Nr. 5 | Oktober 2019 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP

Aus der Praxis

Patching Oracle – Was muss ich tun?

Im Interview

Prof. Dr. Carlo Velten, CEO des IT-Research- und Beratungsunternehmens Crisp Research AG

Cloud Native

Reise in die chinesische Cloud


2019
DOAG
Konferenz + Ausstellung
19. - 22. November in Nürnberg
PROGRAMM ONLINE
2019.doag.org
Eventpartner:
AOUG
SOUG
swiss oracle user group
iJUG Verbund
ORACLE

Matthias Fuchs
Themenverantwortlicher
Cloud & DevOps,
DOAG Infrastruktur &
Middleware Community

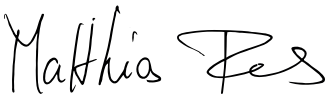
Liebe Mitglieder, liebe Leserinnen und Leser,

Cloud Native, für was steht das überhaupt? Cloud Native beschreibt den Ansatz, Applikationen optimiert für die Cloud-Computing-Architektur zu erstellen. Ziel ist es, die Vorteile der Cloud gezielt für die Anwendung zu nutzen. Ein Charakteristikum der Cloud ist die Möglichkeit, Anwendungen über viele Rechner, Standorte, Rechenzentren und Verfügbarkeitszonen zu verteilen. Microservices, kleine unabhängige Applikationen, passen für das verteilte Szenario hervorragend. Die Micoservices wiederum sind ideal für Orchestrierungsplattformen auf Basis von Docker.

Die Zeit in der Cloud ist fortgeschritten und die ersten Projekte oder Anwendungen wurden in Cloud-Native-Manier entwickelt oder migriert. Man hat die Vor- und Nachteile kennengelernt. Dabei wurden meist Frameworks wie Kubernetes oder spezielle Cloud Services verwendet.

Die DOAG will mit dieser Ausgabe Themen der Cloud ansprechen, die in Hinblick auf Cloud-Native-Implementierungen vorkommen. Die Art und Weise des Handelns ist in den Clouds der verschiedenen Hersteller ähnlich und die praktischen Tipps lassen sich in jeder Cloud verwenden.

Ihr



Cloud Native reift zu einem festen Standard in der IT-Welt

Der Einsatz von Ansible ist ein Gewinn für die Verwaltung der Exadata-Plattform

Die Multitenant-Option bietet DBAs diverse Möglichkeiten, einen Klon einer Datenbank zu erstellen.

Einleitung

3 Editorial

5 Timeline

7 „Das wichtigste Design-Kriterium ist die Hybrid- und Multi-Cloud-Fähigkeit, die durch die Cloud-Native-Technologien unterstützt wird."
Interview mit Dr. Carlo Velten

Cloud Native

12 Auf den Spuren der nächsten IT-Evolutionsstufe: Entwicklung und Grundlagen von Cloud Native
Benjamin Nothdurft

20 Reise in die chinesische Cloud
Jessica Steger

Automatisierung

24 Ansible und Exadata — Eine perfekte Symbiose?
Timo Giese

Datenbank

30 Patching Oracle — Was muss ich tun?
Martin Bracher

36 SQLTXPLAIN — Helfer bei der Performance-Analyse
Stefan Seck

43 Autonomous Database
Sebastian Solbach

47 Oldies but Goldies — Teil 1
Roger Troller

51 Oracle Net 18c — Verschlüsselung ist keine Hexerei
Cornelia Heyde

58 Multitenant — Cloning von Datenbanken
Julian Frey

64 Sicheres Identifizieren von nicht relevanten Indizes
Peter Ramm

Big Data

69 Eintopf und Trennkost — Datenbanken im Ereignishorizont moderner Software-Entwicklung
Daniel Nelle & Gabriel Nelle

Intern

73 Termine

73 neue Mitglieder

74 Impressum

74 Inserenten

Timeline

27. August 2019

Der SOUG Day kann wieder mit hochkarätigen Sprechern aufwarten und bietet rund 80 Teilnehmern ein interessantes und informatives Programm. Vorträge zu APEX und dem SQL Developer, der autonomen Datenbank, dem Thema die «Multi Cloud in Action» über die Zusammenarbeit von Oracle Cloud und Microsoft Azure sowie ein Anwender-Vortrag über die Evaluierung einer Cloud-Infrastruktur stehen auf der Agenda.

Im Track mit dem Focus auf Cloud, Applications und Tools hören die Besucher, was Chatbots heute schon alles können und wie sie die LMS-Cloud nutzen können. Des Weiteren gehen zwei weitere Vorträge auf das Clonen auf einer Exadata und das Automatisieren mit Ansible auf der ODA ein.

2./3. September 2019

Die Go DevOps 2019 feiert ihre gelungene Premiere in Berlin. Der Fokus liegt auf Menschen und Kultur, Prozessen und Praktiken, Tools und Technologien. Die Experten gehen auf die unterschiedlichen Aspekte von DevOps ein und nehmen Stolpersteine sowie Lösungsansätze unter die Lupe.

Die zweitägige Veranstaltung findet mit 55 Teilnehmern im eleganten Ellington Hotel Berlin statt. 30 Vorträge in den drei Streams Tools & Automatisierung, RealLife und Kultur bieten reichhaltige Gelegenheit zu Horizonterweiterung und Networking. Jen Tong, Security Advocate bei Google Cloud, berichtet in ihrer Keynote über Sicherheitsaspekte im Zusammenhang mit DevOps. Die Keynote von Dr. Frank Munz von Amazon Web Services (AWS) fesselt mit dem Thema „Innovation, DevOps und Machine Learning" die Zuhörer. Im Format des Barcamps organisieren die Besucher zwölf Themen, die in angeregten Diskussionen und Gesprächen sowie Unternehmensberichten erörtert werden.

10. September 2019

Die Regionalgruppen Berlin/Brandenburg findet sich zu ihren Regionaltreffen zusammen. In Berlin steht das Thema APEX-SERT - APEX-Security leicht gemacht und Postgres-Datenbanken auf der Agenda.

13. September 2019

Das Webinar zum Thema Monitoring mit Check MK steht auf dem Programm. In seinem Vortrag zeigt Ernst Leber, wie man das System einsetzen kann, um die komplette IT bei Kunden zu überwachen. Am Beispiel eines Datenbank-Servers mit APEX und Tomcat zeigt er in einer Demo die Konfiguration und die Überwachung eines solchen Servers.

16. September 2019

Das Regionaltreffen München/Südbayern in München befasst sich mit dem Thema Cloud und beleuchtet dabei die kaufmännische Seite. Erörtert wird die Frage, welche Kosten auf ein Unternehmen zukommen und wie wirtschaftlich der Einsatz der Cloud ist.

16. bis 19. September 2019

Die Oracle OpenWorld in San Francisco findet statt. Zu den Neuigkeiten gehört beispielsweise die neue Exadata X8M, die Ankündigung einer neuen strategischen Partnerschaft mit VMware, das Angebot zeitlich unbefristeter Always Free Services für Autonomous Database und Cloud Infrastructure und die Tatsache, dass bis zu drei Pluggable Databases ab Oracle 19 kostenfrei in jeder DB-Lizenz enthalten sind.

17. September 2019

Auf der DOAG 2019 Logistik + IT in Frankfurt dreht sich alles um aktuelle Trends bei der Digitalisierung in der Logistik. Im House of Logistics & Mobility (HOLM) hören die rund 40 Besucher spannende Vorträge zu Oracle-Digitalisierungslösungen für Logistik sowie zur digitalen Transformation in der Intralogistik. Prof. Dr. Klaus Mainzer widmet sich in seiner Eröffnungs-Keynote dem Dauerbrenner „Künstliche Intelligenz". Barbara Liebermeister erläutert in ihrer Definition von Digital Leadership, warum es gerade im digitalen Zeitalter kommunikationsstarker und empathischer Führungskräfte bedarf.

17./18. September 2019

Im Berliner Expertenseminar zum Thema „Die Exadata effizient nutzen?“ behandelt Lothar Flatz die wichtigsten Fragen rund um Exadata. Der Exadata-Experte der ersten Stunde erklärt ausführlich das Konzept und die Stärken dieser sehr effizienten Hardware. Wie kann man sie sinnvoll und effizient einsetzen? Warum macht es Spaß, Exadata zu nutzen?

20. September 2019

Das NextGen-Webinar zum Thema Abstimmung von Organisationsform und Systemarchitektur für eine erfolgreiche Produktentwicklung findet statt. Referent Carsten Wiesbaum von esentri macht deutlich, warum die Bereiche der Software- und Produktentwicklung und die Organisationsentwicklung untrennbar miteinander verbunden sein müssen. Er erklärt, warum die meisten technischen Konzepte auch immer einen kulturellen Aspekt beinhalten.

25. September 2019

Der DOAG Data Centric Day findet im Coworking Space „Startplatz“ Köln mit rund 30 Teilnehmern statt. Begriffe wie Big Data, Data Lake, Kryptowährung oder Blockchain sind in aller Munde. In Oracle-Datenbanken schlummern wertvolle Datenbestände. Wie schafft man die Brücke von der traditionellen Oracle-Welt in diese Zukunft? Wie können Sie Oracle in die neuen Themen integrieren? Die Besucher werden in die spannenden Zukunftsthemen der Datenhaltung eingeführt und erhalten wichtige Tipps für Ihren Weg dahin.

27./28. September 2019

Die Vorstandssitzung 3/19 findet in Flonheim-Uffhofen statt. Die Veranstaltungsplanung für das nächste Jahr wird besprochen und festgelegt.

Dr. Dietmar Neugebauer
Ehemaliger DOAG-Vorstandsvorsitzender

Aus der Ferne betrachtet: Die Cloud-Technologie ist nicht mehr aufzuhalten …

Als ich erfahren habe, dass die vorliegende Ausgabe des Red Stack Magazin das Thema Cloud Native zum Schwerpunkt hat, musste ich erst mal Google befragen, um mich schlau zu machen. Hier wurde mir sehr schnell klar, dass die DOAG hier ein Thema anspricht, das sich in letzter Zeit enorm entwickelt hat. Der Cloud-Markt ist schon längst den Kinderschuhen entwachsen und die verschiedenen Hersteller werben mit ihren bereits produktiv im Einsatz befindlichen Applikationen um neue Kunden. Mit Cloud Native können Applikationen, die direkt für die Cloud entwickelt worden sind, auch für diese Architektur optimiert werden. Microservices als Architektur ist hier das Schlagwort für die Verteilung von unabhängigen Applikationen über viele Rechner.

Für die DOAG selbst ist es wichtig, bei diesen überaus schnelllebigen technologischen Entwicklungen auch hier als Wissensvermittler und für den Erfahrungsaustauch zur Verfügung zu stehen. Dies betrifft natürlich die Oracle-Cloud-Produkte, wie sie in der Noon2Noon-Veranstaltung im Januar dieses Jahres in Frankfurt zum Selbstausprobieren den DOAG-Mitgliedern zur Verfügung gestellt wurden. Im Oktober wird zur Oracle-Cloud-Infrastruktur ein zweitägiges Webinar angeboten. Übrigens: Auf der Oracle Groundbreakers EMEA-Tour im Oktober werden auch die Oracle Cloud Native Services präsentiert ….

Doch unsere Mitglieder wollen sich nicht nur über die Oracle Cloud informieren. Anders als im Datenbankumfeld ist im Cloud-Bereich Oracle nicht der Marktführer. Noch dazu ist zu beobachten, dass die Kunden zunehmend verteilte Cloud-Ressourcen verwenden. Dem wurde auf dem DOAG Multicloud Day im Oktober in Frankfurt nachgegangen. Hier präsentierten die Cloud-Anbieter Google und Microsoft zusammen mit Oracle ihre Möglichkeiten. Hinzu kamen noch Erfahrungsberichte bereits umgesetzter Lösungen.

Dies sind alles bereits erste Schritte für die von der DOAG angestrebte und auch von der DOAG- Delegiertenversammlung unter dem Arbeitstitel CloudCon beschlossene Initiative, das Thema Cloud übergreifend mit einer eigenen Konferenz aufzusetzen. In einer solchen Cloud-Konferenz können dann die unterschiedlichsten Aspekte betrachtet werden wie die Berücksichtigung verschiedenster Cloud-Anbieter und -Architekturen verbunden mit den Themen Lizenzierung und rechtlichen Kriterien.

Eine erste Konferenz ist hier im Frühjahr 2020 geplant und …. die DOAG ist gut beraten, dies zusammen mit führenden Cloud-Anbietern, unabhängigen Experten sowie Erfahrungsberichten von Kunden in bewährter Weise als Interessenvertreter ihrer Mitglieder umsetzen …

Quelle: Dr. Carlo Velten

„Das wichtigste Design-Kriterium ist die Hybrid- und Multi-Cloud-Fähigkeit, die durch die Cloud-Native-Technologien unterstützt wird."

Martin Meyer, Redaktionsleiter des Red Stack Magazin, sprach mit Dr. Carlo Velten, CEO des IT-Research- und Beratungsunternehmens Crisp Research AG in Kassel über Standards, Trends und die Entwicklung von Cloud-Native-Technologien sowie Voraussetzungen und Anforderungen für eine erfolgreiche Einführung in Unternehmen.

Was bedeutet Cloud Native?

Cloud Native ist einfach das Synonym für eine Vielzahl von neuen Technologien, die im weitesten Sinne auf Microservice-Architekturen und für den Betrieb in der Public Cloud ausgelegt sind. Die sich sozusagen darauf stützen, dass Software, Prozesse, Applikationen und Daten nicht mehr in Bare Metal oder VMware-basierten Umgebungen, sondern in Containern vorgehalten werden. Dies war zumindest die Ursprungsidee und deswegen ist die Kerntechnologie eines der wesentlichen Elemente in diesem Ökosystem der Cloud-Native-Technologie die Container-Technologie rund um das Thema Docker und auch Kubernetes.

Welche Standards gibt es bei der Orchestrierung und für das Management von Container-Clustern?

Mittlerweile gibt es eine ganze Reihe von Technologien und Tools. Das Wesentliche ist das Ökosystem um die Kubernetes-Familie, also das Open-Source-Management-Framework zur Orchestrierung und Verwaltung von größeren Container-Clustern und Container-Landschaften.

Welche Trends gibt es?

Ein Großteil der Projekte und Technologien sind Technologien, die man rund um das Thema Management der Container braucht *[1]*. Also Kubernetes selbst, Grafana, um Datenvisualisierung und Monitoring von Containern zu ermöglichen, Prometheus für Monitoring und Alerting, dann gibt es Helm als Applikation und Package Manager sowie ein Stück weit auch Istio. Neben diesen rein auf Kubernetes und Container fokussierten Technologien muss man natürlich auch sagen, dass es auch für Unternehmen und Entwickler relevante Cloud-Native-Technologien gibt, die jetzt keinen direkten Container-Bezug haben, sondern die sich entweder auf das Thema Analytics beziehen oder auf das Thema Datenbanken.

Einer der drei strategischen Trends, die aus unserer Perspektive den Einsatz von Container- und Cloud-Native-Technologien vornehmlich in den großen Unternehmen in den letzten Jahren treiben – und aktuell meiner Meinung nach in den letzten 12 Monaten verstärkt auch in der Mittelstands-IT –, ist ganz klar das Thema Digitalisierung. Die Digitalisierungs-Units, die neue digitale Produkte und Lösungen, Apps und Interfaces bauen müssen, die vielleicht im Backend zu einer Oracle-Datenbank oder zu einem Oracle-ERP-System konnektiert werden. Das ist ein ganz wichtiger Impuls und ein zentraler Trend. Dann gibt es natürlich auch die generelle Bewegung in Richtung Cloud. Man überlegt, welche Workloads in der Cloud laufen können und wie sie in der Cloud laufen. Geschieht dies noch in einer virtuellen Maschine oder im Container? Letztendlich, so glaube ich, ist dann auch für sehr viele Entwickler im Oracle-Kontext die Frage der Anwendungsmodernisierung relevant. Welche Anwendungsbestandteile werden weiterhin mit den etablierten Technologien in Entwicklungssprachen, Frameworks und Datenbanken neu programmiert und wie werden diese ergänzt durch zusätzliche Module und Services, die dann eben in den Cloud-Native-Technologien geschrieben sind. Aus unserer Perspektive sind die Digitalisierung, die Cloud-Transformation und die Anwendungsmodernisierung die drei wichtigen Dinge, die Cloud-Native-Technologien vorantreiben.

Weg vom Silo-Prinzip hin zu Cloud-Native. Was müssen Unternehmen aus kultureller und technischer Sicht ändern?

Kulturell müssen Unternehmen zunächst einmal in der Form etwas ändern, sodass das Thema Software-Entwicklung wieder eine höhere Priorität bekommt; sie sollten dazu bereit sein, intern oder extern zu investieren. Im Zuge der klassischen IT-Standardisierungs- und Outsourcing-Denkweise in den letzten ein bis fünfzehn Jahren ist die Individual-Softwareentwicklung in gewisser Weise eine Randposition in der IT geworden. Viele Unternehmen haben gar keine eigene Application-Development- und Management-Kompetenz mehr, stattdessen hat sich diese zunehmend hin zu externen Dienstleistern verlagert. Im Kontext der Digitalisierung, der Innovationsvorhaben und durch den Wunsch, agiler und schneller neue Technologien einsetzen und neue Projekte umsetzen zu können, kommt das Thema Cloud Native in die Unternehmen. Dort stellt man dann fest, dass es an den jeweiligen Development- und Technologie-Skills mangelt und dass man die erst einmal ausbauen muss.

Zunächst muss man sich bewusst machen, dass Software-Entwicklungskompetenzen und -Skills in diesen Technologien gefragt sind. Wir sehen, dass bei einer ganzen Reihe von Großunternehmen, wie beispielsweise Daimler, BMW oder Osram, zumindest ein Insourcing in den Innovationsthemen in der IT stattfindet.

Auch organisatorisch müssen sich Dinge ändern, denn Cloud-Native-Technologien werden fast immer im Kontext von agilen Organisationsmethoden eingesetzt. Wenn man in Microservices entwickeln möchte, dann macht man dies meistens nach agilen Prinzipien, also mit Scrum, Insprints, schnellem Prototyping, hochautomatisierter Verbindung von Development-Tests bis hin zu den Deployment-Mechanismen. Das bedeutet natürlich, dass, wenn man das intern macht, die internen Entwicklungstest- und Deployment-Prozesse über Anwendungs- und Abteilungsgrenzen hinweg neu strukturiert werden. Die Teams und deren Verantwortlichkeiten werden neu zugeschnitten. Den Entwicklern gibt man mehr Entscheidungs- und Spielräume, die Infrastruktur auch im Self-Service-Modus zu nutzen, sodass diese nicht bei der IT fragen müssen, ob man entsprechende Server-Kapazitäten, entsprechende Images, entsprechende Datenbanken überhaupt nutzen darf und wie lange das dauert.

Wir kommen in einen Modus, in dem wir in diesem DevOps-Kontext natürlich Infrastruktur-as-a-Service (IaaS) konsumieren wollen und all die Anforderungen an die Infrastruktur gestellt werden, wie sie auch an die Cloud gestellt werden. Das heißt on demand, skalierungsfähig, breites Portfolio an Infrastruktur- und Plattformdiensten, die dann orchestrierbar und kombinierbar sind bis hin zu einem hochautomatisierten und stark integrierten Development- und Operationsprozess.

Man kann den Wert und den Vorteil von Cloud-Native-Technologien von großen Container-Clustern, von Microservice-Architekturen eigentlich nur dann ausschöpfen, wenn der Betriebsprozess und der DevOps-Prozess ganzheitlich aufgesetzt und durchgeführt werden. Es bringt nichts, die Dinge nur im Development- und Testmodus zu betrachten und nachher keinen professionellen Betrieb dafür zu haben. Denn es kann erhebliche Probleme geben aufgrund der Komplexität, aufgrund der Neuheit vieler Technologien und auch dadurch, dass einige dieser Technologien noch unausgereift sind.

Wie erkennt man die besten Kandidaten für den Umzug in die Cloud?

Kulturelle und organisatorische Grundsätze sollten stets zusammen betrachtet werden. Ein Unternehmen muss bereit sein, seine Organisation auch so zu verändern, dass der kulturelle Wandel sich einstellen kann. Die technische Grundvoraussetzung liegt darin, dass entweder die eigene interne Infrastruktur, also die virtualisierte Infrastruktur beziehungsweise die On-Premises oder Private Cloud, egal wie man das jetzt nennt, die Charakteristika einer Cloud erfüllt. Die kann man auch intern selbst bauen und viele Großunternehmen machen das auch. Gewährleistet sein müssen on demand, Skalierbarkeit, Pay-as-you-go, API-Driven, Self Service, Infrastruktur, Datenbanken- und Monitoring-Services, die man in API konsumiert und die dann entsprechend berechnet werden. Ohne das ergibt der ganze Cloud-Native- und DevOps-Teil überhaupt keinen Sinn und last but not least muss diese Infrastruktur natürlich in der Lage sein, Container-Cluster auch zu

managen und zu betreiben. Dazu gibt es natürlich sehr viele Überlegungen, denn man kann heutzutage Kubernetes auf Bare Metal oder Kubernetes-Cluster und -Container auf der Basis von virtuellen Maschinen betreiben.

Das erste Kriterium ist, danach zu fragen, ob es sich um eine bestehende oder um eine neue Applikation handelt. Bei einem guten Kandidaten geht es um eine neue Anwendung, die für ein digitales Produkt, für eine Applikation nach DevOps-Kriterien und mit Cloud-Native-Ansätzen und -Technologien entwickelt wird, die auch im Browser laufen soll und nicht auf einer Maschine in der Produktion.

Tendenziell kann man sagen, dass neue digitale Applikationen, die auf dem Browser ausgeliefert werden, in Microservice-Architekturen geschrieben und in Container gepackt werden. Es ist immer schon eine gute Ausrichtung, alles Neue auch mit den State-of-the-art-Technologien nach Cloud-Native-Art zu machen.

Bei den Bestandsapplikationen sieht das natürlich anders aus. Hier würden wir folgende Kriterien anlegen, wenn es darum geht, eine Anwendung zu modernisieren. Meistens lässt sich eine Anwendung, die heute On-Premises oder noch in virtuellen Maschinen läuft, nicht so einfach in Container packen, da sie sich hinsichtlich Lastverhalten und Performance auch ganz anders verhält. Um eine Anwendung Cloud-Native-ready zu machen, gibt es einen entsprechenden Modernisierungsaufwand, der meistens recht hoch ist. Müssen Teile der Applikation oder die komplette Applikation umgebaut werden, dann geht es natürlich auch immer um viel Geld- und Ressourcen-Aufwand. Dies lässt einen wiederum schlussfolgern, dass es meistens nur für Applikationen und Geschäftsprozesse sinnvoll ist, die einen extrem strategischen Stellenwert haben und eine hohe Kritikalität oder einen hohen Business Value. Also Applikationen, von denen man beispielsweise weiß, dass sie auch in 2 bis 5 Jahren einen wesentlichen Wert haben, Innovationstreiber im Unternehmen sind und zur Kernkompetenz des Unternehmens gehören.

Als Letztes würde ich sagen, dass sich grundsätzlich Anwendungen eignen und rentieren, die ohnehin auf Distributed Computing ausgelegt sind. Also Anwendungen, die nicht an einer Stelle zentral betrieben werden, sondern vielleicht Anwendungen, die an vielen Stellen in der Welt betrieben werden und die häufig sehr nah am Kunden sind oder die man über Content Delivery Networks oder Edge Computing sehr nahe zum Kunden bringen muss.

Nicht rentiert es sich bei Legacy-Applikationen mit einer extrem hohen Integrationstiefe, bei denen es eine Notwendigkeit gibt, auf bestimmten Datenbank-Technologien oder Application-Server-Technologien zu bleiben. In der Produktionsplanung oder in der Produktionsüberwachung, in einer Fabrik, wo es keine Alternativen gibt, oder bei Applikationen, bei denen man Datenbanken nicht verteilen kann – Stichwort Oracle –, sondern bei denen man hohe Transaktionssicherheit braucht und eine zentrale Datenbank benötigt. Das sind dann Kandidaten, denen man sagt, dass der Mehrwert bei einer Innovation für eine Anwendungsmodernisierung mit Cloud-Native-Technologien nicht so hoch ist.

Welche Vorteile bringt die Cloud gegenüber On-Premises?

Der wirkliche Vorteil – also ein zentraler Benefit – der Cloud-Native-Technologien und speziell von Kubernetes ist die Portabilität sozusagen von der internen IT über Hosting-Provider bis hin in die Public Cloud. Kubernetes hat es geschafft, das Versprechen einzulösen, das mal mit dem Open-Source-Tool Open Sec verbunden war. Das native, echte Kubernetes und Container werden mittlerweile von allen Cloud-Providern unterstützt. Vor zwei Jahren haben sich noch fast alle Cloud-Provider einen eigenen Container-Service gebaut, die untereinander nicht unbedingt kompatibel waren. Man konnte Container zwischen den Clouds nicht umziehen und keine Multi-Cloud-Strategie fahren. Mittlerweile hat sich das geändert; Microsoft, AWS, Google, Oracle meines Erachtens auch, unterstützen alle das Original-Kubernetes und die Kubernetes-APIs.

Dies bedeutet, dass, wenn ein Unternehmen im eigenen Rechenzentrum sich ein Cluster aufsetzt, auf dem es ein Kubernetes betreibt, es in der Lage ist, die Anwendungsbestandteile, die in den Containern sind, wirklich auf Knopfdruck aus dem eigenen Data Center in die eine und die nächste Cloud zu schieben und wieder zurückzuziehen. Das bringt eine extrem große Freiheit und bedeutet, dass die Cloud-Native-Technologien – Kubernetes im Speziellen – es den IT-Entscheidern ermöglichen, endlich diese Flexibilität und Portabilität auf der Technologie-Ebene hinzubekommen und sich ein Stück weit vom Vendor-Lock in den großen Clouds freizumachen. Es stellt sich daher die Frage, welche Anwendungen man noch in virtuellen Maschinen hält und welche Anwendungen man in Container packt, um diese Freiheit und Portabilität dann auch wirklich nutzen und genießen zu können.

Wie sehen die modernen Cloud-Architekturen aus?

Das wichtigste Design-Kriterium ist die Hybrid- und Multi-Cloud-Fähigkeit, die durch die Cloud-Native-Technologien unterstützt wird. Eine moderne Cloud-Architektur muss in der Lage sein, Workloads und Daten in verschiedenen Clouds laufen und halten lassen zu können. Bei Containern ist es einfacher als bei virtuellen Maschinen, aber auch da gibt es natürlich den Trend zu einer stärkeren Kooperation zwischen den verschiedenen Cloud- und Technologie-Providern. So hat etwa VMware eine große Kooperation mit AWS geschlossen, sodass man VMware-basierte Systeme auch in der AWS-Cloud betreiben kann. Google hat eine Version von Kubernetes zur Verfügung gestellt, die Multi- und Hybrid-Cloud-Fähigkeit unterstützt.

Eine moderne Cloud-Architektur ist durch einen extrem hohen Automatisierungsgrad im kompletten Tooling von der Provisionierung über die Entwicklung bis über die Skalierung gekennzeichnet: hoher Automatisierungsgrad vom Self-Service-Prozess der Entwickler über die Orchestrierung bis hin zur Konfiguration und zum Deployment. Moderne Cloud-Architekturen zeichnen sich zudem durch die Programmierbarkeit aus. Bei API first bedeutet dies, dass unabhängig davon, ob etwas intern oder extern läuft, man dafür Sorge tragen sollte, dass die verschiedenen Infrastrukturen und Plattformdienste wirklich sauber über APIs ansprechbar und konsumierbar sind. Sie beinhalten zu einem großen Teil die Unterstützung für Container- und Cloud-Native-Technologien.

Eine wirklich moderne Cloud-Architektur ermöglicht es auch, nicht nur hybrid zwischen On-Prem-eigenem Rechenzentrum und der Cloud zu wechseln, sondern auch On-Premises virtuelle Maschinen und Container immer intelligent zu mixen. Das

bedeutet, dass man eine Unterstützung für alle drei Formate benötigt. Wenn man zum Beispiel einfach nur Rechenpower braucht, will man vielleicht Sachen auf einem Bare-Metal-Server deployen können, vielleicht werden für eine hohe Standardisierung klassische virtuelle Maschinen benötigt, vielleicht will man die Anwendungen auch in Container packen, vielleicht will man auch ein Serverless-Framework unterstützen können. Diese vier Dinge sollten von einer modernen Cloud-Architektur unterstützt werden.

Wie migriert man seine IT in die Cloud?

Ich würde immer empfehlen, nicht die komplette IT in die Cloud zu migrieren, sondern natürlich am Anfang die Why-Fragen zu stellen. Warum gehe ich in die Cloud? Was soll in die Cloud gepackt werden? Im Kontext Cloud Native ist immer die Frage, ob man über eine Migration von Anwendungen auf ein Cloud Native Stack spricht oder davon, Applikationen und Infrastruktur in die Public Cloud zu migrieren. Das kann das Gleiche sein, aber es gibt natürlich eine Differenzierung. Häufig wird davon gesprochen, in die Cloud zu migrieren. Bedeutet das, dass man Anwendungen in die Rechenzentren der großen Public Cloud-Provider migriert, wo man dann entweder bestimmte Plattformdienste konsumiert oder wo man Anwendungen in virtuellen Maschinen betreibt – klassisches Lift and Shift – oder wo man im Rahmen von Cloud-Native-Technologien und DevOps ansetzt, neue digitale Anwendungen zu betreiben.

Man muss zunächst den Bedarf klären – Cloud First ist nicht immer das Richtige für alle Anwendungen. Stattdessen muss man anhand bestimmter Kriterien überlegen, welchen Teil seiner Anwendungslandschaft, Infrastruktur und Dienste man zukünftig in der Public Cloud betreiben möchte. Dazu macht man sich eine Roadmap mit den zu priorisierenden Applikationen und identifiziert dann bei diesen Applikationen und Geschäftsprozessen das Migrations- und Modernisierungspotenzial. Anschließend überlegt man sich, ob die Anwendungen entweder mit einem einfachen Lift and Shift von einer internen virtuellen Maschine auf eine virtuelle Maschine in der Public Cloud geschoben werden oder ob man sozusagen diesen Modernisierungspfad einschlägt und die Anwendung und die Anwendungsarchitektur umbaut, neu konzipiert und die Software dann sozusagen in einem Microservice-Ansatz neu programmiert und in Containern betreibt.

Last but not least bei neu zu entwickelnden Anwendungen kann man sich auch Dienste aus der Cloud zusammenstöpseln und in seinen Software-Code einfügen, dann funktioniert dies auch direkt in der Cloud. Bei Bestands-Applikationen ist dann die Frage, ob Lift and Shift oder Anwendungsmodernisierung angestrebt wird. Dies muss man dann applikationsspezifisch betrachten und natürlich die Kosten im Blick haben. Man überlegt, welche Kosten mit einem Umzug in die Cloud verbunden sind, und stellt eine realistische Betrachtung an, ob man durch eine Migration über einen günstigeren und automatisierten Betrieb in der Public Cloud später wieder Kosten einsparen kann oder ob es nachher nur bedeutend teurer wird, was in vielen Fällen auch der Fall ist. Wenn es allerdings eine Management-Entscheidung ist, strategisch auf einen Hyper Scaler zu gehen, dann ist auch das durchaus eine Möglichkeit.

Welche Mitarbeiter-Skills braucht man für den Betrieb in der Cloud?

Da sind natürlich erst einmal die wirklichen Cloud-Architekten und die Leute, die Erfahrung mit der Orchestrierung von Diensten in der Public Cloud haben. Dann braucht man erfahrene Cloud-Security-Architekten, denn man kann beim Setup natürlich sehr viel falsch machen. Wenn man dann Dienste gerade in der Public Cloud neu zusammenstellt, kann auch sehr viel falsch laufen, wenn man Applikationen in Container packt und auf Kubernetes-Clustern betreibt. Wie gesagt, die Technologien sind relativ neu, es ist recht komplex. Da gibt es viele Fallstricke, da man aus einer monolithischen und eher trägen, aber sicheren Applikationswelt herauskommt in eine potenziell agile Applikationswelt mit sehr schnellen und kurzen Veränderungszyklen, die jedoch hochgradig unsicher sind.

Man muss aufpassen und gestatten Sie mir dazu meine Anmerkung: Viele Unternehmen haben ihr Risikoprofil in den letzten Jahren deutlich zum Schlechteren sich entwickeln lassen, weil durch die Agilitätsanforderung in Software-Entwicklung und -Betrieb sowie im Kontext vielfältiger Digitalisierungsinitiativen Geschwindigkeit immer Vorrang hatte vor Sicherheit. Ein Großteil der deutschen Firmen läuft daher heute mit einem deutlich kritischeren Risikoprofil durch die Welt.

Dazu kommt, dass viele digitale Anwendungen im Rahmen dieser hybriden Architekturen mit den Backend-Systemen verzahnt sind. Wenn man heute ein digitales Frontend hat, das auf eine zentrale Oracle-Produkt- oder -Kundendatenbank zugreift,

Zur Person: **Carlo Velten**

Dr. Carlo Velten ist CEO des IT-Research- und Beratungsunternehmens Crisp Research AG. Seit über 15 Jahren berät Carlo Velten als IT-Analyst namhafte Technologieunternehmen in Marketing- und Strategiefragen. Seine Schwerpunktthemen sind Cloud Strategy & Economics, Data Center Innovation und Digital Business Transformation. Zuvor leitete er 8 Jahre lang gemeinsam mit Steve Janata bei der Experton Group die „Cloud Computing & Innovation Practice“ und war Initiator des „Cloud Vendor Benchmark“. Dr. Carlo Velten ist Jurymitglied bei den „Best-in-Cloud-Awards“ und engagiert sich im Branchenverband BITKOM.

dann hat man ein Problem, wenn es gehackt wird. Daher sind ein Cloud-Solution-Architekt und ein Security-Architekt notwendig, um überhaupt die Grundlagen für einen Betrieb zu schaffen. Zusätzlich eine Handvoll erfahrener DevOps und Cyber Liability Engineers, die dann ein entsprechendes Betriebskonzept für den Betrieb in einer Public oder Hybrid Cloud aufsetzen.

Das berücksichtigt, dass vieler dieser neuen Systeme, dieser Cloud-Native-basierten und Cloud-basierten Anwendungen, also entweder neue Dinge oder strategisch wichtige Applikationen, im globalen Geschäftsprozess praktisch immer im 24/7-Betrieb laufen. Das bedeutet in vielen Fällen, dass uns die Wartungsfenster verloren gehen.

Deshalb muss man DevOps- und Betriebsteams aufsetzen, die auch zusammen mit den Entwicklungsteams in der Lage sind, Updates und das Einspielen neuer Versionen, das Patching von wirklich komplexen Cluster-Umgebungen im Live-Betrieb durchzuführen. Dafür braucht man sehr erfahrene Leute und ein sehr gutes Zusammenspiel der Software-Entwickler und der Betriebsteams. Echte DevOps-Leute und echte Cyber Liability Engineers, die 3 bis 5 Jahre echte Betriebserfahrung sowohl mit neuen Cloud-Native-Technologien als auch mit den Cloud-Plattformen der großen Hersteller wie Oracle, AWS etc. mitbringen, sind im deutschsprachigen Raum recht selten.

Welche Cloud Services werden genutzt – sowohl aus Orchestrierungssicht als auch aus Cloud-Sicht? Wie trifft man die Entscheidung, gerade diese Cloud, diesen Service zu nutzen?

In den meisten Unternehmen stellt sich die Provider-Frage zuerst, da die Freigabe, in der Public Cloud arbeiten und ein System produktiv nehmen zu dürfen, mit sehr vielen Rechten sowie Compliance- und Government-Fragen behaftet ist. Die meisten Unternehmen haben begonnen, in den Digitalisierungs-, Innovations- und F&E-Abteilungen mit einem der großen Cloud-Provider zu arbeiten, haben danach eine Freigabe durch die Corporate IT erwirkt und sich ihre Aktivitäten in der Public Cloud nachträglich rechtfertigen lassen.

In den Großunternehmen gibt es aufgrund der Historie und der Tatsache, dass AWS die erste große globale Public Cloud war, immer noch eine große Verbreitung von Amazon Web Services. AWS ist noch Marktführer und in den Start-ups und den Digitalisierungsabteilungen sehr stark vertreten. Im Kontext der Diskussion um Multi-Provider-Strategien und Überlegungen, dass man sich nicht von einem Cloud-Provider abhängig machen möchte, haben viele Unternehmen im Mittelstand und in den Großunternehmen natürlich in den letzten 24 Monaten darüber nachgedacht, nicht nur mit AWS zu arbeiten. Sie machen stattdessen so eine Art Second Sourcing und arbeiten auch auf einer Azure-Cloud, teilweise auf einer Oracle- und auf einer SAP-Cloud, wenn es um eine Business-Applikation geht, um bestimmte Synergie-Potenziale auszuschöpfen, und natürlich auch, weil sie bestimmte Enterprise-Agreements einfach haben. Es gibt die Top 4 der klassischen Public-Cloud-Provider, die in vielen Unternehmen genutzt werden. In der Mehrheit der Unternehmen liegt eine sogenannte Dual-Provider-Strategie vor, die die Budgets mehrheitlich an AWS und Azure verteilen und dann in kleinerem Stil, opportunistisch und projektbezogen auch andere Clouds nutzten. Im Rahmen der Multi-Cloud-Strategien wird sich das Bild in der Form ändern, dass man vielleicht mit einer Handvoll Cloud-Providern weltweit arbeitet, wenn man ein sehr großes Unternehmen ist. Mittelständische Unternehmen werden sich in Zukunft im IaaS- und PaaS-Bereich auf zwei bis drei Provider fokussieren, und der Einsatz als SaaS ist natürlich viel bunter und vielfältiger. Bei der Frage , welche konkreten Dienste auf diesen Plattformen genutzt werden, lässt sich aufgrund der Tatsache, dass vor allem viele Unternehmen im deutschsprachigen Raum noch nicht so viel Erfahrung mit den großen Cloud-Plattformen haben, feststellen, dass die Basisfunktionalitäten sowie die Basis-Infrastruktur- und Plattformdienste derzeit im Vordergrund stehen *(zu Studienergebnissen siehe Quellen)*. Ich glaube, dass diese einen Großteil der Umsätze in der Public Cloud bei AWS, Azure und auch bei Oracle und anderen ausmachen. Das bedeutet virtuelle Maschinen und Infrastruktur, Storage, Network und Bandbreite und danach geht es quasi los mit Datenbanken aus der Cloud und DevOps-Tools aus der Cloud. Unternehmen, die auf ihrer Digitalisierungsreise schon etwas weiter sind, fangen dann auch an, Analytics Services aus der Cloud zu nutzen, um beispielsweise IoT-Sensor-Datenverarbeitung mit Cloud-Diensten zu machen.

Quellen

[1] *https://www.crisp-research.com/kubernetes-und-cloud-native-trends-2019-das-jahr-de*

Weitere Informationen

1. *https://www.crisp-research.com/top-10-cloud-native-technologien-fuer-den-unternehmenseinsatz/*
2. *https://www.crisp-research.com/publication/kubernetes-cloud-native-trends-2019/*

Auf den Spuren der nächsten IT-Evolutionsstufe: Entwicklung und Grundlagen von Cloud Native

Benjamin Nothdurft, codecentric

In diesem Artikel gehen wir auf die Entwicklung und Grundlagen von ein. Begonnen wird die Spurensuche mit einer kleinen Zeitreise durch Forschung und Industrie, wobei hier auch die wirtschaftlichen Treiber identifiziert werden. Neben wichtigen Institutionalisierungen, die sich mittlerweile gebildet haben, wie etwa der Cloud Native Foundation (CNCF), wird auch auf allgemeine Grundlagen sowie die Vielfalt und den Reifegrad von aktuellen Tools, Frameworks und Cloud Services eingegangen. Abrundend werden Hilfestellungen und Wegweiser aufgezeigt, die bei der schrittweisen Einführung und Entwicklung von Cloud Native Applications (CNA) im eigenen Unternehmen unterstützen können. Sukzessive wird somit die Bedeutung von CNAs herausgearbeitet, die längst kein Trend mehr sind, sondern als nächste IT-Evolutionsstufe angesehen werden können, die dabei helfen kann, einen nachhaltigen Unternehmenserfolg weiterhin zu ermöglichen.

Heutzutage ist es einleuchtend, dass immer mehr neue Cloud Services unter dem Label „cloud native" gestartet werden, jedoch hat sich der Term seit seinem ersten Auftreten über die Jahre hinweg zunächst in der Verbreitung zurückentwickelt, bevor er sich sowohl in der Wissenschaft als auch in der Industrie durchsetzen konnte. Begeben wir uns auf eine kleine Zeitreise.

Der Begriff „cloud native" trat erstmals mit der Geburt von Cloud Computing im Jahr 2006 auf. Am 13. März jenes Jahres ging Amazon mit dem Speicherdienst S3 (Simple Storage Service) unter der neuen Flagge AWS (Amazon Web Services) als erster Public Cloud Provider an den Markt.

In den Folgejahren kamen mehr und mehr Cloud Provider und Services hinzu und der Gebrauch vieler weiterer Begriffe im Cloud-Computing-Umfeld begann sich zu etablieren. Bezeichnungen wie Public/Private/Hybrid Cloud Computing und Akronyme wie IaaS (Infrastructure as a Service), PaaS (Platform as a Service) oder SaaS (Software as a Service) wurden im Alltag häufig benutzt, unterlagen hierbei jedoch noch oft einem unterschiedlichen Verständnis.

Im September 2011 veröffentlichten schließlich Peter Mell und Timothy Grace im Paper „NIST Definition of Cloud Computing" ein Glossar mit Definitions-Empfehlungen des National Institute of Standards and Technology. Obwohl bereits bekannt, wurde auf die Bezeichnung „cloud native" beziehungsweise „Cloud Native Applications" (CNA) hierbei noch nicht eingegangen *[1]*.

