Nr. 3 | April 2023 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP

inklusive BUSINESS NEWS

MICROSERVICES & DOMAIN-DRIVEN-DESIGN

Aus der Praxis

Oracle-Lizenzierung in der OCI-, AWS-, Azure- und Google-Cloud

Im Interview

Christoph Lüttig, Andreas Badelt und Kilian Müller, Compleo Charging Software

Business News

Postpandemische Arbeitswelten

CloudLand 2023

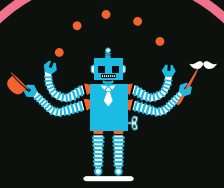
DAS EVENT DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY

Weitere Informationen

20. - 23. JUNI

im Phantasialand in Brühl

www.cloudland.org



Tag 1 / 20. Juni
CloudCamp

Tag 2 / 21. Juni
Summer Night

Tag 3 / 22. Juni
Gaming Night

Tag 4 / 23. Juni
Customer Stories

#CloudLand2023

Eventpartner: Heise Medien

Sven Bernhardt
DOAG Themenverantwortlicher für Methodik und User Experience

Ausgabe Nr. 3/2023
auf Abruf!

Liebe Mitglieder, liebe Leserinnen und Leser,

die vergangenen Jahre haben gezeigt, dass die digitale Transformation nicht nur eine Notwendigkeit, sondern auch eine Chance für Unternehmen ist. Eine wichtige Rolle spielen dabei Microservices und Domain-driven Design, die es ermöglichen, komplexe Anwendungen in kleine, unabhängige und gut skalierbare Teile zu zerlegen. Gerade in Zeiten der digitalen Transformation und des verstärkten Wettbewerbs ist es entscheidend, dass Unternehmen schnell und effektiv auf Veränderungen reagieren können.

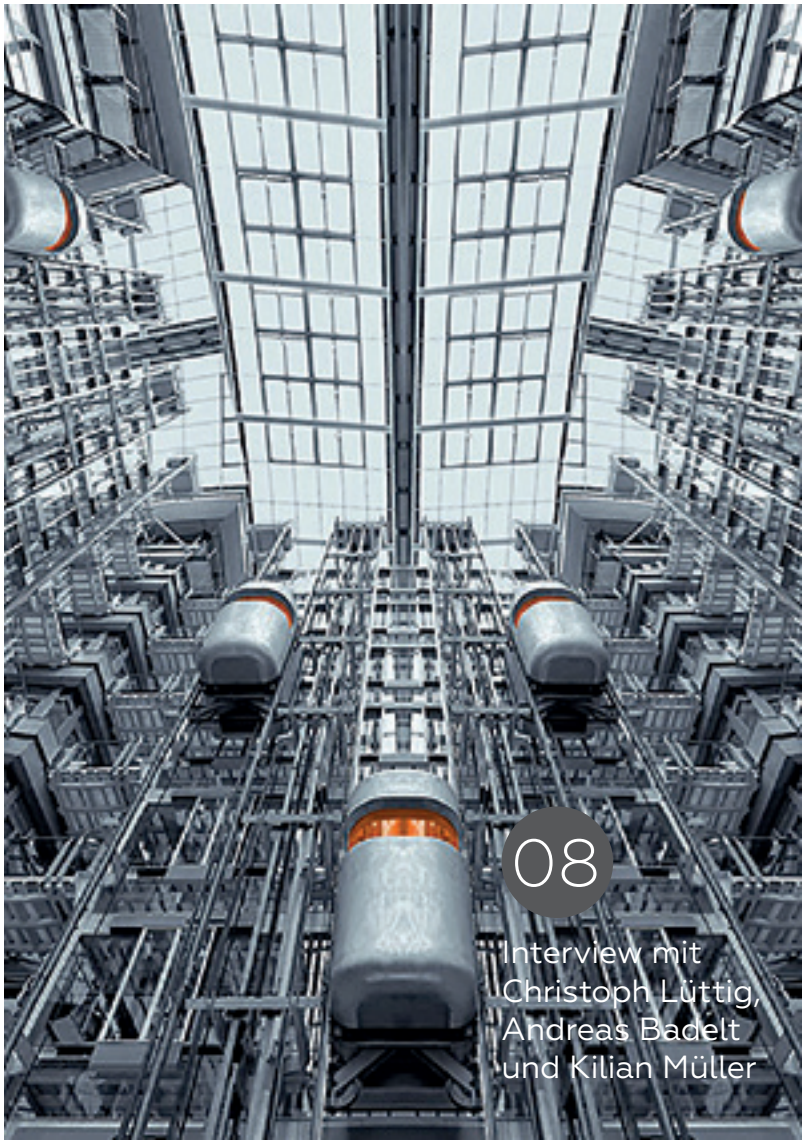
In dieser Ausgabe des Red Stack Magazin beschäftigen wir uns daher damit, wie man Microservices effizient einsetzen kann und welche Bedeutung in diesem Zusammenhang dem Domänendesign zukommt. Da wir aber heute nicht in einer Microservices-only-Welt leben, sondern darüber hinaus stetig daran arbeiten müssen, Bestandssysteme zu warten, zu optimieren und weiterzuentwickeln, dürfen Themen aus den Bereichen Datenbankbetrieb und -optimierung auch nicht fehlen. Als Notiz am Rande sei hier erwähnt, dass man sich auch im Microservices-Umfeld um die nachhaltige Speicherung von Daten kümmern muss. In der Business News geht es zudem diesmal darum, wie sich die Arbeitswelt durch die Pandemie verändert hat. Die Vorteile von Homeoffice und dezentralem Arbeiten wurden aufgezeigt, die Bedürfnisse und Erwartungen der Arbeitnehmerinnen und Arbeitnehmer haben sich verändert. Flexibilität, Work-Life-Balance und ein gutes Arbeitsklima sind für viele Arbeitnehmerinnen und Arbeitnehmer mittlerweile wichtiger als das Gehalt.

Unternehmen stehen in den kommenden Jahren vor vielen Herausforderungen. Die Kombination aus technologischen Fortschritten und gesellschaftlichen Veränderungen erfordert ein Umdenken und eine Anpassung an die neuen Anforderungen. Microservices und Domain-driven Design sind hierbei wichtige Werkzeuge, um den Anforderungen einer sich ständig wandelnden Geschäftswelt gerecht zu werden. Unternehmen müssen jedoch auch ihre Arbeitsstrukturen und -bedingungen überdenken, um für Fachkräfte attraktiv zu bleiben und flexibel auf Veränderungen reagieren zu können.

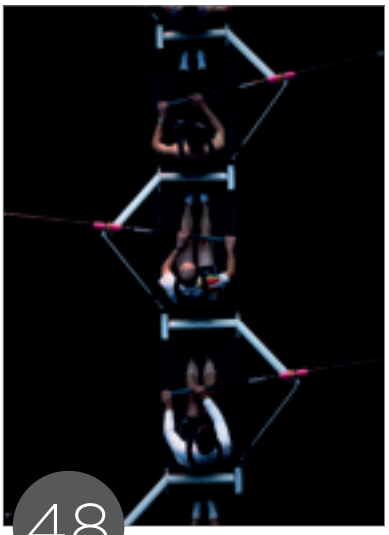
Insgesamt ist es wichtig, dass Unternehmen diese Veränderungen als Chance begreifen und sich agil und flexibel auf die neuen Anforderungen einstellen. Wir hoffen, dass die Artikel in den Ausgaben des Red Stack Magazin und der Business News dazu beitragen werden, für diese Themen zu sensibilisieren und Euch wertvolle Impulse für Eure tägliche Arbeit zu geben.

Mit besten Grüßen

Sven Bernhardt



Quadratur des Kreises – Attribute Clustering, ein Feature der Oracle-Datenbank – Teil 1



Welcome to Oracle Communities!

Einleitung

Microservices

Datenbank

Oracle Communities

Security

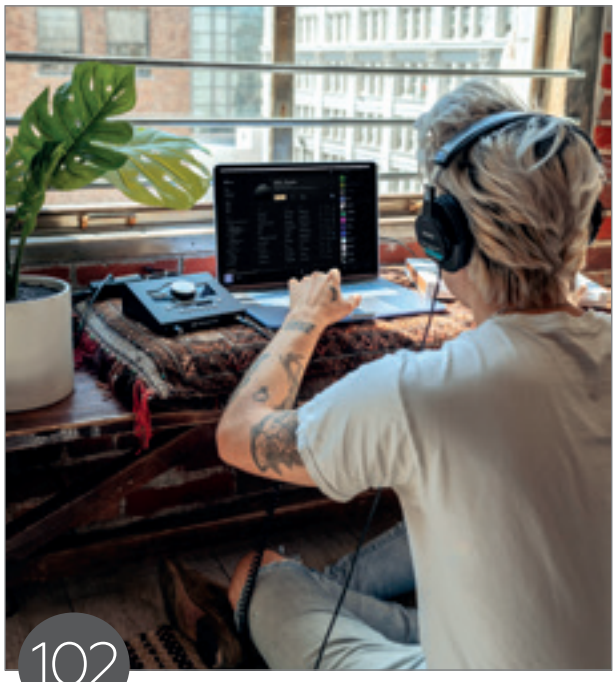
Cloud

PostgreSQL

Development

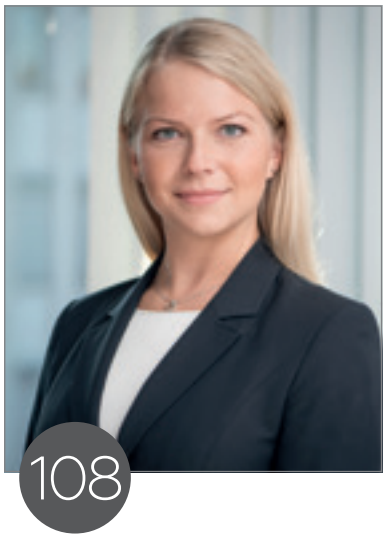
BUSINESS NEWS

Postpandemische Arbeitswelten



Leitartikel | Jetzt alles besser? Arbeitswelten nach Corona

Da steht ein Elefant im Raum



Postpandemische Arbeitswelten – Warum ein Zugehörigkeitsgefühl zum Unternehmen immer wichtiger wird

Raumbuchung trifft smarte Thermostate

Intern

News

TIMELINE

10. Februar 2023

In der DB WebSession zeigt Ernst Leber die Migration von Oracle 11 auf PostgreSQL mithilfe des Migrationstools ora2pg.

23. Februar 2023

Im DevTalk mit Jürgen Menge und Frank Hoffmann werden die „Möglichkeiten des Reportings in Oracle-Anwendungen (APEX, Forms, ...)“ besprochen. Es moderiert Carsten Czarski.

1. und 2. März 2023

Im zweitägigen Berliner Expertenseminar mit Florian Barth erwartet die TeilnehmerInnen der „Praxisworkshop Oracle Database Appliance“.

1. und 2. März 2023

In Hannover treffen sich die TeilnehmerInnen zu einem Hacking-Event für echte Techies: Das DOAG noon2noon 2023 zum Thema „Security“ mit dem Sicherheitsexperten Alexander Kornbrust.

9. März 2023

Im DevTalk mit Niels de Bruijn & Maarten Docter dreht sich alles rund um das Thema „Low-Code-Plattformen“. Es moderiert Kai Donato.

10. März 2023

In der WebSession mit Johannes Ahrends und Markus Flechtner wird das Thema „Multitenant Q&A“ behandelt.

21. bis 23. März 2023

2500 Java-Fans können im Phantasialand Brühl eine grandiose Stimmung auf der JavaLand 2023 genießen. Die TeilnehmerInnen erleben zwei ereignisreiche Event-Tage vollgepackt mit Vorträgen, Community-Aktivitäten, einer begleitenden Ausstellung, Workshops, Austausch und Networking mit einzigartigem Freizeitpark-Flair. Unternehmen reisen mit teilweise über 50-köpfigen Entwicklungsteams an. Ausgebuchte Sponsoring-Sessions und über 600 gebuchte Hotelzimmer spiegeln die herausragende Beliebtheit und Bedeutung der Java-Konferenz wider. Am dritten Tag der Konferenz erwartet die BesucherInnnen ein Schulungstag, bei dem Gelerntes intensiv vertieft werden kann.

22. März 2023

Die Regio NRW trifft sich zum Regionaltreffen NRW. Es referiert Torsten Rosenwald.

22. März 2023

„Kampf der Datenbanken: HA-Konzepte von Oracle, MySQL und PostgreSQL“ heißt das Thema des Treffens der Regio Rhein-Main in Wiesbaden.

23. März 2023

Im DevTalk geht es diesmal um das Thema „DB-Programmierung: Bessere Performance durch Tabellen-Indizes?“ Mit dabei sind Ulrike Schwinn, Carolin Hagemann, Jürgen Sieben und Randolf Eberle-Geist. Es moderiert Christian Brandes.

4. April 2023

Der SOUG Day findet an der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) in Winterthur statt und widmet sich dem Themenschwerpunkt „Security“.

6. April 2023

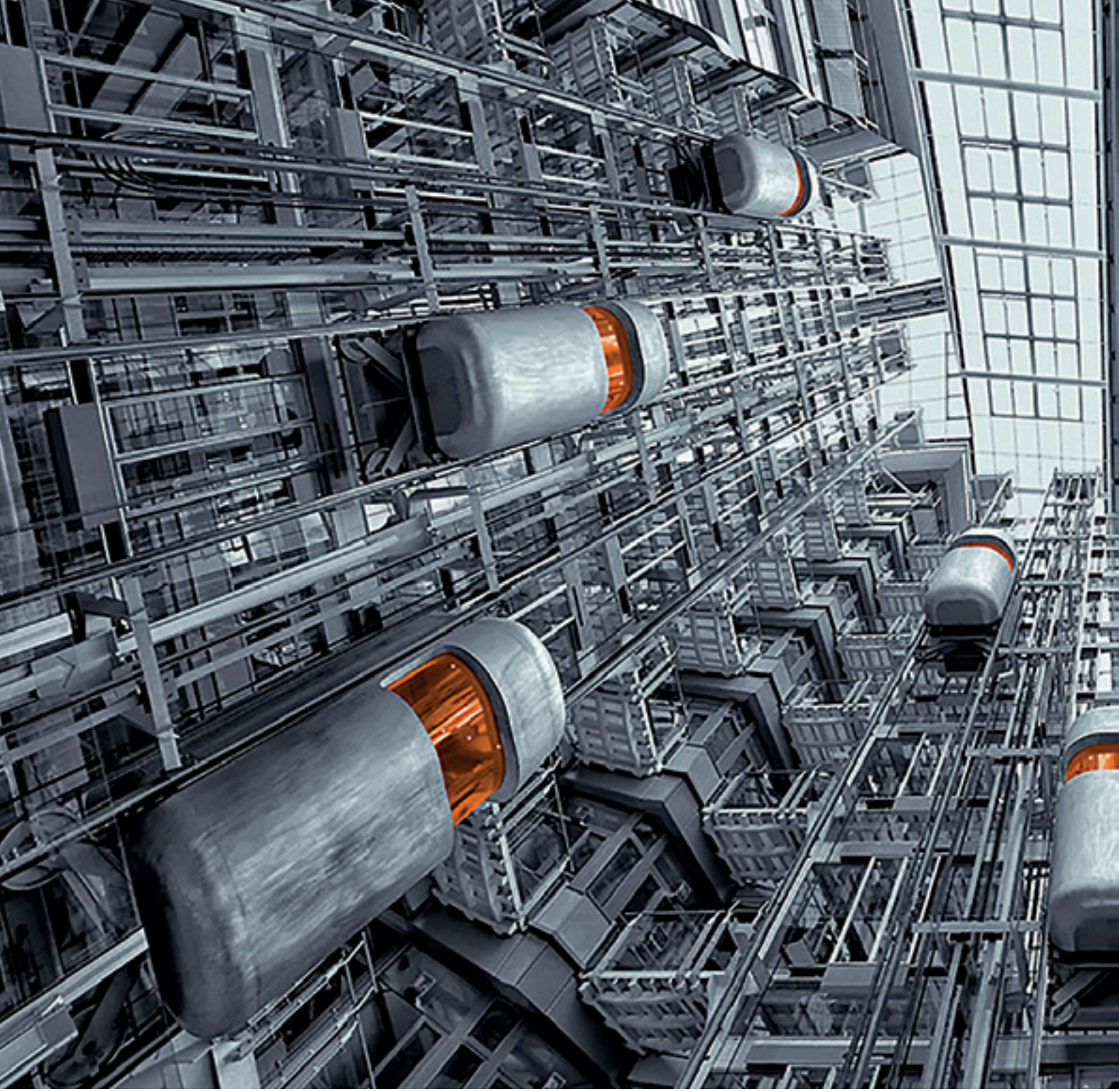
Über „Back-end Testing bei DB-Entwicklungsprojekten“ sprechen Oliver Lemm und Samuel Nitsche mit den TeilnehmerInnen im DevTalk.

13. bis 15. April 2023

Das DOAG Leitungskräfteforum und die Delegiertenversammlung finden in Berlin statt. Thema ist die Öffnung der DOAG für neue Themen und eine damit verbundene Satzungsänderung.

14. April 2023

In der WebSession mit Marco Pachaly-Mischke erfahren die TeilnehmerInnen alles rund um das Thema „Praxiserfahrungen mit Multitenant“.



„Microservices führen zu besserer, gezielter Skalierbarkeit, aber damit nicht zwangsläufig zu geringeren Kosten.“

Martin Meyer, Redaktionsleiter des Red Stack Magazin, sprach mit Christoph Lüttig, Andreas Badelt und Kilian Müller vom Greentech-Unternehmen Compleo Charging Software, einem Plattform-Anbieter für Elektromobilität, über die Migration zu und den Betrieb von Microservice-Architekturen, die Herausforderungen bei der Zusammenarbeit internationaler agiler Entwicklungsteams, Testen, Build- und Deployment-Pipelines sowie einige weitere Themen.

Wann habt ihr angefangen, mit Microservices zu arbeiten?

Christoph: Im Jahr 2018 haben wir angefangen, mit Microservices im Projekt zu arbeiten, die wir 2019 dann erstmals in Produktion ausgerollt haben.

Was war der Grund/Anlass?

Andreas: Zum einen begannen sich Performance- und Skalierbarkeitsprobleme abzuzeichnen, bedingt durch ein absehbar starkes Wachstum an E-Autos, Ladestationen und daraus resultierend Ladevorgängen. Es war klar, dass der Monolith bald an seine Grenzen kommt und wir an vielen Stellen eine grundsätzlich bessere horizontale Skalierbarkeit benötigen.

Auch die funktionale Komplexität und die Geschwindigkeitsanforderungen an die Entwicklung sind gestiegen. Dadurch standen sich immer größer werdende Entwicklungsteams „gegenseitig auf den Füßen". Ein unabhängigeres und möglichst auch häufigeres Releasen von neuen Funktionen war ein Ziel.

Wie habt ihr den Übergang zu Microservices geplant und was hat gut funktioniert, was nicht?

Kilian: Zunächst mal haben wir das „Strangler Pattern" eingesetzt und Stück für Stück Funktionalitäten aus dem Monolithen herausgeschält. Dadurch konnten wir die „Big Bang Migration" vermeiden. Andererseits hatten wir dadurch über einen längeren Zeitraum einen Parallelbetrieb, der auch eine Herausforderung war. Wir mussten unter anderem fortlaufend Daten zwischen altem und neuem System synchronisieren. Und unser Portal, in dem wir selbst und unsere Kunden Ladesäulen, Ladevorgänge, Tarife, Berechtigungen usw. ansehen und verwalten können, wurde während der Migration neu entwickelt, basierte aber dann teilweise noch auf dem alten System, weil die entsprechende Funktionalität noch nicht als neuer Service vorhanden war.

Andreas: Es wurde aber nicht alles komplett neu geschrieben. Dadurch, dass der Backend Code schon relativ gut modularisiert war und wir bei Java als Implementierungssprache geblieben sind (allerdings von Java EE zu Spring Boot migriert), haben wir aus Zeit- und Komplexitätsgründen einige größere Funktionen aus dem Monolithen herausgelöst und in Microservices verpackt. Das hat uns teilweise später auch eingeholt – manche Domänen und vor allem Services hätten wir auf der grünen Wiese sicher anders geschnitten; der Vorteil der Microservice-Architektur bleibt uns aber, sodass wir diese Dinge jetzt meist mit lokal begrenzten Änderungen angehen können. Unangenehm wird es natürlich auch – oder erst recht bei Microservices –, wenn verwandte Funktionalitäten über mehrere Services verteilt sind und nötige Änderungen sich durch alle hindurch ziehen. Aber zum Glück ist das nicht oft der Fall. Im Gegensatz zum Monolithen ist es allein technisch schon deutlich schwieriger, ungewollte Abhängigkeiten zu erzeugen. Und wir schauen natürlich beim Serviceschnitt genau hin und haben da sicher auch noch dazugelernt.

Wie habt ihr denn die Services geschnitten?

Andreas: Im Projekt waren einige Personen mit praktischer Erfahrung im Domain Driven Design aus vorherigen Projekten – wir haben von Anfang an darauf gesetzt. Für das Herauskristallisieren der Services der Domänen haben wir eine ganze Reihe von „Event-Storming"-Meetings durchgeführt. Dort konnten sich dann alle, die etwas beizutragen hatten, aber insbesondere natürlich die Fach-Experten, an einer riesigen Wand mit Klebezetteln austoben. Durch Clustering der Events und ihrer Verarbeitung quer durch das System entstanden dann nach und nach die Domänen und die (Root-) Aggregate, aus denen dann die Microservices entstanden sind. Wir haben das noch einige Male wiederholt, wenn wir zum Beispiel merkten, dass eine Domäne doch zu viel beinhaltete. Wir hatten beziehungsweise haben noch heute einige Entwicklungsteams, denen dann jeweils eine oder mehrere – idealerweise verwandte – Domänen zugeordnet wurden, in die sie sich dann im Laufe der Zeit auch immer besser fachlich eingearbeitet haben.

Kilian: Jede fachliche Domäne besitzt übrigens ihr eigenes, auf ihre Bedürfnisse zugeschnittenes Datenmodell. Dadurch entkoppeln wir die einzelnen Systeme, sodass sie ihre jeweiligen Kernaufgaben unabhängig von anderen Services und Domänen erfüllen können. Mittels eines asynchronen Replikations-Mechanismus werden Änderungen an Datenmodellen durch die jeweiligen Data-Owner publiziert und mittels Messagings an interessierte Domänen verteilt. Diese picken sich dann die relevanten Informationen heraus und bauen daraus ihr eigenes Datenmodell auf. Somit liegen Daten innerhalb der Plattform zwar teils dupliziert vor und es herrscht auch „nur" eine Eventual Consistency, wir steigern damit aber deutlich die Resilienz des Gesamtsystems.

Andreas: Ein Punkt, den wir in diesem Zusammenhang vielleicht ansprechen sollten, sind gemeinsame Definitionen. Die Fachlichkeit in Domänen mit ihrer jeweils eigenen „allgegenwärtigen Sprache" („ubiquitous language") zu unterteilen, sodass jede ihren eigenen Blick auf die Welt behalten kann, ist das eine. Das Objekt einer Ladestation mit seinen Attributen etwa sieht dann je nach Domäne – ob es um die Autorisierung von Ladevorgängen, das Konfigurieren der Station oder die Kommunikation mit ihr geht – komplett anders aus. Nichtsdestotrotz gibt es einige zentrale Attribute, insbesondere eine Reihe von Identifiern, die domänenübergreifend immer gleich sein sollten oder müssen. Wenn die Entwicklung über viele Teams verteilt ist, können die Dinge hier schnell auseinanderlaufen. Das haben wir durch ein zentrales Glossar im Wiki ganz gut gemanagt, aber einige Dinge sind uns doch durchgegangen und haben dann später zu nötigen Änderungen geführt, die sich durch viele Services gezogen haben.

Ihr spracht davon, dass sich die Entwicklungsteams gegenseitig auf den Füßen standen. Wie habt ihr die Teams entkoppelt? Wie stellt ihr sicher, dass parallel arbeitende Teams nicht aneinander vorbei arbeiten?

Andreas: Dafür nutzen wir unter anderem automatisierte Contract Tests für APIs. Wir haben sie für alle APIs eingeführt, die über Domänen-Grenzen hinweg genutzt werden. Die Erfahrungen waren über die Teams hinweg nicht nur positiv. Es gab technische Probleme mit dem Test-Framework insbesondere bei Updates, die den Aufwand in die Höhe getrieben haben. Aber der entscheidende Punkt ist, dass die Teams frühzeitig miteinander über die konkrete Gestaltung eines API reden müssen. Steht dieses dann fest und hat man es in einem Contract festgeschrieben,

kann jedes Team unabhängig gegen den Contract entwickeln, ohne auf das Partner-Team warten zu müssen.

Christoph: Außerdem tragen die Contract Tests am Ende dazu bei, dass wir Änderungen an einem Service, die sich auf die Schnittstellen auswirken, sehr schnell und (fast) voll automatisiert in Produktion bringen können. Manche Services werden bei uns mehrere Male pro Woche in Produktion deployt, wenn sie gerade intensiv weiterentwickelt oder Bugs entdeckt werden. Wenn es wirklich „brennen" sollte, auch zweimal kurz hintereinander.

Stichwort „Testen": Hat sich dort sonst viel bei euch verändert?

Andreas: Auf jeden Fall! Das Testen, insbesondere aber das Zusammenstellen und Konsistenthalten der Testdaten ist schon für einen großen Monolithen eine Herausforderung. Es wird in einer Microservice-Landschaft definitiv nicht einfacher. Ein vernünftiges Konzept, was lokal in einer Domäne getestet wird und welche übergreifenden Ende-zu-Ende-Tests erforderlich sind, und ein entsprechendes Zusammenarbeiten aller Tester ist die Voraussetzung. Eine große Herausforderung dabei ist das gemeinsame Entwickeln eines übergreifenden Testdaten-Setups für Ende-zu-Ende-Tests (idealerweise auf Knopfdruck reproduzierbar, ob über Automatisierung, Snapshots o. ä.). Dazu gehören natürlich die Freiräume, dass die einzelnen Teams jederzeit für ihre Integrationstests eigene Testdaten hinzufügen können, ohne anderen Tests in die Quere zu kommen.

Christoph: Ansonsten versuchen wir, die Automatisierung immer weiter zu treiben, über die gesamte Testpyramide hinweg. In jüngerer Zeit haben wir zum Beispiel für UI-fokussierte Ende-zu-Ende-Tests auf das Framework Cypress umgestellt und lassen diese Tests zeitgesteuert direkt aus der Build Pipeline gegen unsere Testsysteme laufen.

