lediglich die Ressourcen-Gruppe angelegt und die dazugehörigen Ressourcen hinzugefügt werden *(siehe Listing 1)*.

Data Guard

Bei der Weiterentwicklung von Data Guard konzentrierte man sich auf eine einfachere Administration, mehr Kontrolle von Umgebungen und bessere Absicherungen im Fehlerfall. Im Bereich „Active Data Guard“ kommen noch weitere Möglichkeiten hinzu, um die Performance von Abfragen zu verbessern und damit mehr Last auf eine Read Only Standby auslagern zu können.

```
# Create resource group
crsctl add resourcegroup xag_group -type cluster_resourcegroup -attr
"SERVER_POOLS"

# Add resources to group
crsctl modify resource xag_vip -attr "RESOURCE_GROUP=xag_group"
crsctl modify resource xag_mp -attr "RESOURCE_GROUP=xag_group"
crsctl modify resource xag_app -attr "RESOURCE_GROUP=xag_group"
```

Listing 1

Zu den administrativen Vereinfachungen gehört auch die Möglichkeit, endlich mit dem Database Configuration Assistant (DBCA) Standby-Datenbanken erzeugen zu können. Dies ist zwar im Moment nur per Kommandozeile im Silent-Modus des DBCA und für Single-Instanz-Datenbanken möglich, erlaubt aber mit dem neu eingeführten Skripting im Data Guard Broker das komplett automatische Anlegen einer Standby-Datenbank inklusive Konfiguration. Der DBCA ruft dafür automatisch den RMAN auf, um die Standby von der aktiven Primary-Datenbank zu duplizieren. RMAN selbst hingegen wurde erweitert, um auch direkt eine FAR-SYNC-Instanz anzulegen, auch hier entfallen also die manuellen Schritte, um den SYSTEM-Tablespace und andere benötigte Datenbank-Informationen manuell zu kopieren. Zur besseren Handhabung im Web und in der Cloud wurde der Data Guard Broker erweitert und bietet ein REST-Interface, um Standby-Datenbanken zu administrieren.

Mehr Kontrolle über die Standby-Umgebungen im RAC-Umfeld erhält man durch die Möglichkeit, mehrere Apply-Instanzen auf der Standby zu verwenden. Dies erhöht die Apply-Performance fast linear. Einzige Einschränkung ist, dass alle Apply-Instanzen entweder „Read Only" oder nur „Mounted" sind.

Mehr Kontrolle im Multitenant-Umfeld bedeutet eine leichtere Administration und Zuweisung sogenannter „Subset Standby" (Data-Guard-Umgebungen mit nur einer beschränkten Auswahl aller verfügbaren PDBs) sowie die einfachere Migration und Failover auf PDB-Ebene. Allerdings ist für Letzteres eine Standby-Umgebung mit jeweils zwei CDBs notwendig, die sich über Kreuz absichern. Dieses Feature soll übrigens mit den kommenden Patch Bundles auch auf 12.1.0.2 zur Verfügung stehen.

Mehr Schutz bietet Data Guard nun insbesondere durch das einfachere Wiederherstellen von No-Logging-Operationen. Sollten auf der Primärseite No-Logging-Operationen durchgeführt werden, kann RMAN nun die betroffenen Blöcke durch ein Block Recovery auf der Standby wieder-herstellen. Hierzu protokolliert die Primär-Datenbank die einzelnen Bereiche, die durch No-Logging-Operationen betroffen sind, und weiß damit genau, welche Blöcke bei Aufruf des Befehls wiederhergestellt werden müssen. Damit lassen sich nun insbesondere lange Ladejobs in Data-Warehouse-Umgebungen ohne Logging sehr performant ausführen und die Standby nach Erfolg leicht nachziehen.

In dieselbe Richtung geht die Möglichkeit, bei 12.2 einen Block-Vergleich einzelner Datendateien der Primär- und Standby-Datenbank durchführen zu können, um auch schleichende Block-Korruptionen festzustellen und zu beheben. Dies geht einher mit einem besseren Auto-Block-Repair bei Active-Data-Guard-Umgebungen, da sich nun auch Probleme im Datendatei-Header reparieren lassen.

Data Guard bietet nun implizit mehr Schutz für Maximum-Performance-Umgebungen mit asynchronem Redo-Log-Transport. So wird bei Storage-Problemen die Datenbank-Instanz nicht einfach sofort beendet, sondern kann ihre zuletzt angefallenen Redo-Log-Informationen noch an die Standby senden, bevor sie abbricht. Damit ist im Falle eines Storage-Ausfalls auch bei asynchronen Data-Guard-Umgebungen „Zero Data Loss" gewährleistet.

Eine kleine, aber feine Verbesserung im Data-Guard-Bereich ist mit 12.2 der automatische Abgleich der Passwort-Dateien, womit deren manuelles Kopieren nun der Vergangenheit angehört. Ebenso wurde die Verwaltung mehrerer Fast-Start-Failover-Umgebungen verbessert: So ist nur noch eine Konfiguration für alle Standby-Umgebungen notwendig, um mehrere Observer mit optional mehreren Zielen zu konfigurieren.

Für Active Data Guard und damit der zum Lesen geöffneten Standby sind einige Performance-relevante Neuerungen hinzugekommen. Hierzu gehört die Verfügbarkeit eines vom Primärsystem unabhängigen In-Memory-Column-Store und damit die Verwendung von In-Memory auf der Standby-Seite, wenn auch nur in der Oracle Cloud und auf Exadata-Systemen.

In allen Active-Data-Guard-Versionen stehen die Funktionen des Performance-Tunings auf der Standby-Seite zur Verfügung. Dazu gehört sowohl das Automatic Workload Repository auf der Standby-Seite als auch das Ausführen von SQL-Tuning auf derselben. Damit dies funktioniert, bedient sich 12.2 eines einfachen Tricks: Alle notwendigen schreibenden Aktionen werden auf die Primär-Datenbank umgeleitet beziehungsweise im AWR-Fall auch in eine komplett andere Datenbank. Somit kann Tuning anhand von AWR-Reports auf der Standby durchgeführt werden, wie auch die Standby für die Last-intensive Berechnung von SQL-Tuning-Operationen herangezogen wird.

Indirekt mehr Performance bietet der transparente Rollentausch von 12.2-Active-Data-Guard-Instanzen. So werden beim Switchover und Failover bestehende Read-Only-Benutzer nicht mehr getrennt und können einfach weiterarbeiten. Ebenso gibt es ein „Session Prewarming", die Standby hält also Server-Session für den Fehlerfall vorrätig, sodass es bei einem Failover nicht zu einem unvorhergesehenen Connection Storm kommen kann.

Fazit

Die neue Datenbank 12*g* Release 2 bietet auch im Hochverfügbarkeitsbereich viele Neuerungen, die das Betreiben einer hochverfügbaren Oracle-Datenbank erheblich verbessern. Viele Kleinigkeiten stellt man am Anfang nur bedingt fest, wird diese aber sehr schnell bei älteren Versionen vermissen.

Sebastian Solbach
sebastian.solbach@oracle.com

Oracle 12.2 New Features für das Data Warehouse

Alfred Schlaucher, ORACLE Deutschland B.V. & Co. KG

Der Artikel zeigt eine Auswahl aus den vielen neuen Funktionen der Oracle-Datenbank 12*c* Release 2 aus der Perspektive von Data-Warehouse-Anforderungen. Reihenfolge und Umfang der Beschreibung sind zufällig und sagen nichts über die Wichtigkeit der Features aus.

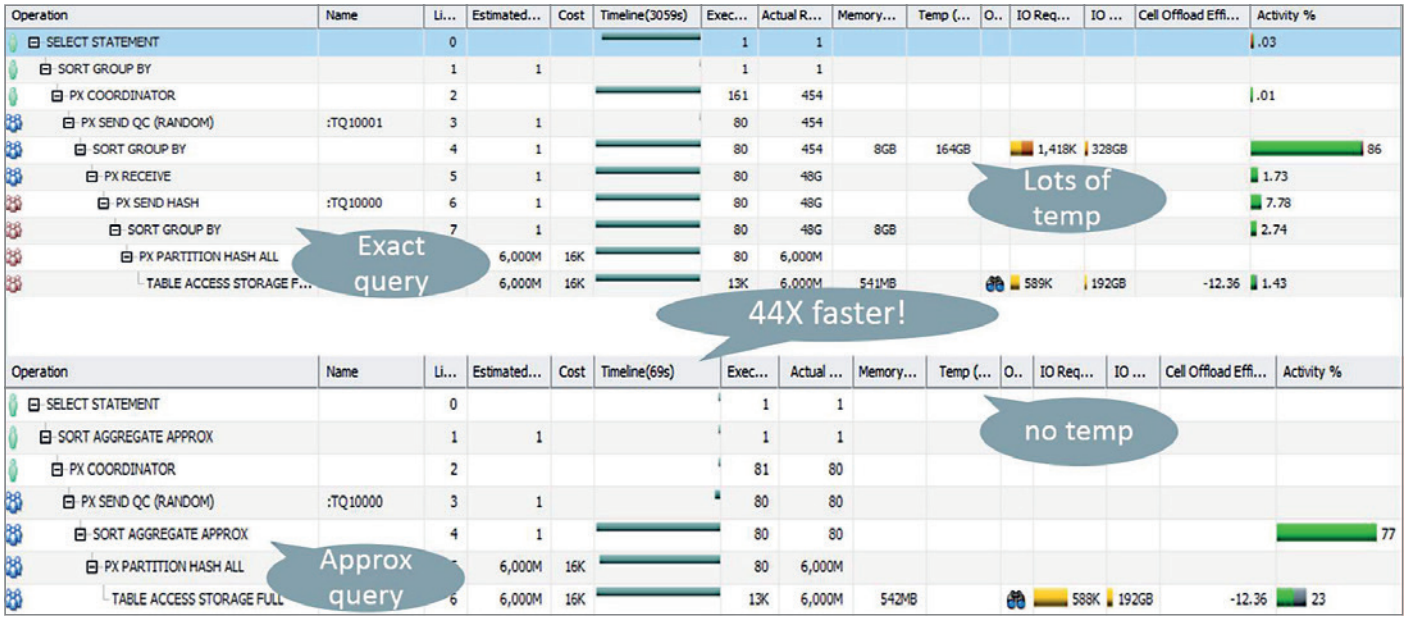


Abbildung 1: Ausführungsplan eines parallelisierten „approx_median"-Statements, oben ohne und unten mit „approx_for_aggregation = true"

Manchmal reicht es auch ungefähr – Approximate Query Processing

Obwohl viele analytische Abfragen nur aggregierte Informationen nutzen, verursachen sie damit eine besonders intensive Ressourcennutzung. Diese teuren Abfragen lesen komplette Datenbestände in den Hauptspeicher, schreiben bei Erreichen der Bufferpool-Grenze in den Temp-Tablespace und liefern ein hundertprozentig korrektes Ergebnis, auch wenn Detail-Informationen durch die Aggregation nicht mehr sichtbar sind und nur wenige Trend-anzeigende Zeilen am Bildschirm erscheinen.

Das schon im Release 1 in Teilen eingeführte Approximate Query Processing minimiert diese Aufwände. Vergleichbar mit dem Sampling von Statistiken, schätzt die Software den wahrscheinlichen Wert eines Ergebnisses. Die Genauigkeit von mehr als 97 Prozent in 95 Prozent aller Fälle reicht für die meisten Anforderungen aus.

Es sind SQL-Funktionen wie „approx_count_distinct" (seit 12.1), „approx_percentile", und „approx_median" (seit 12.2), die dies ermöglichen. Ein Beispiel gibt den Median der Umsatzspalte einer großen Fakten-Tabelle aus und qualifiziert gleichzeitig das Ergebnis, indem es die Fehlerrate und die Wahrscheinlichkeit des richtigen Ergebnisses mit anzeigt *(siehe Listing 1)*.

Die Abfrage ist schneller. Den Grund dafür zeigt der Ausführungsplan, der wesentlich weniger Temp-Space ausweist. Über das Setting „approx_for_aggregation = true/false" wird das Feature ein- und ausgeschaltet *(siehe Abbildung 1)*.

Online-Fähigkeit des Partition-Managements

Bezüglich Partitioning, eine der wichtigsten Einrichtungen für das Oracle-basierte Data Warehouse, bringt das Release 2 eine ganze Menge Erweiterungen. Wer sich technische Warehouse-Architekturen genauer anschaut, der weiß, dass Partitioning neben dem Partition Pruning besonders auch für die Verwaltung und Bereitstellung von Datenbereichen und im Rahmen des ETL genutzt wird. Das sind unter anderem verschieden hohe Komprimierungen, Verteilung auf verschieden schnelle Datenträger, Trennung zwischen Read-Only- und Update-Bereichen, inkrementelles Aktualisieren von Indizes, Statistiken und Materialized Views sowie partitionsbezogene Lade-Steps.

Da auch Warehouse-Systeme zunehmend rund um die Uhr und an sieben Tagen pro Woche zur Verfügung stehen müssen, hat Oracle schon in 12*c* Release 1 viele Verwaltungstätigkeiten online-fähig gemacht. Gemeinsam mit den Erweiterungen im neuen Release 2 können

```
SELECT
   APPROX_MEDIAN (umsatz) AS median_umsatz,
   APPROX_MEDIAN (umsatz, 'ERROR_RATE') AS error_rate,
   APPROX_MEDIAN (umsatz, 'CONFIDENCE') as confidence
FROM F_UMSATZ
```

Listing 1

```
ALTER TABLE f_umsatz
MOVE PARTITION sammelpartition TABLESPACE mai_2017
COMPRESS ONLINE
INCLUDING ROWS WHERE bearbeitungs_status = 'fix';
```

Listing 2

```
CREATE TABLE ex_umsatz (Region_nr number, prod_id number,
cust_id number, amount_sold number, quantity_sold number)
ORGANIZATION EXTERNAL
(TYPE oracle_loader
DEFAULT DIRECTORY load_d1
ACCESS PARAMETERS ( ….) REJECT LIMIT UNLIMITED
PARTITION BY RANGE (Region_nr)
(PARTITION west VALUES LESS THAN (2) LOCATION ('westregion.txt'),
PARTITION nord VALUES LESS THAN (3) DEFAULT DIRECTORY load_d2
  LOCATION ('nordregion.txt'),
PARTITION rest VALUES LESS THAN (4));
```

Listing 3

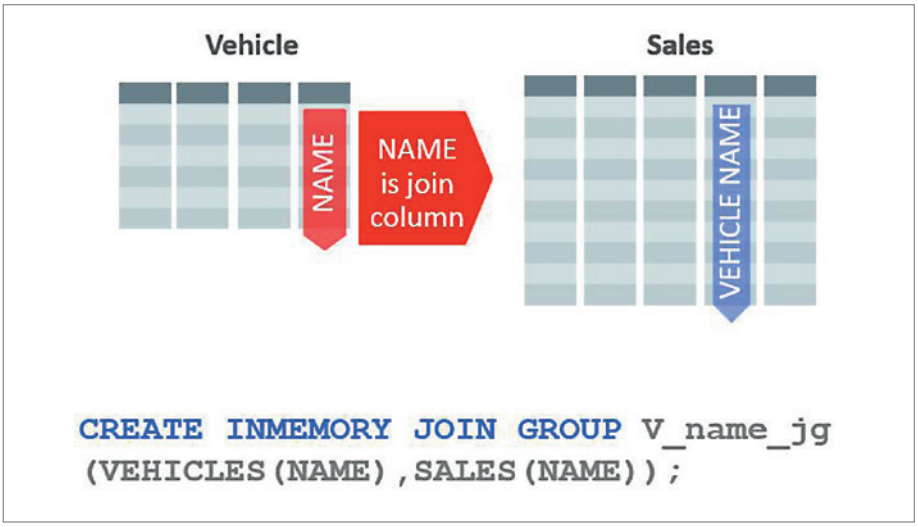


Abbildung 2: Definition eines Join-Group-Objekts über die Foreign-/Primary-Keys

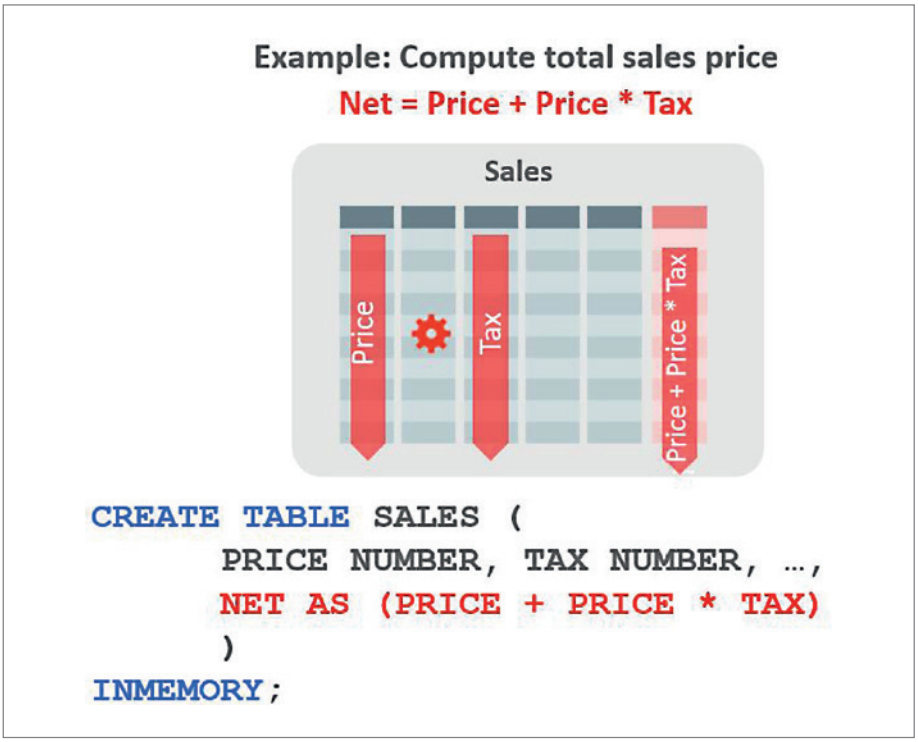


Abbildung 3: Definition einer In-Memory-Expression mithilfe bestehender Spalten

jetzt Partitionen online gesplittet, gemischt und von einem Tablespace zu einem anderen bewegt werden, ohne die Warehouse-Datenbank für die Anwender zu sperren. Auch das nachträgliche Partitionieren einer noch nicht partitionierten Tabelle ist online möglich (neues Schlüsselwort „ONLINE“ in den Statements).

Filter-Möglichkeit bei Partition-Management

Neu ist auch die Möglichkeit der Filter-Setzung bei Move-, Merge- und Split-Operationen. Bislang musste man abwarten, bis alle Sätze einer aktuellen Monats- oder Tagespartition gemeinsam einen bestimmten Bearbeitungszustand erreicht hatten, um die Partition dann als Ganzes in einen endgültigen Zustand zu überführen. Jetzt sind Konzepte möglich, nach denen man eine fixe Partition für laufende Daten-Manipulationen, ETL etc. vorhält und fertig bearbeitete Sätze in regelmäßigen, automatisierten Move-Jobs in ihre endgültige Zielpartition überführt. Das jetzt mögliche Verfahren kann eine Alternative zu dem oft praktizierten „Partition Exchange and Load“-Verfahren sein, bei dem man immer ganze Partitionen austauscht und bei dem es eine Abhängigkeit zwischen der Ladefrequenz des ETL-Prozesses und der Art der Partitionierung („RANGE“ nach Zeit) gibt.

Das Statement im Beispiel bewegt nur Sätze, die in dem Feld „bearbeitungs_status“ einen entsprechenden Wert eingetragen haben. Man findet auch das Schlüsselwort „ONLINE“, das jetzt bei nahezu allen Verwaltungsaktionen möglich ist *(siehe Listing 2)*.

Multi-Column und Automatic List-Partitioning

Im Release 2 ist auch das LIST-Partitioning erweitert. Da die meisten Partitionierungen in Data-Warehouse-Systemen Range- und Zeit-basiert sind, verbleibt über das Sub-Partitioning nur noch eine einzige Option für ein zusätzliches, nicht Zeit-bezogenes Partitionierungskriterium. Man hatte also beispielsweise zunächst Monatspartitionen eingerichtet und dann über ein fachliches Kriterium List-Partitionen definiert. Durch die Einführung des Multi-Column-Partitioning kann man

```
--- Beispiel: Dimension-Objekt
create force attribute dimension Att_dim_artikel
dimension type STANDARD
USING D_ARTIKEL
Attributes (
      ARTIKEL_NAME   AS  ARTIKEL_NAME,
      ARTIKEL_ID     AS  ARTIKEL_ID,
      GRUPPE_NR      AS  GRUPPE_NR,
      GRUPPE_NAME    AS  GRUPPE_NAME,
      SPARTE_NAME    AS  SPARTE_NAME,
      SPARTE_NR      AS  SPARTE_NR)
LEVEL Artikel
   KEY ARTIKEL_ID
   DETERMINES (ARTIKEL_NAME)
LEVEL Artikel_Gruppe
   KEY GRUPPE_NR
   DETERMINES (GRUPPE_NAME )
LEVEL Artikel_Sparte
   KEY SPARTE_NR
   DETERMINES (SPARTE_NAME)

--- Beispiel: Hierarchy-Objekt
CREATE OR REPLACE HIERARCHY REGION_hier
USING Att_dim_REGION
   (ORT CHILD OF KREIS CHILD OF LAND CHILD OF REGION)

--- Beispiel: Analytic View Objekt
CREATE OR REPLACE ANALYTIC VIEW AV_Umsatz
using f_umsatz
dimension by
   (ATT_DIM_ZEIT
       KEY Zeit_ID REFERENCES zeit_id
       HIERARCHIES (
           ZEIT_MONAT_QUARTAL_JAHR default,
           ZEIT_WOCHE_JAHR),
   ATT_DIM_ARTIKEL
       KEY ARTIKEL_ID REFERENCES ARTIKEL_ID
       HIERARCHIES (
           ARTIKEL_HIER),
   ATT_DIM_REGION
       KEY REGION_ID REFERENCES REGION_ID
       HIERARCHIES (
           REGION_HIER)
   )
MEASURES
   (
   UMSATZ FACT UMSATZ
       CLASSIFICATION format_string VALUE '9,999.99',
   MENGE
       CLASSIFICATION format_string VALUE '9,999',
   UMSATZ_GESAMT
       CLASSIFICATION format_string VALUE '9,999.99' )
```

Listing 4

jetzt eine Kombination mehrerer Tabellenspalten als Partitionierungskriterium wählen und kommt somit den fachlichen Anforderungen näher. Das Schlüsselwort „AUTOMATIC“ erlaubt zudem das automatische Anlegen neuer List-Partitionen, wenn neue Werte in der ursprünglichen „Create Table“-Definition noch nicht erfasst waren.

Read-Only-Partitions

Von einem gewissen Verarbeitungsstand an werden in Warehouse-Systemen Daten nicht mehr modifiziert; sie gelten als historisch und unveränderbar. Unter Umständen muss sogar dokumentiert und nachgewiesen werden, dass Daten von einem bestimmten Datum an nicht mehr verändert worden sind. Um unerwünschte Änderungen an historischen Daten, etwa durch fehlerhafte ETL-Prozesse, zu verhindern, kann man Partitionen jetzt auch unabhängig von ihrer Tablespace-Zugehörigkeit auf „Read Only“ setzen.

External Tables sind jetzt partitionierbar

Auch External Tables sind jetzt partitionierbar. Auch heute noch füttern viele Firmen ihre Warehouse-Systeme mit Textdateien, die sie mit External Tables in die Datenbank laden. Erhält man gleich-strukturierte Dateien von mehreren Stellen im Unternehmen, so muss man diese jetzt nicht mehr in eine einzige Datei überführen oder gar mehrere Lesevorgänge starten, nur weil mehrere Dateien vorliegen. Im Release 2 kopiert man alle Files mit unterschiedlichen oder gleichen Namen in ein oder mehrere Directories und kann mit einem einzigen „SELECT“ auf alle Files gleichzeitig zugreifen *(siehe Listing 3)*.

In-Memory-Column-Store – Join Groups und In-Memory-Expressions

Auch der In-Memory-Column-Store gehört mittlerweile zu den wichtigen Hilfsmitteln im Oracle-basierten Data Warehouse, denn die Column-orientierte Speicherung unterstützt das Lesen von zusammenhängenden Datenbereichen, sodass Abfragen selbst gegenüber bereits im Buffer-Pool-Cache liegenden Tabellendaten um Faktoren schneller sind. Dies begünstigt zunächst jedoch nur das Lesen von Daten, also die IO-Operationen, hier das Lesen aus dem Column-Store. Analytische Abfragen müssen darüber hinaus oft noch mehr als 40 Prozent Rechenaufwand für Aggregationen und ähnlich hohe Aufwände für Joins auf große Tabellen leisten. Oracle 12.2 unterstützt diese besonderen Anforderungen durch In-Memory-Expressions und Join-Groups.

Gerade die Joins in Star-Modellen haben es in sich. Einerseits sind es wenige, aber immer wieder gleiche Joins, andererseits ist mindestens eine Tabelle oft extrem groß. Das ist ein ideales Anwendungsfeld für die neuen Join-Groups, die als zusätzliche Objekte im Column-Store

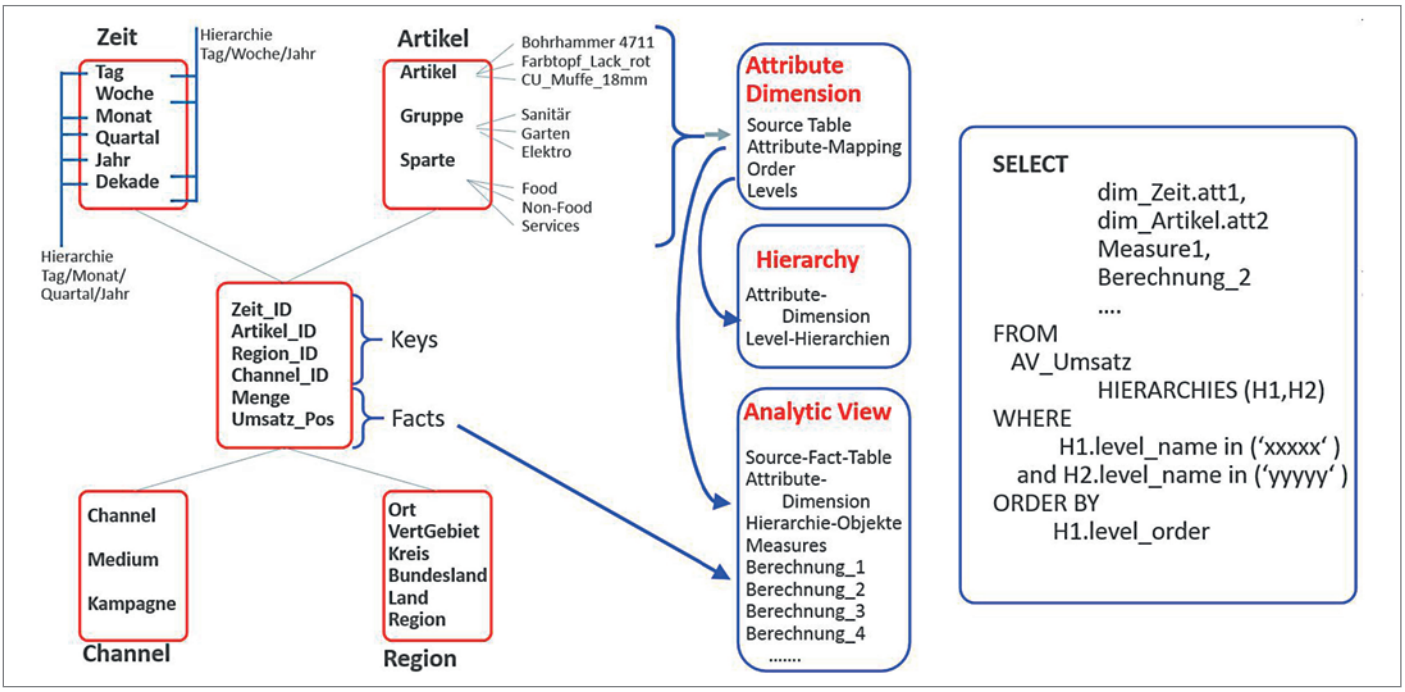


Abbildung 4: Mini-Szenario mit Star-Schema-Tabellen, Analytic-View-Definitionen und Abfrage auf die Analytic Views

die Join-Verbindungen zwischen Dimensions- und Faktentabellen fest in sich verdrahtet haben. Der Aufwand zur Bildung des Joins wird zum Zeitpunkt der Definition eines Join-Group-Objekts geleistet und nicht mehr während der späteren Benutzerabfrage. Eine Performance-Optimierung von Faktor 2-3 gegenüber den eh schon schnellen In-Memory-Joins ist möglich *(siehe Abbildung 2)*.