Erst im Jahr 2012 stieg die Popularität von CNA und damit begann dessen häufigere Verwendung zunächst vor allem im akademischen Sektor bei der Ausarbeitung von wissenschaftlichen Abhandlungen. In diesem Jahr wurde auch der erste Entwurf zu Cloud Native Engineering auf einer akademischen Konferenz vorgestellt, der eine musterbasierte Lösung für Methodologien des Cloud Native Application Design verfolgt, die auf die systematische Identifikation von Konsistenzen, Verfügbarkeiten und Ausfalltoleranzen nach dem CAP-Theorem setzt.

Ab dem Jahr 2015 gewann die Bezeichnung „cloud native" schließlich auch schlagartig in der Industrie an Traktion, wie eine Analyse der Suchtrends bei Google beweist. Typischerweise werden hier „Cloud Native Applications" als logische Weiterentwicklung von Microservices-Architekturen und Container-basierten Ansätzen betrachtet. Später mehr zum Industriesektor; lassen Sie uns zunächst im akademischen Umfeld bleiben.

Cloud Native in der Forschung

In verschiedenen wissenschaftlichen Publikationen wurden über die Jahre immer wieder hier und da Eigenschaften definiert, die Anwendungen zu erfüllen haben, bevor sie sich mit „cloud native" bezeichnen dürfen – ganz im Gegensatz zu klassischen verteilten Anwendungen.

Eine kumulative Betrachtung im akademischen Sektor erfolgte in den Jahren 2016 und 2017 in der Abhandlung „**Understanding Cloud-native Applications after 10 years of Cloud Computing: A systematic mapping study**" durch Nane Kratz und Peter-Christian Quintis von der Cloud Computing Research Group an der Universität Lübeck *[2]*.

Nach einer ausführlichen Analyse und basierend auf bisherigen Forschungsansätzen unterbreiten Kratz und Quintis schlussendlich eine vereinheitlichende Definition für „Cloud Native Applications":

> **A cloud-native application (CNA) is a distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of stateful components. The application and each self-contained deployment unit of that application is designed according to cloud-focused design patterns and operated on a self-service elastic platform.**

Diese fußt unter anderem auf den 2014 von Fehling proklamierten „IDEAL"-Eigenschaften . CNAs sollen demnach IDEAL sein. Dies bedeutet

- die Anwendungen sollen einen isolierten Zustand besitzen (i = isolated state),
- generell verteilt sein (d = distributed in its nature),
- elastisch in der horizontalen Skalierung sein (e = elastic),
- von automatisierten Managementsystemen betrieben werden (a = automated management)
- und ihre Komponenten sollen dabei lose gekoppelt sein (l = loosely coupled).

Zudem berücksichtigen Kratz und Quintis die von Stine 2015 aufgestellten Thesen zur allgemeinen Motivation für den Einsatz von CNA-Architekturen:

- schnellere Auslieferung von softwarebasierten Lösungen (speed)
- höhere Fehlerisolation und Fehlertoleranz mit automatischer Regeneration (safety)
- Ermöglichung von horizontaler statt vertikaler Skalierung (scale)
- Bedienung einer großen Vielfalt von (mobilen) Plattformen und von Legacy-Systemen (client diversity)

Im Jahr 2016 folgerte Balalie zudem, dass aus diesen Motivationsthesen folgende verschiedene Applikations- und Infrastrukturansätze abgeleitet werden können:

- *Microservices:* als Dekomposition von monolithischen Geschäftsanwendungen in unabhängig auslieferbare Services unter dem Motto „doing one thing well".
- *Published and Versioned APIs:* als Standardverfahren für die Interaktion zwischen Services in einer CNA. Häufig basierend auf HTTP, REST und JSON.
- *Collection of Cloud-Focused Patterns:* als anzuwendende Design- und Vernetzungsprinzipien für die unabhängig auslieferbaren Einheiten. Beispiele, die in den Jahren 2014 und 2015 veröffentlicht wurden, sind „The Twelve-Factor App" von den Entwicklern von Heroku, das „Circuit Breaker Pattern" nach Fowler oder die „Cloud Computing Patterns" nach Fehling und Erlet.
- *Self-Service Agile Infrastructure Platforms:* als Auslieferungs- und Betriebswerkzeug der in sich abgeschlossenen, Container-basierten Einheiten. Diese Plattformen sollen zudem zusätzliche Funktionalitäten auf Basis von IaaS anbieten wie eine Request-basierte, automatisierte Skalierung einzelner CNAs, Applikationsmonitoring, dynamisches Routing, Load Balancing sowie Aggregation von Logs und Metriken.

Im oben beschriebenen akademischen Definitionssatz für CNA greifen Kratz und Quintis neben diesen Studien auch auf weitere bereits existente Definitionen zurück, um ihre Definition zu untermauern. Diese implizit vorausgesetzten und verwendeten Begriffe sind:

- *Elasticity:* beschreibt den Grad, zu dem ein System fähig ist, sich an Workload-Änderungen durch die (De-)Provisionierung von entsprechenden Ressourcen automatisch anzupassen, sodass zu jedem Zeitpunkt die Verfügbarkeit dem aktuellen Bedarf so gut wie möglich entspricht.
- *Scalability:* wird differenziert in strukturelle und Last-basierte Skalierbarkeit. Strukturelle Skalierbarkeit bedeutet, sich generell an die gewünschte Dimension anpassen zu können. Last-basierte Skalierbarkeit heißt, sich nach und nach der sich ändernden Dimension anzupassen.
- *Microservices:* werden als der zu entwickelnde Architekturstil, wie er in der erweiterten Definition von Fowler proklamiert wird, verstanden. Unter anderem kommt damit hinzu, dass sich die Modellierung der Services an den Geschäftseigenschaften ausrichten soll. Dies impliziert die Verwendung von Domain-Driven Design (im Gegensatz zu technisch-getriebenen Modellierungsansätzen).
- *Self-contained Deployment Unit:* ist die technische Umsetzung einer Einheit innerhalb der Applikationsauslieferungstopografie. Mehr und mehr spielen hier standardisierte Container eine Rolle, wie sie zum Beispiel von der Open Container Initiative (OCI) in den „5 Principles for Containers" definiert sind.
- *Stateful Components:* werden genutzt, um mehrere Instanzen einer skalierten Anwendungskomponente zu synchronisieren, um so ein einheitliches Verhalten bereitzustellen.
- *Elastic Platform:* ist eine integrierte Verteilungsplattform zur Ausführung der spezifischen Anwendungen, inklusive deren Kommunikation und Datenspeicherung, die deren Dienste über ein eigenständiges Interface im Netzwerk bereitstellt. Beispiele hierfür sind Kubernetes, Docker Swarm, Mesos etc.

Im akademischen Umfeld wurden zudem auch immer wieder ergänzende Betrachtungen zu Trends im Cloud Native Engineering vorgenommen. So wurden vor allem im Zusammenhang von CNA (und Microservices) auch folgende Themen behandelt und Trends identifiziert:

- *DevOps:* Das Bauen, Testen und Veröffentlichen von Software soll durch Automatisierung schneller, häufiger und zuverlässiger erfolgen. Entwickler und Operations sollen hierbei enger zusammenarbeiten. Stichwort: Cross-funktionale Teams.
- *Softwarization Shift:* beschreibt die voranschreitende Automatisierung im Infrastruktur- und Netzwerkbereich durch Softwarelösungen.
- *State Isolation:* Stateless-Komponenten sind leichter zu skalieren, daher sollten Stateful-Komponenten auf ein Minimum reduziert und bevorzugt mithilfe von skalierbaren Datenspeichern wie (No) SQL-Datenbanken abgebildet werden.
- *Loose Coupling:* Diese Art der Servicekomposition kann durch Events oder Daten abgebildet werden. Event Coupling wird hierbei durch Messaging-Lösungen wie den AMQP-Standard realisiert. Data Coupling hingegen wird häufig durch skalierbare und hochperformante Speicherlösungen realisiert, die häufig (nur) dem Konzept der Eventual Consistency folgen. Beispiele sind NoSQL-Datenbanken.

Betrachtet man also den wissenschaftlichen Stand zu CNA, stellt man fest, dass viele Ideen, die wir auch in der industriellen Softwareentwicklung wiederfinden, bereits in der modernen Forschung verankert sind. Nichtsdestotrotz spielt die Industrie in der Entwicklung von Cloud Native seit Langem eine große Rolle, wenn nicht sogar die führende.

Cloud Native in der Industrie

Um verstehen zu können, warum Cloud Native in der Industrie auch in Zukunft an Bedeutung gewinnen wird und warum Ansätze im Umfeld von CNA schon heute eine zentrale Rolle in modernen Unternehmen einnehmen beziehungsweise zunehmend einnehmen werden, wenden wir uns zunächst den Hintergründen in der Entwicklung der freien Märkte zu.

Im nachfolgenden Abschnitt gilt es also, die Treiber in der Marktwirtschaft zu identifizieren, an denen sich wirtschaftlich erfolgreiches Handeln ausrichtet. Zunächst lässt sich feststellen, dass wir in den letzten Jahrhunderten drei große Phasen in Bezug auf die Industrialisierung durchlaufen haben, anhand derer sich die Gegebenheiten lokaler und globaler Märkte recht gut charakterisieren lassen.

Das vorindustrielle Zeitalter war geprägt von hohen Produktionskosten, was sich auch in teuren Gütern widerspiegelte. Hierdurch gab es verhältnismäßig sehr wenige potenzielle Kunden, die überhaupt am Markt teilnehmen konnten. Der Herstellungsprozess war insgesamt sehr aufwendig und konnte praktisch nicht skaliert werden, da die Produktionsstätten noch maßgeblich durch das Handwerk geprägt waren.

Im industriellen Zeitalter wurden diese Manufakturen im Rahmen der Industrialisierung durch Fabriken verdrängt. Die Standardisierung der Einzelschritte diente dabei als Vorbedingung für den Einsatz von Maschinen. Die Fließbandfertigung ermöglichte schließlich eine erhebliche Reduktion der Produktionskosten. Die niedrigeren Preise erhöhten damit die potenzielle Käuferschaft um ein Vielfaches. Dies ging sogar so weit, dass man als Anbieter die Konditionen in diesen ungesättigten Märkten diktieren konnte, da eine Abnahme aller – mit gesundem Menschenverstand produzierten – Güter faktisch quasi garantiert war. Um die Gewinne weiter zu maximieren, erfolgte eine beständige Optimierung von Prozessen, Organisationsformen, Verfahren, Technologien und Denkmodellen. Bezeichnend für den Taylorismus und das Scientific Management war außerdem die Etablierung einer Prozesssteuerung, die auf Arbeitsstudien und ausgeprägtem Management in der Arbeitsvorbereitung beruhte. Hierbei wurde auch die Annahme postuliert, dass exakte und detaillierte Vorgaben für die jeweiligen Arbeitsschritte auch in Bezug auf menschliche Abläufe fixiert und eingehalten werden müssen („one best way“).

Im post-industriellen Zeitalter, in dem wir uns heute befinden, hat sich das Kräfteverhältnis zwischen Anbietern und Kunden umgedreht. Verantwortlich hierfür ist die zunehmende Sättigung der Märkte durch die optimierte, durchsatzstärkere Produktion bei gleichzeitiger Zunahme an Wettbewerbern je Marktteilnehmer. Um hier bestehen (und damit verkaufen) zu können, ist nun wieder eine Ausrichtung am Kunden mit gezielter Befriedigung der persönlichen Bedürfnisse essenziell. Dies beinhaltet das Einholen und die zeitnahe Reaktion auf Kunden-Feedback durch eine adaptierte Produktion von Waren beziehungsweise angepasste Bereitstellung von Dienstleistungen in einer nächsten Iteration oder durch eine Neuentwicklung, die wiederum schnell verprobt werden muss (u.a. durch Rapid Prototyping). Durch das Voranschreiten der Automatisierung hat sich die IT zum zentralen Nervensystem in fast allen heutigen Unternehmen in Betrieb, Forschung und Entwicklung etabliert. Die IT-Durchlaufzeiten sind dabei zu einem limitierenden Faktor bei modernen Wertschöpfungsketten – von der Idee bis zum fertigen Produkt – geworden. Als probate und erforderliche Mittel für den Geschäftserfolg sind nun eine hohe Flexibilität und eine Reduktion der Fertigungstiefe in der IT entscheidend.

Diese wirtschaftlichen Anforderungen an die moderne IT sind die eigentlichen Treiber, die den Einsatz von Cloud Native Applications nicht nur begünstigen, sondern langfristig sogar zunehmend erzwingen können.

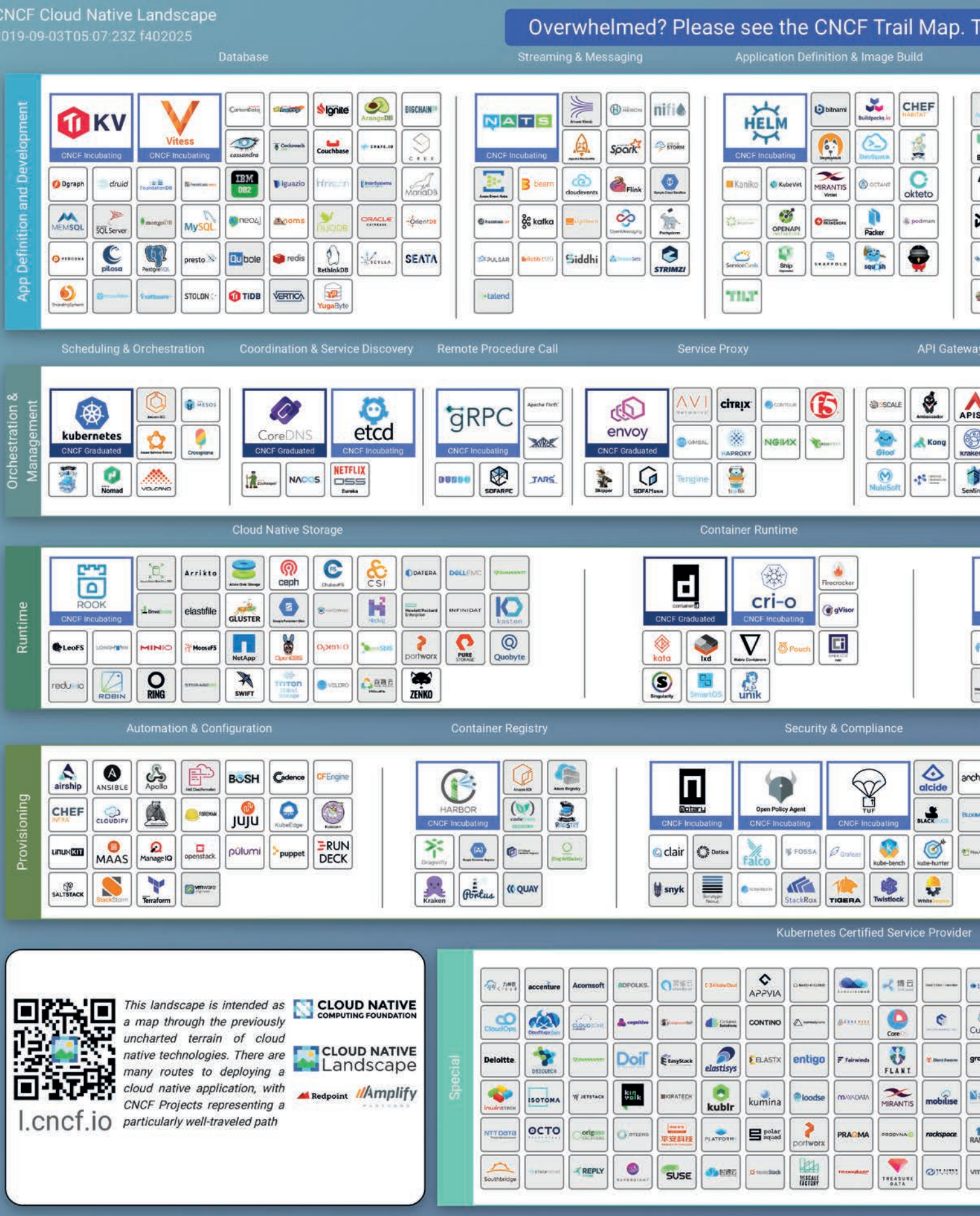
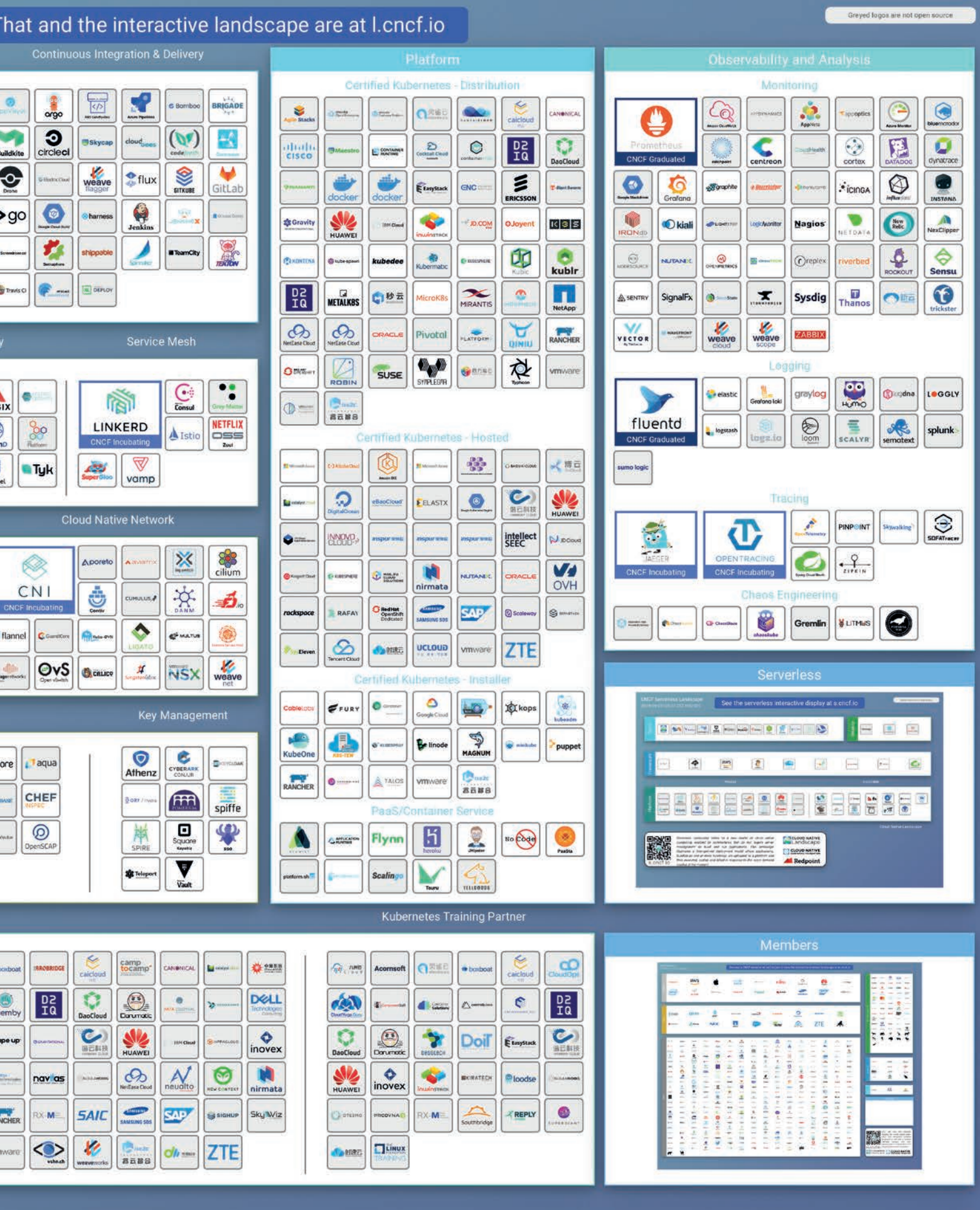


Abbildung 1: CNCF - Cloud Native Landscape (Quelle: Cloud Native Computing Foundation)


That and the interactive landscape are at l.cncf.io
Greyed logos are not open source
Continuous Integration & Delivery
Service Mesh
LINKERD
CNCF Incubating
Cloud Native Network
CNI
CNCF Incubating
Key Management
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Prometheus
CNCF Graduated
Logging
fluentd
CNCF Graduated
Tracing
JAEGER
CNCF Incubating
OPENTRACING
CNCF Incubating
Chaos Engineering
Serverless
See the serverless interactive display at a.cncf.io
Kubernetes Training Partner
Members

Cloud Native Architectures

Dass jedes Unternehmen nun ein Softwareunternehmen werden muss, wie es Marc Andreessen im Artikel „Why software is eating the world“ *[3]* beschreibt, gilt hierbei allerdings als überholt. Wichtig ist es dennoch, den Fokus auch auf die neuen Möglichkeiten, die sich durch CNA ergeben, nicht aus den Augen zu verlieren, um nicht eiskalt von IT-zentrischen Unternehmen mit digitalen und disruptiven Geschäftsmodellen wie Uber, Airbnb oder Netflix überrascht zu werden. Vielmehr können sie von diesen Vorreitern in

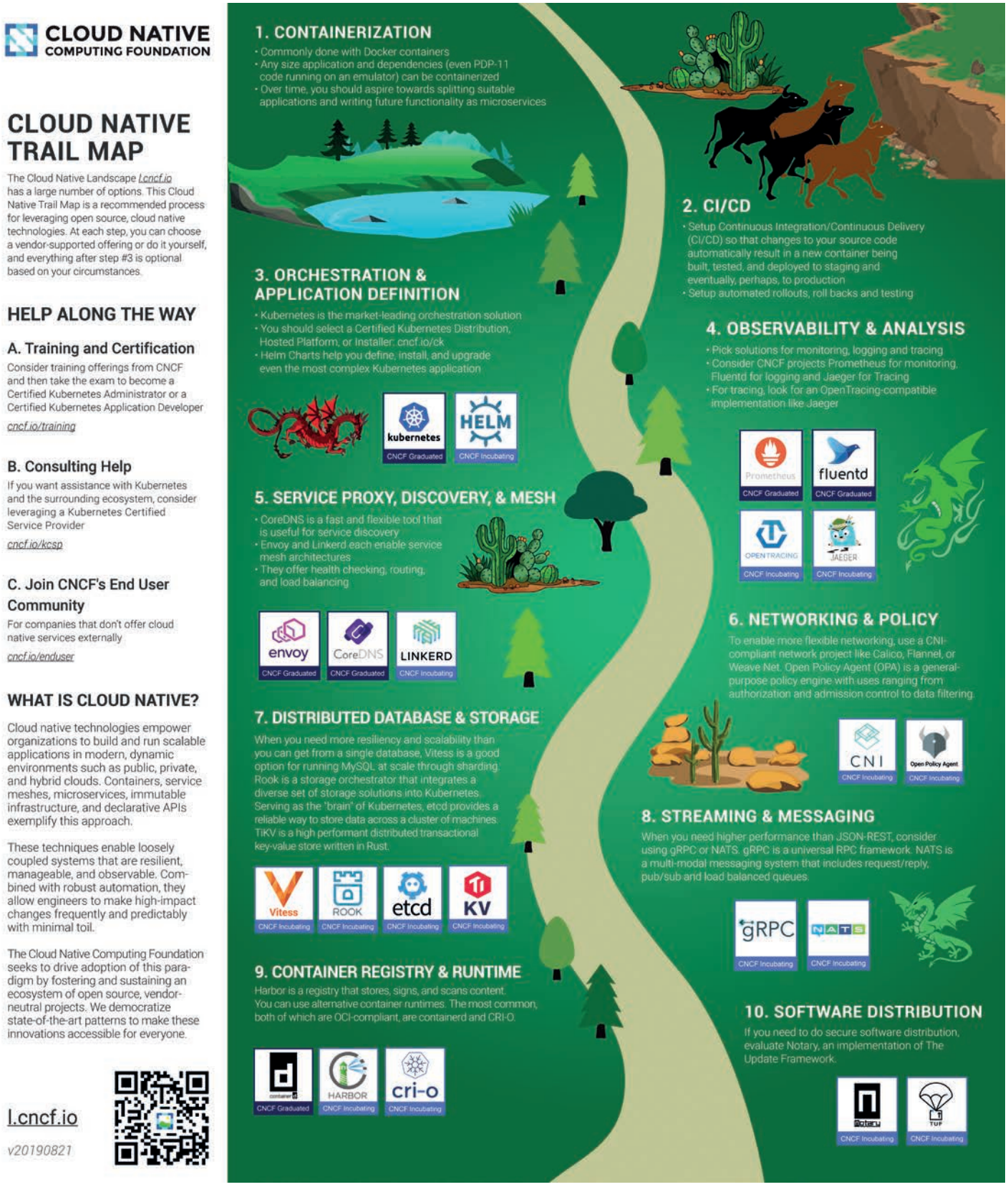


Abbildung 2: CNCF - Cloud Native Trail Map (Quelle: Cloud Native Computing Foundation)

der IT lernen und dabei sogar von ihren Errungenschaften direkt profitieren.

Firmen wie Netflix haben letztlich durch die Zurverfügungstellung von Netflix OSS *[4]* im Jahr 2012 als Open Source auch bewiesen, dass ihre Entwicklerteams in den vergangenen Jahren in der Lage waren, neue Anwendungsarchitekturmuster, welche notwendig waren, um eine Migration in die Cloud zu vollziehen, erfolgreich in Code umzusetzen. Das Teilen dieses Codes wurden später auch als Maßstab genommen, um industrieseitig zu beschreiben, was Cloud Native Architectures beziehungsweise implementierte Cloud Native Patterns ausmacht.

Cloud Native Foundation

Im Jahr 2015 hat sich dann erst mal ein Konsortium bestehend aus weiteren Vorreitern in der IT für Cloud-Anwendungen unter dem Dach der Linux Foundation als neu gegründete Cloud Native Computing Foundation (CNCF, www.cncf.io) zusammengetan, das mittlerweile aus über 100.000 Mitgliedern besteht. Unter den Gründungsmitgliedern sind unter anderem Firmen wie Cloud Foundry, Core OS, Docker, Google, Mesosphere, Red Hat, Twitter, VMware oder Weaveworks anzutreffen, die sich als gemeinsames Ziel gesetzt haben, bestehende Technologien zu standardisieren und Projekte als Open Source zu kultivieren, um so den Einstieg in die Cloud-Native-Welt für Entwickler und auch andere Unternehmen zu erleichtern. Allen soll also ermöglicht werden, skalierbare Software zu erstellen, die hochverfügbar ist und gleichzeitig eine hohe Änderungsrate zulässt. Hierfür müssen aus Sicht der CNCF drei Kernthemen abgedeckt werden:

- Speed: Firmen sollen schnell an Apps arbeiten können, ohne sich lange mit der Infrastruktur, Integration oder einhergehenden Experimenten und Bugs aufhalten zu müssen.
- Freedom: Jede App soll basierend auf Open Source Packaging überall laufen können, um einen Cloud Service Vendor Lock-in zu vermeiden
- Trust: Firmen müssen auf eine Uptime von 24/7 mit hoher Wiederherstellungsrate (Recovery Time) vertrauen können, wobei jede Komponente unabhängig skalierbar ist.

Zudem wurde den firmengetriebenen Stellungnahmen zu Cloud Native, wie zum Beispiel von Pivotal *[5]* und Red Hat *[6]* auf eigenen Landing Pages zelebriert, eine konsolidierte, vereinheitlichte CNCF Cloud Native Definition *[7]* gegenübergestellt, die auch die oben genannten gemeinsamen Zielstellungen der Unterstützer beinhaltet. Weiterhin werden, um diese Ziele zu ermöglichen, konkrete Cloud-Native-Infrastrukturprojekte gefördert, die zur Vereinfachung für Anwender in unterschiedliche Reifegrade eingeordnet wurden:

- Graduated Projects: Kubernetes, Prometheus, Envoy, CoreDNS, containerd, Fluentd,
- Incubating Projects: u.a. OpenTracing, gRPC, Linkerd, Helm, etcd,
- Sandbox Projects: Aktuell sind dies über 20 Projekte.

Weitere Projekte, die sich im Cloud-Umfeld entwickelt haben, hat die CNCF in der Cloud Native und Serverless Landscape *[8] (siehe Abbildung 1)* festgehalten und kategorisiert, wobei sie darauf hinweist, dass man insbesondere ihren gereiften Projekten Vertrauen schenken darf.

Um den Einstieg in die Cloud-Native-Welt weiter zu vereinfachen, hat die CNCF eine Cloud Native Trail Map *[9] (siehe Abbildung 2)* etabliert. Hier beschreibt sie in Form eines Posters 10 Etappen, an denen sich ein Unternehmen bei der eigenen Cloud-Native-Reise entlanghangeln kann.

Ausblick und weitere Initiativen zu Cloud Native Applications

Wir sehen also, dass sich Cloud Native auf gutem Weg befindet, um zu einem festen Standard in der IT-Welt zu reifen. Übrigens gibt es neben den Bemühungen der CNCF auch einige andere nennenswerte industrieseitige Vorstöße und Hilfestellungen, wie das 12-Factor-App-Manifest *[10]*, die Open-Container-Initiative *[11]*, die Resilient Software Design Patterns oder die SOLID Principles for Cloud Native Applications *[12]* oder Cloud Native Container Design Principles *[13]*, bei denen es sich ebenfalls lohnt, sie sich näher anzuschauen, auch wenn es hierzu noch keinen einheitlichen Konsens gibt.

Quellen

[1] Peter Mell, Timothy Grance (2011): The NIST Definition of Cloud Computing. Special Publication 800-145. National Institute of Standards and Technology, Gaithersburg MD 20899-8930.
[2] Nane Kratzke, Peter-Christian Quint (2017): Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study. The Journal of Systems and Software, Lübeck.
[3] Marc Andreesen (2011): Why Software is eating the world. The Wall Street Journal, New York: *https://www.wsj.com/articles/SB10001424053111903480904576512250915629460*
[4] *https://netflix.github.io*
[5] *https://pivotal.io/de/cloud-native*
[6] *https://www.redhat.com/de/solutions/cloud-native-development*
[7] *https://github.com/cncf/toc/blob/master/DEFINITION.md*
[8] *https://landscape.cncf.io*
[9] *https://github.com/cncf/trailmap*
[10] *https://12factor.net*
[11] *https://www.opencontainers.org*
[12] *https://www.redhat.com/files/summit/session-assets/2018/M1135-SOLID-principles-for-cloud-native-applications-Distribution.pdf*
[13] *https://dzone.com/articles/cloud-native-container-design-principles*

Über den Autor

Benjamin ist Technical Lead für Cloud Technologies bei der codecentric AG, regelmäßiger Speaker auf internationalen Konferenzen und anerkannter Oracle Groundbreaker Ambassador. Seine aktuellen Themenschwerpunkte sind neben Cloud Native Applications auch Domain-Driven Design und agiles Projektmanagement. Zudem hat er als Gründer von sechs User Groups in den letzten Jahren über fünfzig IT-Events organisiert, darunter auch einige Konferenzen wie die JavaLand und microXchg, eine führende Konferenz zu Microservices. Nebenbei ist er unter anderem auch als Dozent an verschiedenen Hochschulen in Thüringen unterwegs.

Benjamin Nothdurft
b.nothdurft@gmail.com
Benjamin.nothdurft@codecentric.de

Reise in die chinesische Cloud

Jessica Steger, Logicalis

In diesem Artikel geht es um ein Projekt zum Aufbau einer Kunden-Cloud-Umgebung mit Oracle-Datenbanken und -Applikationsservern in China sowie die dabei aufgetretenen Fallstricke. Denn im Cloud-Umfeld ist alles sehr dynamisch und je nach Anbieter auch erst im Aufbau, daher ist ein hohes Maß an Flexibilität und Geduld gefragt, um Projekte in diesem Umfeld erfolgreich abschließen zu können.

Im Sommer 2018 wurde der Wunsch eines unserer Kunden an uns herangetragen, die Systemlandschaft für deren chinesische Kunden in der Oracle Cloud im Rahmen eines Proof of Concept (PoC) aufzubauen und nach erfolgreichem Abschluss auch im Rahmen eines Managed-Service-Vertrages zu betreiben. Initial war als Ziel-Setup der Aufbau einer PoC-Umgebung in der OCI-Classic-Umgebung in Amsterdam geplant, um dann das für Herbst 2018 geplante Rechenzentrum in Shanghai für die Produktions-Umgebung zu nutzen. Hintergrund war die Notwendigkeit, das Public Peering zwischen den Oracle-Rechenzentren für den interkontinentalen Datenaustausch zu nutzen, da mehrere Hundert Gigabyte an Daten monatlich en bloc von Deutschland nach China transferiert werden sollten. Als Oracle auf der OOW 2018 verkündete, keine OCI-Classic-Rechenzentren mehr aufbauen zu wollen und bis Ende 2019 keine Oracle-Cloud-Rechenzentren in China geplant sind, wurde das Projekt in der PoC-Phase abgebrochen und nach einer Alternative gesucht. Alibaba als möglicher Cloud-Anbieter im Großraum Frankfurt und mit chinesischen Rechenzentren mit Standorten unter anderem in Shanghai bietet dabei über das Cloud Enterprise Network (CEN) die Möglichkeit, den Datenaustausch zwischen Deutschland und China über das Alibaba-eigene Backbone durchzuführen. Je nach freigeschalteter Bandbreite sind die Zeiten für den Massen-Daten-Austausch zum Beispiel für den Transfer von Datenbank-Dumps skalierbar und müssen über eine Kosten-Nutzen-Rechnung vorab geprüft werden. Ein weiterer Vorteil der Alibaba Cloud ist, dass es keine Latenzen und Paketverluste

durch die Great Firewall der chinesischen Regierung beim Datenaustausch über das CEN gibt. Diese Firewall-bedingte Latenz stellt für alternative Anbindungs-Lösungen ein großes Problem dar, wenn zum Beispiel der Zugriff auf eine chinesische Umgebung über eine MPLS-Leitung von Standorten außerhalb Chinas durch die Great Firewall erfolgt. Da diese Einschränkungen durch das CEN bei einer Nutzung der Alibaba Cloud nicht existieren, wurde die Alibaba Cloud als zukünftiger Cloud Provider ausgewählt.

Was muss man während der Planung beachten?

Im Oracle-Datenbank-Umfeld muss generell betrachtet werden, welche Lösungen für eine höhere Verfügbarkeit möglich sind. Von Oracle sind Real-Application-Cluster- Lösungen nur in der Oracle Cloud zertifiziert. Daher wurde für die chinesische Cloud- Umgebung des Kunden zur Erhöhung der Verfügbarkeit für die Produktions-Datenbank eine Standby-Lösung mittels Data Guard umgesetzt. Die Applikationsserver und die Standby-Datenbanken wurden dabei über verschiedene Rechenzentren (Alibaba: Availability Zones) innerhalb der Region Shanghai verteilt.

Ein großes Problem stellt die Lizenzierung von Oracle-Datenbanken in virtualisierten Umgebungen dar. Die Alibaba Cloud ist aktuell kein von Oracle autorisierter Cloud-Anbieter für virtualisierte Oracle-Datenbanken *[1]*. Um diese Thematik zu umgehen, werden in der Alibaba Cloud Bare-Metal-Systeme mit Intel-Prozessoren für die Systeme mit Oracle-Datenbanken eingesetzt. Aktuell ist das Core-zu-RAM-Verhältnis dieser Umgebungen aus Kosten-Sicht nicht optimal; ein Wunsch zu einem besseren Core-zu-RAM-Verhältnis wurde bei Alibaba platziert.

Die aktuell möglichen Ausbaustufen finden Sie in *Tabelle 1*.

Weitere Herausforderungen ergeben sich durch die staatlichen Vorgaben Chinas. So muss für kommerzielle Webseiten eine ICP Commercial License vorliegen. Diese können Firmen mit einem Standort in China beim Ministry of Industry and Information Technology (MIIT) für die Domäne ihrer chinesischen Webseite beantragen. Ist die Webseite nicht registriert und im chinesischen Adressraum verfügbar oder passt die verwendete URL nicht zur ICP-Lizenz, so wird diese in der Great Firewall geblockt und darauf verwiesen, dass diese Webseite nicht registriert ist. Auf der Start-Webseite der registrierten Domain muss die ICP-Lizenz genannt und ein Link zur ICP-Seite des MIIT vorhanden sein.

Für die Nutzung von Elastic-Compute-Service-Instanzen und Object Storage Buckets in Rechenzentren des chinesischen Festlandes gibt es eine weitere Vorgabe der chinesischen Regierung: die Real-Name Registration. Für Firmen-Accounts muss ein Handelsregister-Auszug hochgeladen werden. Dieser wird mit den weiteren Firmen-Angaben geprüft, und erst nach Freigabe durch Alibaba kann man oben genannten Service in den chinesischen Rechenzentren benutzen. Dies kann ein paar Tage beanspruchen und muss im Projekt mit eingeplant werden.

Um ein Cloud-Projekt erfolgreich umsetzen zu können, sollte man vorab bereits evaluieren, ob die Umgebung neu für die Cloud erstellt wird (Cloud-native) oder ob eine bestehende Applikation in die Cloud migriert werden soll (Lift and Shift). Ein oft vernachlässigtes Problem stellt dabei die fehlende Standardisierung von Anwendungen dar. In diesem Projekt wurde eine stark optimierte Applikation mit hohen Anforderungen an ein Loadbalancer-Regelwerk in einer Cloud abgebildet. Cloud-Lösungen basieren aber auf Industrie-Standards, und jede Abweichung im Leistungsumfang erzeugt Probleme bei der Umsetzung. So wird aus einem Lift and Shift schnell ein Re-Architect mit entsprechenden Mehraufwänden sowie Verzögerungen im Projekt-Ablauf. Für dieses Projekt wäre die Nutzung eines F5-Marketplace-Image eine schnelle, aber kostenintensive Lösung gewesen. Solche Mehraufwände wurden durch ein versiertes Team abgefangen, indem fehlende Funktionsumfänge von Cloud-Loadbalancern durch alternative Loadbalancer-Lösungen auf IaaS-VMs abgebildet werden.

Generell sollte man das Thema Netzwerk sehr ernst nehmen, auch wenn einem die leichte Bereitstellung von Netzwerk-Infrastruktur durch die GUIs der Cloud-Anbieter den Eindruck vermitteln, dass das alles kein Hexenwerk sei. Die Best Practices der Cloud-Anbieter sollten vorab studiert werden und es ist von Vorteil, im Projekt-Team Mitarbeiter mit Netzwerk-Know-how einzubinden. Generell sollten sich zum Beispiel alle vom Kunden genutzten privaten Netzwerke in unterschiedlichen IP-Adress-Bereichen befinden, egal ob in der Cloud oder On-Premises. Wenn die beiden Welten miteinander verbunden werden und sich Netzwerk-Adressbereiche überschneiden, muss sonst leider nachträglich angepasst werden, sodass keine Überlappungen mehr existieren. Hat man die Cloud-Umgebungen nicht über Tools wie Ansible oder Terraform bereitgestellt, dann ist der Aufwand zur Korrektur deutlich höher. Vom Cloud-Anbieter reservierte Adress-Bereiche dürfen nicht genutzt werden. Oracle verwendet zum Beispiel in seiner Gen 2 Cloud OCI den Adress-Bereich 169.254.0.0/16 für seine iSCSI-Services zur Bereitstellung von Boot- und Block-Volumes, der nicht für eigene IP-Adressen genutzt werden darf. Solche Vorgaben sind in der Dokumentation des jeweiligen Cloud-Anbieters zu finden und müssen berücksichtigt werden.

Auch das Thema Sicherheit muss bei der Planung der Netzwerke von Cloud-Lösungen höchste Priorität haben. Es muss vorab geklärt werden, welche Systeme zwingend eine öffentliche IP-Adresse benötigen und welche Systeme über NAT-Gateways mit dem Internet kommunizieren können. Jedes System mit

Instanz-Typ	Intel-Cores	RAM (GB)	Bandbreite (Gbit/s)
ecs.ebmhfg5.2xlarge	4	32	6
ecs.ebmc4.8xlarge	16	64	10
ecs.ebmg5.24xlarge	48	384	10

Tabelle 1

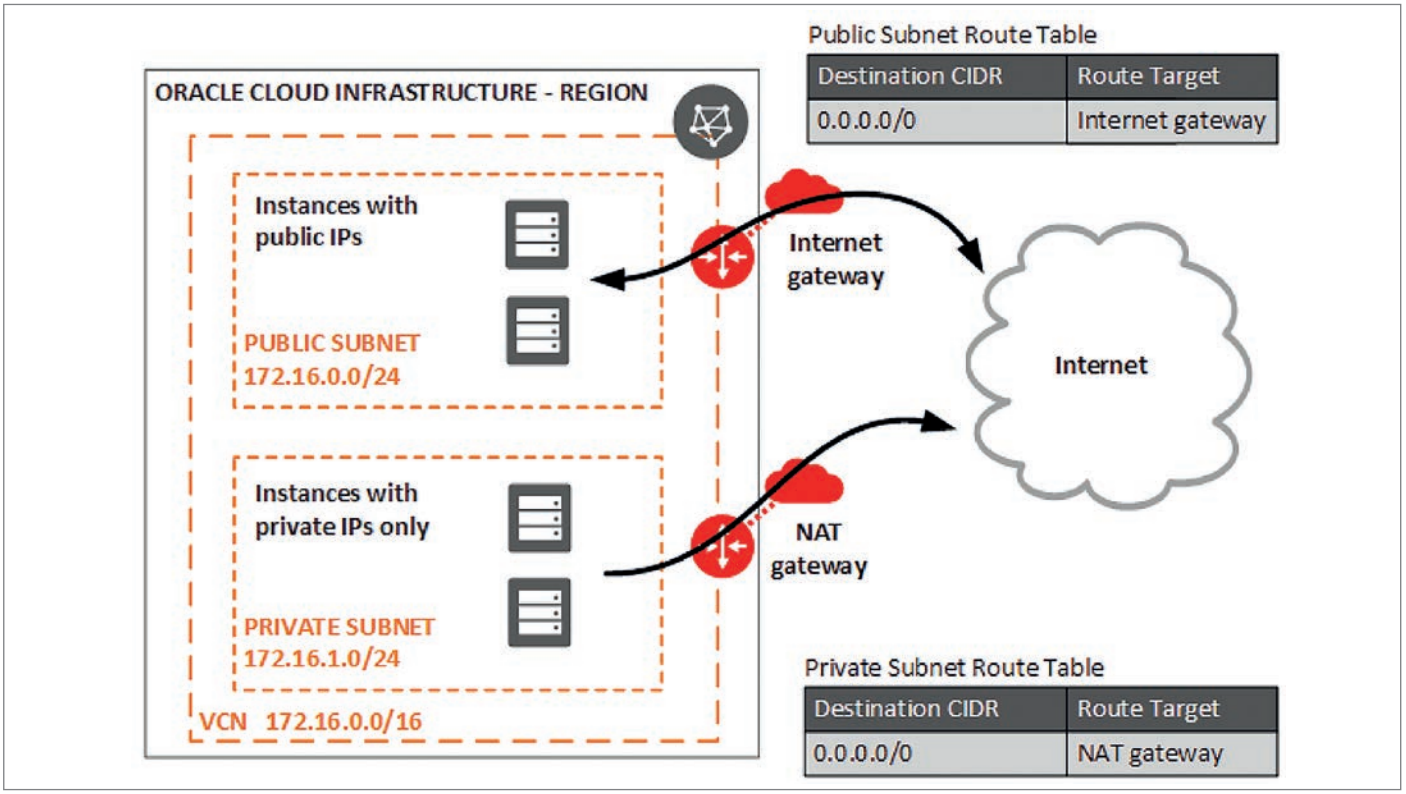


Abbildung 1: Private und öffentliche Netzwerke in der Cloud (Quelle © Oracle, OCI User Guide [2])

einer öffentlichen IP-Adresse ist angreifbar und daher ein potenzielles Sicherheitsrisiko.