Oft wird die „Freie Wahl der Technologie pro MS" als Vorteil der Microservice-Architektur angeführt. Wie ist das bei euch und warum?

Andreas: Auch uns fordert der Spagat zwischen Makro- und Mikro-Architektur immer wieder aufs Neue heraus – oder das Verhältnis zwischen einer stabilen Grundlage, die alles zusammenhält, und dem Freiraum für Innovationen und maßgeschneiderte Lösungen.

Um es mal so auszudrücken: Eine gemeinsame technische Basis hilft! Wir setzen über das gesamte Backend hinweg auf Spring Boot, sodass hier zwischen den Teams auch ein großer Erfahrungsaustausch stattfinden kann. Kommunikation zwischen Services findet in der Regel über REST APIs oder Messaging per RabbitMQ statt. Auch bei den Persistenztechnologien versuchen wir, uns auf einige wenige zu beschränken, die große Masse der Services nutzt die relationale MariaDB oder die dokumentenzentrierte DocumentDB. Generell versuchen wir, den „Zoo" der Technologien möglichst klein zu halten und eine breite gemeinsame Basis zu haben. Aber wir geben den Teams auch ihre Freiräume, um besondere Bedürfnisse der Domänen angemessen umzusetzen oder neue Technologien auszuprobieren.

Kilian: Unglaublich hilfreich ist dabei unser SRE-Team, also Site Reliability Engineering, man könnte auch von einem zentralen Plattform-Team sprechen. Der Haupt-Fokus des Teams ist, den Entwicklerinnen und Entwicklern das Leben möglichst einfach zu machen. Das geschieht durch ein zentrales Source Code Management mit vorgefertigten, aber flexiblen Build- und Deployment Pipelines, die zum Beispiel verschiedene Java- und Spring-Versionen unterstützen, da die Services ja nicht alle auf dem gleichen Stand sein können. Auch automatisierte DB-Migrationen gehören dazu und seit Kurzem auch Canary Deployments – bei denen automatisch und allmählich die Instanzen mit der alten Version durch die neue Version ausgetauscht werden, wobei parallel bestimmte Kriterien ausgewertet werden, zum Beispiel die Fehlerrate für bestimmte Requests, und bei Problemen auch automatisch zurückgerollt wird. Dazu kommen unter anderem Management-Dashboards für die verschiedenen Umgebungen und ein kompletter „Observability Stack" basierend auf Grafana. In ihm werden aus sämtlichen Umgebungen Logs, Metriken und Traces (also die Verfolgung von Requests quer durch das System, auch über Messaging hinweg) bereitgestellt, damit sie ad hoc oder über generische sowie spezifisch von den Teams angepasste Dashboards ausgewertet und visualisiert werden können. Die Grafana-Dashboards werden übrigens nicht nur von den „Techies" genutzt, sondern viele visualisieren Geschäftskennzahlen und werden eher von Product Ownern, dem Produktmanagement etc. genutzt.

Christoph: Es gibt noch viele weitere Dinge, die das SRE-Team bereitstellt: Die umgebungsspezifische Konfiguration von Services sei hier noch als Beispiel genannt, die als zentraler Service bereitgestellt wird. Viele weitere hilfreiche Funktionen, etwa die Self-Service-Bestellung einer neuen Datenbank-Instanz oder der schnelle Überblick, welcher Service mit welcher Version in welcher Umgebung läuft inklusive Verweis auf die API-Dokumentation, werden den Teams über eine selbstentwickelte Self-Service-Plattform bereitgestellt. Außerdem gibt es über alle Oberflächen hinweg ein Single-Sign-on, sodass wirklich alle Werkzeuge schnell bereitstehen, wenn neue Kollegen „ongeboarded" werden. Last, but not least gehört auch die Unterstützung des Qualitätsmanagements mit automatisierten Qualitätskontrollen dazu.

Was genau ist damit gemeint?

Christoph: Damit ist hauptsächlich das gemeint, was in den Build und Deployment Pipelines abläuft. Die werden zentral vorgegeben, allerdings mit Freiheitsgraden, um den Teams nicht die Flexibilität zu nehmen. In diesen läuft eine ganze Menge ab. Zunächst natürlich das, was auch im lokalen Build abläuft: Kompilieren des Codes, das Ausführen der Unit-, Contract- und Integrations-Tests, also der entwicklerseitigen Tests, die Ermittlung der Code-Abdeckung und das Bauen der Images. Dazu gekommen sind noch die bereits erwähnten Cypress-Tests. Darüber hinaus haben wir das Generieren und Publishen von Dokumentation, speziell API-Doku, sowie die statische Code-Analyse inklusive des Zusammenstellens von allgemeinen Kennzahlen zur Code-Qualität, aber insbesondere von Informationen zu möglichen Sicherheitslücken im Code. Ergänzt wird das durch einen Dependency Tracker, der die direkten und transitiven Abhängigkeiten unserer Services analysiert und fortlaufend gegen offizi-

elle Vulnerability-Datenbanken prüft, sodass jede neu entdeckte Schwachstelle in verwendeten Bibliotheken sofort gemeldet wird. All diese Informationen sind auch in einem Deployment verfügbar und können gegebenenfalls auch dazu führen, dass das System ein Deployment in unsere Abnahmetest- oder Produktionsumgebung verweigert. Wobei es natürlich immer noch einen manuellen „Override" gibt, den bestimmte Personen genehmigen können – wenn wirklich mal ein Service dringend neu deployt werden muss, während „gleichzeitig" irgendeine größere Schwachstelle in einer Bibliothek bekannt geworden ist. Zum Glück standen wir aber noch nie vor dieser Abwägung. Aber die ständig neu entdeckten Schwachstellen werden natürlich insofern zum Problem, als die Abhängigkeiten unserer Services eigentlich regelmäßig geprüft und aktualisiert werden müssen, um die entsprechenden Fixes zu bekommen. Um diese mühsame Arbeit zu erleichtern, nutzen wir einen automatisierten Bot, der genau das den ganzen Tag lang macht und die Ergebnisse als Merge Requests im Source Code Management anlegt. Sofern es sich um Patch- oder Minor-Versionen handelt, werden diese beim erfolgreichen Durchlaufen der Pipelines direkt gemergt, ansonsten müssen diese Merge Requests durch die Teams manuell akzeptiert werden.

Andreas: Das ist alles viel Arbeit, aber gibt uns die Sicherheit, dass wir bei „log4shell reloaded" oder auch kleineren Sicherheitslücken frühzeitig informiert sind und genau wissen, ob wir etwas tun müssen und gegebenenfalls was. Wobei gerade bei Exploits wie log4shell natürlich auch andere Maßnahmen wichtig sind, wie zum Beispiel sicherzustellen, dass nicht alle Services nach Belieben kommunizieren und vor allem „nach draußen telefonieren" dürfen. Aber das ist auch wieder ein Thema für sich.

Lief nach der Migration alles glatt? Oder gab es neue Herausforderungen?

Kilian: Wenn wir behaupten würden, dass nach einer so großen Migration alles von Anfang an glatt gelaufen wäre, würde uns das sowieso niemand abnehmen. Ein paar Probleme haben wir ja oben schon genannt: Domänen, die zu groß geworden sind, ebenso wie einzelne Services. Allein durch die Dynamik der E-Mobilität sind aber auch schon wieder ganze Domänen hinzugekommen und wir können einzelnen Services beim Wachsen zusehen, bis wir irgendwann „entscheiden" (oder akzeptieren), dass es eigentlich zwei oder mehr Services sind. Das Testdaten-Setup für Ende-zu-Ende- oder Lasttests beschäftigt uns immer noch sehr stark. Ebenso wie wir fortlaufend an der Observability und an der Datensammlung für Business-Intelligence-Funktionen arbeiten.

Andreas: Was hinzukommt: Die ursprüngliche Migration war nur eine aus einer Serie. Wir sind vom Monolithen auf Services in Cloud Foundry in einem On-Premises-Stack migriert, um dann irgendwann auf Kubernetes in der AWS Cloud zu landen. Mal sehen, ob das für lange Zeit der letzte Schritt war.

Wie sieht euer organisatorisches Setup aus?

Kilian: Unsere agilen Entwicklungsteams sind nach dem Scaled Agile Frameworks (SAFe) organisiert. Während der Migration auf die Microservice-Architektur umfasste unser Agile Release Train (ART) bis zu 180 Personen in 17 parallel arbeitenden Entwicklungsteams. Heute arbeiten an der Plattform ca. 65 Mitarbeiter, verteilt auf 7 Entwicklungsteams, ein System-Test-Team, unsere bereits erwähnten Site Reliability Engineers (SRE-Team), ein Architektur-Team sowie Agile Coaches und einige weitere Querschnittsfunktionen, wie Chief Product Owner, Delivery- und Testmanager.

Jedes der crossfunktionalen Entwicklungsteams verantwortet eine oder mehrere fachliche Domänen und arbeitet nach agilen Prinzipien als Scrum- oder Kanban-Team. Die Teams setzen sich jeweils aus einem Product Owner sowie mehreren Software-Entwicklern und agilen Testern zusammen. Die Product Owner tragen die Verantwortung für die fachliche Weiterentwicklung ihrer Domänen. Die Entwickler kümmern sich um die Mikro-Architektur innerhalb der Domäne und die Entwicklung der Microservices sowie das Unit-Testing. Die agilen Tester übernehmen die funktionalen und integrativen Tests der eigenen Domäne beziehungsweise die Integrationstests mit umgebenden Domänen. Unterstützt werden die Teams zusätzlich durch das System-Test-Team, das gerade bei komplexen Test-Szenarien über mehrere Domänen und Ende-zu-Ende-Tests unterstützt.

Jedes der Teams ist im Übrigen nicht nur crossfunktional, sondern auch international besetzt – interne Mitarbeiter arbeiten hier Hand in Hand mit ihren Kollegen aus der Slowakei, Österreich, Spanien und Vietnam, die im Auftrag von Partnerunternehmen für die Compleo Charging Software GmbH tätig sind.

Was sind eurer Meinung nach die wichtigsten Punkte, die man beim Aufbau einer Microservices-Plattform beachten sollte?

Andreas: Wie vorhin schon erwähnt, ein sorgfältiger Schnitt der Domänen und Services. Wobei das immer ein Prozess bleiben wird, niemals statisch. Außerdem ein ausgewogenes Verhältnis zwischen Mikro- und Makro-Architektur und eine starke gemeinsame Plattform mit viel Self-Service, aber auch Innovations-Möglichkeiten.

Observability wird viel wichtiger, wenn es nicht um ein paar Instanzen von einem oder zwei Monolithen geht, sondern um fünfzig Services mit jeweils zwei oder mehr Instanzen. Entsprechende aggregierende Visualisierungen, aber auch mit der Möglichkeit eines schnellen Drill-Downs, sind wichtig, um Anomalien zu entdecken.

Christoph: Microservices führen zu besserer gezielter Skalierbarkeit, aber damit nicht zwangsläufig zu geringeren Kosten. Gerade wenn man nicht genau hinschaut, ist meist das Gegenteil der Fall. Daher ist ein Monitoring der genutzten Ressourcen und besonders das Limitieren der Ressourcen essenziell, um die Kosten in Grenzen zu halten.

Andreas: Es hilft auch zu schauen, ob ein Service wirklich ständig laufen muss oder nur sporadisch. Vielleicht kann er sogar als Lambda Function, Nano-Service oder wie auch immer man es nennen möchte nur bei Bedarf schnell hochgefahren werden. Die Startup-Zeiten sind auch mit Java und großen Frameworks wie Spring oder Microprofile heute meist kein Hindernis mehr. Dafür gibt es zum Beispiel „native images" oder Lösungen der Cloud-Anbieter wie „vorgewärmte Instanzen". Welcher Ansatz

für ein konkretes Szenario infrage kommt, ist natürlich immer eine Abwägungssache.

Abschließend: Hat sich der Umstieg gelohnt? Ist das Ziel der besseren Skalierbarkeit aufgegangen?

Kilian: Auf jeden Fall. Die Arbeit, die hineingeflossen ist, war schon enorm. Aber wir haben eine stabile Plattform, die sicher nicht perfekt ist, die aber in einem hochdynamischen, stark wachsenden Umfeld mitwächst – funktional wie Performance-technisch. Mit dem Monolithen wären wir mit ziemlicher Sicherheit nicht so weit gekommen.

Über die Compleo Charging Software GmbH
Die Compleo Charging Software GmbH (CCS) zählt zu den führenden Anbietern von Softwarelösungen zur Verwaltung von Ladestationen und damit zu den Treibern der Elektromobilität in Deutschland und Europa. Mit den SaaS-Lösungen sind heute bereits über 50.000 Ladepunkte in 18 Ländern direkt verbunden. Monatlich werden dabei über 200.000 Lade-Sessions an öffentlichen Ladestationen vermarktet. Im stark wachsenden Markt der Elektromobilität liegt der Fokus bei der Entwicklung verstärkt auf Skalierbarkeit, Ausfallsicherheit und Resilienz – Faktoren, die in den Cloud-basierten Lösungen der CCS mithilfe einer Microservice-Architektur erreicht werden.

ANDREAS BADELT

Andreas Badelt hat ein Informatikstudium an der Uni Dortmund absolviert sowie einen Masterstudiengang in Wirtschafts-Informatik an der Uni Bamberg und Duisburg-Essen. Nach vielen Jahren als Entwickler und Architekt für IT-Consulting-Unternehmen ist er seit 2016 freiberuflich unterwegs. Mit Elektro-Mobilität beschäftigt er sich intensiv seit 2019 – beruflich aktuell als Software-Architekt für Compleo. Daneben engagiert er sich seit gut zwei Jahrzehnten in der DOAG und hat dort seine Heimat in der Cloud Native Community gefunden.

CHRISTOPH LÜTTIG

Christoph Lüttig hat von 2015 bis 2019 ein duales Studium in IT- und Softwaresystemen am ITC in Dortmund absolviert. Er ist seit 2015 im Bereich der Elektromobilität tätig, zunächst als Entwickler, seit 2021 dann als Software-Architekt. Im Jahr 2022 übernahm Christoph die Leitung des Site Reliability Engineering Teams bei Compleo.

KILIAN MÜLLER

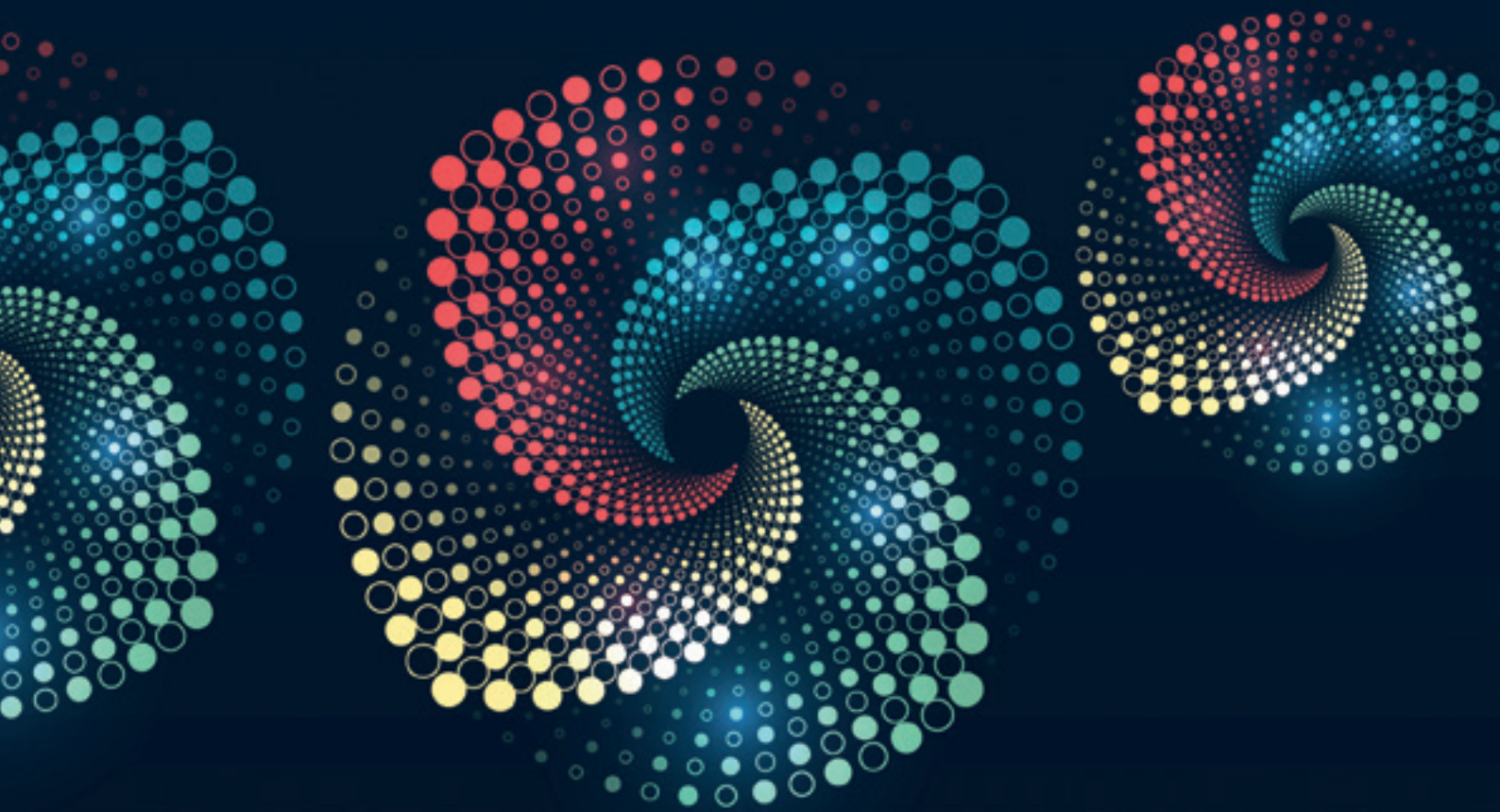
Kilian Müller hat sein Studium der Informatik an der TU Dortmund als Diplominformatiker abgeschlossen. Als Berater, Entwickler und Software-Architekt für einen IT-Dienstleister lernte er verschiedenste Branchen und Projekte kennen, bevor er 2020 zur Elektromobilität kam. Das e-Mobility-Backend der Compleo begleitet er als System-Architekt auf dem Weg vom On-Premises-Montolithen zur Microservice-Landschaft in der Cloud.

DOAG

DOAG Datenbank mit Exaday 2023

24. und 25. Mai in Düsseldorf

datenbank.doag.org



Oracle-Datenbanken für Microservices

Ralf Durben, Oracle Deutschland

Mit dem vermehrten Einsatz von Microservices stellt sich die Frage, ob und wie die Oracle-Datenbank in diesen Umgebungen eingesetzt werden kann. Welche Varianten sind verfügbar und wann ist es sinnvoll, sie einzusetzen?

Microservices-Umgebungen basieren derzeit zumeist auf der Kubernetes-Technologie, die durch den Einsatz diverser Tools zu einem Betriebsmodell inklusive Access Management, Monitoring und mehr komplettiert wird. Oracle bietet in diesem Zusammenhang mit seiner Lösung Verrazzano ein solches Betriebsmodell an.

In einer Kubernetes-Umgebung wird bekanntermaßen mit Containern gearbeitet, die einerseits miteinander kommunizieren, andererseits aber auch jederzeit austauschbar sein sollen, um dem Gedanken der agilen Entwicklung Rechnung zu tragen. Aus der Sicht der Datenbank gibt es dann also Datenbankcontainer und Container, die auf die Datenbankcontainer zugreifen – klassischerweise, aber nicht ausschließlich, Anwendungscontainer.

Die Oracle-Datenbank kann in einem Kubernetes-Container betrieben werden, wobei zu beachten ist, dass die dabei verwendeten Images dem Docker-Format entsprechen müssen und damit in Laufzeitumgebungen einsetzbar sind, die mit der Open-Container-Initiative kompatibel sind *[1]*. Der Vorteil dieser Vorgehensweise ist unter anderem, dass Gesamtsysteme, aus verschiedenen Bausteinen zusammengesetzt, in einer Kubernetes-„Laufzeitumgebung" betrieben werden können. So kann damit zum Beispiel ein IT-Verfahren sehr gut mobil gestaltet werden und damit „auf Reisen" gehen, sei es in einem Fahrzeug oder in einer mobilen Lokation.

Neben dem Betrieb von vielen (kleinen) Micro-Datenbanken in Kubernetes-Containern gibt es mit der Multitenant-Architektur aber auch die Alternative, Oracle- Datenbanken als Pluggable-Datenbank (PDB) in einer zentralen Container-Datenbank (CDB) zu betreiben. In diesem Fall könnte die CDB als Pendant zum Kubernetes-Cluster angesehen werden, also als Betriebsplattform für Datenbank- Container im „Oracle-Format". Gerade für Datenbanken, die nicht unter Volllast stehen, gibt es mit der Oracle-Multitenant-Architektur ja ein enormes Einsparpotenzial bezüglich aktiver Ressourcen (Memory und Prozessorleistung). So können derartige Datenbanken in viel höherer Anzahl auf einem (virtuellen) Server betrieben werden.

Es stellt sich bei dieser Variante natürlich die Frage zur Integration von Oracle CDB und Kubernetes, um eine zentrale Steuerung zum Provisionieren und Verwalten sowohl der Anwendungs- als auch der Datenbankseite umzusetzen.

Mit den passenden Operatoren können Sie aus Kubernetes heraus die Oracle-Datenbank

- als Single Instance in einem Kubernetes-Container
  (Oracle Database Operator for Kubernetes (OraOperator)) [9]
- als Sharded-Datenbank in Containern im Kubernetes-Cluster
  (Oracle Database Operator for Kubernetes (OraOperator)) [9]
- als Autonomous-Datenbank in der Oracle Cloud
  (Oracle OCI Service Operator for Kubernetes (OSOK)) [8]
- als PDB in einer CDB über die Oracle-Datenbank-REST-Schnittstelle

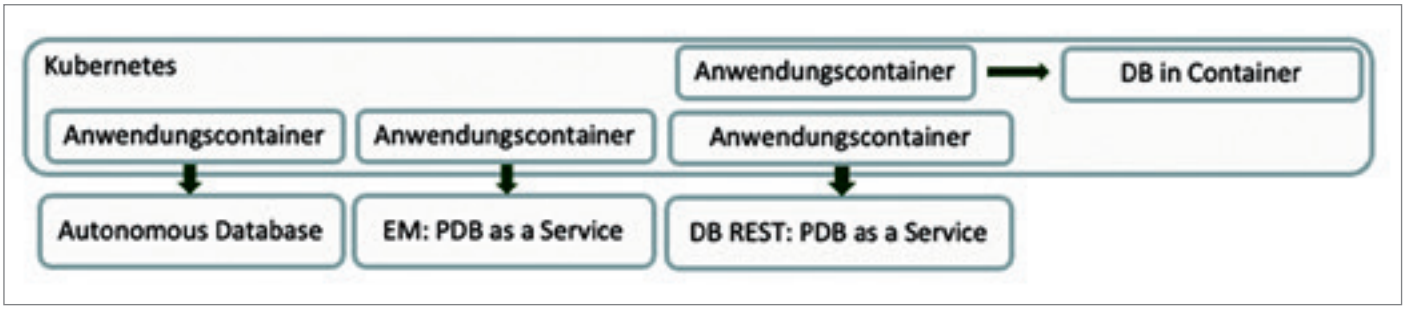


Abbildung 1: Verschiedene Varianten zur Nutzung der Oracle-Datenbank (Quelle: Ralf Durben)

(Oracle Database Operator for Kubernetes (OraOperator)) [9]

- als PDB in einer mit Oracle Enterprise Manager erstellten „PDB as a Service"-Umgebung
  (Kubernetes Operator for Oracle Cloud Management Pack (OracleCMP)) [10]

erstellen *(siehe Abbildung 1).*

Die Verbindung zwischen beispielsweise Anwendungscontainer und Datenbankcontainer findet über eine ConfigMap statt, die vom jeweiligen Operator für Kubernetes automatisch erstellt wird.

Alle oben genannten Operatoren, die neben der Bereitstellung natürlich den gesamten Lebenszyklus der Datenbank abdecken, also auch zum Beispiel das Starten, Stoppen und Löschen, stehen in GitHub zur Verfügung und werden im Database & Cloud Technology Blog *[4]* beschrieben.

Oracle-Datenbank im Kubernetes-Container

Bevor Sie Oracle-Datenbank-Container in einer Kubernetes-Umgebung erstellen können, benötigen Sie ein dazu verwendbares Image und den passenden Operator für Kubernetes, um nicht alle Aktionen des Lebenszyklus einer Datenbank selbst implementieren zu müssen. Für die Images gibt zwei Bezugsquellen von Oracle:

1. Oracle stellt auf der hauseigenen Container Registry [5] entsprechende Images zur Verfügung.
2. Oracle stellt auf GitHub [6] Skripte zur Verfügung, mit denen eigene Images erstellt werden können.

Für die Images beider Quellen gilt, dass die Datenbank-Container jeweils eine Container-Datenbank (CDB) und eine Pluggable-Datenbank (PDB) erstellen. Nur Letztere wird von der Anwendungsseite verwendet, die CDB dient dem Betrieb der PDB.

Die Unterschiede und Handhabung der beiden Quellen für Images sollten vor dem Einsatz verstanden und berücksichtigt werden. Auf den jeweiligen Webseiten beider Quellen wird die Nutzung (bzw. der Aufbau) der Images anhand von Syntax unter Verwendung von Docker beschrieben. Bekanntermaßen gibt es dazu mit Podman eine Alternative, die in diesem Bereich syntaktisch kompatibel ist. Die Nutzung der Images in Kubernetes ergibt sich durch die Verwendung von passenden YAML-Dateien und zum Beispiel dem „Oracle Database Operator for Kubernetes", der auch zunächst in Kubernetes installiert werden muss.

Beispiele dafür finden Sie unter anderem in sogenannten „Oracle Live Labs" *[7]*. Das Modul „Oracle Database Kubernetes Operator" gibt einen schnellen Einstieg hierzu und zeigt auch, wie der Oracle Database Operator in Kubernetes installiert wird.