Ähnlich funktionieren auch die In-Memory-Expressions. Damit lassen sich immer wiederkehrende analytische Berechnungen im Vorwege durchführen; sie müssen nicht mehr zum Abfragezeitpunkt gebildet werden. Bis zu zwanzig solcher Vorabberechnungen lassen sich pro Tabelle definieren *(siehe Abbildung 3)*.

Die Datenbank pflegt eine separate Statistik über die benutzten Ausdrücke in „SELECT"-Statements. Mithilfe dieser Statistik kann die Datenbank aus den am häufigsten verwendeten Ausdrücken automatisch und selbständig In-Memory-Expressions bilden.

Analytical Views

Ein besonderes Highlight in 12c Release 2 sind aus analytischer Sicht die Analytical Views. Star-Modelle sind als multidimensionale Konstruktion mehr als nur eine über Joins verbundene Ansammlung von Tabellen. In den Fakten-Tabellen finden sich die Kennzahlen wieder. Das sind entweder einfache Spalten oder berechnete Ausdrücke. In den Dimensionen sind mehrere Columns zu Attribut-Gruppen, den sogenannten „Level" zusammengefasst, und die Level sind untereinander entweder horizontal oder vertikal in Level-Hierarchien (bekannt als „Drillpfade") geordnet.

Diese logischen Zusammenhänge, die das multidimensionale Modell ausmachen, lassen sich im Release 2 in der Oracle-Datenbank als Analytical Views dokumentieren. Das sind Metadaten-Objekte im Dictionary der Datenbank, die sich in ihrer Definition auf die Tabellen eines Star-Schemas beziehen.

Es gibt Dimensions-Objekte zur Beschreibung der benötigten Dimensions-Attribute und der Attribut-Gruppierungen, die die Levels ergeben, sowie die Hierarchie-Objekte, die die Level-Definition in den Dimensions-Objekten in Hierarchien ordnen. Hinzu kommen die Analytic-View-Objekte, in denen man die Kennzahlen und die zugehörigen Dimensions-Objekte festlegt *(siehe Listing 4)*.

Worin liegt aber jetzt der Nutzen des Ganzen? Ganz einfach: SQL-Abfragen lassen sich jetzt auf einer eher fachlichen Ebene – den Hierarchie-Leveln – formulieren und Drillpfade kann man damit über ihre Namen abfragen. Neben den in der Fakten-Tabelle bereits vorkommenden Kennzahlenspalten kann man in der Analytic-View-Definition zusätzliche Kennzahlen über Ableitungen und Formeln definieren, die noch nicht als Spalte der Fakten-Tabelle existieren. Auch diese sind direkt ansprechbar. Man spart also zusätzliche Aufbereitungen, etwa in Aggregat-Tabellen oder in den Business-Layern der BI-Tools *(siehe Abbildung 4)*.

Alfred Schlaucher
alfred.schlaucher@oracle.com

Schöner Coden – PL/SQL analysieren mit PL/Scope

Sabine Heimsath, its-people GmbH

In der Schatzkiste der unbekannten Features der Oracle-Datenbank gibt es seit Version 11.1 ein Tool für Entwickler, das mit dem Release 12.2 noch spannender geworden ist: PL/Scope. Es handelt sich nicht um eine fertige Anwendung, die Code optimiert, aber es ermöglicht detaillierte Analysen, mit denen der Code verbessert werden kann. Außerdem lassen sich die Informationen nutzen, um einen Überblick über die Struktur des Codes zu bekommen. Der Artikel zeigt, welche Neuerungen mit 12.2 hinzugekommen sind und was man damit anfangen kann.

Eine Installation von PL/Scope ist nicht notwendig, wenn die Datenbank mindestens auf Version 11.1.0.7 ist. Für ältere 11er-Versionen sei auf die Doku verwiesen. Das Einzige, was zu tun ist, ist, PL/Scope mit „alter session set plscope_settings = ‚IDENTIFIERS:ALL'" für die aktuelle Session zu aktivieren beziehungsweise in 12.2 mit „alter session set plscope_settings = ‚IDENTIFIERS:ALL,STATEMENTS:ALL'".

Diese Einstellung kann auch im SQL Developer unter „Preferences > Database > PL/SQL-Compiler" vorgenommen werden. Wegen eines Bugs funktioniert dies allerdings nur, wenn die GUI auf Englisch eingestellt ist. Nach der Umstellung werden bei jedem Compile-Vorgang in dieser Session die Metadaten für alle Bezeichner und SQL-Statements der jeweiligen Compilation-Unit generiert. Die Einstellung der aktuellen Session lässt sich abfragen *(siehe Listing 1)*. Möchte man wissen, welche Objekte mit welchen Einstellungen kompiliert wurden, hilft die Abfrage „select owner, name, type, plscope_settings from all_plsql_object_settings".

```
select name, value
   from v$parameter
 where name = 'plscope_settings'
```

Listing 1

Die PL/Scope-Views

Interessanter wird es bei der Frage, wie das Ergebnis in den Views aussieht. Als Beispiel dient die kleine Prozedur „tables_out". PL/Scope nummeriert alle Verwendungen von Bezeichnern in der Reihenfolge ihres Auftretens durch *(siehe Abbildung 1)*. Eine Ausnahme gibt es: SQL-Statements werden in umgekehrter Richtung geparst, daher ist die Nummerierung hier umgedreht. *Abbildung 2* zeigt das Ergebnis der View „USER_IDENTIFIERS".

```
create or replace procedure hr.emps_out
  authid definer
is
  v_string varchar2(2000);
  v_int integer;
begin
  for rec in (select last_name, salary from employees)
  loop
    v_string := rec.last_name;
    v_int := rec.salary;
    sys.dbms_output.put_line(v_string || ' ' || sys.standard.to_char(v_int,99999));
  end loop;
end;
/
```

Abbildung 1: So sieht PL/Scope den Code: Die „USAGE_ID" wird in fortlaufender Reihenfolge vergeben

NAME	SIGNATURE	TYPE	USAGE	USAGE_ID	LINE	COL	USAGE_CONTEXT_ID	OBJECT_NAME	OBJECT_T...
EMPS_OUT	CB5CCED	PROCEDURE	DECLARATION	1	1	14	0	EMPS_OUT	PROCEDURE
EMPS_OUT	CB5CCED	PROCEDURE	DEFINITION	2	1	14	1	EMPS_OUT	PROCEDURE
V_STRING	5C35983	VARIABLE	DECLARATION	3	4	3	2	EMPS_OUT	PROCEDURE
VARCHAR2	FEE1E7E	CHARACTER DATATYPE	REFERENCE	4	4	12	3	EMPS_OUT	PROCEDURE
V_INT	8721790	VARIABLE	DECLARATION	5	5	3	2	EMPS_OUT	PROCEDURE
INTEGER	4EEE7F4	SUBTYPE	REFERENCE	6	5	9	5	EMPS_OUT	PROCEDURE
REC	20A90B4	ITERATOR	DECLARATION	7	7	7	2	EMPS_OUT	PROCEDURE
EMPLOYEES	85DA2A5	TABLE	REFERENCE	9	7	45	8	EMPS_OUT	PROCEDURE
SALARY	55D86C5	COLUMN	REFERENCE	10	7	33	8	EMPS_OUT	PROCEDURE
LAST_NAME	1687ACC	COLUMN	REFERENCE	11	7	22	8	EMPS_OUT	PROCEDURE
V_STRING	5C35983	VARIABLE	ASSIGNMENT	12	9	5	8	EMPS_OUT	PROCEDURE
REC	20A90B4	ITERATOR	REFERENCE	13	9	17	12	EMPS_OUT	PROCEDURE
V_INT	8721790	VARIABLE	ASSIGNMENT	15	10	5	8	EMPS_OUT	PROCEDURE
REC	20A90B4	ITERATOR	REFERENCE	16	10	14	15	EMPS_OUT	PROCEDURE
DBMS_OUTPUT	6CDB513	PACKAGE	REFERENCE	18	11	9	8	EMPS_OUT	PROCEDURE
PUT_LINE	720C63F	PROCEDURE	CALL	19	11	21	18	EMPS_OUT	PROCEDURE
V_STRING	5C35983	VARIABLE	REFERENCE	20	11	30	19	EMPS_OUT	PROCEDURE
STANDARD	26BE90B	PACKAGE	REFERENCE	21	11	53	19	EMPS_OUT	PROCEDURE
V_INT	8721790	VARIABLE	REFERENCE	22	11	70	21	EMPS_OUT	PROCEDURE

Abbildung 2: PL/Scope-Metadaten für die Bezeichner

SIGNATURE	TYPE	USAGE_ID	LINE	COL	USAGE_CONTEXT_ID	SQL_ID	TEXT
3AFC9CE	SELECT	8	7	15	7	0c8jruncvt7tp	SELECT LAST_NAME,

Abbildung 3: PL/Scope-Metadaten für SQL-Statements

Hier findet man zunächst den Namen des Bezeichners, dann eine global eindeutige Signatur (wichtig zum Beispiel zur Unterscheidung überladener Methoden oder bei mehrfacher Verwendung eines Variablennamens in unterschiedlichen Methoden desselben Packages), den Typ des Bezeichners und die – pro Compilation-Unit eindeutige – „USAGE_ID" jedes Bezeichners. „LINE" und „COL" geben die konkrete Fundstelle an und die „USAGE_CONTEXT_ID" die jeweils übergeordnete „USAGE_ID". Diese ist „0" für alle Compilation-Units, also Prozeduren, Funktionen, Packages, Trigger und Synonyme. Warum ist die „USAGE_ID" nicht lückenlos? Bis 12.1 war sie es, was daran lag, dass statische SQL-Statements, die im Code vorkamen, schlichtweg ignoriert wurden – wie auch Kommentare und Leerzeilen. Seit 12.2 werden diese Statements ebenfalls analysiert und können in der neu dazugekommenen View „USER_STATEMENTS" abgefragt werden. Hier finden wir auch unsere „USAGE_ID 8", die zum „SELECT"-Statement gehört *(siehe Abbildung 3)*.

Die Spalten sind größtenteils dieselben wie bei „USER_IDENTIFIERS"; sie umfassen eine globale Signatur, einen Typ (wie „SELECT", „UPDATE", „EXECUTE IMMEDIATE", „COMMIT", „SAVEPOINT", „OPEN" etc.), die „USAGE_ID", „LINE" und „COL" der Fundstelle sowie die „SQL_ID". Zwei gleiche SQL-Statements haben zwar eine unterschiedliche Signatur, lassen sich aber über dieselbe „SQL_ID" finden. Die View enthält noch weitere Spalten, die das jeweilige Statement detaillierter beschreiben. Es gibt zum Beispiel Informationen dazu, ob das Statement einen Hint beinhaltet, ob es „BULK COLLECT INTO" oder „FOR UPDATE" nutzt, ob es eine „RETURNING"-Klausel besitzt oder ob es Bindevariablen enthält.

IDENTIFIERS- und STATEMENTS-View

Über „UNION ALL" lassen sich die beiden Views so verbinden, dass sich über die hierarchische Beziehung zwischen „USAGE_ID" und „USAGE_CONTEXT_ID" ein vollständiger Baum bilden lässt. Dazu eine Anmerkung: Die Implementierung ist möglicherweise noch nicht ganz komplett. In unserem Beispiel fehlt die Funktion „TO_CHAR", was allerdings nicht zu einer Lücke bei „USAGE_ID" führt. Hingegen erzeugen die fehlenden Referenzen der Record-Variablen „rec.last_name" und „rec.salary" in Zeile 9 und 10 offensichtlich eine Lücke *(siehe Abbildung 4)*. Dies wird hoffentlich in kommenden Versionen behoben.

LINE	COL	IDENTIFIER_HIERARCHY	USAGE_ID	USAGE_CONTEXT_ID
1	14	PROCEDURE EMPS_OUT (DECLARATION)	1	0
1	14	PROCEDURE EMPS_OUT (DEFINITION)	2	1
4	3	VARIABLE V_STRING (DECLARATION)	3	2
4	12	CHARACTER DATATYPE VARCHAR2 (REFERENCE)	4	3
5	3	VARIABLE V_INT (DECLARATION)	5	2
5	9	SUBTYPE INTEGER (REFERENCE)	6	5
7	7	ITERATOR REC (DECLARATION)	7	2
7	15	SELECT 0c8jruncvt7tp (STMT)	8	7
7	22	COLUMN LAST_NAME (REFERENCE)	11	8
7	33	COLUMN SALARY (REFERENCE)	10	8
7	45	TABLE EMPLOYEES (REFERENCE)	9	8
9	5	VARIABLE V_STRING (ASSIGNMENT)	12	8
9	17	ITERATOR REC (REFERENCE)	13	12
10	5	VARIABLE V_INT (ASSIGNMENT)	15	8
10	14	ITERATOR REC (REFERENCE)	16	15
11	9	PACKAGE DBMS_OUTPUT (REFERENCE)	18	8
11	21	PROCEDURE PUT_LINE (CALL)	19	18
11	30	VARIABLE V_STRING (REFERENCE)	20	19
11	53	PACKAGE STANDARD (REFERENCE)	21	19
11	70	VARIABLE V_INT (REFERENCE)	22	21

Abbildung 4: Die hierarchische Darstellung der Bezeichner

```
select s.type, s.object_name, s.object_type, s.line
  from user_statements s
 where s.type in ('UPDATE','INSERT','MERGE')
  and exists (select *
         from user_identifiers i
         where s.usage_id = i.usage_context_id
         and s.object_name = i.object_name
         and s.object_type = i.object_type
         and i.name = 'FIRST_NAME'
         and i.type = 'COLUMN')
  and exists (select *
         from user_identifiers j
         where s.usage_id = j.usage_context_id
         and s.object_name = j.object_name
         and s.object_type = j.object_type
         and j.name = 'EMPLOYEES'
         and j.type = 'TABLE')
```

Listing 2: Stellen, an denen die Spalte „EMPLOYEES.FIRST_NAME" geschrieben wird

Beispiel 1: Wo eine bestimmte Tabellenspalte manipuliert wird

Fangen wir gleich mit einem Bespiel an, das die neue Funktionalität der SQL-Analyse nutzt. Die Fragestellung lautet: „Wo wird die Spalte FIRST_NAME der Tabelle EMPLOYEE geändert?" Normalerweise sollte dies nur an wenigen Stellen erfolgen oder noch besser über ein definiertes API, wie einige Evangelisten nicht müde

werden zu predigen, aber in der Praxis werden trotzdem nicht selten SQL-Statements direkt aufgerufen. Dies machen wir uns hier zunutze *(siehe Listing 2)*.

Im Sourcecode stellt man fest, dass tatsächlich an diesen Stellen die gesuchten Statements stehen. Diese können dann als Ausgangspunkt für weitere Analysen dienen *(siehe Abbildung 5)*. Nebenbei angemerkt: Die Zeile 13, die auch eine Referenz auf „FIRST_NAME" enthält, taucht übrigens nicht im Suchergebnis auf. Solche und ähnliche Referenzen wären mit einem klassischen „SELECT * FROM USER_SOURCE" nicht so einfach auszuschließen.

```
12  ,p_PHONE_NUMBER in EMPLOYEES.PHONE_NUMBER%type default null
13  ,p_FIRST_NAME in EMPLOYEES.FIRST_NAME%type default null
14  ,p_COMMISSION_PCT in EMPLOYEES.COMMISSION_PCT%type default null
15  ,p_MANAGER_ID in EMPLOYEES.MANAGER_ID%type default null
16  ) is
17  begin
18    pit.enter('ins');
19  insert into EMPLOYEES(
20  JOB_ID
21  ,EMPLOYEE_ID
22  ,SALARY
23  ,HIRE_DATE
24  ,DEPARTMENT_ID
25  ,LAST_NAME
26  ,EMAIL
27  ,PHONE_NUMBER
28  ,FIRST_NAME
29  ,COMMISSION_PCT
```

Abbildung 5: Der untersuchte Sourcecode mit Vorkommen von FIRST_NAME

Beispiel 2: Naming Conventions checken

Ein kurzes, aber beliebig zu verkomplizierendes Beispiel ist der Check von Namenskonventionen wie in *Listing 3*. Hier hilft die Einschränkung auf die Deklaration, denn dadurch werden nicht nur Doppelungen vermieden, sondern auch alle Bezüge zu Objekten außerhalb der betrachteten Compilation-Unit ausgeblendet, also Tabellen, Spalten, (Standard)-Packages und -Typen.

In *Abbildung 6* sieht man sehr schön, dass die Prozedur „ADD_JOB_HISTORY" und die Konstante „CNT" nicht den Konventionen entsprechen. Über die Verknüpfung von „USAGE_ID" und „USAGE_CONTEXT_ID" lassen sich natürlich wunderbar weitere Sachverhalte abprüfen und der Einsatz von „LIKE"- und „REGEXP"-Funktionen ermöglicht ausgefeiltere Checks.

```
select case i.type when 'VARIABLE'
                   then instr(upper(i.name),'V_')
                   when 'CONSTANT'
                   then instr(upper(i.name),'C_')
                   when 'FUNCTION'
                   then instr(upper(i.name),'FNC_')
                   when 'PROCEDURE'
                   then instr(upper(i.name),'PRC_')
                   when 'FORMAL IN'      -- In-Parameter
                   then instr(upper(i.name),'P_')
            end chk
     , i.name, substr(i.signature,1,7) signature, i.type
  , i.line, u.text user_source_text
 from sys.dba_identifiers i
 join user_source u
     on u.line = i.line and u.name = i.object_name
  and u.type = i.object_type
where i.object_name = 'ADD_JOB_HISTORY' and i.owner = 'HR'
  and i.usage = 'DECLARATION'
order by i.object_name, i.object_type, i.line, i.col
```

Listing 3: Alle Bezeichner auf richtige Benennung prüfen

Unbenutzte Variablen finden

Ein Problem bei der Wartung von Code sind stehengebliebene Reste, die in einer früheren Version einen Sinn hatten, jetzt aber nur noch die Lesbarkeit des Codes verschlechtern. Die Abfrage in *Listing 4* findet solche Variablen (und lässt sich auch auf andere Typen anpassen). Dieses Statement findet einen Treffer in „PRC_UNUSED" *(siehe Abbildung 7)*.

In der Prozedur „PRC_UNUSED" *(siehe Listing 5)* wird die Variable „V_STRING" zwar angelegt und erhält sogar einen Wert; sie wird allerdings nicht weiter benutzt und ist somit überflüssig.

CHK	NAME	SIGNATURE	TYPE	LINE	USER_SOURCE_TEXT
0	ADD_JOB_HISTORY	58A79BD	PROCEDURE	1	procedure add_job_history
1	P_EMP_ID	9B9AD81	FORMAL IN	2	(p_emp_id job_history.employee_id%type
1	P_START_DATE	4F94CA9	FORMAL IN	3	, p_start_date job_history.start_date%type
1	P_END_DATE	D31184C	FORMAL IN	4	, p_end_date job_history.end_date%type
1	P_JOB_ID	7E70403	FORMAL IN	5	, p_job_id job_history.job_id%type
1	P_DEPARTMENT_ID	C2873DF	FORMAL IN	6	, p_department_id job_history.department_id%type
1	C_PROC_NAME	BB863E1	CONSTANT	9	c_proc_name constant varchar2(30) := 'add_job_history';
0	CNT	052F060	VARIABLE	10	cnt pls_integer;

Abbildung 6: Die Guten ins Töpfchen, die Schlechten ...

Beispiel 3: Doppelte Deklarationen von Cursor-Variablen finden

Nehmen wir einen fiktiven Fall an, in dem eine Menge an Code von expliziten auf implizite Cursor umgestellt werden soll, um das lästige (und fehleranfällige) Öffnen und Schließen des Cursors zu vermeiden. Bei der Umstellung wurde allerdings an einigen Stellen vergessen, die Deklaration der Variable für die Ergebniszeilen des Cursors zu löschen. Den Compiler stört das überhaupt nicht, weil für ihn immer klar ist, in welchem Scope welche Deklaration zu verwenden ist. Hier können wir PL/Scope als Hilfsmittel gut gebrauchen.

Listing 6 zeigt, wie unterschiedliche Deklarationen gefunden werden. Die explizite Cursor-Variable erkennt man daran, dass die Variable in ihrem Kontext eine Referenz („REFERENCE") auf den Cursor hat; die implizite Cursor-Variable ist ein Iterator, in dessen Kontext ein Cursor aufgerufen wird („CALL").

Das Ergebnis ist in *Abbildung 8* zu sehen: Wenn die Anzahl unterschiedlicher Deklarationen „größer als 1" ist, weil wir eine Variable als „explizit" und als „implizit" deklariert haben, sollten wir den Code genauer prüfen, so wie in „CURSOR_DEMO_2" und „CURSOR_DEMO_3". In „CURSOR_DEMO_4" und „CURSOR_DEMO_5" gibt es kein Problem, weil hier nur implizite Deklarationen verwendet werden, wie man in der Textspalte rechts sehen kann.

```
select object_name
          , object_type
          , name
          , line
   from user_identifiers u
 where usage = ,DECLARATION'
       and type = ,VARIABLE'
       and not exists (select *
                       from user_identifiers i
                       where i.signature = u.signature
                       and i.usage not in
                           (,DECLARATION',‘ASSIGNMENT'))
```

Listing 4: Variablen suchen, die deklariert, aber nicht genutzt werden

OBJECT_NAME	OBJECT_TYPE	NAME	LINE
PRC_UNUSED	PROCEDURE	V_STRING	4

Abbildung 7: Die Variable ohne Nutzen und ihr Fundort

```
procedure prc_unused as
   cursor cur is select last_name, salary from employees;
   v_string varchar2(2000) := 'Hurz!';
   v_int integer;
begin
   for r in cur
   loop
      v_int := r.salary;
      sys.dbms_output.put_line(r.last_name || ' ' || v_int);
   end loop;
end prc_unused;
```

Listing 5: Die unbenutzte Variable im Kontext

Beispiel 4: Prüfung von paarweisen Methoden-Aufrufen bei der Verwendung von Logging-Frameworks

Bei vielen Logging-Lösungen gibt es eine Methode, die beim Einstieg in einen Abschnitt aufgerufen wird, und eine, die beim Ausstieg aufgerufen wird, sodass eine geschachtelte Aufrufhierarchie entsteht. Fehlt eine der beiden Methoden, gerät dieser (selbstgebaute) Call-Stack durcheinander. Mit PL/Scope lässt sich leicht feststellen, ob zu jedem „Enter" auf gleicher Ebene ein „Leave" existiert. *Listing 7* zeigt ein kurzes Beispiel mit dem PL/SQL Instrumentation Toolkit (PIT) von Jürgen Sieben. Nach dem Kompilieren dieser Prozedur fragen wir alle Methoden im HR-Schema explizit auf die Enter- und Leave-Prozeduren mit dem Statement aus *Listing 8* ab.

Gleich die erste Prozedur „ADD_JOB_HISTORY" fällt unangenehm auf, denn hier wird nur die Enter-Prozedur aufgerufen *(siehe Abbildung 9)*. Die Prozedur „DEL" im Package „EMPLOYEES_TAPI" ist ein möglicher Kandidat für eine genauere Prüfung,

NAME	SIGNATURE	TYPE	OBJECT_NAME	OBJECT_TYPE	USAGE_ID	USAGE_CONTEXT_ID	LINE	COL	ANZAHL	TEXT
REC	DAF57DB	VARIABLE	CURSOR_DEMO_2	PACKAGE BODY	7	1	4	3	2	rec x%rowtype;
REC	8140490	ITERATOR	CURSOR_DEMO_2	PACKAGE BODY	13	1	9	7	2	for rec in x
REC	5DCC946	VARIABLE	CURSOR_DEMO_3	PACKAGE BODY	7	1	4	3	2	rec x%rowtype;
REC	EAB9C18	ITERATOR	CURSOR_DEMO_3	PACKAGE BODY	13	1	9	7	2	for rec in x
REC	E6F3618	ITERATOR	CURSOR_DEMO_3	PACKAGE BODY	25	1	16	7	2	for rec in x
REC	72BD987	ITERATOR	CURSOR_DEMO_4	PACKAGE BODY	11	1	8	7	1	for rec in x
REC	39BC562	ITERATOR	CURSOR_DEMO_4	PACKAGE BODY	23	1	15	7	1	for rec in x
R	D9266F4	ITERATOR	CURSOR_DEMO_5	PROCEDURE	12	2	8	7	1	for r in cur
R	9123C5A	ITERATOR	CURSOR_DEMO_5	PROCEDURE	24	2	15	7	1	for r in cur
R	CBB0631	ITERATOR	CURSOR_DEMO_5	PROCEDURE	33	2	21	7	1	for r in cur

Abbildung 8: Gleichzeitige Verwendung eines Bezeichners für explizite und implizite Cursor-Variablen (rot)

da hier das Logging entweder vergessen oder absichtlich nicht eingebaut wurde. Die Prozeduren „INS“ und „UPD“ wurden ordnungsgemäß behandelt, die Prozedur „EMPS_OUT“ (siehe oben) besitzt offensichtlich gar kein Logging und die Prozedur „EMPS_OUT_LOG“ ist wie beabsichtigt mit Enter und Leave versorgt. Natürlich ließe sich hier mit noch mehr SQL auch sicherstellen, dass die Aufrufe an der richtigen Stelle stehen – der Ansatz ist beliebig ausbaufähig.

Beispiel 5: Öffentlich deklarierte Variablen finden

Eine Anleihe bei Steven Feuerstein ist das Statement zum Auffinden von Variablen, die in der Package-Spezifikation deklariert sind. Dies sollte aus Sicherheitsgründen nur im Package-Body erfolgen, sodass der Zugriff nur über definierte Methoden möglich ist. Führt man das Statement aus *Listing 9* aus, bekommt man ein Ergebnis wie in *Abbildung 10*.