Im folgenden Bild aus dem Oracle Cloud Infrastructure Users Guide wird beispielhaft die Problematik erläutert. Dort ist eine Aufteilung in zwei Netzwerk-Bereiche dargestellt. Der Netzwerk-Traffic aus dem Subnetz mit Instanzen, die öffentliche IP-Adressen besitzen können, wird über das Internet Gateway in das Internet geroutet (bei Alibaba wird hier die DNAT-Funktion des Alibaba NAT Gateway verwendet). Diese Systeme sind über ihre Public IPs auch aus dem Internet erreichbar. Anfragen aus dem Subnetz mit Instanzen mit rein privaten Adressen werden über das NAT Gateway geroutet (bei Alibaba wird hier die SNAT-Funktion des Alibaba NAT Gateway verwendet). Aus dem Internet sind diese Instanzen nicht erreichbar. Aus Sicherheitsgründen sollte der Anteil der Instanzen mit öffentlichen IP-Adressen daher so gering wie möglich sein und freigegebene Ports für Zugriffe aus dem Internet sorgfältig gewählt werden.

Ein weiterer Baustein für die Planung ist der Standort für das Monitoring der Cloud-Lösung. Für das Monitoring der Cloud-Umgebung in diesem Projekt wurde eine eigene Oracle-Enterprise-Manager-Umgebung (OEM-Umgebung) im gleichen Cloud-Netzwerk wie die produktiven Instanzen gewählt. Über diese können auch bei einem Ausfall des CEN Backbone die Alarme über das Internet an das ServiceNow der Logicalis gesendet werden. Ein Monitoring vom OEM des Kunden aus dessen On-Premises-Rechenzentrum birgt das Risiko, dass die Metriken von den OEM-Agenten der Cloud-Infrastruktur bei Netzwerk-Problemen zwischen dem Kunden-Rechenzentrum und der Cloud-Lösung nicht oder nur verzögert beim OEM eintreffen.

Ein weiterer Aspekt bei der Planung sind die vom Cloud-Anbieter unterstützten Betriebssystem-Versionen der Cloud-VMs. Diese sind wegen der Standardisierungen oftmals nicht in gleicher Version vorhanden wie im Kunden-Umfeld. Daher muss vorab ermittelt werden, welche Abweichung tolerierbar ist und welche nicht. So war zum Zeitpunkt des PoC für die Bare-Metal-Instanzen Red Hat nicht von Alibaba freigegeben und diese Systeme mussten mit SUSE Linux installiert werden. Für Datenbank-Umgebungen sind solche Abweichungen weniger problematisch als für die VMs der Applikationsserver. Dort sollten die Betriebssystem-Versionen möglichst gleich denen der Entwicklungs-Umgebungen beim Kunden sein.

Nach dem Go-Live

Ein Thema, von dem man völlig überrascht wird, ist die Abrechnung der Cloud-Nutzung. Da die Leistungen in der Cloud neu sind, müssen die entsprechenden Posten der Einkaufsabteilung des Cloud-nutzenden Unternehmens bekannt gemacht werden, damit dieser die in der Cloud aufgetretenen Aufwände passend gegenbuchen kann.

Wenn als Zahlungsmöglichkeit eine Kreditkarte hinterlegt ist, muss der eingerichtete Kreditrahmen für die zu buchenden Cloud-Leistungen ausreichen. Im Falle einer Umstellung von „Pay as You Go“ oder „Monthly Subscription“ auf eine „Yearly Subscription“ ist schnell der Kreditrahmen ausgereizt. Eine Subscription bezeichnet hier eine Vorauszahlung für den jeweiligen Zeitraum mit entsprechenden Rabattierungen der Services. Weitere notwendige Leistungen, deren Abbuchung von der Kreditkarte im gleichen Monat anfallen, können dann nicht mehr gebucht werden. Dies sollte beach-

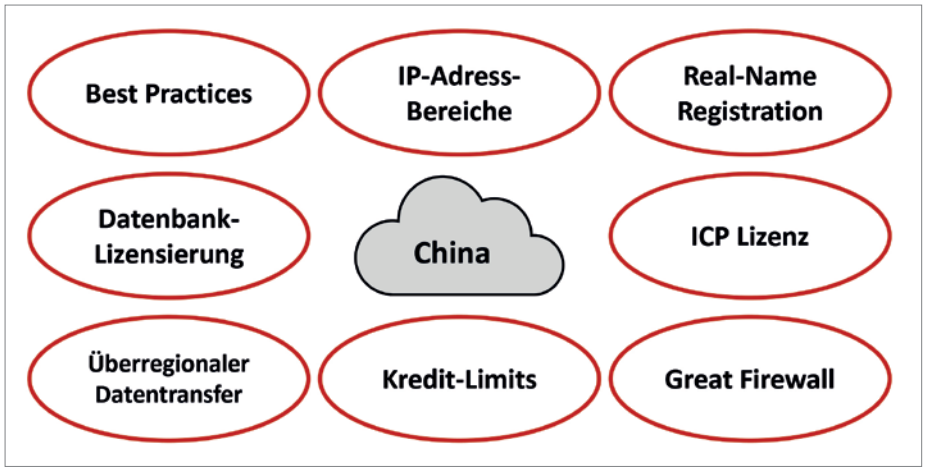


Abbildung 2: Wichtige Bausteine in diesem Projekt (Quelle: Jessica Steger)

tet werden, um ungewollte Downtimes zu verhindern.

Die Rechnungen der gebuchten Cloud-Leistungen waren initial bei einer Abrechnung über eine Kreditkarte nur im Cloud-Portal verfügbar. Dies ist insofern schwierig, als normalerweise Einkaufsabteilungen nicht mit solchen Portalen vertraut sind und die Consultants im Projekt, die administrative Rechte im Cloud-Portal besitzen, nichts mit Rechnungsstellungen und -buchungen zu tun haben. Um diese Probleme zu beheben, wurde im Projekt in Abstimmung mit der Alibaba-Cloud-Rechnungsabteilung auf ein monatliches Kredit-Limit mit monatlicher Rechnungsstellung umgestellt. Seitdem wurden die Rechnungen direkt der Einkaufsabteilung zur Verfügung gestellt. Solche Kleinigkeiten zeigen, dass das Handling solcher bereits im Unternehmen etablierten Prozesse beim Einsatz von Cloud- Lösungen eventuell neu überdacht werden müssen.

Generell sollte man zur Optimierung seiner Cloud-Kosten prüfen, welche Ressourcen wegen der damit verbundenen Kosten-Einsparung auf eine jährliche Abrechnung (Yearly Subscription) umgestellt werden sollten. Man sollte sich dann auch sicher sein, dass diese auch wirklich für ein ganzes Jahr benötigt werden. Nachfolgende Umstellungen oder Anpassungen in der Architektur können unter Umständen nicht mehr ausgeglichen werden; man zahlt dann zum Beispiel für virtuelle Maschinen den Jahrespreis, obwohl diese doch nicht für das ganze Jahr benötigt werden. Im Projekt wurden zum Beispiel der Block Storage für die Daten der Produktions-Datenbank aus Performance-Gründen auf SSDs umgestellt. Da die Bare-Metal-Systeme aktuell nur über Monthly Subscription abgerechnet werden können (Yearly Subscriptions sind zukünftig möglich), waren mit dieser Anpassung glücklicherweise keine großen Mehrkosten verbunden. Die Performance der Datenbank war danach deutlich besser als zuvor, daher empfiehlt es sich, bereits im PoC die unterschiedlichen Storage-Typen zu verproben, wenn das den vom Kunden vorgegebenen Kostenrahmen nicht sprengt.

Eine weitere Herausforderung für das Projekt im Bereich der Cloud-Kosten war die monatliche Bandbreiten-Erhöhung des CEN-Netzwerkes von 2 Mbit auf 10 Mbit für den Daten-Upload nach China, da nach einer Umstellung der CEN-Kosten auf eine Yearly Subscription immer der volle Jahrespreis für jede Bandbreiten-Anpassung anfällt. Wenn also eine Bandbreite von 2 Mbit bereits bezahlt wurde, wird bei einer Erhöhung der Bandbreite auf 10 Mbit der Fehlbetrag zum Jahrespreis der 10-Mbit-Bandbreite abgebucht. Der mit Alibaba vereinbarte Kreditrahmen beziehungsweise das Kreditlimit der Kreditkarte sollte für solche Anwendungsfälle mindestens verfügbar sein. Nach dem erfolgreichen Massendaten-Austausch wird bei der anschließenden Reduktion der Bandbreite auf die ursprünglichen 2 Mbit ein Cloud-Guthaben im Cloud-Konto hinterlegt. Man zahlt also nur kurzfristig für die schnellere Bandbreite. Man darf nur nicht die Übersicht über die anfallenden Kosten und die noch verfügbaren Kreditlimits verlieren. Eine zentrale Ansprechperson, die diese Themen im Blick hat, sollte vorab bestimmt sein, damit es dort keine Abstimmungsprobleme gibt.

Fazit

In der folgenden Grafik sind zusammenfassend die für dieses Projekt wichtigen Bausteine aufgezeigt.

Die gemachten Erfahrungen zeigen deutlich, dass es bei der Umstellung auf Cloud-Lösungen viele Kleinigkeiten zu beachten gibt, die einem im Nachgang bei Nicht-Beachtung vor Probleme stellen können. Eine sorgfältige Planung der Architektur und da vor allem des Netzwerks ist hilfreich, da sich strukturelle Änderungen nicht immer leicht beheben lassen. Auch müssen Länder-spezifische Besonderheiten berücksichtigt werden. Proof of Concepts helfen, die geplante Lösung hinsichtlich der Funktionalität und der Performance zu testen, daher sollten diese immer mit eingeplant werden. Denn die final umgesetzten Infrastruktur-Lösungen in der Cloud weichen nach Projektende oftmals von dem ab, was ursprünglich geplant war.

Quellen

[1] *https://www.oracle.com/assets/cloud-licensing-070579.pdf*
[2] *https://docs.cloud.oracle.com/iaas/pdf/ug/OCI_User_Guide.pdf*

Jessica Steger
Jessica.Steger@logicalis.de

Ansible und Exadata – Eine perfekte Symbiose?

Timo Giese, Fiducia & GAD IT

Die eigene Infrastruktur zu automatisieren, gehört heute zum guten Ton. Daher liegt es nur nahe, dies auch für Oracle-Exadata-Datenbanksysteme zu tun. Ein Tool, das sich dafür anbietet, ist Ansible, das gleichzeitig nachvollziehbare Aktionen wiederholbar mit gleichem Ergebnis ausführen kann und im selben Zug die Infrastruktur dokumentiert. Schluss mit langen Listen an manuellen Anpassungen, die nach der Basisinstallation für die Integration in die eigene Unternehmens-Infrastruktur abgearbeitet werden müssen, und her mit der automatischen Konfiguration durch Ansible. Dieser Artikel zeigt die Konzepte hinter Ansible und die Herausforderungen in Kombination mit Exadata-Systemen.

Was ist Ansible?

Ansible wurde von Michael DeHaan als Projekt im Jahre 2012 geschrieben und gestartet. Ansible ist eine Configuration-Management-Software, die sich durch ein „Agentless"-Konzept auszeichnet. Dies wird dadurch erreicht, dass für den Zugriff auf das Zielsystem lediglich SSH benötig wird. Ein weiterer Punkt, der für Ansible spricht, ist, dass es Open Source ist und von jedem frei verwendet werden kann.

Mit sogenannten „Playbooks" und einer großen Sammlung an fertigen Modulen für viele Aufgaben im Systems-Managementbereich ist es möglich, mit einfachen und verständlichen Mitteln nicht nur seine Infrastruktur und Softwarekomponenten zu installieren, sondern sie zusätzlich in lesbarer Form zu dokumentieren. Ansible greift dafür auf die Beschreibungssprache YAML und den Template-Filter Jinja zurück.

Charakteristisch für YAML ist die Verwendung von Leerzeichen und Einrückungen, um Aufgaben, sogenannte Tasks, und deren Parameter aneinanderzureihen.

Ansible-Designprinzipien

Eines der wichtigsten Prinzipien von Ansible ist das der Idempotenz. Prägend hierfür ist die Eigenschaft, dass ein definierter Ablauf von Aktionen, zusammenhängend beschrieben in einem Playbook in YAML-Syntax, bei wiederholter Ausführung des

Playbooks immer zum gleichen Zielzustand führt. Wird eine Konfiguration beim ersten Aufruf durch Ansible angepasst, führt ein erneuter Aufruf zu keiner weiteren Änderung am System, da der korrekte Wert bereits gesetzt ist und erkannt wird.

Werden Ansible-Standardmodule verwendet, die es für die meisten Systemadministrations-Tasks gibt, so ist dieses Prinzip gewährleistet. Werden keine Standardmodule verwendet, so muss der Benutzer selbst für die Idempotenz sorgen oder mit ungewollten Quereffekten leben.

Ansible ist vom Design her so konzipiert, dass es so wenig wie möglich zusätzliche Abhängigkeiten für die gemanagten Systeme mitbringt, genauso wie für den Managementserver, auf dem die Ansible Binaries installiert sind. Als Hauptabhängigkeit benötigt Ansible einen Python Interpreter, der auf Quelle und Ziel vorhanden sein muss. Es ist jedoch auch möglich, eine eigene Version davon einzusetzen, um keine Abhängigkeiten zu der auf dem Betriebssystem installierten Version zu haben. Dadurch ist gewährleistet, dass Ansible sehr zuverlässig und mit minimaler Einarbeitungszeit eingesetzt werden kann.

Zusätzlich zum „Core"-Ansible, der Open-Source-Komponente, gibt es noch einen Management Server, den Ansible Tower, der als Add-on mit grafischer Benutzeroberfläche und Scheduler kostenpflichtig angeboten wird. Dieser Artikel beschäftigt sich ausschließlich mit den Ansible-Core-Komponenten und kommt ohne zusätzlichen Tower aus.

```
[exa01:children]
exa01_vm01
exa01_vm02
exa01_dom0

[exa01:vars]
env=test

# VM01
[exa01_vm01]
exa01a01vm01 single_exec=yes
exa01a02vm01 single_exec=no

[exa01_vm01:vars]
host_type =domU
vlan_public=42
...
### DOM0
[exa01_dom0:children]
exa01_dom0_1
exa01_dom0_2

[exa01_dom0:vars]
host_type=dom0
vlan_public=42

[exa01_dom0_1]
exa01domz01
exa01domz02

# Infiniband
[exa01_ib]
exa01iba01 ansible_user=root host_type=infiniband
exa01ibb01 ansible_user=root host_type=infiniband

# Cellserver
[exa01_cellserver]
exa01c01 ansible_user=root host_type =cellserver
exa01c02 ansible_user=root host_type =cellserver
exa01c03 ansible_user=root host_type =cellserver
...
```

Listing 1: Exadata Inventory

Einsatzgebiete für Ansible

Ansible hat seine Stärken in der Systemautomation, der Verwaltung von Konfigurationen und der Software-Installation. Darüber hinaus ist es möglich, Ansible auch für eigene Use Cases aus dem Unternehmensumfeld einzusetzen. Dies kann durch die Erstellung eigener Module oder die Einbindung eigener Skripte in unterschiedlichen Programmiersprachen erfolgen. Python wird bevorzugt für die Erstellung von Modulen verwendet; ein Einsatz der Skriptsprache Shell sowie weiterer Sprachen ist ebenfalls möglich.

Wie schon im vorigen Abschnitt erwähnt, ist ein idempotentes Design wichtig und im Besonderen beim Einsatz von Modulen und eigenen Skripten selbst sicherzustellen.

Von der Idee zur Umsetzung

Exadata Engineered Systems von Oracle sind vordefinierte Systeme mit Komponenten, bei denen nicht nur die Hardware, sondern auch die Software perfekt aufeinander abgestimmt sind.

Für die Hardwarekomponenten wird auf Standard-X86-Hardware gesetzt, sowie im Softwarebereich auf das Betriebssystem Linux. Dabei könnte man meinen, dass es abseits der Datenbankadministration eigentlich nichts mehr an diesen Systemen zu tun gibt. Weit gefehlt, denn in jedem Unternehmen gibt es zentrale Dienste wie Logserver, zentrale Verzeichnisdienste, Drittsoftware etc., mit denen Exadata-Systeme ebenfalls in Berührung kommen. Beim Aufbau des ersten Systems ist es vielleicht noch möglich, diese Konfigurationsanpassungen manuell durchzuführen und penibel genau zu dokumentieren. Steht aber nach kurzer Zeit die Erweiterung dieser oder die Anschaffung neuer Systeme an, kann selbst bei 100-prozentig lückenloser Dokumentation der berühmte Copy-Paste-Fehlerteufel zuschlagen. Ist dies der Fall, dauert es erheblich länger, bis die Systeme einsatzbereit sind, da der Fehler erst gefunden und zusätzlich die Dokumentation ebenfalls noch angepasst werden muss. Was wäre also besser, als das Ganze zu automatisieren und dies mit Ansible zu tun?

Damit dies gelingt, wird in den folgenden Abschnitten zuerst auf die Grundlagen eingegangen, anschließend werden ausgewählte Use Cases vorgestellt.

Das Inventory

Bevor die Use Cases dargestellt werden, gilt es zuerst, das Augenmerk auf das „Inventory" zu richten. Das Inventory ist der zentrale Anlaufpunkt für die Automatisierung mit Ansible. Darin sind mindestens die zu verwaltenden Hosts inklusive einer logischen Gruppenzuordnung definiert, ergänzt durch weitere Konfigurationsvariablen, die das zu verwaltende System näher beschreiben.

Für die Exadata werden dafür die Compute Nodes, eventuell die virtuellen Hosts bei virtualisierter Exadata, Cellserver und InfiniBand-Switches angelegt.

Die Verwendung eines Inventory für mehrere Exadatas sowie die Erstellung eines Inventory pro Exadata ist ebenfalls möglich.

Beim Aufruf des Playbooks wird das zu verwendende Inventory dann als Parameter mitgegeben.

Beispielhaft ist in *Listing 1* ein Inventory für eine virtualisierte Exadata dargestellt. Eine Variablenzuweisung ist entweder auf Hostebene und/oder auf Gruppenebene möglich. Bei der Ausführung des Playbooks werden alle Variablen, die einen Host direkt oder indirekt referenzieren, den Playbooks zur Verfügung gestellt.

Gruppen von Hosts werden mit eckigen Klammern gebildet. Übergruppen (Gruppe in Gruppe) werden zusätzlich durch „:children" dargestellt.

```
/home/ansible/
  group_vars/    <- Basisverzeichnis für Gruppenvariablen
     all/        <- Gilt für alle Gruppen
        my_credentials.yml
     exa01/      <- Gilt für Targets der Gruppe exa01
        network_env.yml
        syslog_env.yml
     …
  host_vars/     <- Basisverzeichnis für Hostvariablen
     my_host/    <- Name des Hosts im Inventory
        my_vars.yml
  exa_inventory <- Inventory-Datei
```

Listing 2: Gruppenstruktur im Inventory

Bei der Exadata hat dies den Charme, dass beispielsweise mehrere Compute Nodes und im Falle einer virtualisierten Exadata zusätzlich die virtuellen Maschinen zusammen für die Ausführung von Aktionen ausgewählt werden können. Darüber hinaus kann abhängig vom Typ des Hosts zum Beispiel ein Parameter einen unterschiedlichen Wert haben und dieser individuell automatisiert auf allen Komponenten einer Gruppe ausgebracht werden.

In der Regel wird ein eigener Benutzer für die Verbindung auf die Zielserver verwendet. Im Standardfall wird der Benutzer verwendet, unter dem auf dem Managementserver das Ansible Playbook aufgerufen wird.

Dies sollte aber auf den Exadata-Cellservern vermieden werden, da an diesen nichts verändert werden darf. Für diesen Zweck ist es möglich, in Ansible bis auf Host-Ebene einen Benutzer zum Verbindungsaufbau zu definieren, der auf den Cellservern auf „root" gesetzt wird.

Damit die Exadata-Systeme optimal angepasst werden können, werden alle Hosts mit weiteren zusätzlichen Variablen versehen. Auf der Exadata ist es von Vorteil, die Hosts mit einer Variablen für den Typ des Host-Target zu versehen: Hierfür wird die Variable „host_type" definiert, die als Ausprägung „dom0" (Hypervisor), „domU" (VM), „cellserver" und „infiniband" kennt. Darüber kann in den Playbooks im „when"-Teil eines Tasks zum Beispiel eine Aktion auf den VMs ausgeführt werden, auf den Compute Nodes jedoch nicht.

Eine weitere wichtige Variable, „single_exec", sollte ebenfalls nicht fehlen: Diese wird zur Steuerung von Tasks verwendet, die nur auf einem Knoten ausgeführt werden dürfen.

Zusätzlich zur Inventory-Datei und zu den dort definierten Variablen können

```
Wallet Datei anlegen wallet.yml
---
  my_secret_pw: 'SuperGeheimesPw45#!'

Datei wallet.yml verschlüsseln:
  ansible-vault create wallet.yml
     New Vault password:
     Confirm New Vault password:

Verschlüsselte Ergebnisdatei:
cat wallet.yml

$ANSIBLE_VAULT;1.1;AES256
65366332643937373761373661623430656565303735313232373630303739343165633761376434653761346436356362633232643031
65353638653732663330a62623064313934643532393235303130346664656137383865626431336532353338643333303066653932616132
32616139303464376161376561373537373537 0a6633326666373865656231623837663937636563363462346230346534336435363763
30363166333139636531383232323561303531346632346638383662636537383566633930376461363030323761613636373062333833
313030343232

Wallet nachträglich editieren:

ansible-vault edit wallet.yml [--vault-passwordfile=<file_with_my_unencrypted_pw> ]
```

Listing 3: Automatische Entschlüsselung

über eine Ordnerstruktur von Gruppen und Hosts weitere Dateien mit Variablen zur besseren Übersicht verwendet werden. Dabei erweist sich für Exadata-Systeme die Verwendung von in der Gruppenstruktur in Dateien abgelegten Variablen als sinnvoll, da meist mehrere Komponenten die gleichen Einstellungen teilen und somit gleiche Einstellungen nicht n-mal bei jedem Host definiert werden müssen. Zu beachten gilt, dass die Gruppen- und Host-Verzeichnisstruktur parallel zur Inventory-Datei liegen muss *(siehe Listing 2)*.

Will man Passwörter ebenfalls im Inventory speichern, zum Beispiel für die Anbindung der Exadata an zentrale LDAP-Server, so können diese darin verschlüsselt abgelegt werden. Erstellt werden die Passwörter mit dem Utility „ansible-vault“, das ebenfalls in der Standard-Installation von Ansible mitgeliefert wird.

```
ansible.cfg
 [default]
 inventory = /home/ansible/exadata_inventory
 host_key_checking = false
 timeout = 120
 …
```

Listing 4: „host_key_checking“-Parameter

Der Ablauf ist wie folgt: Zuerst wird die Datei mit den Variablen und Passwörtern unverschlüsselt im Gruppenverzeichnis des Inventory abgelegt. Danach wird sie verschlüsselt. Die Variablen und Passwörter wiederum stehen den Playbooks zur Laufzeit zur Verfügung und werden beim Ausführen automatisch entschlüsselt, sofern dem Ansible-Aufruf die Datei mit dem Schlüssel zum Entschlüsseln mitgegeben wird *(siehe Listing 3)*.

Alternativ zur gesamten verschlüsselten Datei kann auch ein String einer Variable einzeln via „ansible-vault encrypt_string ‘my_secret_PW‘ -name ‘my_pw_var‘ “ verschlüsselt werden. Dabei steht der String hinter „-name“ für den Variablennamen. Der verschlüsselte String kann ebenfalls im Inventory abgelegt werden.

```
Aufruf:
ansible-playbook -i exadata_inventory deploy_ansible_user.yml --user
root --ask-pass -limit exa01

   tasks:
   - name: Gruppe anlegen
   group:
       name: ansible
       gid: 13246
       state: present
   become: true

   - name: Benutzer anlegen
   user:
       name: ansible
       comment: "ansible deployment user"
       uid: 28224
       shell: /bin/bash
       group: "ansible"
       state: present
   become: true

- name: authorized_keys kopieren
   copy:
       src: files/ansible_authorized_keys
       dest: "/home/ansible/.ssh/authorized_keys"
       owner: ansible
       group: ansible
       mode: 0400
   become: true

- name: sudoers kopieren
   copy:
       src: ansible_sudoers
       dest: "/etc/sudoers.d/ansible"
       owner: root
       group: root
       mode: 0440
   become: true
```

Listing 5: Zusätzlich muss beim Aufruf der Benutzername „root“ einmalig mitgegeben werden

Ansible-Konfiguration

Eine weitere wichtige Konfigurationsdatei ist „ansible.cfg“. Diese kann unter „/etc/ansible“, im Homedirectory als „.ansible“ oder im Verzeichnis des Playbooks als „ansible.cfg“ abgelegt werden und Parameter für die Laufzeit und Umgebung enthalten. Ein wichtiger Parameter ist der Timeout-Parameter, der den SSH-Connect-Timeout definiert. Die Erhöhung des Connect-Timeouts verhindert, dass Ansible bei längerer Verbindungsaufbauzeit zu frühzeitig aufgibt und die Aktionen nicht ausführt.

Ein weiterer wichtiger Parameter ist der „host_key_checking“-Parameter, der mit „false“ eine Überprüfung des Keys während des Verbindungsaufbaus nicht durchführt. Beide Parameter haben für Exadatas bei der Erstinstallation eine Bedeutung: Ist „host_key_checking“ auf „true“, so muss jeder gelernte Key manuell beim ersten Connect bestätigt werden *(siehe Beispiel ansible.cfg in Listing 4)*.

Use Cases

Nachdem in den vorangegangen Abschnitten die Grundlagen dargestellt wurden, geht es jetzt um die Use Cases für die Exadata. Ein Ziel der ganzen Automatisierung ist es, manuelle Schritte in reproduzierbaren Playbooks abzubilden und schlussendlich die einzelnen Playbooks durch ein „Überplaybook“ aneinanderzureihen. Dieses kann alle Schritte zur Einpassung neuer Exadatas oder Erweiterungen von Exadatas um Compute, VMs und Cellserver enthalten, um diese schnell in die eigene IT-Infrastruktur einzubinden.

```
tasks:
- name: rpcbind service starten
   service:
       name: rpcbind
       state: started
       enabled: yes
   become: true
   notify:        <-- Benachrichtigung des Handlers
       - run host_access_control get-runtime

handlers:
- name: run host_access_control get-runtime
   command: /opt/oracle.cellos/host_access_control get-runtime
   become: true
```

Listing 6: Aufruf des Handler

```
   - name: kopieren und entpacken des tfa installers
       unarchive:
           src: installTFA.zip
           dest: "/tmp/"
       become: true
       when:
           - single_exec == "yes"

   - name: Install oder Upgrade TFA
       command: "/tmp/installTFA -tfabase {{tfa_base}} -noorachk -silent"
       become: true
       when:
           - single_exec == "yes"
```

Listing 7: Vermeidung des „orachk"-Laufs bei Installation und Upgrade durch den Parameter -noorachk

Für dieses Ziel werden die Aufgaben auf mehrere Playbooks verteilt, sodass zusammenhängende Einstellungen zusammen, jedoch jede Aufgabe für sich allein ausgeführt und einzeln wartbar bleibt.

Für die Erreichung dieses Ziels ist als erster Schritt das Henne-Ei-Problem zu lösen: Es soll eine neue Exadata mit Ansible verwaltet werden. Dafür ist es nötig, den Ansible-Deployment-Benutzer initial auf der Exadata anzulegen sowie für die Cellserver nur den SSH-Key in den Root-Benutzer zu verteilen. Voraussetzung dafür ist – wie für alle Playbooks – ein Inventory, in dem die Exadata-Komponenten bereits definiert sind. Die Herausforderung für dieses Playbook ist, dass primär der Ansible-Benutzer nicht existiert und eine Verbindung initial als „root" ausgeführt werden muss, damit der Benutzer angelegt wird. Ansible wird normalerweise ohne Benutzereingaben ausgeführt. Für diesen Fall hier ist aber zwingend die Eingabe des Root-Passworts nötig. Dies erreicht man mit dem Parameter „--ask-pass". Zusätzlich muss beim Aufruf der Benutzername „root" einmalig mitgegeben werden, der hier abweichend von dem definierten im Inventory ist *(siehe Listing 5)*. Der Aufruf setzt sich immer aus dem Programm „ansible-playbook", der Spezifizierung des Inventory „-i <inventory-file>", dem Namen des Playbooks und eventuell einer Eingrenzung der Hosts, auf denen das Playbook ausgeführt wird, „--limit <Host|Gruppe>", zusammen.

Inhaltlich führt das Playbook folgende Aktionen aus:

- Benutzer und Gruppe anlegen
- authorized_keys mit SSH-Key verteilen
- sudoers-Berechtigungen vergeben

Nach Anpassung der Systemkonfiguration sollte am Ende des Playbooks das Kommando „host_access_control get-runtime" aufgerufen werden. Dadurch wird verhindert, dass nach einem Neustart des Hosts einige der geänderten Werte wieder durch einen Exadata Default überschrieben werden.

Dies trifft zum Beispiel auf Daemon-Prozesse zu, die standardmäßig nicht gestartet sind. Des Weiteren werden auch Werte bezüglich des Passwort-Ablaufs von vorhandenen oder neu angelegten Benutzern mit dem Aufruf dieses Programms permanent festgeschrieben.

Am Beispiel von NFS soll dies verdeutlicht werden: Ein weiteres Playbook beinhaltet den Task, den „rpcbind"-Daemon für eine reibungslose NFS-Funktion zu aktivieren. Am Ende des Playbooks wird eine Aktion über den sogenannten „Handler" aufgerufen, der die Änderungen permanent macht *(siehe Listing 6)*. Der Trick hierbei ist, dass der Task, der eine permanente Änderung benötigt, eine Benachrichtigung an den Handler-Dienst schickt, um eine bestimmte Aktion auszuführen. Dies können auch mehrere Tasks tun, der angesprochene Handler wird erst am Ende einmal ausgeführt.

Ein weiterer Use Case ist die Installation oder das Upgrade des TFA. Dafür eignet sich Ansible hervorragend. TFA besteht aus einer Setup-Datei von Oracle, wofür es kein fertiges Ansible-Modul gibt. Ist dies nicht der Fall, kann auf eines der Ansible-Standardmodule zurückgegriffen werden. Das „Command"-Modul ist hier primär die erste Wahl: In einer Shell führt dieses ein einfaches Kommando mit Parametern aus.

Im resultierenden Playbook wird zuerst die neueste Version des TFA gepackt als ZIP-Datei auf die Exadata kopiert und extrahiert. TFA wird im Falle einer RAC-Umgebung auf allen Knoten installiert und ausgeführt. Ein Upgrade des TFA wird in der Regel nur auf einem Knoten angestoßen. Der Installer selbst sorgt für das Upgrade der weiteren Knoten. Diese Besonderheit sollte sich auch im Ansible Playbook widerspiegeln. Durch die Kontrollstruktur „when" in Kombination mit der Inventory-Variablen „singel_exec=yes" kann dieses Ziel erreicht werden.

```
   - name: set vm.nr_hugepages in /etc/sysctl.conf
   sysctl:
       name: vm.nr_hugepages
       value: 131072
       sysctl_file: /etc/sysctl.conf
       reload: yes
   become: true
   when: host_type == "domU"
```

Listing 8: Anpassung eines Parameters im Inventory auf der Exadata

Wichtig hierbei ist die Ausführung des TFA-Installers als Root-Benutzer, was durch den Parameter „become=true“ forciert wird.

Ein weiterer wichtiger Setup-Parameter des TFA ist die Vermeidung des „orachk“-Laufs bei Installation und Upgrade. Wird dieser nicht mitgegeben, so wird für jede Datenbank ein Report erzeugt, was zu einer unnötig langen Gesamtlaufzeit des Playbooks führt. Dies lässt sich durch den Parameter „-nooorachk“ abschalten *(siehe Listing 7)*.

Wer kennt es nicht, die ständigen Anpassungen der Huge Pages auf Linux-Systemen für Oracle-Datenbanken, Gleiches gilt für die Exadata. Dies kann Ansible mit simplen Mitteln für alle Datenbankknoten lösen:

Am besten ist es, den Huge-Page-Wert im Inventory zentral pro virtualisierter oder physikalischer Exadata abzulegen und zu pflegen.

Bei jeder Ausführung des Tasks wird die Konfiguration aus dem Inventory ausgelesen und mit dem aktuellen Wert des Servers verglichen. Bei Abweichung oder Anpassung des Parameters im Inventory wird dieser auf den Exadatas entsprechend angepasst *(siehe Listing 8)*. Der Parameter „reload=yes“ führt dazu, dass der neue Wert nicht nur in die Konfigurationsdatei sysctl.conf geschrieben, sondern auch gleich aktiviert wird. Wird eine virtualisierte Exadata eingesetzt, so kann durch den Vergleich des Inventory-Parameters „host_type“ gesteuert werden, welche Anpassung mit welchem Parameter auf welchem Server stattfinden soll. Denn der Wert ist nur auf der Komponente sinnvoll anzuwenden, auf dem die Oracle-Datenbanken laufen.

Fazit

Beginnt man erst einmal, manuelle Tätigkeiten für die Exadata zu automatisieren, und schreibt das Ganze über die Zeit fort, so erhält man am Ende eine Sammlung von Playbooks, die nicht nur für die bereits im Betrieb befindlichen Exadatas von Vorteil sind, sondern auch für neue Exadatas oder neue Exadata-Komponenten. Diese können dadurch schneller in Betrieb genommen werden. Am Ende entsteht ein Ablauf, der die einzelnen Schritte und Playbooks aneinanderreiht und eine neue/aufgerüstete Exadata schnell mit den nötigen Anpassungen versorgt.

Gleichzeitig reduziert es die Fehleranfälligkeit sowie das Vergessen von Anpassungsschritten auf den einzelnen Exadata-Komponenten.

In diesem Sinne ist der Einsatz von Ansible ein Gewinn für die Verwaltung der Exadata-Plattform und man kann zu Recht von einer Symbiose sprechen.

Timo Giese
timo.giese@fiduciagad.de

Patching Oracle - Was muss ich tun?

Martin Bracher, Trivadis

CPU, SPU, PSU, BP, RU, RUR, Patchset, One-Off?
Wenn es ums Patchen von Oracle geht, wird man schnell auf diese Abkürzungen treffen. Aber was bedeuten sie? Bei welchem Release muss beziehungsweise kann ich welche Art von Patch installieren? Was sind die Unterschiede? Wo finde ich diese Patches und mit welchen Tools muss ich sie installieren? Dieser Vortrag gibt einen Überblick über die einzelnen Patch-Arten mit ihrem Umfang und den Release-Zyklen sowie der Versionsabhängigkeit. Dann wird gezeigt, wo man die aktuelle Version des gewünschten Patches findet und wie man diese installiert (opatch, runInstaller). Beim Patchen der Datenbank wird auch kurz auf das Konzept des „Rolling Upgrade“ mit Standby-Datenbanken eingegangen.

Zuerst schauen wir uns an, welche Arten von Patches es überhaupt gibt:

Interim Patches

Diese Patches werden auch als „One-Off“ Patches bezeichnet. Sie enthalten nur einen Fix für ein ganz bestimmtes Problem und werden spezifisch für Kunden mit diesem Problem hergestellt. Wie dieser Patch einzuspielen ist, ist der beiliegenden Beschreibung zu entnehmen *(siehe Listing 1)*.

Die Versionsnummer verändert sich bei einem Interim Patch nicht.

```
Patch 13448206      : applied on Tue Apr 03 22:19:47 CEST 2012
Unique Patch ID: 14515642
   Created on 30 Jan 2012, 12:05:49 hrs PST8PDT
   Bugs fixed:
      13448206
```

Listing 1: Interim Patch

Sammlung von Patches

Immer ungefähr Mitte Januar, April, Juli und Oktober gibt Oracle ein „Bundle“ von

```
Patch 28655784       : applied on Sun Oct 28 00:54:52 CEST 2018
Unique Patch ID: 22509982
Patch description: "Database Release Update : 18.4.0.0.181016 (28655784)"
   Created on 8 Oct 2018, 21:27:28 hrs PST8PDT
   Bugs fixed:
      28571483, 9062315, 13554903, 14221306, 20436508, ...
```

Listing 2: Patch-Bundle mit mehreren Patches (Bugs fixed)

Patches raus, zu Deutsch eine Sammlung oder Zusammenstellung von einzelnen Patches. Je nach Typ des Bundle beinhaltet es alle oder Teile der bisher angefallenen Interim Patches, unveröffentlichte Patches und gegebenenfalls auch Funktions- oder Optimizer-Erweiterungen *(siehe Listing 2)*.

Die Bundles sind kumulativ, das heißt, wenn man das Bundle vom Juli 2019 installiert, sind auch die Patches vom April 2019 und vorher enthalten; es braucht also nur die aktuellste Version eingespielt zu werden.

Die Bundles durchlaufen einen vollständigen Regressions-Test.

Nun zu den einzelnen Bundles: CPU, SPU, PSU, BP, RU, RUR? Wenn Sie bei diesem Wildwuchs den Überblick verloren haben, geht es Ihnen wie mir, bevor ich einen Vortrag zu dem Thema halten durfte.

Critical Patch Update (CPU)

Dieses Bundle gibt es seit 2005 und enthält nur Security-Patches. Es ist nur erhältlich für Versionen bis und mit 11g. Nicht sicherheitsrelevante Bugs werden also mit einem CPU nicht behoben.

Security Patch Update (SPU)

Oracle hat das CPU in den sprechenderen Namen SPU umbenannt. Geplant wäre das ab Version 12.1 gewesen, aber das SPU kam nie raus. Ab 12.1 gibt es nur noch das umfangreichere PSU, auf das wir sogleich eingehen.

Patch Set Update (PSU)

Das Patch Set Update enthält die Security Patches des CPU und zusätzlich weitere Korrekturen von nicht sicherheitsrelevanten Fehlern, jedoch keine Änderungen am Optimizer. Das PSU ist auch noch für Version 12 erhältlich.

Ein PSU ändert die 5. Stelle der Versionsnummer; seit Januar 2016 ist die 5. Stelle das Release-Datum des PSU im Format YYMMDD, beispielsweise 12.2.0.1.181016.

Database Proactive Bundle Patches (BP oder DBBP)

Der BP enthält die Korrekturen vom PSU und kann zusätzlich funktionale Anpassungen und Optimizer-Änderungen beinhalten. Ein BP erfordert also einen etwas höheren Testaufwand, da sich das Verhalten der Applikation ändern könnte.

Der BP ist inkompatibel mit dem PSU. Falls bereits ein PSU installiert ist, muss dieses vorgängig deinstalliert werden.

Der BP ist nur für Release 12.2 verfügbar.

Release Update (RU)

Diesen Patch-Typ gibt es seit Juli 2017 und ist erhältlich für Release 12.2 und höher.

Er ist inhaltlich vergleichbar mit dem BP und löst diesen ab. Die erste Stelle entspricht dem Erscheinungsjahr des initialen RU (z.B. 18.1 im Jahr 2018), die zweite dem Quartal des RU, wobei Oracle da inkonsequent nummeriert hat. Im Jahr 2018 stand die 1 für Quartal 1, im Jahr 2019 bekam das Quartal 1 jedoch die Nummer 2 (die 1 wurde für die Beta-Version verwendet) *(siehe Abbildung 1)*.

Release Update Revision (RUR)

Dies enthält die Fixes des RU, jedoch ohne Feature- und Optimizer-Änderungen. In der Versionsnummer ändert sich die 3. Stelle. Beispiel: Ein RUR 18.2.1 entspricht dem RU 18.3 ohne die Feature- und Optimizer-Änderungen. Ein RU ist also ähnlich einem PSU.

Es gibt nur 2 RURs für Nicht-Terminal-Releases; danach muss man auf ein RU wechseln.