Mein Tipp: Jedes Modul der Live Labs (und es gibt für fast alle Oracle-Themen ein entsprechendes Modul) können Sie in der Oracle Cloud (OCI) durchführen, wobei Sie für die Dauer dieses Moduls kostenfrei die dafür notwendigen Komponenten in der Oracle Cloud zur Verfügung gestellt bekommen. Entscheiden Sie sich für die Option „Run on Your Tenancy", erscheint eine Übungsanleitung, die auch im „heimischen IT-Umfeld" eingesetzt werden kann oder einfach nur zur Lektüre dient.

Oracle Container Registry

Die Images aus der Container Registry sind für den schnellen Einsatz gedacht. Mit einem „Pull" in einer YAML-Datei kann das Image bezogen und sofort eingesetzt werden. Es gibt derzeit (Stand Februar 2023) die Images für die Versionen 12.1.0.2, 12.2.0.1, 19.3.0.0 und 21.3.0.0 (hier auch Express Edition).

In *Listing 1* sehen Sie ein einfaches Beispiel für eine YAML-Datei und die Verwendung des „Oracle Database Operator for Kubernetes", mit der ein Oracle-Datenbank-Container gestartet wird.

Im obigen Beispiel wird ein Datenbank-Container unter Verwendung eines Oracle-19.3-Image gestartet. In der Startphase wird geprüft, ob es schon Datenbankdateien gibt. Falls ja, werden diese verwendet, falls nein, wird eine neue Da-

Credb.yaml

```
apiVersion: database.oracle.com/v1alpha1
kind: SingleInstanceDatabase
metadata:
  name: sidb-test1
  namespace: default
spec:
  sid: ORCL1
  edition: enterprise
  adminPassword:
    secretName: admin-secret
    keepSecret: true
  charset: AL32UTF8
  pdbName: orclpdb1
  flashBack: false
  archiveLog: false
  forceLog: false
  initParams:
    cpuCount: 0
    processes: 0
    sgaTarget: 0
    pgaAggregateTarget: 0
  image:
    pullFrom: container-registry.oracle.com/database/enterprise:19.3.0.0
    pullSecrets: oracle-container-registry-secret
  persistence:
    size: 100Gi
    storageClass: "oci"
    accessMode: "ReadWriteOnce"
  loadBalancer: true
  serviceAnnotations:
     service.beta.kubernetes.io/oci-load-balancer-shape: "flexible"
     service.beta.kubernetes.io/oci-load-balancer-shape-flex-min: "10"
     service.beta.kubernetes.io/oci-load-balancer-shape-flex-max: "10"
```

Listing 1: YAML-Datei für eine Oracle-Datenbank ohne Volume

```
persistence:
    size: 224Gi
    storageClass: "oci-fss"
    accessMode: "ReadWriteMany"
    volumeName: "nfs-vol-1"
```

Listing 2: YAML-Fragment für die Nutzung von Volumes

```
SOFTWARE_STAGE=<Zielverzeichnis fuer die Skripte>
mkdir -p $SOFTWARE_STAGE/GIT
cd $SOFTWARE_STAGE/GIT
git clone https://github.com/oracle/docker-images.git
```

Listing 3: Herunterladen der Skripte von GitHub

```
DEST=docker-images/OracleDatabase/SingleInstance/dockerfiles/19.3.0
cp LINUX.X64_193000_db_home.zip $SOFTWARE_STAGE/GIT/$DEST
```

Listing 4: Oracle-Datenbanksoftware zu den Skripten hinzufügen

```
cd $SOFTWARE_STAGE/GIT/$DEST/..
./buildContainerImage.sh -v 19.3.0 -e -t oracle/database:19.3.0 \
                         -o '--build-arg SLIMMING=false'
```

Listing 5: Image erstellen

```
EXT_D=docker-images/OracleDatabase/SingleInstance/extensions
P_ONEOFF=$EXT_D/patching/patches/one_offs
P_RU=$EXT_D/patching/patches/release_update
cp p6880880_190000_Linux-x86-64.zip $SOFTWARE_STAGE/GIT/$P_ONEOFF
cp p34411846_190000_Linux-x86-64.zip $SOFTWARE_STAGE/GIT/$P_RU
cd $SOFTWARE_STAGE/GIT/$EXT_D
./buildExtensions.sh -x patching \
  -b oracle/database:19.3.0 \
  -t oracle/database:19.17.0 \
  -o '--build-arg SLIMMING=false'
```

Listing 6: Image patchen

tenbank erstellt. Da im obigen Beispiel kein Volume angegeben wird, werden die Datenbankdateien innerhalb des Containers erstellt. Wenn dieser Container später abgebaut wird, ist damit auch die Datenbank gelöscht.

Es empfiehlt sich daher in aller Regel, die Datenbankdateien in externe Volumes zu legen. Der Persistenzblock in der YAML-Datei ändert sich dann entsprechend zum Beispiel wie folgt (hier ist das Volume ein Filesystem in der Oracle Cloud) *(siehe Listing 2)*.

Die Nutzung der Images aus der Oracle-Container-Registry ist also sehr einfach und kann spontan erfolgen. Es ist jedoch festzustellen, dass diese Images nicht dem aktuellen Patchstand entsprechen. Um aktuelle Releases zu verwenden, können Sie sich eigene Images erstellen.

Images selbst erstellen

Oracle stellt auf GitHub Skripte zur Verfügung, mit denen Sie Ihre Container-Images erstellen können. Dieses Skriptpaket bietet dabei auch die Möglichkeit, Patches in das Image einzubauen. Somit sind die damit erstellten Images auf der einen Seite variabler einsetzbar, auf der anderen Seite erfordern sie einige Vorarbeiten, bis sie einsetzbar sind.

Mit dieser Methode können Sie derzeit (Stand Februar 2023) die Images für die Versionen 11.2.0.2 (XE), 12.1.0.2 (EE/SE2), 12.2.0.2 (EE/SE2), 18.3.0 (EE/SE2), 18.4 (XE), 19.3.0.0 (EE/SE2) und 21.3.0.0 (EE/SE2/XE) erstellen. Neue Projekte sollten aber möglichst nicht mit mittlerweile nicht mehr im Support befindlichen Versionen begonnen werden!

Zum Erstellen eines Image benötigen Sie zum einen die oben erwähnten Skripte von GitHub und zum anderen die Datenbanksoftware (bzw. Patches).

Laden Sie zunächst die Skripte von GitHub herunter (Clone) *(siehe Listing 3)*.

Fügen Sie nun die Zip-Datei für die Datenbanksoftware hinzu. Die Datei wird in das für die jeweilige Datenbankversion vorgesehene Verzeichnis kopiert (hier für 19.3) *(siehe Listing 4)*.

Jetzt erstellen Sie das Image mit dem vorgefertigten Skript „buildContainerImage.sh“. Sie benötigen dazu entweder die Software Docker oder Podman *(siehe Listing 5)*.

Die Option „-o'--build-arg SLIMMING=false'“ ermöglicht es Ihnen, dieses Image später zu patchen. Mit der Option -v geben Sie die Datenbankversion an. Die Option -e steht für die Enterprise Edition (-s für Standard Edition, -x für Express Edition).

Nachdem das Image erstellt ist, können Sie dieses durch Patchen auf einen aktuellen Stand bringen, hier im Beispiel mit Release Update 17. Sie müssen die ZIP-Datei für das passende Opatch und vom Release-Update in die dafür vorgesehenen Verzeichnisse kopieren. Sie können mehrere One-off-Patches gleichzeitig einspielen, jedoch nur ein RU-Patch. Sollen mehrere RU-Patches eingespielt werden (zum Beispiel RU DB Patch und RU OJVM Patch), muss dieses nacheinander erfolgen *(siehe Listing 6)*.

Wenn Sie nun dieses Image in einer eigenen, von Kubernetes aufrufbaren Image-Registry platzieren, können Sie in der oben beschriebenen YAML-Datei einfach nur den Image-Namen durch Ihr eigenes Image ersetzen und so einen neuen Datenbank-Container starten oder einen bestehenden Container patchen. Dies funktioniert auch, wenn der bestehende Container mit ei-

nem Image aus der Oracle-Container-Registry erstellt wurde.

Wenn Sie ein altgepatchtes Image durch ein neugepatchtes Image austauschen und den Container neu starten, wird die bestehende Datenbank auch inhaltlich gepatcht, also das Tool „datapatch" ausgeführt.

In den obigen Beispielen wurde die Oracle-Datenbank als Single Instance erstellt. Auch eine RAC-Datenbank kann in Containern betrieben werden. Galt dies früher nur für Test und Entwicklungsumgebungen, so wird es ab Oracle 19.16 sogar auch produktiv unterstützt *[2]*. Ein eigenes Handbuch *[3]* beschreibt die Vorgehensweise.

Oracle-Datenbank als PDB in zentraler CDB

Wird eine Oracle-Datenbank in einem Kubernetes-Container betrieben, so befindet sich in jedem dieser Container jeweils eine CDB und eine PDB. Da jede CDB (mindestens) eine Instanz hat, bedeutet dies, dass damit immer auch aktive Ressourcen (Memory und Prozessorleistung) verbraucht werden.

So wird zum Beispiel Memory allokiert, der nicht mehr anderweitig zur Verfügung steht, auch wenn die Datenbank gar keine Last bekommt. Mit einer wachsenden Anzahl derartiger Container stellt sich die Frage, ob das nicht auch effizienter geht.

Des Weiteren muss darüber nachgedacht werden, ob es wirklich sinnvoll ist, den allgemeinen Betrieb von Datenbanken, also alles, was nicht projektabhängig ist, in die Hände dezentraler Teams zu legen. Ein allgemeines Monitoring (Performance und Compliance) oder regelmäßige Backups (bzw. eventuelles Recovery) sind in einem zentralen Team vielleicht besser aufgehoben.

Dazu kommt noch die Bereitstellungszeit: Für den Fall, dass eine Oracle-Datenbank in einem Kubernetes-Container erstellt werden soll, ist es wichtig zu verstehen, dass dafür eine CDB mit einer PDB erstellt wird. Es findet also ein CREATE DATABASE statt, das bekanntermaßen seine Zeit braucht. Wenn es mal wirklich schnell gehen soll, wäre es viel ökonomischer, wenn nur eine neue PDB in einer bestehenden CDB erstellt wird – mit einer Bereitstellungszeit von nur wenigen Minuten. Mit der Multitenant-Option können Sie ja auch mehr als vier PDBs pro CDB betreiben.

Neben der schnelleren Bereitstellung (nur eine PDB) gibt es auch noch weitere Vorteile dieses Ansatzes, vor allem für den Betrieb: Zentrale CDBs, eingebunden in bestehende Monitoring- und Compliance-Prüfungen oder Backupverfahren, sind ein stabiler Faktor eines Datenbankbetriebs. Die Dynamik von neuen und wieder verschwindenden PDBs kann so gestaltet werden, dass für den Betrieb dadurch kein Mehraufwand entsteht.

Wenn zum Beispiel ein „PDB as a Service" mit Oracle Enterprise Manager (mit Cloud Management Pack) betrieben wird, werden in EM für neue PDBs automatisch die vorgegebenen Regeln für den Betrieb angewendet. Gerade durch die in der letzten Zeit auftretenden Einbrüche in IT-Systeme werden Lösungen zur Sicherheitsüberprüfung auch von Datenbanken stärker nachgefragt und können hier durch automatisierte Compliance-Prüfungen umgesetzt werden. Zusätzlich steht auch eine Lösung für eine Kostenverrechnung zur Verfügung.

Fazit

Oracle-Datenbanken können im Microserviceumfeld eingesetzt werden – sowohl als Datenbank innerhalb von Containern als auch unter Einbindung von zentral verwalteten Datenbanklandschaften. Die konkret vorliegenden Anforderungen bestimmen letztlich die Umsetzung.

Quellen

[1] Oracle MOS Note „Oracle Support for Database Running on Docker" (Doc ID 2216342.1)

[2] Oracle MOS: Note „Oracle RAC on Docker - Released Versions and Known Issues" (Doc ID 2488326.1)

[3] Handbuch: Real Application Clusters Installation Guide for Docker Containers Oracle Linux x86-64 - *https://docs.oracle.com/en/database/oracle/oracle-database/19/racdk*

[4] Marcel Boermann-Pfeiffer: Database & Cloud Technology Blog - *https://blogs.oracle.com/coretec/post/3x-k8s-operator-for-oracle-database*

[5] Oracle Container Registry: *https://container-registry.oracle.com*

[6] GitHub: *https://github.com/oracle/docker-images*

[7] Oracle Live Labs: *https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/home*

[8] Oracle OCI Service Operator for Kubernetes (OSOK): *https://github.com/oracle/oci-service-operator*

[9] Oracle Database Operator for Kubernetes (OraOperator): *https://github.com/oracle/oracle-database-operator*

[10] Kubernetes Operator for Oracle Cloud Management Pack (OracleCMP): *https://github.com/ilfur/k8s_operator_cloudmg-mtpack*

Über den Autor

Ralf Durben ist als Solution Engineer bei Oracle Deutschland im Public-Sektor tätig. Zuvor war er über viele Jahre zentraler Ansprechpartner für die Themen Datenbankverwaltung und Oracle Enterprise Manager und hat dazu regelmäßig auf DOAG-Veranstaltungen vorgetragen.

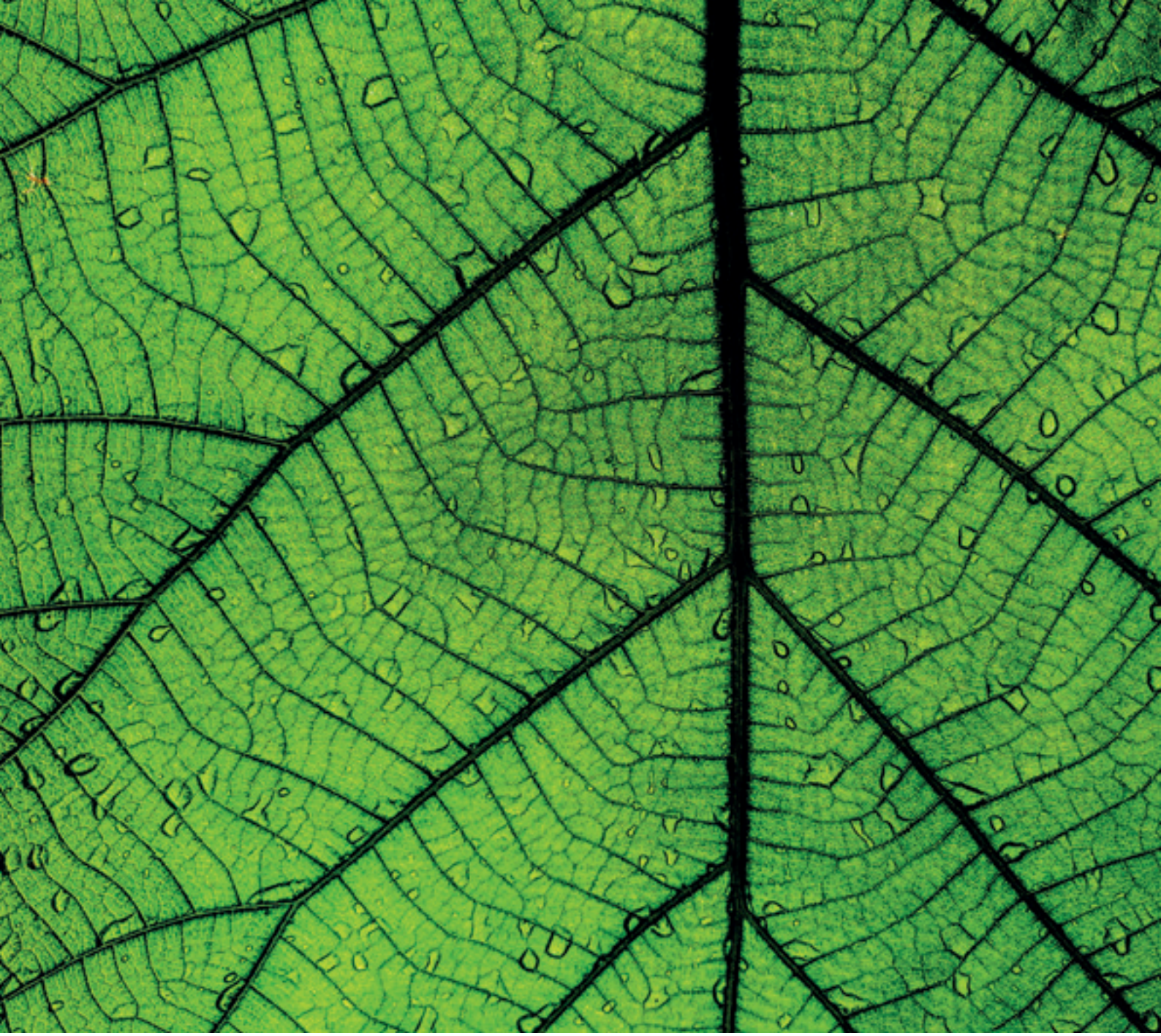
Ralf Durben
Ralf.Durben@oracle.com

European NetSuite User Days

IN NÜRNBERG

netsuite.doag.org

DOAG



Microservices – Effizient und sinnvoll einsetzen

Markus Schwabeneder, razzfazz.io, sage

Microservices sind ganz klar auf dem Vormarsch. Besonders die Trends zu SaaS und Cloud-Produkten befeuern diesen Trend noch zusätzlich, da für deren Nutzung eine Service-orientierte Architektur große Vorteile bringt.Der folgende Artikel soll dem geneigten Leser einen Überblick darüber bieten, was man bei der Konzeption einer Microservice-Architektur, der Entwicklung und Wartung von Microservices und deren Betrieb beachten muss, um die Vorteile zu betonen und die Nachteile abzuschwächen. Zusätzlich werden noch Kriterien geliefert, die dem Leser bei der Entscheidung helfen können, ob und wie man monolithische Software in Microservices aufspaltet.

Was verstehen wir unter Microservices?

Ein Microservice ist ein abgeschlossenes und unabhängiges Programm, das für genau einen bestimmten Zweck eingesetzt wird. Die Kommunikation mit dem Microservice erfolgt über sprachunabhängige API-Schnittstellen (in den meisten Fällen haben sich REST-Webservices durchgesetzt).
Folgende Eigenschaften haben Microservices:

- Sie haben einen bestimmten, eng begrenzten Zweck.
- Implementierungsdetails sind nach außen hin nicht bekannt, die Schnittstellen bieten keine Hinweise darauf.
- Sie sind isoliert und unabhängig lauffähig, auch mehrere Instanzen eines Microservice können nebeneinander laufen.

Welche Vorteile bieten Microservices beziehungsweise eine Architektur, die auf Microservices aufbaut?

Unabhängige Entwicklung

Die Implementierung eines Microservice kann durch ein relativ kleines Entwicklungsteam ohne großen Abstimmungsaufwand (zumindest im Vergleich zu Modulen, die Teil eines großen monolithischen Programms sind) erfolgen. Die Entwicklung kann dadurch auch sehr gut an externe Dienstleister ausgelagert werden oder der Service generell zugekauft werden.

Unabhängiger Betrieb

Der Microservice kann auf genau angepassten Umgebungen betrieben werden. Auch der Betrieb kann hier einfach bei Bedarf ausgelagert werden.

Geringe Komplexität

Durch den eng abgesteckten Einsatzzweck mit eng abgesteckter Funktionalität bestehen die einzelnen Microservices aus relativ kurzem, einfachem und verständlichem Code.

Einfache Änder- bzw. Erweiterbarkeit

Solange die API-Schnittstellen sich nicht ändern, kann ein Microservice verändert, erweitert oder komplett ausgetauscht werden, ohne dass sich Auswirkungen auf das Gesamtsystem ergeben.

Sehr gute Skalierbarkeit

Bei einem richtig geplanten System, das auf einer Microservice-Architektur aufbaut, können die einzelnen Microservices bei Bedarf mehrfach gestartet und so die Last einfach verteilt werden.

Große Robustheit des Gesamtsystems

Ein Ausfall eines Microservice bedeutet nicht den Ausfall des Gesamtsystems.

Nachteile gegenüber einem großen, monolithischen Programm

Größere Komplexität des Gesamtsystems

Zwar sind die einzelnen Microservices selbst relativ einfach und wenig komplex, das Zusammenspiel der unterschiedlichen Komponenten kann allerdings schwierig zu durchschauen sein und erfordert gute Planung und auch gute Dokumentation.

Potenzieller Kommunikationsoverhead

Die Kommunikation über API-Schnittstellen zwischen den Microservices bedeutet im Normalfall einen Mehraufwand gegenüber dem Datenaustausch von Modulen innerhalb eines monolithischen Programms.

Zusätzlicher Bedarf an Netzwerkressourcen

Da die Microservices typischerweise miteinander über das Netzwerk kommunizieren, wird dies zusätzlich belastet.

Welche Paradigmen müssen bei der Architektur des Gesamtsystems und der Implementierung der einzelnen Microservices beachtet werden?

Um die Vorteile optimal zu nutzen und die Nachteile bestmöglich zu mildern, ist es wichtig, folgende Punkte zu beachten:

„Do one thing and do it well"

Jeder Microservice sollte nur eine bestimmte, eng gesteckte Aufgabe übernehmen.

Hohe Robustheit der einzelnen Microservices

Die einzelnen Microservices müssen mit nicht erreichbaren API-Schnittstellen und höheren Latenzen umgehen können. Die über die Schnittstellen ausgetauschten Daten müssen immer als potenziell „dirty" betrachtet werden, fehlerhafte Aufrufe beziehungsweise Antworten der API-Schnittstellen müssen über eine saubere Fehlerbehandlung abgefangen werden.

Wahl der richtigen Technologie und Infrastruktur für die einzelnen Services

Bei der Auswahl der Programmiersprache und Ähnlichem sollte man Rücksicht auf die Bedürfnisse der zu erledigenden Aufgaben und auf die Fähigkeiten des Umsetzungsteams nehmen, nicht jedoch auf die anderen Microservices.

Bei der Wahl der Infrastruktur kann man oft kostengünstigere Setups treffen, als dies bei einem monolithischen System möglich wäre. So kann man je nach Relevanz, Auslastung und verarbeiteten Daten verschiedene Server mit unterschiedlicher Ausfallsicherheit und Leistungsfähigkeit wählen. Es bietet sich auch die Möglichkeit, DSGVO-relevante beziehungsweise besonders sicherheitsrelevante Daten in anderen Umgebungen als die übrigen Daten zu verarbeiten. Durch solche Trennungen allein erhöht sich die Sicherheit, es spart aber auch Kosten, wenn nicht alle Microservices auf teuren, hochperformanten, ausfallsicheren Servern unter speziellen Sicherheitsbedingungen laufen müssen.

Als Beispiel soll hier folgende vereinfachte Skizze eines Gesamtsystems für einen Webshop dienen *(siehe Abbildung 1)*.

Zustandslosigkeit

Jeder Aufruf der API-Schnittstellen ist atomar und unabhängig, die für eine Aufgabe benötigten Daten müssen daher bei jedem Aufruf mitgeschickt werden. Dies bedeutet zwar etwas Kommunikationsoverhead, ermöglicht jedoch eine einfache Skalierbarkeit und größere Robustheit. Services können so ohne großen

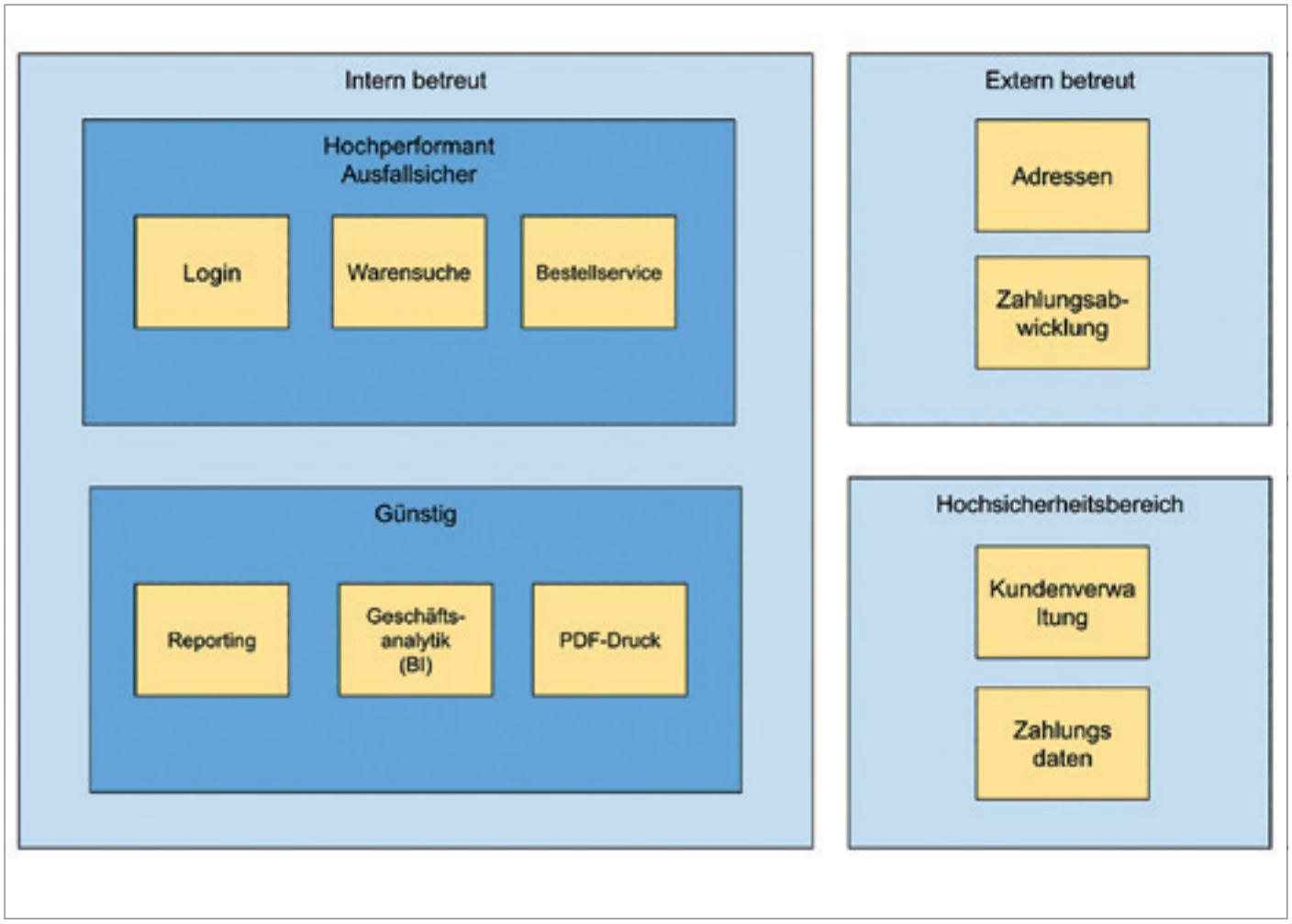


Abbildung 1: Skizze eines Gesamtsystems für einen Webshop (Quelle: SEQIS GmbH)

Aufwand neu gestartet werden, mehrere Instanzen eines Service können parallel laufen und die Last kann einfacher verteilt werden, ohne dass man sich um die Synchronisierung von Zuständen kümmern muss.