Im Quellcode in *Listing 10* sieht man nichts Verbotenes, sondern nur eine ganz normale „RECORD“-Deklaration. Offensichtlich werden die Felder des Record-Typs ebenfalls dem Typ „VARIABLE“ zugeordnet. Das heißt, das Statement aus *Listing 9* muss noch ein wenig erweitert werden, um diese Fälle auszufiltern, wie man in *Listing 11* sieht.

Führt man das Statement in *Listing 11* aus, erhält man keine falsch-positiven Ergebnisse mehr. Eine solche Erweiterung ist übrigens auch für das Beispiel „Naming Conventions“ notwendig – es sei denn, es ist wirklich beabsichtigt, dass die Felder des Record mit „v_“ beginnen sollen wie die normalen Variablen.

Die Nachteile von PL/Scope

Natürlich gibt es all diese schönen Möglichkeiten nicht zum Nulltarif, aber die Nachteile beim Einsatz von PL/Scope sind schnell aufgezählt. An erster Stelle steht der Platzverbrauch, denn natürlich verbrauchen die Metadaten etwas Platz in der Datenbank. Dieser lässt sich mit „select space_usage_kbytes from v$sysaux_occupants where occupant_name='PL/SCOPE'“ abfragen. Die Dokumentation sagt nichts Konkretes über den potenziellen Platzverbrauch. Nach Beobachtungen in diversen Umgebungen scheint 1 KB pro Codezeile die obere Grenze zu sein. Es ist allerdings fraglich, ob die Anzahl der Codezeilen als Basis für die Berechnung taugt; es seien nur die Stichwörter „Formatierung“, „Kommentare“ und „Abhängigkeiten“ erwähnt, die diese Ratio beeinflussen.

```
with explizit as (select ui1.*
                    from user_identifiers ui1
                    left outer join user_identifiers ui2
                    on ui1.usage_id = ui2.usage_context_id
                    and ui1.object_name = ui2.object_name
                    and ui1.object_type = ui2.object_type
                    where ui1.type = 'VARIABLE'
                    and ui1.usage = 'DECLARATION'
                    and ui2.type 'CURSOR'
                    and ui2.usage = 'REFERENCE‘)'
      implizit as (select ui1.*
                    from user_identifiers ui1
                    left outer join user_identifiers ui2
                    on ui1.usage_id = ui2.usage_context_id
                    and ui1.object_name = ui2.object_name
                    and ui1.object_type = ui2.object_type
                    where ui1.type = 'ITERATOR'
                    and ui1.usage = 'DECLARATION
                    and ui2.type = 'CURSOR'
                    and ui2.usage = 'CALL')
select d.name, substr(d.signature,1,7) signature
      , d.type, d.object_name, d.object_type
      , d.usage_id, d.usage_context_id, d.line, d.col
      , count (distinct d.type)
          over (partition by d.object_name
                    , d.object_type
                    , d.usage_context_id) anzahl
      , s.text
from (select * from explizit
      union
      select * from implizit) d
join user_source s
  on s.name = d.object_name
 and s.type = d.object_type
 and s.line = d.line
order by d.object_name, d.object_type, d.usage_id
```

Listing 6: Überflüssige Cursor-Variablen finden

```
procedure hr.emps_out_log
   authid definer
is
   v_string varchar2(2000);
   v_int integer;
begin
   pit.enter('emps_out'); -- neue Ebene im Aufruf-Stack
   for rec in (select last_name, salary from employees)
   loop
      v_string := rec.last_name;
      v_int := rec.salary;
      sys.dbms_output.put_line(v_string || ' '
                  || sys.standard.to_char(v_int,99999));
   end loop;
   pit.leave; -- Prozedur vom Stack entfernen
exception
   when others then
      pit.sql_exception; -- pit.leave + Exception Handling
end;
```

Listing 7: Einfaches Beispiel für den Einsatz eines Logging-Frameworks

```
with methoden as (
   select owner, object_name, object_type, usage_id
       , type, name, usage_context_id
   from dba_identifiers
   where owner = 'HR'
       and type   in ('PROCEDURE','FUNCTION')
       and usage = 'DEFINITION'
)
select m.owner, m.object_name, m.object_type
       , m.name method_name
       , i.name log_package, j.name log_method
       , count (distinct j.name)
       over (partition by m.owner, m.object_name
              , m.object_type, m.name) ok
   from methoden m
   left outer join dba_identifiers i
       on i.object_name = m.object_name
       and i.object_type = m.object_type
       and i.name = 'PIT' and i.type = 'SYNONYM'
       and i.usage_context_id = m.usage_id
       left outer join dba_identifiers j
       on j.object_name = m.object_name
       and m.object_type = j.object_type
       and j.name in ('ENTER','LEAVE')
       and j.usage_context_id = i.usage_id
 order by m.owner, m.object_name, m.object_type, m.name
          , i.usage, i.object_name, j.usage, j.name
```

Listing 8: Aufrufe der Logging-Methoden „PIT.ENTER“ und „PIT.LEAVE“ prüfen

O...	OBJECT_NAME	OBJECT_TYPE	METHOD_NAME	LOG_PACKAGE	LOG_METHOD	OK
HR	ADD_JOB_HISTORY	PROCEDURE	ADD_JOB_HISTORY	PIT	ENTER	1
HR	EMPLOYEES_TAPI	PACKAGE BODY	DEL			0
HR	EMPLOYEES_TAPI	PACKAGE BODY	INS	PIT	LEAVE	2
HR	EMPLOYEES_TAPI	PACKAGE BODY	INS	PIT	ENTER	2
HR	EMPLOYEES_TAPI	PACKAGE BODY	UPD	PIT	LEAVE	2
HR	EMPLOYEES_TAPI	PACKAGE BODY	UPD	PIT	ENTER	2
HR	EMPS_OUT	PROCEDURE	EMPS_OUT			0
HR	EMPS_OUT_LOG	PROCEDURE	EMPS_OUT_LOG	PIT	ENTER	2
HR	EMPS_OUT_LOG	PROCEDURE	EMPS_OUT_LOG	PIT		2
HR	EMPS_OUT_LOG	PROCEDURE	EMPS_OUT_LOG	PIT	LEAVE	2

Abbildung 9: Sind die Logging-Methoden wie beabsichtigt verwendet worden?

```
select object_name, name, line
   from user_identifiers ai
 where ai.type = 'VARIABLE'
       and ai.usage = 'DECLARATION'
       and ai.object_type = 'PACKAGE'
 order by object_name, line
```

Listing 9: Suche nach öffentlich deklarierten Variablen

OBJECT_NAME	NAME	LINE
EMPLOYEES_TAPI	JOB_ID	8
EMPLOYEES_TAPI	EMPLOYEE_ID	9
EMPLOYEES_TAPI	SALARY	10
EMPLOYEES_TAPI	HIRE_DATE	11
EMPLOYEES_TAPI	DEPARTMENT_ID	12
EMPLOYEES_TAPI	LAST_NAME	13

Abbildung 10: Öffentlich deklarierte Variablen in generiertem Code?

Der zweite Faktor, der sich verändert, ist die Compile-Zeit. Im Kurztest ergab sich eine Verlängerung zwischen 10 und 20 Prozent. Ausdrücklich sei aber darauf hingewiesen, dass die Laufzeit gänzlich unbeeinflusst bleibt. Alles in allem handelt es sich also um sehr überschaubare und ohnehin nur temporäre Einschränkungen, die die Produktionsumgebung überhaupt nicht betreffen.

Grenzen von PL/Scope

Es gibt einige Ausnahmen bei der Generierung der Metadaten, an denen sich auf absehbare Zeit wohl auch nichts ändern wird; diese umfassen:

- Wrapped Code
- Anonyme Blöcke
- Dynamisches SQL

Zudem beschränkt sich die Analyse von SQL-Statements auf recht grundlegende Informationen. Will man hier eine wirklich detaillierte Analyse starten und etwa Data-Lineage betreiben, kommt man nicht darum herum, einen weiteren Parser einzusetzen.

Ein Aspekt, den man im Hinterkopf behalten sollte, ist, dass PL/Scope quasi blind für anonyme Blöcke im Code ist. Die Usage-Hierarchie für eine Prozedur mit einem Label vor einem Block sieht also anders aus als für denselben Code ohne Label, da das Label eine eigene Usage-ID erhält und alle Bezeichner in diesem Block eine Hierarchie-Ebene tiefer rutschen.

Selbermachen

Die Hauptquellen zu PL/Scope sind überschaubar und lassen sich sehr schnell googeln: Steven Feuerstein hat einige Artikel geschrieben und auf LiveSQL *(siehe „http://livesql.oracle.com“)* einige „Beispiele to go“ bereitgestellt. Zusätzlich ist ein Blick in die Doku empfohlen („Using PL/Scope“) und ein weiterer Blick zu GitHub, wo die hier benutzten Statements mit Beispielcode abgelegt sind *(siehe „https://github.com/its-people/plscope2)* und wo Philipp Salvisberg einige sehr nützliche Views und Packages für die Arbeit mit PL/Scope bereitstellt *(siehe „https://github.com/PhilippSalvisberg/plscope-utils“)*.

```
package employees_tapi
is
type employees_tapi_rec is record (
      job_id employees.job_id%type
      , employee_id employees.employee_id%type
      , salary employees.salary%type
      , hire_date employees.hire_date%type
      …
```

Listing 10: Deklaration eines Record-Typs

```
select object_name, name, line
   from user_identifiers ai
 where ai.type = 'VARIABLE'
   and ai.usage = 'DECLARATION'
   and ai.object_type = 'PACKAGE'
   and not exists
       (select * from user_identifiers ui
          where ui.object_type = ai.object_type
              and ui.object_name = ai.object_name
              and ui.usage_id = ai.usage_context_id
              and ui.type = ‚RECORD')
 order by object_name, line
```

Listing 11: Im übergeordneten Kontext prüfen, ob es sich um einen „RECORD“ handelt

Fazit

PL/Scope ist in der aktuellen Version 12.2 noch nicht ganz frei von einigen Merkwürdigkeiten, bietet aber umfangreiche Basis-Informationen für viele Anwendungsfälle. Ein paar davon wurden hier angerissen; in der Praxis stößt man mit Sicherheit auf viele weitere Ideen, wie sich PL/Scope in seinen Entwicklungsumgebungen einsetzen lässt. Die Autorin freut sich, Berichte darüber zu hören oder zu lesen.

Sabine Heimsath
sabine.heimsath@its-people.de

Transparent Data Encryption: Tablespace Live Conversion

Roland Zerfass, Muniqsoft GmbH

Mit steigender Anzahl der Anwendungen, die eine 24/7-Verfügbarkeit erfordern, wird die Minimierung von Ausfallzeiten ein immer wichtigeres Anliegen. Datenbanken in der Oracle Cloud verlangen, dass Produktions-Datenbanken mit Oracle Advanced Security Transparent Data Encryption (TDE) verschlüsselt sind. Dieser Artikel betrachtet einige Aspekte, die die neue Funktionalität der Tablespace-Live-Verschlüsselung mit Zero Downtime aufzeigen. Sie ist damit ein äußerst effektives Mittel zur Reduzierung von Ausfallzeiten.

Die TDE Tablespace Live Conversion bietet die Möglichkeit, existierende Tablespaces zu verschlüsseln („ENCRYPT"), zu entschlüsseln („DECRYPT") und neu zu verschlüsseln („REKEY"). Die Verschlüsselung auf den Tablespace-Daten erfolgt im Hintergrund, sodass der Tablespace weiterhin für SQL-Anweisungen zur Verfügung steht („SELECT", „INSERT", „UPDATE", "DELETE" etc.).

Voraussetzungen für die TDE-Tablespace-Verschlüsselung

Grundsätzlich gilt, dass der „COMPATIBLE"-Parameter auf mindestens 12.2.0.0 stehen muss. Zur Verschlüsselung der normalen Tablespaces reicht die „SYSKM"-Rolle; für die Tablespaces „SYSTEM" und „SYSAUX" ist das „SYSDBA"-Privileg erforderlich *(siehe Listing 1)*.

TDE verwendet Wallets beziehungsweise Keystores, um den Master-Encryption-Key zu speichern. Obwohl das Default-Database-Wallet verwendet werden kann, empfiehlt Oracle einen eigenen Keystore für TDE unter Einsatz des „ENCRYPTION_WALLET_LOCATION"-Parameters in der „sqlnet.ora"-Datei.

Neben einem Passwort-basierten Wallet gibt es auch die Möglichkeit, mit einem (Local) Auto-Login-Wallet beim Neustart der Datenbank das Wallet automatisch zu öffnen.

Das folgende Beispiel dient zur Demonstration einer Passwort-basierten Online-Verschlüsselung eines Tablespace. Die weiteren Möglichkeiten der Entschlüsselung („DECRYPT") beziehungsweise einer Schlüsselerneuerung („REKEY") werden hier nicht betrachtet.

```
COMPATIBLE              >= 12.2.0.0
SYSTEM, SYSAUX TBS         SYSDBA
Andere Tablespaces         SYSKM
```

Listing 1

```
SQL> SELECT wrl_type, wrl_parameter, status, wallet_type
  FROM v$encryption_wallet;
WRL_TYPE      WRL_PARAMETER                       STATUS              WALLET_TYPE
----------    ----------------------------------- ------------------- -------------
FILE          /u01/app/oracle/admin/o12c/wallet   NOT_AVAILABLE       UNKNOWN
```

Listing 2

Schritt für Schritt zur TDE Tablespace Live Conversion

Folgende Schritte sind erforderlich:

1. Das Keystore-Verzeichnis in der „SQLNET.ORA“-Datei festlegen
2. Den Software-Keystore erzeugen
3. Den Software-Keystore öffnen
4. Den Software-TDE-Master-Encryption-Key setzen
5. Die Daten verschlüsseln

Schritt 1

Zunächst wird ausgelesen, in welchem Verzeichnis per Default das Wallet/der Keystore erwartet wird *(siehe Listing 2)*. Anschließend legt man das gewünschte Wallet-/Keystore-Verzeichnis in der „SQLNET.ORA“-Datei fest. *Listing 3* zeigt ein Beispiel.

```
ENCRYPTION_WALLET_LOCATION=
   (SOURCE=
      (METHOD=FILE)
         (METHOD_DATA=
            (DIRECTORY=/u01/app/oracle/admin/o12c/wallet)))
```

Listing 3

Schritt 2

Nun wird der Software-Keystore erzeugt. Um diesen Schritt durchführen zu können, ist das „SYSKM“-Privileg beziehungsweise das Recht „ADMINISTER KEY MANAGEMENT“ erforderlich. *Listing 4* zeigt die Syntax für einen Passwort-basierten Keystore und *Listing 5* ein Beispiel dafür.

Durch diesen Schritt wird im Keystore-Verzeichnis eine Datei, der Keystore, mit dem Namen „ewallet.p12“ angelegt. Die Verschlüsselung ist zum jetzigen Zeitpunkt noch nicht wirksam. Erst durch das Öffnen des Keystore selbst wird die Verschlüsselung aktiviert. „V$ENCRYPTION_WALLET“ gibt Auskunft darüber, ob das Wallet/der Keystore geöffnet oder geschlossen ist *(siehe Listing 6)*.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '<keystore_location>'
IDENTIFIED BY <software_keystore_password>;
```

Listing 4

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/u01/app/oracle/admin/
o12c/wallet' IDENTIFIED BY strenggeheim;
Keystore geändert.
```

Listing 5

```
SQL> SELECT wrl_type, wrl_parameter, status, wallet_type
  FROM v$encryption_wallet;
WRL_TYPE      WRL_PARAMETER                       STATUS  WALLET_TYPE
----------    ----------------------------------- ------  -----------
FILE          /u01/app/oracle/admin/o12c/wallet   CLOSED  UNKNOWN
```

Listing 6

Schritt 3

Als Nächstes wird der Software-Keystore geöffnet. *Listing 7* zeigt die Syntax zum Öffnen des Passwort-basierten Keystore und *Listing 8* ein Beispiel dazu. Hinweis: Das Passwort kann auch in Anführungszeichen stehen.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <soft-
ware_keystore_password>;
```

Listing 7

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
strenggeheim;
```

Listing 8

„V$ENCRYPTION_WALLET“ gibt Auskunft darüber, dass das Wallet/der Keystore zwar geöffnet wurde, aber noch kein Master-Key erzeugt ist *(siehe Listing 9)*. Gleichzeitig kann man hier den jeweiligen Wallet-Typ entnehmen („password_based“, „auto_login" beziehungsweise „local auto_login“).

Schritt 4

Der Master-Encryption-Key ist ebenso im Oracle-Wallet gespeichert und dient zum Schutz der Column-Encryption-Keys. Stan-

```
SQL> SELECT wrl_type, wrl_parameter, status, wallet_type
  FROM v$encryption_wallet;
WRL_TYPE WRL_PARAMETER                    STATUS                WALLET_TYPE
-------- -------------------------------- --------------------  -----------
FILE     /u01/app/oracle/admin/o12c/wallet/   OPEN_NO_MASTER_KEY    PASSWORD
```

Listing 9

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY [USING TAG 'tag'] IDENTIFIED BY keystore-password WITH BACKUP USING
'backup_identifier' [CONTAINER=ALL | CURRENT];
```

Listing 10

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY strenggeheim WITH BACKUP USING 'backup MENCKey';
Keystore geändert.
```

Listing 11

```
[root@xxx wallet]# ls
ewallet_2017030114492434_backup MENCKey.p12
ewallet.p12
```

Listing 12

```
SQL> ALTER TABLESPACE testtbs ENCRYPTION ONLINE ENCRYPT file_name_convert = ('/u01/app/oracle/oradata/o12c/
testtbs01.dbf',
'/u01/app/oracle/oradata/o12c/testtbs01enc.dbf')

SQL> SELECT ts#, encryptionalg, encryptedts, encryptedkey,
            masterkeyid, blocks_encrypted, con_id
      FROM v$encrypted_tablespaces;
TS#   ENCRYPT ENC ENCRYPTEDKEY                     MASTERKEYID                      BLOCKS_ENCRYPTED CON_ID
---   ------- --- -------------------------------- -------------------------------- ---------------- -------
 7    AES128  YES 7F9968F4FE9287E244CF37D39B2ACDE8 F473DC8689644FA3BF1D7EF3D3412817          12925       1
```

Listing 13

dardmäßig ist der Master-Encryption-Key ein zufälliger Schlüssel, den TDE erzeugt. Es kann auch ein bestehendes Schlüsselpaar aus einem PKI-Zertifikat sein, das für die Verschlüsselung bestimmt ist. Um den TDE-Master-Encryption-Key setzen zu können, muss der Passwort-basierte Software-Keystore geöffnet sein. Um bei Multitenant-Datenbanken den TDE-Master-Encryption-Key zu setzen, loggt man sich in der „CDB$ROOT“ oder einer PDB ein. Der Benutzer braucht das „SYSKM“-Privileg und die Datenbank muss „READ WRITE“ sein. *Listing 10* zeigt die Syntax und *Listing 11* ein Beispiel.

„USING TAG 'tag'“ erlaubt, eigene Informationen zu hinterlegen, und die Klausel „WITH BACKUP“ ist bei Passwort-basierten Keystores verpflichtend, sonst kommt es zur Fehlermeldung „ORA-46631: Backup von Keystore muss angefertigt werden“. Oracle legt daraufhin im Keystore-Verzeichnis „/u01/app/oracle/admin/o12c/wallet/“ die Backup-Datei mit dem gewünschten Namen an *(siehe Listing 12)*.

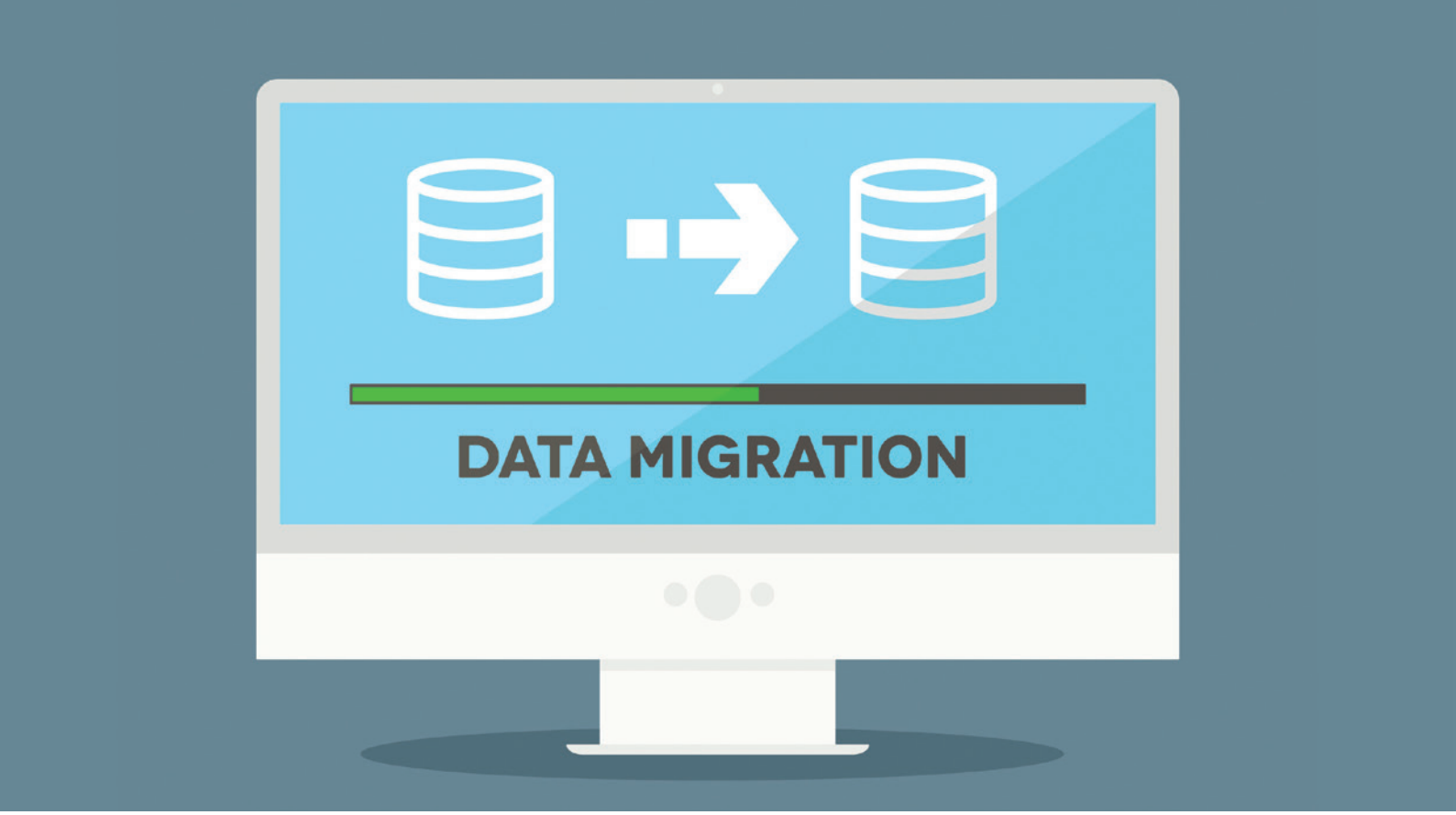
Schritt 5

Listing 13 zeigt die Verschlüsselung der Daten am Beispiel des „TESTTBS“-Tablespace.

Fazit

Dieses neue Feature in der Datenbank-Version 12.2 stellt eine interessante und einfache Möglichkeit dar, existierende Tablespaces ohne Ausfallzeiten zu verschlüsseln.

Roland Zerfaß
roland.zerfass@muniqsoft.de



Migration ohne Downtime – ein 12.2-New-Feature

Peter Sorger, DBConcepts GmbH

Mit dem Release 2 der Oracle-Datenbank 12c bekommt der DBA eine Vielfalt neuer Funktionalitäten, die ihm die Arbeit um einiges erleichtern. Eines dieser neuen Features ist das Hot Cloning und Relocate von Pluggable Databases.

Die Oracle-Datenbank 12c Release 2 bringt mit der Multitenant-Architektur viele tolle Features. Eine Migration ohne Downtime ist mehr ein Wunsch als Wirklichkeit, aber mit der Version 12.2 kommen wir dem Ziel näher. Der Artikel zeigt die Möglichkeit, Pluggable Databases online zu kopieren beziehungsweise zu migrieren.

Ein Blick in die Vergangenheit

In der Version 11.2 hatte der Datenbank-Administrator folgende Möglichkeiten, um eine Datenbank von einem System auf ein anderes zu migrieren:

- Duplicate
- Datapump
- Restore/Recover via RMAN
- GoldenGate

Welchen Weg man letztendlich wählt, hängt ganz von der Situation und den Voraussetzungen ab. Es geht hier nicht um Cross-Platform-Migrationen oder um Migrationen zwischen Systemen mit unterschiedlicher Endianness. Jede Option verlangt gewisse Vorbereitungen und birgt auch mögliche verborgene Gefahren mit sich, die dann erst im Zuge der Migration sichtbar werden.

Bei einem Duplicate kann der Benutzer entweder direkt über SQL*Net oder aus einem vorhandenen Backup heraus eine Kopie der Datenbank erstellen. Die Duplicate-Methode ist die einfachste, weil man hier entweder nur Firewall-Freischaltungen zwischen den Servern benötigt oder alternativ auf ein Backup zugreift, meistens von einem NFS-Share oder einer Bandsicherung.

Wenn zwischen den Servern ein leistungsstarkes Netzwerk zur Verfügung steht, dann führt man am besten den Befehl „DUPLICATE TARGET DATABASE TO ... FROM ACTIVE DATABASE!“ aus oder benutzt Backups, die entweder in einer „BACKUP LOCATION“ gespeichert sind oder aus Bandsicherungen gezogen werden. Oracle automatisiert da-

bei einige Tätigkeiten, die man bei Restores sonst manuell machen muss. Grundsätzlich ist es ein automatischer Restore/Recover der Quell-Datenbank mit Veränderungen der Datafile-Pfade und der DBID.

Die Variante „Datapump" ermöglicht es dem Datenbank-Administrator, logische Datenbank-Backups zu erstellen, auf ein neues Ziel zu übertragen und dort wieder zu importieren. Datapump bietet auch Features wie Transportable Tablespace oder Full Transportable Export. Dabei werden nur Metadaten exportiert, diese dann mit allen Datafiles kopiert und am Ziel importiert.

Bei der klassischen Migration einer Datenbank mittels RMAN Restore/Recover wird eine Datenbank auf Basis eines Backups neu aufgebaut. Als letzte Option bleibt GoldenGate. Dabei handelt es sich um eine kostenpflichtige Replikationssoftware, die es erlaubt, nahezu Zero-Downtime-Migrationen durchzuführen. GoldenGate unterstützt zudem auch noch andere Datenbank-Anbieter wie MySQL, MS SQL und Hadoop.

Mit der Version 12.1 hat Oracle eine neue Datenbank-Architektur eingeführt - die Multitenant-Architektur (CDB). Diese wird in Zukunft die derzeitige Non-CDB-Architektur ersetzen. In der 12.1-Dokumentation findet man bereits den entsprechenden Hinweis auf die „Deprecation of Non-CDB Architecture" für kommende Oracle-Releases nach 12.2. Oracle empfiehlt hier, auf die CDB-Architektur zu schwenken. Multitenant ist prinzipiell eine kostenpflichtige Option zur Enterprise Edition, jedoch als Single-Tenant in allen Oracle-Editionen enthalten.