Patchset oder initiales RU YY.1.0

Seit Oracle 11.2 handelt es sich beim Patchset installationstechnisch um einen vollständigen Release, der in ein neues ORACLE_HOME installiert werden muss. Beim Installationsprozess gibt es also keinen Unterschied, ob man nun ein Upgrade auf eine neue Version macht (z.B. 11.2.0.4 auf 12.1.0.1) oder ein Patchset installiert (z.B. 12.1.0.1 auf 12.1.0.2). Das Patchset ändert den 4. Teil der Versionsnummer (z.B. 11.2.0.4).

Das erste Release-Update eines Jahres (18.1, 19.2) entspricht einem Patchset. Installationstechnischer Unterschied zum Patchset: Die Software wird nicht mehr temporär entpackt und von dort installiert, sondern direkt im neuen ORACLE_HOME entpackt und dann dort konfiguriert (ebenfalls mit dem runInstaller).

Noch eine kleine Besonderheit wegen Oracles Cloud-Strategie: Da die On-Premises-Version erst ein bis zwei Quartale

	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
RU	18.1.0	18.2.0	18.3.0	18.4.0	18.5.0	18.6.0		
RUR1			18.2.1	18.3.1	18.4.1	18.5.1		
RUR2				18.2.2	18.3.2	18.4.2		
RU					19.2.0	19.3.0	19.4.0	19.5.0
RUR1							19.3.1	19.4.1
RUR2								19.3.2

Abbildung 1: Release-Zyklen ab dem 1. Quartal 2018 (Quelle: Martin Bracher)

später verfügbar wird, gibt es dafür einen eigenen initialen Release mit einer höheren Nummer: 18.3.0 und 19.3.0.

Abbildung 2 fasst nochmals die Verfügbarkeit der verschiedenen Bundled Patches zusammen:

- CPU wurde umbenannt zu SPU
- BP ist mit RU vergleichbar
- PSU ist mit RUR vergleichbar
- Patchset ist mit initialem RU vergleichbar

In den folgenden Kapiteln widmen wir uns nun der Frage, welche Tools es zum Patchen gibt.

Oracle Universal Installer (OUI)

Beim OUI handelt es sich um eine Java-Applikation. Mit ihm werden Grundreleases sowie Patchsets und initiale RUs installiert. Vor der Installation prüft der OUI sehr genau, ob die Voraussetzungen erfüllt sind. Bei gewissen fehlenden Voraussetzungen kann er ein Korrektur-Skript erzeugen, das man dann nur noch mit Root-Privilegien ausführen muss. Der OUI ist auch in der Lage, die Software in einem Cluster (Grid Infrastructure, RAC) zu installieren.

Die Bedienung erfolgt interaktiv als grafische Applikation oder aber, für Automatisierung interessant, im Textmodus mit einem Response-File *(siehe Listing 3)*.

OPatch

Alle anderen Arten von Patches werden mit „OPatch" installiert. Auch hierbei handelt es sich um eine Java-Applikation, die jedoch nur im Textmodus arbeitet. OPatch befindet sich im ORACLE_HOME im OPatch/-Verzeichnis *(siehe Listing 4)*. Oft muss jedoch vor dem Einspielen eines Patches zuerst die Version von OPatch aktualisiert werden. Die aktuelle Version findet man immer unter der Patch-Nummer 6880880 oder unter folgender URL: *https://updates.oracle.com/download/6880880.html*

Aktualisiert wird OPatch, indem man das Zip-File im ORACLE_HOME entpackt und alle Dateien überschreibt. Vorsicht: Es funktioniert nicht, wenn man es an einem anderen Ort entpackt und von dort startet!

	CPU	SPU	PSU	RUR	BP	RU	Patchset
11.1	■		■				■
11.2	■		■				■
12.1		n/a	■		■		■
12.2				■		■	(18.1.0)
18				■		■	(19.2.0)
19				■		■	

Abbildung 2: Verfügbarkeit der Bundled Patches (Quelle: Martin Bracher)

```
./runInstaller -responseFile /home/oracle/db.rsp -silent
```

Listing 3: Installation mit Response-FileMit der Option „--help" werden sämtliche möglichen Parameter angezeigt.

```
$ORACLE_HOME/OPatch/opatch version
OPatch Version: 12.2.0.1.14
OPatch succeeded.
```

Listing 4: OPatch-Version

```
opatch query -all /tmp/15994107/
Oracle Interim Patch Installer version 12.1.0.1.3
Copyright (c) 2017, Oracle Corporation. All rights reserved.
[..]
Patch created on 19 Dec 2013, 08:29:40 hrs PST8PDT
Need to shutdown Oracle instances: true
Patch is roll-backable: true
Patch is a "Patchset Update": false
Patch is a rolling patch: true
Patch is a sql patch: false
Patch has sql related actions: false
Patch is an online patch: false
Patch is a portal patch: false
Patch is an "auto-enabled" patch: false
[..]
List of platforms supported:
226: Linux x86-64
List of bugs to be fixed:
15994107: LNX64-12.1 ORA-600 [KFCDEADLOCKAVOID01] DURING DELETE FILES IN ASMCMD
[..]
```

Listing 5: Anzeige von Patch-Informationen

```
opatch prereq CheckConflictAgainstOH \
> -phBaseDir /u00/app/oracle/stage/15994107
```

Listing 6: Prüfung der Patch-Verträglichkeit

```
opatch rollback -id 12345678
```

Listing 7: Rollback im Unterverzeichnis

```
opatch lspatches
opatch lsinventory
```

Listing 8: Befehl zur Anzeige bereits installierter Patches

```
cd <patch_location>/<patch_number>
$ORACLE_HOME/OPatch/opatch apply
```

Listing 9: Befehl zur Patch-Installation

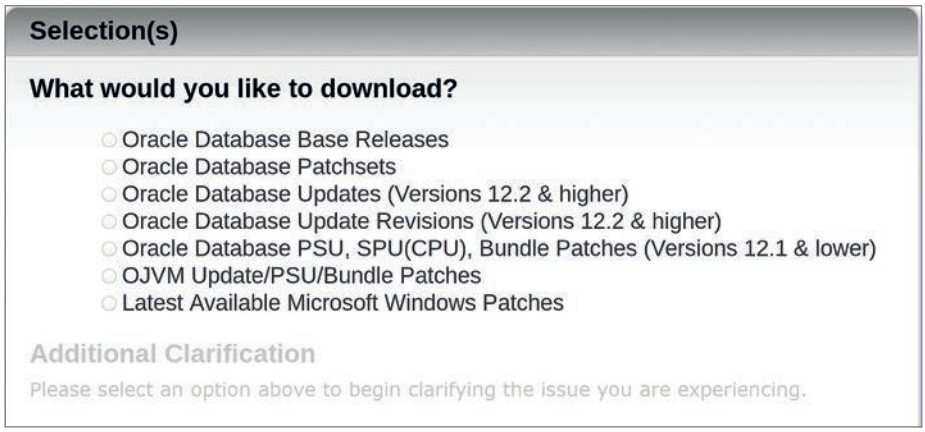


Abbildung 3: Auswahl des Patch-Bundle (Quelle: Martin Bracher)

Abbildung 4: Auswahl der Version (Quelle: Martin Bracher)

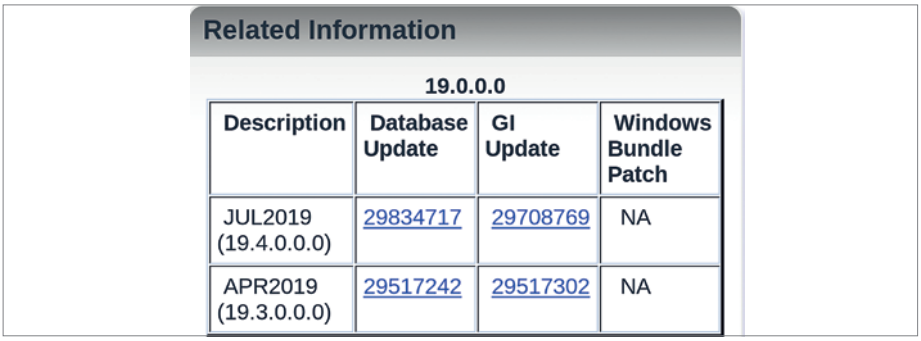


Abbildung 5: Links zu den Patches (Quelle: Martin Bracher)

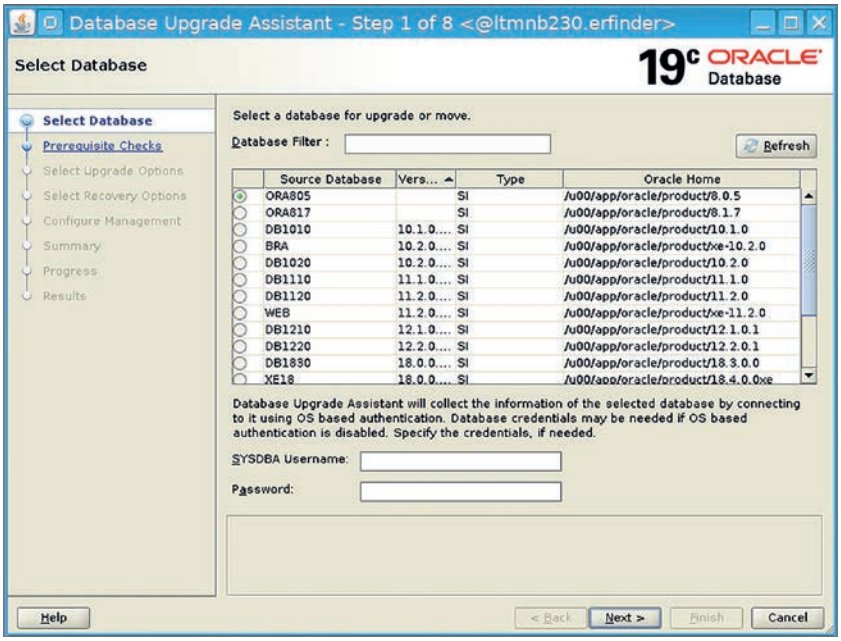


Abbildung 6: Database Upgrade Assistant (Quelle: Martin Bracher)

Bevor man einen Patch einspielt, sollte man jeweils zuerst das beiliegende Readme lesen. OPatch selbst kann auch noch Informationen über den Patch anzeigen *(siehe Listing 5)*.

Ebenso sollte man prüfen, ob sich der Patch mit anderen, schon installierten Patches verträgt *(siehe Listing 6)*.

Im Falle eines Konflikts kann der andere Patch deinstalliert werden. Das ist möglich, weil Oracle den alten Zustand im Unterverzeichnis .patch_storage/ abgelegt hat *(siehe Listing 7)*. Eine Übersicht beziehungsweise eine detaillierte Anzeige der schon installierten Patches bekommt man mit folgenden Befehlen *(siehe Listing 8)*.

Der Patch selbst wird dann mit „opatch apply" installiert *(siehe Listing 9)*.

Hinweis: Bei einem Cluster mit der Grid-Infrastructure kann auch der Befehl „opatchauto" verwendet werden, der die einzelnen Schritte von root und oracle/grid automatisiert.

Woher bekomme ich die Patches

Die Patchsets für die Versionen 9.2 – 12.1 findet man bei support.oracle.com unter dem Tab „Patches & Updates". Dort gibt es den Link „Latest Patchsets".

PSUs und andere empfohlene Patches findet man unter der ID 756671.1.

Der bisher wenig bekannte Patch Download Assistant bietet die Links zu den Patch-Bundles von 8.1 bis zur neusten Version an sowie auch die Grund-Releases ab 11.2; man erreicht ihn unter folgender URL: *https://support.oracle.com/epmos/faces/DocumentDisplay?id=2118136.2*

Dieser Assistent führt uns Schritt für Schritt zum Ziel. Zuerst wählt man *(siehe Abbildung 3)* die gewünschte Patch-Art (die RUs heißen hier „Oracle Database Updates").

Danach wählt man die Version aus *(siehe Abbildung 4)* und bekommt die Links zu den entsprechenden Patches angezeigt *(siehe Abbildung 5)*.

In den nächsten Kapiteln schauen wir uns an, wie die Datenbank gepatcht wird.

Patchset und initiales RU

Beim Upgrade auf ein neues Patchset oder ein initiales RU erfolgt das Upgrade über

den **Database Upgrade Assistant (DBUA)**. Der DBUA, ebenfalls eine Java-Applikation, wird aus dem neuen ORACLE-HOME gestartet. Danach wählt man die zu migrierende Datenbank aus *(siehe Abbildung 6)*.

Nun schlägt er vor, einen garantierten Restore-Point zu erzeugen. Dies sollte man tun. Falls das Upgrade fehlschlägt, kann er die Datenbank selbständig wieder auf den ursprünglichen Stand zurücksetzen. Falls dies passiert, sollte man zuerst abklären, welche der installierten Optionen von der Applikation überhaupt gebraucht werden, und dann alle nicht benötigten Optionen (z.B. Java, XML) deinstallieren. Wenn das auch nicht hilft, deinstalliert man alle Optionen, macht das Update und installiert die Optionen anschließend wieder.

Seit Kurzem gibt es noch ein neues Tool von Oracle, das „AutoUpgrade" (Doc ID 2485457.1). Bei größeren Datenbank-Umgebungen ist dies sicher interessant für die Automatisierung.

One-Off Patches

Bei den One-Off Patches ist gemäß dem beiliegenden Readme.html vorzugehen.

Alle anderen Patches

Bei allen anderen Patch-Bundles ist abhängig von der Version einer der folgenden Befehle auszuführen. Bis Oracle 11g ist ein SQL-Skript im sqlplus zu starten *(siehe Listing 10)*:

Ab Oracle 12c existiert ein Shell-Skript, das gestartet werden muss. Bei einer Multitenant-Datenbank werden dabei gleich alle geöffneten PDBs gepatcht *(siehe Listing 11)*.

Patchen von Data-Guard-Umgebungen

Noch ein Hinweis zu Physical-Standby-Datenbanken (Data Guard): Gepatcht werden (so wie oben beschrieben) muss die Primary-Datenbank. Die Standby muss nur mit dem gepatchten ORACLE_HOME gestartet werden. Sie patcht sich selbst durch die Applizierung der Redo-Informationen.

Zum Schluss möchte ich noch kurz das Konzept des „Rolling Upgrade" vorstellen. Das Ziel ist die Reduktion der Downtime bei der Migration. Hierzu muss eine Logical-Standby-Datenbank verwendet werden. Im Gegensatz zur Physical Standby, die sich durch Applizierung von Redo aktualisiert, wird die Logical Standby durch Applizierung von SQL-Statements aktualisiert *(siehe Abbildung 7)*.

```
SQL> @catbundle.sql psu apply
```

Listing 10: Starten eines SQL-Skripts

```
# $ORACLE_HOME/OPatch/datapatch
```

Listing 11: Patchen aller geöffneten PDBs

Es handelt sich somit um eine eigenständige Datenbank, die Read/Write geöffnet ist. Der Trick ist nun, dass man die Standby-Datenbank auf die neue Version migriert, während man auf der Primary normal weiterarbeitet. Nach der Migration appliziert man die angefallenen Änderungen der Primary auf die Standby und macht einen Switchover auf die Logical Standby, die sich nun schon im gepatchten Zustand befindet. Die Downtime reduziert sich dabei auf die Zeit des Switchover. Dieses Verfahren sollte man jedoch nur anwenden, wenn man die Downtime unbedingt reduzieren muss und man sich gut mit Logical-Standby-Datenbanken

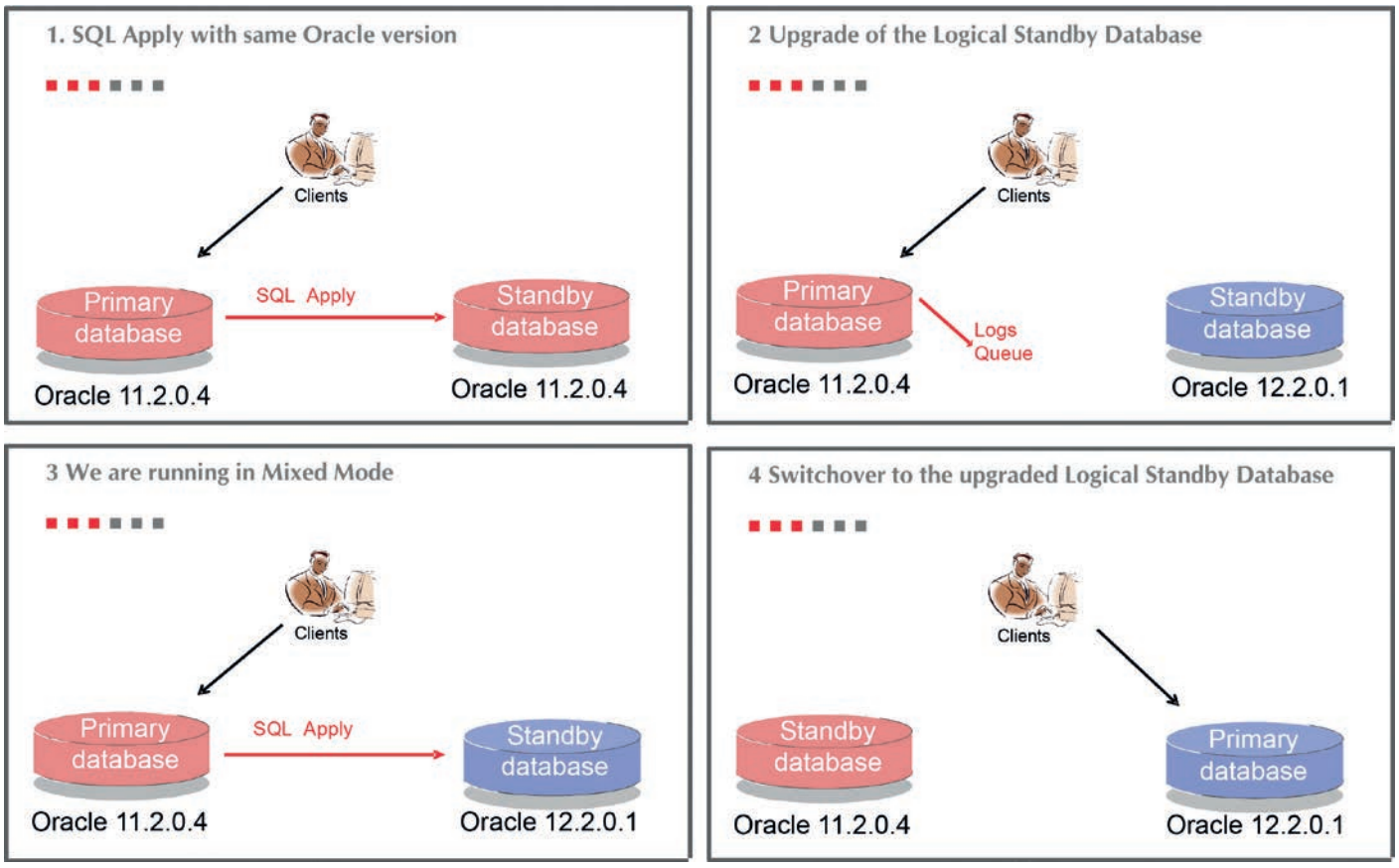


Abbildung 7: Upgrade mit Logical-Standby-Datenbanken (Quelle: Martin Bracher)

auskennt. Aufgrund von Einschränkungen bei Logical Standbys (z.B. unsupportete Datentypen) kann dieses Verfahren auch nicht immer zum Einsatz kommen.

Quellen

[1] *https://mikedietrichde.com/2017/10/24/differences-psu-bp-ru-rur/*
[2] *https://docs.oracle.com/cd/E24628_01/doc.121/e39376/opatch_overview.htm*
[3] *https://www.markusdba.de/?p=1312*

Über den Autor

Martin Bracher ist seit über 20 Jahren beruflich in der Informatik tätig, davon über 15 Jahre bei Trivadis. Nach dem Informatikstudium an der Universität Bern erfolgte ab 1997 der Umstieg in die Oracle-Welt. Er war als Analytiker, Software Engineer und DBA in verschiedenen Oracle-Projekten in verschiedensten Branchen tätig. Wegen seiner sehr guten Kenntnisse in Unix/Linux wird er gern bei konzeptioneller Beratung, Installation und Konfiguration eingesetzt. Zudem hat er große Erfahrung im Bereich Automatisierung/Schnittstellen zwischen Operating Systems (Unix), Datenbank und der Entwicklung. Er ist auch Spezialist in Sachen Oracle-Hochverfügbarkeit (Data Guard, RAC). Bei der Trivadis-Toolbox-Software BasEnv zum Setzen von Datenbankumgebungen ist er der Hauptentwickler.

Martin Bracher
martin.bracher@trivadis.com

SQLTXPLAIN – Helfer bei der Performance-Analyse

Stefan Seck, Logicalis

Nicht jede Oracle-Datenbank ist im selbstfahrenden Modus, sodass kein Eingriff von außen nötig wäre. Ab und zu gibt es doch Performance-Probleme bei Datenbanken, die nicht unbedingt sofort ersichtlich sind oder bei denen ein automatisch erstellter Index direkt hilft. Um diese Probleme zu lösen, bedarf es eines strukturierten Vorgehens und des Einsatzes geeigneter Tools. Eines der möglichen Tools, SQLTXPLAIN, möchte ich hier in Grundzügen vorstellen.

Laufzeiten von SQL-Abfragen verändern sich. Meistens dauert eine Abfrage „länger als sonst", was dann zu einem Problem in der Applikation führt. Die Lösung dieser Performance-Probleme kann durchaus eine Herausforderung werden.

Hier bietet es sich an, ein entsprechendes Modell bereitzuhaben, das sich zu jeder Zeit abarbeiten lässt *(siehe Abbildung 1)*.

Für mich sollte es zumindest folgende Punkte beinhalten:

Vorgehensmodell

1. Problem definieren

Zur Findung einer Lösung stehen am Anfang einige Fragen:

- Was ist seit wann wie langsam?
- Welcher Prozess oder welche Statements sind betroffen?
- Gab es Änderungen an der Applikation / am System?

Es ist wichtig, sich so genau wie möglich ein Bild davon zu machen, was aus Anwendersicht nicht so läuft, wie es normalerweise laufen sollte. Im Fokus steht hier also erst einmal das Erkennen eines Problems.

2. Ziele definieren

Zu einem wichtigen Teil dieses Prozesses gehört es ebenso, dass Ziele geklärt werden. Aus Anwendersicht soll sich die Applikation / das System wieder genauso ver-

halten wie vorher. Um das zu erreichen, sollte natürlich klar sein, wie „genauso" definiert ist. Es muss geklärt sein, wann das Performance-Problem gelöst ist.

Bekanntlich führen mehrere Wege zur Lösung eines Problems. An dieser Stelle stellen sich nun die nächsten Fragen.

- Ist das Problem reproduzierbar?
- Gibt es eine Test-Umgebung, die wir für Tests nutzen dürfen?
- Dürfen wir die Datenbank durchstarten?
- Welche Tools können wir nutzen?
- Dürfen wir die Statements anpassen oder ändern, vielleicht auch Indizes hinzufügen?

3. Daten sammeln

Sobald der grundsätzliche Rahmen geklärt ist, geht es daran, Daten zu sammeln. Um ein Performance-Problem in einer Datenbank anzugehen, werden Statistiken und Metriken benötigt, die wir analysieren können. Es gibt verschiedene Quellen, die wir dazu nutzen können.

- Statspack Reports
  Statspack muss in der Datenbank installiert und konfiguriert sein, um es nutzen zu können (Installing and Configuring StatsPack Package (MOS 149113.1)). Es muss zusätzlich ein regelmäßiger Job eingerichtet sein, der per Snapshot relevante Daten aus dem System sichert. Statspack benötigt dazu keine zusätzliche Diagnostic-Pack-Lizenz, weil die Daten für die Snapshots aus den aktuellen Systemstatistiken der Datenbank kommen.

- AWR-Reports
  Mit der Version 10g hat Oracle das Automatic Workload Repository (AWR) eingeführt. Das AWR sammelt per Default alle 60 Minuten Systemstatistiken und speichert sie in eigenen Tabellen. Um das AWR für die Lösung von Performance-Problemen zu nutzen, also die gespeicherten Daten abzufragen und zu analysieren, muss eine Diagnostic-Pack-Lizenz vorhanden sein. Damit ist auch klar, dass wir das AWR nicht in einer Standard-Edition- Datenbank nutzen dürfen. Der SQL Tuning Advisor nutzt unter anderem die vorhandenen Daten, um Vorschläge zu präsentieren. Hierfür ist eine Tuning-Pack-Lizenz nötig.

- ASH-Abfragen
  Neben dem AWR hat Oracle mit 10g auch die Active Session History (ASH) eingeführt. Mittels ASH können wir sehr granular herausbekommen, was eine bestimmte Session zu einer bestimmten Zeit getan hat, unter anderem wissen wir beispielsweise, welche Wait Events aufgekommen sind. Auch

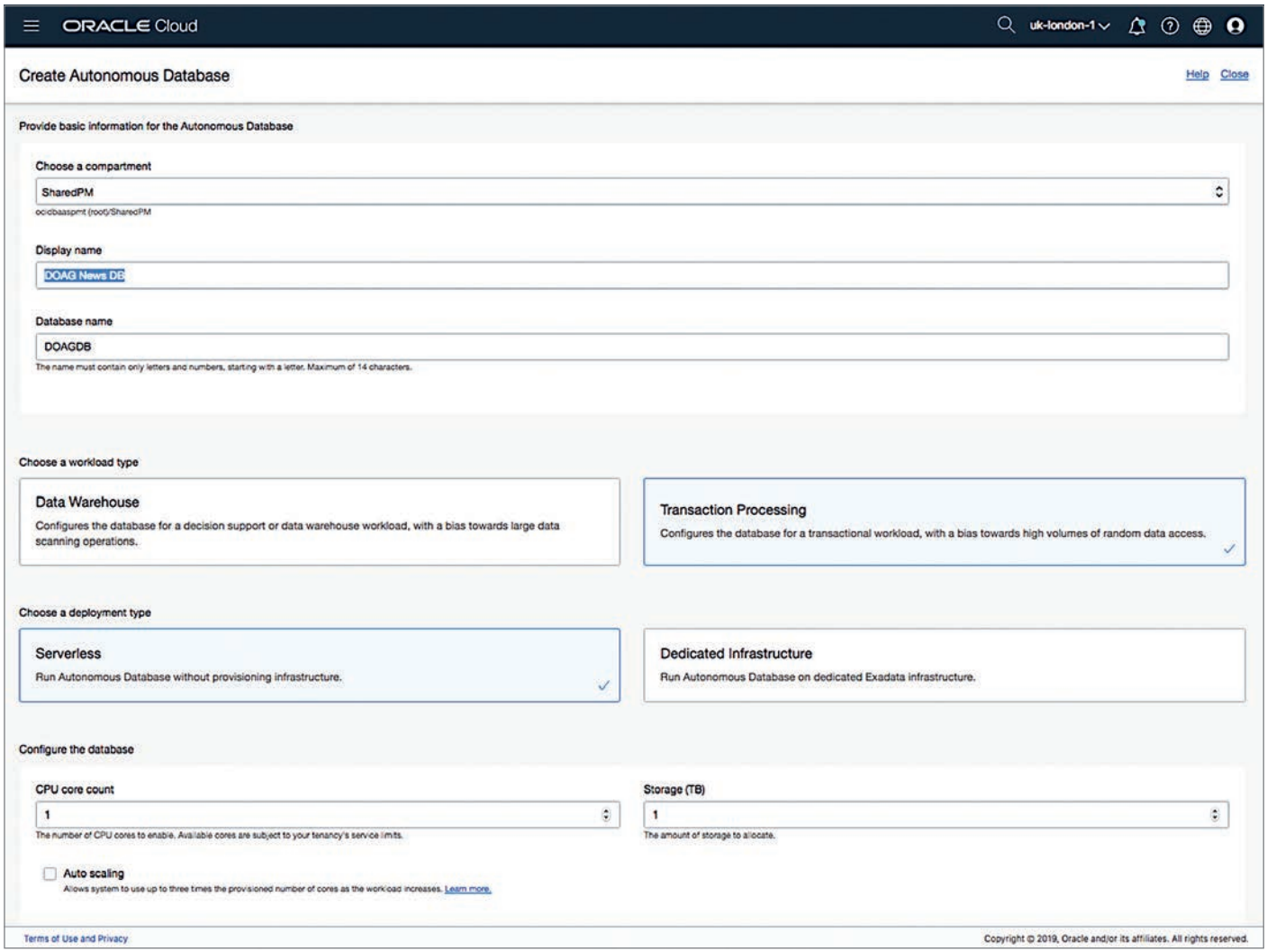


Abbildung 1: Vorgehensmodell (Quelle: Stefan Seck)

das ASH darf nur bei einer gültigen Diagnostic-Pack-Lizenz genutzt werden.

- SQL-Trace
  Natürlich besteht die Möglichkeit, jedes SQL in seinen einzelnen Schritten zu analysieren, indem ein Trace File erstellt wird. Dieses File kann durchaus groß werden, sodass auch zur Auswertung weitere Tools genutzt werden sollten, wie zum Beispiel tkprof, TRCA oder auch der SQL-Developer.

- Autotrace
  Wenn ein zu überprüfendes SQL wiederholt ausgeführt werden kann, so kann es sich durchaus anbieten, Autotracing einzuschalten. Sofort nach Ausführung des Statements werden der tatsächliche Ausführungsplan und weitere Informationen ausgegeben.

- SQLdb360
  SQLdb360 ist ein frei verwendbares Toolset, um eine erste Bewertung einer gesamten Oracle-Datenbank oder einer bestimmten SQL-Anweisung durchzuführen. SQLdb360 besteht aus zwei unabhängigen Werkzeugen, eDB360 (datenbankweite Analyse) und SQLd360 (individuelle SQL-Analyse). Für die Nutzung von SQLdb360 muss nichts in der zu analysierenden Datenbank installiert werden. SQLdb360 liefert als Output auch einen HTML-Report, mit dem alle relevanten Statistiken und Metriken vorhanden sind, um ein Performance-Problem beispielsweise eines SQL zu analysieren.

- SQLTXPLAIN
  Ein von Carlos Sierra entwickeltes Tool, das den SQL-Tuning-Prozess durch den Einsatz von automatisierten Skripten beziehungsweise Methoden beschleunigen kann.

Die Datenbank ist langsam	Ein Prozess / SQL ist langsam
Serverauslastung	Statements
I/O-Last	Objekte
Netzwerklast	Statistiken
Serverparameter	Ausführungspläne
Datenbankparameter	Datenbankparameter
Architektur / DB-Design	

Tabelle 1

4. Daten analysieren

Unabhängig von den Tools, der Herangehensweise und vom Datensammeln möchten wir klären, was das System ausbremst, sodass Anwender deutlich gestört werden.

Tabelle 1 dient als Beispiel dafür, welche Aspekte bei der Analyse und vor allem beim Sammeln der Daten berücksichtigt werden können.

5. Problem lösen

Zunächst ist Ruhe zu bewahren! Anschließend ist es gut, die unten stehenden Punkte, die keinen Anspruch auf Vollständigkeit erheben, abzuarbeiten. Dabei sollten nicht mehrere Dinge auf einmal geändert werden. Die gesammelten Daten und deren Analyse geben uns die entscheidenden Hinweise.

- Eventuelle Änderungen zurücknehmen
- Statement umschreiben / Prozess ändern
- Speicherzuteilung ändern (SGA / PGA / Log-Buffer)

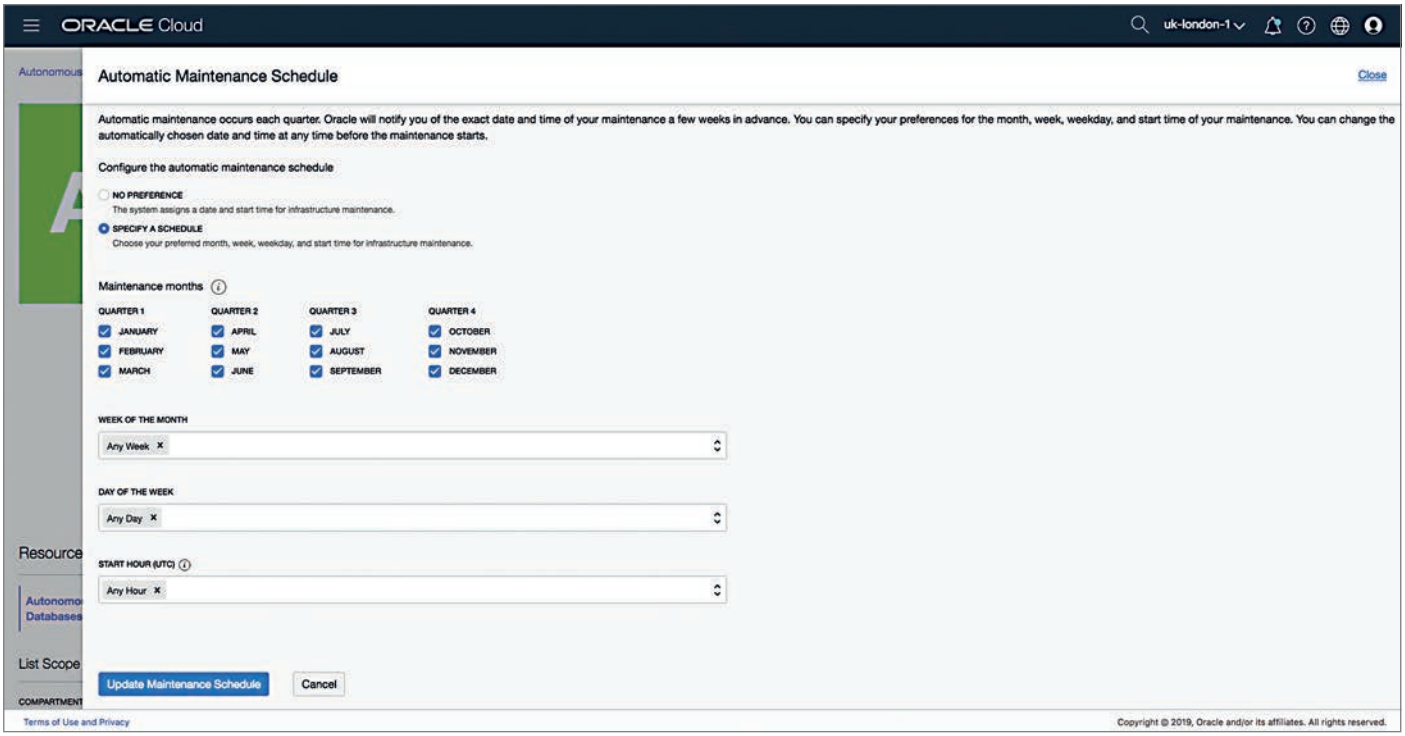


Abbildung 2: Verzeichnisse (DB-Server und/oder Client) (Quelle: Stefan Seck)

- Statistiken anpassen, Histogramme gezielt einsetzen
- Batchläufe verschieben
- Index anlegen, ändern oder löschen
- Optimizer-Parameter anpassen

SQLT – ein Überblick

SQLTXPLAIN (im Weiteren SQLT) wurde 2007 von Carlos Sierra entwickelt und kann von MOS heruntergeladen werden. SQLT ist geeignet, DBAs bei der Analyse von Performance-Problemen zu unterstützen.

Ein sinnvoller Einsatz von SQLTXPLAIN kann erst nach einer Gesamtanalyse der Datenbank / des Systems geschehen, in der die Last, Wait Events, Performance-Metriken und Weiteres betrachtet wurden. Zumeist zeigen sich dann ein paar wenige SQLs, die problematisch sind, weil sie vermeintlich zu viele Ressourcen benötigen. SQLT kommt also erst in einem späteren Stadium der Analyse ins Spiel. Natürlich gibt es auch Probleme mit und in einer Datenbank, bei denen SQLT nicht helfen kann.

Nach dem Entpacken der Skripte und der Installation in der Datenbank sammelt SQLT Daten zu einem SQL mit allen Punkten, die für die Ausführung relevant sind, und schreibt nicht nur einen HTML-Report, sondern auch Diagnosedateien, Trace Files (10046, 10053), Testcase-Dateien und vieles mehr. Diese werden in ein ZIP-Archiv gepackt, das zur Analyse genutzt werden kann.

Viele unterschiedliche Methoden können vom Client aus gestartet werden, um die Analyse zu starten. Jeder Aufruf generiert eine eindeutige ID des Statements, die sich im SQLT-Repository und in den Logfiles wiederfinden.

Die SQLT-Hauptmethoden sammeln Informationen über Ausführungspläne, Optimizerstatistiken, Objektdaten und -statistiken, Performancestatistiken sowie Konfigurationsparameter.

Die weiteren Methoden dienen zum Beispiel zur Analyse von Trace-Dateien oder auch zur Analyse der Unterschiede zweier SQL-Ausführungen.

Neben der Nutzung für die eigene Analyse fordert auch der Oracle Support bei Performance-Problemen SQLT-Reports an. Es gibt Methoden innerhalb von SQLT, die dann in Zusammenarbeit mit dem Support ausgeführt werden.

SQLT – Installation

Wenn SQLT von MOS heruntergeladen und entpackt ist, liegen folgende Verzeichnisse vor *(siehe Abbildung 2)*.

Nun muss noch das Tool in die zu analysierende DB installiert werden. Das geschieht mit einem einfachen Aufruf, allerdings muss dazu ein User benutzt werden, der SYSDBA-Rechte besitzt. Ansonsten wird eine Fehlermeldung ausgeworfen. Es bietet sich an, einen eigenen Tablespace für SQLT zu erstellen *(siehe Listing 1)*.

Nach der Installation der aktuellen Version gibt es zwei neue User in der

```
@sqcreate.sql
Optional Connect Identifier (ie: @PROD): @SSE18D
Password for user SQLTXPLAIN:
Default tablespace [UNKNOWN]: SQLTXPLAIN
Temporary tablespace [UNKNOWN]: TEMP
Main application user of SQLT: QTUNE
Oracle Pack license [T]:
...
SQCREATE completed. Installation completed successfully.
```

Listing 1: Installation SQLT

Datenbank mit entsprechenden Objekten:

- SQLTXPLAIN
- SQLTXADMIN

Bei der Installation muss der Applikationsuser angegeben werden, der die SQL-Abfragen ausführt, die analysiert werden sollen. Es ist möglich, weitere User einzurichten *(siehe Listing 2)*.

Ein wichtiger Punkt, der bei der Nutzung von SQLT nicht vernachlässigt werden darf, ist die Klärung der Nutzung von Funktionen, die durch Diagnostic Pack oder Tuning Pack bereitgestellt werden.

Bei jedem Aufruf wird mitgegeben, welches Pack lizenziert ist:

- „T“ if you have license for Diagnostic and Tuning
- „D“ if you have license only for Oracle Diagnostic
- „N“ if you do not have these two licenses

Dies lässt sich zunächst abfragen *(siehe Listing 3)*. Und dann auch anpassen *(siehe Listing 4)*.

Für die Deinstallation gibt es ein entsprechendes Skript: sqdrop.sql. Ein gegebenenfalls angelegter Tablespace und die Rolle SQLT_USER_ROLE müssen manuell gelöscht werden. Es werden keine Objekte in Applikationsschemata installiert und Histogrammwerte können verborgen werden. SQLT nutzt, analog zum AWR oder Statspack, ein eigenes Repository. Der User SQTXPLAIN besitzt folgende Rechte:

- select_catalog_role
- gather_system_statistics
- einige System- und Objektberechtigungen auf SYS- und OUTLN-Objekte.

Die Analyse geschieht mit einem Applikationsuser, der dazu die Rolle sqlt_user_role benötigt. Diese beinhaltet die select_catalog_role. Vor Nutzung von SQLT muss also geklärt sein, dass diese umfassenden Rechte gewährt werden dürfen. Nach Abschluss der Analyse sollte die Rolle auch wieder weggenommen werden.

SQLT protokolliert die ausgeführten Schritte in der View SQLTXPLAIN. SQLT$LOGV. Zusätzlich gibt es Logfiles der Methoden in sqlt/run.

SQLT – Hauptmethoden

Es gibt 5 Hauptmethoden:

```
grant SQLT_USER_ROLE to <user>;
```

Listing 2: Weitere User einrichten

```
select sqltxplain.sqlt$a.get_pack_access from dual;
```

Listing 3: Lizenznutzung abfragen

```
exec sqltxadmin.sqlt$a.enable_tuning_pack_access;
exec sqltxadmin.sqlt$a.enable_diagnostic_pack_access;
exec sqltxadmin.sqlt$a.set_param('sql_tuning_advisor', 'N');
exec sqltxadmin.sqlt$a.set_param('sql_monitoring', 'N');
exec sqltxadmin.sqlt$a.set_param('sql_tuning_set', 'N');
exec sqltxadmin.sqlt$a.set_param('automatic_workload_repository', 'N');
```

Listing 4: Lizenznutzung anpassen

Objekte SQLTXPLAIN	Anzahl
Table	215
Index	242
LOB	110
Synonym	17
Sequence	12
Table Partition	32
Procedure	1
Objekte SQLTXADMIN	**Anzahl**
Synonym	247
View	105
Package + Procedure	17
Type	8

Tabelle 2

1. sqltxract.sql

Die XTRACT-Methode analysiert bereits ausgeführte SQL-Statements. Dazu muss das SQL-Statement entweder im Memory oder im AWR vorhanden sein. Die SQL_ID oder der HASHVALUE wird für den Aufruf genutzt. Ein AWR-Report, ein Trace File, v$sql oder v$sqlarea liefern die entsprechende SQL_ID oder den HASHVALUE. Kann ein SQL nicht erneut ausgeführt werden, so hilft SQLTXTRACT, die notwendigen Statistiken zu sammeln.

Wichtige Metriken (z.B. die Anzahl von Zeilen pro Ausführungsoperation) werden nur geliefert, wenn der Parameter STATISTICLEVEL auf ALL gesetzt ist. Alternativ kann der Hint /*+ gather_plan_statistics */ gesetzt werden. Zur Ausführung der SQLTXTRACT-Methode meldet man sich als Applikationsuser an der Datenbank an und führt die Methode aus *(siehe Listing 5)*.