Sorgfältige Dokumentation

Das Gesamtsystem der verschiedenen Services kann sehr komplex werden und die Abhängigkeiten und Aufgaben der einzelnen Services sind manchmal nicht offensichtlich. Darum sind eine sorgfältige Dokumentation der API-Schnittstellen, eine Beschreibung und Auflistung der Aufgaben der einzelnen Systeme wichtig.

Auch die Architektur des Gesamtsystems muss beschrieben werden, im Idealfall unterstützt mit Diagrammen, die Abhängigkeiten illustrieren und einfacher sichtbar machen.

Wie stellt man einen effizienten Betrieb und einfache Wartbarkeit des Gesamtsystems sicher?

Verteilung der Last, Virtualisierung und Containerisierung

Die einzelnen Services sollten virtualisiert und in OS-Containern betrieben werden. Tools zur Lastverteilung sollten eingesetzt werden und den Services vorgeschaltet sein.

Nutzung von Orchestrierungstools

Es gibt etliche Orchestrierungsaufgaben, die im Betrieb einer Microservice-Architektur anfallen:

- Überwachung der Services
- Sammlung und Archivierung von Log-Einträgen der Services
- Starten und Stoppen der Services
- Überwachung des Ressourcenverbrauchs der OS-Container
- Anpassung der Ressourcen für die OS-Container
- Anpassung der Anzahl der laufenden Instanzen der Services an die aktuelle Last

Es gibt Tools, die diese Aufgaben automatisieren oder zumindest erheblich erleichtern können. Prominente Vertreter dieser Tools sind Kubernetes und Docker Swarm.

Die Implementierung eines gemeinsamen „Cockpits“, in dem diese Aufgaben zentralisiert übernommen werden können, ist mit modernen Tools möglich und sinnvoll.

Einsatz von Continuous Integration und Continuous Delivery

Im Gegensatz zu typischen „Big-Bang“-Deployments bei großen monolithischen Programmen werden Änderungen in Microservice-Architekturen typischerweise in mehreren kleinen Updates einiger weniger Services ausgeliefert. Die Gesamtzahl dieser Auslieferungen ist in Summe

Heuristiken zur Aufspaltung des Gesamtsystems in Microservices

Indikatoren, die für ein Aufspalten des Service sprechen:

- Zwischen Teilen des Service gibt es keine Abhängigkeiten.
- Manche Codeteile eines Service sind sehr volatil, während andere sich kaum verändern.
- Manche API-Aufrufe sind sehr ressourcenintensiv, während andere kaum Aufwand bedeuten.
- Manche API-Aufrufe haben konstante Last, während andere mal sehr intensiv, dann wieder kaum genutzt werden.
- Teile des Service müssen hochverfügbar sein, bei anderen Teilen kann mit Ausfällen gut umgegangen werden.
- API-Aufrufe werden von unterschiedlichen Nutzern verwendet (z. B. sollten das API für eine Webpage und das API für eine App nicht von ein und demselben Service zur Verfügung gestellt werden).
- Nur Teile des Service behandeln hochsensible Daten.

Indikatoren, die gegen ein Aufspalten des Service sprechen:

- Änderungen am Code eines Teils ziehen Änderungen an allen anderen Teilen nach sich.
- Aufspaltung würde zu großer Code-Duplizierung führen (es gilt aber hier zu prüfen, ob es nicht sinnvoll ist, den gemeinsamen Code in einen eigenen Microservice herauszuziehen).

Tabelle 1: Heuristiken zur Aufspaltung des Gesamtsystems in Microservices

viel höher als bei einem monolithischen System und bedeutet aufgrund der verteilten Services mehr Aufwand. Dieser Aufwand ist jedoch hochgradig automatisierbar und es gibt eine große Menge an CI/CD-Lösungen am Markt, die so etwas übernehmen können.

Wie sichert man die Qualität des Gesamtsystems effizient?

Großer Fokus auf automatisierte API-Tests

API-Tests sind unabdingbar notwendig zur Qualitätssicherung des Gesamtsystems. Man muss jederzeit damit rechnen, dass Microservices anders implementiert oder ausgetauscht werden. Auch kann es sein, dass man auf die Testergebnisse (z. B. Unit-Tests) keinen Zugriff hat, weil der Service extern entwickelt und/oder betreut wird.

API-Tests sollten unbedingt automatisiert und idealerweise in eine CI/CD-Pipeline inkludiert werden.

Hier machen sich die Eigenschaften der Microservices ganz besonders bezahlt:

- Die Forderung nach Zustandslosigkeit führt dazu, dass Testfälle fast immer nur aus einem einzigen API-Aufruf bestehen.
- Die Forderung nach einem sehr eng abgesteckten Aufgabenbereich führt dazu, dass nur wenige Testfälle für jeden einzelnen Service benötigt werden.
- Möglichst konstante API-Schnittstellen führen zu wenig Wartungsaufwand bei den Testfällen.

Einsatz von „Mock"-Services

Beim Durchführen der API-Tests für einen einzelnen Microservice muss sichergestellt werden, dass potenziell gefundene Fehler wirklich von diesem Service stammen und nicht aufgrund von fehlerhaften anderen Services, von denen der zu testende Microservice abhängt. Um dies sicherzustellen, müssen in der Testumgebung alle ausgehenden Aufrufe an sogenannte Mock-Services geleitet werden. Diese müssen nicht die Funktion der produktiv im Einsatz befindlichen Services erfüllen, sondern nur auf die Aufrufe des zu testenden Systems die für den Test passenden Antworten liefern. Im Idealfall ist so etwas relativ leicht mittels einer Key-Value- Datenbank umsetzbar, die eben zu jedem Aufruf (Key) die passende Antwort (Value) liefert *(siehe Tabelle 1)*.

Über den Autor

Markus Schwabeneder ist Principal Consultant, Chief Architect bei razzfazz.io und Mobile Lead-Developer bei sage.
Er begann seine Karriere in der Softwarebranche als Consultant für komplexe mathematische Themen und Optimierungsaufgaben. Seitdem ist er in den Bereichen Softwareentwicklung, Anforderungsanalyse, Testing und Softwarearchitektur tätig. Seine Fähigkeiten setzt er in unterschiedlichen Branchen ein. Besonders begeistert er sich für das Erarbeiten von komplexen Anforderungen und spezifischen Vorgehensweisen. In seiner Privatzeit begeistert er sich für Schach und Sport.

Markus Schwabeneder
markus.schwabeneder@seqis.com

Multitalent Multitenant

Susanne Jahr, Herrmann & Lenz Services

Die Multitenant-Architektur ist spätestens ab Oracle 21c Pflicht. Doch es ist eine gute Idee, schon jetzt bei neuen Umgebungen sowie Upgrade- und Migrationsprojekten den Schritt zur Multitenant-Architektur zu machen, denn diese ist gerade bei Migrationsprojekten ein wahres Multitalent. Hier werden Möglichkeiten bei der Migration auch großer Datenbanken (bereits PDBs oder auch Non-CDBs) in eine 19c-PDB oder beim Aufbau von Test-Datenbanken beschrieben. Es wird aufgezeigt, wie mit keiner oder minimaler Änderung an den Client-Konfigurationen Datenbank-Umstellungen schnell und sicher über die Bühne gehen können.

Multitenant-Architektur – muss ich wirklich...?

Kurze Antwort: JA! Die Multitenant-Architektur existiert bereits seit 12c, also seit fast 10 Jahren – der nun für 21c definitiv festgelegte Pflicht-Umstieg wurde schon seit der Einführung „for future releases" angekündigt. Die Multitenant-Architektur ist seit 12.2 gut, spätestens seit 18c sehr gut einsetzbar und transparent für so gut wie 100 % der Anwendungen. Für den Umstieg gibt es also viele Gründe, dagegen eigentlich keinen (mehr).

Anlage CDB/PDB

Die Anlage einer neuen Container-Datenbank unterscheidet sich kaum von einer in der traditionellen Architektur. Erforderlich sind lediglich der Parameter `enable_pluggable_database=true` im spfile sowie der `Zusatz enable pluggable database LOCAL UNDO ON` im CREATE-DATABASE-Kommando.

Für die Anlage einer Pluggable Database (PDB) gibt es mehrere Möglichkeiten:

- „from seed", also als Kopie des mitgelieferten Templates PDB$SEED
- per Unplug/Plug, also als Kopie oder Verschiebung einer bereits in einem anderen Container existierenden PDB
- per Remote Clone als Kopie einer bestehenden PDB oder Non-CDB über Database Link – dieses Verfahren ist bestens für Migrationen und den Aufbau von Testsystemen geeignet und wird daher hier später noch eingehender betrachtet.

Client-Zugriff

Der Zugriff ist unverändert gegenüber dem auf eine Non-CDB – vorausgesetzt, man verwendet bereits Service-Namen und nicht SIDs im Connect String (dies gilt für OCI-Zugriffe ebenso wie für jdbc-Connects).
Empfehlenswert ist die Verwendung von benutzerdefinierten Service-Namen anstelle der Default-Service-Namen (`db_unique_name.db_domain` für eine Non-CDB oder eine CDB bzw. `pdb_name.db_domain` für eine Pluggable Database). Die gewünschten Services können bei vorhandener Grid Infrastructure per srvctl, ohne Grid Infrastructure mittels dbms_service in der gewünschten PDB angelegt werden; der Start erfolgt dann bei Öffnung der PDB entweder über die GI oder zum Beispiel durch einen Startup-Trigger *(siehe Abbildung 1 und 2)*.
Hinweis: Der Instanz-Parameter `service_names` ist in 19c deprecated und soll nicht mehr manuell gefüllt werden!

Sollte der Einsatz von Service-Namen aufgrund von Anwendungsanforderungen oder Ähnlichem absolut nicht möglich sein, kann der Parameter `USE_SID_AS_SERVICE_<LISTENER_NAME>=ON` in der listener.ora gesetzt werden. Dieser bewirkt, dass eine SID in einem eingehenden Request als Service-Name interpretiert wird und der Zugriff somit doch noch funktioniert.

Durch die zusätzliche Verwendung von DNS-Aliasen für den Datenbank-Server kann die Anpassung der Client-tnsnames.ora-Dateien während der Migration auf ein Minimum begrenzt werden – der DNS-Alias kann dann einfach vom alten auf den neuen Host geschwenkt werden. Im Vorfeld ist dann allerdings gegebenenfalls eine Anpassung der Clients auf die Verwendung sowohl von Service-Namen als auch von DNS-Aliasen erforderlich.

Migrationen

Gegenüber dem Klassiker unter den Migrationswegen, dem Data Pump Export/Import, bietet das Remote Cloning nur wenige Nachteile, aber diverse Vorteile, insbesondere bei großen Umgebungen. In Datenbanksystemen von mehreren Hundert GB oder gar mehreren TB fallen die Vorteile des Remote Cloning besonders ins Gewicht; die dem gegenüberstehenden Vorteile des Data Pump müssen im Einzelfall abgewogen werden *(siehe Tabelle 1)*.

Voraussetzungen für ein Remote Cloning

- Quell-Datenbank:
  - PDB oder Non-CDB (min. 12.2)
  - Kommunikation mit Ziel-DB über Oracle Net möglich

```
srvctl add service -db mycdb -service mydb_svc -pdb mypdb
```

Abbildung 1: Anlage Service mit Grid Infrastructure (Quelle: Susanne Jahr)

```
alter session set container=mypdb;
exec dbms_service.add_service('mydb_svc','mydb_svc');
exec dbms_service.start_service('mydb_svc');
```

Abbildung 2: Anlage Service ohne Grid Infrastructure (Quelle: Susanne Jahr)

```
create pluggable database mypdb from <src_pdb>@<dblink>;
Ggfs. noch:
$ORACLE_HOME/Opatch/datapatch -verbose -pdbs mypdb -skip_upgrade_check
```

Abbildung 3: Remote Clone – Quelle PDB (Quelle: Susanne Jahr)

```
create pluggable database mypdb from NON$CDB@<db_link>;
alter session set container=mypdb;
@?/rdbms/admin/noncdb_to_pdb.sql
ggfs. noch:
$ORACLE_HOME/OPatch/datapatch -verbose -pdbs mypdb -skip_upgrade_check
```

Abbildung 4: Remote Clone – Quelle Non-CDB (Quelle: Susanne Jahr)

```
global.autoupg_log_dir=/u01/app/oracle/admin/DAT18/autoupgrade/log
dat18upg.source_home=/u01/app/oracle/product/18.0.0.0/dbhome_1
dat18upg.target_home=/u01/app/oracle/product/19.0.0.0/dbhome_1
dat18upg.sid=DAT18
dat18upg.source_dblink.DAT18=CLONEDB
dat18upg.target_cdb=CDB19
dat18upg.target_pdb_name.DAT18=PDB19
dat18upg.target_version=19.16
dat18upg.target_pdb_copy_option.DAT18=file_name_convert=NONE
```

Abbildung 5: Beispiel-Konfig-Datei Autoupgrade mit Remote Clone (Quelle: Susanne Jahr)

  - User mit dem CREATE PLUGGABLE DATABASE Privileg (Common User oder normaler User in Non-CDBs) installierte Komponenten Quell-(P)DB <= installierte Komponenten Ziel-CDB
- Ziel-Datenbank
  - CDB ab 12c; am besten 19c oder 21c
  - Database Link zur Quell-DB unter Verwendung des o. a. Users

Quell- und Zieldatenbank können sich auf demselben, aber auch auf unterschiedlichen Hosts befinden.

Die Durchführung des Remote Clone ist dann recht simpel *(siehe Tabelle 2 und Abbildung 3 und 4)*.

Migrationen – Sonderfall unterschiedliche Versionen

Haben Quell- und Zieldatenbank unterschiedliche Oracle-Versionen (z. B. 18c/19c), ist als zusätzlicher Schritt das Upgrade der neuen PDB erforderlich. Das Upgrade erfolgt nach dem Cloning auf der Zielseite. Vorteil hierbei ist, dass die Quellseite intakt bleibt und somit ein Fallback-Szenario gewährleistet ist.

Auch hier können Quell- und Ziel-Host identisch oder unterschiedlich sein. In der Quell-Datenbank müssen wie üblich zunächst Pre-Upgrade-Checks durchgeführt werden (`preupgrade.jar` oder – verpflichtend bei Upgrade nach 21c oder neueren Versionen – `autoupgrade.jar -mode analyze`). Empfehlenswert ist, immer die aktuelle Version des autoupgrade.jar zu verwenden; diese steht auf MOS zum Download bereit (Doc ID 2485457.1). Sowohl Upgrade als auch Patching sind In-Place und auch im Rahmen des Remote Cloning in Autoupgrade integriert. Die Voraussetzungen sind ansonsten identisch zum manuellen Remote Cloning *(siehe Tabelle 3)*.

Eine Beispiel-Konfig-Datei zeigt *Abbildung 5*.

Durchführung des Remote Clone mit Autoupgrade:

1. **Pre-Upgrade-Check:**
   $ORACLE_HOME/jdk/bin/java –jar autoupgrade.jar –mode analyze
2. **Gegebenenfalls erforderliche Fixups:**
   $ORACLE_HOME/jdk/bin/java –jar autoupgrade.jar –mode fixups
3. **Remote Clone und Upgrade:**
   $ORACLE_HOME/jdk/bin/java –jar autoupgrade.jar –mode upgrade

Das Upgrade der Datenbank mit `$ORACLE_HOME/bin/dbupgrade`, das erforderliche `Skript noncdb_to_pdb.sql` sowie die eventuell benötigten Post-Fixups werden durch Autoupgrade nacheinander automatisch anhand der Informationen aus der Konfig-Datei durchgeführt. Fortschritte sind sowohl an der Kommandozeile als auch im Browser abzulesen. Am Ende gibt es eine Zusammenfassung der Ergebnisse des Autoupgrade-Jobs.

Testumgebungen

Analog zur Migration von produktiven Umgebungen lassen sich auch Testumgebungen per Remote Clone schnell und einfach neu aufbauen. Ab Oracle 19c sind in der SE2 sowie der EE ohne Multitenant-Option bis zu 3 PDBs pro Container erlaubt. Es kann also eine neue PDB per Remote Clone einfach auch in ein bereits bestehendes Sys-

Vorteile Data Pump Export / Import	Vorteile Remote Cloning
Reorganisation von Tabellen und Indizes	insbesondere bei großen Systemen und/oder vielen LOBs deutlich schneller als Data Pump Export/Import, dadurch i. A. deutlich kürzere Downtime
weitere Aktionen möglich (Unicode-Migration, Remapping, Übernahme Subset der Original-Daten, ...)	keine großen zusätzlichen Archivelog-Mengen
über Plattformen und beliebige Oracle-Versionen hinweg möglich	kein zusätzlicher Platzbedarf für Dump-Dateien
	Umwandlung der alten in die neue Architektur „in einem Aufwasch“
	gleichzeitiges Upgrade möglich – manuell oder integriert in Autoupgrade – sogar refreshable
	im Fehlerfall schnell reproduzierbar/Fallback-Szenario

Tabelle 1

Zu beachten:

- *Keine Voraussetzung, empfohlen ist jedoch die Verwendung von Oracle Managed Files (OMF). Ansonsten müssen abweichende Pfade auf dem neuen System durch Anwendung von file_name_convert definiert werden.*
- *Kollidierende Service-Namen können durch die Anwendung von service_name_convert verhindert werden.*
- *Achtung: während des Cloning-Vorgangs dürfen auf der Quellseite keine aktuellen Archivelogs gelöscht werden – diese werden nach Abschluss der Kopie zum Recovery ähnlich wie bei einem RMAN-Restore benötigt!*

Tabelle 2

Zu beachten:

- *Der PDB-Creator-User auf der Quell-DB benötigt bei Verwendung von Autoupgrade zusätzlich das READ-Recht auf die Tabelle* `sys.enc$` *– auch wenn keine Encryption verwendet wird!*
- „…`The PDB created from the non-CDB must continue to use the source non-CDB name. You cannot change the name of the database.`“ `Quelle: Oracle Upgrade-Guide 21c`

→ *Falsch! Abweichende PDB-Namen sind möglich, müssen aber in der Config-Datei angegeben werden.*

Tabelle 3

tem integriert werden; ein Shutdown des Root-Containers oder eventuell bereits vorhandener weiterer PDBs ist nicht erforderlich. Derartige Aktionen sind einfach per Skript und zum Beispiel cronjob zu automatisieren. Zudem sind Pre- und Post-Aktionen einplanbar, etwa ein Export vorhandener Metadaten per Data Pump Export vor Beginn des Clone, Anpassung von Services und/oder Parametern nach Abschluss etc.

Susanne Jahr
susanne.jahr@hl-services.de

Mit Datenbank-Tuning die Welt retten? DBA als Klimaschützer!

Peter Hoffmann und Bojan Milijas, Oracle Deutschland

Unsere Gesellschaft wird aktuell verstärkt mit Herausforderungen in den Bereichen Klimaschutz, Umweltschutz und Ressourcenschonung konfrontiert. In vielen Bereichen, wie Stromproduktion und Mobilität, können wir auch bereits Fortschritte beobachten. In diesem Artikel widmen wir uns der Fragestellung, wie die IT und insbesondere der Betrieb der produktiven Datenbanken (und die Entwicklung von datenbankbasierten Anwendungen) zur Bewältigung dieser Herausforderungen beitragen können.

Wie viel Strom verbrauchen die Datenbanken?

Die Datenbanken bilden die Grundlage jeglicher Datenverarbeitung und sind der häufigste Workload-Typ in privaten, hybriden und öffentlichen Cloud-Rechenzentren weltweit *(siehe Abbildung 1)*.

Zum ähnlichen Ergebnis kommt die Umfrage über Workloads auf Kubernetes aus dem Jahr 2022, mit 76 % Databases, 67 % Analytics und 50 % AI *[1]*.

Laut Statista sind Server (mit Datenbanken als Workload #1) Stromverbraucher Nummer eins in der IT. Rechenzentren und kleinere IT-Installationen verbrauchten 16 Mrd. kWh im Jahr 2020, was 3 % des gesamten deutschen Stromverbrauchs ausmacht, mehr als Verkehr mit 2 %! In den Jahren 2016 - 2020 ist der Rechenzentrum-Stromverbrauch um 30 % gestiegen *[2] (siehe Abbildung 2)*.

Nun stellt sich die Frage, wie viel Strom eine Datenbank verbraucht oder wie viel Energie insgesamt allein die Datenbanken in der IT verbrauchen. In unseren von Virtualisierung und Ressourcen-Sharing geprägten IT-Landschaften ist diese Frage nicht leicht zu beantworten. Man kann den genauen Stromverbrauch einer Datenbank beziehungsweise Datenbank-Anwendung auf einem Rechner im Allgemeinen nicht isolieren und unabhängig von anderen auf der gleichen physikalischen Maschine laufenden Prozessen und Diensten messen und genau beziffern.

Energieverbrauch mit Exadata um ein Vielfaches reduzieren

Eine gesonderte Betrachtung und direkter „Vorher-nachher"-Vergleich des Datenbanken-Energieverbrauches ist immer dann möglich, wenn (meistens viele) herkömmliche Datenbank-Server durch (wenige) spezialisierte Datenbank-Maschinen wie zum Beispiel Oracle Exadata Database Machine oder Oracle Database Appliance ersetzt werden. Solche Konsolidierungsprojekte führen in der Regel zu Performance-Steigerungen, Reduzierung des Platzbedarfes und Beschleunigung der Antwortzeiten um Faktor 10, 50 oder gar 150 mit entsprechender Energieeinsparung und CO2-Reduzierung als direkter Folge *[3]*.

Abgesehen von solchen Spezialfällen, bei denen die komplette Rechenanlage dem Datenbanken-Betrieb dient, kann die Frage des Stromverbrauchs einer Datenbank auf einem herkömmlichen Rechner beziehungsweise „auf einem normalen Server" im Allgeneinen als „direkter durch Software verursachter Energieverbrauch" formuliert werden. Mit dieser Frage hat sich das Umweltbundesamt beschäftigt und dabei festgestellt, dass der Energieverbrauch durch elektrische Leistungsaufnahme über eine Zeitdauer entsteht und daher in einer Vielzahl von Fällen die Optimierung der Algorithmen, mit dem Ziel, die Rechenzeit zu verkürzen, zu Energieeinsparung führt *[4]*. Dies erleichtert die Aufgabe des DBA, als Klimaschützer die Welt zu retten. Der „gute alte DBA" kann die Umwelt schonen, indem er die Datenbank-Performance „wie seit eh und je" optimiert und dafür sorgt, dass die vorhandenen Ressourcen optimal ausgelastet sind. Die künstliche Intelligenz kann den DBA dabei gerne unterstützen. On Premises und in der Cloud. Egal ob auf Multipurpose-Servern, virtuellen Maschinen oder auf spezialisierten Engineered Systems von Oracle!

Intelligente Abfrageformulierung und -Optimierung sind das A und O

Folgendes Beispiel soll verdeutlichen, wie sehr sich unterschiedliche Formulierungen der gleichen Abfrage mit äquivalenten Ergebnissen auf die Laufzeiten und

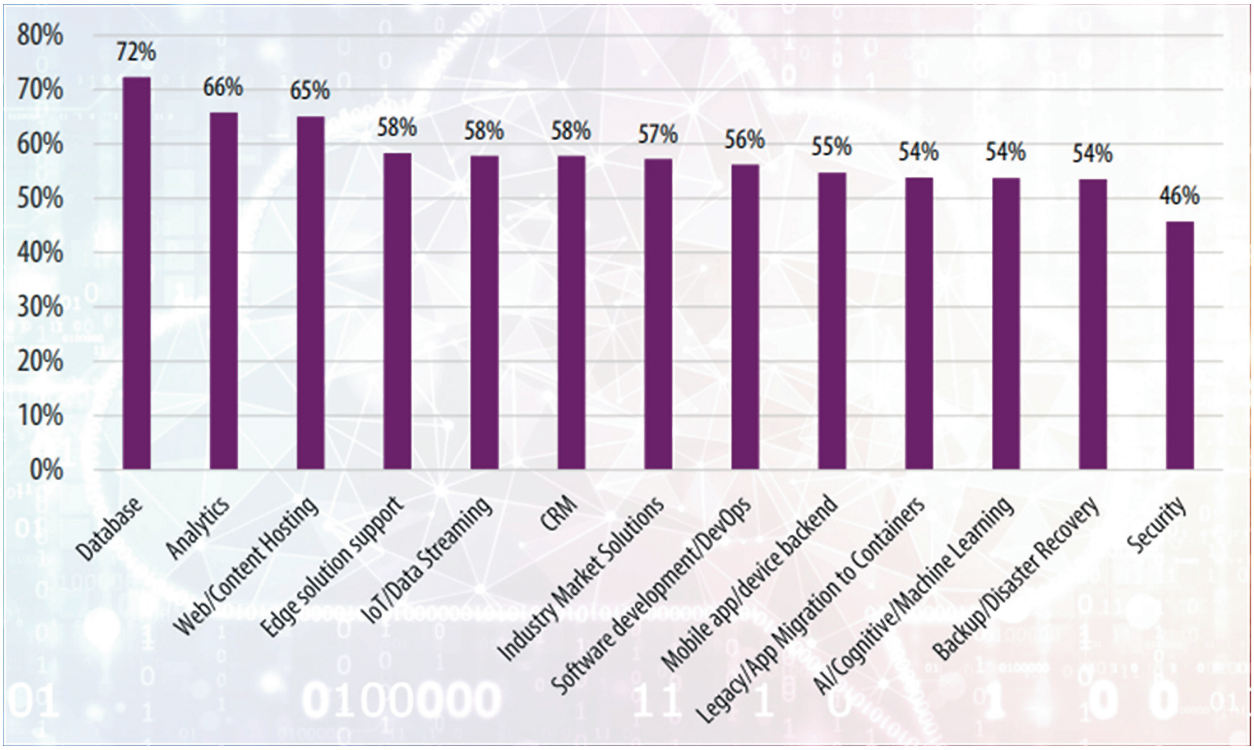


Abbildung 1: Häufigste Workload-Typen (© TECHnalysis Research)

```
-- #1 INTERSECT
SELECT *
  FROM sales_row
INTERSECT
SELECT *
  FROM sales;
Elapsed: 00:00:01.42
```

```
-- #2 hash JOIN verschachtelt
SELECT *
 FROM sales
 WHERE sales.hash_diff = (SELECT hash_diff FROM sales_row);

Elapsed: 00:00:00.05
```

```
-- #3 JOIN alle Attribute
SELECT DISTINCT *
 FROM sales_row, sales
 WHERE sales_row.prod_id = sales.prod_id
  AND sales_row.cust_id = sales.cust_id
  AND sales_row.time_id = sales.time_id
  AND sales_row.channel_id = sales.channel_id
  AND sales_row.quantity_sold = sales.quantity_sold;
Elapsed: 00:00:00.16
```

```
-- #4 hash JOIN entschachtelt
SELECT DISTINCT *
 FROM sales_row, sales
 WHERE sales_row.hash_diff = sales.hash_diff;



Elapsed: 00:00:00.10
```

Listing 1: Unterschiedliche Antwortzeiten vier äquivalenter Abfragen

somit den Energieverbrauch auswirken *(siehe Listing 1)*. Je nach Formulierung liefern die Abfragen das gleiche Ergebnis nach 0.05 s, 0.10 s, 0.16 s und 1.42 s, sind also um den Faktor 2, 3 oder 30 schneller beziehungsweise langsamer!