Während es vor 12.1 nur eine „1:1"-Beziehung zwischen Instanz und Datenbank gab (beim RAC ist es eine „n:1"-Beziehung), verändert sich das jetzt mit der Multitenant-Architektur. Hier werden alle gemeinsamen Systemobjekte in eine neue, zentrale Datenbank – die Container-Datenbank (CDB) – gepackt, die alle notwendigen Funktionalitäten bereits vorinstalliert hat. Die eigentlichen Benutzerdaten befinden sich in sogenannten „Pluggable Databases" (PDB), die in die Container-Datenbank eingehängt werden.

In der Version 12.1 konnte man Datenbanken, die bereits als Pluggable Database definiert waren, wesentlich einfacher kopieren als noch in älteren Versionen ohne Multitenant-Architektur. Man musste die Quell-Datenbank lediglich in den „Read only"-Modus setzen und anschließend ein „CREATE PLUGGABLE DATABASE"-Statement ausführen. Im einfachsten Fall bekam man mit „CREATE PLUGGABLE DATABASE ORCL2 FROM ORCL1" bereits eine lokale Kopie der Datenbank. Mit der Multitenant-Architektur bringt Oracle dem Benutzer eine Vielfalt von Features, die das Provisionieren von Datenbanken leichter machen. Man erhält eine Konsolidierungsplattform für Datenbanken, in der man mit dem Resource Manager System-Ressourcen optimal zuteilen kann, um wichtigeren Datenbanken mehr Performance zuzusichern.

Hinzu kommt eine Automations-Plattform zur Vordefinition von Datenbank-Templates, um schnell PDBs zu erstellen oder klonen zu können. Das Kopieren von PDBs geht in 12.1 nicht ganz ohne Downtime ab. Im Internet findet man Artikel, die ohne den „Read only"-Modus eine Kopie der PDBs ermöglichen; in 12.1 ist es jedoch nicht möglich, PDBs online zu kopieren. Die Quell-Datenbank geht hier in den Quiesce-Modus, was mit einer Downtime vergleichbar ist.

```
sys@cdb1 SQL> show pdbs

    CON_ID CON_NAME                        OPEN MODE  RESTRICTED
---------- ------------------------------  ---------- ----------
         2 PDB$SEED                        READ ONLY  NO
         3 ORCL                            MOUNTED
sys@cdb1 SQL> alter pluggable database orcl open ;

Pluggable database altered.

sys@cdb2 SQL> show pdbs

    CON_ID CON_NAME                        OPEN MODE  RESTRICTED
---------- ------------------------------  ---------- ----------
         2 PDB$SEED                        READ ONLY  NO
```

Listing 1

```
sys@cdb1 SQL> create user c##dba identified by oracle container=all ;

User created.

sys@cdb1 SQL> grant create session, create pluggable database to c##dba
container=all ;

Grant succeeded.

sys@cdb1 SQL> grant sysoper to c##dba container=all ;

Grant succeeded.
```

Listing 2

```
CDB1 =
   (DESCRIPTION =
       (ADDRESS = (PROTOCOL = TCP)(HOST = ora01)(PORT = 1521))
       (CONNECT_DATA =
           (SERVER = DEDICATED)
           (SERVICE_NAME = cdb1)
       )
   )
```

Listing 3

Die Gegenwart: Hot Cloning und Relocate

Mit der Version 12.2 wird der Traum des DBAs endlich wahr: die von vielen erwartete Lösung für das Online-Kopieren von PDBs. Sie kommt zusammen mit einem anderen neuen Feature, den Local Undo Ta-

blespaces. Um Datenbanken wirklich online kopieren zu können, braucht man die Container-Datenbank im Local Undo Mode (und natürlich im Archivelog Mode). Das Aktivieren des Local Undo Mode in der CDB erstellt automatisch in jeder PDB einen eigenen Undo Tablespace, der wiederum Voraussetzung für Flashback, PDBPITR und Hot Cloning in der PDB ist.

Das RELOCATE-Feature transferiert die Datenbank automatisch auf eine andere Instanz; alle DMLs und DDLs bis zum Öffnen der Pluggable-Datenbank werden automatisch im Hintergrund verarbeitet, um die Datenbank konsistent zu halten. In unserem Szenario haben wir zwei virtuelle Maschinen, „ora01" und „ora02", mit der Enterprise-Edition-Version 12.2.0.1 installiert; die „ora01" verwaltet die „cdb1" und „ora02" die „cdb2". Beide Datenbanken sind im gleichen Netz, als Storage benutzen wir ASM, das erleichtert auch die Namenskonvention für die Datafiles. Diese Funktionalität (mit und ohne RELOCATE) funktioniert auch in der Standard Edition 2, Version 12.2. Um keine Lizenzverletzung zu begehen, ist daran zu denken, dass in einer CDB nur eine Benutzer-PDB existieren darf, mehrere davon verlangen die Multitenant-Option und damit die Enterprise Edition *(siehe Listing 1)*.

Um die Datenbank auf die neue Maschine zu transferieren, sind ein paar Vorbereitungen nötig. Zuerst ist ein Common-User anzulegen, der die notwendigen Privilegien bekommt *(siehe Listing 2)*. Ein in der Dokumentation entdeckter Fehler ist am Ende des Beispiels beschrieben.Um die PDB zu erreichen, brauchen wir einen Connect-Descriptor, der die Datenbank identifiziert. In unserem Beispiel werden wir den Alias „CDB1" in der „tnsnames.ora" der „CDB2" anlegen *(siehe Listing 3)*. Als vorletzte Voraussetzung brauchen wir einen Public-Database-Link von der CDB2 in die CDB1 *(siehe Listing 4)*.

Nachdem wir alle Voraussetzungen erfüllt haben, können wir die Datenbank auf die CDB2 migrieren. Diese Tätigkeit erledigt man mit dem Kommando „create pluggable database orcl from orcl@cdb1 relocate ;" auf der CDB2. Nach dem Erstellen der PDB bleibt die neue PDB in der CDB2 im Zustand „MOUNTED". Ohne den Zusatz „RELOCATE" würden wir einen Hot-Clone der PDB „orcl" erstellen. Um die Near-Zero-Downtime-Migration zu zeigen, erstellen wir in der PDB „orcl" auf CDB1, die immer noch im Read-Write-Modus ist, einen neuen Benutzer und eine Tabelle *(siehe Listing 5)*.

```
sys@cdb2 SQL> create public database link cdb1 connect to c##dba iden-
tified by oracle using 'cdb1' ;

Database link created.
```

Listing 4

```
sys@cdb1 SQL> alter session set container=orcl ;

Session altered.

sys@cdb1 SQL> create tablespace data ;

Tablespace created.

sys@cdb1 SQL> create user psorger identified by oracle default tables-
pace data quota unlimited on data;

User created.

sys@cdb1 SQL> grant connect, resource to psorger ;

Grant succeeded.

sys@cdb1 SQL> create table PSORGER.EMP as select level id from dual
connect by level<=10 ;

Table created.
```

Listing 5

```
sys@cdb2 SQL> alter pluggable database orcl open ;

Pluggable database altered.

sys@cdb2 SQL> alter session set container=orcl ;

Session altered.

sys@cdb2 SQL> select count(*) from psorger.emp ;

   COUNT(*)
-----------
          10
```

Listing 6

Nachdem wir die Tabelle erstellt haben, können wir die neue PDB in der CDB2 öffnen. Das Kommando in *Listing 6* löscht die PDB auf CDB1 und öffnet die PDB auf CDB2 im Read-Write-Modus.

Nach dem Öffnen der neuen PDB sehen wir, dass alle DML/DDL-Statements erfolgreich übertragen wurden. Um den Fehler in der Dokumentation zu beschreiben, wird dem „C##DBA" die Rolle „SYSOPER" entzogen und „SYSDBA" vergeben. Mit dieser Rolle vergibt man Superprivilegien und logischerweise erwartet man, dass das funktionieren wird *(siehe Listing 7)*.

Wie wir hier sehen, ist ein Online-Clone der Datenbank möglich, aber beim Relocate bekommt der „C##DBA" mit „SYSDBA"-Privilegien dennoch einen „Insufficient Privileges"-Fehler. Es scheint, dass man dieses Verhalten als einen Bug entweder in der Dokumentation oder in der Software bezeichnen kann.

Um den Artikel abzurunden, seien noch die neuen Features der Oracle Public Cloud erwähnt. Der direkte SQL*Net-Zugang in die Cloud erleichtert ab 12.2 die Migration

```
sys@cdb1 SQL> revoke sysoper from C##DBA container=all ;

Revoke succeeded.

sys@cdb1 SQL> grant sysdba to C##DBA container=all ;

Grant succeeded.

sys@cdb2 SQL> select sysdate from dual@cdb1 ;

SYSDATE
---------
09-MAY-17

sys@cdb2 SQL> create pluggable database orcl from orcl@cdb1 ;

Pluggable database created.

sys@cdb2 SQL> create pluggable database orcl1 from orcl@cdb1 relocate ;
create pluggable database orcl1 from orcl@cdb1 relocate
*
ERROR at line 1:
ORA-17628: Oracle error 1031 returned by remote Oracle server
ORA-01031: insufficient privileges
```

Listing 7

von On-Premise-Datenbanken in die Cloud und zurück. Mit diesen zwei neuen Features bekommt der Benutzer die Möglichkeit, sehr einfach in die Cloud zu wechseln, dort etwa Performance-Tests durchzuführen und dann die Datenbank wieder zurück ins eigene Rechenzentrum zu verschieben. Gerade das ist ein Feature, das bei anderen Cloud-Anbietern aktuell kaum zu finden ist.

Ing. Peter Sorger
peter.sorger@dbconcepts.at

Aus der Ferne betrachtet:
Oracle12*c* Release 2 – aber bitte diskriminierungsfrei!

Günther Stürner
dbms publishing

Das zweite Release der Oracle-Datenbank 12*c* wurde im März 2017 für Linux und in den folgenden Wochen für weitere Plattformen freigegeben. Dies ging erstaunlich ruhig, fast heimlich über die Bühne. Auch deshalb erstaunlich, weil es sich bei diesem Release um einen richtig großen Wurf handelt und nicht nur um Bug-Fixes und kleinere Erweiterungen oder Vervollständigungen von Release-1-Funktionen. Oracle 12*c* Release 2 ist ein großes Ding und richtig gut. Mehr als vierhundert einzelne Projekte wurden definiert und in die Datenbank eingebaut. In allen Bereichen hat man massiv aufgerüstet, bis hin zu einer (für Oracle) völlig neuen Architektur für massive Skalierung. Ja, man hätte eigentlich erwartet, dass Oracle dieses Release offensiver seiner Stammkundschaft offeriert.

Der aufmerksame Oracle-Beobachter wird nun einwenden, dass Oracle 12*c* Release 2 ja bereits auf der Open World 2016 im September groß angekündigt wurde und jeder diese Software als Datenbank-Cloud-Service seit Herbst letzten Jahres nutzen konnte. „Oracle-Software zuerst in der Cloud, irgendwann später dann für die Kunden, die noch konventionell unterwegs sind" ist in Kurzform, was sich hinter dem von Oracle genutzten Begriff „Cloud First" verbirgt.

Dass Oracle diverse Cloud-Services und eine eigene Cloud-Infrastruktur anbietet, ist richtig und nachvollziehbar. Hier entsteht ein großer und wichtiger Markt, den Oracle nicht links liegen lassen kann. Ob allerdings die „Cloud First"-Strategie der Weisheit letzter Schluss ist, möchte ich – bei allem Respekt – bezweifeln. Es sollte Oracle ein wichtiges Anliegen sein, dass die große Kundenbasis in den Genuss einer neuen Version kommt, wenn alle Tests positiv durchlaufen sind und sie den Produktionsstempel bekommen hat. Bei Oracle 12.2 war dies im Herbst 2016 der Fall, denn zu diesem Zeitpunkt wurde diese Version in unterschiedlichen Varianten exklusiv als Cloud-Service angeboten.

Eine künstliche Auslieferungs-Verschleppung ist nicht das, was sich Kunden wünschen, zumal sie die neue Version mit ihren jährlichen Support-Gebühren bereits vorfinanziert haben. Zu meinen, dass man mit dieser Vorgehensweise der Oracle-Cloud einen Gefallen tut, ist zumindest naiv. Bei anderen Gelegenheiten haben sich ähnliche Ideen als veritable Rohrkrepierer erwiesen.

Wir leben in Zeiten, in denen Daten so prominent sind wie in keiner anderen Periode der kurzen IT-Geschichte. Wir leben in Zeiten, in denen Datenbank-Systeme so wichtig sind wie nie und in denen es noch nie so viele konkurrierende Datenbank-Systeme und noch nie so viele Datenbank-Startups gab. Im Klartext heißt das: Oracle ist nicht alleine auf dem Markt, Kunden haben die Qual der Wahl und deshalb tut Oracle gut daran, sich jeden Tag aufs Neue intensiv um seine Kunden zu bemühen.

Mit einer Auslieferungs-Verzögerung ist keinem gedient. Eher wäre ein groß angelegtes Know-how-Transfer-Projekt angebracht, das den Oracle-Kunden hilft, die vielen Neuerungen im Detail zu verstehen, um daraus Projekte zu aktivieren. Diese Oracle-Investition (ja, Oracle sollte hier aktiv investieren) würde sich mehr als bezahlt machen und langfristig positiv wirken.

Mit den vielen erstklassigen Partnern und dem erstklassigen Team müsste das für Oracle eine leichte Übung sein, die eine wirkliche Win-Win-Win-Situation darstellt – für Kunden, für Partner und für Oracle!

Zentrale Benutzerverwaltung mit EUS, OUD und Active Directory

Stefan Oehrli, Trivadis AG

Sicherheit ist heutzutage eine der zentralen Herausforderungen für On-Premise- und Cloud-basierte Datenbanken. Mit der Umsetzung von Sicherheitskonzepten für Datenbanken werden diese Herausforderungen gemeistert. Doch viele dieser Bestrebungen sind nur sinnvoll, wenn bereits die Authentifizierung und Autorisierung mit entsprechender Sorgfalt umgesetzt wird.

Anstelle der dezentralen Verwaltung der Benutzer, Rechte und Rollen in jeder Datenbank ist es übersichtlicher und vor allem sicherer, diese zentral zu verwalten. Dieser Artikel zeigt, wie die Benutzerverwaltung mit Oracle Enterprise User Security und Oracle Unified Directory zentral verwaltet und mit MS Active Directory integriert werden kann.

In der Regel wird für die Authentifizierung des Oracle-Datenbank-Benutzers die lokale Authentifizierung in der Datenbank mit Benutzername und Passwort verwendet. Mit einem entsprechend sicheren Passwort ist diese Methode für viele Umgebungen weiterhin eine akzeptable Authentifizierungsmethode. Sie entspricht allerdings nicht mehr den aktuellen Sicherheitsempfehlungen. Je mehr Benutzer und Datenbanken vorhanden sind, desto unübersichtlicher und somit unsicherer wird dabei die Benutzer-Administration. Wo hat welcher Benutzer Zugriff? Wer hat wo welche Rechte? Werden bei Funktions- oder Rollenwechseln keine Rechte vergessen? Viele dieser Fragen lassen sich bei einer dezentralen Benutzerverwaltung nicht so einfach beantworten.

Die verzeichnisbasierte Authentifizierung beziehungsweise Oracle Enterprise User Security (EUS) bietet eine alternative Methode zur Authentifizierung von Datenbankbenutzern. Im Gegensatz zur lokalen Datenbank-Authentifizierung geht EUS über die reine Authentifizierung hinaus. Mit einem Oracle-Verzeichnis lassen sich Benutzer, Gruppen und verschiedene Rollen verwalten, also auch die Autorisierung. Der wesentliche Vorteil dieser Lösung ist die zentrale Verwaltung. Es gibt nur eine Stelle, an der Benutzer erstellt und Berechtigungen erteilt werden. So ist es beispielsweise nicht mehr nötig, dass der Datenbank-Administrator in jeder seiner Datenbanken Benutzer anlegt oder löscht.

EUS setzt auf die Oracle-Identity-Management-Infrastruktur, die wiederum einen LDAP-kompatiblen Verzeichnis-

dienst verwendet, um Benutzer zentral zu speichern und zu verwalten. Dabei werden folgende Verzeichnisdienste unterstützt:

- OID Oracle Internet Directory ist ein LDAP-v3-konformes Verzeichnis. Es basiert auf einer Oracle-Datenbank und ist vollständig in Oracle Fusion Middleware und Oracle Applications integriert.
- OUD Oracle Unified Directory ist ein Java-basiertes Verzeichnis. Es ist eine All-in-one-Lösung mit Storage, Proxy, Synchronisations- und Virtualisierungsfähigkeiten.

Die eigentliche Authentifizierung erfolgt bei EUS je nach Konfiguration entweder durch Passwort, Kerberos oder SSL-Authentifizierung. Die Benutzer profitieren dabei von einer Single-Sign-on (SSO) oder Single-Passwort-Authentifizierung.

Trotz der Vorteile einer zentralen Benutzerverwaltung wird bei vielen Unternehmen die Authentifizierung und Autorisierung weiterhin dezentral in den Oracle-Datenbanken gelöst. Die Gründe dafür sind vielfältig. Ohne im Detail weiter auf diese einzugehen, lässt sich aber mit Sicherheit sagen, dass viele Unternehmen nicht die nötigen Ressourcen aufbringen können, um einen weiteren mehr oder weniger komplexen Verzeichnisdienst zu betreiben.

Obwohl Benutzer- und Gruppen-Informationen bereits zentral in einem MS-Active-Directory oder LDAP-Verzeichnis verwaltet werden, lassen sich diese nicht direkt für EUS verwenden. Es braucht immer zwingend ein Oracle-Verzeichnis wie OID oder OUD. Das bedeutet, dass neben einem weiteren Verzeichnis auch eine Synchronisation aufgebaut und betrieben werden muss. Dies führt zu einer redundanten Datenhaltung sowie komplexeren Architektur und birgt das Risiko erhöhter Fehleranfälligkeit, was schlussendlich nicht im Interesse des Kunden ist.

Oracle Unified Directory AD Proxy Server

Im Gegensatz zu OID kann OUD aber nicht nur als reiner Verzeichnis-Server, sondern auch als Proxy-Server betrieben werden. Dabei nutzt OUD die Benutzer- und Gruppen-Informationen im bestehenden Verzeichnis und ruft diese direkt mit LDAP-Abfragen ab. EUS-spezifische Informationen wie Datenbank-Registrierungs-Informationen, Benutzer- und Rollen-Zuordnungen sowie weiteren EUS-spezifischen Metadaten werden lokal im OUD gespeichert. Auf diese Weise arbeitet OUD als Echtzeit-Interpreter für die Oracle-Datenbanken.

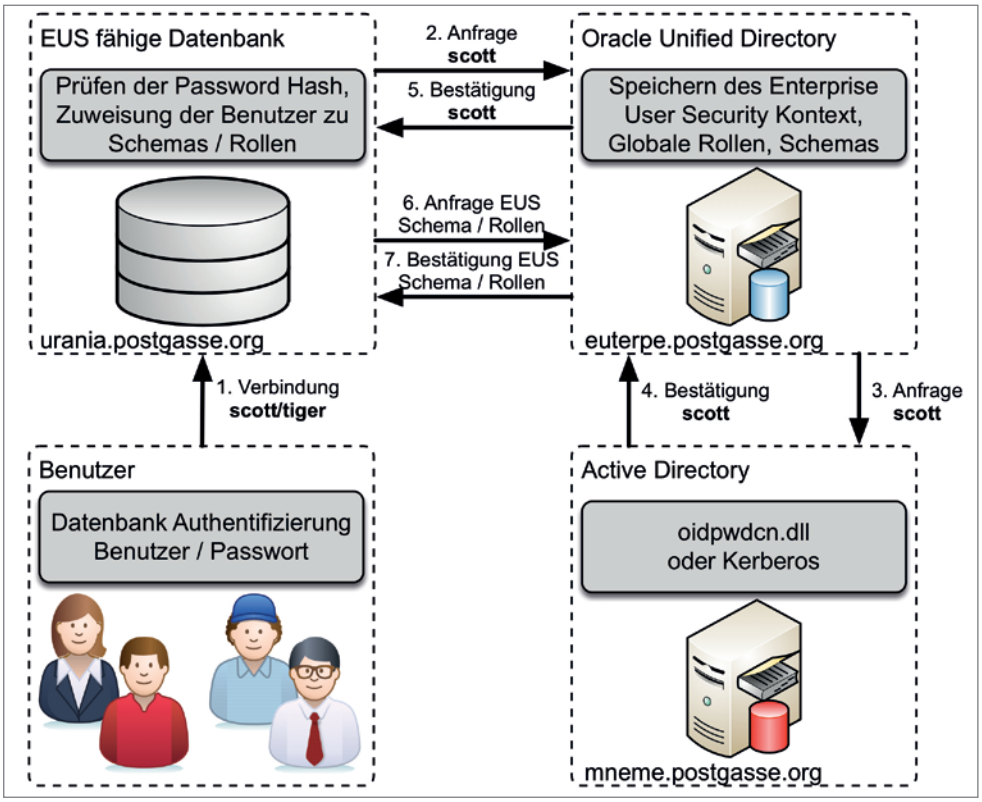


Abbildung 1: Enterprise-User-Security mit Active-Directory-Integration

Anfragen nach Benutzer-Informationen werden transparent beantwortet, unabhängig davon, ob diese lokal im OUD abgelegt sind oder aus einem zentralen Verzeichnis abgerufen werden. Somit müssen die Daten nicht synchronisiert oder redundant abgelegt werden, was wiederum die Gesamtbetriebskosten reduziert.

Abbildung 1 zeigt den schematischen Aufbau einer EUS-Datenbank mit einem OUD-AD-Proxy und MS-Active-Directory als zentralen Verzeichnis-Server. Der Verbindungsaufbau vom Benutzer „Scott" und damit die Authentifizierung und Autorisierung ist mit den Pfeilen dargestellt.

Authentifizierung

Je nach Konfiguration stehen Passwort, Kerberos oder SSL-Authentifizierung zur Verfügung. Am einfachsten ist die Passwort-Authentifizierung, bei der sich der Datenbank-Benutzer mit seinem Windows-Account-Passwort an der Datenbank anmeldet, wobei auch diese nicht ganz ohne Hürden umgesetzt werden kann. Die vom Microsoft verwendeten Passwort-Hashes sind nicht direkt mit Oracle-Datenbanken kompatibel. Daher muss auf jedem Active-Directory-Domain-Controller das Oracle-Password-Change-Notification-Plug-in („oidpwdcn.dll") installiert sein. Dazu gehört auch eine Active-Directory-Schema-Erweiterung, damit das Plug-in den Oracle-spezifischen Hashwert abspeichern kann. Wird hingegen Kerberos als Authentifizierungsmethode eingesetzt, sind das Plug-in und die Schema-Erweiterung nicht erforderlich. Dafür kann mit Kerberos zusätzlich ein Single-Sign-on (SSO) erreicht werden.

Voraussetzungen

Für den Aufbau der in *Abbildung 1* vorgestellten Architektur sind folgende Voraussetzungen zu erfüllen:

- Oracle Enterprise Edition 12c Datenbank (12.1.0.2) mit LDAP-Client-Patch 19285025 und einer EUS-Test-Datenbank

- Oracle Unified Directory 11.1.2.3.0 mit aktuellem Bundle-Patch 25383162
- MS Active Directory Domain Controller auf Windows 2012 R2
- Oracle Enterprise Manager Cloud Control 13*c* R2 für die Konfiguration von Enterprise User Security. Alternativ kann hier auch eine ältere OEM-Version oder das EUSM-Command-Line-Utility verwendet werden (siehe MOS Note 1085065.1, EUSM, Command Line Tool for EUS Administration and Some EUS Good to Know)

Konfiguration Active Directory

Damit die EUS-Passwort-Authentifizierung auch mit Active-Directory funktioniert, sind ein OUD-Passwort-Synchronisations-Plug-in sowie eine Schema-Erweiterung notwendig. OUD liefert die entsprechende Java-Klasse, um das Attribut „orclCommonAttribute" zu erstellen. Die Schema-Erweiterung wird mit dem Aufruf der Java-Klasse direkt aus dem OUD-Home-Verzeichnis „$OUD_HOME/OUD1/config/EUS/ActiveDirectory" erstellt *(siehe Listing 1)*.

Das OUD-Passwort-Synchronisations-Plug-in „oidpwdcn.dll" liegt im selben Verzeichnis und steht als 32- oder 64-Bit-Version zur Verfügung. Die DLL-Datei muss für die Installation auf dem Active-Directory-Server in das Verzeichnis „C:\Windows\system32" kopiert werden. Damit sie zum Einsatz kommt, muss in der Registry zusätzlich der Registry Key „HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Notification" um „oidpwdcn" erweitert werden. Mit dem Restart des Active-Directory-Servers ist die Installation abgeschlossen. Sind mehrere Active-Directory-Server vorhanden, muss das Plug-in analog auf jedem Server installiert sein. Wird dagegen ausschließlich Kerberos zur Authentifizierung verwendet, sind die oben aufgeführten Anpassungen nicht erforderlich.

Aufbau der OUD-Proxy-Instanz

Für das Erstellen der verschiedenen OUD-Instanzen liefert Oracle entsprechende Setup-Programme, die alle im OUD-Installations- beziehungsweise OUD-Home-Verzeichnis bereitstehen. Je nach Bedarf können die Programme mit einer grafischen Oberfläche, als Command-Line-Installation oder auch als Silent-Installation ausgeführt werden. Die Silent-Installation ist vorwiegend hilfreich, wenn mehrere Instanzen mit einem Skript automatisiert aufgebaut werden sollen.