2. sqltxecute.sql

Mit dieser Methode kann ein Statement erneut ausgeführt werden *(siehe Listing 6)*. Dazu wird das SQL in einem Textfile in einem definierten Format gespeichert. Nach der Ausführung werden die notwendigen Daten aus dem Memory herausgezogen. Nötige Bind-Variablen können im Textfile deklariert werden. Ein entsprechender Hint erzwingt bei jeder Ausführung ein Hard-Parsen. Ebenso ist es nützlich, zu Beginn des Skripts das STATISTICLEVEL auf ALL zu setzen.

3. sqltxplain.sql

Mit SQLTXPLAIN lässt sich ein Ausführungsplan für das übergebene SQL erstellen *(siehe Listing 7)*.

4. sqltxtrsby.sql

Es ist möglich, auf einer Primärdatenbank die Methode zu starten, aber die Informationen aus dem Memory beziehungsweise AWR zum SQL von der Standby-Datenbank zu sammeln *(siehe Listing 8)*.

Methode	Skript	Funktion
SQLTXTRACT	sqlt/run/sqltxtract.sql	Eingabe SQL_ID SQL wird nicht ausgeführt
SQLTXECUTE	sqlt/run/sqltxecute.sql	Eingabe Skript SQL wird ausgeführt
SQLTXTRXEC	sqlt/run/sqltxtrxec.sql	Eingabe SQL_ID Kombination aus SQLXTRACT und SQLTXECUTE
SQLTXPLAIN	sqlt/run/sqltxplain.sql	Eingabe Skript Nutzt Bind-Variablen oder Literale
SQLTXTRSBY	sqlt/run/sqltxtrsby.sql	Eingabe SQL_ID und DBLINK Für Read-Only-Datenbanken

Tabelle 3: 5 SQLT–Hauptmethoden

```
@sql/run/sqltxtract D|T|N f995z9antmhxn
```

Listing 5: SQLTXTRACT-Anmeldung

```
@sql/run/sqltxecute.sql D|T|N input/sample/test_script.sql
```

Listing 6: SQLTXTRACT-Ausführung

SQLT – HTML-Report

Im HTML-Report werden alle relevanten Informationen präsentiert *(siehe Abbildung 3)*. Wie umfangreich diese Informationen sind, hängt zum einen von der gewählten Methode, aber auch von den Funktionen des Diagnostic und Tuning Pack ab, sofern sie genutzt werden dürfen. Sind bestimmte Funktionen nicht vorhanden, so funktionieren die entsprechenden Links nicht.

Der Einstieg sollte gleich der erste Link sein. „Observation", das über das Healthcheck-Modul gefüllt wird, liefert wichtige Punkte und erste Erklärungen. Diese Erklärungen muss jedoch jeder für sich verifizieren und nachvollziehen.

Anschließend sollten die einzelnen Links zu Ausführungsplänen, Statistiken etc. geprüft werden.

SQLT – README

Zu jedem Report wird auch eine README-Datei erstellt, die Hilfestellungen dafür

Abbildung 3: SQLT HTML-Report (Quelle: Stefan Seck)

bietet, wie mit dem Output weiter verfahren werden kann *(siehe Abbildung 4)*.

Besonders interessant ist der Punkt „SQLT Test Case“, denn hier bietet SQLT die Möglichkeit, einen Test Case zu bauen, der auf einem anderen System genutzt werden kann. So haben wir die Möglichkeit, das Tuning eines Statements in Ruhe durchzuführen, ohne das produktive System zusätzlich zu belasten.

SQLT Health Check

Neben den oben genannten Methoden gibt es im Verzeichnis /utl das Skript sqlhc.sql, das einen Healthcheck-Report generiert. Der Aufruf funktioniert auch ohne eine Installation in der zu analysierenden Datenbank, dazu muss allerdings ein User verwendet werden, der Zugriff auf das Data Dictionary hat. Folgende Aspekte werden geprüft:

- Statistiken für die Objekte, auf die zugegriffen wird
- Datenbank-Parameter
- System-Statistiken
- Data-Dictionary-Statistiken
- Fixed-Objects-Statistiken

Fazit

SQLT ist ein sehr mächtiges Tool und liefert umfangreiche Informationen, die zur Analyse von SQL-Statements genutzt werden können. Um sie zu nutzen, muss es jedoch möglich sein, SQLT zu installieren.

```
@sql/run/sqltxplain.sql input/sample/test_script.sql
```

Listing 7: SQLTXPLAIN

```
@sql/run/sqltxtrsby D|T|N f995z9antmhxn link2stby
```

Listing 8: SQLTXTRSBY

Sofern die Lizenzen für Diagnostic und Tuning Pack vorhanden sind, werden die Daten noch detailreicher. Mithilfe des Outputs ist es möglich, die Ausführung einzelner SQLs genau zu analysieren und Ansätze dafür zu finden, wie ein SQL verbessert werden kann.

Auch wenn die Reports sehr detailliert sein können und Hilfestellungen bieten, muss doch jeder Anwender selbst die Auswertung sowie eine jeweilige Schlussfolgerung vornehmen.

Dokumentation

Informationen zu den folgenden Punkten finden Sie unter My Oracle Support (MOS): *https://support.oracle.com*

MOS 215187.1
„All About the SQLT Diagnostic Tool“

MOS 1614107.1
„SQLT Usage Instructions“

MOS 1454160.1
„FAQ:SQLT (SQLTXPLAIN) Frequently Asked Questions“

MOS 1366133.1:
„SQLT Tuning Health-Check Script (SQLHC)“

MOS 1417774.1:
„FAQ: SQL Health Check (SQLHC) Frequently Asked Questions“

SQLTXPLAIN Programm-Dokumentation (Verzeichnis sqlt/doc)

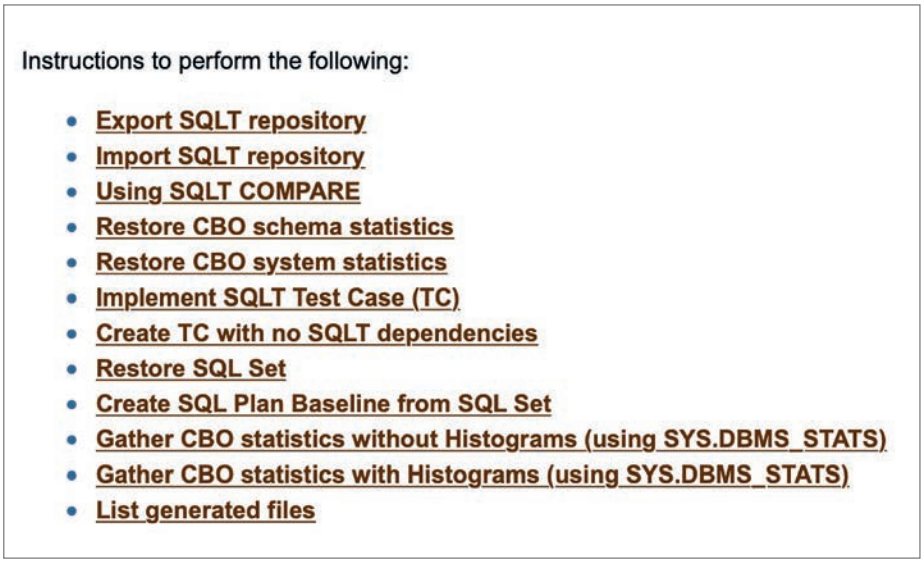


Abbildung 4: SQLT Readme (Quelle: Stefan Seck)

Stefan Seck
stefan.seck@logicalis.de

Autonomous Database

Sebastian Solbach, Oracle Global Services Germany

Oracle stellte bereits letztes Jahr mit der Autonomous Database Serverless den ersten vollkommen gemanagten Datenbank-Dienst in der Oracle-Cloud-Infrastruktur zur Verfügung. Auch wenn es sich dabei technisch um eine vollständig kompatible Oracle-Datenbank handelt, so unterscheidet sich dieser Service doch erheblich von anderen „managed" Oracle-Datenbank-Diensten anderer Cloud-Anbieter. So setzt Autonomous auf den Best Practices für Oracle-Datenbanken auf und bietet auch in der Cloud dieselbe zuverlässige Oracle-Datenbank, die Kunden gewohnt sind, allerdings ohne sich selbst mit der Komplexität derselben auseinandersetzen zu müssen.

Die bestehende „Serverless"-Umgebung wurde im Juni durch einen dedizierten Autonomous Database Service ergänzt, da Oracle-Datenbanken auch in einigen kritischen Geschäftsbereichen von Unternehmen eingesetzt werden. Für diese ist der Betrieb einer Oracle-Datenbank in der Cloud mit einem öffentlichen Zugang (Public End Point) einfach nicht möglich. Allerdings wollten auch sie in den Genuss der Vorteile von Autonomous Database kommen, weshalb das dedizierte Deployment der autonomen Datenbank in das virtuelle Cloud-Netzwerk des Kunden die ideale Lösung bietet. Welche Vorteile Autonomous bietet und welche Einsatzbereiche beide Services haben, wird im folgenden Artikel näher beleuchtet.

Oracle Autonomous Database

Auch wenn der Name vermuten lässt, dass es sich hierbei um ein anderes Datenbank-Release der Oracle-Datenbank handelt, ist dies nicht der Fall. Auch bei der Oracle Autonomous Database handelt es sich technisch gesehen um eine 18c- beziehungsweise 19c-Datenbank. Autonom wird die Datenbank erst durch 2 wichtige weitere Komponenten: die Oracle Cloud-Infrastruktur und die vielen autonomen Funktionen dank Machine Learning. Denn bei der Autonomous Database handelt es sich nicht um eine Lösung, die als reine Software dem Kunden zur Verfügung steht, sondern um einen Datenbank-Cloud-Dienst in der Oracle Cloud. Viele Funktionen wie Performance-Optimierungen, Patching, Backup und vieles mehr werden bei diesem Service automatisch übernommen; es bedarf keines manuellen Eingreifens des Benutzers.

Die Autonomous-Database-Funktionen

Die autonome Oracle-Datenbank selbst läuft auf dem Exadata Cloud Service innerhalb der Oracle-Cloud-Infrastruktur. Somit wird der autonome Datenbank-Service mit allen Oracle Best Practices bereitgestellt, inklusive Real Application

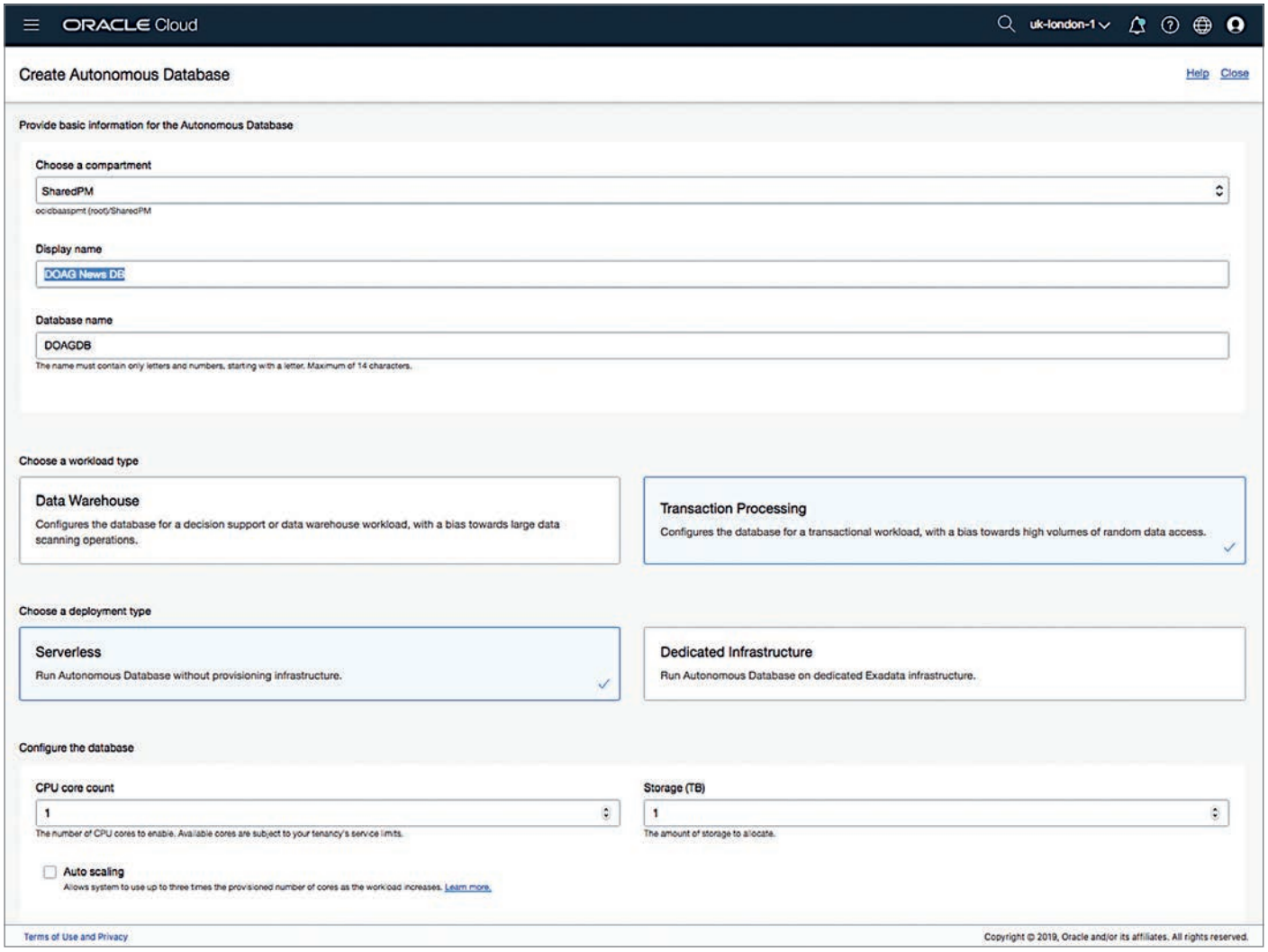


Abbildung 1: Autonomous Deployment (Quelle: Sebastian Solbach)

Clusters und Exadata-Verbesserungen wie dem Auslagern von Abfragen auf die Storage-Zellen. Die Infrastruktur stellt dabei sicher, dass alle wichtigen Patches – unabhängig davon, ob es sich dabei um sicherheitsrelevante oder Patches mit neuen Funktionen handelt – entweder komplett online oder Rolling eingespielt werden.

Durch die Zugriffsarchitektur und die Verwendung von Funktionen wie Application Continuity *[1]* hat Autonomous Serverless im letzten Jahr stets zur Verfügung gestanden, ohne den Kunden durch Patch-Aktivitäten mit Downtimes zu konfrontieren. Auch wenn der Kunde es nicht bemerkte, wurde die Infrastruktur immer aktuell gehalten (mindestens 1x im Monat, wenn nicht sogar öfter).

Auch wenn Oracle für diese Patch-Aktivitäten Zugriff auf den Server der Kunden braucht, sind dessen Daten absolut sicher. Technologien wie Transparent Data Encryption und Database Vault sichern die Daten der Kunden, selbst wenn ein Oracle-Administrator ausnahmsweise auf die Server zugreifen sollte.

Ein ganz entscheidendes Merkmal der Autonomous Database ist dabei die Online-Skalierbarkeit, egal ob es sich dabei um CPU-Skalierung oder Storage- Skalierung handelt. Sie erlaubt es Anwendern, zwischen 1 und 128 CPUs sowie 1 TB und 128 TB Storage zu skalieren.

Das eigentliche Herzstück, weshalb dieser „fully managed" Datenbank-Service „Autonomous" heißt, sind die autonomen Optimierungen, die viele der „klassischen" Performance-Tuning-Maßnahmen überflüssig machen. Dazu gehört sicher auch die neue 19c-Funktion der automatischen Indizierung, die bei Autonomous Database ihre Anwendung findet.

Autonomous Database gibt es dabei in den Varianten Data Warehouse (Autonomous Data Warehouse) und Autonomous Transaction Processing (OLTP). Bei beiden handelt es sich um die gleiche Oracle-Datenbank. Allerdings sind die Datenbanken leicht anders parametrisiert und unterscheiden sich in der Art, wie Daten gespeichert werden. Autonomous Data Warehouse verwendet hierbei eine spaltenweise Komprimierung, während bei Autonomous Transaction Processing zeilenorientiert gearbeitet wird. Es handelt sich aber, wie eingangs schon erwähnt, um exakt dieselbe Datenbank-Version.

Autonomous Database steht seit Kurzem nun in zwei Arten zur Verfügung: als Autonomous Serverless, was einen schnellen Zugriff auf eine Oracle-Datenbank von überall her erlaubt, und Autonomous Dedicated, die Kunden exklusiv im kundeneigenen privaten Virtual Cloud Network (VCN) zur Verfügung steht.

Autonomous Serverless

Das Ziel von Autonomous Serverless ist es, dem Kunden möglichst einfach und schnell eine Oracle-Datenbank zur Verfügung zu stellen und sich dabei um alle notwendigen Dinge zu kümmern: angefangen vom simplen automatischen Backup bis hin zum automatischen Patchen der Umgebung.

Dies kann auch jeder Kunde schnell ausprobieren, da man innerhalb von 5 Minuten eine fertige Autonomous-Datenbank-Umgebung in der Oracle-Cloud-Infrastruktur zur Verfügung gestellt bekommt. Zugreifen kann man auf den Autonomous Serverless Service dabei – wie es sich für einen öffentlichen Cloud-Service gehört – über das Public Internet. Damit die Kommunikation zur autonomen Oracle-Datenbank jedoch nicht unverschlüsselt passiert, erhält der Kunde ein Wallet, in dem der TLS-Schlüssel enthalten ist, um verschlüsselt mit seinen Autonomous-Datenbanken in seinem Cloud Account zu kommunizieren *[2]*.

Das Erstellen benötigt dazu nur fünf Angaben *(siehe Abbildung 1)*:

- Datenzentrum (Region)
- Datenbank-Name
- Anzahl gewünschter CPUs (1 – 128)
- Storage (in TB)
- Administrations-Passwort

Selbstverständlich lassen sich Autonomous Databases wie alle anderen Oracle-Cloud-Infrastruktur-Services dabei nicht nur über die Console, sondern auch über Rest API, Command Line Interface (OCI CLI), Terraform, Ansible, Chef sowie etliche SDKs anlegen und administrieren.

Technisch bekommt der Kunde bei einer Autonomous Database eine Pluggable Datenbank (PDB) in einer von Oracle administrierten Container-Datenbank (CDB) zur Verfügung gestellt. Die Datenbanken laufen bei Autonomous Serverless auf Exadata Full Racks. Der Kunde erhält nur Zugriff auf seine Datenbank und nicht auf das darunterliegende Betriebssystem oder CDB. Aus diesem Grund sind auch einige Datenbank-Funktionalitäten in diesen PDBs eingeschränkt. Dies hat Auswirkungen auf die Technologien, die benutzt werden können, um Daten in Autonomous zu laden, da zum Beispiel Transportable Tablespaces dadurch nicht möglich sind. Eine Alternative dazu gibt es mit einem speziellen Package (DBMS_CLOUD), das den direkten Zugriff auf das Oracle Object Storage ermöglicht, um direkt vom Object Storage Dateien sowie Export-Daten in Autonomous laden zu können. Ebenso sind einige Oracle-Technologien noch nicht verfügbar, werden aber nach und nach bereitgestellt. Zum Zeitpunkt des Artikels war beispielsweise Java in der Datenbank nicht verfügbar, andere wie Application Express (APEX) wurden jedoch schon zur Verfügung gestellt. Alle Restriktionen findet man im „Autonomous Database for Experienced User"-Appendix der Dokumentation *[3]*.

Einige Erweiterungen der letzten Zeit beinhalteten dabei die Möglichkeiten, eingehende und ausgehende Datenbank-Links zu definieren und Datenbanken innerhalb der Umgebung für Testzwecke klonen zu können.

Interessant ist auch die Autoskalierung von Autonomous-Datenbanken, die es erlaubt, die verwendeten CPUs automatisch auf das Dreifache der Ausgangs-CPUs anzuheben. Dies erlaubt es auch ohne manuelles Eingreifen, die benötigten Datenbank-CPUs automatisch an die Abfragelast anzupassen.

Ganz neu ist die Integration des Performance-Hubs und von SQL Developer Web, über die man dedizierte Informationen zur autonomen Datenbank bekommt *[4]*. Den Performance Hub wird der eine oder andere in ähnlicher Form vom Enterprise Manager Express von seinen Datenbanken im Unternehmen kennen, während der SQL Developer Web eine veränderte Variante des SQL Developer explizit für das Web darstellt.

Mehr Kontrolle für den Zugriff auf die Autonomous-Datenbank bieten Access Control Lists (ACLs), um den Zugriff auf

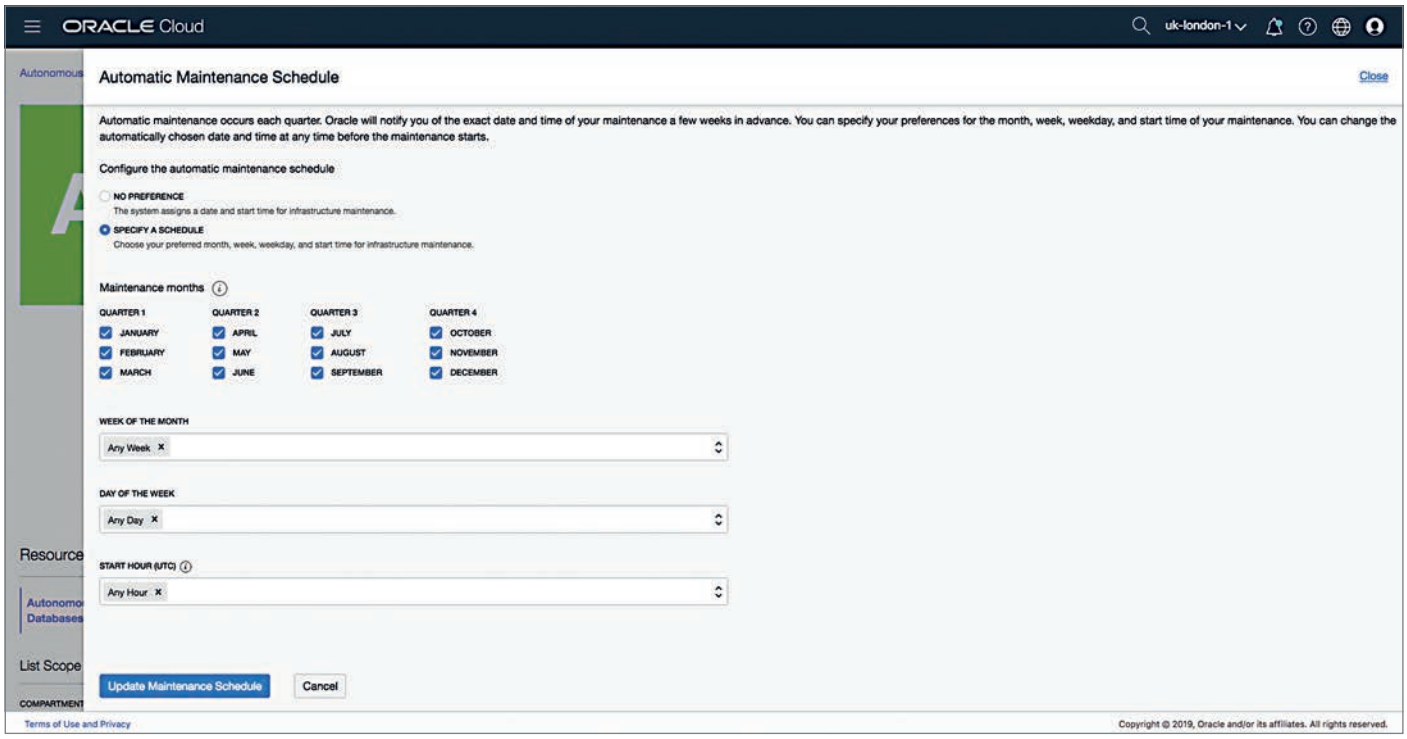


Abbildung 2: Autonomous Dedicated Patching Schedule (Quelle: Sebastian Solbach)

bestimmte IPs zu beschränken. Diese Funktion wird noch weiterentwickelt, um den Zugriff auf ein bestimmtes Virtual-Cloud- Netzwerk einzuschränken.

Autonomous Dedicated

Genau hier ist einer der Ansatzpunkte für das dedizierte Deployment von Autonomous Database. Im Gegensatz zu Autonomous Serverless, bei dem der Kunde nur die Kontrolle über eine PDB bekommt, erhält der Kunde bei Autonomous Dedicated seine eigene Exadata, die in das kundeneigene Virtual Cloud Network (VCN) „deployed" wird. Im Moment steht dabei nur ein Quarter Rack Exadata zur Verfügung; Half, Full und das sogenannte Base Rack sind in Planung. Der Kunde bekommt somit die Kontrolle über die Exadata-Infrastruktur und die Container-Datenbank. Dies bedeutet nicht, dass er Zugriff auf dieselbe bekommt, aber zumindest bekommt er mehr Kontrolle über die Wartungsfenster, wenn Patches eingespielt werden dürfen. Gibt es bei Serverless nur in Bezug auf zukünftige Major Releases Einflussmöglichkeiten, so kann der Kunde auf Basis der Exadata-Infrastruktur genau festlegen, wann überhaupt gepatcht werden darf *(siehe Abbildung 2)*. Dies ist insbesondere dann wichtig, wenn etwa in einem Monat gar nicht gepatcht werden sollte, da hier selbst ein „online rolling upgrade" ein Problem darstellen würde: Der Service stände zwar weiterhin zur Verfügung, da jedoch ein Knoten kurzzeitig ausfällt, macht sich dies in der Gesamtperformance des Systems bemerkbar. Dies wäre zum Beispiel bei Firmen, die während der Weihnachtszeit ihren Hauptumsatz machen, nicht gewünscht.

Der andere Aspekt ist, dass die autonome Datenbank nur innerhalb des VCN des Kunden zur Verfügung steht und die Hardware dem Kunden dediziert gehört. Damit kann er sich sicher sein, dass die Performance rein von seinen autonomen Datenbanken abhängt und der Zugriff nur innerhalb seines privaten VCN möglich ist.

Auch bietet das für die Zukunft mehr Einflussmöglichkeiten auf das System. Da es hier keine Notwendigkeit gibt, Sicherheitsvorkehrungen von anderen PDBs zu treffen, bietet Autonomous Dedicated eine andere Basis und kann somit auch schneller Funktionen zur Verfügung stellen, die auf Autonomous Serverless mehr Entwicklungsaufwand benötigen. So waren bei Autonomous Dedicated schon ab dem ersten Tag Oracle Datenbank 19c, APEX und SQL Developer Web verfügbar. Theoretisch wäre ebenso eine andere Ebene des Datenladens denkbar, auch wenn dies im Moment den Vorgaben von Serverless folgt.

Im Moment gibt es noch funktionale Unterschiede zwischen Dedicated und Serverless. So stehen bei Dedicated die Service Console, Performance Hub, Cloning und Auto Scaling noch nicht zur Verfügung. Dies ist allerdings nur eine Frage der Zeit.

Interessant ist, dass man bei Dedicated nun zwar die Exadata-Infrastruktur mit einrichtet und die CDB definiert, das eigentliche Deployment der Autonomous- Datenbank sich jedoch nur in einem Punkt unterscheidet und daher der Kunde, der „nur" den Zugriff auf die Datenbank benötigt, keinen großen Unterschied bei Dedicated zu Serverless merkt. Daher werden auch alle Autonomous-Datenbanken in der Console gemeinsam gelistet *(siehe Abbildung 3)*.

Autonomous Database – Verfügbarkeit und Preis

Wie alle Datenbank-Cloud-Services von Oracle bietet auch Autonomous die Möglichkeit, seine existierenden Lizenzen in die Cloud zu übernehmen. So kann man bei Autonomous jederzeit zwischen BYOL (Bring your own License) und License Included (Lizenz + Support werden abgerechnet) wechseln *[5]*. Die Abrechnung erfolgt stündlich.

Dennoch gibt es zwischen Serverless und Dedicated einen erheblichen Unterschied: Werden bei Serverless lediglich die verwendeten Datenbank-CPUs und der Speicherplatz in Rechnung gestellt, so ist bei Dedicated die Exadata-Infrastruktur fix mit zu bezahlen. Diese richtet sich dabei nach denselben Preisen wie der Exadata Cloud Service.

Fazit

Oracle Autonomous Database bietet eine ideale Plattform, um in der Cloud seine Oracle-Datenbank betreiben zu lassen.

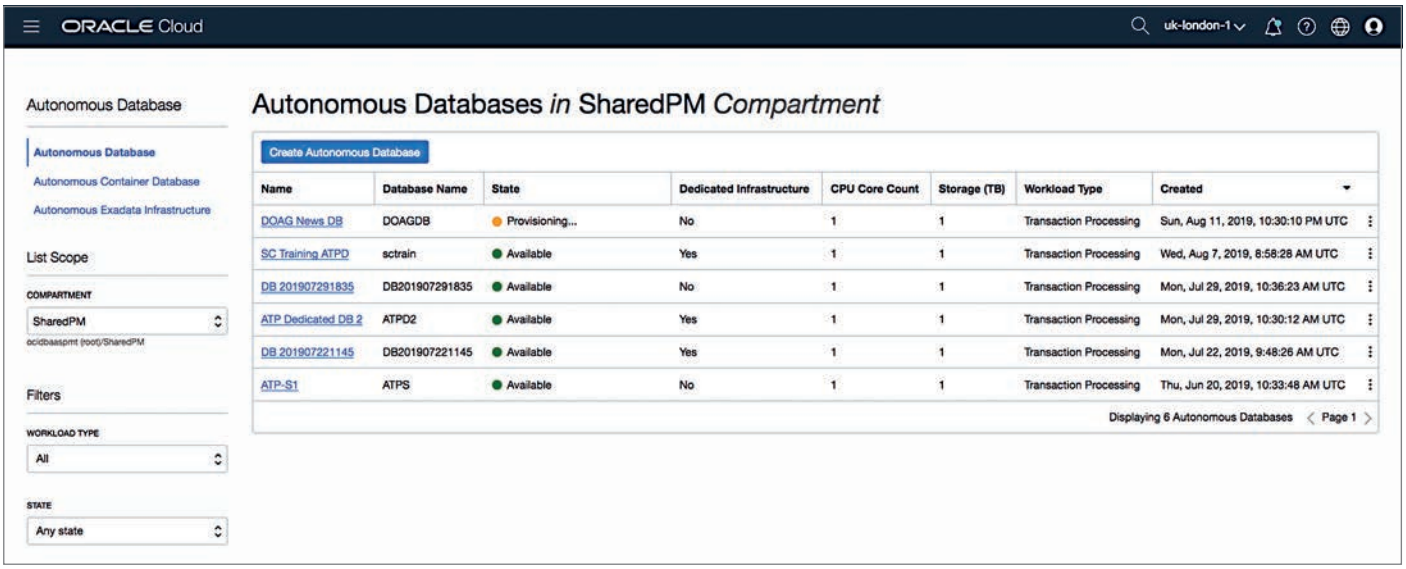


Abbildung 3: Listing aller Autonomous-Datenbanken (Quelle: Sebastian Solbach)

Über viele „lästige“ Arbeiten wie erfolgreiches Backup (und mögliches Gecovert) und Patchen sowie ständiges Datenbank-Tuning braucht man sich ebenso keine Sorgen zu machen. Allerdings erfordert dies auch ein Umdenken darüber, wie Applikationen gegen eine Oracle-Datenbank zertifiziert werden und wie generell eine Entwicklung vonstattengeht. Ist Autonomous Database gerade für Neuentwicklungen eine ideale Plattform, um sich auf das Wesentliche zu konzentrieren, so können die bestehenden Restriktionen einer „managed“-Datenbank-Plattform gerade für ältere Applikationen durchaus eine Herausforderung sein.

Nichtsdestotrotz sollte jeder, der mit Oracle-Datenbanken arbeitet, Autonomous Database einmal ausprobiert haben, da es vieles an Komplexität der Oracle-Datenbank entschärft und damit den Betrieb einer hochverfügbaren, zuverlässigen Datenbank-Plattform bietet, die weit mehr als nur ein relationaler Datenspeicher ist!

Quellen

[1] White Paper Application Continuity: *https://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/adb-continuousavailability-5169724.pdf*

[2] Connecting to Autonomous Database *https://docs.cloud.oracle.com/iaas/Content/Database/Tasks/adbconnecting.htm*

[3] Autonomous Transaction Processing for Experienced Oracle Database Users *https://docs.oracle.com/en/cloud/paas/atp-cloud/atpug/experienced-database-users.html*

[4] Using Performance Hub to Analyze Database Performance in Oracle Cloud Infrastructure *https://docs.cloud.oracle.com/iaas/Content/Database/Tasks/perfhub.htm*

[5] FAQs for Autonomous Database *https://www.oracle.com/database/technologies/datawarehouse-bigdata/adb-faqs.html*

Sebastian Solbach
sebastian.solbach@oracle.com

Oldies but Goldies Teil 1

Roger Troller, Finnova bankware

In regelmäßigen Abständen beschäftigen wir uns mit den neuen Möglichkeiten, die jedes Oracle Release mit sich bringt. Dabei sollten wir uns auch die Frage stellen, ob wir die alten Möglichkeiten, deren Einführung zum Teil viele Releases zurückliegen, auch nutzen. Dieser Artikel legt den Fokus auf ein paar dieser „old features“, die völlig zu Unrecht ein wenig in Vergessenheit geraten sind und deswegen ein Mauerblümchendasein fristen. Der Grund für diesen Umstand ist vielfach, dass in den Releases, in denen diese Möglichkeiten angeboten wurden, spektakulärere Neuerungen geboten wurden, oder aber, dass zu jener Zeit zu wenig Anwendungsfälle präsentiert wurden, in denen diese „Oldies“ ihre Stärken hätten ausschöpfen können. Es ist also Zeit, in frühere Versionen zurückzublicken und ein paar „Goldies“ hervorzukramen.

Partitionierter Outer-Join

Mit der Version 10 ermöglichte Oracle über die ANSI-Syntax, einen partitionierten Outer-Join auszuführen. Wie der „normale" Outer-Join wird auch der partitionierte Outer-Join benutzt, um lückenhafte Daten zu ergänzen beziehungsweise Lücken zu füllen. Dabei wird der partitionierte Outer-Join auf einzelne Gruppen von Datensätzen statt auf die gesamte Resultatmenge angewandt. Dadurch werden in jeder dieser Gruppen die fehlenden Daten eingefügt, was im nachfolgenden Beispiel zur Lösung der Problemstellung führt.

Nehmen wir an, dass wir eine Liste erzeugen möchten, in der wir sehen, in welchen Monaten pro Department wie viele Personen eingestellt worden sind. Ein einfacher Outer-Join zwischen der Monatssicht und den Mitarbeitern (HIRE_DATE) würde dazu führen, dass jeder Monat mindestens einmal im Resultat enthalten wäre, gewünscht ist jedoch, dass jeder Monat für jedes Department im Resultat einmal enthalten ist. Es bräuchte also einen Outer-Join zwischen der Monatssicht und den Mitarbeitern pro Department und genau diese Funktionalität bietet uns der partitionierte Outer-Join *(siehe Abbildung 1)*.

Der Outer-Join zwischen jeder dieser Mitarbeitergruppen und der Monatstabelle erzeugt das gewünschte Resultat *(siehe Listing 1)*.

Monate

MM	Monat
01	Januar
02	Februar
03	März
04	April
05	Mai
06	Juni
07	Juli
08	August
09	September
10	Oktober
11	November
12	Dezember

Mitarbeiter pro Monat und Department

DEPARTMENT	MONAT	ANZAHL
10	09	1
20	02	1
20	08	1
30	05	1
30	07	1
30	08	1
30	11	1
30	12	2
40	06	1
...		

Monate

MM	Monat
01	Januar
02	Februar
03	März
04	April
05	Mai
06	Juni
07	Juli
08	August
09	September
10	Oktober
11	November
12	Dezember

Mitarbeiter pro Monat Department 10

DEPARTMENT	MONAT	ANZAHL
10	09	1

Mitarbeiter pro Monat Department 20

DEPARTMENT	MONAT	ANZAHL
20	02	1
20	08	1

Mitarbeiter pro Monat Department 30

DEPARTMENT	MONAT	ANZAHL
30	05	1
30	07	1
30	08	1
30	11	1
30	12	2

Mitarbeiter pro Monat Department ...

DEPARTMENT	MONAT	ANZAHL
...		

Abbildung 1: Gruppierung der Mitarbeiter-Sicht auf Ebene Department, vor Ausführung des Outer-Join (Quelle: Roger Troller)

„Interval"-Datentypen

Jeder, der schon nach Varianten gesucht hat, um die Zeitdifferenz zwischen zwei Datumswerten zu ermitteln, ist wohl auf eine Lösung wie eine der folgenden gestoßen. Hier gibt es zum einen die mathematische Variante *(siehe Listing 2)*:

Zum anderen steht die Konvertierungsvariante zur Verfügung *(siehe Listing 3)*.

Wenn man diese Abfragen betrachtet, dann wird man sich zu Recht fragen: „Nett, ... aber geht das auch einfacher?" Und ja, natürlich geht es einfacher, denn SQL kann die Differenz zwischen zwei Datumswerten, die bei einer einfachen Subtraktion der beiden Werte als Differenz in Tagen ermittelt wird, in einen „Interval"-Datentyp umwandeln und so eine Zeitdauer von Tagen/Stunden/Minuten/Sekunden darstellen *(siehe Listing 4)*.

Das gleiche Resultat erhält man, wenn man statt eines Date einen Timestamp als Minuend bei der Subtraktion verwendet. Im Gegensatz zu einer Date-Date-Operation wird bei einer Timestamp-Date-Operation das Resultat den Datentyp „Interval" statt „Number" haben *(siehe Listing 5)*.

Die „Interval"-Datentypen sind in Oracle seit der Version 9 bekannt und genauso lange gibt es auch die Möglichkeit, mit diesen zu rechnen beziehungsweise in diese zu konvertieren.

KEEP FIRST / KEEP LAST

FIRST und LAST sind als Zusatz zu den Aggregatsfunktionen mit der Version 9 von Oracle ausgeliefert worden. Sie werden dazu verwendet, um innerhalb einer Gruppe von Datensätzen über eine Rangierung den- beziehungsweise diejenigen zu bestimmen, die für die vorhergehende Aggregatsfunktion relevant sind.

Als Beispiel möchten wir in der EMPLOYEES-Tabelle das Durchschnittssalär der Mitarbeiter des letzten Anstellungsjahres pro Department ermitteln. Dazu ist zu sagen, dass das letzte Anstel-

lungsjahr pro Department unterschiedlich sein kann. Dieses Problem kann man lösen, indem man in einer Unterabfrage das letzte Einstellungsjahr pro Department ermittelt und das Resultat dieser Unterabfrage verwendet, um die Hauptabfrage einzuschränken, bevor in der Hauptabfrage über AVG das durchschnittliche Salär ermittelt wird *(siehe Listing 6)*.

Im *Listing 6* lesen wir die Employees-Tabelle doppelt und auch sonst ist diese Abfrage viel komplizierter, als sie eigentlich sein müsste. Mit FIRST beziehungsweise LAST als Zusatz zur Aggregatsfunktion AVG lässt sich das gleiche Resultat wesentlich einfacher erreichen, indem man die relevanten Datensätze innerhalb eines Departments bestimmt (letztes Einstellungsjahr), auf welche die Gruppenfunktion (AVG) angewandt werden soll *(siehe Listing 7)*.

```
with MonthList as (select rownum as mm
                        ,to_date(rownum,'MM') mon
                    from dual connect by level <= 12)
      ,EmpList   as (select extract(month from e.HIRE_DATE) mm
                        ,e.DEPARTMENT_ID
                        ,count(*) as AnzMA
                    from EMPLOYEES e
                    group by e.DEPARTMENT_ID
                        ,extract(month from e.HIRE_DATE))
select emp.department_id
      ,to_char(ml.mon,'Month') as mon
      ,COALESCE(emp.AnzMA,0) as AnzMA
   from       MonthList ml
   left join EmpList   emp partition by (emp.department_id)
                       on (emp.mm = ml.mm)
   order by emp.department_id
   ,ml.mon;


DEPARTMENT_ID  MON                                  ANZMA
-------------  ------------------------------------  ------
…
           30   January                               0
           30   February                              0
           30   March                                 0
           30   April                                 0
           30   May                                   1
           30   June                                  0
           30   July                                  1
           30   August                                1
           30   September                             0
           30   October                               0
           30   November                              1
           30   December                              2
…
```

Listing 1: Outer-Join zwischen Monaten (Master) und Mitarbeiter pro Department

```
with dates (d1,d2)
      as (select to_date('03.01.2018 08:15:27'
                        ,'dd.mm.yyyy hh24:mi:ss')
             ,to_date('05.01.2018 15:03:49'
                        ,'dd.mm.yyyy hh24:mi:ss')
             from dual)
select trunc(d2 - d1)                            as Tage
      ,trunc(mod(d2 - d1,1) * 24)                as Stunden
      ,trunc(mod((d2 - d1) * 24,1) * 60)         as Minuten
      ,round(mod((d2 - d1) * 24 * 60,1) * 60)    as Sekunden
   from dates;

      TAGE    STUNDEN    MINUTEN    SEKUNDEN
----------    --------   --------   ----------
      2          6          48         22
```

Listing 2: Verbreitete Variante, um eine Datumsdifferenz in Tagen/Stunden/Minuten/Sekunden zu errechnen (mathematische Variante).