Man kann sich für jede SQL-Abfrage den Ausführungsplan anzeigen. Dies geht sehr einfach im SQL Developer mit dem Klick auf die entsprechende Schaltfläche beziehungsweise F10 im SQL Worksheet *(siehe Abbildung 3)*.

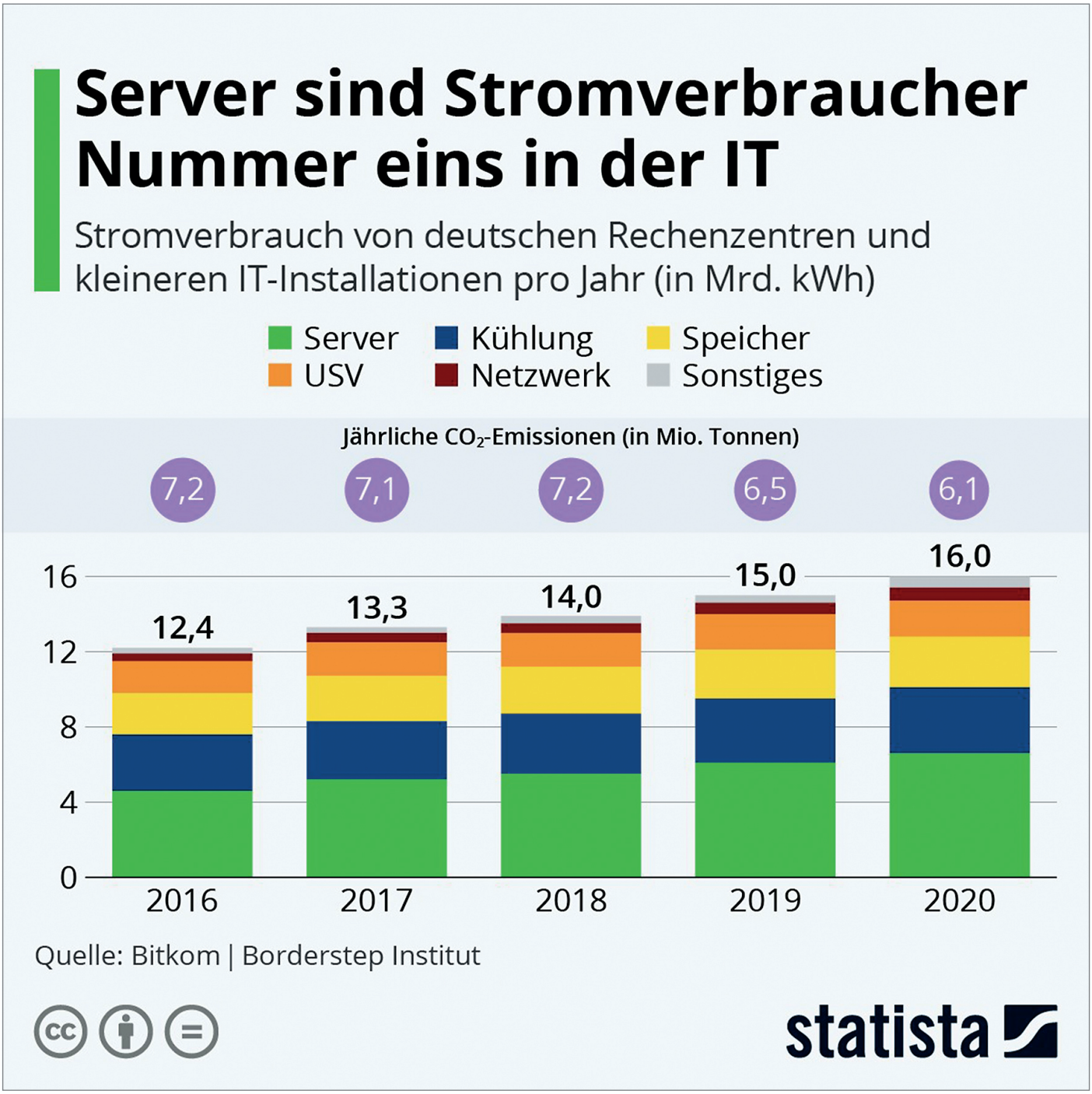


Abbildung 2: Serverstromverbrauch in DE (© statista | Bitkom |Borderstep Institut)

Die im Ausführungsplan enthaltenen Kosten entsprechen der Abfragedauer, das heißt dem Energieverbrauch (elektrische Leistungsaufnahme über die Zeitdauer der Abfrage).

Eine produktive Datenbank verarbeitet gleichzeitig Tausende Abfragen und jede dieser Abfragen könnte ein potenzieller „Performance- beziehungsweise Klimakiller" sein. Die Grundlage für effizientes und professionelles Arbeiten in der Datenbank bilden Monitoring-, Diagnose- und Tuning-Werkzeuge wie beispielsweise das weit verbreitete kostenpflichtige Diagnostics und Tuning Pack sowie das AWR (Automatic Workload Repository) mit ADDM (Automatic Database Diagnostic Monitoring), ASH (Active Session History) und SQL Tuning Advisor. Mithilfe dieser Werkzeuge können ineffiziente Abfragen ermittelt und durch zusätzliche Indexierung, Datenkomprimierung oder Spalten-basierte Speicherung optimiert werden. Auch hier gilt: Was für die Datenbank-Performance gut ist, ist auch für gut für die Umwelt.

Datenbank-Komprimierung spart Platz, I/O, Kühlung und Hardware

Mit der Datenbank-Komprimierung (Oracle Basic bzw. Advanced Compression) können bestehende Datenmengen mit Faktor 2 bis 4 reduziert werden. Ein positiver Nebeneffekt ist, dass die betroffenen Abfragen um den gleichen Faktor schneller werden *(siehe Listing 2)*, da doppelt bis vierfach kleinere Datenmengen verarbeitet werden. Das I/O wird drastisch reduziert. Weniger Speicher-

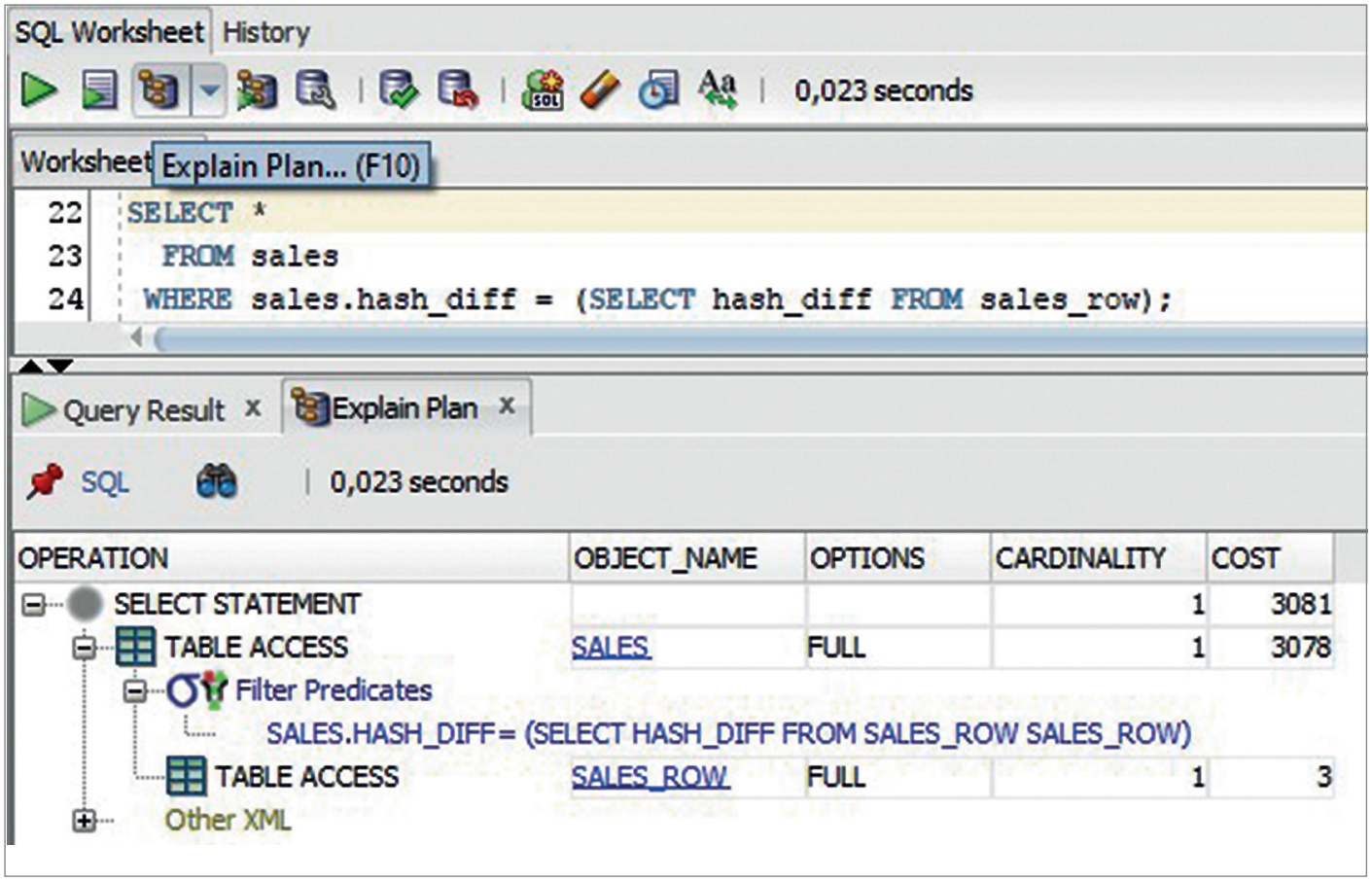


Abbildung 3: Ausführungsplan einer Datenbank-Abfrage in SQL Developer (Quelle: Peter Hoffmann und Bojan Milijas)

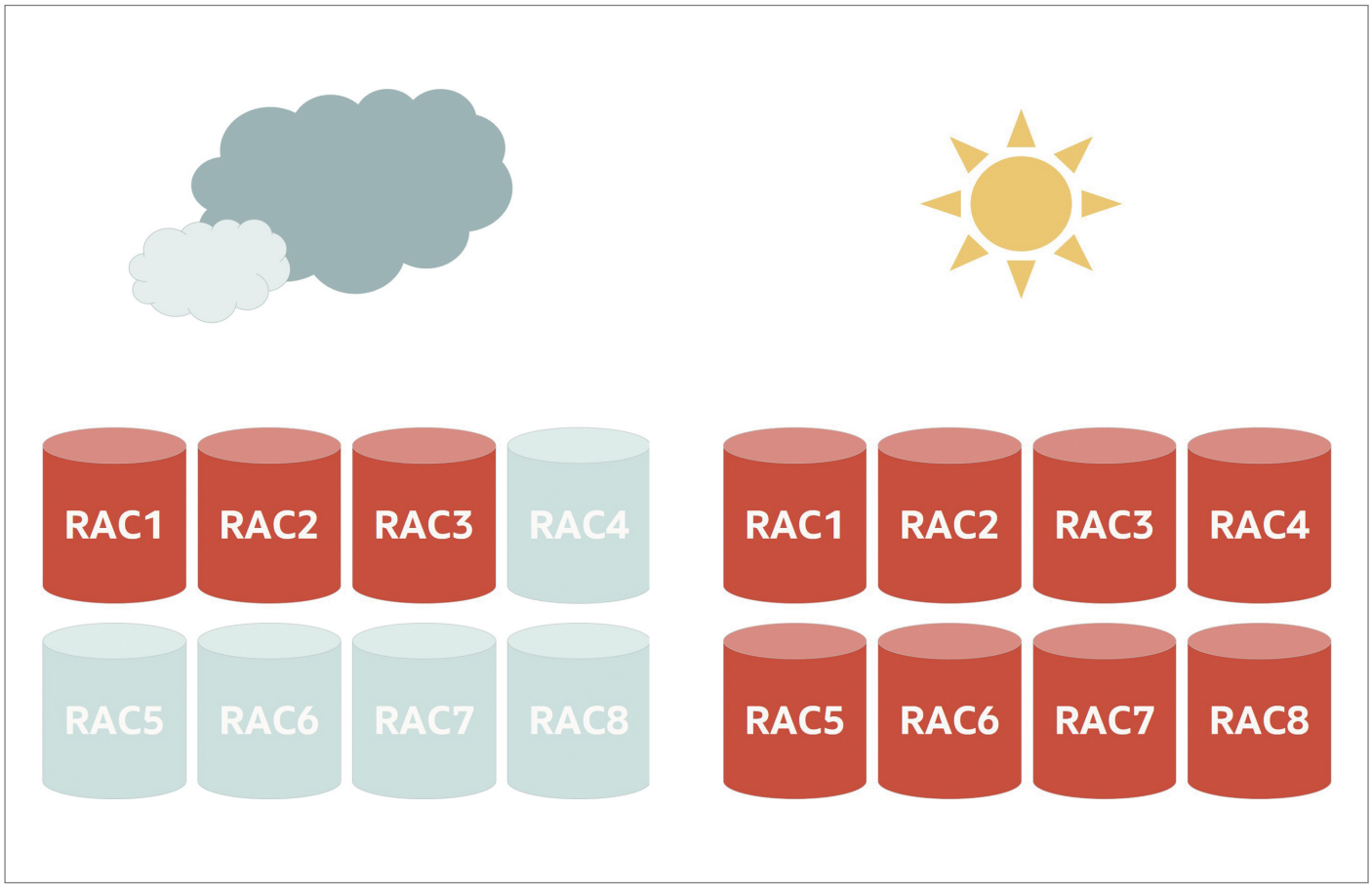


Abbildung 4: Real Application Clusters als „atmende ökologische Ressource“ (Quelle: Peter Hoffmann und Bojan Milijas)

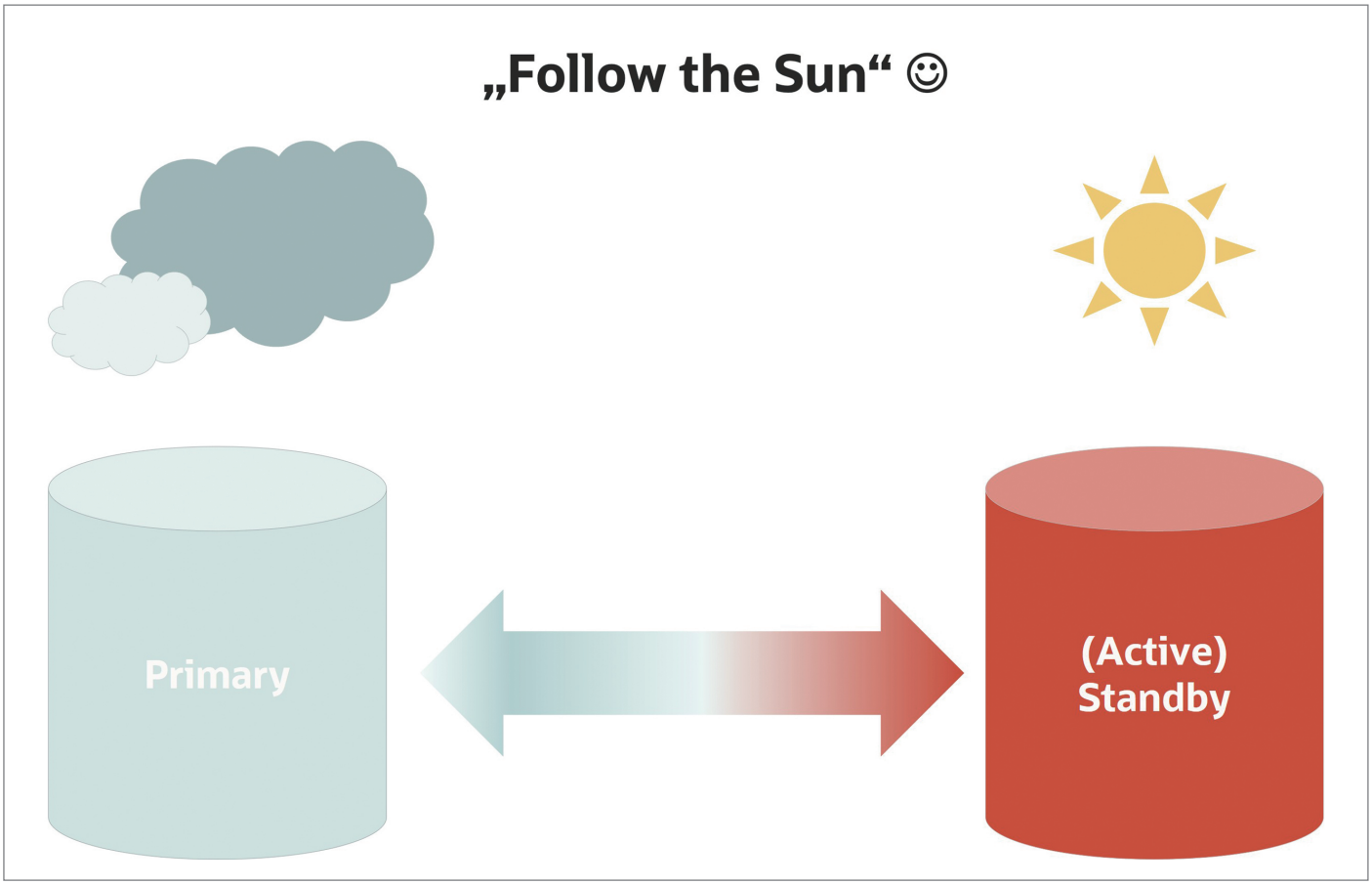


Abbildung 5: Mit (Active) Data Guard „der Sonne folgen“ und Ökobilanz verbessern (Quelle: Peter Hoffmann und Bojan Milijas)

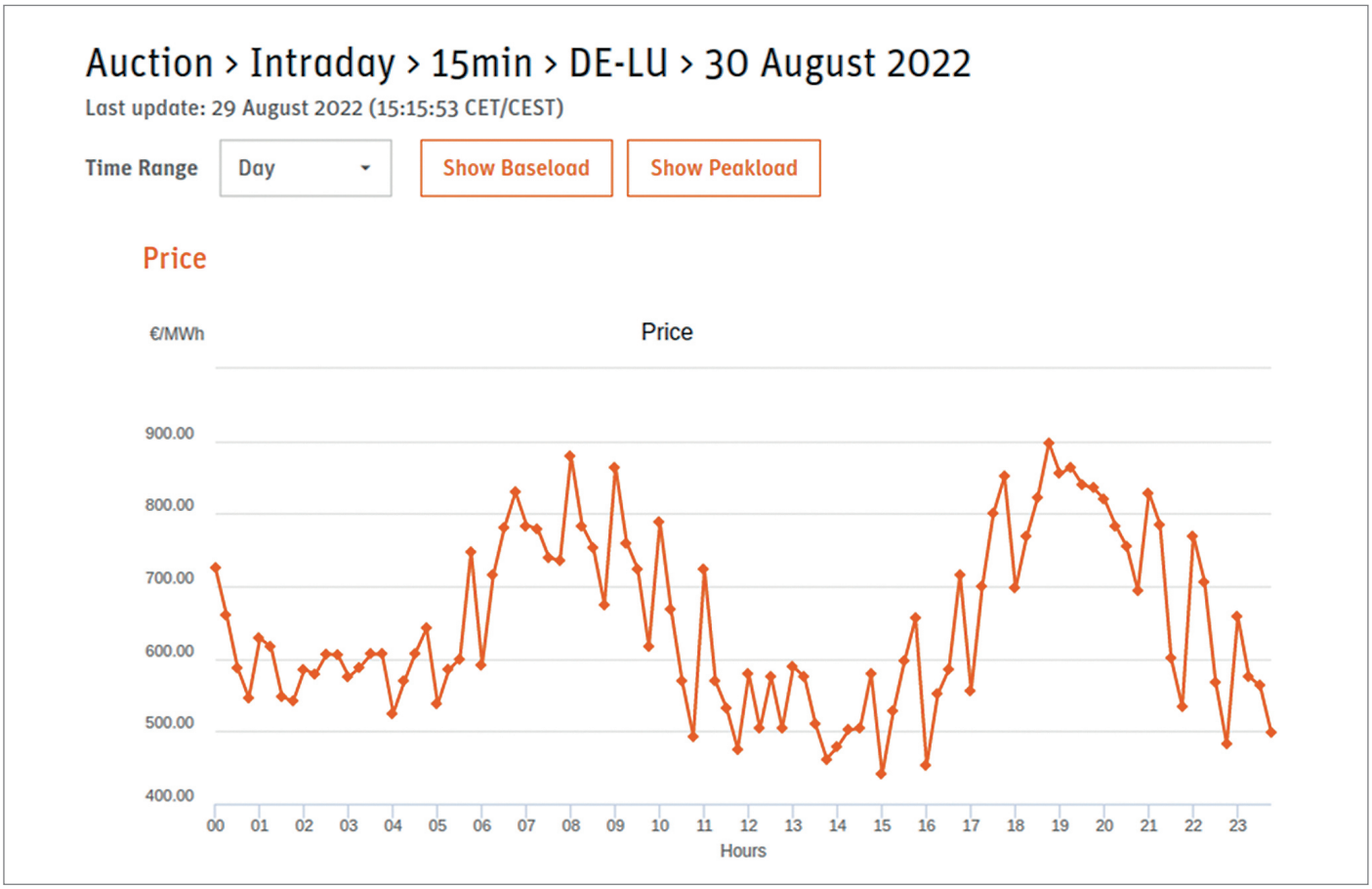


Abbildung 6: Untertägige Strompreisschwankungen (© epexspot.com)

```
SQL> ALTER TABLE sales4doag MOVE NOCOMPRESS;

Table SALES4DOAG altered.

Elapsed: 00:02:12.47
SQL> select segment_name, segment_type,
sum(bytes/1024/1024/1024) GB
  2  from dba_segments
  3  where segment_name='SALES4DOAG'
  4 group by segment_name,segment_type;

SEGMENT_NAME SEGMENT_TYPE GB
------------ ------------ ------
SALES4DOAG   TABLE        2

Elapsed: 00:00:00.01
SQL> SELECT count(*) FROM sales4doag;

  COUNT(*)
----------
  58805952

Elapsed: 00:00:25.16
```

```
SQL> ALTER TABLE sales4doag MOVE COMPRESS;

Table SALES4DOAG altered.

Elapsed: 00:01:03.09
SQL> select segment_name, segment_type,
sum(bytes/1024/1024/1024) GB
  2  from dba_segments
  3  where segment_name='SALES4DOAG'
  4 group by segment_name,segment_type;

SEGMENT_NAME SEGMENT_TYPE GB
------------ ------------ -------
SALES4DOAG   TABLE        ,74218

Elapsed: 00:00:00.01
SQL> SELECT count(*) FROM sales4doag;

  COUNT(*)
----------
  58805952

Elapsed: 00:00:09.74
```

Listing 2: DB Compression: Speicherplatz und Antwortzeit um Faktor 3 reduzieren

platz bedeutet weniger Server und weniger Kühlung. Somit ist die Oracle-Datenbank-Komprimierung wahrscheinlich die wirkungsvollste ökologische Maßnahme, die sofort messbare Ergebnisse zeigt, zumal Basic Compression ein kostenloses Enterprise Edition Feature ist und die kostenpflichtige Advanced Compression außer ROI-Betrachtung keine zusätzlichen Konfigurations- oder Installationsarbeiten erfordert.

In-Memory verkürzt die Abfragedauer (Energieverbrauch) um ein Vielfaches

Hinzu kommt, dass bei der Einschaltung der spaltenbasierten Verarbeitung die umständliche Indexpflege durch den DBA komplett entfällt. Die Datenbank übernimmt das Bitmap-konforme Speichern und Referenzieren der sich wiederholenden Spaltenwerte. In-Memory ist eine kostenpflichtige Zusatzoption. Daher empfehlen wir eine vorherige ROI-Betrachtung inklusive Zeitgewinn (Stromeinsparung) bei der Ausführung ressourcenintensiver Workloads *(siehe Listing 3)*.

Skalierbarkeit und Hochverfügbarkeit in den Diensten des Umweltschutzes

Mit Oracle Real Application Clusters (RAC) kann eine einzelne Datenbank auf mehreren Servern parallel ausgeführt werden, um maximale Verfügbarkeit und horizontale Skalierbarkeit zu erreichen. Mit Rolling Patches und Rolling Upgrades können geplante und wartungsbedingte Ausfälle vermieden werden *[5]*.

Der umweltbewusste DBA könnte parallellaufende RAC-Datenbankinstanzen als „atmende Ressourcen" einsetzen, die Performance bei ungünstigen Bedingungen „drosseln" und bei viel Wind und Sonne wieder hochfahren *(siehe Abbildung 4)*.

Viele Unternehmen haben bereits die Anforderung, wegen Disaster-Recovery hochproduktive Datenbanken redundant an geografisch verteilten Standorten zu betreiben beziehungsweise vorzuhalten, und setzen seit Jahrzehnten den (Active) Data Guard von Oracle ein. Ähnlich wie bei RAC könnte man je nach Wetterlage und Uhrzeit, das heißt Verfügbarkeit sauberer Energie (Wind, Solar, Kühlung), am jeweiligen Standort bei der Abarbeitung rechenintensiver Workloads zwischen Primary- und (Active)-Stand-By-Datenbanken wechseln und nicht nur notgedrungen im Katastrophen-Fall. Die flexible Verwendung sauberer Energie würde die Ökobilanz des Unternehmens verbessern, bei der Nutzung bereits bestehender redundanter Ressourcen *(siehe Abbildung 5)*.