Die OUD-Proxy-Instanz selbst wird mit „oud-proxy-setup" erstellt. Ohne Angabe eines Parameters wird direkt die grafische Oberfläche für die Eingabe der Server- und Instanz-Konfiguration gestartet. Für EUS und die MS-AD-Integration muss dabei zwingend LDAPS aktiviert sein. Entweder wird ein kundenspezifisches SSL-Zertifikat verwendet oder wie beim Aufbau der Testumgebung ein selbstsigniertes Zertifikat *(siehe Abbildung 2)*. Für Produktionsumgebungen wird empfohlen, jeweils nur korrekt signierte Zertifikate zu verwenden.

```
java ExtendAD -h mneme.postgasse.org -p 389 -D cn=administrator,cn=user
s,dc=postgasse,dc=org -w <pwd> -AD
dc=postgasse,dc=org -commonattr
```

Listing 1

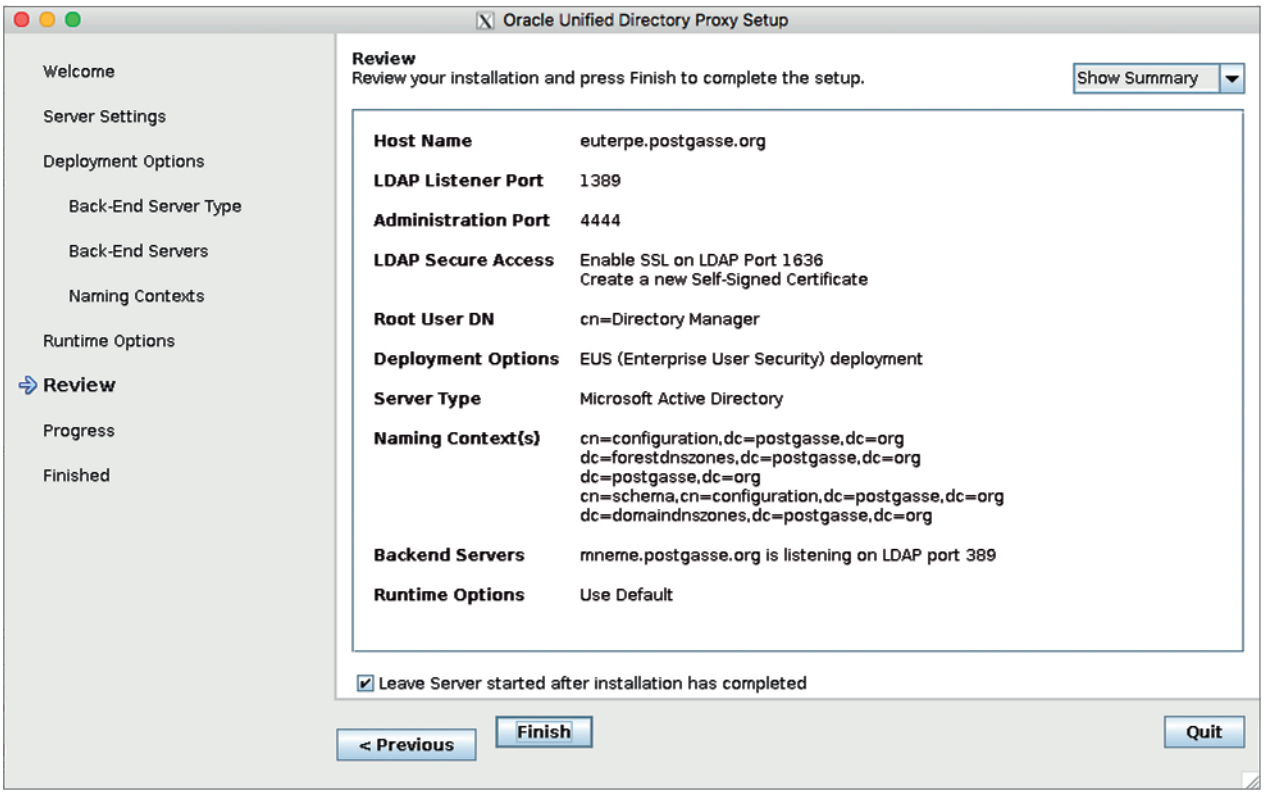


Abbildung 2: Zusammenfassung des OUD-Proxy-Setup

Umgebungsvariable	Pfad
ORACLE_BASE	/u00/app/oracle
OUD_HOME	/u00/app/oracle/product/oud11.1.2.3
ORACLE_HOME	/u00/app/oracle/product/oud11.1.2.3/OUD1
Verzeichnis für die OUD Instanzen	/u00/app/oracle/instances
INSTANCE_NAME	../../instances/oud_ad_proxy

Tabelle 1

In den nächsten Schritten werden als Proxy-Konfiguration „Configure EUS" festgelegt und MS-Active-Directory als Datenquelle mit dem entsprechenden EUS-Namenskontext ausgewählt. *Abbildung 2* zeigt zusammenfassend alle Einstellungen. Durch die Bestätigung mit dem Button „Finish" wird anschließend eine OUD-Proxy-Instanz erzeugt. Als Standard entsteht dabei immer eine Instanz mit dem Namen „ASINST_1" im OUD-Home-Verzeichnis.

Grundsätzlich wird empfohlen, die OUD-Instanz explizit zu benennen und in einem anderen Verzeichnis beziehungsweise auf einem anderen Filesystem als die OUD-Software anzulegen. Ähnlich wie bei der Oracle Flexible Architecture (OFA) ist so sichergestellt, dass Daten und Konfiguration getrennt von der Software verwaltet werden. Um eine OUD-Instanz explizit zu benennen, muss der Instanz-Name vor dem Aufruf von „oud-proxy-setup" via Umgebungsvariable „INSTANCE_NAME" festgelegt sein. Der Instanz-Name ist dabei als Pfad relativ zum OUD-Home-Verzeichnis anzugeben. Für die EUS-Test-Umgebung wurde als Instanz-Name „oud_ad_proxy" gewählt. *Tabelle 1* zeigt die verschiedenen Verzeichnisse und Umgebungsvariablen.

Bevor die neu erstellte OUD-Proxy-Instanz nun für EUS verwendet werden kann, braucht es noch kleinere Anpassungen. Als Erstes werden das Proxy-Workflow-Element angepasst und die Remote-Bind-Anmelde-Informationen festgelegt. Im Folgenden wird dabei der AD-Benutzer „OUD Admin" verwendet. Grundsätzlich kann dies aber ein beliebiger AD-Benutzer sein, der vollen Lesezugriff auf Benutzer- und Gruppen-Informationen hat. Zusätzlich wird auch der Client-Connection-Mode auf „use-specific-identity" gesetzt. Diese Einstellung ist nötig, wenn das externe Verzeichnis keine anonymen Bind-Anfragen zulässt. Die Anpassungen werden mit dem OUD-Command-Line-Admin-Tool „dsconfig" ausgeführt *(siehe Listing 2)*.

Damit EUS die Benutzer- und Gruppen-Informationen findet, müssen die entsprechenden Einträge im EUS-Kontext ebenfalls angepasst werden. Oracle liefert hierzu im Verzeichnis „$OUD_HOME/OUD1/config/EUS" eine LDIF-Template-Datei „modifyRealm.ldif", die wie folgt angepasst werden muss:

- Ersetzen von „dc=example,dc=com" mit dem korrekten Namenskontext
- Ersetzen von „ou=people" und „ou=groups" durch den entsprechenden Ort für die Benutzer- beziehungsweise Gruppen-Informationen

Das angepasste LDIF-Template wird anschließend mit „ldapmodify" geladen: „ldapmodify -h euterpe.postgasse.org -p 1389 -D „cn=Directory Manager" -w <PASSWORT> -f modifyRealm.ldif". Grundsätzlich ist die OUD-Proxy-Instanz nun bereit für EUS. Oracle-11*g*-Datenbanken können direkt in OUD registriert und für EUS konfiguriert werden.

Für Oracle 12*c* beziehungsweise EUSM 12*c* und Oracle Cloud Control 12*c*/13*c* ist hingegen noch eine weitere Anpassung notwendig. Die aktuellen Versionen der Oracle-Datenbank und Oracle-Cloud-Control verwenden den SASL-Mechanismus, um sich mit der OUD-Instanz zu verbinden. Allerdings funktioniert dies in OUD standardmäßig nicht. Das Admin-Passwort des EUS-Users muss zusätzlich zwingend mit einem reversiblen Algorithmus abgespeichert sein. Dazu ist das Root-Passwort-Profile zu erweitern, sodass neue Passwörter auch mit dem AES-Algorithmus abgespeichert werden *(siehe Listing 3)*. Mit einem expliziten Neusetzen des Passworts ist sichergestellt, dass auch das neue Passwort-Storage-Schema verwendet wird *(siehe Listing 4)*.

Für Produktionsumgebungen ist es sinnvoll, dass anstelle des Root-Passwort-Profils ein dezidiertes Passwort-Profil für EUS sowie entsprechende EUS-Admin-Benutzer erstellt werden.

Datenbank-Konfiguration

Damit die Datenbank im OUD registriert werden kann, wird im Netzwerk-

```
dsconfig set-workflow-element-prop \
  --element-name proxy-we1 \
  --add exclude-list:cn=directory\ manager \
  --add exclude-list:cn=oraclecontext,dc=postgasse,dc=org \
  --set remote-root-dn:"cn=OUD Admin,cn=users,dc=postgasse,dc=org" \
  --set remote-root-password:<PASSWORT> \
  --set remote-ldap-server-bind-dn:"cn=OUD Admin,cn=users,dc=postgasse,dc=org" \
  --set remote-ldap-server-bind-password:<PASSWORT> \
  --set ldap-server-extension:proxy1 \
  --set client-cred-mode:use-specific-identity \
  -h euterpe.postgasse.org -p 4444 -X \
  -D cn=directory\ manager -j $etc/oud_ad_proxy.pwd -n
```

Listing 2

Verzeichnis der Datenbank die „ldap.ora“-Datei angepasst. Dazu werden der OUD-Server und der Default-Kontext angegeben. Im Folgenden ist die „ldap.ora “-Datei aufgeführt, wie sie bei der in *Abbildung 1* beschriebenen Testumgebung verwendet wird. Als Directory-Server-Typ muss auch bei OUD „OID“ angegeben sein *(siehe Listing 5)*.

In „sqlnet.ora“ wird die Namesauflösung angepasst und neu „LDAP“ als zusätzliche Quelle angegeben. Dazu ein Auszug aus der Datei: „NAMES.DIRECTORY_PATH=(LDAP, TNSNAMES, EZCONNECT)“.

Die eigentliche Registrierung der Datenbank in OUD erfolgt mit dem „dbca“ interaktiv wie in *Abbildung 3* oder „silent“ via Command-Line. Für die Registrierung ist die Anmelde-Information des EUS-Admins erforderlich. Als Standard-Database-CN wird der DB-Unique-Name verwendet, optional lässt sich dieser aber auch anpassen. Ist für diese Datenbank noch kein Oracle-Wallet vorhanden, legt der „dbca“ ein neues Wallet im Admin-Verzeichnis der Datenbank an und speichert darin die „OUD DB“-Credentials.

Neben der OUD-Registrierung und dem Wallet passt der „dbca“ zusätzlich auch die Datenbank-Parameter „ldap_directory_access“ sowie „ldap_directory_sysauth“ entsprechend an. Die Basis für EUS ist nun sichergestellt. Was nun noch fehlt, sind entsprechende EUS-Rollen beziehungsweise Datenbank-Benutzer und -Rollen sowie die Zuweisung zu den verschiedenen EUS-Datenbanken. Als einfaches Beispiel wird mit SQL*Plus ein globales Shared-Schema erstellt, das die lokalen Rolle „connect“ erhält *(siehe Listing 6)*. Anschließend wird mit EUSM für die Datenbank „TDB121A“ ein einfaches Mapping für alle Benutzer einer bestimmten Organisations-Unit („ou“) erstellt. Somit kann sich nun jeder Benutzer in der Organisations-Unit „ou=People,dc=postgasse,dc=org“ in MS-AD mit der Datenbank verbinden. Die Abfrage des „SYS_CONTEXT“ gibt entsprechend Auskunft über die Enterprise Identity *(siehe Listing 8)*.

Das Beispiel zeigt, wie einfach man ein „1:1“-Mapping erstellen und sich anschließend als Enterprise-Benutzer an der Datenbank anmelden kann. In der Praxis wird das Mapping komplexer ausfallen. Gewisse EUS-Benutzer werden globale private Schemata haben, andere dagegen mit globalen Shared-Schemata und Enterprise-Rollen arbeiten. In der Kombination mit Proxy-Rechten und Administrativen-Privilegien wie „SYSDBA“ lassen sich so umfangreiche und komplexe Benutzer- und Rollen-Konzepte umsetzen.

```
dsconfig set-password-policy-prop \
  --policy-name "Root Password Policy" \
  --add default-password-storage-scheme:"AES" \
  -h euterpe.postgasse.org -p 4444 -X \
  -D cn=directory\ manager -j $etc/oud_ad_proxy.pwd -n
```

Listing 3

```
ldappasswordmodify -h euterpe.postgasse.org -p 4444 \
  -D "cn=Directory Manager" -j $etc/oud_ad_proxy.pwd –Z \
  -c <PASSWORT> -n <PASSWORT>

ldapsearch -h euterpe.postgasse.org -p 4444 \
  -D "cn=Directory Manager" \
  -j $etc/oud_ad_proxy.pwd -Z -X \
  -b "cn=Directory Manager,cn=Root DNs,cn=config" \
  -s base objectclass=* userpassword

dn: cn=Directory Manager,cn=Root DNs,cn=config
userpassword: {AES}AfLk8S3k8k3ekXLV6qaP+mFUe/DStQ4ngUAdo6P5EjOKOM-
m4AN27Xw==
userpassword: {SSHA512}YSI4tlv3gO4z0czFU9EdTOGnfmGhQqOtbWUopV7xgcPDlYS/
Eea3ydQUt
 YT5Qog/ngZ8w4M3EHNf4/MObnPC8M+lblEIKivf
```

Listing 4

```
DIRECTORY_SERVERS= (euterpe.postgasse.org:1389:1636)
DEFAULT_ADMIN_CONTEXT = "dc=postgasse,dc=org"
DIRECTORY_SERVER_TYPE = OID
```

Listing 5

Patch, Bugs und weitere Sorgen

Im Zusammenhang mit LDAP, SSL und der Oracle-Datenbank gibt es noch einen Bug, der zwingend behoben werden muss, damit die Anmeldung so wie oben beschrieben funktioniert. Der Bug 19285025 ist bis dato in keinem Base-Release oder Patch Set Update behoben. Daher muss der entsprechende Patch 19285025 weiterhin explizit bei allen aktuellen Versionen wie Oracle 11.2.0.4, 12.1.0.2 und leider auch 12.2.0.1 installiert werden.

Neben diesem Bug gibt es im SSL- und Kerberos-Umfeld generell den einen oder anderen Bug. Diese können mit entsprechenden Workarounds oder Patches umgangen werden. Nichtsdestotrotz ist das Troubleshooting nicht immer ganz einfach. Schließlich ist die Architektur bei der zentralen Benutzerverwaltung komplexer. Neben der Datenbank sind eben auch OUD und MS-AD bei einer Anmeldung an der Datenbank beteiligt. Die effektiven Ursachen von ORA-01017 oder ORA-28030 sind daher häufig nicht direkt identifizierbar. Mithilfe des OUD Access Log, SQL Net Tracing oder eines entsprechenden Oracle-Events kommt man der effektiven Ursache jedoch in der Regel schnell auf die Spur.

Weitere Überlegungen

Die in *Abbildung 1* aufgezeigte Architektur ist für eine Test- und Engineering-Umgebung ausreichend. Für den produktiven

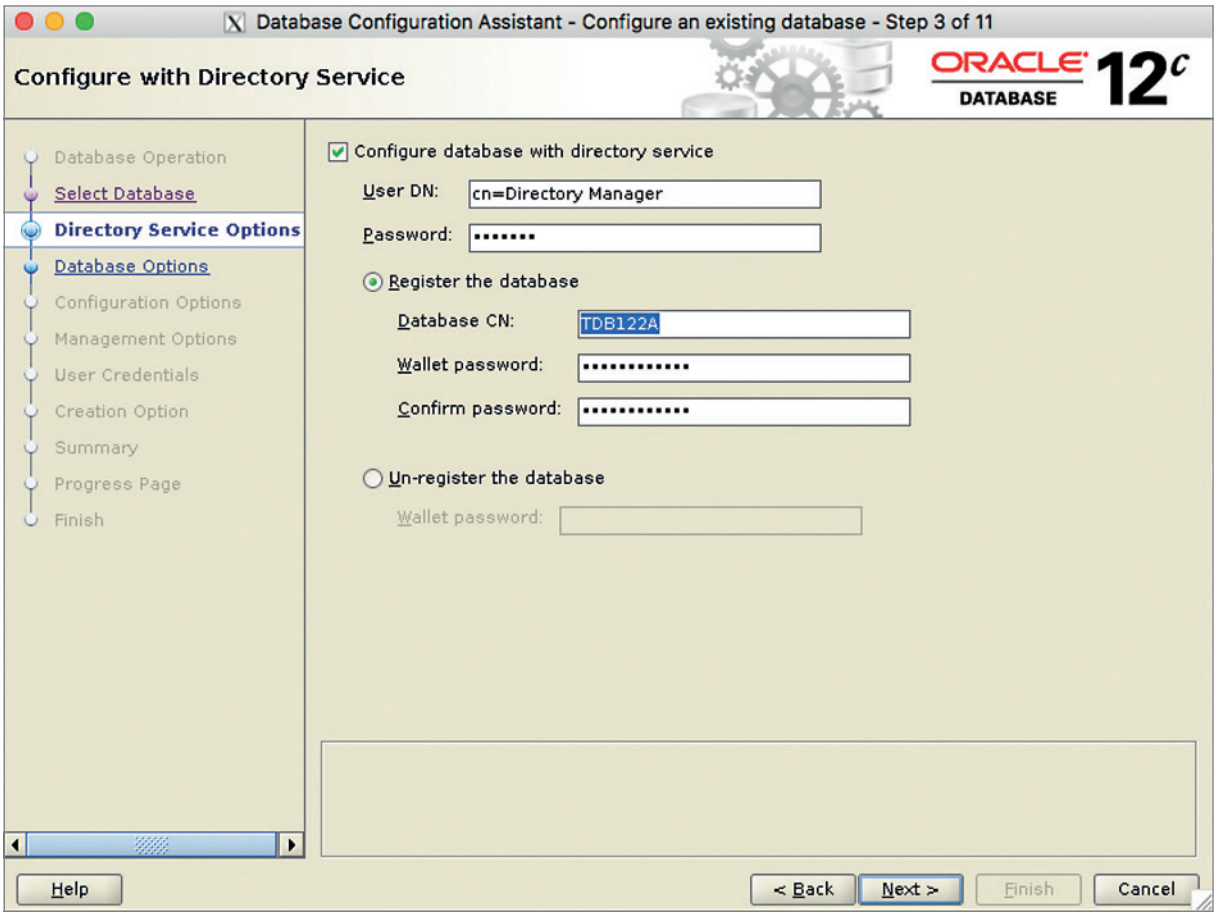


Abbildung 3: Datenbank-Registrierung mit „dbca“

Betrieb sind sicher weitere Maßnahmen nötig, damit die Hochverfügbarkeit und das Desaster Recovery sichergestellt werden können. Schlussendlich ist eine zentrale Benutzerverwaltung ein Single Point of Failure. Ist sie nicht verfügbar, können sich die Benutzer nicht mehr an den Datenbanken anmelden. OUD bietet verschiedene Möglichkeiten wie Online-Backup und Replikation, um die OUD-Instanzen sicher und hochverfügbar zu betreiben. Diese Maßnahmen sind in einem entsprechenden OUD-Konzept zu planen und umzusetzen.

Fazit

Mit Oracle-Unified-Directory und dem AD-Proxy wird die Umsetzung einer zentralen Benutzerverwaltung mit Enterprise User Security wesentlich vereinfacht. Ein OUD-AD-Proxy ist grundsätzlich schnell aufgebaut, sodass man direkt den ersten Nutzen hat. Nichtsdestotrotz liegen

```
CREATE USER global_shared IDENTIFIED GLOBALLY;
GRANT connect TO global_shared;
```

Listing 6

```
eusm createMapping domain_name="OracleDefaultDomain" realm_
dn="dc=postgasse,dc=org" map_type=SUBTREE map_dn="ou=People,dc=pos
tgasse,dc=org" schema=GLOBAL_SHARED ldap_host="euterpe.postgasse.
org" ldap_port=1389 ldap_user_dn="cn=Directory Manager" ldap_user_
password=<PASSWORT>
```

Listing 7

```
SQL> connect oehrli
Enter password:
Connected.
SQL> show user
USER is "GLOBAL_SHARED"
SQL> SELECT
  sys_context('userenv','SESSION_USER') "USER",
  sys_context('userenv','AUTHENTICATED_IDENTITY') "AUTH_IDENTITY",
  sys_context('userenv','ENTERPRISE_IDENTITY') "ENTERPRISE_IDENTITY"
FROM DUAL;
USER             AUTH_IDENTITY  ENTERPRISE_IDENTITY
---------------  -------------- ----------------------------------------
GLOBAL_SHARED OEHRLI         cn=Stefan Oehrli,ou=People,dc=postgasse,dc=org
```

Listing 8

die Herausforderungen in den Details. So sind die MS-AD-Integration und Kerberos-Konfiguration bei komplexen AD-Architekturen um einiges aufwändiger. Zudem braucht es für den unternehmensweiten Einsatz zwingend ein geeignetes Benutzer- und Rollen-Konzept. Dafür muss häufig mehr Zeit eingeplant werden als für die einfache Konfiguration von OUD. Die verschiedenen Bugs im Kerberos-, LDAP- beziehungsweise SSL-Umfeld erschweren einem das Leben zusätzlich. Sind diese Hürden erst einmal gemeistert, hat man aber ein verlässliches und sicheres System für die zentrale Benutzerverwaltung von Oracle-Datenbanken.

Stefan Oehrli
stefan.oehrli@trivadis.com

Scripting mit SQLcl – Batch Scripts auf einem neuen Level

Robert Marz, its-people GmbH

SQLcl steht bereit, das gute alte SQL*Plus als Kommandozeile für die Datenbank abzulösen und dessen Funktionsumfang auf die Höhe der Zeit bringen. Die spannendste Neuerung ist das Scripting: Batch Scripts können in JavaScript verfasst werden. Aber auch andere Sprachen wie Python oder Lua sind möglich. Dadurch eröffnet sich ein wahres Füllhorn an Möglichkeiten, von denen „viele erstmal nicht offensichtlich sind.

SQLcl steht für „Oracle SQL Developer Command Line". Wie der Name vermuten lässt, wurde der komplette Funktionsumfang des SQL-Developer-Worksheet als Kommandozeile ausgekoppelt. SQLcl wird aktiv entwickelt: Zur Oracle Open World im September 2016 wurde ein Produktions-Release veröffentlicht, das seitdem alle sechs bis acht Wochen ein Update erhält.

SQLcl möchte das neue SQL*Plus werden und ist seit der Datenbank-Version 12c Release 2 im „$ORACLE_HOME/bin"-Verzeichnis zu Hause. Wer die neueste Datenbank-Version bereits im Einsatz hat, kann das Tool direkt benutzen. Alle anderen können es von der SQL-Developer-Seite des Oracle-Technologie-Networks herunterladen. Der Download ist mit einer Größe von rund 19 MB überschaubar. Die Installation ist mit dem Entpacken in ein beliebiges Verzeichnis abgeschlossen. Winzige Voraussetzung ist ein aktuelles Java im Suchpfad des Betriebssystems. Ein zusätzlicher Oracle-Client ist nicht nötig.

SQL*Plus „plus"

SQLcl deckt bis auf einige gewollte Ausnahmen den kompletten Funktionsumfang von SQL*Plus ab und enthält außerdem die Erweiterungen des SQL-Developer-Worksheet. Die Entwickler haben dem Tool darüber hinaus die Möglichkeit gegeben, Scripts in anderen Programmiersprachen auszuführen. Einzige Bedingung: Es muss eine Implementierung der Sprache existieren, die dem Standard „JSR-223" folgt und damit in der

Java Virtual Machine läuft. Java 1.8 bringt von Haus aus die Nashorn-Library mit. Das ist eine Implementierung von JavaScript für die JVM. Daher ist JavaScript die Standard-Sprache für solche Scripts.

SQLcl wird genau wie SQL*Plus aufgerufen. Der einzige Unterschied: Das Kommando heißt „sql". Man spart also beim Tippen die vier Tastendrücke für das „plus" *(siehe Abbildung 1)*. Leider wird diese Einsparung durch die verlängerte Startzeit des Java-Programms gegenüber dem in C implementierten SQL*Plus mehr als wieder aufgefressen. Nach dem erfolgten Start ist aber kein Geschwindigkeitsunterschied mehr festzustellen.

*Abbildung 1: SQLcl wird wie SQL*plus aufgerufen*

Die Dokumentation der neuen Scripting-Features ist äußerst spärlich. Es gibt noch kein eigenes Handbuch, sondern lediglich das Readme und die Beispiele im offiziellem Oracle-DB-Tools-GitHub-Repository *(siehe „https://github.com/oracle/oracle-db-tools/tree/master/sqlcl")*. SQLcl erweitert die verwendete Sprache um globale Objekte, die dann direkt im Script genutzt werden können:

- *ctx*
  Mit „ctx.write(„Hallo Welt")" wird Text in die Konsole geschrieben. „ctx.cloneCLIConnection()" eröffnet eine neue Datenbanksitzung mit den Zugangsdaten der aktuellen SQLcl-Verbindung.

- *sqlcl*
  „sqlcl.setStmt(„prompt Hallo Welt")" füllt den Kommandozeilen-Buffer von SQLcl, als würde in die Tastatur getippt, „sqlcl.run()" führt den Buffer anschließend aus. Alle Befehle, die auf diese Weise ausgeführt werden, können anschließend mit den Cursor-Tasten wieder aus der Historie von SQLcl hervorgeholt werden.

- *util*
  Schließlich gibt es noch „util.execute();" in verschiedenen Varianten, mit denen, ähnlich dem „execute immediate" aus PL/SQL, mit Bindevariablen versehene SQL-Statements abgesetzt werden können.

Aufwärmen

Neben dem „@"-Kommando zum Starten von althergebrachten SQL-Scripts kennt SQLcl das neue Schlüsselwort „script". Gefolgt von einem Dateinamen lädt es diese Datei und führt sie aus. Folgt nach „script" ein Zeilenumbruch, kann JavaScript direkt über die Tastatur in den Buffer eingegeben werden. Ein abschließendes „/" führt den Buffer dann aus *(siehe Abbildung 2)*.

Unser erstes Script in SQLcl könnte wie in *Listing 1* aussehen. Nach der Ausführung passiert … nichts. Der einzige Inhalt des Scripts ist eine Variablen-Deklaration. Variablen und Funktionen überleben in einer SQLcl-Session allerdings das Ende ihres Scripts. Im nächsten Script-Aufruf kann – ohne Deklaration – erneut darauf zugegriffen werden *(siehe Listing 2)*.

Ein nützliches neues Feature von SQLcl sind Aliase. Sie können in SQL, PL/SQL oder als Script angelegt werden. SQLcl speichert die Definition in einer eigenen Datei, sodass einmal angelegte Aliase auch beim nächsten Start von SQLcl wieder zur Verfügung stehen. Im Beispiel in *Listing 3* legen wir einen Alias „rememberMe" an, der den Inhalt der Variable „dontForget" ausgibt, wenn sie denn bereits deklariert wurde. Der Befehl „alias list" zeigt alle definierten Aliase an. Bei

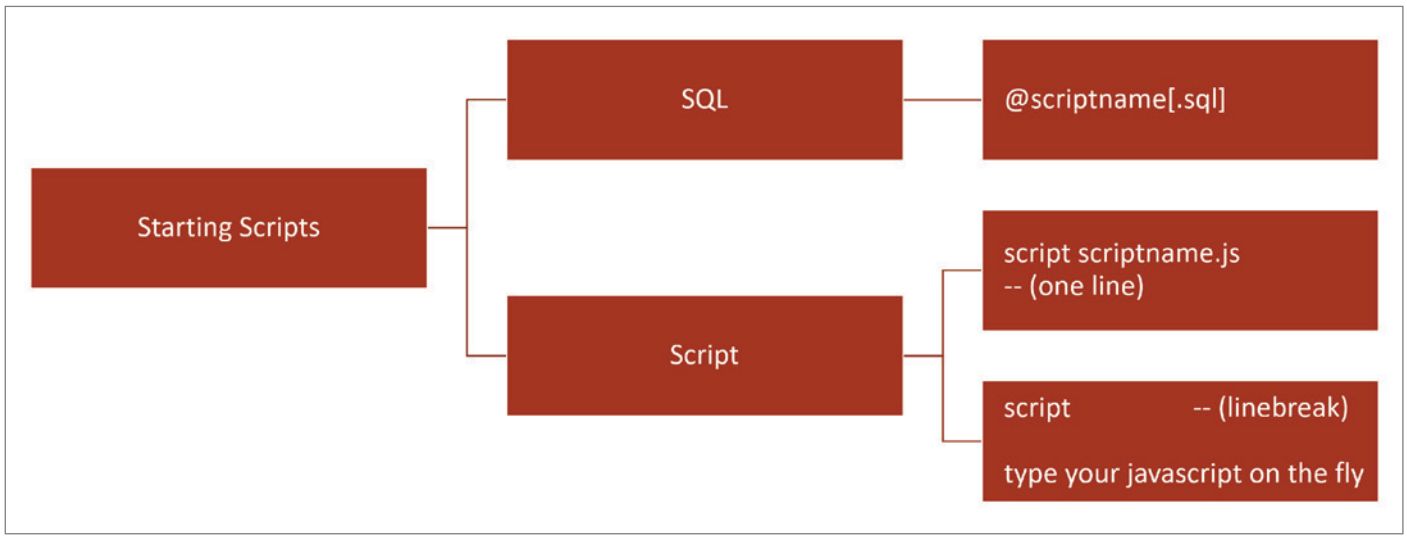


Abbildung 2: Starten von Scripts in SQLcl

der Installation bringt SQLcl bereits einige Beispiele mit.