User-specific Quote-Operator

Überall, wo wir mit der Verarbeitung von Zeichenketten konfrontiert sind, tritt immer wieder der Fall auf, dass innerhalb dieser Zeichenkette ein einfaches Hochkomma vorkommen kann. Ein solches Hochkomma lässt sich durch ein vorgestelltes weiteres Hochkomma entwerten, was je nach Situation interessante Konstrukte zur Folge haben kann. Zwei Fundstücke hierzu *(siehe Listing 8)*:

Seit Oracle 10 lassen sich solche Morsecodes dadurch vermeiden, dass man anstelle des Hochkommas ein beliebiges anderes Zeichen als String-Begrenzer verwendet. Die oben gezeigten Beispiele könnten so sehr viel les- und wartbarer gestaltet werden, indem man sie wie folgt schreibt *(siehe Listing 9)*:

Das **q** (oder **nq** für NCHAR- oder NVARCHAR-Zeichenketten) leitet dabei den String ein, das erste Zeichen hinter dem Hochkomma (hier {) ist der neue String-Begrenzer. Die Zeichenkette geht danach bis zum nächsten Auftreten des String-Begrenzers, der unmittelbar von einem Hochkomma gefolgt wird. Speziell ist dabei das Verhalten bei Klammern – eine schließende Klammer gilt als Ende für einen String, der durch eine öffnende Klammer eingeleitet wurde. Dies gilt für alle Arten von Klammern: () {}[] sowie <>.

NULLIF

Eine weitere Funktion, die Bestandteil von Oracle 9 war und viel zu selten genutzt wird, ist NULLIF. NULLIF gibt NULL zurück, wenn beide Parameter den gleichen Wert enthalten. Wir können uns diese Funktion überall dort zunutze machen, wo wir eine Division durchführen und nicht sicher sein können, ob der Divisor 0 enthält, was zu einer „ORA-01476: divisor is equal to zero"-Exception führen würde. Eine Division durch NULL da-

gegen führt zum Resultat NULL und verhindert einen Abbruch *(siehe Listing 10)*.

Via CASE wird eine Division durch 0 verhindert *(siehe Listing 11)*.

Mit NULLIF lässt sich eine Division durch 0 verhindern *(siehe Listing 12)*.

Ein weiterer Ort, an dem NULLIF eingesetzt werden kann, ist eine bedingte Verkettung von Strings. Wir möchten an den bestehenden String ein Element anfügen, das vom aktuellen Inhalt durch ein Pipe-Zeichen getrennt ist, … allerdings nur dann, wenn der String aktuell nicht leer ist. Um dies zu erreichen, findet man im Code häufig eine entsprechende IF-Anweisung oder aber ein TRIM von führenden oder am Schluss stehenden Begrenzern *(siehe Listing 13)*.

```
with dates (d1,d2)
     as (select to_date('03.01.2018 08:15:27'
                        ,'dd.mm.yyyy hh24:mi:ss')
                ,to_date('05.01.2018 15:03:49'
                        ,'dd.mm.yyyy hh24:mi:ss')
           from dual)
select case
          when d2 -d1 > 1 then
             to_char((trunc(sysdate,'year') -1) + (d2 - d1)
                   ,'DDD HH24:MI:SS')
          else
             '000 '||to_char(trunc(sysdate,'year') + (d2 - d1)
                           ,'HH24:MI:SS')
          end as TimeDiff
  from dates;

TIMEDIFF
------------
002 06:48:22
```

Listing 3: Verbreitete Variante, um eine Datumsdifferenz in Tagen/Stunden/Minuten/Sekunden zu errechnen (Konvertierungsvariante).

```
with dates (d1,d2)
  as (select to_date('03.01.2018 08:15:27'
                     ,'dd.mm.yyyy hh24:mi:ss')
            ,to_date('05.01.2018 15:03:49'
                     ,'dd.mm.yyyy hh24:mi:ss')
        from dual)
select numtodsinterval(d2-d1,'DAY') as Differenz
  from dates;

DIFFERENZ
-------------------
+02 06:48:22.000000
```

Listing 4: Ermitteln der Datumsdifferenz über die Funktion numtodsinterval

```
with dates (d1,d2)
  as (select to_date('03.01.2018 08:15:27'
                     ,'dd.mm.yyyy hh24:mi:ss')
            ,to_date('05.01.2018 15:03:49'
                     ,'dd.mm.yyyy hh24:mi:ss')
        from dual)
select to_timestamp(to_char(d2,'dd.mm.yyyy hh24:mi:ss'))
     - d1 as Differenz
      from dates;
DIFFERENZ
-------------------
+02 06:48:22.000000
```

Listing 5: Ermitteln der Datumsdifferenz als Resultat einer Subtraktion von einem Timestamp.

Und hier wiederum die Frage – ließe sich dies nicht einfacher schreiben? *(siehe Listing 14)*.

Dies ist so zu lesen, dass, falls das Resultat der Konkatenation von l_string und Pipe ein Pipe sein sollte, dieses durch NULL ersetzt wird, bevor es mit dem Wert von l_value verkettet wird.

Dies sind nur fünf Features, die in den vergangenen Oracle-Versionen enthalten waren und viel zu selten eingesetzt werden. Es gäbe noch viele weitere, die ebenfalls mehr Aufmerksamkeit verdienten. Einige davon sind auf der DOAG-Konferenz im November zu sehen.

Über den Autor

Roger Troller ist Software-Entwickler bei der Finnova bankware AG in Lenzburg. Er ist dort in der Abteilung Datamanagement und Performance tätig, die sich um Basisfunktionalitäten der Finnova kümmert. Daneben engagiert er sich für die interne Weiterbildung der Mitarbeiter im SQL- und PL/SQL-Umfeld.

Die Listings 6 bis 14 finden Sie unter folgendem Link:
http://www.doag.org/go/redstack/201905_listings_troller

Roger Troller
roger.troller@finnova.com

Oracle Net 18c – Verschlüsselung ist keine Hexerei

Cornelia Heyde, Robotron Datenbank-Software

In vielen Bereichen ist eine verschlüsselte Verbindung aufgrund der heutigen Sicherheitsanforderungen erforderlich. Seit Oracle 10g unterstützt Oracle verschiedene Verschlüsselungsmethoden auch für Oracle SE/SE1/SE2 ohne eine Nutzung der Advanced-Security-Option. Der Weg zur verschlüsselten Verbindung kann wegen Anpassungen in den verschiedenen Konfigurationsdateien etwas aufwendiger sein. Neben den hier vorgestellten Methoden können auch mithilfe von Infrastrukturmitteln wie beispielsweise VPN gesicherte Verbindungen aufgebaut werden.
In diesem Artikel werden die beiden Verschlüsselungsmethoden der Oracle-Container-Datenbank detailliert vorgestellt.

Welche Netzwerkverschlüsselung ist die richtige?

Alle Befehle und Daten, die an die Oracle-Datenbank in einer Benutzersession gesendet oder empfangen werden, gehen im Klartext über das Netzwerk. Dabei gibt es eine Ausnahme: Die LOGIN-Informationen werden stets verschlüsselt übertragen. Alle anderen Daten bilden damit einen idealen Angriffspunkt. Mit geeigneten Netzwerktools kann eine bestehende Session eines Benutzers abgehört werden. Wenn das Netzwerk nicht gesichert ist, haben also potenzielle Angreifer die Chance, mit Netzwerk-Sniffern den Netzwerkverkehr abzuhören und zu analysieren. Um die zu übertragenen Daten abzusichern, muss entschieden werden, ob und wie überhaupt verschlüsselt werden soll.

Es gibt zwei von Oracle empfohlene Methoden, verschlüsselte Daten über das Netzwerk zu senden – die native und die SSL-Verschlüsselung. Beide Methoden verschlüsseln den Datenstrom, der über das Netz geht, sodass dieser nicht abge-

ENCRYPTION_TYPES_CLIENT	ENCRYPTION_TYPES_SERVER	Verschlüsselung
rejected	required	Keine Verbindung!
required	rejected	Keine Verbindung!
rejected	rejected | accepted | reuested	AUS
rejected | accepted | requested	rejected	AUS
accepted	accepted	AUS ← Default
requested	accepted | requested | required	AN
required	accepted | requested | required	AN
accepted | requested | required	requested	AN
accepted | requested | required	required	AN

Abbildung 1: Überblick möglicher Kombinationen der Verschlüsselung (Quelle: Cornelia Heyde)

hört und entschlüsselt werden kann. Die Konfiguration erfolgt bei beiden Methoden auf Server und Client, nur der Ort der Entschlüsselung der Daten unterscheidet sich. Bei der nativen Verschlüsselung mit Oracle Net wird die Verschlüsselung von Client und Server aufgelöst. Dagegen endet die Verschlüsselung bei der SSL-Verschlüsselung an der Netzwerkkarte.

Native Verschlüsselung

Eigenschaften der nativen Verschlüsselung

- ist schnell (ohne Betriebsunterbrechung) und einfach konfigurierbar
- die Entschlüsselung der Daten erfolgt durch interne Prozeduren der Datenbank und nicht über einen externen Algorithmus
- kann ohne weiteren Konfigurationsaufwand für Datenbankverbindungen wie SQL, OCI oder JAVA-Anwendungen eingesetzt werden

Eigenschaften der SSL-Verschlüsselung

- Benutzer kann eindeutig identifiziert werden
- die Entschlüsselung der Daten erfolgt extern an der Netzwerkkarte
- ist aufwendiger zu konfigurieren (z.B. Verwaltung von Zertifikaten)
- wird oft von Unternehmen eingesetzt, die SSL-Konfiguration und Zertifikatserver bereits in der Infrastruktur verwenden

Datenintegrität

Die Verschlüsselung des Netzwerkverkehrs kann nur der erste Schritt sein, denn die Überprüfung der korrekten Übertragung des Datenpaketes wird standardmäßig nicht überwacht. Die verschlüsselten Netzwerkpakete könnten aber während der Übertragung unbemerkt durch Angreifer verändert werden. Deshalb ist ein zusätzlicher Schutz der Datenintegrität der übertragenen Pakete notwendig. Aus diesem Grund sollte immer der Parameter CRYPTO_CHECKSUM_TYPES_[CLIENT|SERVER] in der sqlnet.ora konfiguriert werden. Mögliche Werte für diesen Parameter sind zum Beispiel SHA256 (Default) oder MD5.

Netzwerkverschlüsselung

Diese Verschlüsselung eignet sich als schneller Schutzmechanismus innerhalb einer DMZ, weil meist nicht die Absicherung des gesamten Netzwerkes erforderlich ist. Die Absicherung der Verbindung zwischen Datenbank-Server und zum Beispiel Webserver kann mit diesem Verfahren sehr schnell erfolgen. Die Einrichtung erfolgt ohne Unterbrechungen. Nach der Konfiguration werden alle neuen Verbindungen automatisch verschlüsselt. Für die native Verschlüsselung gibt es verschiedene Verschlüsselungsalgorithmen, die Oracle unterstützt, beispielsweise RC4_40, AES256 (siehe: „Oracle Database Net Services Reference“ und „Oracle Database Security Guide“). Weiterhin muss festgelegt werden, ob Server und/oder Client die Verschlüsselung fordern oder akzeptieren. Die Kombinationsmöglichkeiten sind in *Abbildung 1* gezeigt. Es gibt vier Werte für den Parameter:

SQLNET.ENCRYPTION_TYPES_(SERVER|CLIENT)

- rejected Verschlüsselung abgelehnt
- accepted Verschlüsselung, wenn der Partner das möchte
- requested Verschlüsselung wird gewünscht
- required Verschlüsselung wird gefordert

Für die genaue Analyse der Verbindung sollte das lokale Tracing in der sqlnet.ora für einen begrenzten Zeitraum einge-

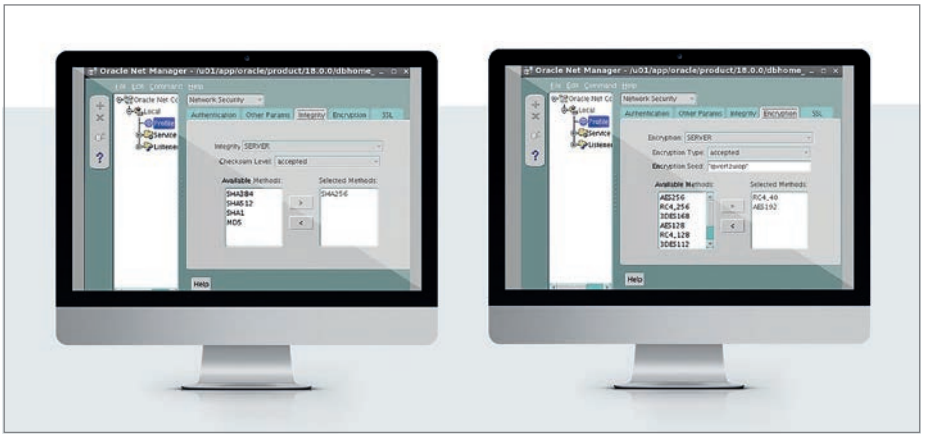


Abbildung 2: Oracle-Net-Manager-Konfiguration – Beispiel Server (Quelle: Cornelia Heyde)

```
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (SHA256)
SQLNET.ENCRYPTION_SERVER = requested
SQLNET.CRYPTO_SEED = 'qwertzuiop'
SQLNET.ENCRYPTION_TYPES_SERVER = (RC4_40, AES192)
```

Listing 1: Auszug sqlnet.ora zu Abbildung 2

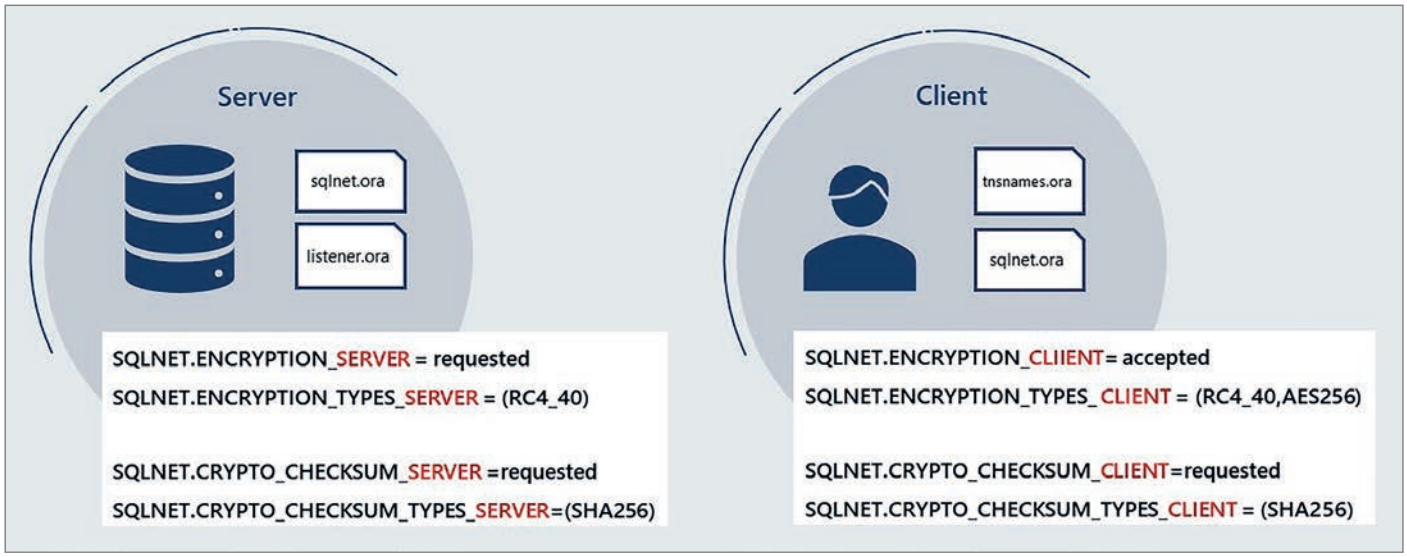


Abbildung 3: Beispiel Client- / Server-Konfiguration (Quelle: Cornelia Heyde)

schaltet werden. Dabei wird für jede Session eine Trace-Datei geschrieben.

Nach der Entscheidung für einen Algorithmus wird die Konfiguration für Client und Server durchgeführt. Für die Netzwerkkonfiguration können verschiedene Tools eingesetzt werden. Als grafische Tools können der Oracle Net Manager (netmgr) oder Enterprise Manager Cloud Control genutzt werden. Mit dem netmgr *(siehe Abbildung 2 und Listing 1)* muss der Parameter SQLNET.CRYPTO_SEED (Zeichenkette von 10-70 Zeichen, „Encryption Seed" im netmgr) konfiguriert werden, obwohl dieser nicht verwendet wird.

Alternativ kann die Konfiguration auch mit einem Editor erstellt werden. In der *Abbildung 3* werden die minimal notwendigen Parameter für Client und Server gezeigt.

Nach der Konfiguration der Parameter in der sqlnet.ora ist ein Neustart des Listeners zu empfehlen.

Eine Möglichkeit zu kontrollieren, wie die Kommunikation für Encryption konfiguriert ist, ist die Abfrage der aktuellen Session-Informationen *(siehe Listing 2)*.

Ein möglicher Fehler, der nach der Konfiguration auftreten kann, ist die Konfiguration von unterschiedlichen Verschlüsselungsmethoden, die inkompatibel sind. In diesem Beispiel ist der Parameter SQLNET.ENCRYPTION_TYPES_SERVER=(RC4_40) am Server und SQLNET.ENCRYPTION_TYPES_CLIENT=(AES128) am Client konfiguriert. Folgende Fehlermeldung erscheint *(siehe Listing 3)*.

Mithilfe des aktuellen Trace-Files einer verschlüsselten Benutzersession kann die Verschlüsselung nachgewiesen werden *(siehe Abbildung 4)*.

In der Dokumentation „JDBC Thin Driver Support for Encryption and Integrity" wird auf die Eigenschaft der JDBC-Verbindungen hingewiesen, dass die Default-Verschlüsselungs- und -Datenintegritätsmethoden sowohl für die Server- wie auch für die Client-Konfiguration auf „accepted" steht. Somit kann durch die Konfiguration von nur einer Seite die gewünschte Verschlüsselung erreicht werden. Der einfachste Weg, die Verschlüsselung auf mehreren Clients zu garantieren, ist die Angabe des Parameters „required" in der Server-Datei sqlnet.ora.

```
select substr(NETWORK_SERVICE_BANNER,1,45) NETWORK_BANNER
from v$session_connect_info
where sid ='12345'

NETWORK_BANNER
---------------------------------------------
TCP/IP NT Protocol Adapter for Linux: Version
Encryption service for Linux: Version 18.0.0.
RC4_40 Encryption service adapter for Linux:
Crypto-checksumming service for Linux:
```

Listing 2: Überblick, ob Encryption und Checksum für eine Session konfiguriert ist

```
SQL> connect system/oracle_4U@PDB1
ERROR:
ORA-12650: No common encryption or data integrity algorithm
```

Listing 3: Fehlermeldung bei unterschiedlichen Verschlüsselungsmethoden

SSL-Verschlüsselung

Im Folgenden wird die Konfiguration einer SSL-Verbindung (Secure Sockets Layer) zu einer CDB-Datenbank (SID=ORCL) mit einer PDB (SID=PDB1) gezeigt. Eine SSL-Verbindung benötigt ein Wallet, in dem die Zertifikate abgelegt werden. Server und Client benötigen jeweils ein gültiges Zertifikat, mit dessen Hilfe sich Client und Server jeweils gegenseitig authentifizieren. Ein Zertifikat ist Bestandteil des öffentlichen Teils des Schlüsselpaares, das beim Erstellen des Zertifikates angelegt wird. Der private Teil des Schlüssels bleibt im Wallet. Der öffentliche Schlüssel kann verteilt werden. Zertifikate werden von einer vertrauenswürdigen Zertifizierungsstelle ausgestellt. In diesem Beispiel werden selbst signierte Zertifikate verwendet.

```
Abfrage ohne Verschlüsselung (Auszug):
(9853824) [29-APR-2019 10:29:40:612] nsbasic_brc: 07 09 42 65 72 6E 73 74  |..Bernst|
(9853824) [29-APR-2019 10:29:40:612] nsbasic_brc: 65 69 6E 05 44 61 76 69  |ein.Davi|
(9853824) [29-APR-2019 10:29:40:612] nsbasic_brc: 64 08 06 00 77 B1 2C 00  |d...w.,.|

Abfrage mit Verschlüsselung (Auszug):
(2470673280) [30-APR-2019 14:02:12:581] nspsend: EA 60 C5 F5 44 03 EC 7E   |.`..D..~|
(2470673280) [30-APR-2019 14:02:12:581] nspsend: C7 DC F0 9D BC 11 92 6B   |.......k|
(2470673280) [30-APR-2019 14:02:12:581] nspsend: 25 82 49 09 68 BD CD DE   |%.I.h...|
```

Abbildung 4: Auszug aus der Session-Trace-Datei mit/ohne Verschlüsselung (Quelle: Cornelia Heyde)

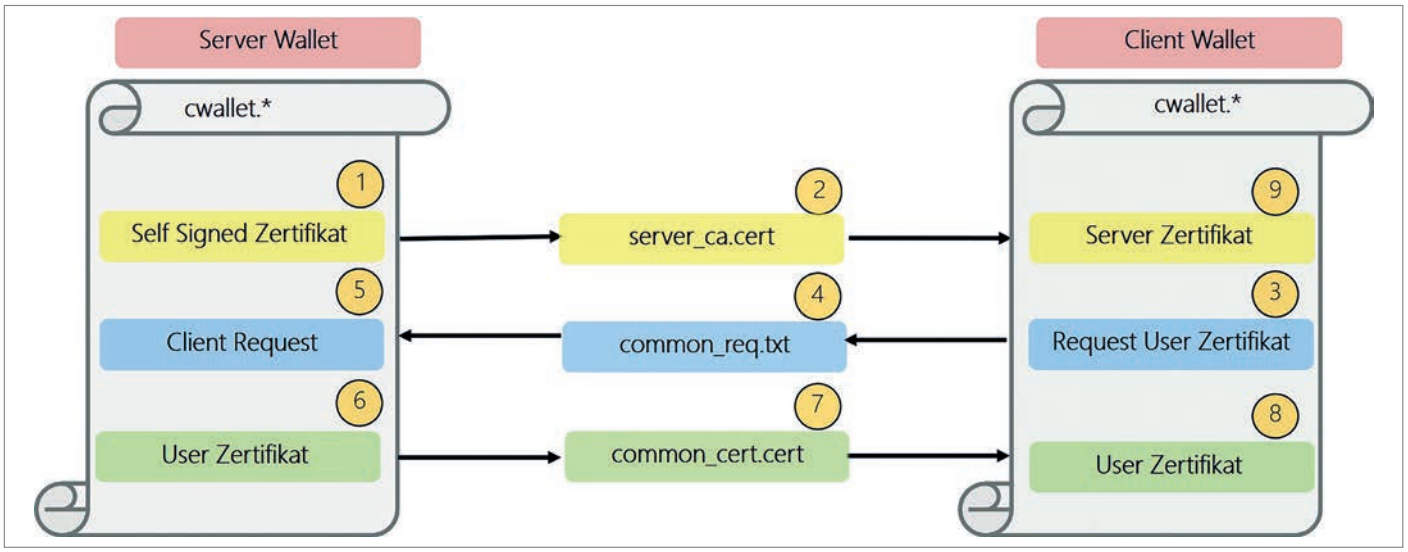


Abbildung 5: Überblick der Schritte zur Konfiguration der SSL-Verschlüsselung (Quelle: Cornelia Heyde)

Einen Überblick der Schritte am Server (host01) und Client (host02) zeigt *Abbildung 5*. Im Folgenden werden beide Seiten konfiguriert, sodass eine SSL-Verschlüsselung für die Client-Server-Verbindung im Netzwerk besteht.

Wallet erstellen auf dem Server

Das Oracle-Tool orapki kann in einer Oracle-Umgebung eingesetzt werden, um Wallets und Zertifikate zu verwalten. Der Oracle Wallet Manager (owm) ist die grafische Alternative für ausgewählte Schritte. Nachdem der Zielordner für das Wallet im Dateisystem angelegt wurde, wird ein leeres Wallet angelegt. Hierfür muss ein Passwort mit der Option „-pwd" vergeben werden, das aus Sicherheitsgründen immer vom Prompt abgefragt werden sollte. Für Demozwecke wird das Passwort im ersten Beispiel sichtbar angegeben *(siehe Abbildung 6)*. Weiterhin kann das Wallet für

```
1. SQL> connect system/oracle_4U@SSL_ORCL
ERROR:
ORA-12547: TNS:lost contact

--Ursache
SORCL =
. . .  (ADDRESS = (PROTOCOL = TCP)(HOST = host01 )(PORT = 1577)) . . .

2. SQL> connect system/oracle_4U@SSL_ORCL
ERROR:
ORA-28865: SSL connection closed
--Ursache
WALLET_LOCATION =
(SOURCE=   (METHOD=File)
      (METHOD_DATA= (DIRECTORY = /u01/app/oracle/WALLET_CLIENT )))
```

Abbildung 6: Wallet anlegen (Quelle: Cornelia Heyde)

„-auto_login" konfiguriert werden, sodass diese Verbindung ohne weitere Passwortabfragen erstellt werden kann, also ständig geöffnet ist. Im Zielordner werden vier Dateien erstellt.

Zertifikat erstellen oder hinzufügen

In diesem Beispiel wird ein self-signed Zertifikat für die Datenbank mit der SID = ORCL

```
$ orapki wallet add -wallet . -dn "cn=ORCL" -keysize 1024 -self_signed
-validity 1000

$ orapki wallet display -wallet .
Requested Certificates:
User Certificates:
Subject:      CN=ORCL
Trusted Certificates:
Subject:      CN=ORCL
```

Listing 4: Self-signed Zertifikat erzeugen und anzeigen

```
$ orapki wallet export -wallet . -dn "cn=ORCL" -cert server_ca.cert

$ cat server_ca.cert
-----BEGIN CERTIFICATE-----
MIIBoDCCAQkCEH+axpV7XzUPPyS4Si33ZK0wDQYJKoZIhvcNAQELBQAwETEPMA0G
A1UEAxMGT1JDTDEwMB4XDTE5MDUwNjE5MjczM1oXDTIyMDEzMDIwMjczM1owETEP
MA0GA1UEAxMGT1JDTDEwMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCML0ve
FLbXGGyfVwb/FR8LfVxOAqMZ0nJC3Puci3gw3dcHpcPSpbeWCzGH/TMuANXCCjKU
ibTCARxn+VkBI++hTzI8x+2sS4jmnLPCYWUmaFqCNyIJu4ujxHl3QCGgC5JxZpVj
cTIC2tAKXC7r+wTyHB+Y6KURqslrKwiK+S+9zwIDAQABMA0GCSqGSIb3DQEBCwUA
A4GBAC+8uMCeESMR5tMvyzk+L79+Z7mu3QKqDeBUq+dHAIXehLM8SI72fworxDXP
9GY4K19KczYBBvYl7LOFT02ku05ISwEpPUljT5U+y65sAYoItdhQDxkyVRfiZC2P
YUaomJfM5ax++SjRsCzx6g2y4mQ/7ZcdBbTjUHFKjR+gPZ//
-----END CERTIFICATE-----
```

Listing 5: Self-signed Zertifikat exportieren und anzeigen

```
LISTENER =
(DESCRIPTION_LIST =
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=host01)(PORT=1577)))
)
WALLET_LOCATION =
(SOURCE= (METHOD=File)
(METHOD_DATA=(DIRECTORY=/u01/app/oracle/admin/WALLET_SERV)))
```

Listing 6: Konfiguration listener.ora

```
$ cat sqlnet.ora
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
SQLNET.AUTHENTICATION_SERVICES= (BEQ, TCPS)
SSL_CLIENT_AUTHENTICATION=TRUE
WALLET_LOCATION =
   (SOURCE= (METHOD=File)
   (METHOD_DATA=(DIRECTORY=/u01/app/oracle/admin/WALLET_SERV)))
```

Listing 7: Konfiguration sqlnet.ora

erzeugt *(siehe Abbildung 5, Schritt 1)*. Hier können auch geeignete Zertifikate von der Zertifizierungsstelle aus der Infrastruktur eingesetzt werden. Wenn der orapki-Befehl direkt im Ordner des Wallet-Pfads abgesetzt wird, muss der komplette Verzeichnispfad nicht angegeben werden, stattdessen wird mit „." auf das aktuelle Verzeichnis verwiesen. Ansonsten muss bei jedem Statement der Parameter „-wallet" mit dem kompletten Speicherort angegeben werden. Die Schlüsseltiefe des Zertifikats kann 512, 102 oder 2048 Bit sein. Der Parameter validity (mandatory) gibt die Gültigkeit des Zertifikats in Tagen an *(siehe Listing 4)*. Das gerade erstellte Zertifikat kann nun mithilfe von orapki angezeigt werden. Das self-signed-Zertifikat besteht aus den beiden Teilen „Requested Certificates" und „Trusted Certificates".

Export Server-Zertifikat in eine Datei

Damit das Server-Zertifikat (öffentlicher Schlüssel) auch im Wallet (Zertifikatspeicher) des Clients vorhanden ist, muss dieses in eine Datei exportiert *(siehe Abbildung 5, Schritt 2)* werden. Danach ist der erste Schritt abgeschlossen. Jetzt kann das Server- Zertifikat angezeigt werden *(siehe Listing 5)*.

Konfiguration der Oracle-Net-Dateien auf dem Server

Für die Kommunikation müssen die Oracle-Net-Dateien sqlnet.ora und listener.ora auf dem Server angepasst werden. In der Datei listener.ora wird zunächst der Speicherort des Wallet angegeben. Für die SSL-Kommunikationen werden weiterhin ein Eintrag für das Protokoll TCPS sowie der definierte und in der Infrastruktur frei geschaltete Port 1577 hinzugefügt *(siehe Listing 6)*.

```
SYS@ORCL> alter system set local_listener='(ADDRESS=(PROTOCOL=tcps)(HOST=host01)(PORT=1577))' scope=both;

$ lsnrctl status
   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=host01)(PORT=1577)))
. . .
Service "ORCL" has 1 instance(s).
   Instance "ORCL", status READY, has 1 handler(s) for this service
Service "ORCLXDB" has 1 instance(s).
   Instance "ORCL", status READY, has 1 handler(s) for this service
Service "PDB1" has 1 instance(s).
   Instance "ORCL", status READY, has 1 handler(s) for this service
```

Listing 8: Abfrage Local Listener (Auszug)

Der Parameter SSL_CLIENT_AUTHENTICATION wird angegeben, damit sich ein Client mithilfe SSL authentifizieren kann. Der Default-Parameter ist TRUE *(siehe Listing 7)*.

Ist der Speicherort des Wallet nicht in der sqlnet.ora angegeben, wird seit Oracle 12c automatisch die WALLET_LOCATION aus der listener.ora verwendet. Nach diesen Konfigurationsanpassungen muss der Listener neu gestartet werden.

Anpassungen in der Datenbank

Damit sich die Datenbank auch an dem neu gestarteten Listener für SSL registrieren kann, muss nun der Parameter LOCAL_LISTNER in der Datenbank geändert werden *(siehe Listing 8)*. Nach dieser Anpassung sollte die Kontrolle erfolgen, ob sich die Datenbank auch am SSL-Listener registriert. Bei Bedarf ist ein Neustart der Datenbank erforderlich.

In *Listing 9* sind die Empfehlungen von Oracle für die Multitenant-Datenbank zu sehen. Nach dem Neustart muss ein „common"-Benutzer für die SSL- Authentifizierung eingerichtet werden.

Für einen Test am Server wird noch der Eintrag für CDB und PDB1 in der tnsnames.ora mit dem Protokoll TCPS und dem dazugehörigen Port hinzugefügt *(siehe Listing 10)*. Wenn alles passt, kann es mit der Client-Seite weitergehen.

Wallet erstellen auf dem Client

Auch hier kommt das Oracle-Tool orapki zum Einsatz. Die einzelnen Schritte sind ähnlich denen auf dem Server. Nach dem Anlegen des Zielordners im Dateisystem wird auch hier ein leeres Wallet mit auto_login angelegt *(siehe Listing 11)*.

Erstellen der Anfrage für ein User-Zertifikat

Damit der Client sich mit dem Server verbinden kann, benötigt er ein gültiges Zertifikat für die Verbindung zur Datenbank. Nach der Erstellung des Zertifikates für den Benutzer „common" kann der Inhalt „Requested Certificates" überprüft werden. Aus diesem gerade angelegten Client-Zer-

```
SYS@ORCL>connect system/oracle_4U@ORCL
alter system set common_user_prefix='' scope=spfile;
alter system set os_authent_prefix='' scope=spfile;
alter system set remote_os_authent=FALSE scope=spfile;
shutdown immediate
startup
create user common identified externally as 'cn=common' container=all;
grant connect to common container=all;
```

Listing 9: Konfiguration der Container-Datenbank

```
SSL_ORCL=
 (DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=host01)(PORT=1577))
 (CONNECT_DATA= SERVER=DEDICATED)(SERVICE_NAME= ORCL)))
SSL_PDB1=
 (DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=host01)(PORT=1577))
 (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME= PDB1)))

SQL>connect system/oracle_4U@SSL_ORCL
Connected.
SQL>connect system/oracle_4U@SSL_PDB1
Connected.
```

Listing 10: Testen der TCPS-Verbindung

```
$ mkdir /u01/app/oracle/admin/WALLET_CLIENT
$ orapki wallet create -wallet . -auto_login
```

Listing 11: Wallet für den Client erstellen

```
$ orapki wallet add -wallet . -dn "cn=common" -keysize 1024

$ orapki wallet display -wallet .
Requested Certificates:
Subject:      CN=common
User Certificates:
Trusted Certificates:

$ orapki wallet export -wallet . -dn "cn=common" -request common_req.txt
```

Listing 12: Zertifikat erstellen, Inhalt Wallet und Zertifikatanfrage exportieren

```
$ scp common_req.txt host01:/u01/app/oracle/admin/WALLET_SERV
$ orapki cert create -wallet . -request common_req.txt -cert common_
cert.cert -validity 1000
$ scp common_cert.cert host02:/u01/app/oracle/admin/WALLET_CLIENT
```

Listing 13: Signiertes Zertifikat erstellen und auf Client kopieren

```
$ orapki wallet add -wallet . -trusted_cert -cert server_ca.cert

$ scp server_ca.cert host02:/u01/app/oracle/admin/WALLET_CLIENT
$ orapki wallet add -wallet . -trusted_cert -cert server_ca.cert

$ orapki wallet display -wallet .
Requested Certificates:
Subject:      CN=common
User Certificates:
Trusted Certificates:
Subject:      CN=ORCL
Subject:      CN=common
```

Listing 14: Zertifikate in Client Wallet importieren und anzeigen

```
SSL_CLIENT_AUTHENTICATION = TRUE
WALLET_LOCATION =
(SOURCE = (METHOD = FILE)
(METHOD_DATA=(DIRECTORY=/u01/app/oracle/admin/WALLET_CLIENT)))

SSL_ORCL =
(DESCRIPTION =
 (ADDRESS=(PROTOCOL=TCPS)(HOST=host01)(PORT=1577))
 (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCL)))
SSL_PDB1 =
(DESCRIPTION =
 (ADDRESS=(PROTOCOL=TCPS)(HOST=host01)(PORT=1577))
 (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=PDB1)))
```

Listing 15: sqlnet.ora und tnsnames.ora beim Client erweitern

```
$ tnsping SSL_ORCL
Attempting to contact (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)
(PORT=1577)) (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=ORCL)))

SYS>connect system/oracle_4U@SSL_ORCL
SYSTEM@SSL_ORCL>select INSTANCE_NAME,HOST_NAME from v$instance;

INSTANCE_NAME HOST_NAME
------------------------------
ORCL          host01
```

Listing 16: Test der Verbindung

tifikat wird die Zertifikatanforderung (auch hier der öffentliche Schlüssel) erstellt und in einer Datei gespeichert *(siehe Abbildung 5, Schritt 3)*. Mit diesem Benutzer kann später eine Datenbank-Verbindung ohne Passwort aufgebaut werden *(siehe Listing 12)*.

Bestätigen der Anfrage des Servers

Nach dem Kopieren der Zertifikatanforderung auf dem Server wird diese Anforderung bestätigt; das zertifizierte Benutzerzertifikat wird in einer neuen Datei angelegt und auf den Client kopiert *(siehe Abbildung 5, Schritt 3-7 und Listing 13)*.

Zertifikat am Client

Im letzten Schritt wird das signierte Benutzer-Zertifikat in das Wallet importiert. Jetzt fehlt für eine vertrauenswürdige Verbindung noch das Server-Zertifikat im Client Wallet. Das Zertifikat wird vom Server kopiert und in das Client Wallet importiert *(siehe Abbildung 5, Schritt 9)*. Mit dem Kommando „orapki wallet display" erfolgt die Kontrolle, ob sich beide Zertifikate im Wallet im Bereich „Trusted Certificates" befinden *(siehe Listing 14)*.

Anpassung am Client

Der Wallet-Speicherort und die Erlaubnis der SSL-Verbindung durch den Client werden in der sqlnet.ora konfiguriert. Außerdem wird die tnsnames.ora um die Verbindungsdaten für SSL ergänzt *(siehe Listing 15)*.

Testen der Verbindung

Jetzt ist der Test der neuen SSL-Verbindung erfolgreich *(siehe Listing 16)*.

In diesem Beispiel wird nur eine einzige Verbindung getestet. In der Praxis müssen für alle Clients die Verbindungen überprüft werden. Für die Konfiguration der anderen Clients reicht das Kopieren des Wallet mit den Zertifikaten und den Konfigurationsdateien. Die Dateien ewallet.p12, cwallet.sso, sqlnet.ora und tnsnames.ora müssen in die Konfiguration des Zielhosts (Client) kopiert werden. Als nächster Schritt den Parameter WALLET_LOCATION anpassen und den Parameter TNS_ADMIN konfigurieren, der dann auf das gerade erstellte Verzeichnis zeigen muss.

Abschließend sollen noch zwei typische Fehler gezeigt werden *(siehe Abbildung 7)*. Im ersten Beispiel ist das falsche Protokoll in der tnsnames.ora eingetragen. Im zweiten Beispiel ist der Pfad zum Wallet fehlerhaft.

Fazit

Verschlüsselte Datenbank-Verbindungen können in allen Oracle Releases lizenzfrei

```
$ mkdir /u01/app/oracle/admin/WALLET_SERV

$ orapki wallet create -wallet /u01/app/oracle/admin/WALLET_SERV -auto_login -pwd <WalletPassword>

Operation is successfully completed.

[oracle@host01]$ ll /u01/app/oracle/admin/WALLET_SERV
-rw------- 1 oracle oinstall 194  6. Mai 10:25 cwallet.sso
-rw------- 1 oracle oinstall   0  6. Mai 10:25 cwallet.sso.lck
-rw------- 1 oracle oinstall 149  6. Mai 10:25 ewallet.p12
-rw------- 1 oracle oinstall   0  6. Mai 10:25 ewallet.p12.lck
```

Abbildung 7: Mögliche Fehler (Quelle: Cornelia Heyde)

konfiguriert werden. Nur der Aufwand ist je nach gewählter Methode unterschiedlich. Für die Konfiguration sind keine weiteren Tools notwendig. Nur für die SSL- Konfiguration wird auf der Kommandozeile orapki zur Verwaltung der Wallets verwendet. Ein wichtiger Aspekt zur Auswahl der Verschlüsselungsvariante ist die Frage, wie viele Clients eine verschlüsselte Verbindung nutzen werden. Ist nur die Absicherung der Kommunikation zwischen Webserver und Datenbank-Server gefordert oder soll eine Vielzahl von Clients geschützt werden?

Die native Verschlüsselung ist einfach und schnell von einem DBA zu konfigurieren. Dagegen ist die SSL-Verschlüsselung zwar die sicherere Verbindung, aber etwas aufwendiger zu konfigurieren. Zusätzlich zur DBA-Tätigkeit wird die Unterstützung durch einen Netzwerkadministrator zur Portfreischaltung oder bei der Erstellung der Infrastrukturzertifikate benötigt.

Quellen

[1] HEINZ-WILHELM FABRY,2014: ORACLE DOJO #10, Daten verschlüsseln, Oracle

[2] Oracle-Dokumentation: *https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/index.html*

[3] My ORACLE Support: *https://support.oracle.com*

Cornelia Heyde
Cornelia.Heyde@robotron.de

Multitenant – Cloning von Datenbanken

Julian Frey, Edorex

Sei es für Tests, die Entwicklung oder die Fehlersuche – die Gründe für das Klonen einer Datenbank sind vielfältig. Leider verursacht diese Arbeit meist viel Aufwand, gerade auch für den Datenbank-Administrator. Für diesen gehören die Cloning-Tasks meist zum Daily Business.
Mit der ab Oracle 12c verfügbaren Multitenant-Architektur haben sich die Möglichkeiten auch in diesem Bereich geändert und zum Teil stark vereinfacht. In diesem Artikel werden die neuen Möglichkeiten zum Klonen einer Datenbank beschrieben.

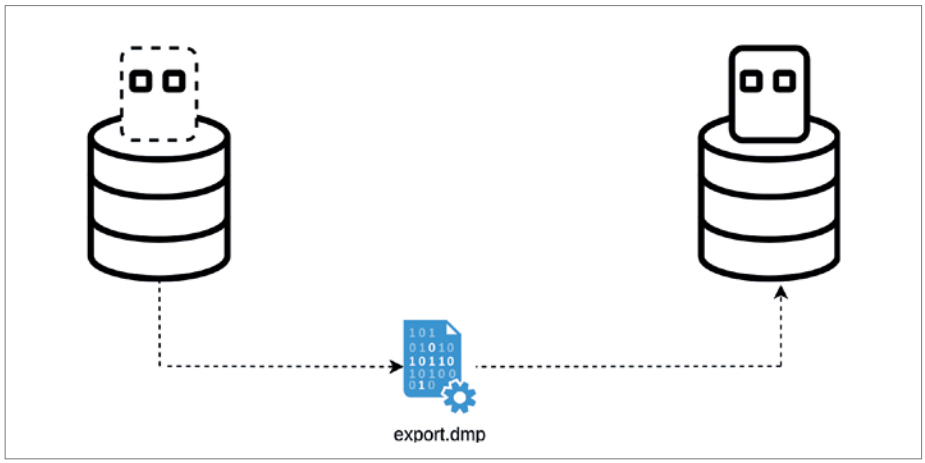


Abbildung 1: Schematische Darstellung Oracle Data Pump (Quelle © Julian Frey)

Weshalb klonen wir Datenbanken und welche Auswirkungen haben die Gründe auf das Klonen?