Ökologisches Datenbank-Scheduling

Die beste Möglichkeit, den Betrieb von Informationstechnologie ökologisch zu gestalten, ist, die Verarbeitungsmengen zu reduzieren und die Verarbeitung effizienter zu gestalten. Die nächstbeste Möglichkeit ist, die Verarbeitung auf Zeiten zu verlagern, in denen externe Ressourcen, wie etwa Strom oder Kühlung, verfügbar beziehungsweise nicht knapp sind. Beide Ansätze wirken sich zusätzlich positiv auf die Kosten für den IT-Betrieb aus.

Das klassische Scheduling basiert auf der Verteilung von zur Verfügung stehenden Betriebssystem-Ressourcen, wie zum Beispiel der CPU-Zeit. Hier bietet sich ein

```
SQL> SELECT count(*) FROM sales_classic;
  COUNT(*)
----------
  58805952
Elapsed: 00:00:24.98                          Elapsed: 00:00:00.02
SQL> SELECT channel_id, count(1) FROM sales_classic GROUP BY channel_id;
CHANNEL_ID   COUNT(1)
---------- ----------
         2   16513600
         4    7578624
         3   34580992
         9     132736
Elapsed: 00:00:25.83                          Elapsed: 00:00:00.31
```

Listing 3: Zigfache Verkürzung der Abfragen mit der In-Memory Verarbeitung

```
https://monitoringapi.mini-pv.de/site/1/power?startTime=2022-05-5%2011:00:00&endTime=2022-05- 05%20
13:00:00&api_key=L4QLVQ1LOKCQX2193VSEICXW61NP6B1O

Method: GET Formats: JSON, XML and CSV
```

Listing 4: API zum Abfragen der Kennzahlen

```
{ "power":{ "timeUnit":"QUARTER_OF_AN_HOUR", "unit":"W", "values":[{ "date":"2022-05-05 11:00:00", "val-
ue":7987.03 }, …
```

Listing 5: Ermittlung der aktuellen Leistung des Wechselrichters mittels API

```
curl -i -X POST -u username:password
-d @request_body.json
-H "Content-Type:application/json" https://rest_server_url/ords/_/db-api/stable/database/pdbs/
{
  "method": "CREATE",
  "pdb_name": "pdb_sample",
  "adminName": "pdbadmin",
  "adminPwd": "W3lc0m31",
  "fileNameConversions": "('/disk1/oracle/dbs/pdbseed/','/disk1/oracle/dbs/pdb_sample/')",
  "unlimitedStorage": true,
  "reuseTempFile": true,
  "totalSize": "UNLIMITED",
  "tempSize": "UNLIMITED"
}
```

Listing 6: Erzeugen einer neuen Datenbankinstanz mit einfachem REST-Kommando

guter Ansatz, um zusätzliche ökologisch sinnvolle Kriterien in den Scheduling-Prozess einfließen zu lassen. Die folgenden Grundlagen helfen uns dabei:

1. Verarbeitungsprozesse müssen priorisiert werden.
2. Die Verfügbarkeit von externen Ressourcen muss ermittelt werden.
3. Datenbank- und Betriebssystemfunktionen (APIs) müssen genutzt werden, um das Scheduling zu ermöglichen.

Priorisierung von Verarbeitungsprozessen

Zunächst hilft es, die zu verarbeitenden Lasten in zwei Kategorien einzuteilen.

Da gibt es zum einen die Aufgaben, die eine bestimmte Mindestantwortzeit voraussetzen, zum Beispiel „Online Transaction Processing" (OLTP). Hier muss festgelegt werden, welche Mindestleistung zur Verfügung stehen muss, um die Antwortzeit zu garantieren. Diese Mindestleitung kann auch „Baseline" genannt werden. Bei Leistungsanforderungen des Systems über die „Baseline" hinaus kann dann ökologisch skaliert oder *„scheduled"* werden.

Bei der zweiten Kategorie handelt es sich um Aufgaben, deren Ergebnisse nicht zeitnah zur Verfügung stehen müssen und eine größere Flexibilität bei der Verarbeitung erlauben. In dieser Kategorie finden sich analytische Aufgaben oder Aufgaben, die in Stapeln verarbeitet werden, auch „Batch-Processing" genannt.

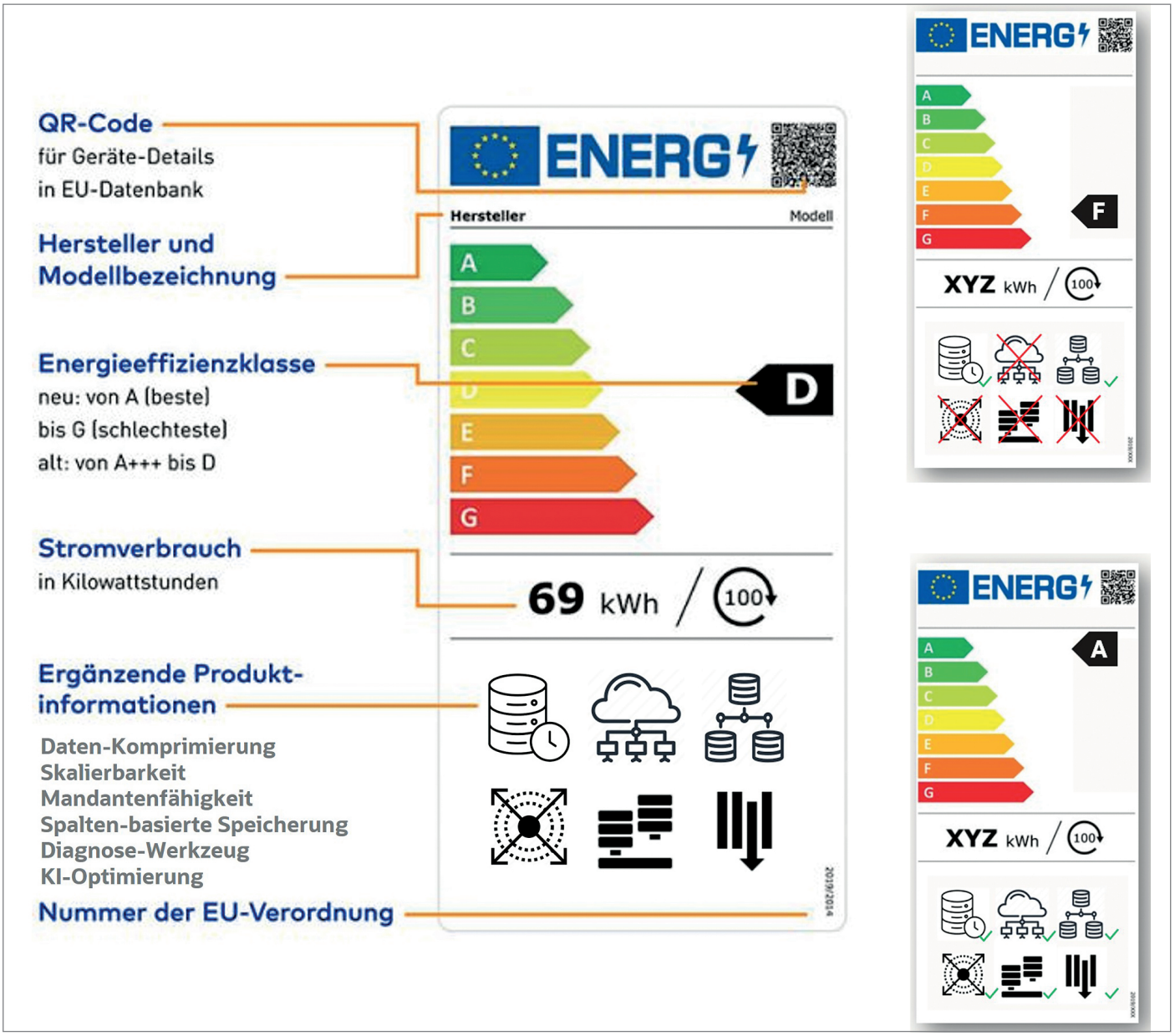


Abbildung 7: Öko-Label für Datenbanken (Quelle: Peter Hoffmann und Bojan Milijas)

Mit diesen Voraussetzungen wird das Scheduling oder die Skalierung im Hinblick auf einen Endzeitpunkt, zu dem die Aufgabe erledigt sein muss, festgelegt.

Verfügbarkeit externer Ressourcen

Ein maßgebliches Beispiel für den Einfluss einer externen Ressource ist Strom. Die Strompreise verändern sich kontinuierlich durch die Schwankung von Angebot und Nachfrage. Das Angebot wird dabei heute zu einem Großteil durch erneuerbare Energien abgedeckt. Für eine ökologisch nachhaltige IT ist es daher wichtig, den Großteil der Verarbeitung in Zeiten zu verlagern, in denen das Stromangebot und der Anteil der erneuerbaren Energien hoch sind *(siehe Abbildung 6)*.

Damit diese Strategie aufgehen kann, müssen Energieversorger Tarife anbieten, die sowohl die Schwankungen im Angebot als auch den Energiemix berücksichtigen. Von den Energieversorgern müssen auch die Kennzahlen zur Steuerung der IT-Verarbeitung kommen.

Strom ist in diesem Zusammenhang nicht die einzige knappe Ressource. Andere Beispiele sind Kühlung, Arbeitskraft, Fläche oder Netzanbindung.

Datenbank-Funktionen für geplante Verarbeitung

Die Oracle-Datenbank stellt eine ganze Reihe von Funktionen für das Scheduling der Verarbeitung und die Skalierung der Datenbank zur Verfügung. Diese Funktionen sind über APIs steuerbar und lassen sich daher leicht in ein Gesamtkonzept zum ökologischen Computing einbinden. Hier einige Beispiele:

- Oracle Resource Management – Für die Datenbank können verschiedene Profile erstellt werden, zum Beispiel ein Profil, das eine hohe Verarbeitungskapazität zur Verfügung stellt, und eines für Zeiten, in denen nur wenige Ressourcen vorhanden sind.
- Agiles Deployment – Eine Datenbankinstanz wird bei Bedarf dynamisch erstellt und wieder gelöscht, wenn sie nicht mehr benötigt wird oder keine Ressourcen zur Verfügung stehen.

- Oracle Multitenant – Viele Datenbanken teilen sich eine Infrastruktur und das Scheduling der Ressourcen wird dynamisch gesteuert.
- Instance Caging – Nur ein Teil der vorhandenen Rechenkapazität wird genutzt.
- Global Compute – Die Compute-Ressourcen können in verschiedenen Teilen der Welt zur Verfügung gestellt werden und die Aufgaben werden je nach Bedarf verlagert.
- Dynamische Clustererweiterung – Die Bearbeitungskapazität der Datenbank wird dynamisch skaliert.
- Sharding – Durch Partitionierung der Daten können aktuelle Daten mit höherer Priorität und einem höheren Energiebedarf verarbeitet werden.

Pseudocode-Beispiel für eine Solarsteuerung

Wie das Zusammenspiel von Kennzahlen mit der dynamischen Steuerung von Datenbankkapazität aussehen kann, soll das folgende Beispiel verdeutlichen:

Zur Stromversorgung der Datenbank steht eine Solaranlage zur Verfügung. Die Kapazität der Anlage schwankt je nach Sonneneinstrahlung. Der Wechselrichter der Solaranlage verfügt über ein API zum Abfragen der Kennzahlen *(siehe Listing 4)*.

Über das API lässt sich die aktuelle Leistung des Wechselrichters ermitteln *(siehe Listing 5)*.

Mit dieser Information können wir entscheiden, ob genügend erneuerbare Energien zur Verfügung stehen, um die Datenbank-Verarbeitung zu starten. Das Erzeugen einer neuen Datenbankinstanz ist dann ein einfaches REST-Kommando *(siehe Listing 6)*.

Nachdem die Aufgabe der Datenbank beendet ist, kann sie wieder gelöscht werden.

Dieses kurze Beispiel gibt uns einen Einblick, wie eng in Zukunft Verarbeitungsprozesse mit dem Betrieb der Datenbank verzahnt sein können und welche Vorteile zum Schutz der Umwelt daraus entstehen können.

Datenbank-Effizienzklassen als Öko-Label für Datenbanken

Abschließend könnte man, ähnlich wie bei Waschmaschinen, Kühlschränken oder Gebäuden, je nach Produkteigenschaften und Vorhandensein bestimmter „Functions & Features“ einen Datenbank-Energieausweis ausstellen *(siehe Abbildung 7)*.

Eine überall (in jeder Cloud und OnP) einsetzbare, an mehreren Standorten verfügbare, nach oben und nach unten skalierbare, mandantenfähige Datenbank, die über Datenkomprimierung, Partitionierung (Sharding), Zeilen- und Spalten-basierte Speicherung, KI-unterstütze Anfrageoptimierung, professionelle Diagnose und Tuning-Werkzeuge sowie Ressourcen-Manager und Job-Scheduler verfügt, würde die Kriterien für die Effizienzklasse A erfüllen. Datenbanken ohne solche Eigenschaften würde man schließlich als klimaschädlicher einstufen.

Quellen

[1] *https://thenewstack.io/more-database-analytics-workloads-ran-on-kubernetes-in-2022*
[2] *https://de.statista.com/infografik/27846/stromverbrauch-von-deutschen-rechenzentren-und-kleineren-it-installationen-pro-jahr/*
[3] *https://atos.net/content/dam/global/documents/we-are/atos-oracle-exadata-services.pdf*
[4] *https://www.oeko.de/oekodoc/2318/2015-489-de.pdf Öko-Institut e.V. im Auftrag des Umweltbundesamtes, Berlin 2015*
[5] *https://www.oracle.com/de/database/real-application-clusters/*

Über die Autoren

Peter Hoffmann und Bojan Milijas sind als Oracle-Systemberater und Lehrbeauftragte für Informatik, Datenbanken und Datenmanagement an den Hochschulen für Angewandte Wissenschaften in Frankfurt und München tätig.

Peter Hoffmann
peter.hoffmann@oracle.com

Bojan Milijas
bojan.milijas@oracle.com


12
1
2
3
4
5
6
7
8
9
10
11
century

Quadratur des Kreises – Attribute Clustering, ein weithin unterschätztes Feature der Oracle-Datenbank – Teil 1

Randolf Eberle-Geist, Unabhängiger Berater

Auch in Zeiten von Cloud und skalierbaren Ressourcen kann das physische Design immer noch eine entscheidende Rolle spielen, wenn es um die Effizienz einer datenbankgestützten Applikation geht. Oracle hat mit der Version 12c ein weniger beachtetes Feature eingeführt, das es erlaubt, die physische Organisation der in sogenannten „Heap"-Tabellen gespeicherten Daten aktiv zu beeinflussen und damit die Effizienz und Performance bestimmter Zugriffswege auf die Daten zu optimieren, was sehr signifikante Auswirkungen auf unterschiedliche Aspekte haben kann – nicht nur die Performance von Abfragen, sondern unter anderem auch der eventuelle Kompressionsfaktor der Daten kann positiv beeinflusst werden.

Um was geht es beim „Attribute Clustering"?

Standardmäßig verwendet Oracle bei Tabellenobjekten den Typ „Heap" – wenn nicht anders angegeben. „Heap"-Tabellen heißen so, da sie keiner vorgegebenen Organisationsform folgen – die Daten werden als unorganisierter „Haufen" verwaltet – es gilt die einfache Regel, dass dort, wo die Datenbank Platz findet in der Tabelle, neu einzufügende Zeilen abgelegt werden. Das macht das Schreiben von Daten extrem effizient, da nur eine grundlegende Freiplatzverwaltung benötigt wird, und erlaubt sogar Spezialformen beim Schreiben von neuen Datensätzen wie den sogenannten „APPEND"-Modus, bei dem eine Session exklusiv neue Datensätze oberhalb der aktuellen Obergrenze („HighWaterMark" HWM) anlegt – was das Schreiben noch effizienter machen kann: Da die Daten nicht in bereits existierende Blöcke geschrieben werden, kann sich die Datenbank sowohl den Zugriff auf die Freiplatzverwaltungsinformation dieser Blöcke sparen als auch die sonst notwendige Veränderungsprotokollierung im Undo-Tablespace zum Rückgängigmachen („Undo"-Information) – andere Sessions sehen die neu geschriebenen Daten vor einem COMMIT einfach noch nicht, da ja oberhalb der aktuellen HWM platziert, benötigen also auch keine „Undo"-Information, um Veränderungen an existierenden Blöcken aus Lesekonsistenz-Gründen gegebenenfalls rückgängig zu machen. Das eventuelle Rückgängigmachen der Transaktion – also der neu geschriebenen Zeilen oberhalb der HWM – ist auch trivial; der belegte Platz oberhalb der aktuellen HWM wird einfach freigegeben und nicht wie im COMMIT-Fall die HWM entsprechend angepasst, also auch dafür wird keine „Undo"-Information benötigt. Selbst das Pflegen eventuell bereits existierender Indizes auf der Tabelle kann im „APPEND"- Modus anders gehandhabt werden – anstatt für jede erzeugte Zeile einzeln die Indizes zu aktualisieren, kann dies auch in einer Art „Batch"-Operation durchgeführt werden, was meistens effizienter ist als eine zeilenweise Verarbeitung.

Dieser „APPEND"-Modus erlaubt es daher auch, die neu erzeugten Blöcke oberhalb der HWM speziell zu behandeln, zum Beispiel in Form von Datenkompression. Bis zur Version 12.2 konnte die Datenbank neu eingefügte Datensätze nur dann mit der „Hybrid Columnar Compression" (HCC – lizenztechnisch nur auf Exadata-basierten Umgebungen wie „Exadata On-Premise", „Exadata Cloud at Customer" oder auch „Autonomous Database" on Exadata verfügbar) versehen, wenn beim INSERT der APPEND-Modus verwendet wurde. Seit der Version 12.2 geht das mit der HCC-Kompression auch bei Batch/Array Inserts, die nicht den APPEND-Modus verwenden.

Auf den APPEND-Modus kommen wir später nochmal zurück – hier nur nochmal der Hinweis, dass aufgrund der gerade beschriebenen Vorgehensweise dabei immer nur eine Session gleichzeitig diesen Modus verwenden kann, also dabei ein exklusiver Lock auf die Tabelle entsteht und andere DML-Ope-

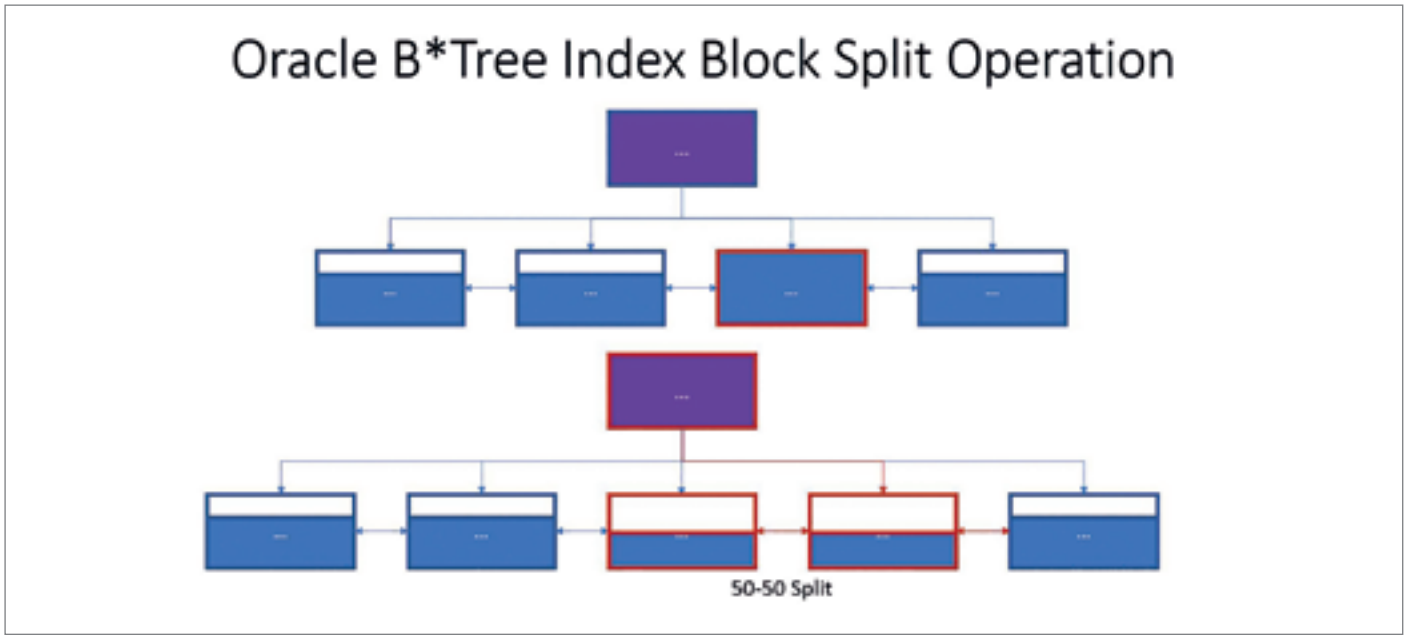


Abbildung 1: Eine Index-Block-Split-Operation ist aufwendig und kostspielig (Quelle: Randolf Eberle-Geist)

rationen auf den Abschluss dieser Operation warten müssen. Der Modus ist also nicht geeignet, wenn viele Sessions gleichzeitig die Tabelle verändern können sollen.

Im Umkehrschluss bedeutet diese Freiheit beim Ablegen von Daten in einer „Heap"-Tabelle allerdings auch, dass beim Lesen der Daten von keinerlei Voraussetzungen ausgegangen werden kann. Die gesuchten Daten können eben „irgendwo" in der Tabelle stehen und es gibt keine Annahme oder Regel, wo diese in der Tabelle zu finden sein könnten.

Daher gibt es ohne weitere Sekundärstrukturen bei „Heap"-Tabellen eben nur eine Zugriffsform, den „Full Table Scan", bei dem schlicht alle Blöcke unterhalb der „High Water Mark" der Tabelle durchsucht werden müssen.

Um bestimmte Zugriffsmuster zu beschleunigen, werden daher in vielen Fällen weitere Hilfsstrukturen über die „Heap"-Tabelle hinaus benötigt – typischerweise in Form einer Index-Struktur, also einer vorsortierten, redundanten Kopie (üblicherweise) eines Teils der Tabellendaten. Durch die andere physische Organisation der Kopie kann beim Zugriff von bestimmten Voraussetzungen ausgegangen werden, was je nach Zugriffsform bedeutet, dass eben nicht die gesamte Struktur durchsucht werden muss, sondern nur ein mehr oder weniger kleiner Teil, was dann weniger Arbeit beim lesenden Zugriff bedeuten kann. Das Pflegen einer solchen alternativen, regelbehafteten Organisationsform ist allerdings deutlich aufwendiger und komplexer als bei einer „Heap"-Struktur – die Daten können eben nicht einfach „irgendwo" abgelegt werden, sondern müssen laut der Regel logisch gesehen an einer bestimmten Stelle oder in einem bestimmten Bereich gespeichert werden. Hier kann es dann auch zu allerlei Seiteneffekten kommen – zum Beispiel dazu, dass an der Stelle, wo die Daten gemäß den Regeln logisch abgelegt werden müssen, nicht mehr ausreichend Platz zur Verfügung steht. Dies kann eine Kaskade von Datenreorganisationen nach sich führen, die sehr viel Aufwand verursachen und zusätzlich den gleichzeitigen Zugriff mehrerer Sessions beeinflussen und verlangsamen können *(siehe Abbildung 1)*.

Das gleichzeitige Schreiben von ähnlichen Daten, die also dann gemäß den Vorgaben an gleicher Stelle abgelegt werden müssen, kann bei solchen Strukturen auch zu Verlangsamungen führen, da eine Veränderung der Daten intern für einen kurzen Moment nur bei exklusivem Zugriff möglich ist – und andere Sessions müssen dann eben warten, bis sie den exklusiven Zugriff durchführen können, sollte eine andere Session gerade an gleicher Stelle tätig sein (Oracle instrumentiert das unter anderem als „Buffer Busy Waits"-Warteereignis).

Das heißt im Grunde, dass eine „Heap"-Tabelle optimal zum effizienten (und auch gleichzeitigen) Schreiben von Daten ist, aber für effiziente Lesezugriffe bei vielen Zugriffsmustern auf Sekundärstrukturen angewiesen ist, die wiederum entsprechende Nachteile beim Schreiben und Verändern von Daten mit sich bringen.

Bei anderen Datenbanken ist die „Heap"-Organisationsform für Tabellen nicht immer der Standard – bei Microsoft SQL Server zum Beispiel werden die Daten standardmäßig in einem „Clustered Index" gespeichert, wenn ein PRIMARY KEY auf einer Tabelle definiert wird – ein „Clustered Index" bedeutet jedoch, dass die Tabellendaten selbst in der vorsortierten Index-Struktur abgelegt werden und es keine separate „Heap"-Struktur gibt.