Die Methode „ctx.write()“erzeugt die Ausgaben in der Konsole. Zeichenketten, die damit geschrieben werden sollen, müssen zwingend mit einem „\n” abgeschlossen sein, ansonsten erfolgt keine Ausgabe. Die Funktion „print()“ erfüllt den gleichen Zweck und ist sogar bequemer, weil hier das abschließende Zeichen nicht erforderlich ist. Beide benutzen unterschiedliche Output-Streams, wie man sehen kann, wenn die Aufrufe vermischt werden *(siehe Listing 4)*. Das führt dazu, dass die Reihenfolge der ausgegebenen Zeilen nicht wie erwartet ist.

Flusskontrolle

Jeder, der schon einmal ein SQL-Script für SQL*Plus geschrieben hat, weiß, dass diese nicht allzu dynamisch werden können: Über die Anweisung „whenever sqlerror” konnte die Ausführung abgebrochen oder der Fehler ignoriert werden. Schleifen oder Verzweigungen sind nicht möglich. Wir alle haben uns mit den üblichen Workarounds abgefunden: das „Zusammenspoolen” von Scripts oder das Ausweichen auf PL/SQL. In SQLcl steht für Batch Scripts endlich der ganze Umfang moderner Script-Sprachen zur Verfügung. Das Beispiel in *Listing 5* legt eine Tabelle nur dann an, wenn sie nicht bereits existiert. Selbstverständlich sind auch Schleifen möglich. Das nächste Beispiel berechnet eine Fibonacci-Folge und fügt die Werte mithilfe von Bind-Variablen in die Tabelle ein *(siehe Listing 6)*.

Blobs vom Dateisystem laden

Da die benutzten Script-Sprachen für die Java Virtual Machine geschrieben und dort ausgeführt werden, ist es einfach, aus der Script-Sprache heraus auf Java-Klassen und Objekte zuzugreifen. Das Beispiel in *Listing 7* macht sich dies für das Laden von Dateien aus dem Dateisystem zunutze: Der Dateiname wird beim Aufruf als Argument an das Script übergeben, die Datei gelesen und als Blob in der Datenbank gespeichert. Sollen anstelle von Blobs Clobs gespeichert werden, lauten die Zeilen 7 und 8 wie in *Listing 8*.

```
script
   var dontForget = 'Blumen kaufen';
/
```

Listing 1

```
script
   ctx.write(dontForget+'\n');
/
```

Listing 2

```
alias rememberMe=script
   if(typeof dontForget != 'undefined') print("Don‘t forget: "+ dontFor-
get);
```

Listing 3

```
script
   ctx.write('1 - ctx\n');
   print(    '2 - print');
   ctx.write('3 - ctx\n');
   print(    '4 - print');
   ctx.write('5 - ctx\n');
   print(    '6 - print');
/
2 - print
4 - print
6 - print
1 - ctx
3 - ctx
5 - ctx
```

Listing 4

```
script
// Check auf Vorhandene DB Objekte
/* Test ob die Zieltabelle vorhanden ist */
var tabName = „SQLCL_DEMO“;
var tabCnt = util.executeReturnOneCol(
      'select count(*) '+   // JS ist es egal, welche
      ' from tabs '     +   // Anführungszeichen verwandt
      " where table_name = ,"+tabName+"'" // werden
      );

if (tabCnt === 0){
   ctx.write("Tabelle "+tabName+" nicht vorhanden\n" +
       "Lege sie an.\n");

   sqlcl.setStmt( "set echo on\n"
       + "set feedback on \n"
       + "create table "+tabName+"   ( indx     number not null    \n"
       + "                           , fibonacci number             \n"
       + "                           );                             \n"
       // Alles was in SQLcl geht, geht auch hier:
       + "set serveroutput on size unl \n"
       + "alter table "+tabName+" add constraint pk_"+tabName+" primary
key (indx);");
   // Die einzelnen Befehle sind anschließend in der SQLcl Historie ab-
gelegt...
   // und können mit den Cursor-Tatsten wieder abgerufen werden
   sqlcl.run();
} else {
   ctx.write("Tabelle "+tabName+" schon vorhanden.\n");
}
/
```

Listing 5

Array Magic

Arrays sind mächtige Sprachkonstrukte. In JavaScript können Arrays sowohl einfache Werte als auch komplexe Objekte aufnehmen und beliebig tief geschachtelt sein. Im Beispiel in *Listing 9* definieren wir eine Benutzerliste als Array, bestehend aus Objekten mit je zwei Feldern „user" und „password". Wenn das Array einmal angelegt ist, bleibt es wie alle Variablen für den Rest der SQLcl-Session persistent. Das nächste Script durchläuft alle Elemente des Arrays in einer Schleife und gibt den Inhalt des Felds „user" aus *(siehe Listing 10)*.

In JavaScript muss keine Schleife bemüht werden, um ein Array zu durchsuchen: Arrays können mit Filterfunktionen erweitert werden. Die Funktion „getPasswd" aus dem nächsten Beispiel definiert einen solchen Filter, der die anonyme Funktion innerhalb der runden Klammern für alle Elemente des Arrays aufruft. Die Filterfunktion erzeugt ein neues Array mit allen Elementen, für die die anonyme Funktion „true" zurückliefert *(siehe Listing 11)*. In dem Beispiel wird davon ausgegangen, dass die User im Array eindeutig sind. Es ist also in Ordnung, nur das Kennwort des ersten Treffers zurückzuliefern.

Background Sessions

JavaScript kennt keine Threads, Java hingegen schon. Mit dem Thread-System von Java lassen sich aus einer SQLcl-Session beliebig viele weitere Verbindungen in die Datenbank eröffnen. Auf diese Weise können Aufgaben parallel abgearbeitet oder die Hauptsession überwacht werden.

Im offiziellem Oracle-DB-Tools-GitHub-Repository gibt es zwei Beispiele, die den Einsatz verdeutlichen, für diesen Artikel jedoch zu lang sind: „bg.js" verdeutlicht das Prinzip und „longops.js" ist eine Anwendung, die aus einer zweiten Session „v$session_longops" überwacht wird und bei langen Operationen eine Fortschrittsanzeige ausgibt.

Andere Script-Sprachen

Die Nashorn-Bibliothek des aktuellen Java 1.8 implementiert die bereits etwas in die Jahre gekommene JavaScript-Version „ECMA Script 5". Der aktuelle Stan-

```
script
// Schleife
var binds = {};
var a = 0, b = 1, f = 1, n=10, ret;
for(var i = 2; i <= n; i++) {
   f = a + b;
   a = b; b = f;
   binds.i = i;
   binds.f = f;
   util.execute( "insert into "+tabName
                 + "(indx,fibonacci) values(:i, :f)"
                 , binds);
}
sqlcl.setStmt( "commit; \n"
             + "select * from "+tabName+"; \n"
             );
sqlcl.run();
/
```

Listing 6

```
var HashMap = Java.type("java.util.HashMap");
var bindmap = new HashMap();

print("\nreading file: "+ args[1]);
var filePath=args[1];

var blob=conn.createBlob();
var bstream=blob.setBinaryStream(1);

java.nio.file.Files.copy( java.nio.file.FileSystems.getDefault().getPath(filePath)
                        , bstream );
bstream.flush();

bindmap.put("inhalt", blob);
bindmap.put("pfad", filePath);


if(!util.execute( "insert into dokument (datei_inhalt,datei_pfad, datum)
values(:inhalt, :pfad, sysdate)"
                 , bindmap)
   ){ print("insert failed, exiting");
      exit;
}
sqlcl.setStmt( "commit; \n"
             + "set sqlformat ansiconsole \n"
             + ' select datei_pfad "Dateipfad",dbms_lob.getlength(datei_in-
halt) "Größe", to_char(datum,\'DD.MM.YYYY HH24:MI:SS\') "Zeit" ,
             + "from dokument;");
sqlcl.run();
```

Listing 7

```
var blob=conn.createClob();
var bstream=blob.setAsciiStream(1);
```

Listing 8

```
script
users=[
   {"user" : "DATA", "password" : "E_DXxXxXe2r" }
,  {"user" : "DOOTZ","password" : "E_OXxXxX_uas" }
,  {"user" : "BARRY","password" : "Barry's Secret" }
];
/
```

Listing 9

```
script
for (var i=0; i < users.length; i++ ){
   ctx.write ("Testing Schema "+ users[i].user+"\n");
/
```

Listing 10

```
script
// Searching Password for user
function getPasswd(username){
   return users.filter(function ( obj ) {
                                        return obj.user === username;
                                        }
                      )[0].password;
}

// Using in Scripts:
var uname = "BARRY";
ctx.write( "The password of Schema " + uname + " is "
   + getPasswd(uname) + "\n");

/
```

Listing 11

```
script
/* ES6 Demo */
const user = util.executeReturnOneCol('select user from dual');

const a="Hello ECMA Script 6";
let b="const and let are new to ES6";

print(`ES 6 allows multiline strings
   enclosed in backticks and variables includes with "$"
   and curly braces ${a} ${b} User: ${user}`);
/
```

Listing 12

dard „ES6“ bringt eine Fülle von Verbesserungen, unter anderem die beiden neuen Schlüsselwörter „const“ und „let“ für die Variablendeklaration sowie Mehrzeilenzeichenketten *(siehe Listing 12)*.

Die Java-Version 9, die zurzeit als Preview verfügbar ist, bietet eine ES6-fähige Nashorn-Implementierung. Wenn man SQLcl mit Java 1.9 startet, wird das obige Beispiel ohne Fehler ausgeführt. SQLcl erkennt anhand der Dateiendung, dass ein Script in einer anderen Sprache als JavaScript geschrieben wurde. Wenn ein passendes JAR-File im Classpath zu finden ist, wird dieses anstelle von Nashorn als Interpreter verwendet. Erfolgreich ausprobiert hat der Autor das mit „lua“ (JVM Implementierung „luaj“) und „Python“ (Jython-standalone).

Fazit

Die Scripting-Funktion im neuen SQLcl hat großes Potenzial, die Arbeit von DBAs und Entwicklern zu erleichtern. Die Einstiegshürde ist gering. Es gibt noch viel zu entdecken und auszuprobieren. Die kompletten Beispiele dieses Artikels sind unter *„https://github.com/its-people/sqlclscripts“* zu finden.

Robert Marz
robert.marz@its-people.de

12.2-Beta vor Ort

Oliver Pyka und Thomas Tretter

Die DOAG durfte im Jahr 2016 zwei Personen zum Beta-Test „Install and Upgrade 12.2“ entsenden. Am Rande der Noon-2-Noon-Veranstaltung im Januar in Würzburg hat uns dann Christian Trieb, Leiter der Datenbank Community, darauf angesprochen, ob wir uns dies vorstellen könnten. Wir, beide Freelancer, mussten natürlich abwägen, ob wir die fünf Arbeitstage investieren wollten. Den Artikel schreiben wir nun gut ein Jahr später und die Antwort ist – ja, der Einsatz hat sich gelohnt.

Oracle hatte zu diesem Termin entsprechende Vorgaben gemacht: maximal zwölf Teilnehmer aus den Usergruppen IOUG, UKOUG, DOAG, AUSOUG und ODTUG. Dass wir uns entschieden hatten, war eines, jetzt mussten wir Oracle vorgeschlagen werden und hätten auch ohne Angaben von Gründen abgelehnt werden können – was aber nicht der Fall war. Dabei ist zu erwähnen, dass keinerlei fachliche Qualitäten in irgendeiner Form nachzuweisen waren. Natürlich musste ein „Non Disclosure Agreement“ unterschrieben werden. Bereits im Vorfeld sollten wir eine Auswahl aus den zur Verfügung stehenden Testfällen treffen:

- Upgrade 11.2.0.4 nach 12.2
- Upgrade 11.2.0.x nach 12.1 nach 12.2
- Single Instance, RAC (Grid Infrastructure)
- Manual Upgrade, Tool-based
- Testdaten (anonymisiert) mitbringen

Gut drei Monate später konnte die Reise beginnen. Nach der Ankunft im Lab bekam jeder Teilnehmer einen Windows-Rechner zur Verfügung gestellt, von dem er Zugriff auf unzählige, speziell für ihn vorbereitete virtuelle Maschinen hatte. Natürlich erfolgte dazu auch eine Einführung. Die Tests an den folgenden drei Tagen wurden immer wieder für New-Feature-Sessions unterbrochen. Hochrangige Personen – manche extra für diesen Zweck eingeflogen – referierten dabei über die Besonderheiten im neuen Release.

Abschließend stellten alle Gruppen ihre Ergebnisse in einem vorgegebenen Folien-Satz vor. Zu dieser Veranstaltung waren angeblich dreißig Personen in der Telefonleitung und die gleiche Anzahl im Raum. Schon im Vorfeld wurden wir von der DOAG gelobt, da wir eben auf manche Dinge eine andere Sichtweise hatten. Was für andere ein Bug war, war für uns normal (so fehlte etwa einem Shell-Skript das Executable Flag). Wir hatten Folgendes getestet:

- GI fresh install, Upgrade von 11.2 nach 12.2
- ASM Upgrade
- Keine speziellen RAC-Tests
- Single Instance Upgrade, Pre-Upgrade Check, manuell Prozess abgebrochen und wieder aufgesetzt
- Single Instance Upgrade, „dbua“
- Konvertierung der Architektur: Container DB

Unsere Erkenntnisse und Ergebnisse:

- DBCA & DBUA/ASMCA: Neues Look & Feel – intuitiv
- Pre-Upgrade-Tool sehr hilfreich, sollte man bereits vorab nutzen
- Multitenant-Feature wie Relocate to the Cloud
- Golden Image der GI
- PDB Hot Cloning

Fazit

Es gab keine kritischen Ereignisse, nur kleine Fehler und Dokumentations-Bugs. Wir sprechen klar eine Empfehlung für 12.2 aus und können aus eigener Erfahrung sagen, dass ein Upgrade von 11*g* direkt auf 12.2 anzuraten ist.

Oliver Pyka
oliver.pyka@doag.org

Thomas Tretter
thomas.tretter@doag.org

Best Practices für das **Datenbank-Audit in Oracle 12.2**

Elke Fritsch, Muniqsoft

Dieser Artikel liefert anhand eines fiktiven Szenarios einige Tipps zur Konfiguration des Unified Auditing in der Version 12*c* Release 2, die ein zielgerichtetes und leichter auszuwertendes Datenbank-Audit ermöglichen und das Housekeeping der Audit-Daten vereinfachen. Darüber hinaus wird angesprochen, worauf man nach einer Migration von 11*g* auf 12*c* achten muss und was sich in Version 12.2 geändert hat.

Der Inhaber eines florierenden Software-Unternehmens möchte das Audit für seine Kundenverwaltung auf Vordermann bringen. Er hat seine Oracle-Datenbanken gerade von der Version 11.2.0.4 auf 12.2 (Standard Edition 2) migriert. Unter 11*g* wurden die Audit-Daten einfach nur gesammelt und jeden Monat über einen Job gelöscht. Nur die Default-Einstellungen waren aktiv. Das soll sich jetzt ändern. Folgendes würde er gerne auditieren:

- Alle Aktionen der DBAs (Stopps und Starts der Instanz, Änderungen an Parametern und Datenbank-Dateien etc.)
- Fehlgeschlagene Login-Versuche
- Rechte-Vergabe beziehungsweise Entzug
- DML-Aktionen auf der Tabelle „Kundendaten" inklusive SQL-Befehl
- Benutzungen der Packages „UTL_SMTP" und „UTL_FILE"
- Änderungen an Tabellen und PL/SQL-Code in den Schemata „KUNDEN" und „KD_VERW" (Kundenverwaltung)

Nun stellt sich erst einmal die Frage, welcher Audit-Modus nach dem Upgrade aktiv ist: Unified Auditing, der alte 11g-Audit-Modus oder der Mixed Mode, unter dem beide Optionen (teilweise) aktiviert sind? Ein Select liefert „FALSE" *(siehe Listing 1)* und der nächste bringt keine Zeilen zurück *(siehe Listing 2)*.

Also ist noch der alte Modus aktiv. Das steht auch so im Oracle Database Upgrade Guide 12c Release 2 *[1]*. Was alles auditiert wird, lässt sich in diesem Fall über folgenden Select überprüfen *(siehe Listing 3)*.

Der DBA entscheidet sich für den Umstieg auf Unified Auditing und aktiviert diesen Modus, wie im Upgrade-Guide beschrieben. Nach dem Neustart steht der Wert für den Parameter „Unified Auditing" in „v$option" auf „true". Bei den beiden anderen Selects hat sich zwar noch nichts geändert, der alte Modus ist jedoch deaktiviert und Statements wie „audit create trigger by kunden" bleiben folgenlos.

User für das Audit-Management einrichten

Es empfiehlt sich, für das Audit einen eigenen User einzurichten, damit die Verwaltung übersichtlicher und sicherer wird. Die Eigentümer von Objekten können beim Unified Auditing das Audit für ihre Objekte nicht mehr selbst festlegen oder ein- beziehungsweise ausschalten *(siehe Listing 4)*. Um den Audit-Trail selektieren zu können, reichen die Rolle „audit_viewer" oder das Recht „select any dictionary".

Audit-Strukturen unter 12.2

Der Audit-Trail gehört dem gesperrten „Read Only"-Schema „AUDSYS". Die Benutzung des Syslog oder die Speicherung der Standard-Audit-Daten im Betriebssystem ist im reinen Unified Auditing Mode nicht mehr möglich.

Im Release 1 wurden die Audit-Daten noch in sogenannten „common logging infrastructure queues" in der SGA gesammelt und erst nach Erreichen einer bestimmten Größe (per Default 1 MB) in

```
SELECT value FROM v$option
WHERE parameter = 'Unified Auditing';
```

Listing 1

```
SELECT policy_name, audit_option, object_schema, object_name
FROM audit_unified_policies
WHERE policy_name IN (SELECT policy_name
                      FROM audit_unified_enabled_policies)
ORDER BY policy_name;
```

Listing 2

```
SELECT audit_option FROM dba_stmt_audit_opts
UNION
SELECT privilege FROM dba_priv_audit_opts
UNION
SELECT owner||'.'||object_name FROM dba_obj_audit_opts;
```

Listing 3

```
GRANT CREATE SESSION, CREATE TABLE, CREATE JOB, CREATE VIEW,
      CREATE PROCEDURE, AUDIT_ADMIN, SELECT_CATALOG_ROLE
TO udita IDENTIFIED BY <pwd>;
GRANT SELECT ON unified_audit_trail TO udita;
GRANT EXECUTE ON DBMS_AUDIT_MGMT TO udita;
```

Listing 4

eine Tabelle mit dem kryptischen Namen „CLI_SWP$xxxxxxx$1$1“ (xxxxxxx: variabler Teil) geschrieben.

Unter 12.2 sind die Daten gleich in einer Tabelle namens „AUD$UNIFIED“ im Schema „audsys“ gespeichert. Laut Security Guide *[2]* sollte es sich zwar um eine nicht-partitionierte Tabelle handeln, „wenn die Datenbank-Version kein Partitioning unterstützt“, aber auch in der Standard Edition erscheint die Tabelle unter „DBA_TAB_PARTITIONS“. Die Partitionierung basiert auf dem Zeitpunkt des Audit-Events und lässt sich in der Enterprise Edition mit der Prozedur „alter_partition_interval“ des „DBMS_AUDIT_MGMT“-Packages verändern. Per Default liegt die Tabelle im Tablespace „SYSAUX“.

Wenn man von 12.1 auf 12.2 migriert, lassen sich die Inhalte der CLI-Tabelle mit der Prozedur „treansfer_unified_audit_records“ aus dem „DBMS_AUDIT_MGMT“-Package in die neue Tabelle übertragen. Das Verschieben des Audit-Trails auf einen eigenen Tablespace war bei der Standard Edition unter der Version 12.1.0.2 noch nicht möglich *[3]*. Unter 12.2 gibt es hier keine bösen Überraschungen mehr *(siehe Listing 5)*.

Wenn die Instanz nicht zum Schreiben geöffnet ist, werden binäre Audit Files in das Verzeichnis „$ORACLE_BASE/audit/$ORACLE_SID“ geschrieben. Beim Starten der Instanz werden sie in die Audit-Tabelle hochgeladen. Um die Dateien von Zeit zu Zeit auf Betriebssystem-Ebene zu löschen, kann man die Prozedur aus dem „DBMS_AUDIT_MGMT“-Package mit dem etwas irreführenden Namen „load_unified_audit_files“ ausführen.

Housekeeping einrichten

Zum regelmäßigen Aufräumen der Audit-Daten braucht man zunächst einen Job, der den Zeitpunkt setzt, bis zu dem die Daten gelöscht werden sollen *(siehe Listing 6)*. Danach wird der eigentliche Purge-Job eingerichtet *(siehe Listing 7)*.

Wenn die Daten archiviert werden sollen, muss man hier ein Backup auf einem anderen Tablespace oder im Betriebssystem einbauen. Das Ergebnis lässt sich über die Views „dba_audit_mgmt_last_arch_ts“ und „dba_scheduler_jobs“ beziehungsweise „dba_scheduler_job_run_details“ kontrollieren.

```
BEGIN
   DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_LOCATION(
       audit_trail_type => dbms_audit_mgmt.audit_trail_unified,
   audit_trail_location_value => 'AUDIT_TS');
END;

SELECT parameter_value
FROM dba_audit_mgmt_config_params
WHERE audit_trail = 'UNIFIED AUDIT TRAIL'
AND parameter_name = 'DB AUDIT TABLESPACE';

PARAMETER_VALUE
----------------------
AUDIT_TS
```

Listing 5

```
CREATE OR REPLACE PROCEDURE prc_set_last_archive_ts AS
BEGIN
   DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP(
       dbms_audit_mgmt.audit_trail_unified,
          TRUNC(SYSDATE)-31);
END;

BEGIN
   DBMS_SCHEDULER.CREATE_JOB (
                  job_name => 'SET_LAST_ARCHIVE_TIMESTAMP',
                  job_type => 'STORED_PROCEDURE',
                  job_action => 'PRC_SET_LAST_ARCHIVE_TS',
                  start_date => sysdate,
               repeat_interval => 'freq=daily;byhour=0;',
                  enabled => TRUE);
   DBMS_SCHEDULER.RUN_JOB (
                  job_name => 'SET_LAST_ARCHIVE_TIMESTAMP';
       use_current_session => FALSE);
END;
```

Listing 6

```
BEGIN
   DBMS_AUDIT_MGMT.CREATE_PURGE_JOB(
          audit_trail_type =>
              DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
   audit_trail_purge_interval => 24,
       audit_trail_purge_name => 'DAILY_PURGE_JOB',
       use_last_arch_timestamp => TRUE);
END;
```

Listing 7

Session-Audit konfigurieren

Bei einer frisch installierten 12c-Datenbank sind die beiden Policies „ORA_LOGON_FAILURES“ und „ORA_SECURECONFIG“ per Default aktiv. Nach einem Upgrade muss man sich selber um das Einrichten und Aktivieren der Policies kümmern. Das geht einfach mit „AUDIT POLICY ora_logon_failures;“. Nach ein paar Tests stellt der DBA allerdings verblüfft fest, dass auch erfolgreiche Logons immer noch auditiert werden, und erstellt sich dann doch eine eigene Policy *(siehe Listing 8)*.

Für die Auswertung des Session-Audits legt man sich am besten eigene Views zu. Die View „unified_audit_trail“ ist ein Monster mit 99 Spalten, von de-

```
NOAUDIT POLICY ora_logon_failures;
CREATE AUDIT POLICY my_failed_logins ACTIONS LOGON;
AUDIT POLICY my_failed_logins WHENEVER NOT SUCCESSFUL;
```

Listing 8

```
CREATE OR REPLACE VIEW trail AS
SELECT os_username,userhost,dbusername,
   REGEXP_SUBSTR(authentication_type, '[^=();]*',1,24) ip_addr,
       TO_CHAR(event_timestamp,'dd.mm.rr hh24:mi:ss') datum,
       action_name, return_code, sql_text, sessionid, scn,
       dblink_info, client_program_name,dbproxy_username,
       external_userid, client_identifier,
       object_schema,object_name,system_privilege_used,
       system_privilege, object_privileges, role,
       unified_audit_policies, audit_option
FROM unified_audit_trail
ORDER BY event_timestamp DESC;

CREATE OR REPLACE VIEW failed_logins AS
SELECT os_username, userhost, dbusername, datum, ip_addr, dblink_info,
client_program_name,return_code
FROM trail
WHERE unified_audit_policies = 'MY_FAILED_LOGINS';
```

Listing 9

```
CREATE AUDIT POLICY kunden_dml_pol
   ACTIONS INSERT ON kunden.kundendaten,
       UPDATE ON kunden.kundendaten,
       DELETE ON kunden.kundendaten;

AUDIT POLICY kunden_dml_pol;

CREATE OR REPLACE VIEW kundendaten_dml_audit AS
SELECT os_username, userhost, dbusername, client_program_name, datum,
action_name, return_code, sql_text
FROM trail
WHERE unified_audit_policies = 'KUNDEN_DML_POL';
```

Listing 10

```
CREATE AUDIT POLICY ddl_pol
ACTIONS CREATE TABLE, DROP TABLE, ALTER TABLE, TRUNCATE TABLE, CREATE
PROCEDURE, DROP PROCEDURE, CREATE VIEW, DROP VIEW, ALTER VIEW, CREATE
INDEX, DROP INDEX, CREATE TRIGGER, ALTER TRIGGER, DROP TRIGGER
WHEN 'SYS_CONTEXT(''USERENV'',''CURRENT_SCHEMA'')
       IN (''KUNDEN'', ''KD_VERW'')' EVALUATE PER STATEMENT;
AUDIT POLICY ddl_pol;

CREATE OR REPLACE VIEW ddl_audit AS
SELECT os_username, userhost, dbusername, object_schema, object_name,
client_program_name, ip_addr, datum, action_name, return_code, system_
privilege_used, sql_text
FROM trail
WHERE unified_audit_policies = 'DDL_POL';
```

Listing 11

nen sich fast die Hälfte auf Extra-Optionen wie Database Vault, Label Security, FGA und Real Application Testing beziehen, die in der Standard Edition nicht verfügbar sind. Der DBA erstellt eine View für die wichtigsten Spalten des Unified-Audit-Trails und eine darauf basierende für die fehlgeschlagenen Logins *(siehe Listing 9)*.