Je nach Umfeld sind sowohl die Gründe als auch die Anforderungen an einen Klon sehr unterschiedlich und unter Umständen sogar widersprüchlich.

Als DBA möchte man vielleicht eine komplette Datenbank klonen, um Fehler zu reproduzieren, oder man wird vom Oracle Support aufgefordert, einen Parameter anzupassen. Beides möchte man nicht auf der produktiven Datenbank ausführen. Bei Änderungen an den Datenstrukturen wie bei einem Wechsel von Basic File Lobs auf Secure File oder beim Verschieben von Tabellen kann es wichtig sein, dass sowohl die Art als auch der Ablageort der Daten identisch bleibt.

Ein Entwickler möchte nur ein Sub-Set der Daten auf einer Test- oder Entwicklungsdatenbank haben, um bestimmte Statements oder Abläufe zu testen. Es kann auch sein, dass nur die Metadaten zur Verfügung gestellt werden sollen, ohne Daten.

Bei Vorgängen, bei denen die direkte Speicherung der Daten keine übergeordnete Rolle spielt, muss nicht zwangsweise eine Kopie der vollen Datenbank erstellt werden.

```
SQL> alter pluggable database begin backup;

Pluggable database altered.

SQL> exec dbms_pdb.describe('/home/oracle/DEMO_ROOT.xml','DEMO_ROOT');

PL/SQL procedure successfully completed.

SQL> alter pluggable database end backup;

Pluggable database altered.
```

Listing 1: Begin/End Backup einer PDB

```
SQL> CREATE PLUGGABLE DATABASE PDBTGT
AS CLONE USING '/home/oracle/DEMO_ROOT.xml' NOCOPY
TEMPFILE REUSE;

CREATE PLUGGABLE DATABASE PDBTGT
*
ERROR at line 1:
ORA-65368: unable to open the pluggable database due to errors during recovery
ORA-65089: pluggable database is not clean
ORA-01113: file 41 needs media recovery
ORA-01110: data file 41: '/u02/oradata/TDB193/pdbtgt/undotbs01.dbf'

SQL> show pdbs

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

Listing 2: Erstellen einer Pluggable Datenbank aus einem Cold Backup

Aufbau der verwendeten Umgebung

In den Beispielen wurde eine Oracle 19.3-Datenbank Enterprise Edition mit Multitenant-Option auf einem Oracle Linux Server Release 7.6 verwendet.

Der Name der CDB ist TDB193, die Source PDB, von der geklont wird, heißt immer DEMO_ROOT. Wo spezifische Parameter- oder sonstige Änderungen gemacht wurden, werden diese explizit erwähnt.

Wie verhalten sich die bekannten Möglichkeiten, um Daten zu klonen?

Weil in den meisten Unternehmungen bereits Klonprozesse etabliert sind, ist es wichtig zu wissen, welche bestehenden Abläufe sich ändern oder welche nahezu identisch bleiben.

Oracle Data Pump:
Exportieren und Importieren von Daten mittels Oracle Data Pump entspricht eigentlich nicht dem klassischen Bild von einem Klon. Es ist jedoch eine weitverbreitete Technologie, um Daten zu transferieren. Oracle Data Pump bietet darüber hinaus diverse Optionen und Möglichkeiten, die mit anderen Tools wie zum Beispiel RMAN nicht gegeben sind. Falls sich die Daten zwischen der Ursprungsdatenbank und der Zieldatenbank unterscheiden sollen (z.B. Datasubsetting, Datamasking), ist Oracle Data Pump die einfachste Methode, um dies zu bewerkstelligen *(siehe Abbildung 1)*.

Beim Transport der Daten durch Oracle Data Pump hat sich in Bezug auf

die Multitenant-Architektur nichts geändert. Die einzige Voraussetzung hierfür ist, dass man sich an der Pluggable Datenbank (PDB) anmeldet und nicht an der CDB. Dies bedeutet, dass mit der Multitenant-Architektur keine Oracle Data Pump Jobs mit „/ as sysdba" mehr möglich sind.

Cold Backup:
Unter dem Begriff Cold Backup versteht man die Sicherung einer Datenbank im geschlossenen Zustand. Bei einer Oracle-Datenbank kann dies auch bedeuten, dass die Datenbank für die Sicherung in den Backup-Modus versetzt wird. Diese Backup-Methode wird häufig bei Storage-Technologien verwendet, die Snapshots auf Speicherebene durchführen. Mit diesen Snapshots können auch Datenbanken geklont werden. Ein Vorteil dieser Klonmechanismen ist, dass ein Snapshot nur einen Bruchteil (die Änderungen) der Daten der Ursprungsdatenbank beinhaltet und daher wesentlich schneller beim Erstellen ist.

Diese Mechanismen kann man auch bei der Multitenant-Architektur verwenden. Der Klonvorgang benötigt hier jedoch einen manuellen Eingriff.

Als Erstes muss die PDB in den Backup-Modus versetzt werden. Dies kann nur innerhalb der PDB mit einem Benutzer mit SYS<DBA/BACKUP>-Privilegien geschehen. Anschließend kann ein Snapshot der Pluggable-Datenbank-Dateien erstellt werden. Um die Beschreibung der Pluggable Datenbank zu erhalten, muss für diese DB eine beschreibende XML-Datei erstellt werden. Zu diesem Zweck steht das Package DBMS_PDB mit der Procedure DESCRIBE zur Verfügung *(siehe Listing 1)*.

Nach dem Snapshot der Dateien und dem Erstellen der XML-Datei zur Beschreibung der Datenbank kann der Backup-Modus beendet werden.

Nun müssen in der XML-Datei die Pfade für die Snapshots der Dateien angepasst werden. Nach dem Anpassen der Pfade kann die Kopie, respektive der Snapshot der Pluggable Datenbank, an die ursprüngliche oder an eine andere CDB angehängt werden *(siehe Listing 2)*.

Da die Datenbankdateien beim Kopieren und Sichern nicht geschlossen waren, erhalten wir den Fehler, dass die Datenbankdateien ein Media Recovery benötigen. Weil sich aber die Pluggable Datenbank während des Sicherns im Backup-Modus befand, sind die Dateien an sich konsistent und können mittels RMAN „recovered" werden. Die Pluggable Datenbank befindet sich zu diesem Zeitpunkt im Status MOUNTED und es muss keine Statusänderung vorgenommen werden.

Nach dem Recovery kann die Pluggable Datenbank geöffnet werden und unser Klon, basierend auf einem Storage-Snapshot, ist erstellt *(siehe Listing 3)*.

Falls die Pluggable Datenbank geschlossen oder im Read-Only-Modus geöffnet werden kann, wird dasselbe Verfahren angewendet. Dabei wird jedoch kein Recovery der geklonten Pluggable Datenbank benötigt.

```
oracle@localhost.localdomain:~/ [TDB01] rman target /

Recovery Manager: Release 18.0.0.0.0 - Production on Mon Feb 18 21:10:03 2019
Version 18.5.0.0.0

Copyright (c) 1982, 2018, Oracle and/or its affiliates. All rights reserved.

connected to target database: TDB01 (DBID=2948586735)

RMAN> recover pluggable database PDBTGT;

Starting recover at 18.02.2019 21:10:12
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=142 device type=DISK
datafile 40 not processed because file is offline
datafile 41 not processed because file is offline

starting media recovery
media recovery complete, elapsed time: 00:00:01

Finished recover at 18.02.2019 21:10:14

RMAN>
SQL> alter pluggable database PDBTGT open;

Pluggable database altered.

SQL> show pdbs

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

Listing 3: Recovery einer geklonten PDB aus einem Cold Backup

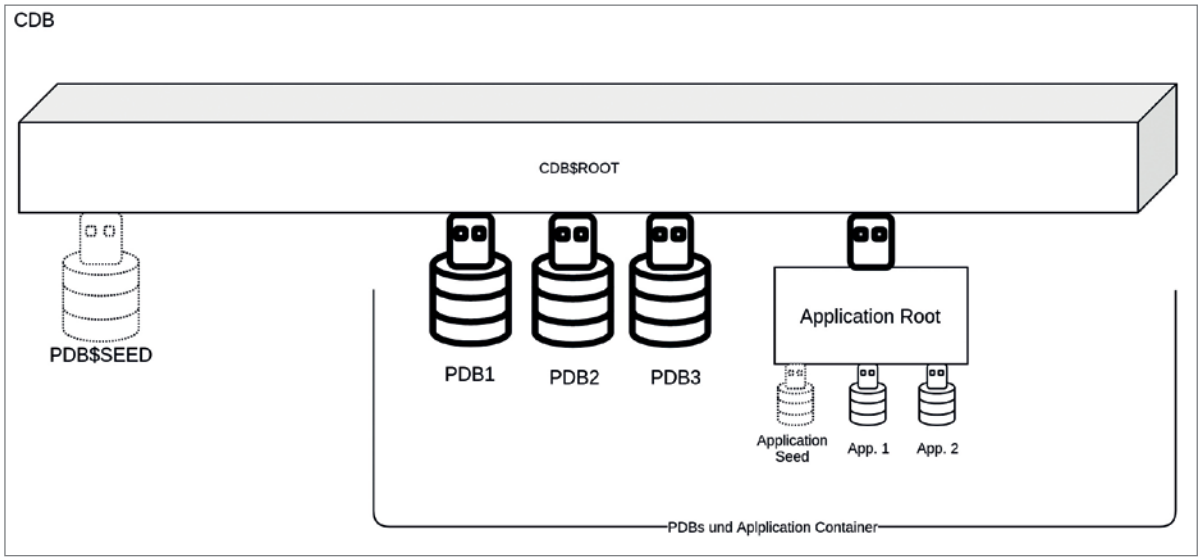


Abbildung 2: Containers in a CDB (Quelle: [2])

RMAN Duplicate:
Das Klonen einer CDB unterscheidet sich nicht vom klassischen Klonen einer Single-Instance-Datenbank. Mit der Multitenant-Architektur hat man aber weitere Möglichkeiten, die Auxiliary-Datenbank zu konfigurieren. Einerseits kann eingeschränkt werden, welche PDBs im Klon vorhanden sein sollen, und andererseits können auch die einzelnen PDBs verändert werden. So kann unter anderem für jede Pluggable Database angegeben werden, welche Tablespaces geklont werden sollen *(siehe unter https://docs.oracle.com/en/database/oracle/oracle-database/19/rcmrf/DUPLICATE.html#GUID-E13D8A02-80F9-49A2-9C31-92DD3A795CE4).*

Beim Klonen von Pluggable Datenbanken in eine bestehende CDB gibt es aktuell noch diverse Restriktionen:

1. Es kann nur von einer aktiven Datenbank geklont werden.
2. Es können nur folgende Klauseln beim DUPLICATE-Kommando verwendet werden: NORESUME, DB_FILE_NAME_CONVERT, SECTION SIZE, USING COMPRESSED BACKUPSET.
3. Die folgenden Klauseln sind beim DUPLICATE-Kommando nicht unterstützt: SPFILE, NO STANDBY, FARSYNC STANDBY, LOG_FILE_NAME_CONVERT.
4. Es kann kein DUPLICATE in eine Standby-Container-Datenbank ausgeführt werden.
5. Es kann pro Klonvorgang nur eine PDB geklont werden.
6. Es kann nur die komplette PDB geklont werden, es können zum Beispiel keine Tablespaces ausgeschlossen werden.
7. Es kann keine Non-CDB-Datenbank in eine bestehende CDB geklont werden.
8. Mit Transparent Data Encryption (TDE) verschlüsselte Datenbanken können nicht geklont werden. *[1]*

Was gibt es für neue Möglichkeiten seit 12c?

Seit der Einführung der Multitenant-Architektur hat sich das Erstellen einer Datenbank grundlegend geändert. Es wird nicht mehr wie bisher eine Datenbank erstellt, sondern immer eine Kopie einer bestehenden Pluggable Datenbank. Die einfachste Art eines Klons ist ein Klon der PDB$SEED *(siehe Abbildung 2)*.

Eine Pluggable Datenbank kann nicht nur von der PDB$SEED erstellt werden, sondern auch von diversen anderen Quellen.

Quellen können sowohl Pluggable Datenbanken sein, die in derselben Container-Datenbank liegen, als auch PDBs und Non-CDBs, die über einen Datenbanklink mit der Zieldatenbank verbunden sind.

```
SQL> SELECT PDB_NAME, CLONED_FROM_PDB_NAME FROM DBA_PDB_HISTORY;

PDB_NAME             CLONED_FROM_PDB_NAME
-------------------- --------------------
PDBP1                PDB$SEED
PDBP2                PDB$SEED
PDBP3                PDB$SEED
PDBP4                PDBP1
PDBP5                PDBP1
PDBP6                PDBP1@CLONE_LINK
```

Listing 4: DBA_PDB_HISTORY mit einem Klon über DB_LINK

```
SQL> CREATE PLUGGABLE DATABASE PDBP2 FROM PDBP1 SNAPSHOT COPY;
```

Listing 5: Snapshot Copy einer PDB

```
ALTER PLUGGABLE DATABASE MATERIALIZE;
```

Listing 6: Materialisieren einer Snapshot Copy PDB

Um zu prüfen, wann welche PDB von welcher Quelle erstellt wurde, kann die View DBA_PDB_HISTORY abgefragt werden.

In diesem Beispiel wurde die PDBP6 über einen DB_LINK namens CLONE_LINK von der Pluggable Datenbank PDBP1 erstellt *(siehe Listing 4)*.

Das Kommando CREATE PLUGGABLE DATABASE bietet nicht nur die Möglichkeit, vollständige Kopien einer Datenbank oder Pluggable Datenbank zu erstellen, sondern auch nur teilweise Kopien oder Snapshot-Kopien. Die Möglichkeiten, die seit 12.1 zur Verfügung stehen, wurden mit jedem neuen Release erweitert. Neu in der Version 19c ist zum Beispiel die Möglichkeit, Pluggable Datenbanken zwischen verschlüsselten und nicht verschlüsselten Container-Datenbanken zu klonen.

Um zu prüfen, welche Möglichkeiten in der verwendeten Oracle-Version verfügbar sind, bietet Oracle die Database-Features-Applikation. Mit dieser können die entsprechenden Optionen und Verfügbarkeiten geprüft werden *(siehe https://apex.oracle.com/database-features/)*. Jedoch gilt es in dieser Applikation zu beachten, dass nicht erwähnt wird, unter welcher Lizenz und Plattform ein Feature verfügbar ist. Diverse Features sind zum Beispiel nur auf Engineered Systems (Exadata) und in der Oracle Cloud verfügbar.

Falls man die Datenbankendateien entweder auf einem ACFS-Dateisystem oder auf einem mittels Direkt NFS gemounteten NFS Share gespeichert hat, gibt es die Möglichkeit, PDBs als Snapshot direkt aus der Datenbank zu erstellen. Bedingung ist, dass das Filesystem eine Snapshot-Funktion anbietet. Oracle nutzt dann die vom Dateisystem zur Verfügung gestellte Snapshot-Funktion.

Vorsicht bei der Snapshot-Terminologie! Leider wurde derselbe Name für unterschiedliche Dinge verwendet, was zu Verwirrungen führen kann. In der Dokumentation zu Oracle 19c findet sich eine Tabelle, die erläutert, was sich hinter welchem Snapshot-Begriff verbirgt *[3]*.

Falls mit der Klausel „AS SNAPSHOT COPY“ eine PDB erstellt wird, wird ein Snapshot auf dem Speichersystem erstellt *(siehe Listing 5)*.

Dieser Snapshot der Pluggable Datenbank existiert nur als Sparse File und kann daher nicht „unplugged“ werden. Die Sparse Files enthalten nur die Änderungen gegenüber den Originaldateien. Falls dies notwendig ist, muss dieser zuerst materialisiert werden *(siehe Listing 6)*.

Beim Kommando „ALTER PLUGGABLE DATABASE SNAPSHOT“ wird ein Snapshot der aktuellen Pluggable Datenbank erstellt, dies hat nichts mit dem zuvor beschriebenen „SNAPSHOT COPY“-Befehl zu tun. Dies ist ein manuell erstellter Snapshot, von dem Kopien der Pluggable Database erstellt wer-

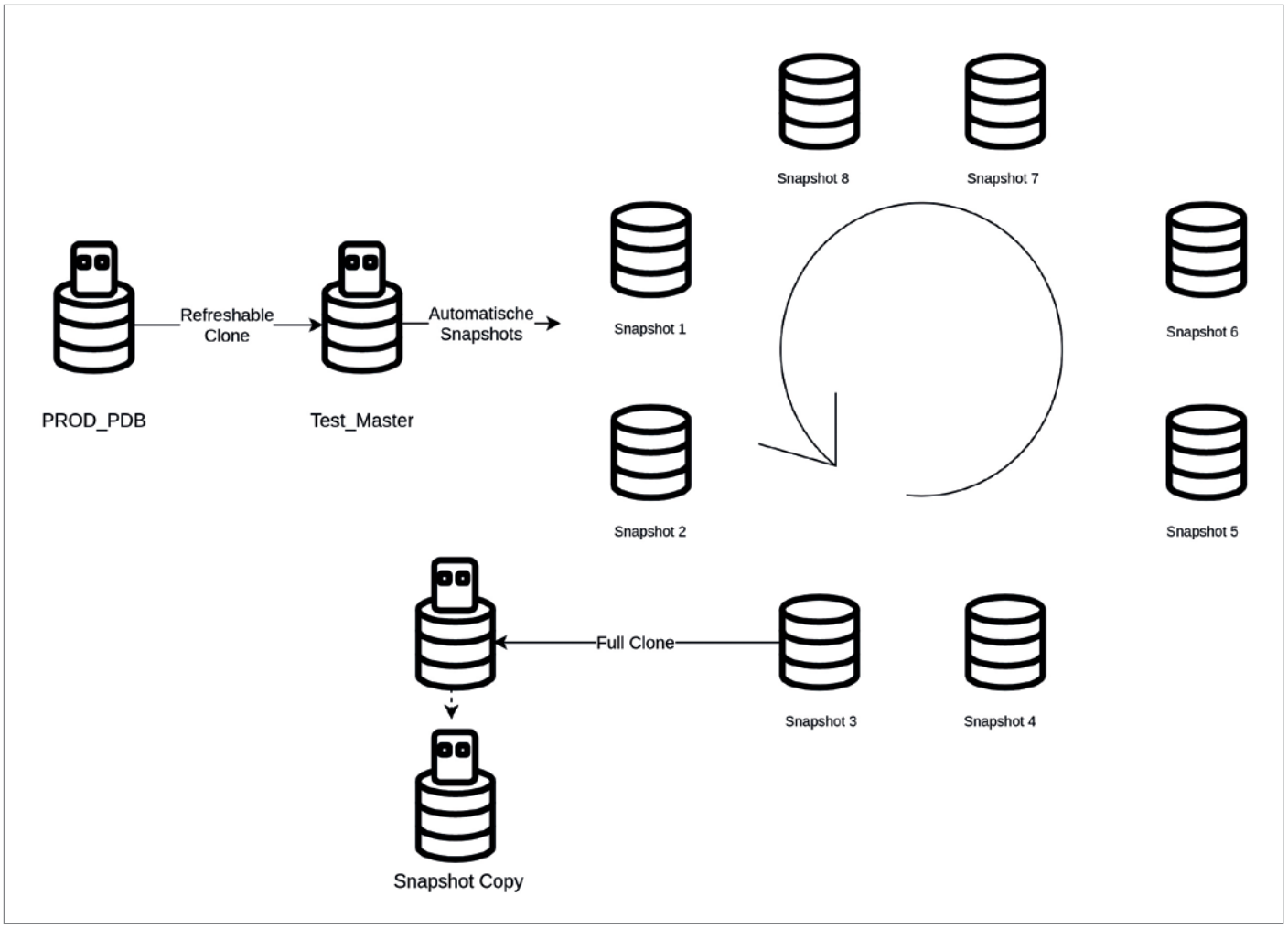


Abbildung 3: Automatic Snapshots of a Refreshable Clone PDB (Quelle © https://docs.oracle.com/en/database/oracle/oracle-database/18/multi/img/multi_pb_001a1.png)

```
oracle@dbserver: unzip -l snap_844695344_3023917.pdb
Archive: snap_844695344_3023917.pdb
Length     Date       Time   Name
---------  ---------- -----  ----
304087040  02-20-2019 08:03  o1_mf_system_g6wm9w1z_.dbf
393216000  02-20-2019 08:03  o1_mf_sysaux_g6wm9w20_.dbf
424673280  02-20-2019 08:03  o1_mf_undotbs1_g6wm9w21_.dbf
7063       02-20-2019 08:04  /u02/oradata/TDB01/snap_844695344_3023917.xml
---------                    -------
1121983383                   4 files
```

Listing 7: Inhalt eines PDB-Snapshots

den können. Dieselbe Art Snapshot wird auch bei dem neuen Feature „Snapshot Carousel" *[4]* verwendet.

Ein Snapshot Carousel erstellt anhand eines definierten Intervalls und in der definierten Anzahl Snapshots einer Pluggable Datenbank. Von diesen Snapshot können anschließend Kopien der Datenbank erstellt werden.

Mit der Datenbankversion 19c wurde diese Option dahingehend erweitert, dass, wenn der Parameter CLONEDB auf TRUE gesetzt ist, nach Möglichkeit Sparse-Kopien der Pluggable Datenbank erstellt werden. Falls der Parameter auf FALSE gesetzt ist, werden volle Kopien der Pluggable Datenbank als Archive-Datei mit der Endung „.pdb" erstellt. Diese Archive-Dateien sind ZIP-Dateien, die Kopien der Datafiles, die für ein Recovery benötigten archivierten Online Redo Logs sowie eine XML-Datei (analog der Unplugg XML-Dateien) beinhalten *(siehe Listing 7)*.

In *Abbildung 3* ist eine weitere Möglichkeit abgebildet, wie Kopien der Datenbank erstellt werden können.

Oracle bietet die Möglichkeit, sogenannte „Refreshable Clones" einer Pluggable Datenbank zu erstellen. Diese „Refreshable Clones" sind „Read Only"-Kopien der Quell-Pluggable-Datenbank, die entweder manuell oder nach einem definierten Intervall aktualisiert werden. Dieser „Refreshable Clone" kann verwendet werden, wenn die Kopie nicht die aktuellsten Daten benötigt und „Read Only" genügt. Dies kann zum Beispiel bei diversen Last-Tests oder bei analytischen Abfragen der Fall sein. Eine weitere Möglichkeit von „refreshable clones" ist der „PDB Switchover". Für diese Switchover gibt es zwei Möglichkeiten: Einerseits kann dieser geplant sein und führt zu keinem Datenverlust. Es wird vor dem Rollenwechsel ein Refresh der Kopie durchgeführt und anschließend werden die Rollen getauscht. Andererseits ist es möglich, im Fehlerfall der Quelle einen ungeplanten Switchover durchzuführen, in diesem Falle wird bewusst ein Datenverlust in Kauf genommen.

Die Multitenant-Option bietet den DBAs diverse Möglichkeiten, einen Klon einer Datenbank zu erstellen. Der große Vorteil dieser Möglichkeiten ist, dass die Container-Datenbank nicht neu erstellt werden muss und die entsprechenden Monitoring- und Backup-Scripts nicht angepasst werden müssen. Die meisten Möglichkeiten sind auch auf einer Standard-Edition-Datenbank oder auch ohne Multitenant-Option anwendbar. Dabei muss darauf geachtet werden, dass maximal eine Pluggable Datenbank in der CDB erstellt wird.

Über den Autor

Julian Frey arbeitet seit mehr als 10 Jahren als Oracle Consultant in verschiedenen Bereichen und bei diversen Firmen. Seit 5 Jahren arbeitet er als Oracle Consultant bei der Edorex AG. Nebenbei ist er auch Gründer des Oracle Beer Bern Meetup und hält diverse Vorträge.

Quellen

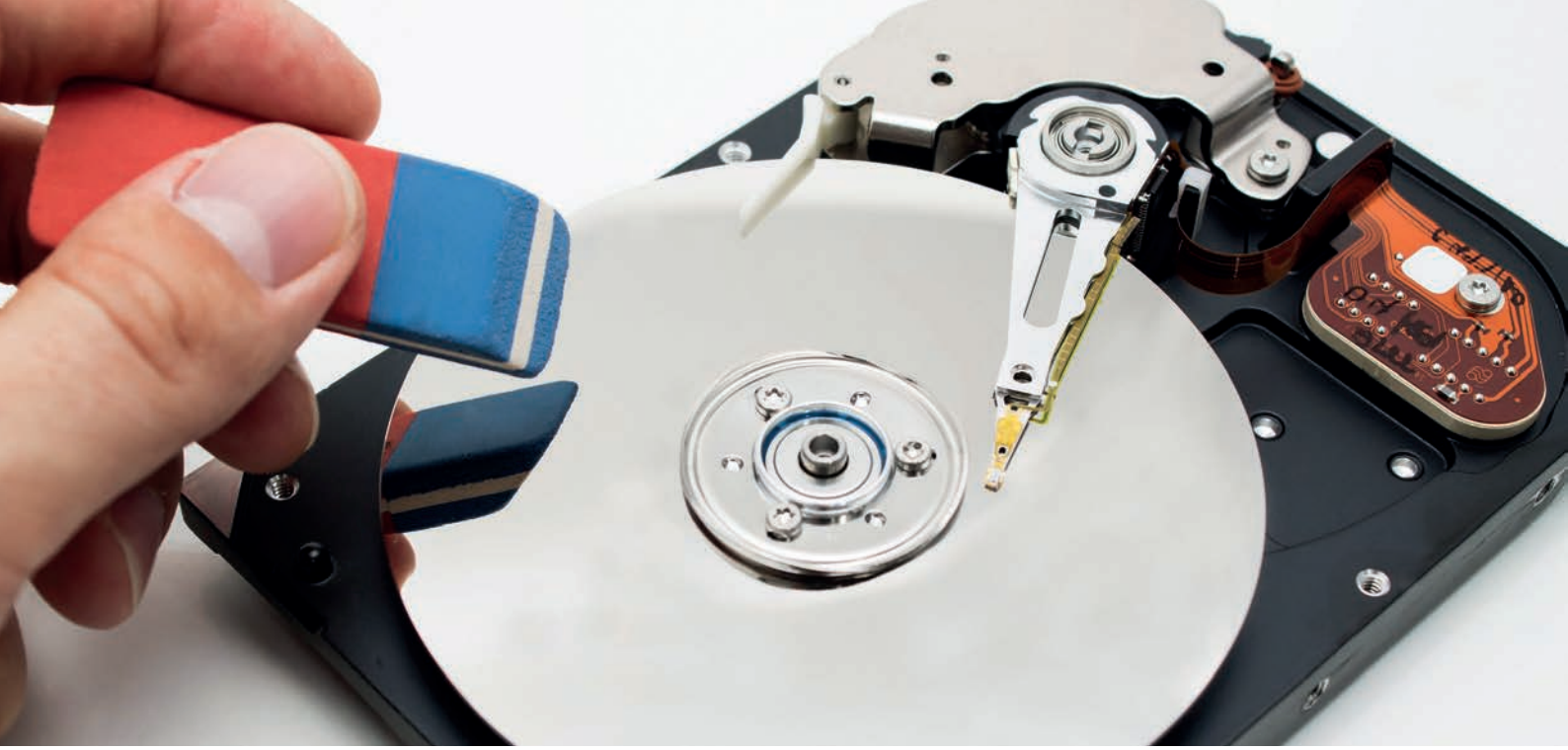
[1] *https://docs.oracle.com/en/database/oracle/oracle-database/19/bradv/rman-duplicating-databases.html#GUID-539E89F6-C0BC-49CB-8D8C-CD8FF-88BAF32*

[2] *https://docs.oracle.com/en/database/oracle/oracle-database/19/multi/img/admin089.png*

[3] *https://docs.oracle.com/en/database/oracle/oracle-database/19/multi/cloning-a-pdb.html#GUID-133B6D33-3B52-433B-9C85-C4BEE4F7284E*

[4] *https://docs.oracle.com/en/database/oracle/oracle-database/19/multi/administering-pdb-snapshots.html#GUID-72CA6DD5-4DE5-4375-BEA6-A01E5E-31A01B*

Julian Frey
julian.frey@edorex.ch



Sicheres Identifizieren von nicht relevanten Indizes

Peter Ramm, Otto Group Solution Provider (OSP)

Ein heikles Thema für DBAs und Entwickler ist vielfach das Erkennen und Entfernen von nicht beziehungsweise nicht mehr benötigten Indizes in Oracle-Datenbanken. Im Worst Case finden sich in der Praxis über Jahre gewachsene Systeme mit über 50% des Storage-Bedarfs nur durch Indizes ohne produktive Relevanz. Trotzdem wird dieses Potenzial zur einfachen Reduktion von Storage-Bedarf und Systemlast regelmäßig nicht genutzt, frei nach dem Motto „never touch a running system" beziehungsweise. „warum soll ich mir hier die Finger verbrennen und dann wird dieser Index doch noch irgendwo benötigt". Der Beitrag demonstriert die Vorgehensweise für die Ermittlung der zu überprüfenden Indizes und sicheren Bewertung ihrer Relevanz oder Irrelevanz unter Nutzung des frei verfügbaren Analyse-Tools „Panorama" *[1]*.

Abbildung 1 nennt die drei möglichen Motivationen für die Erstellung von Indizes auf Tabellen. Im Umkehrschluss gilt damit: Wenn ein existierender Index keine der drei genannten Rollen erfüllt, kann er eigentlich auch entfernt werden.

Was bringt die Entfernung unnötiger Indizes?

Reduktion von Storage-Bedarf der Datenbank

- Reduktion des Aufwands für Index-Maintenance bei DML-Operationen
- Effizientere Nutzung des DB-Cache
- Verbesserung des Laufzeit- und Antwortzeitverhaltens von Applikationen

Warum findet trotzdem oftmals keine aktive Bewertung und Entfernung unnötiger Indizes statt?

- Es gibt keine Rolle im Projekt/Produkt, die diese Aufgabe wahrnimmt
- Die DBAs haben nur begrenzten Einblick in die Fachlichkeit der Software
- Den Entwicklern fehlen die Skills und technischen Mittel zur sicheren Bewertung
- Never touch a running system: Das Entfernen von Indizes hat ein latentes Risiko, sich die Finger zu verbrennen und Betriebsstörungen zu verursachen

Im Folgenden wird gezeigt, wie für alle der in *Abbildung 2* genannten Kriterien eindeutig festgestellt werden kann, ob ein Index eine der genannten Rollen erfüllt oder nicht.

Rolle 1: Identifikation der Index-Nutzung durch User-SQL

Per ALTER INDEX <IndexName> MONITORING USAGE wird die Überwachung der

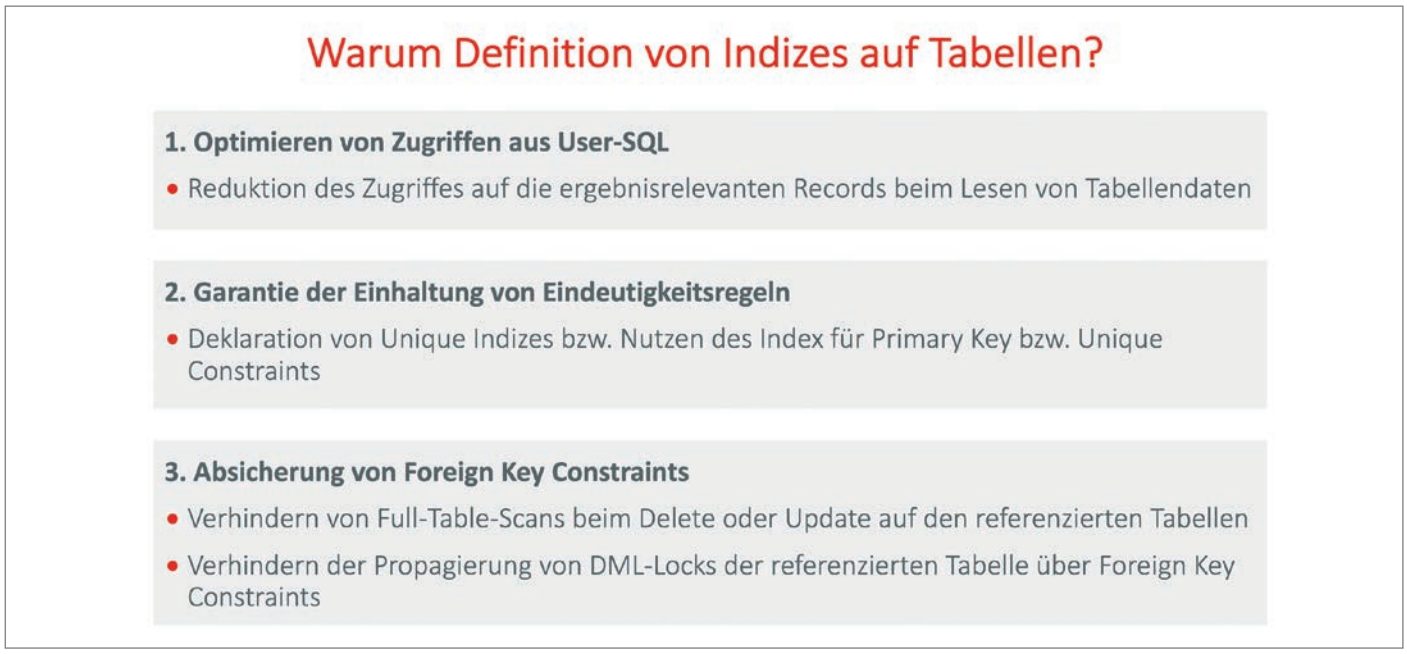


Abbildung 1: Gründe für die Existenz von Indizes (Quelle: Peter Ramm)

Nutzung eines Index in SQL-Statements aktiviert. Protokolliert werden dabei der Beginn der Überwachung sowie der Status der Nutzung (YES/NO), nicht jedoch der Zeitpunkt der Nutzung. Der Nutzungsstatus ist auswertbar über die Systemviews V$OBJECT_USAGE (nur für das aktuelle Schema) oder sys.OBJECT_USAGE (für die gesamte DB).

Per MONITORING USAGE wird dabei nur die Nutzung des Index in direkten SQLs protokolliert. Die Nutzung eines Index in rekursiven SQLs der DB selbst wird nicht protokolliert. Das heißt, die implizite Nutzung eines Index für den Check eines Foreign Key Constraint (zum Beispiel Lookup beim Delete auf einer referenzierten Tabelle) wird nicht erfasst.

Durch die Aktivierung von MONITORING USAGE auf einem Index ergibt sich kein messbarer Performance-Impact, jedoch: Die Ausführung von ALTER INDEX ... MONITORING USAGE als DDL führt zum erneuten Parsen aller diesen Index nutzenden SQL-Cursors bei der nächsten Ausführung.

Die erneute Ausführung von ALTER INDEX ... MONITORING USAGE setzt den Startzeitpunkt der Überwachung neu sowie den Nutzungsstatus wieder zurück auf NO.

Für eine permanente Überwachung eines Systems ist das regelmäßige Zurücksetzen des Nutzungsstatus von als genutzt markierten Indizes wesentlich, um so auch Indizes zu erkennen, die zwar in

Abbildung 2: Bewertung der Rollen von Indizes (Quelle: Peter Ramm)

der Vergangenheit einmal genutzt wurden, aber seit geraumer Zeit nicht mehr.

Skripte zum permanenten Tracking des Usage-Status

Listing 1 und *Listing 2* zeigen Skripte zum Zurücksetzen des Nutzungsstatus nur für Indizes, die:

- bereits als genutzt gekennzeichnet sind,
- seit mehr als x Tagen nicht zurückgesetzt wurden,
- nicht in aktuellen Execution-Plänen in der SGA vorkommen (außer bei der ersten Aktivierung von MONITORING USAGE). Damit wird das Risiko von Reparse-Bursts existierender SQL-Cursors durch Ausführung dieser Skripte eliminiert.

Listing 1 zeigt die Version zum Setzen beziehungsweise Rücksetzen des Nutzungsstatus für das aktuelle Schema des angemeldeten Users, *Listing 2* die Version für alle Schemata der DB mit Ausnahme der System-Schemata. Unter *[2]* stehen die in *Listing 1* und *Listing 2* gezeigten Skripte zum Kopieren zur Verfügung.

Auswertung des Nutzungsstatus von Indizes per SQL

Listing 3 zeigt das SQL für die Ermittlung seit definierter Zeit nicht genutzter Indizes des aktuellen Schemas auf Basis von V$OBJECT_USAGE, *Listing 4* zeigt dies für die gesamte DB auf Basis von sys.OBJECT_USAGE. Auch diese SQLs stehen unter *[2]* zum Kopieren zur Verfügung.

Auswertung des Nutzungsstatus von Indizes per Panorama

Das Analyse-Tool Panorama bietet auf Basis von sys.OBJECT_USAGE eine Auswertung über die Index-Nutzung in Rolle 1, verbunden mit Aussagen zu:

- Rolle 2: Nutzung der Indizes für Absicherung von Eindeutigkeiten
- Rolle 3: Nutzung der Indizes für Absicherung von Foreign Keys

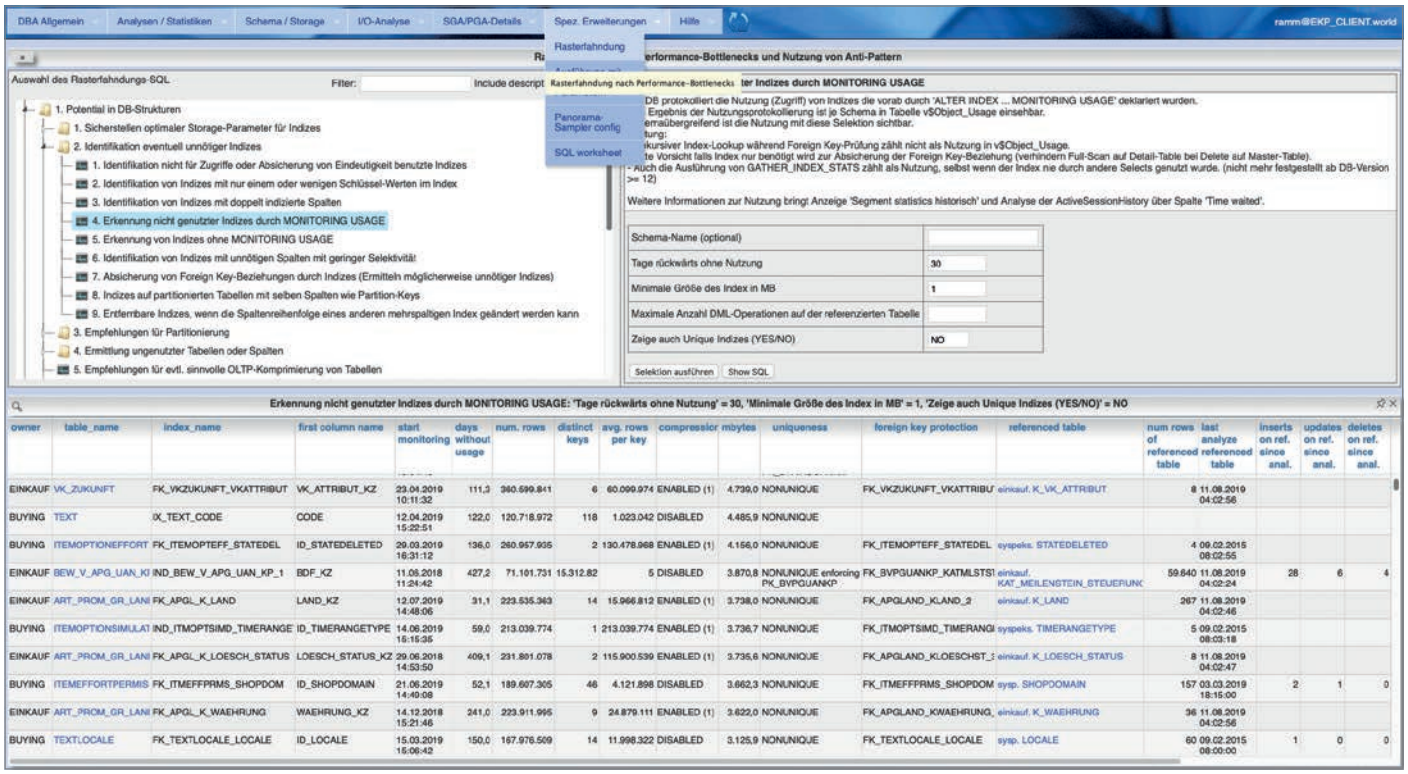


Abbildung 3: Auswertung des Nutzungsstatus per Panorama (Quelle: Peter Ramm)

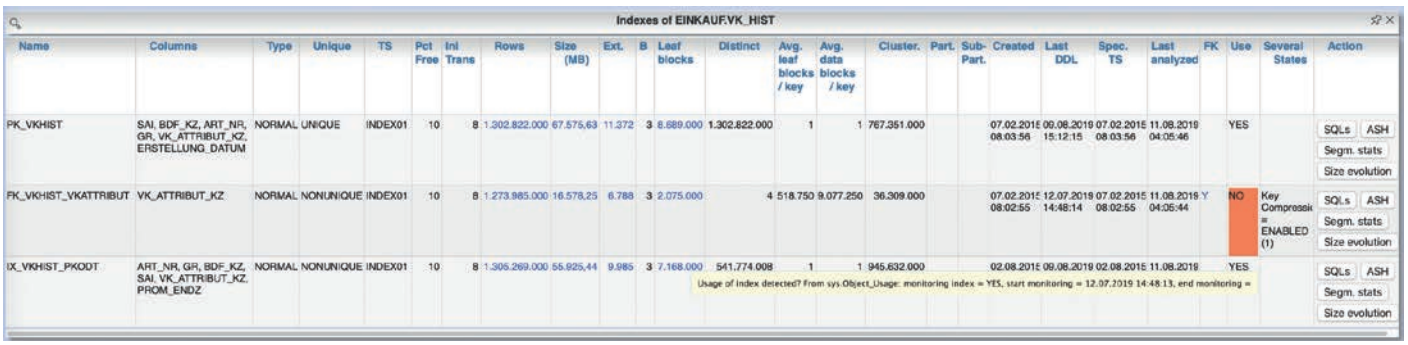


Abbildung 4: Auflistung der Indizes einer Tabelle (Quelle: Peter Ramm)

Abbildung 3 zeigt einen Beispiel-Screenshot mit detaillierten Aussagen unter anderem zu:

- der Anzahl der Tage ohne Nutzung des Index in User-SQL
- der eventuellen Nutzung des Index als Unique Index beziehungsweise für Enforcement von Unique oder Primary Constraints
- dem Vorhandensein von Foreign Key Constraints, die durch diesen Index abgesichert werden
- der Anzahl der Rows der über Foreign Key Constraint referenzierten Tabelle
- dem Zeitpunkt der letzten Analyse der über Foreign Key Constraint referenzierten Tabelle sowie der Anzahl ihrer DML-Operationen (Insert/Update/Delete) seit der letzten Analyse

Aus dieser Liste nicht genutzter Indizes kann mit einem Klick die Strukturdefinition der Tabelle des Index und auch der eventuell über Foreign Key Constraint referenzierten Tabelle aufgerufen werden.