Alternative Organisationsformen – Vorteile und Nachteile

Daten einer Tabelle in dieser Form zu organisieren – auch bei Oracle ist diese Speicherform möglich in Form einer sogenannten „Index Organized Table" (IOT). Das kann dann große Vorteile bieten, wenn ein maßgeblicher Teil der Zugriffe auf die Daten gemäß der Vorsortierung passiert – dann nämlich stehen die gesuchten Daten alle zusammenhängend an der gleichen Stelle der Struktur

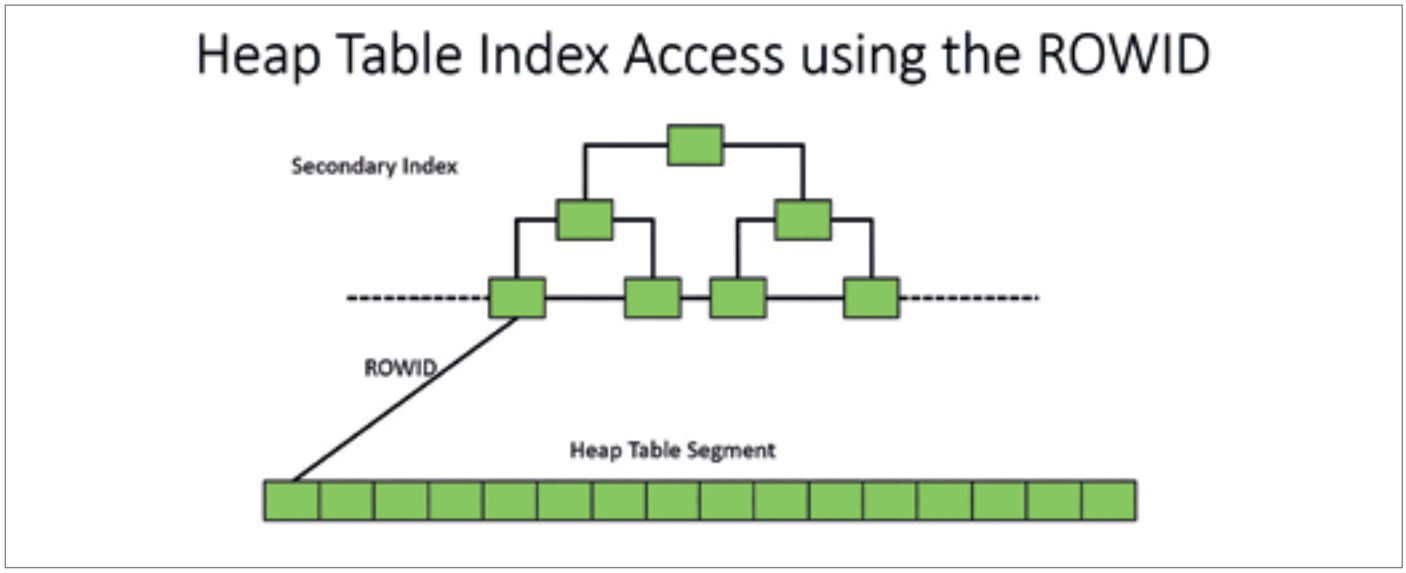


Abbildung 2: Auf Zeilen einer Heap-Tabelle kann per ROWID zugegriffen werden (Quelle: Randolf Eberle-Geist)

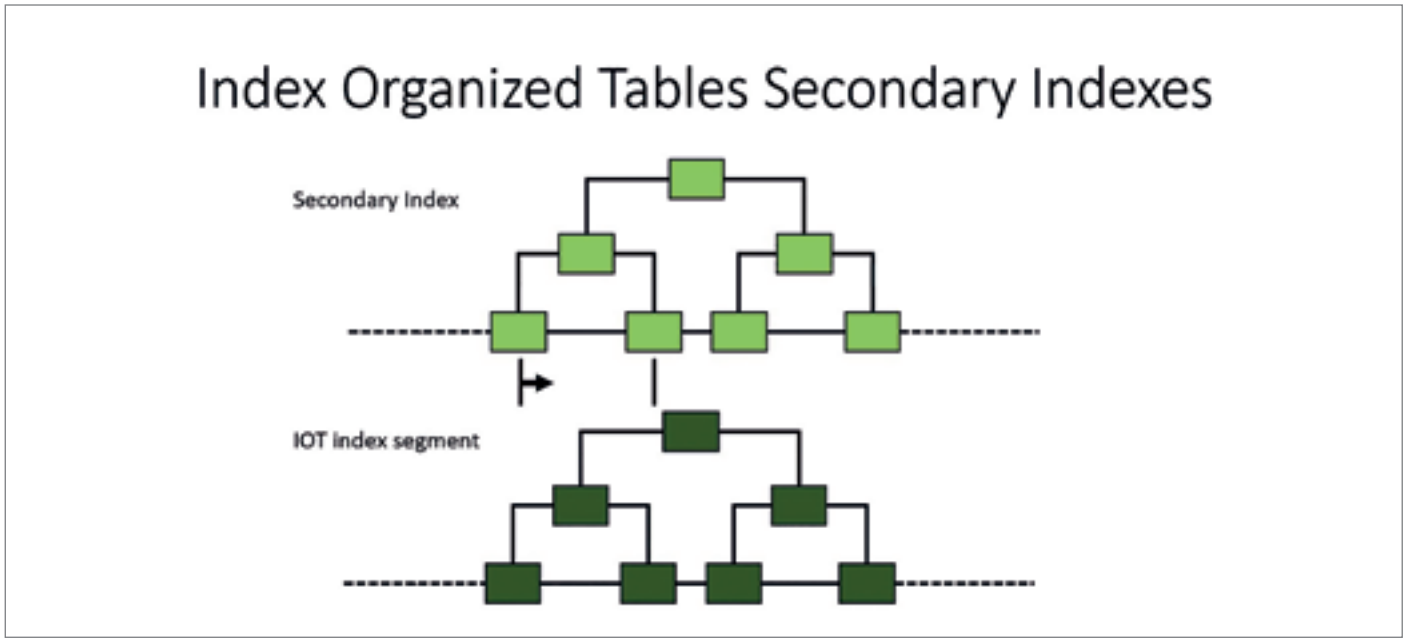


Abbildung 3: Ein Zugriff auf eine Index-Struktur kann keine ROWID verwenden (Quelle: Randolf Eberle-Geist)

und es muss nur genau so viel gelesen werden, wie die zusammenhängenden Daten an Platz benötigen. Bei Oracle hat diese „Index Organized Table" (IOT) jedoch eine signifikante Einschränkung: Die Vorsortierung ist nur gemäß dem „Primary Key" der Tabelle möglich, ein alternativer „Cluster Key" ungleich dem „Primary Key" ist nicht vorgesehen. „Clustered Index"-Objekte in Microsoft SQL Server zum Beispiel sind hier flexibler, was das angeht.

Nicht immer aber wird auf die Daten entsprechend dieser Vorsortierung zugegriffen, dann werden für einen effizienten Zugriff eventuell wieder weitere Sekundärobjekte benötigt. Und hier kann es eben zu größeren Nachteilen kommen im Vergleich zur „Heap"-Tabelle, denn diese Sekundärobjekte – also meistens Indizes mit anderer Sortierreihenfolge – müssen ja einen Zeiger auf die entsprechende Tabellenzeile beinhalten, um eventuell weitere Spalten der Tabelle für die identifizierte Zeile lesen zu können, die nicht in der Indexstruktur abgebildet sind. Bei „Heap"-Tabellen reicht hier ein sogenannter „Row Pointer", der die physische Adresse der Zeile in der Tabelle beschreibt – bei Oracle die sogenannte „ROWID". Über diesen Pointer kann die entsprechende Zeile der Tabelle direkt ohne weitere Zugriffe angesteuert werden *(siehe Abbildung 2)*.

Bei Index-basierten Strukturen („Clustered Index", „Index Organized Table") reicht dies allerdings nicht, da die Zeilen in einer solchen Struktur über die Zeit physisch gesehen an unterschiedlichen Stellen gespeichert sein können – eben unter anderem aus dem Grund, dass beim Einfügen neuer Daten eventuell nicht mehr ausreichend Platz an der gefragten Stelle ist und die Daten deswegen umorganisiert werden müssen („Index Block Split"-Operation). Insofern muss ein solcher Sekundärindex immer das Schlüsselkriterium für die primäre (Tabellen-)Index-

```
create table stock_history (
ticker_code   varchar2(32),
trade_date    date,
price_close   number(15,2),
trade_volume  number(10),
price_high    number(15,2),
price_low     number(15,2),
constraint stock_history_pk primary key (ticker_code, trade_date)
)
;
begin
  for i in 1..1000 loop
    insert into stock_history (
    ticker_code,
    trade_date,
    price_close,
    trade_volume,
    price_high,
    price_low)
    select
    ticker_code,
    trade_date,
    price_close,
    trade_volume,
    price_high,
    price_low
    from (
      select
            ticker_code
          , date '2000-01-01' + i - 1 as trade_date
          , i * company_code as price_close
          , i * company_code as trade_volume
          , i * company_code as price_high
          , i * company_code as price_low
      from (
              select /*+ cardinality(1000) */
                    'COMPANY_' || to_char(level - 1, 'FM000') as
ticker_code
                  , level - 1 as company_code
              from
                    dual
              connect by level <= 1000
            ) companys
    )
    ;
  end loop;
end;
/

commit;

exec dbms_stats.gather_table_stats(null, 'STOCK_HISTORY');
```

Listing 1: Erzeugung und Befüllung einer einfachen Beispieltabelle

Struktur beinhalten und ein Zugriff per Sekundärindex bedeutet für jede Zeile, die aus der Tabelle gelesen werden muss, dass die Index-Struktur der Tabelle auch durchsucht wird, um die Zeile zu lokalisieren. Das hat zwei wesentliche Effekte: Erstens benötigt der Sekundärindex potenziell deutlich mehr Platz pro Eintrag, je nachdem wie das Schlüssel-/Sortierkriterium der Tabelle aussieht (man stelle sich zum Beispiel mehrere VARCHAR-basierte Spalten als Schlüssel vor), zweitens ist der Zugriff auf eine Tabellenzeile per Sekundärindex deutlich aufwendiger, da nicht einfach per „Row Pointer" direkt auf die Zeile zugegriffen werden kann, sondern per Index-Suche *(siehe Abbildung 3)*.

Verwende ich also eine alternative Speicherform zur „Heap"-Tabelle und setze diese nicht effizient ein, benötige also zum Beispiel im Falle einer indexbasierten Organisationsform noch weitere Sekundärindizes für einen effizienten Zugriff, dann erkaufe ich mir möglicherweise deutliche Nachteile mit dieser Entscheidung gegenüber einer „Heap"-Tabelle.

Es gibt bei Oracle noch weitere praktische Gründe, die gegen eine Nutzung von alternativen Speicherformen wie IOTs oder Cluster sprechen können. IOTs bei Oracle können im Index-Segment nur eine begrenzte Breite von Informationen speichern, da es ein internes Limit dafür gibt, wie breit ein Eintrag in einer B*Tree- Indexstruktur sein kann. Spalten, die in einem IOT gespeichert werden sollen, aber nicht innerhalb dieses Limits bezüglich der Breite liegen, benötigen zwingend ein sogenanntes „Overflow"-Segment – dies ist eine zusätzliche „Heap"-Segment- Struktur, in der diese restlichen Spalten einer Zeile abgespeichert werden. Beim Zugriff auf solche Spalten muss dann dieses zusätzliche Segment angesprochen werden, was die Effizienz wieder je nach Zugriffsfrequenz und -art deutlich verringern kann. Das Feature kann natürlich auch gezielt zur vertikalen Partitionierung von Daten eingesetzt werden, bei der Spalten, die nicht so häufig verwendet werden, von den restlichen, häufiger verwendeten Spalten separiert werden, um das IOT-Segment möglichst kompakt zu halten – eine Möglichkeit, die es für „Heap"-Tabellen so direkt nicht gibt, auch wenn man über Objekt-Typen/Nested Tables so etwas auch bei „Heap"-Tabellen erreichen kann.

Außerdem gelten für viele Features und Funktionalitäten bei Oracle Einschränkungen, wenn keine „Heap"-Tabellen zum Einsatz kommen oder, anders herum ausgedrückt, viele – gerade neuere – Features nur im Zusammenhang mit „Heap"-Tabellen zur Verfügung stehen und– wenn überhaupt – nur mit einiger Verzögerung für alternative Speicherformen umgesetzt werden, viele auch gar nicht.

Warum gibt es dann diese alternativen Speicherformen, wenn diese durchaus mit einigen Nachteilen einhergehen können, insbesondere bei „unsachgemäßem" Einsatz? Weil sie eben bei bestimmten Zugriffsmustern signifikant effizienter sein können als eine „Heap"-Tabelle. Warum ist das so? Weil bei der „Heap"-Tabelle laut Definition eben keine direkte Beeinflussung auf die Art

```
select * from stock_history;
TICKER_CODE                    TRADE_DATE        PRICE_CLOSE TRADE_VOLUME PRICE_HIGH  PRICE_LOW
------------------------------ ----------------- ----------- ------------ ---------- ----------
COMPANY_000                    20000101 00:00:00           0            0          0          0
COMPANY_001                    20000101 00:00:00           1            1          1          1
COMPANY_002                    20000101 00:00:00           2            2          2          2
COMPANY_003                    20000101 00:00:00           3            3          3          3
COMPANY_004                    20000101 00:00:00           4            4          4          4
COMPANY_005                    20000101 00:00:00           5            5          5          5
.
.
.
COMPANY_995                    20000101 00:00:00         995          995        995        995
COMPANY_996                    20000101 00:00:00         996          996        996        996
COMPANY_997                    20000101 00:00:00         997          997        997        997
COMPANY_998                    20000101 00:00:00         998          998        998        998
COMPANY_999                    20000101 00:00:00         999          999        999        999
COMPANY_000                    20000102 00:00:00           0            0          0          0
COMPANY_001                    20000102 00:00:00           2            2          2          2
COMPANY_002                    20000102 00:00:00           4            4          4          4
COMPANY_003                    20000102 00:00:00           6            6          6          6
COMPANY_004                    20000102 00:00:00           8            8          8          8
COMPANY_005                    20000102 00:00:00          10           10         10         10
.
.
.
```

Listing 2: Erzeugung und Befüllung einer einfachen Beispieltabelle

```
set autotrace traceonly

select * from stock_history where ticker_code = 'COMPANY_100' and trade_date between date '2000-01-01' and
date '2001-12-31' order by trade_date;
731 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2261377770

-------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                  |   732 | 29280 |   740   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| STOCK_HISTORY    |   732 | 29280 |   740   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | STOCK_HISTORY_PK |   732 |       |     8   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("TICKER_CODE"='COMPANY_100' AND "TRADE_DATE">=TO_DATE(' 2000-01-01
              00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "TRADE_DATE"<=TO_DATE(' 2001-12-31 00:00:00',
              'syyyy-mm-dd hh24:mi:ss'))

Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        741  consistent gets
          0  physical reads
          0  redo size
      30304  bytes sent via SQL*Net to client
        383  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        731  rows processed
```

Listing 3: AUTOTRACE-Ergebnis einer Abfrage über zwei Jahre eines Aktienkurses

und Weise möglich ist, wie die Daten in der Tabelle organisiert sind. Es ergibt sich bei einer „Heap"-Tabelle zwar häufig eine „natürliche" Organisation der Daten in dem Sinne, dass Daten, die zu einem ähnlichen Zeitpunkt erzeugt worden sind, meistens auch physisch nahe beisammen oder zusammenhängend in der „Heap"-Tabelle gespeichert sind, also in den gleichen Blöcken (oder auch Seiten, je nach Datenbanksystem). Das muss aber nicht unbedingt der Fall sein, je nachdem, wo eben laut Freiplatzverwaltung gerade Platz für die neuen Daten ist. Das bedeutet dann eben auch, dass Zugriffsmuster, die auf Daten zugreifen, die in der „Heap"-Tabelle nicht zusammenhängend gespeichert sind, auf deutlich mehr Blöcke/Seiten zugreifen müssen als im optimalen Fall notwendig – im schlechtesten Fall muss für jede zugegriffene Zeile ein jeweils anderer Block gelesen werden, was vor allem deutlich mehr physisches I/O verursachen und die Effizienz des Daten-Cache der Datenbank signifikant verschlechtern kann. Im schlechtesten Falle entstehen dabei Szenarien, bei denen ein Vielfaches an I/O notwendig ist im Vergleich zum Full Table Scan – bei diesem lese ich jeden Block genau einmal, bei einem solchen „Worst Case"-Zugriffsmuster jedoch so viele Blöcke wie Zeilen, auf die zugegriffen wird. Hat eine Tabelle also zum Beispiel im Durchschnitt 100 Zeilen pro Block und ich lese die gesamte Tabelle mittels eines solchen Zugriffsmusters, greife ich auf jeden Block 100-mal zu – erzeuge also um den Faktor 100 mehr I/O – meistens dann auch tatsächlich 100-mal mehr physisches I/O, wenn die Tabelle deutlich größer als der Daten-Cache der Datenbank ist. Dieser Effekt entsteht dadurch, dass die gelesenen Blöcke sich gegenseitig aus dem Cache verdrängen und beim nächsten Zugriff auf den gleichen Block dieser sich schon nicht mehr im Cache befindet und wieder einen physischen Lesezugriff benötigt, um in den Cache eingelesen zu werden.

Ein praktisches Beispiel

Anhand eines einfachen Beispiels soll dieser Effekt des unterschiedlichen Clusterings der Daten deutlich gemacht werden. Nehmen wir an, es sollen Aktienkurse in einer Tabelle abgespeichert werden – ein Stand pro Tag der Einfachheit halber. Verwende ich eine „Heap"-Tabelle dafür und ignoriere für den Augenblick weitere Effekte, die durch das Löschen/Verändern von Daten entstehen können, werden die Daten eines jeden Tages mit großer Wahrscheinlichkeit physisch zusammenhängend in der Tabelle abgelegt werden, da sie zur gleichen Zeit der Tabelle per Insert hinzugefügt werden. Das könnte man mit folgender Tabellendefinition und PL/SQL-Blocks in Oracle simulieren *(siehe Listing 1)*.

```
create table stock_history (
ticker_code   varchar2(32),
trade_date    date,
price_close   number(15,2),
trade_volume  number(10),
price_high    number(15,2),
price_low     number(15,2),
constraint stock_history_pk primary key (ticker_code, trade_date)
)
organization index
compress 1
;
```

Listing 4: Erzeugung der Beispieltabelle als „Index Organized Table" (IOT)

Damit erstelle ich genau eine Million Zeilen – eintausend Einträge für eintausend verschiedene Aktien jeweils für eintausend Tage. Schaue ich mir die Daten in der Tabelle an, sehe ich, dass die Einträge pro TRADE_DATE zusammenhängend in der Tabelle gespeichert sind – es kommen also erst alle TICKER_CODEs für ein TRADE_DATE, danach folgen die gleichen TICKER_CODEs nochmal für das folgende TRADE_DATE *(siehe Listing 2)*.

Wenn ich jetzt eine Abfrage auf dieser Tabelle ausführe, die die Kurse einer bestimmten Aktie für zwei Jahre ausgeben soll, bekomme ich folgendes Bild - alle Beispiele wurden auf der aktuellen Version 19c ausgeführt *(siehe Listing 3)*.

Es werden also 731 Zeilen per Primär-Index-Zugriff (TICKER_CODE, TRADE_DATE) selektiert. Das sieht auf den ersten Blick wie ein effizienter Zugriff aus – nur wenn wir genauer hinschauen, bemerken wir, dass Oracle ausgibt, dass für das Selektieren der 731 Zeilen 741 Blöcke besucht werden mussten („consistent gets" – logisches I/O in diesem Fall, also kein physisches I/O, um die Blöcke in den Cache zu lesen). Im Grunde also wurde für jede selektierte Zeile per Index auf einen anderen Block der Tabelle zugegriffen. Warum ist das so?

Wenn wir uns die Ausgabe von oben *(siehe Listing 2)* nochmal genau anschauen, dann wird klar, dass jeder der gesuchten Einträge für eine bestimmte Aktie jeweils eintausend Zeilen auseinander in der Tabelle gespeichert ist, da ja immer erst mal alle eintausend Einträge/Aktienkurse für ein bestimmtes Datum hintereinander folgen und erst danach die nächsten eintausend für den nächsten Tag. Dadurch muss diese Abfrage immer per Index eintausend Zeilen in der Tabelle weiter springen, was dazu führt, dass mit großer Wahrscheinlichkeit jede gesuchte Zeile in einem anderen Tabellenblock steht, auch wenn ein Block in Oracle heutzutage standardmäßig 8 KB groß ist und je nach Zeilenbreite Hunderte von Zeilen aufnehmen kann.

Wenn es sich nun bei dieser Art von Abfrage um eine für meine Applikation kritische handeln würde, die sehr häufig auf einer entsprechend großen Datenmenge ausgeführt würde, wäre es natürlich toll, wenn ich die Effizienz erhöhen könnte. Denn so, wie es derzeit aussieht, lese ich für jede Zeile 8 KB an Daten in den Cache und damit eine große Menge an Zeilen, die mich in diesem Moment überhaupt nicht interessieren. Das ist sehr ineffizient, denn es erhöht die Wahrscheinlichkeit, dass die vielen verschiedenen Blöcke nicht im Cache sind und vom Storage gelesen werden müssen, was um Faktoren langsamer ist, als die Daten direkt aus dem Cache/Arbeitsspeicher zu lesen – außerdem „verschmutze" ich den Cache mit vielen unnützen Daten für diese Abfrage und mindere damit indirekt auch die Chan-

```
set autotrace traceonly

select * from stock_history where ticker_code = 'COMPANY_100' and trade_date between date '2000-01-01' and
date '2001-12-31' order by trade_date;
731 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2271383356

-------------------------------------------------------------------------------------------
| Id  | Operation         | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                  |   732 | 29280 |     6   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN| STOCK_HISTORY_PK |   732 | 29280 |     6   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("TICKER_CODE"='COMPANY_100' AND "TRADE_DATE">=TO_DATE('
              2000-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "TRADE_DATE"<=TO_DATE('
              2001-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          7  consistent gets
          0  physical reads
          0  redo size
      29376  bytes sent via SQL*Net to client
        383  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        731  rows processed
```

Listing 5: AUTOTRACE-Ergebnis der Abfrage über zwei Jahre eines Aktienkurses auf der IOT

cen für andere Abfragen, dass sie Daten im Cache halten können.

Um also diese Art der Abfrage effizienter gestalten zu können, müsste ich die Daten in der Tabelle nicht nach TRADE_DATE hintereinander abspeichern, sondern zuerst alle Einträge für eine Aktie hintereinander speichern – dann würden die Zeilen, die ich für eine bestimmte Aktie suche, alle zusammenhängend in wenigen Blöcken der Tabelle gespeichert stehen und ich müsste nicht für jede Zeile in einen anderen Block springen.

Um das zu erreichen, gibt es in relationalen Datenbanken unterschiedliche Möglichkeiten. Ich könnte zum Beispiel versuchen, einen Index zu erzeugen, der alle benötigten Spalten/Ausdrücke im Indexsegment abspeichert – dann muss die Datenbank gar nicht mehr auf das Tabellensegment zugreifen (manchmal „Covering Index“ genannt). Das macht allerdings den Index sehr breit, speichert viele Daten redundant ab und ist nicht immer sinnvoll umsetzbar – wenn auch auf jeden Fall eine mögliche Option je nach Szenario.

Optimierung durch Einsatz einer „Index Organized Table“ (IOT)

Alternativ dazu könnte man die Tabelle direkt als Index-Struktur ablegen – in anderen Datenbanksystemen gibt es dazu einen „Clustered Index“, in Oracle heißt das äquivalente Konstrukt „Index Organized Table“ (IOT). Das hat jedoch einige Eigenschaften und Nachteile, die man kennen sollte, wenn man es zum Einsatz bringt – siehe Beschreibung oben. Für das konkrete Szenario STOCK_HISTORY und die beschriebene Abfrage aus *Listing 3* wäre es potenziell eine sehr effiziente Lösung, je nachdem, was sonst noch mit der Tabelle und den Daten gemacht wird.

Wenn man das machen wollte, würde die Tabelle wie in *Listing 4* dargestellt, erzeugt werden.

Was hat es mit der Klausel „organization index“ auf sich? Die Tabelle würde nun also nicht als „Heap“-Tabelle erzeugt werden, sondern es würde stattdessen eine Index-Struktur (intern wie ein B*Tree-Index organisiert) angelegt werden, in der die Daten nach den Primärschlüssel-Kriterien sortiert ab-

gelegt werden. IOTs benötigen also in Oracle immer eine Primary Key Definition, ansonsten können sie nicht erzeugt werden. Durch die Definition TICKER_CODE, TRADE_DATE würden die Daten also primär nach dem Namen der Aktie sortiert werden und innerhalb einer Aktie nach TRADE_DATE. Ich hätte damit die Daten perfekt für die Abfrage nach den unterschiedlichen Aktienkursen für eine einzelne Aktie abgelegt, was sich auch in der Ausgabe der gleichen Abfrage auf einer entsprechend erzeugten und befüllten IOT widerspiegelt *(siehe Listing 5)*.

Schaue ich mir die Ausgabe genauer an, fallen mehrere Punkte auf: Erstens gibt es gar keinen Tabellenzugriff mehr – kein TABLE ACCESS BY ROWID im Ausführungsplan wie in der vorherigen Ausgabe –, denn es gibt eben nur noch die Index-Struktur der IOT, kein „Heap"-Tabellensegment mehr. Zweitens hat die gleiche Abfrage auf den gleichen Daten jetzt nur noch sieben Blöcke lesen müssen, um die 731 Zeilen zu selektieren, da die gesuchten Daten nun alle zusammenhängend in der Index-Struktur an der gleichen Stelle stehen – eben für eine Aktie alle TRADE_DATEs hintereinander gemäß der Primärschlüssel-Definition.

Das ist im Sinne der logischen I/Os um Faktor 100 effizienter als vorher. Führe ich also diese Art der Abfrage häufig aus, spare ich extrem viel Arbeit in der Datenbank ein. Außerdem habe ich jetzt maximal sieben Blöcke in den Cache einlesen müssen für die gleiche Datenmenge anstatt über 740 wie vorher. Die Wahrscheinlichkeit, dass diese im Cache gehalten werden können, ist also viel größer – und für andere Abfragen/Blöcke ist auch noch deutlich mehr Platz im Cache verfügbar, selbst wenn ich diese Art der Abfrage häufig ausführe.

Index-Kompression

Ich habe bei der Definition der IOT auch noch eine weitere Klausel angegeben – COMPRESS 1. Diese ist optional, kann aber auch sehr nützlich sein, denn Oracle kann eine B*Tree-Index-Struktur ohne signifikante Nachteile komprimieren. Das funktioniert deshalb so effizient, da Oracle einfach den führenden Teil des Index-Ausdrucks innerhalb eines Blocks deduplizieren kann, wenn sich dieser wiederholt. Da sich der TICKER_CODE für jedes TRADE_DATE wiederholt, also in dem konkreten Beispiel eintausend Mal der gleiche TICKER_CODE mit unterschiedlichen TRADE_DATEs abgelegt wird, spart es Platz im Index, wenn der TICKER_CODE stattdessen nur einmal abgespeichert wird und in den eigentlichen Daten nur ein Verweis auf den Eintrag. Diese Art der Komprimierung ist also intern von Oracle sehr einfach umzusetzen, benötigt nicht viel zusätzliche CPU-Zeit, spart potenziell, aber signifikant Platz im Index und ermöglicht so, deutlich mehr Einträge pro Index-Block abzuspeichern, wenn sich die führenden Werte im Index wiederholen – macht allerdings auch nur Sinn, wenn das der Fall ist. Sollten sich die Einträge nicht oder nur wenig wiederholen, würde diese Art der Kompression tatsächlich mehr Platz als ohne Kompression benötigen. Insofern kann man es leider nicht in jedem Fall anwenden.