DML-Aktionen auditieren

Die folgende Policy überwacht Änderungen an der Tabelle „kundendaten“. Auch hierfür kann man eine eigene View einrichten. Änderungen an Objekt-Audit-Optionen greifen sofort, nicht erst in der nächsten Session *(siehe Listing 10)*.

Die Policy ist übrigens nicht unabhängig vom User-Schema und dem Objekt. Wenn man die Tabelle oder sogar den User droppt, sieht es zwar so aus, als wäre die Policy noch aktiv, das ist jedoch nicht der Fall. Man muss sie deaktivieren, löschen, neu erstellen und reaktivieren, nachdem man die Tabelle (und vorher gegebenenfalls den User) neu erstellt hat. Das gilt auch, wenn der User mit aktivierter Policy exportiert und wieder importiert wurde.

DDL-Aktionen auditieren

Die Policy „ORA_SECURECONFIG“ auditiert nur einen Teil der Aktionen, die überwacht werden sollen. Es fehlen folgenden Aktionen:

- Audit von Triggern, auch mit „ANY“-Rechten
- Audit von Views, Sequenzen, Indizes, Typen, Materialized Views und Materialized View Logs
- Audit von „TRUNCATE TABLE“-Aktionen
- Audit von „DDL“-Aktionen im eigenen Schema

Die Policy „ddl_pol“ soll die wichtigsten „DDL“-Aktionen in den Applikationsschemata „kunden“ und „kd_verw“ aufzeichnen. „PROCEDURE“ steht auch für „FUNCTION“ und „PACKAGE“. Auch „DDL“-Aktionen im Zusammenhang mit Datapump-Importen werden hier erfasst *(siehe Listing 11)*.

Wichtig: Wenn man den Quotation-Operator einsetzen will, darf die „WHEN“-Klausel keinen Zeilenumbruch enthalten, sonst bekommt man eine

nicht wirklich hilfreiche Fehlermeldung *(siehe Listing 12)*.

Der DBA möchte das DDL-Audit auf alle User der Datenbank erweitern und auch die Benutzung der Packages „UTL_FILE“ und „UTL_SMTP“ miteinbeziehen. Das ist auch nachträglich ohne Probleme möglich *(siehe Listing 13)*.

```
....WHEN q'[SYS_CONTEXT('USERENV','CURRENT_SCHEMA')
      IN ('KUNDEN', 'KD_VERW')]'....
ORA-46368: Audit-Policy enthält keine einfache Regelbedingung.
```

Listing 12

```
ALTER AUDIT POLICY ddl_pol CONDITION DROP;
ALTER AUDIT POLICY ddl_pol ADD ACTIONS
EXECUTE ON sys.utl_file, EXECUTE ON sys.utl_smtp,;
```

Listing 13

Administrative Aktionen auditieren

Die Policy „ORA_SECURECONFIG“ deckt einen großen Teil der DBA-Aktionen ab. Es fehlen aber zum Beispiel:

- Erstellen, Löschen und Verändern von Tablespaces
- Erstellung von PFILES
- Rechtevergabe unter den Normal-Usern, wenn etwa der User „kd_verw“ dem User-Kunden Objektrechte verleiht beziehungsweise entzieht

Insofern ist es auch hier interessant, sich eine eigene Policy für Änderungen an der Datenbank-, User- und Rechteverwaltung sowie andere nicht alltägliche Aktionen zu erstellen. Alle Statement- und Privilege-Audit-Optionen bleiben für die Dauer der Session aktiv. Änderungen an den Audit-Optionen greifen erst mit Beginn der nächsten Session. Stopps und Starts der Instanz sowie Änderungen am Audit-Management werden nicht über eine Policy auditiert. Sie fallen unter das obligatorische Audit *(siehe Listing 14)*.

```
CREATE AUDIT POLICY admin_aktionen_pol
PRIVILEGES CREATE EXTERNAL JOB, CREATE JOB, DROP PUBLIC SYNONYM, CREATE
PUBLIC SYNONYM, PURGE DBA_RECYCLEBIN
ACTIONS ALTER DATABASE, ALTER SYSTEM, CREATE PFILE, CREATE TABLESPACE,
ALTER TABLESPACE, DROP TABLESPACE, CREATE USER, ALTER USER, DROP USER,
GRANT, REVOKE, CREATE ROLE, ALTER ROLE, DROP ROLE, SET ROLE, CREATE
PROFILE, ALTER PROFILE, DROP PROFILE, CREATE DATABASE LINK, ALTER DATA-
BASE LINK, DROP DATABASE LINK, CREATE DIRECTORY, DROP DIRECTORY;

AUDIT POLICY admin_aktionen_pol;

CREATE OR REPLACE VIEW admin_audit AS
SELECT os_username,userhost, dbusername, datum, action_name, return_
code, client_program_name, object_schema, sql_text,
object_name, system_privilege_used, system_privilege, role
FROM trail
WHERE unified_audit_policies = 'ADMIN_AKTIONEN_POL'
OR unified_audit_policies IS NULL;
```

Listing 14

Fazit

Mit vier Policies und den dazugehörigen Views hat der DBA nun erst einmal alle Audit-Wünsche seines Chefs erfüllt. Bevor diese Audit-Konfiguration auf die produktive Datenbank übertragen werden kann, sollte das Ganze allerdings ausgiebig getestet werden, um die Performance zu überprüfen, etwaige Lücken festzustellen und die Frequenz des Housekeepings anzupassen.

In der Version 12.1 gab es verschiedene Probleme mit ausufernden LOB-Segmenten des Audit-Trails *[4, 5]* – allerdings nur, wenn der Unified-Audit-Trail noch im „SYSAUX“-Tablespace lag. Die Größe der Tabelle „AUD$UNIFIED“ sollte man also immer im Auge behalten.

Wichtig ist auch, die neuesten PSU-Patches immer zeitnah einzuspielen, auch wenn das Unified Auditing in der Version 12.2 längst nicht mehr so wirkt, als sei es eine ständige Baustelle. Mit den neuen Verbesserungen ist das Unified Auditing auch für die Standard Edition attraktiv geworden.

Quellen

[1] Oracle Database Upgrade Guide 12c Release 2 (12.2) E49634-15, 4.6.14.1 Understanding Unified Auditing Migration Process for Oracle Database

[2] Oracle Database Security Guide 12c Release 2 (12.2), E49708-20, 23.1.1 When and Where Are Audit Records Created?

[3] MOS-Note 1925415.1: ORA-46099, “Dbms_audit_mgmt.set_audit_trail_location” Not Working

[4] Bug 18109788; Cleanup of unifies audit trail does not release lob segment space

[5] Bug 25615191: can’t reclaim free space in sysaux tablespace after purging audit records 12.1

Elke Fritsch
elke.fritsch@muniqsoft.de

Oracle Database Cloud Performance – Konsistenz

Randolf Geist

In der letzten Ausgabe haben wir die Oracle Database Cloud (DBaaS) mit besonderem Augenmerk auf die Performance der zur Verfügung gestellten Datenbanken evaluiert. Dieser Artikel zeigt einige Testergebnisse bezüglich der Konsistenz der Performance, also wie konsistent die Plattform eine bestimmte Performance liefern kann.

Es wurden unterschiedliche Tests über einen längeren Zeitraum durchgeführt, typischerweise mindestens eine Woche am Stück, und die entsprechenden Laufzeiten pro Testdurchlauf protokolliert. Damit lassen sich entsprechende Schwankungen sehr gut aufzeigen, zum Beispiel Leistungsschwankungen zu bestimmten Tages- oder Wochenzeiten. Dazu wurden folgende Tests durchgeführt:

- CPU-gebundener Test mittels PL/SQL-Schleife – also Auslastung über die PL/SQL-Engine
- CPU-gebundener Test mittels SQL-Engine (logical I/O)
- I/O-gebundener Test mit maximalem physischen I/O (direct + asynchronous), nur Lesen
- I/O-gebundener Test mit maximalem physischen I/O (direct + asynchronous), Schreiben und Lesen

Dieselben Tests wurden zum Vergleich auf einem dedizierten Host durchgeführt, um eine Vergleichsbasis der gebotenen Konsistenz zu haben. Der dedizierte Host verfügte nur über vier Cores, während die Oracle-DBaaS-Umgebung, die für den Test zum Einsatz kam, acht Cores hatte.

Für den Storage im dedizierten Host wurde eine Samsung-SSD-Festplatte eingesetzt. Bei den I/O-Tests wurde ein minimaler Buffer Cache verwendet, um sicherzustellen, dass der physische I/O maximiert wird. Außerdem wurde dafür gesorgt, dass „Direct I/O und Asynchronous I/O" verwendet wird, um keine Verfälschungen des Ergebnisses durch sekundäres Caching durch das Filesystem zu bekommen und damit die maximale Leistung des I/O-Systems ausgenutzt werden kann. Die *Abbildungen 1 und 2* zeigen die Testergebnisse für Test 1

```
declare
   n number;
begin
   loop
       n := 0;
       for i in 1..1000000000 loop
           n := n + 1;
       end loop;
       insert into timings(testtype, thread_id, ts)
       values ('PLSQL', &thread_id, systimestamp);
       commit;
   end loop;
end;
/
```

Listing 1

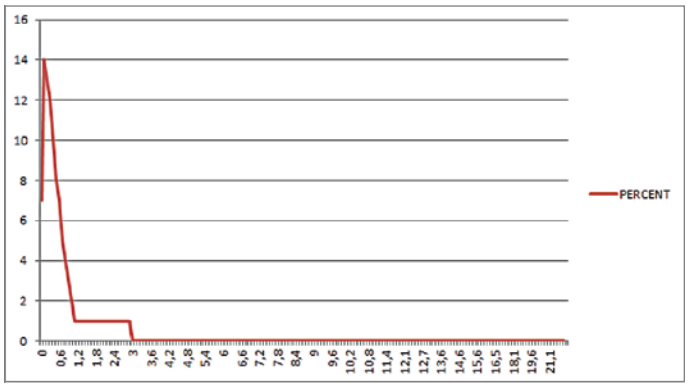


Abbildung 1: Gesamt-Konsistenz Oracle DBaas

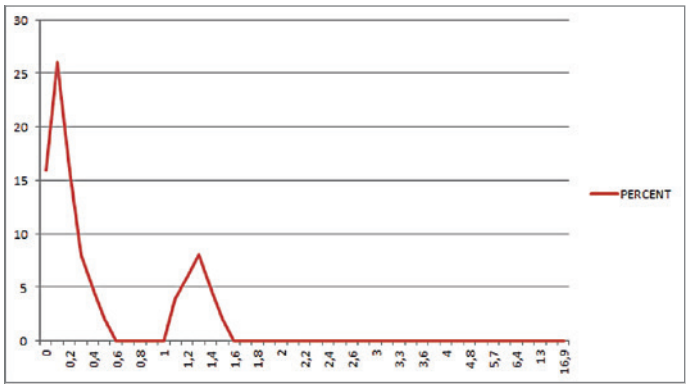


Abbildung 2: Gesamt-Konsistenz dedizierter Host

(siehe Listing 1) bei der CPU-Auslastung über die PL/SQL-Engine.

Jeder Lauf hat ungefähr zwischen 35 und 50 Sekunden gedauert, je nach Umgebung, da die Cores des dedizierten Hosts langsamer waren als die der Oracle-DBaaS-Umgebung. Da die Tests mehrere Tage lang liefen, kamen mehrere Tausend solcher Testläufe zusammen. Die Grafiken zeigen an, wie viel Prozent der Läufe wie viel Prozent vom Mittelwert der Laufzeit abgewichen sind, zum Beispiel haben mehr als 25 Prozent der Läufe (x-Achse) beim dedizierten Host 0,1 Prozent (y-Achse) Abweichung vom Mittelwert gehabt.

Die Oracle-DBaaS-Umgebung zeigt eine ähnliche Konsistenz, wenn auch die Spreizung der Abweichungen etwas größer ist und es auch extremere Ausreißer gibt, allerdings in beiden Umgebungen bei nur wenigen Läufen. Beim dedizierten Host gibt es dafür im Bereich zwischen 1 und 1,5 Prozent Abweichung nochmal eine signifikante Delle in der Kurve, die so in der Oracle-DBaaS-Umgebung nicht beobachtet wurde. Die *Abbildungen 3 und 4* zeigen die gleiche Betrachtung pro Tag.

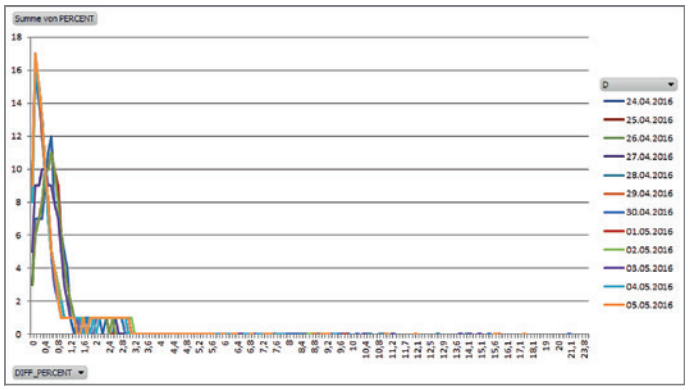


Abbildung 3: Konsistenz pro Tag, Oracle DBaaS

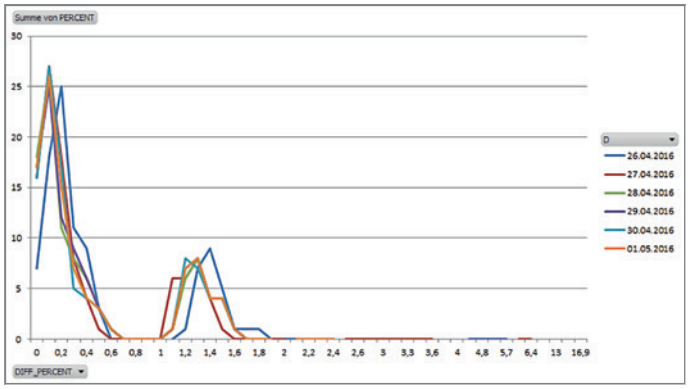


Abbildung 4: Konsistenz pro Tag, dedizierter Host

In beiden Umgebungen gibt es keine extremen Unterschiede zwischen den einzelnen Tagen, auch wenn es in der Oracle-DBaaS-Umgebung etwas größere Abweichungen als auf dem dedizierten Host gibt. Abschließend zeigen die *Abbildungen 5 und 6* den Verlauf der absoluten Laufzeiten der einzelnen Threads über die Zeit.

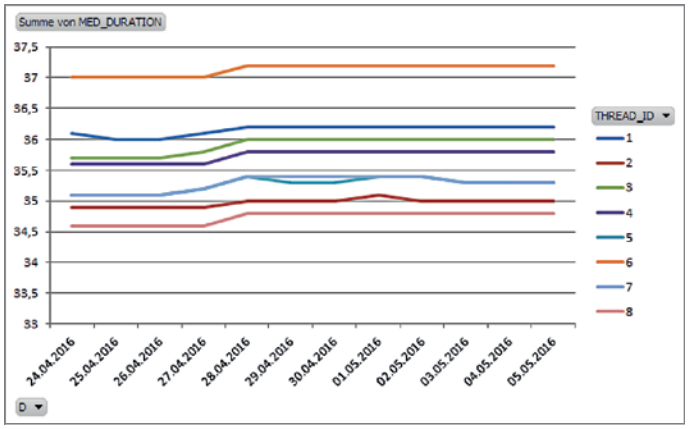


Abbildung 5: Oracle DBaaS

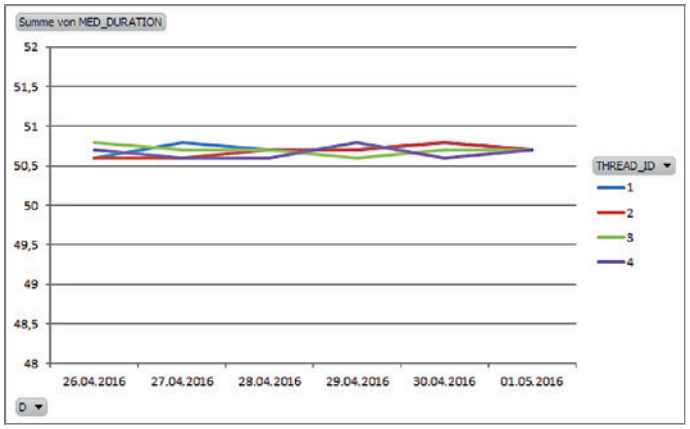


Abbildung 6: Dedizierter Host

```
alter session set "_old_connect_by_enabled" = true;

declare
  n number;
begin
  loop
      select
      count(*) X
      into n
      from
      kill_cpu&tabname
      connect by
      n > prior n
      start with
      n = 1;
      insert into timings(testtype, thread_id, ts)
      values ('SQLLIO', &thread_id, systimestamp);
      commit;
  end loop;
end;
/
```

Listing 2

Hier ist interessant zu beobachten, dass in der Oracle-DBaaS-Umgebung die einzelnen Threads zwar eine recht konsistente Laufzeit pro Tag haben, aber jeder Thread etwas unterschiedlich schnell ist, während dies auf dem dedizierten Host so nicht reproduziert werden kann. Die *Abbildungen 7 und 8* zeigen die Testergebnisse für Test 2 *(siehe Listing 2)* bei der CPU-Auslastung über die SQL-Engine.

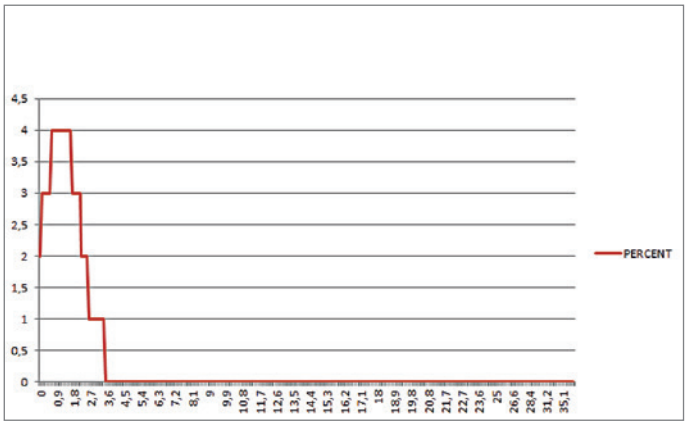


Abbildung 7: Gesamt-Konsistenz Oracle DBaas

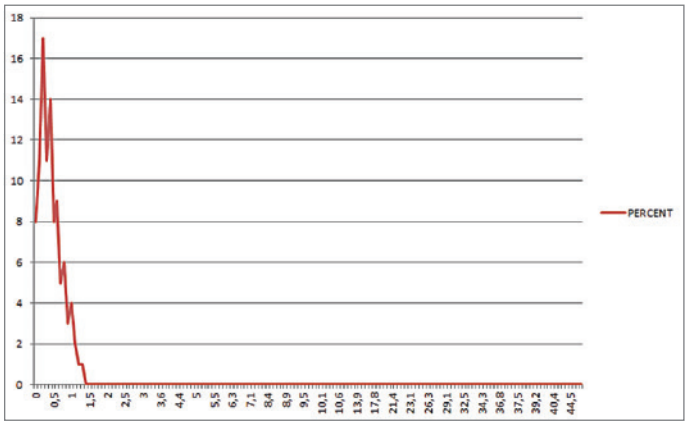


Abbildung 8: Gesamt-Konsistenz dedizierter Host

Beim Test der CPU-Auslastung mithilfe der SQL-Engine zeigt sich eine größere Gesamt-Abweichung vom Mittelwert in der Oracle-DbaaS-Umgebung, auch wenn es beim dedizierten Host zu extremeren Ausreißern kommt (bis zu über 44 Prozent Abweichung vom Mittelwert, *siehe Abbildungen 9 und 10)*.

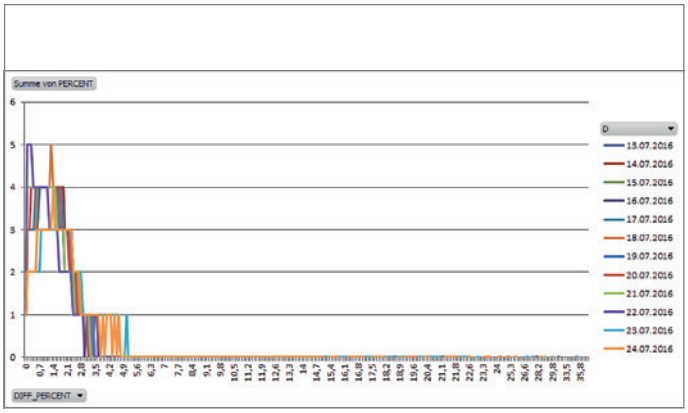


Abbildung 9: Konsistenz pro Tag, Oracle DBaaS

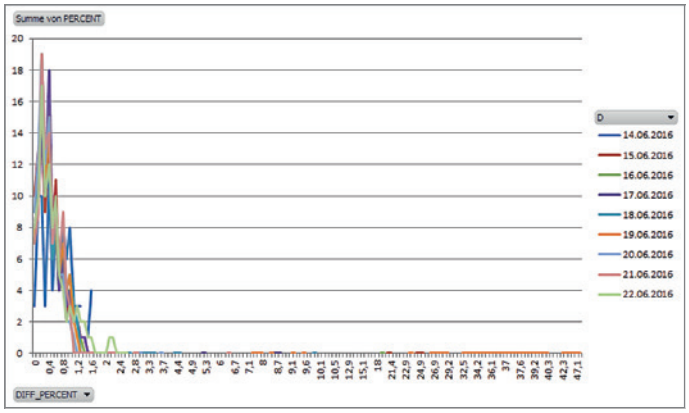


Abbildung 10: Konsistenz pro Tag, dedizierter Host

Auch bei diesem Test zeigt sich bei der Betrachtung der Konsistenz pro Tag in beiden Umgebungen kein allzu signifikanter Unterschied. Die Abbildungen 11 und 12 zeigen abschließend wiederum die absoluten Laufzeiten pro Thread.

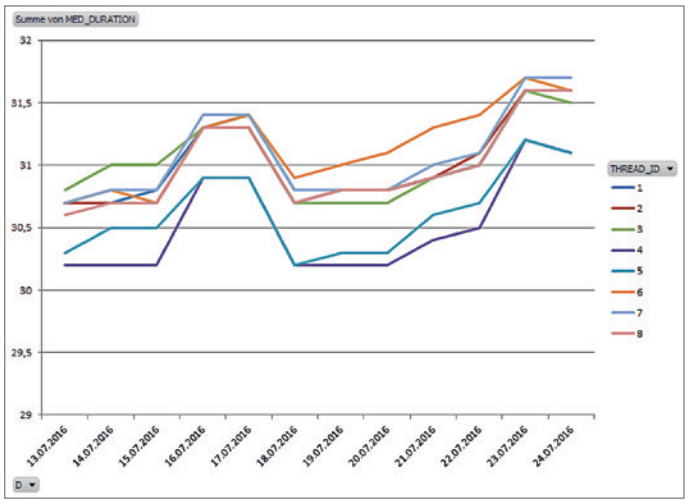


Abbildung 11: Oracle DBaaS

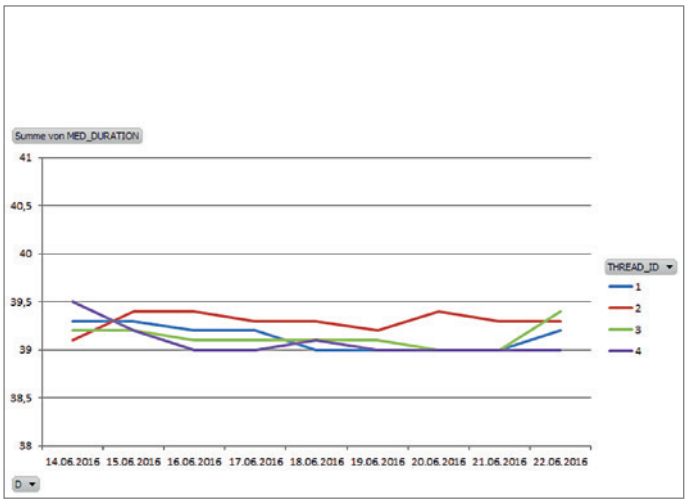


Abbildung 12: Dedizierter Host

Hier zeigt sich ein ähnlicher Effekt wie bei Test 1 in der Oracle DBaas-Umgebung – die einzelnen Threads sind unterschiedlich schnell, schwanken aber in einer ähnlichen Art und Weise pro Tag. Die Schwankungen in der Oracle-DBaaS-Umgebung sind bei diesem Test größer als auf dem dedizierten Host. In den *Abbildungen 13 und 14* sind die Testergebnisse für Test 3 *(siehe Listing 3)* dargestellt, ein I/O-gebundener Test mit maximalem physischen I/O, nur Lesen.

```
declare
   n number;
begin
   loop
       select /*+
              leading(t_o)
              use_nl(t_i)
              index(t_o)
              index(t_i)
          */
          sum(t_i.n)
          into n
       from
          t_o
          , t_i&tabname t_i
       where
          t_o.id_fk = t_i.id;
       insert into timings(testtype, thread_id, ts)
       values ('&testtype', &thread_id, systimestamp);
       commit;
   end loop;
end;
/
```

Listing 3

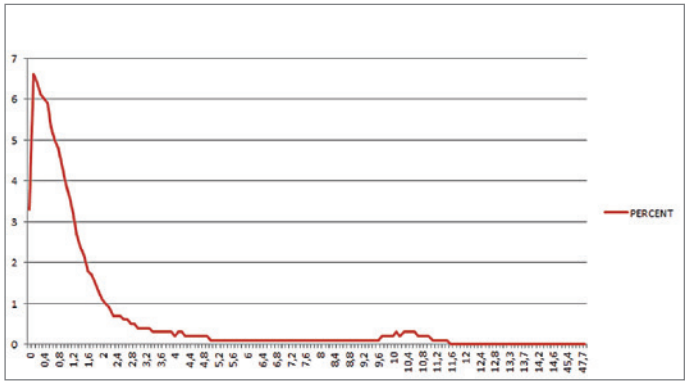


Abbildung 13: Gesamt-Konsistenz, Oracle DBaaS

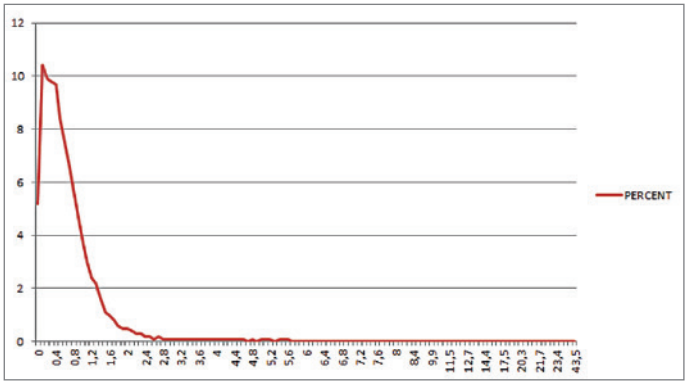


Abbildung 14: Gesamt-Konsistenz, dedizierter Host

Die Profile der beiden Umgebungen sind sehr ähnlich, aber die Spreizung der Abweichungen ist in der Oracle-DBaaS-Umgebung größer als beim dedizierten Host. Die maximalen Ausreißer sind recht vergleichbar und auch hier in beiden Umgebungen nur durch vereinzelte Läufe verursacht. Die Abbildungen 15 und 16 zeigen die gleiche Betrachtung pro Tag.