Rolle 2: Relevanz eines Index für Absicherung von Eindeutigkeiten

In der Liste nicht genutzter Indizes laut *Abbildung 3* ist über die Spalte UNIQUENESS ersichtlich, ob ein Index Unique deklariert ist beziehungsweise als Basis von Unique oder Primary Key Constraints dient und somit eine Existenzberechtigung hat, auch wenn er nicht in SQLs genutzt wird.

Als Sonderfall kann allerdings auch in solch einem Szenario die problemlose Entfernung eines mehrspaltigen ungenutzten Unique Index möglich sein, wenn bereits die Eindeutigkeit einer einzelnen der Spalten dieses Index mit einem weiteren Unique Index/Constraint abgesichert ist.

Rolle 3: Relevanz eines Index für Absicherung von Foreign Key Constraints

Gründe für einen Index auf der referenzierenden Tabelle eines Foreign Key Constraint können sein:

- Verhindern von Full Table Scan auf der referenzierenden Tabelle bei Delete oder Update auf der Referenz
- Verhindern von Lock-Propagierung von DML-Locks bei DML auf der referenzierten Tabelle

Das bedeutet, dass auch bei Existenz eines Foreign Key Constraints dessen Absicherung mit einem Index nicht notwendig ist, wenn auf der referenzierten Tabelle kein DML stattfindet.

Dies ist regelmäßig der Fall beispielsweise bei Foreign Key Constraints gegen statische Stammdatentabellen.

Tolerierbarkeit von DML auf referenzierter Tabelle

Selbst wenn auf der referenzierten Tabelle in geringer Größenordnung DML stattfindet, kann oftmals auf eine Absicherung mit Index verzichtet werden:

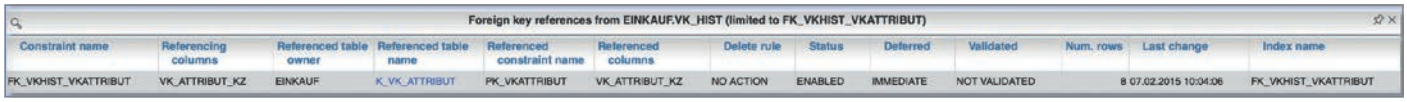
- Ein Full Table Scan auf der referenzierenden Tabelle beim Delete auf der Referenz kann eventuell billigend in Kauf genommen werden für sporadische DML-Aktivitäten. Zum Beispiel ist ein einmaliger Full Table Scan alle x Wochen oft weniger aufwendig als das permanente Vorhalten eines Index.
- Lock-Propagierung stellt nur in besonderen Konstellationen ein Problem dar. Insbesondere dann, wenn Updates auf Primärschlüsselspalten stattfinden, wofür insbesondere bei Verwendung von technischen Primärschlüsseln (IDs) keine Notwendigkeit besteht.

Im Blog-Post *[3]* wird die Relevanz der Lock-Propagierung über Foreign Key Constraints detaillierter untersucht:

- Eine Aufstellung verschiedener konkurrierender DML-Operationen auf zwei Tabellen
- Die jeweilige Relevanz eines Index auf dem Foreign Key Constraint zum Verhindern von Blocking Locks

Bewertung von DML-Operationen auf referenzierter Tabelle

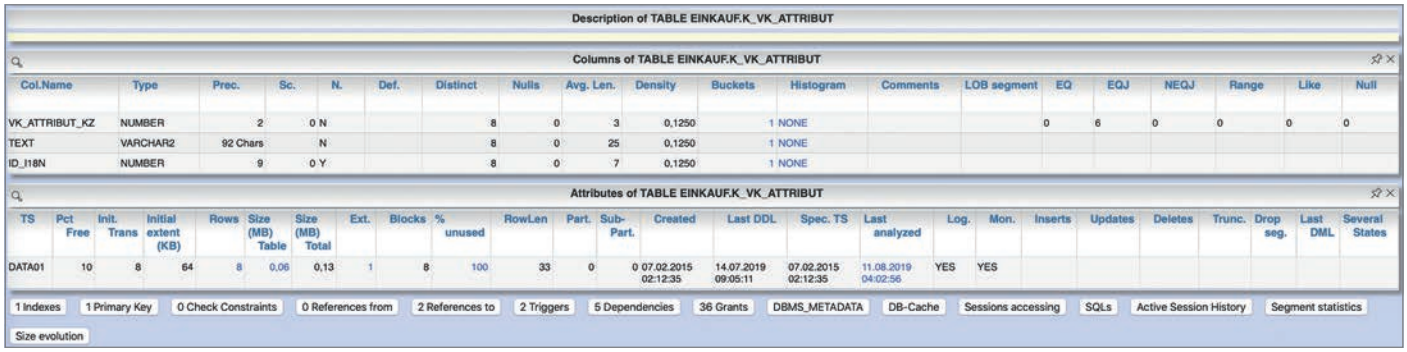
Das Vorhandensein und die Größenordnung von DML auf einer Tabelle kann über die View DBA_TAB_MODIFICATIONS nachvollzogen werden.



Foreign key references from EINKAUF.VK_HIST (limited to FK_VKHIST_VKATTRIBUT)

Constraint name	Referencing columns	Referenced table owner	Referenced table name	Referenced constraint name	Referenced columns	Delete rule	Status	Deferred	Validated	Num. rows	Last change	Index name
FK_VKHIST_VKATTRIBUT	VK_ATTRIBUT_KZ	EINKAUF	K_VK_ATTRIBUT	PK_VKATTRIBUT	VK_ATTRIBUT_KZ	NO ACTION	ENABLED	IMMEDIATE	NOT VALIDATED	8	07.02.2015 10:04:06	FK_VKHIST_VKATTRIBUT

Abbildung 5: Details eines Foreign Key Constraint (Quelle: Peter Ramm)



Description of TABLE EINKAUF.K_VK_ATTRIBUT

Columns of TABLE EINKAUF.K_VK_ATTRIBUT

Col.Name	Type	Prec.	Sc.	N.	Def.	Distinct	Nulls	Avg. Len.	Density	Buckets	Histogram	Comments	LOB segment	EQ	EQJ	NEQJ	Range	Like	Null
VK_ATTRIBUT_KZ	NUMBER	2	0	N		8	0	3	0,1250	1	NONE			0	6	0	0	0	0
TEXT	VARCHAR2	92 Chars		N		8	0	25	0,1250	1	NONE								
ID_I18N	NUMBER	9	0	Y		8	0	7	0,1250	1	NONE								

Attributes of TABLE EINKAUF.K_VK_ATTRIBUT

TS	Pct Free	Init. Trans	Initial extent (KB)	Rows	Size (MB) Table	Size (MB) Total	Ext.	Blocks	% unused	RowLen	Part.	Sub-Part.	Created	Last DDL	Spec. TS	Last analyzed	Log.	Mon.	Inserts	Updates	Deletes	Trunc.	Drop seg.	Last DML	Several States
DATA01	10	8	64	8	0,06	0,13	1	8	100	33	0	0	07.02.2015 02:12:35	14.07.2019 09:05:11	07.02.2015 02:12:35	11.08.2019 04:02:56	YES	YES							

1 Indexes | 1 Primary Key | 0 Check Constraints | 0 References from | 2 References to | 2 Triggers | 5 Dependencies | 36 Grants | DBMS_METADATA | DB-Cache | Sessions accessing | SQLs | Active Session History | Segment statistics | Size evolution

Abbildung 6: Struktur-Details einer Tabelle (Quelle: Peter Ramm)

In dieser View werden seit dem Zeitpunkt der letzten Analyse die Anzahl von Inserts, Updates und Deletes protokolliert. Voraussetzung für die Aufzeichnung in DBA_TAB_MODIFICATIONS ist laut Oracle-Dokumentation bis hin zu 19c der Status MONITORING=YES auf Tabellenebene. In der Realität werden aber ab Rel.11 DML-Operationen auch bei Status NOMONITORING aufgezeichnet.

Diese Anzahl von DML-Operationen seit der letzten Analyse erlaubt die Bewertung, ob die Häufigkeit und Frequenz der DML-Operationen auf der referenzierten Tabelle die Absicherung eines Foreign Key Constraint auf der referenzierenden Tabelle durch Index erfordern.

Schritte in Panorama für die Bewertung von DML auf referenzierter Tabelle

Ein Bestandteil der Anzeige der Tabellenstrukturen ist die Liste der Indizes einer Tabelle *(siehe Abbildung 4)*. Spalte „FK" zeigt die Absicherung eines Foreign Key Constraint durch diesen Index, Spalte „Use" den Nutzungsstatus des Index seit der letzten Aktivierung von MONITORING USAGE. Per MouseOver-Hint werden weitere Details eingeblendet, wie zum Beispiel der Zeitpunkt der letzten Aktivierung von MONITORING USAGE.

Ein Klick in die Spalte „FK" in *Abbildung 4* führt zu den Details des Foreign Key Constraint *(siehe Abbildung 5)*. Die Anzahl der Records (Spalte „Num. rows") der referenzierten Tabelle erlaubt hier eine erste Bewertung der möglichen Relevanz des Index.

Ein Klick in Spalte „Referenced table name" in *Abbildung 5* führt zur Strukturbeschreibung der referenzierten Tabelle *(siehe Abbildung 6)*. Hier erlaubt die Anzahl der DML-Operationen seit der letzten Analyse (Spalten „Inserts"/„Updates"/„Deletes") die genauere Bewertung, ob ein Index auf dem Foreign Key Constraint eine Relevanz zur Verhinderung der Lock-Propagierung ausgehend von der referenzierten Tabelle hat.

Entfernbarkeit auch aktuell wirklich genutzter Indizes

Trotz nachgewiesener Nutzung durch User-SQL kann es trotzdem sinnvoll sein, einen Index zu entfernen. Einige mögliche Konstellationen dafür sind:

- Spalten eines Index sind bereits anderweitig durch die führenden Spalten eines mehrspaltigen Index indiziert. Nach Entfernung eines Index übernimmt der weitere Index mit den führenden Spalten des ersten dessen Funktion.
- Die Partitionierung der Tabelle hat die gleiche Filterwirkung wie der Index. Eine Reduktion der Treffermenge durch Partitionierung ist effektiver als ein Index-Zugriff, wenn das Partitionierungskriterium eindeutig ist.
- Die Änderung der Reihenfolge der Spalten eines anderen mehrspaltigen Index macht einen Index mit den dann identischen führenden Spalten entbehrlich.
- Ein Zugriff per Index Fast Full Scan kann auch über einen anderen Index der Tabelle ausgeführt werden. Wenn die relevante(n) Spalte(n) auch anderweitig indiziert sind oder nur die Anzahl selbst relevant ist (etwa für COUNT(*)), dann kann alternativ auch ein anderer Index verwendet werden.

Zur Identifikation weiterer möglicherweise entbehrlicher Indizes bietet Panorama noch diverse weitere „Rasterfahndungs"-Abfragen *(siehe Abbildung 7)*.

Zum verwendeten Analyse-Werkzeug Panorama

Panorama *[1]* ist das Schweizer Taschenmesser des Autors, ein frei nutzbares Werkzeug für die Performance-Analyse von Oracle-DB. Es basiert auf rein lesend

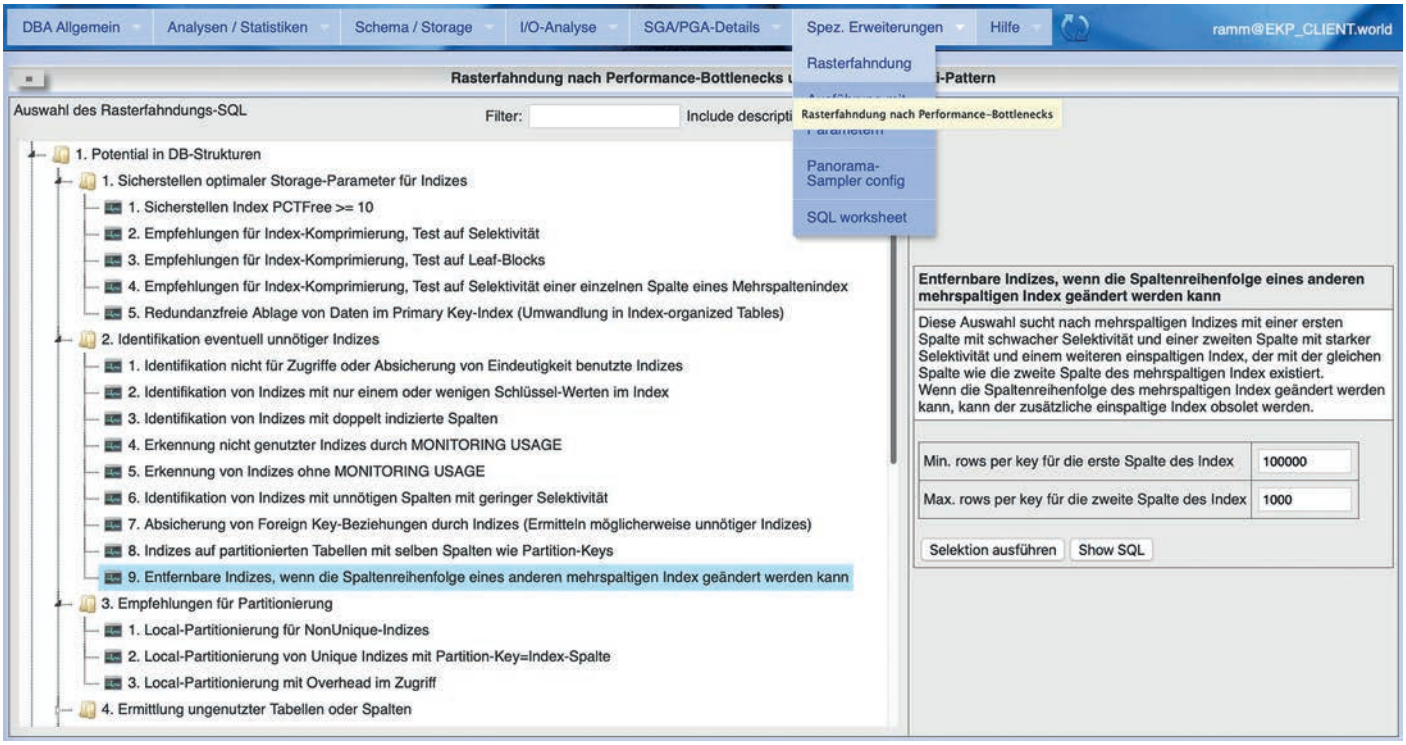


Abbildung 7: Weitere Aspekte zur Erkennung entfernbarer Indizes (Quelle: Peter Ramm)

über SQL zugreifbaren Informationen der DB, unter anderem auf Daten des AWR.

Für Nutzung mit Standard Edition beziehungsweise ohne Lizenzierung des Diagnostics Pack besitzt Panorama eine eigene Funktion zum Sampling von historischen Daten *[4]*.

Fazit

Das vorgestellte Verfahren zeigt, wie einfach und risikoarm bis risikofrei oftmals eine drastische Reduktion des Storage-Bedarfs und der Systemlast von Oracle-DB möglich ist.

Quellen

[1] *https://rammpeter.github.io/panorama.html*
[2] *https://rammpeter.blogspot.com/2017/10/oracle-db-identify-unused-indexes.html*
[3] *https://rammpeter.blogspot.com/2016/11/clarify-myths-of-indexing-foreign-key.html*
[4] *https://rammpeter.github.io/panorama_sampler.html*

Über den Autor

Der Autor arbeitet als Leiter eines Teams für strategisch-technische Beratung bei der Otto Group Solution Provider (OSP) GmbH in Dresden.

Peter Ramm
Peter.Ramm@ottogroup.com

Eintopf und Trennkost – Datenbanken im Ereignishorizont moderner Software-Entwicklung

Daniel Nelle, merlin.zwo InfoDesign & Gabriel Nelle, race result

Die moderne Softwareentwicklung bewegt sich gerade weg von monolithischen Architekturen zu Microservices. Für diese Entwicklung gibt es triftige Gründe. Das kann als Problem oder Chance gesehen werden. Fakt ist, dass dies erhebliche Auswirkungen auf die Datenbank-Welt hat, die zunehmend spürbar werden.

In der freien „Wildbahn" findet sich so manche relationale Datenbank, die einst unter dem Motto „gegründet" wurde: „Wir machen nur Oracle!" Warum auch nicht? Oracle bietet einen immensen Funktionsumfang. Es ist möglich, allerlei Technologien zu verwenden, ohne sich über die Grenzen der eigenen Datenbank hinaus begeben zu müssen. Ich (Daniel Nelle) war jahrelang ein eifriger Verfechter der sogenannten Thick-Database: eine Datenbank, die alles anbietet, was das Herz begehrt. Doch mehr und mehr sehe ich Herausforderungen mit dem Umgang solcher großen Datenbanken. Auch die Oracle-Datenbank selbst befindet sich stark im Wandel. Denken wir nur an Pluggable Database, Sharding und Autonomous Database, die meiner Meinung nach eine Reaktion auf die eingangs genannten Herausforderungen darstellen. In diesem Artikel wollen wir uns mit der modernen Software-Entwicklung auseinandersetzen, denn schließlich wird die Art und Weise, wie Datenbanken benutzt werden, dort maßgeblich beeinflusst.

Ich war sicherlich nicht der einzige Fan der Thick-Database, denn schließlich gibt es unzählige Datenbank-Installationen, die eine sehr lange Oracle-Vergangenheit haben. Damit sind wir auch schon mitten im Thema: Historisch gewachsene Systeme, aber auch junge, schnell wachsende Systeme, stellen eine beachtliche Herausforderung an Datenbank-Hersteller, -Entwickler und -Administrator. Immer häufiger treffe ich auf Datenbanken, bei denen mich das Gefühl beschleicht, dass hier etwas nicht rund läuft. Die Gründe dafür können sehr unterschiedlich sein. Hier ein paar Beispiele:

- Enorme Mengen an historischen Daten innerhalb der relationalen Oracle-Datenbank
- Verfügbarkeitskonzepte, die nicht zur Software-Architektur oder zur restlichen Infrastruktur passen
- Veraltete Datenbank-Versionen mit bekannten Sicherheitslücken

Performance

In letzter Zeit treffe ich vermehrt auf folgendes Phänomen: sehr große Datenbanken, die zu über 90% aus historischen Daten bestehen. Um diese Daten bereitzustellen (Antwortzeit und Verfügbarkeit), wird ein immenser Aufwand betrieben. Um eine gute Performance zu erzielen, sind meist Lizenzen für Enterprise und zusätzliche Optionen erforderlich. Das Problem wird, wie man so schön sagt, mit Geld erschlagen.

Als Beispiel möchte ich eine Datenbank im Finanzsektor anführen, die enorme Mengen an Daten speichert. Diese werden in einigen partitionierten Tabellen abgelegt. Jeden Tag wird eine Sub-Partition erstellt, in die dann mehrere Milliarden Datensätze geschrieben werden. Um eine ordentliche Performance zu erzielen, ist fast jede Minute der Nacht ausgebucht, da die Daten für die Business-Anwendungen aufbereitet werden müssen. Ohne diese Jobs wäre das Tagesgeschäft erheblich eingeschränkt.

Verfügbarkeit

Viele Administratoren lieben ihre Verfügbarkeitslösungen, die aus ihrer Sicht bis ins letzte Detail optimiert wurden. In vielen Beratungsgesprächen muss ich diese Euphorie leider zerstören. Oft werden ganz entscheidende Details nicht berücksichtigt oder funktionieren nicht so wie angenommen. Zum Beispiel habe ich in 15 Jahren als Datenbankadministrator nur wenige Anwendungen angetroffen, die tatsächlich Transparent Application Failover unterstützen.

Das Kernproblem ist meiner Meinung nach die Komplexität der Hochverfügbarkeitslösungen. Sobald eine Oracle-Datenbank hochverfügbar sein soll, steigert sich die Komplexität des Setups enorm. Ein ganz prominentes Beispiel ist ein Real Application Cluster (RAC). Als Oracle RAC eingeführt wurde, gab es Single-Core-Prozessoren. Außerdem waren große Arbeitsspeicher wegen der 32-Bit-Grenze nicht nutzbar oder sehr teuer. Hinzu kam, dass damals die Hardware deutlich unzuverlässiger war als heute. Das RAC löste damals Skalierbarkeits- und Verfügbarkeitsprobleme. Heute sieht die Sache jedoch ganz anders aus: Die Skalierbarkeit eines RAC wird nur für sehr wenige Datenbanken tatsächlich benötigt. Meistens wird RAC zur Ausfallsicherung eingesetzt. Gerade das weitverbreitete 2-Knoten-RAC ist jedoch problematisch: Beim Ausfall eines Knotens fällt der zweite Knoten oft ebenfalls aus, weil die Schuldfrage nicht zweifelsfrei geklärt werden kann.

Verfügbarkeit bei Oracle geht immer mit einer enormen Steigerung der Komplexität einher, die oft von Software-Entwicklern, Administratoren und IT-Verantwortlichen nicht richtig verstanden wird. Das führt zu Systemen, die nicht jedes Ausfallszenario abdecken und im Worst Case Datenverlust zur Folge haben können, während sich der Kunde in einer falschen Sicherheit wiegt. Dabei gilt: Komplexität ist der größte Feind der Verfügbarkeit.

Releasewechsel

Eine andere Anwendung speichert Messdaten aus der Infrastruktur (also Wasser, Gas, Strom etc.) in ihrer Oracle-Datenbank. Hier sendet jeder Sensor einen Datensatz pro Sekunde. Neben den bereits beschriebenen Performance-Problemen durch große Bestände an historischen Daten kommt es hier auch zu Schwierigkeiten bei Releasewechseln.

Die erforderliche Downtime läge bei ca. einer Stunde für Grid Infrastructure und zusätzlich ca. einer Stunde für die Oracle-Datenbank. Das lassen die Service Level Agreements nicht zu. Andererseits unterliegen Infrastruktur-Provider immer strengeren Gesetzen in Bezug auf die IT-Sicherheit. Daher ist der Kunde verpflichtet, eine Version zu nutzen (und gegebenenfalls zu migrieren), die von Oracle supportet wird. Die Kosten für eine Golden-Gate-Lizenz empfindet der Kunde als „üblen Scherz". Selbst wenn diese vorhanden wäre, würde sich ein sauber geplanter und vollzogener Releasewechsel über mehrere Wochen oder gar Monate hinziehen, um am Ende eine Downtime von wenigen Sekunden zu erreichen.

Gerade bei Datenbanken, die einen sehr hohen Verfügbarkeitsanspruch haben, wird es sehr kompliziert, wenn es an einen Releasewechsel geht. Viele Datenbank-Spezialisten finden zwar genau diese Themen sehr interessant, aber Business-freundlich ist das nicht. Tatsächliches Zero-Downtime-Patching gibt es immer noch nicht und nahe dran kommt man nur mit Golden Gate. Das Ergebnis ist, dass Datenbanken oft alte Softwarestände haben.

Problemanalyse

Wann immer versucht wird, mit aller Macht alle Daten in einen einzigen Datenbank-Typ

zu pressen, entstehen Lösungen, die ich eingangs mit „nicht rund" beschrieben habe. Es stellt sich also die Frage, warum dies nicht anders gelöst wird. Jeder Datenbank-Typ (RDBMS, NoSQL etc.) hat seine Stärken, die andere Datenbank-Typen nicht adäquat imitieren können. Warum werden die Daten nicht nach Anforderung getrennt und genau in den Datenbanken gespeichert, die für ihre Bedürfnisse am besten geeignet sind?

Beispielsweise könnte die erwähnte Datenbank aus dem Finanzsektor ihre Stammdaten in der Oracle-Datenbank und ihre historischen Daten in einer Big-Data-Datenbank (z.B. Oracle NoSQL) speichern. Dieser Mix aus RDBMS und NoSQL-Datenbank würde genau den Anforderungen der Anwendung entsprechen. Auch für den Infrastruktur-Provider würden durch einen Mix aus RDBMS und NoSQL oder Time-Series-DB einige Probleme beim Releasewechsel entfallen. Zum Beispiel gehören Konzepte für Zero-Downtime-Patching bei NoSQL-Datenbanken meist zur Basisausstattung. Die verbleibende relationale Datenbank, welche die Stammdaten speichert, wird damit deutlich kleiner und ist beim Releasewechsel einfacher und kostengünstiger zu handhaben.

Moderne Software-Entwicklung geht ebenfalls den Weg der Trennung von Daten in verschiedene Datenbanken und Datenbank-Typen. Das bringt neue Herausforderungen mit sich: sowohl für die Software-Architekten wie auch für die Datenbank-Spezialisten. Im Folgenden übergebe ich an Gabriel Nelle, einen herausragenden Software-Entwickler mit über 20 Jahren Erfahrung, um einen Perspektivwechsel aus Sicht der Software vorzunehmen. Er wird ein paar zentrale Elemente der modernen Software-Entwicklung herausgreifen und aufzeigen, wie das die Datenbank-Welt mehr und mehr beeinflussen wird.

Datenbanken in der modernen Software-Entwicklung

Wir alle haben schon von der „Cloud" gehört und dass jetzt „alles in die Cloud muss" und die „Cloud ist die Zukunft". Vielleicht geht es Ihnen auch so, dass Sie das langsam nicht mehr hören können. Deshalb möchte ich zunächst einmal festhalten, dass es für mich als Softwareentwickler nicht um Amazon vs. Oracle vs. On Premises geht, sondern darum, die Grundideen der Cloud zu verstehen und in meiner Software umzusetzen. So kann ich die Vorteile einer Cloud-fähigen Applikation genießen, die ich vor allem in Skalierbarkeit, Sicherheit, Performance und Verfügbarkeit sehe.

Software wird in modernen Cloud-Architekturen meist in kleine bis kleinste Services heruntergebrochen, die alle für sich einzeln skalierbar sind, indem sie mehrfach gestartet werden. In Kubernetes-Umgebungen wird einmal definiert, welche Services es gibt, wie viele Instanzen von jedem Typ laufen sollen, nach welchen Regeln hoch- oder runterskaliert werden soll, wie die Teile miteinander kommunizieren dürfen, welche von außen erreichbar sind und so weiter. Dann kümmert sich das System komplett selbstständig um das Starten, die Überwachung, das Herunterfahren und Neustarten hängender Instanzen oder das Verschieben auf andere Nodes, wenn ein Rechner ausfällt.

Um auf einem solchen System sinnvoll zum Einsatz kommen zu können, muss ein Service unter anderem stateless geschrieben worden sein.

Stateless

„Twelve-factor processes are stateless and share-nothing. Any data that needs to persist must be stored in a stateful backing service, typically a database." *[1]* Wenn ein Service stateless programmiert ist, kann er jederzeit heruntergefahren und neu gestartet werden, ohne dass wichtige Informationen (State) verloren gehen. Es macht auch keinen Unterschied, welche der gerade laufenden Instanzen eine Anfrage erhält. Der Output ist nicht von Informationen abhängig, die im Memory gehalten werden.

Ein typisches Beispiel für State wären Sessions in einem Webserver. Um State in einer solchen Applikation halten zu können, wird dieser in schnelle Datenbanken ausgelagert.

Das ist für Webserver kein neues Konzept, aber durchaus für viele andere Applikationen. Nehmen wir zum Beispiel einen IOT-Server, der stehende Verbindungen zu aktiver Hardware unterhält. Solange nur einer dieser Server läuft, ist es einfach, eine Liste aller verbundenen Geräte aus dem Memory zu erzeugen, denn diese eine Instanz hält alle Verbindungen zu allen verbundenen Geräten. Wenn jedoch mehrere Instanzen laufen sollen und jeder dieser Server Verbindungen halten kann, muss die Liste der verbundenen Geräte ausgelagert werden. Nur so kann jede Instanz des Service die gleiche, komplette Liste ausliefern.

Für diesen Use-Case benötigen wir eine Datenbank, die schnelle Antwortzeiten auf viele kleine Anfragen liefern kann, zum Beispiel einen In-Memory Key-Value-Store. Optimal wäre auch, wenn dieser auf der gleichen physikalischen Hardware läuft. Die Datenbank muss zudem auf jeden Fall aus mehreren Instanzen bestehen, um die Ausfallsicherheit, die wir durch die Multi-Instanz-Fähigkeit des Service gewonnen haben, ebenfalls gewährleisten zu können. Auch muss sie allen Instanzen die gleichen Daten liefern, da es sich hierbei um shared State handelt.

In unserem Beispiel muss darüber hinaus mit jedem der Geräte von jeder Instanz aus kommuniziert werden können. Wenn etwa über eine Admin-Oberfläche ein Befehl für eines der Geräte auf einer beliebigen Instanz eingeht, muss dieser von der verbundenen Instanz an das Gerät weitergegeben werden. Die dazu notwendige Kommunikation zwischen den Instanzen sollte nicht voraussetzen, dass die Instanzen voneinander wissen. Das wiederum würde sich nicht mit dem einfachen Dazu- und Abschalten von Instanzen vertragen. Diese Herausforderung kann zum Beispiel mit einer Message-Queue gelöst werden, an die sich jede Instanz anhängt; nur die verbundene Instanz bearbeitet den Befehl.

Das Prinzip der Statelessness deutet schon an, dass in der Entwicklung moderner, Cloud-fähiger Applikationen viel auf mehrere kleine, hoch-verfügbare, verteilte und hoch-spezialisierte Datenbanken gesetzt wird. Als Nächstes möchte ich das Prinzip der Single Responsibility erläutern, das noch deutlich weiter geht, als nur den Einsatz verschiedener, spezialisierter Datenbank-Typen zu fordern.

Single Responsibility

Der Grundgedanke moderner Software-Architektur ist das Aufsplitten von Applikationen in viele kleine Services, die miteinander kommunizieren, statt einer großen monolithischen Applikation. Dabei hat jeder Service eine Aufgabe und kein anderer Service ist für die gleiche Aufgabe oder Teile davon zuständig. Übertragen wir den Grundgedanken auf die Datenbank-Welt,

darf eine Datenbank nur Daten für eine Aufgabe enthalten. Zum Beispiel nur die Log-Daten. Gleichzeitig darf es keine andere Datenbank geben, die Log-Daten der gleichen Kategorie enthält.

Folgt man den Regeln der Microservice-Architektur, so kann ein Microservice auf mehrere Datenbanken zugreifen, aber auf eine Datenbank darf nur von einem Service aus zugegriffen werden. Selbst Datenbanken vom gleichen Typ (z.B. Redis als schneller Cache) würden pro Service aufgesetzt und können damit auch pro Service unabhängig verwaltet werden. Brauchen mehrere Services Zugriff auf die gleichen Daten, bekommen sie die Daten durch einen dedizierten Service, der als einziger die Datenbank direkt ansprechen darf. Damit wird klar, dass die Aufteilung der Daten sehr eng mit der Aufteilung der Services verknüpft ist und andersherum. Die Größe eines Service ist dabei nicht entscheidend. Wichtig ist, dass sowohl Service als auch Daten sinnvoll und an den gleichen Stellen abgegrenzt werden. Ein gut durchdachtes Gesamtkonzept ist essenziell. Das Prinzip der Single Responsibility hilft dabei als roter Faden, verbessert im entstehenden System den Überblick und erleichtert später besonders in großen Applikationen die Fehlersuche enorm.

Aus Sicht eines Datenbankadministrators hat dies auch einige Vorteile. Mehrere Datenbanken nach dem Prinzip der Single Responsibility aufgeteilt, werden leichter verwaltbar, ersetzbar und migrierbar. Selbst in einer riesigen Server-Landschaft muss sich der Administrator nur mit einem Ansprechpartner oder einem Team absprechen. Änderungen der Datenstruktur, Optimierung der Indizes und dergleichen können immer in Bezug auf einen einzigen, überschaubaren Service und dessen Nutzungsprofil der Datenbank vorgenommen werden. Migrationen werden überschaubar. Das Volumen der zu migrierenden Daten beschränkt sich auf den Teil, den ein Service benutzt. Bei den meisten Services ist das ein geringer Bruchteil der Gesamtdaten, wenn es sich nicht gerade um den Big-Data-Service handelt. Vor allen Dingen aber sind die Abhängigkeiten überschaubar, die zu bedenken und anzupassen sind. Dadurch werden die Wartungsfenster kleiner oder entfallen je nach Datenbank komplett. Da an jeder Datenbank nur maximal ein Service hängt, sind eventuelle Anpassungen an eine neue Version der Datenbank überschaubar. Das minimiert das Risiko für Fehler und Ausfälle.

Das Trennen der Datenverwaltung nach dem Single-Responsibility-Prinzip liefert nebenbei auch noch den Bonus, für jeden Use Case eine optimale Wahl treffen zu können. Wo vorher noch historische Messdaten in einer relationalen Datenbank gelandet sind – weil dort alle Daten sind –, gibt es jetzt keinen Grund mehr, diese dort abzulegen. Time-Series-Datenbanken mit einem starken Fokus auf die Auswertbarkeit der Daten sowie leichtem Auslagern und Nachladen von alten Daten sind relationalen Datenbanken in diesem Gebiet weit überlegen und können jetzt zum Einsatz kommen. Weniger strukturierte Daten müssen nicht mehr mit Tricks in eine relationale Datenbank oder einen speziellen Datentyp „gepresst" werden, sondern können in einer auf solche Daten optimierte Dokument- oder Objekt-Datenbank gespeichert werden. Hochsensible Nutzerdaten können in extra dafür konzipierten, verschlüsselten Vaults abgelegt werden und sind im Falle eines Einbruchs nicht mit den anderen Daten zusammen kompromittiert. Über Sicherheitslücken in weniger sensiblen Endpunkten wird damit kein Zugriff auf sensible Inhalte erreicht, was besonders in Systemen mit vielen verschiedenen Teams und Zuständigkeitsbereichen von großem Vorteil ist.

Durch die Nutzung der passenden Datenbank, die Optimierung der Daten und Indizes sowie die Verkleinerung der Datenmenge, die eine Datenbank verwaltet, kann die Performance eines Service verbessert werden. Performance kann genau dort optimiert werden, wo sie benötigt wird, ohne dabei die Performance anderer Teile der Applikation negativ zu beeinflussen. Dazu kommt, dass jeder einzelne Service inklusive seiner Datenbank unabhängig skalierbar ist. Entwickelt sich ein Teil der Applikation durch vermehrte Nutzung zu einem Bottleneck, kann sich das Team aus Datenbankspezialisten und Software-Entwicklern auf die Skalierung genau dieses einen Teilbereichs konzentrieren.

Fazit

In diesem Artikel haben wir uns fast ausschließlich auf die Vorteile der aktuellen Entwicklung beschränkt. Wir wollten aufzeigen, welche Chancen darin liegen und wie die modernen Architekturen den IT-Betrieb verbessern und erleichtern können.

Uns ist allerdings auch bewusst, dass dies zunächst einmal Umdenken von IT-Spezialisten wie auch Software-Entwicklern erfordert. Diese Entwicklung bringt auch neue Herausforderungen mit sich: im Dschungel der Services und Datenbanken den Überblick zu behalten, im Fehlerfall brauchbare Tracing-Informationen bereitzustellen sowie Metriken und Log-Informationen zu sammeln, Auswertungsmöglichkeiten bereitzustellen, die durch die Aufteilung der Daten erschwert wurden, sinnvolle Duplizierung von Daten zu erkennen und synchron zu halten und vieles mehr.

Die aktuelle Entwicklung wird vor den Datenbanken nicht halt machen, wie wir insbesondere im Bereich der Single Responsibility herausgearbeitet haben. Auch wenn die Datenbank von Key-Value-Store über relationale, dokumentbasierte und objektbasierte Strukturen bis hin zur Time-Series- und Graphdatenbank alles unterstützt, was unser Herz begehrt: Sobald mehrere Services auf die gleiche Datenbank beziehungsweise die gleichen Daten zugreifen, sind diese nicht mehr sauber getrennt. Was auf Software-Seite mühsam nach dem Prinzip der Single Responsibility erarbeitet wurde, ginge auf Daten-Ebene wieder verloren.

Quellen

[1] The Twelve-Faktor App: *https://12factor.net/processes*

Daniel Nelle
daniel.nelle@merlin-zwo.de

Gabriel Nelle
nelle@raceresult.com

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

- Gerrit Griebel
- Prakash Pandey
- Carsten Wegscheider
- Christina Kraus
- Dieter Henig
- Matthias Becker
- Markéta Starobová Starobová
- Markus Sparrer
- Goran Dokmanovic

Firmenmitglieder DOAG

- MHP Management- und IT-Beratung GmbH, Stefan Hess
- CPA SoftwareConsult GmbH, Jörg Spiegelhoff
- Siltronic AG, Günter Riedhofer

Termine

25.10.2019
Regionaltreffen Hannover
Andreas Ellerhoff

05.11.2019
Regionaltreffen Bremen
Ralf Lölling

08.11.2019
DOAG Datenbank Webinar: Patching und Upgrade aktueller Datenbankversionen
Referent: Thilo Künkel
online

19. - 22.11.2019
DOAG 2019 Konferenz + Ausstellung
Nürnberg

28.11.2019
Regionaltreffen Trier/Saar/Luxemburg
Bernd Tuba, Holger Fuchs
Eppelborn

02.12.2019
Regionaltreffen München/Südbayern
Andreas Ströbel
München

04.12.2019
Regionaltreffen Berlin/Brandenburg
Michel Keemers
Berlin

06.12.2019
Vorstandssitzung 4/19
Berlin

10.12.2019
Regionaltreffen Hannover
Andreas Ellerhoff
Hannover

12.12.2019
Regionaltreffen Dresden/Sachsen
Andreas Ellerhoff

13.12.2019
DOAG Datenbank Webinar: Transportmöglichkeiten mit Cascading/Cascaded Standby DBn, Active Data Guard Far Sync und Redoroutes
Referent: Marc Wagner
online

19.12.2019
Regionaltreffen Nürnberg
Martin Klier, Thomas Köppel
Nürnberg 04.12.2019

15.01.2020
Regionaltreffen NRW
Martin Schmitter, Torsten Rosenwald
Köln

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.)
ViSdP: Mylène Diacquenod
Redaktionsleitung: Martin Meyer
Kontakt: redaktion@doag.org
Weitere Redakteure (in alphabetischer Reihenfolge): Lisa Damerow, Mylène Diacquenod, Marina Fischer, Sanela Lukavica, Robert Marz, Yann Neuhaus, Fried Saacke

Fotonachweis:
Titel: © kchung | *https://de.123rf.com*
S. 12: © Tithi Luadthong | *https://de.123rf.com*
S. 20: © Li Jiuming | *htt ps://de.123rf.com*
S. 24: © Athapet Piruksa | *https://de.123rf.com*
S. 30: © Mathias Rosenthal | *https://de.123rf.com*
S. 36: © Minerva Studio | *https://de.fotolia.com*
S. 43: © bagotaj | *https://de.123rf.com*
S. 47: © macrovector | *https://de.123rf.com*
S. 51: © WICHAI WONGJONGJAIHAN | *https://de.freepik.com*
S. 58: © andriano | *https://de.123rf.com*
S. 64: © baloon111 | *https://de.fotolia.com*
S. 69: © rawpixel | *https://de.123rf.com*
S. 73: Hintergrund: © Dzianis Kuryanovich | *https://de.123rf.com*

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

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

Inserentenverzeichnis

B4BMEDIA.NET AG *www.e3zine.com* **S. 29**
dbi services sa *www.dbi-services.com* **S. 15**
DOAG e.V. *www.doag.org* **U 2**
Logicalis GmbH *www.de.logicalis.com* **U 3**
MuniQsoft Consulting GmbH *www.muniqsoft-consulting.de* **S. 39**
MuniQsoft Training GmbH *www.muniqsoft-training.de* **S. 3**
Robotron Datenbank-Software GmbH *www.muniqsoft-training.de* **S. 35**
Trivadis AG *www.trivadis.com* **U 4**

WIR LEBEN DIGITALISIERUNG.

Die Digitale Transformation macht vor keiner Branche halt und verändert Wertschöpfungsketten und Strukturen auch Ihres Unternehmens. Sie müssen sich neuen Herausforderungen stellen. In andere Richtungen denken. Ihr Geschäftsmodell anpassen und weiterentwickeln, damit Sie auch in Zukunft wettbewerbsfähig bleiben. Wir sind mittendrin im Geschehen und gestalten partnerschaftlich Ihren Weg ins Zeitalter der Digitalisierung. Sprechen Sie mit uns.

m.trivadis.com/digitalisierung | info@trivadis.com

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

trivadis