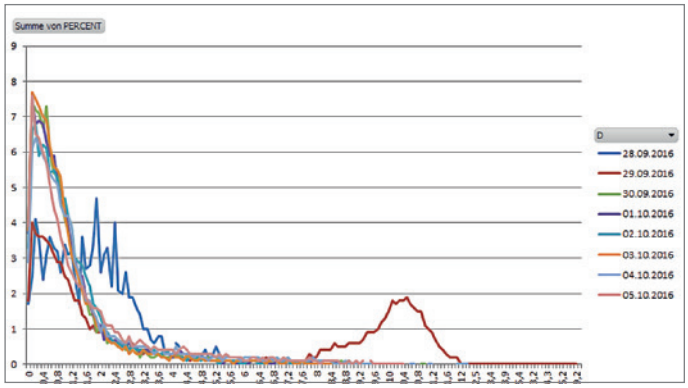


Abbildung 15: Oracle DBaaS

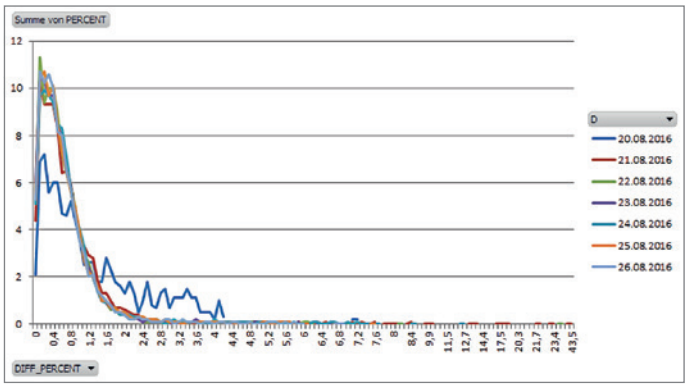


Abbildung 16: Dedizierter Host

Hier wird zum ersten Mal deutlich, dass es signifikante Unterschiede zwischen den einzelnen Tagen geben kann – in beiden Umgebungen. Insbesondere in der Oracle-DBaaS-Umgebung stechen zwei Tage hervor, einer der beiden ist für die Delle bei der Gesamtkonsistenz-Betrachtung zwischen 8 und 12 Prozent Abweichung verantwortlich und auch maßgeblich an den maximalen Ausreißern beteiligt. Die Abbildungen 17 und 18 zeigen die absolute Laufzeit der Threads.

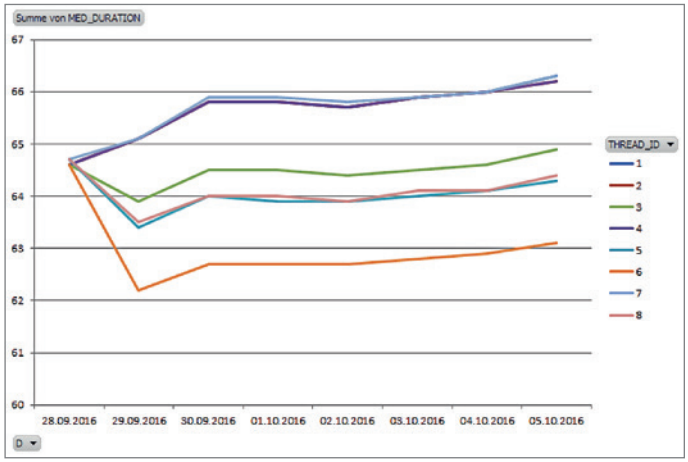


Abbildung 17: Oracle DBaaS

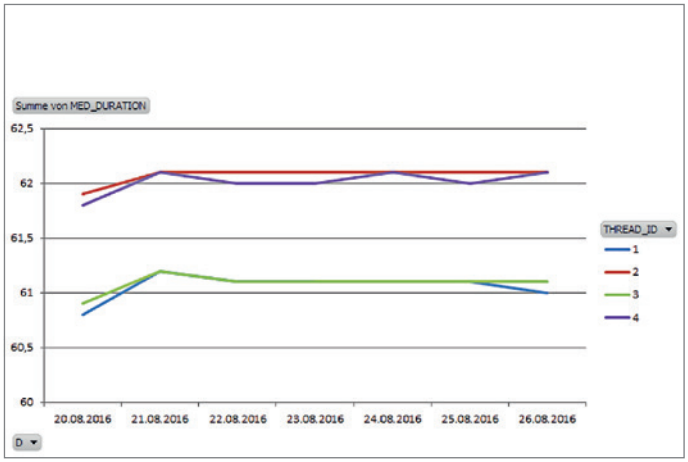


Abbildung 18: Dedizierter Host

Auch hier tritt wieder der Effekt der unterschiedlichen Laufzeiten pro Thread auf, diesmal auch auf dem dedizierten Host, auch wenn die Unterschiede absolut gesehen in beiden Umgebungen nicht sehr signifikant sind. In den *Abbildungen 19 und 20* sind die Testergebnisse für Test 4 *(siehe Listing 4)* gezeigt, der I/O-gebundene Test mit maximalem physischen I/O, Schreiben und Lesen.

```
begin
  loop
      for rec in (
              select /*+
                  index(t_o)
              */
              id_fk
      from
              t_o
      ) loop
              update t_i&tabname t_i
              set n = rec.id_fk
              where id = rec.id_fk;
      end loop;
      insert into timings(testtype, thread_id, ts)
      values ('&testtype', &thread_id, systimes-
tamp);
      commit;
  end loop;
end;
/
```

Listing 4

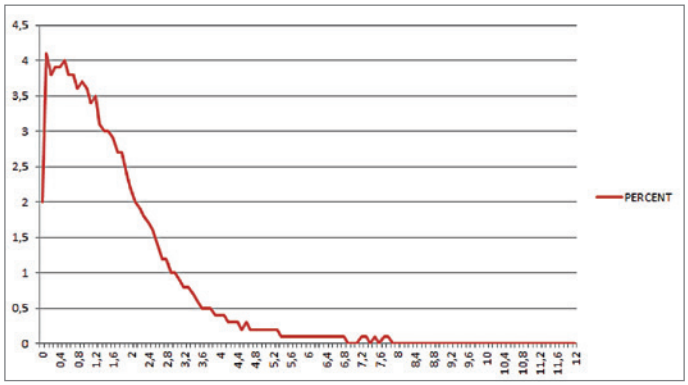


Abbildung 19: Die Gesamt-Konsistenz, Oracle DBaaS

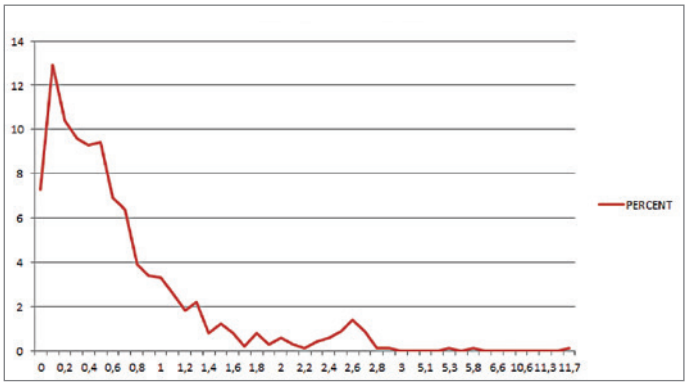


Abbildung 20: Dedizierter Host

Bei diesem Test gibt es einen signifikanten Unterschied in den Abweichungen zwischen den beiden Umgebungen: Die Spreizung in der Oracle-DBaaS-Umgebung ist deutlich größer als auf dem dedizierten Host. Der Grund dafür liegt in der limitierten Schreib-Performance des Systems; ein entsprechender AWR-Bericht der Oracle-DBaaS-Umgebung macht das Problem offensichtlich *(siehe Abbildungen 21 und 22)*.

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
free buffer waits	1,607,793	14.2K	9	49.0	Configuration
db file sequential read	23,513,342	13K	1	44.9	User I/O
DB CPU		2462.6		8.5	
log file switch (private strand flush incomplete)	25	18.8	751	.1	Configuration
log file sync	457	.8	1	.0	Commit
control file sequential read	750	.3	0	.0	System I/O
direct path write temp	2	.2	81	.0	User I/O
undo segment extension	5	.1	30	.0	Configuration
Disk file operations I/O	40	.1	3	.0	User I/O
latch: shared pool	493	.1	0	.0	Concurrency

Abbildung 21: Oracle DBaaS

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	39,461,303	26,758	1	92.57	User I/O
DB CPU		1,319		4.56	
free buffer waits	199,906	449	2	1.56	Configuration
cr request retry	136,608	2	0	0.01	Other
lock deadlock retry	63,255	1	0	0.00	Other

Abbildung 22: Dedizierter Host

Während also beim dedizierten Host mehr als 90 Prozent der Datenbank-Zeit mit Lesevorgängen verbracht wird, muss in der Oracle-DBaaS-Umgebung fast 50 Prozent der Datenbank-Zeit mit Warten auf freie Blöcke im Cache verbracht werden, was bedeutet, dass der DB-Writer-Prozess die veränderten Blöcke nicht schnell genug wegschreiben kann, um Platz für neue Blöcke zum Lesen zu schaffen. Das passiert in dieser Form nicht auf dem dedizierten Host, dort können die Daten offensichtlich schnell genug weggeschrieben werden. Die Betrachtungen pro Tag liefern hier keine größeren Erkenntnisse *(siehe Abbildungen 23 und 24)*.

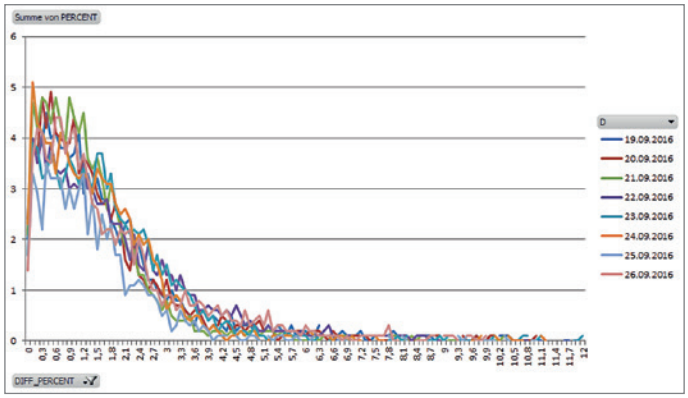


Abbildung 23: Oracle DBaaS

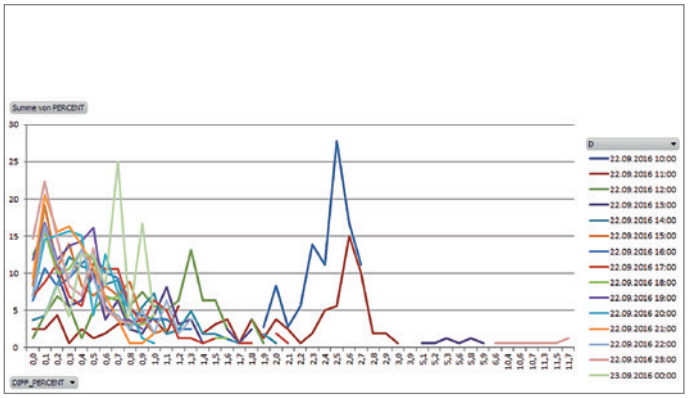


Abbildung 24: Dedizierter Host

Beim dedizierten Host ist die Grafik nicht wirklich vergleichbar mit der Oracle-DBaaS-Umgebung, da der Test im Gegensatz zur Oracle-DBaaS-Umgebung nur für einige Stunden lief. Beim Vergleich der absoluten Laufzeiten der einzelnen Threads wird wiederum deutlich, dass hier die Oracle-DBaaS-Umgebung zum ersten Mal deutlich langsamer als der dedizierte Host ist *(siehe Abbildungen 25 und 26)*.

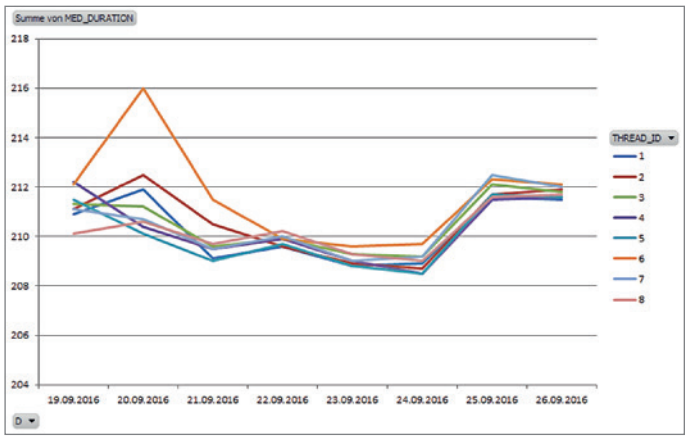


Abbildung 25: Oracle DBaaS

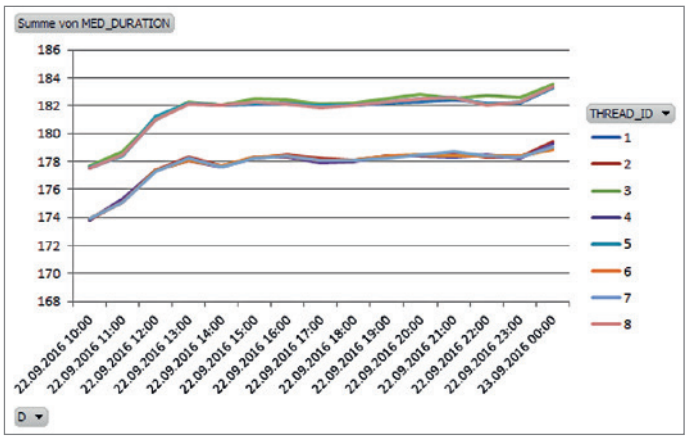


Abbildung 26: Dedizierter Host

Fazit

Sowohl im CPU- als auch im Storage-Bereich zeigt die Oracle-DBaaS-Umgebung größere Spreizungen in den Abweichungen als die dedizierte Umgebung, die zum Vergleich herangezogen wurde. Dies ist erst einmal erwartungsgemäß, da es sich von vornherein um einen unfairen Vergleich gehandelt hat – also virtuelle gegen dedizierte Umgebung; ein fairer Vergleich hätte mit einer ähnlich virtualisierten Umgebung stattfinden müssen.

Nichtsdestotrotz zeigt die Oracle-DBaaS-Umgebung selbst im Vergleich zur dedizierten Umgebung zumindest im Testzeitraum durchaus vergleichbare Ergebnisse, was sicherlich positiv und nicht als selbstverständlich zu bewerten ist.

Randolf Geist
randolf.geist@oracle-performance.de

Tipps und Tricks aus Gerds Fundgrube: Die Forms-Gemeinschaft startet durch

Gerd Volberg, OPITZ CONSULTING Deutschland GmbH

Die deutschsprachige Forms-Szene ist seit dem Erscheinen von Forms 12*c* wie verwandelt. Inzwischen sind viele Projekte von und für Forms-Enthusiasten entstanden.

Lange Zeit machte die Forms-Szene den Eindruck, im tiefsten Dornröschenschlaf zu liegen. Mit der Veröffentlichung des lang ersehnten Release 12*c* im Oktober 2015 kam endlich neuer Schwung in die Forms-Gemeinschaft: Diverse Stammtische in vielen Bundesländern diskutierten frische neue Themen, auf dem DOAG DevCamp 2016 gab es einen regelrechten Forms-Hype. Letztendlich entschied sich die DOAG im Herbst 2016, den Schwerpunkt enger zu begleiten – mit Jürgen Menge als DOAG-Themenverantwortlicher für das Fachgebiet Forms/Reports.

Richtig los ging es im Jahr 2016 auf der DOAG-Jahreskonferenz in Nürnberg. Dort hatten Forms-Enthusiasten Raum und

Ansprechpartner	Frank Hoffmann
E-Mail	frank.hoffmann@forms12c.de
URL	https://www.forms12c.de

Tabelle 1

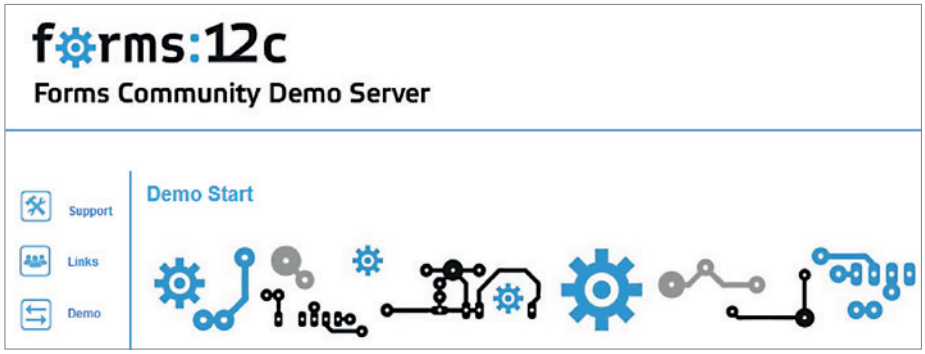


Abbildung 1: Der Forms-Demo-Server

Ansprechpartner	Gerd Volberg
E-Mail	gerd.volberg@forms12c.de
URL	https://github.com/doag/forms

Tabelle 2

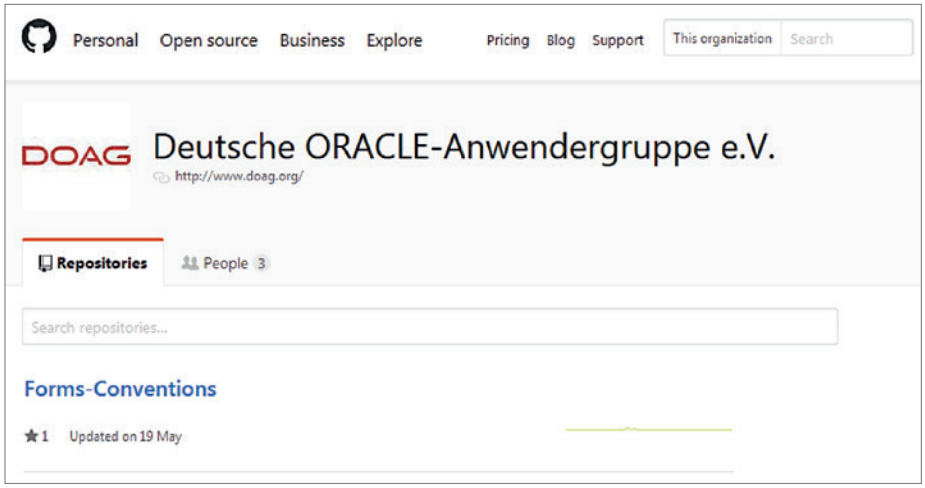


Abbildung 2: Forms GitHub

Zeit, sich mit Gleichgesinnten auszutauschen und die Weichen für die Arbeit innerhalb der Forms-Gemeinschaft zu stellen. In diesem Rahmen entstanden auch spannende Projekte von Forms-Begeisterten.

Forms-Demo-Server

Frank Hoffmann hat von Oracle eine offizielle Lizenz für einen deutschen Forms-Demo-Server bekommen, der nun über die Domänen „forms12c.de“ und „forms12c.com“ erreichbar ist *(siehe Abbildung 1)*. Auf diesem Server können Interessenten nun ihre Demo-Applikationen hosten: Ob Best Practices, Tipps & Tricks oder kleinere Demo-Applikationen – wer Lust hat, sich daran zu beteiligen und eine Demo hochzuladen, nehme mit Frank Hoffmann Kontakt auf. Alle Demos lassen sich von der Homepage aus aufrufen *(siehe Tabelle 1)*.

Forms GitHub

Der Autor hat vor einiger Zeit auf GitHub im DOAG-Space einen Bereich reserviert, den die Forms-Gemeinschaft nutzen kann. Alle Source-Codes, die auf dem Demo-Server veröffentlicht sind, lassen sich parallel im GitHub der DOAG speichern und versionieren. Demos, deren Source-Code hier nicht veröffentlicht wird, sind auf dem Forms-Server nicht zugelassen. Die Community soll völlige Transparenz haben, was Masken, Reports, Source Codes und Libraries angeht, die in den Demos genutzt werden. Unter „github.com/DOAG“ sind auch noch andere interessante Projekte zu finden *(siehe Abbildung 2)*. Hier hosten beispielsweise auch APEX- und PL/SQL-Entwickler ihre Source-Codes und Best Practices *(siehe Tabelle 2)*.

Forms auf der Oracle-Community-Plattform

Jürgen Menge hat auf der Oracle-Plattform eine Forms-Gruppe eingerichtet, um Ideen auszutauschen und über die Forms-Applikationen des Demo-Servers zu diskutieren *(siehe Abbildung 3)*. Der Space liegt in einem Unterbereich der „community.oracle.com“-Seiten. Darüber hinaus können dort Blog-Einträge, Umfragen, Ideen und Diskussionen gehostet werden *(siehe Tabelle 3)*. Last but not least gibt es noch einen Bereich mit Events.

Fazit

Diese Projekte ergänzen sehr gut die offizielle DOAG-Plattform, auf der seit März 2017 dem Themenkomplex Forms/Reports eine Webseite gewidmet ist. Unter der Subdomain „forms.doag.org“ werden aktuelle News, Webinare und Videos veröffentlicht *(siehe Tabelle 4)*. Wer up to date bleiben möchte, kann zudem auch den Forms-Newsletter abonnieren und sich so in regelmäßigen Abständen über neue Aktivitäten im Rahmen der DOAG informieren.

Forms-Interessenten erhalten auf der Themenseite auch einen Überblick über die geplanten DOAG-Veranstaltungen und finden einen Einstieg zu weiteren Informationsquellen wie zum Beispiel den hier vorgestellten Projekten. Zudem inte-

Ansprechpartner	Jürgen Menge
E-Mail	juergen.menge@doag.org
URL	https://community.oracle.com/community/other-languages/deutsche-oracle-entwickler-community/forms-developer-community

Tabelle 3

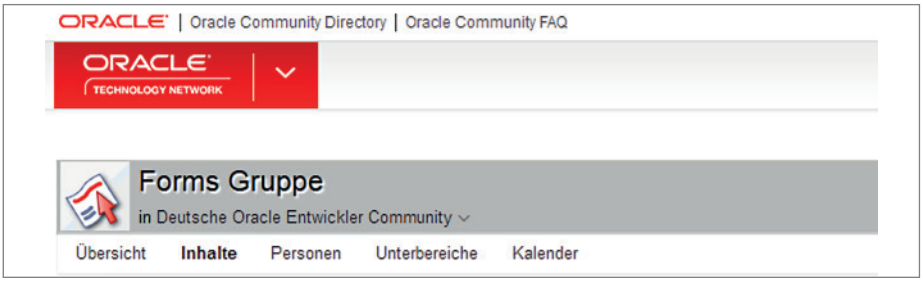


Abbildung 3: Die Forms-Community-Plattform

Themenverantwortlicher der DOAG	Jürgen Menge
E-Mail	juergen.menge@doag.org
URL	https:forms.doag.org

Tabelle 4

ressant für die Forms-Gemeinschaft und darüber hinaus: das Archiv der DOAG mit seinen unzähligen Vortragsunterlagen und Zeitschriften, in dem nun auch bequem gesucht werden kann.

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

Termine

September

05.09.2017
Regionaltreffen NRW
Martin Schmitter

07.09.2017
DOAG IMC DAY
SIG (IMW) Infrastruktur und Betriebssysteme
Berlin

08.09.2017
DOAG Datenbank Webinar:
Tuning unter Zeitdruck

11.09.2017
Regionaltreffen Thüringen
Jörg Hildebrandt
Weimar

11.09.2017
Regionaltreffen Osnabrück/Bielefeld/Münster
Andreas Kother, Klaus Günther

12.09.2017
Berliner Expertenseminar mit Johannes Ahrends: ORACLE 12C Hochverfügbarkeit mit Multitenant Database
Cornel Albert
expertenseminare@doag.org

14.09.2017
DOAG Vorstandssitzung
Bad Herrenalb

19.09.2017
Regionaltreffen Hamburg/Nord
Jan-Peter Timmermann, Benjamin Kurschies

20.09.2017
Regionaltreffen München/Südbayern
Andreas Ströbel
München

21.09.2017
Regionaltreffen Nürnberg/Franken
Martin Klier, Thomas Köppel
Nürnberg

21.09.2017
DOAG 2017 Reporting Day
SIG (DEV) Development / Oracle Tools
Kassel

21.09.2017
Regionaltreffen Nürnberg/Franken
Martin Klier, Thomas Köppel

21.09.2017
DOAG 2017 BIG DATA DAYS
SIG (DEV) Development / Oracle Tools
Kassel

22.09.2017
DOAG 2017 GEODATA DAY
SIG (DEV) Development / Oracle Tools
Kassel

25.09.2017
Regionaltreffen Halle/Leipzig
Matthias Reimann

26.09.2017
DOAG Data Integration Day
Martin Aurich, Lutz Bauer
Ratingen

26.09.2017
Berliner Expertenseminar mit Andreas Koop: ORACLE Jet Entwicklung im Enterprise
Cornel Albert
expertenseminare@doag.org

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

- Waldemar Rohde
- Christof Kaller
- Michael Hansen
- Horst Hanke
- Rainer Willems

Firmenmitglieder DOAG

- financial.com AG, Malgorzata Dyda

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): Mylène Diacquenod, Marina Fischer, Klaus-Michael Hatzinger, Sebastian Höing, Yann Neuhaus, Fried Saacke, Christian Trieb

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

Fotonachweis:
Titel: © Bakhtiar Zein/123RF
Foto S. 18: © firewings/123RF
Foto S. 28: © 1tjf/123RF
Foto S. 33: © aimage/123RF
Foto S. 45: © hollygraphic/123RF
Foto S. 48: © Felix Pergande/123RF
Foto S. 52: © bacho12345/123RF
Foto S. 58: © Len Neighbors/123RF
Foto S. 64: © szefei/123RF
Foto S. 69: © Lukas Kurka/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

Jetzt **Ticket** sichern

Early-Bird bis
28. September

2017
DOAG
Konferenz + Ausstellung
21. - 24. November in Nürnberg

PROGRAMM ONLINE
mit rund 450 Vorträgen

2017.doag.org

Eventpartner: AOUG Austrian Oracle User Group · SOUG Swiss Oracle User Group · iJUG Verbund · ORACLE

Treffen Sie die international herausragenden Orcale Datenbank-Spezialisten und profitieren Sie vom Know-how-Transfer. Erfahren Sie anhand praxisnaher Vorträge, wie Sie mehr Performance und Effizienz erreichen. Die Performance Days bieten Ihnen wertvolle Informationen zu Diagnose, Lösung und Vermeidung von Performance-Problemen bei Applikationen mit der Oracle Datenbank. Nehmen Sie an der Top-Veranstaltung teil – live in Zürich oder im Virtual Classroom.
Mehr Infos und Anmeldung unter: www.trivadis.com/tvdpdays

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

trivadis
makes IT easier.