Leider kommen IOTs in Oracle mit vielen Einschränkungen und Nachteilen. Je nachdem, wie auf die Daten zugegriffen werden soll, kann dies nicht immer sinnvoll als Primärschlüssel abgebildet werden. Wenn noch mit anderen Kriterien auf die Daten per Index zugegriffen werden soll, sind weitere Indizes auf einer IOT unter Umständen deutlich größer und ineffizienter als auf „Heap"-Tabellen. Falls die Tabelle viele oder breite Spalten hat, können nicht alle Spalten im Index-Segment abgelegt werden, da Oracle hier interne Beschränkungen hat. Außerdem unterstützen IOTs einige Features nicht, die für „Heap"-Tabellen zur Verfügung stehen – ausführlichere Erklärungen dazu gab es am Anfang des Artikels.

Über den Autor

Randolf Eberle-Geist ist seit über 25 Jahren als Freiberufler aktiv und hat sich auf Oracle-Datenbank-Performance-Themen spezialisiert. Im Bereich der SQL-Performance-Analyse und der Oracle-Optimizer-Technologie gehört er zu den Top-Experten weltweit und gibt sein Wissen regelmäßig auf Konferenzen sowie in Seminaren und Veröffentlichungen weiter. Er steht für kurzfristige Einsätze im Bereich Performance-Troubleshooting zur Verfügung, bietet aber auch eine Reihe von Workshops für Datenbank-Entwickler und DBAs an. Eine ausführliche Leistungsbeschreibung und eine Auswahl von Workshops finden Sie unter *http://www.oracle-performance.de*

Randolf Eberle-Geist
randolf.geist@oracle-performance.de

Welcome to Oracle Communities!

Jörg Sobottka, Robotron Schweiz

Nein, mit Oracle Communities sind in diesem Fall nicht die diversen Oracle User Groups weltweit gemeint, auch wenn die gegenseitige Hilfe in beiden Fällen im Vordergrund steht. Communities sind von Oracle eingerichtete Foren, die fast jeder passiv schon genutzt hat – und die trotzdem kaum einer kennt und aktiv nutzt. Dabei gibt es unzählige Gelegenheiten, sie zu nutzen. Willkommen sind alle – sofern einige Regeln beachtet werden.

Warum Oracle Communities?

Wer regelmäßig mit Oracle-Produkten arbeitet, verwendet genauso regelmäßig Suchmaschinen oder den Oracle Support. Viele Anfragen beim Oracle Support sind einerseits reine Verständnisprobleme, oftmals technische Fragen zur Architektur oder zur Umsetzung, andererseits bleiben Service Requests beim Support immer auch wieder über Wochen hängen, man dreht sich mit dem Support im Kreis oder das verwendete Produkt ist – noch schlimmer – überhaupt nicht mehr im Support und nein, man kann beim Auftreten eines Problems auch nicht immer ein „Update/Upgrade to newest version" durchführen. Jeder weiß selbst, dass eine Softwareversion veraltet oder nicht mehr supported ist, aber wenn es die Applikation(shersteller) oder andere Restriktionen nicht zulassen, muss mit dem ausgekommen werden, was da ist. Außerdem lief genau diese Kombination schon über Jahre gut, was soll also schon schiefgehen? Hätten die Beatles schon Computer gekannt, man könnte annehmen, die ein oder andere Liedzeile stamme aus dem Leben eines IT-Administrators. Ein Beispiel? *„Yesterday, all my troubles seemed so far away, now it looks as though they're here to stay"*.

Manchmal schießen spontan neue Ideen in den Kopf und dann wäre es schön zu wissen, was andere dazu denken. Vielleicht hat das schon jemand anderes ausprobiert oder es ist ein neues Feature, das für das Product Management interessant wäre? Wie oft denkt man sich, das würde ich gerne testen, aber vielleicht gibt es unter Umständen Komplikationen, die man vermeiden könnte, hätte man vorher gewusst, welche und wie?

Oracle fasst seine Communities so zusammen: „Oracle's communities provide access to experts, peers and practitioners that inspire innovation, celebrate successes and empower collaboration." *[1]*

Tatsächlich tummeln sich in den verschiedensten Foren unzählige Experten, unter anderem auch Oracle Aces, viele Consultants oder einfach Oracle-Praktiker aus aller Welt. Darüber hinaus findet man auch immer wieder Antworten oder Posts von Oracle Product Managern, zum Beispiel von Michael Ferrante (Oracle Forms), oder von Mitgliedern spezieller Teams, etwa dem RAC-Team. Alles in den Foren passiert auf freiwilliger Basis und es wird keine Customer-Support-Identification-Nummer benötigt. Einzige Voraussetzung zur Teilnahme ist ein vorhandener Oracle Technology Network Account und diesen dürfte jeder haben, der mit Oracle zu tun hat.

Der Einstieg

Wie kommt man an die Foren? Es gibt zwei offizielle Einstiegsmöglichkeiten, sowohl zum Lesen als auch zum Schreiben beziehungsweise zum Fragenstellen. Der leichte und übersichtliche Einstieg funktioniert über *https://community.oracle.com*. Hier kann zwischen verschiedenen Communities direkt ausgewählt werden, die beliebtesten sind dabei unter „Customers" sicher die „Cloud Customer Connect" und die „My Oracle Support Community" beziehungsweise neben den Customers die Groundbreaker „Developer"(s) Community. Mehr dazu im Verlauf des Artikels. Ein zweiter möglicher Einstieg ist über *https://support.oracle.com* möglich. Wer hier seine Suchbegriffe eingibt, kann bei den „KM Search Results" einen Haken beim „Community"-Tab setzen und bekommt entsprechende Forenbeiträge auch angezeigt. Der dazugehörige Bereich trägt den Titel „Community Search Results". Am Ende des Abschnitts gibt es einen Button „Ask in Community". Dieser führt in die „My Oracle Support Community" genauso wie der Button „Ask in Community" auf dem My Oracle Support Dashboard im Abschnitt „Technical Service Requests".

Wer nur sucht, findet übrigens auch in den einschlägigen Suchmaschinen Ergebnisse aus den Communities, da Oracle einen Teil der Threads für die Suchmaschinen bereitstellt. Die Ergebnisse erkennt man an der entsprechenden URL *(siehe Abbildung 1)*.

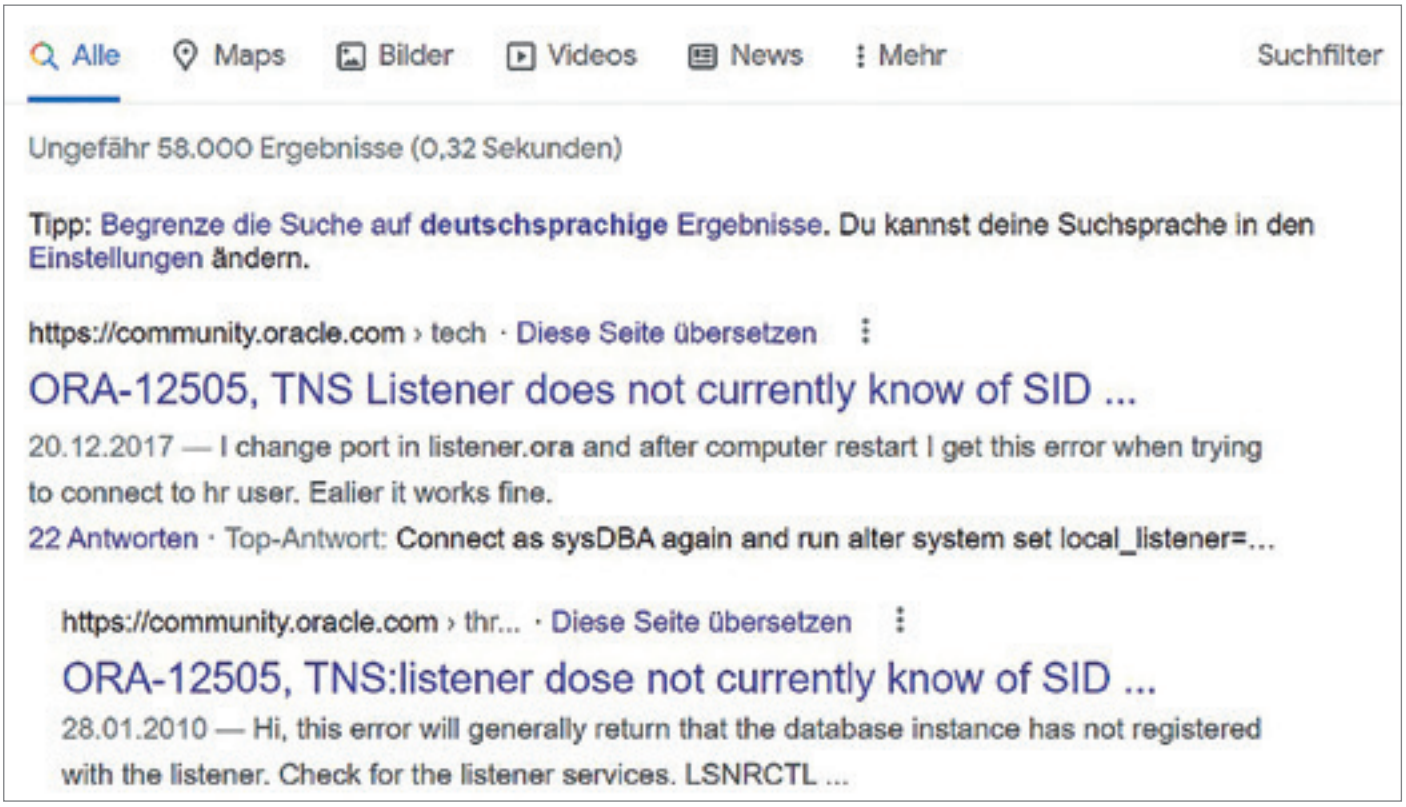


Abbildung 1: Suchergebnis aus Google (Quelle: Jörg Sobottka)

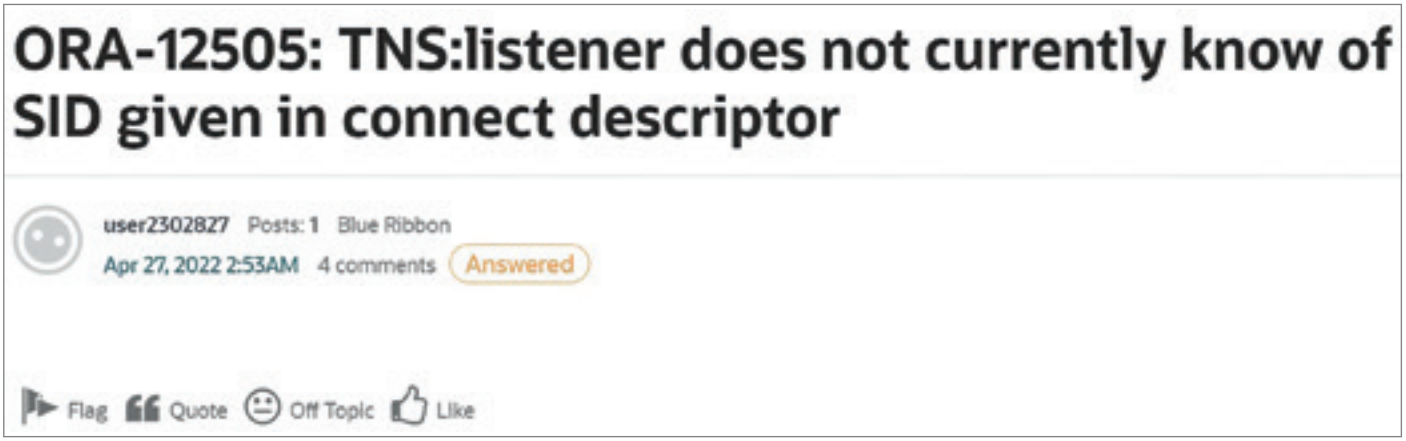


Abbildung 2: Post „ORA-12505“ mit Überschrift ohne Details [https://tinyurl.com/3j5xnyr9] (Quelle: Jörg Sobottka)

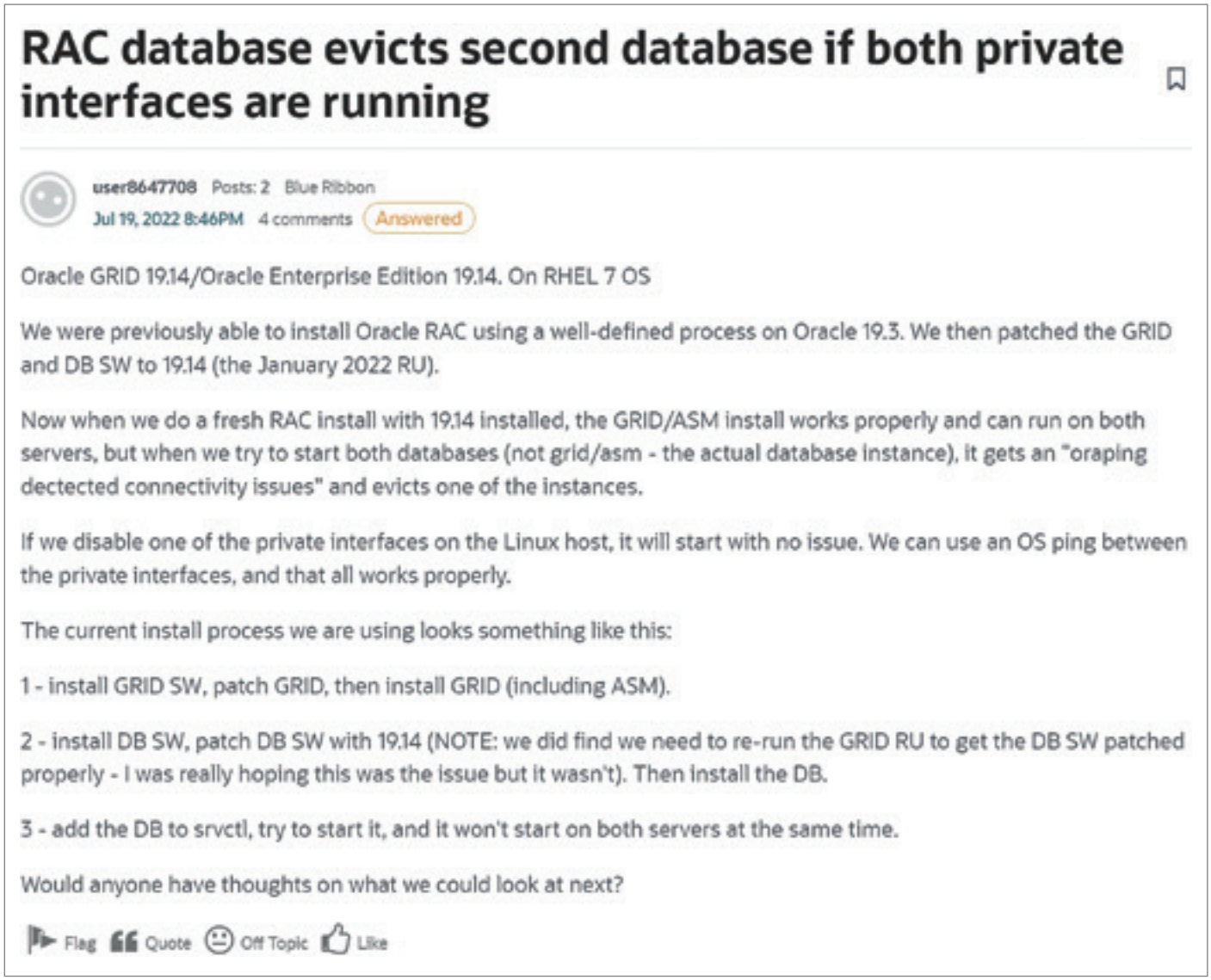
Fragen stellen und Hilfe bekommen

Wird über die Suchfunktion keine passende Antwort auf eine Fragestellung gefunden, so kann, darf und sollte eine eigene Anfrage in der Community gestellt werden. Das auch gerne parallel zu einem Service Request, es ist auch egal, woher die Hilfe letztendlich kommt – oder, wie es die Beatles ausgedrückt hätten: *„I need somebody (Help), not just anybody (Help), you know I need someone, help“*.

Wie bei allen anderen Foren, die zum Beispiel aus dem Open-Source-Umfeld bekannt sind, gibt es auch dabei einige Dinge zu beachten, die die Kommunikation oder die Hilfe leichter machen. Wer sich zum ersten Mal in der Oracle Community tummelt, sollte sein Profil anpassen und mindestens den Benutzernamen von einem generierten „userXYZ“ ändern. Unschön ist allerdings, dass das Profil mehrfach gepflegt werden muss, da es zum Beispiel zwischen der „My Oracle Support Community“ und der „Developers Community“ nicht synchronisiert wird. Oracle-Mitarbeiter erkennt man am nachgestellten „-Oracle“ beim Benutzernamen.

Im folgenden Teil des Artikels sind die Beispiele meist auf die „My Oracle Support Community“ bezogen (weil diese am häufigsten genutzt wird, die anderen Communities funktionieren analog).

Die Auswahl der richtigen Kategorie beziehungsweise des richtigen Forums für die eigene Frage ist extrem wichtig. Es gibt diverse Foren mit zum Teil weiteren Unterforen, in denen die Fragen gestellt werden können, namentlich erwähnt seien hier die Oracle-Datenbank beziehungsweise Oracle Engineered Systems mit ihren diversen Unterfachgebieten. Da die überwiegende Mehrzahl der Helfer je-



RAC database evicts second database if both private interfaces are running

user8647708 Posts: 2 Blue Ribbon
Jul 19, 2022 8:46PM 4 comments Answered

Oracle GRID 19.14/Oracle Enterprise Edition 19.14. On RHEL 7 OS

We were previously able to install Oracle RAC using a well-defined process on Oracle 19.3. We then patched the GRID and DB SW to 19.14 (the January 2022 RU).

Now when we do a fresh RAC install with 19.14 installed, the GRID/ASM install works properly and can run on both servers, but when we try to start both databases (not grid/asm - the actual database instance), it gets an "oraping dectected connectivity issues" and evicts one of the instances.

If we disable one of the private interfaces on the Linux host, it will start with no issue. We can use an OS ping between the private interfaces, and that all works properly.

The current install process we are using looks something like this:

1 - install GRID SW, patch GRID, then install GRID (including ASM).

2 - install DB SW, patch DB SW with 19.14 (NOTE: we did find we need to re-run the GRID RU to get the DB SW patched properly - I was really hoping this was the issue but it wasn't). Then install the DB.

3 - add the DB to srvctl, try to start it, and it won't start on both servers at the same time.

Would anyone have thoughts on what we could look at next?

Flag Quote Off Topic Like

Abbildung 3: Post mit Detailinformationen/Problembeschreibung [https://tinyurl.com/ycy23h7s] (Quelle: Jörg Sobottka)

weils ihre technologischen Spezialgebiete haben und nur in den passenden Foren mitlesen, können (qualifizierte) Antworten auch nur erwartet werden, wenn das Forum wirklich passt. Verpönt sind Quer- beziehungsweise Mehrfach-Postings von Fragen in unterschiedlichen Foren – damit wird der ein oder andere Helfer abgeschreckt und antwortet oftmals überhaupt nicht. Bestenfalls besteht die Antwort aus einem Link auf einen der anderen bestehenden Threads des gleichen Themas. Apropos Thema: Eine kurze, prägnante Überschrift bewährt sich ebenfalls – in den Foren werden zuerst nur die Überschriften angezeigt. Wenn ein potenzieller Helfer mit der Überschrift nichts anfangen kann, sinkt die Wahrscheinlichkeit, qualifizierte Antworten zu erhalten, enorm. Wie überall bei sozialen Medien ist die Netiquette wichtig. Je freundlicher Fragen gestellt werden, desto höher ist die Hilfsbereitschaft. Englische Posts haben naturgemäß die größte Reichweite – es gibt aber auch immer wieder Posts in allen möglichen anderen Sprachen.

Details zählen

Auch wenn die Überschrift aussagekräftig ist, im folgenden Screenshot fehlt dennoch eine Detailbeschreibung des Problems.

Verständlich, dass es hier keine bis wenige Antworten dazu gibt, auch wenn es sich um ein Allerweltsproblem handelt. Wichtig ist also, dem Hilfesteller so viele Details wie möglich, aber so wenig wie nötig zu geben. Entfernt werden sollten sämtliche interne Details, also Hostnamen, IP-Adressen, Passwörter etc. – und das auch aus Screenshots oder Reports. Apropos Reports. Wer einen ASH- oder AWR-Report in Gänze anhängt, wird häufig keine Antwort erhalten. Wie bereits erwähnt, basiert ein großer Teil der Communities auf Freiwilligkeit. Da kann nicht erwartet werden, dass sich die Helfer erst über einen längeren Zeitraum in ein Problem einarbeiten. Auch Anhänge mit Makro-Möglichkeiten, wie Microsoft-Office-Formate, werden von vielen aus Sicherheitsgründen nicht geöffnet. Deshalb ist es sinnvoll, bei Dateianhängen lieber auf csv- oder txt-Dateien zurückzugreifen oder kurze Snippets als Screenshot oder Text direkt im Post zur Verfügung zu stellen. Außerdem sollte bekannt gegeben werden, welche Software(-teile), Releases, Versionen oder CPUs betroffen sind. Dies ist vor allem dann wichtig, wenn Fragen zu älteren Releases gestellt werden – das kann auch schon einmal eine 9i-Daten-

Abbildung 4: Diskussion „Deinstallation" [https://tinyurl.com/3mw4ze5w] (Quelle: Jörg Sobottka)

bank, Forms 6 oder ein alter WebLogic sein, die schon lange aus dem offiziellen Support, aber eben doch noch irgendwo im produktiven Einsatz sind. Da viele Experten eine lange Oracle-Historie mit sich herumtragen, erhält man selbst für solche alten Versionen durchaus Hilfe.

Was schon probiert wurde, allerdings nicht zu einem Erfolg führte, gehört ebenfalls zu einem guten Post mit dazu. Hinzuzufügen, wann und wie etwas aufgetreten ist oder welche Informationen bei einer Umsetzung direkt fehlen, erhöhen ebenfalls die Wahrscheinlichkeit, qualifizierte Antworten zu erhalten. Ein nahezu perfektes Beispiel sehen Sie in *Abbildung 3*.

Wissen teilen

Die einfachste Art, Wissen zu teilen, ist natürlich, selbst Fragen in den Communities zu beantworten. Schließlich hat jeder seine Kenntnisse und aufgrund der Freiwilligkeit der Communities kann man ganz einfach, wenn etwas Zeit übrig ist, durch die offenen oder neuesten Posts der Communities durchscrollen und Fragen beantworten. (Manchmal findet man dabei auch Dinge, die man selbst noch nicht wusste.) Da es häufig kein „richtig" oder „falsch" gibt, entspinnt sich oft eine Diskussion von mehreren Personen. Oder es wird ein „als Zusatz bitte noch an dieses oder jenes denken" hinzugefügt.

Neben den reinen Frage-Posts gibt es auch noch sogenannte Diskussions-Threads. Hier besteht die Möglichkeit, Dinge anderen einfach mitzuteilen, die als wissenswert erachtet werden. Ein Beispiel des Autors ist der Diskussions-Thread aus der *Abbildung 4*, in dem auf einen Deinstallationsfehler bei Datenbankversion 19.11 und 19.12 hingewiesen wird. Leider kann nicht erkannt werden, wie häufig solche Threads von anderen Personen gefunden werden.

Der dritte Weg, Wissen zu teilen, ist der richtige Abschluss eines Frage-Threads. Bei einer Antwort auf eine Frage wird der entsprechende Thread automatisch als „Answered" markiert. Da reicht allerdings schon eine Nachfrage nach „mehr Details des Problems", dies beinhaltet also keine Wertung. Wurde die Fragestellung abschließend beantwortet, sollte der entsprechende Thread nun auch als „gelöst" markiert werden. Dadurch wird ein „Answered" grün mit einem Häkchen dargestellt *(siehe Abbildung 5)*. So markierte Threads werden bei der Lösungssuche natürlich von allen bevorzugt, die selbst nach einer Lösung suchen.

Allgemeines

Wie bereits erwähnt, sind die Profile in den verschiedenen Communities nicht unbedingt synchronisiert. Dadurch ergibt sich leider auch das Problem, dass Notifications (z.B. neue Antworten auf Threads, in denen man selbst aktiv ist) oder direkte Nachrichten nicht aus allen Communities automatisch ersichtlich sind. Man muss zwischen verschiedenen Communities also hin- und herhüpfen. Wer nicht unbedingt gleich jemanden mit direkten Nachrichten überfallen möchte, kann auch einen anderen Benutzer in einem Beitrag erwähnen, das geht mit sogenannten Mentions (also @ zuzüglich zum Benutzernamen). Der entsprechende Benutzer erhält also eine Notification, auch wenn er bisher nicht aktiv am entsprechenden Thread beteiligt war. Notifications lassen sich auf dem eigenen (Community-bezogenen) Profil jeweils sehr fein granuliert einstellen – getrennt für die beiden Wege E-Mail beziehungsweise Popup (entspricht dem Glockensymbol im Header). Wer etwas aktiver an der Community teilnimmt, wird froh sein, dass man E-Mails komplett abstellen kann.

Interessant ist auch, dass es in verschiedenen (Unter-) Foren immer wieder „Announcements" durch Oracle-Mitarbeiter gibt, zum Beispiel mit Hinweis auf neue Patches/RUs/Releases von Produkten. In diesen Threads können dann Fehler/Probleme, die mit dem Patch auftreten, oder die Verwendung neuer Features diskutiert werden.

In der My Oracle Support Community gibt es einen speziellen Bereich „My Oracle Support Tools & Trainings". Neben Hilfe zu Themen wie „Autonomous Health Framework" (TFA/orachk) oder „Auto Service Request" (ASR) findet sich auch ein Bereich „Patch – Reviews (General)", in dem über Patches/RUs von diversen Produkten diskutiert wird.

Sicherlich gibt dieser Artikel nur einen kleinen Einblick für einen möglichen Einstieg in die Oracle Communities. Es ist allerdings nach sehr kurzer Zeit leicht, sich in den Communities zu bewegen und seinen Nutzen aus den verschiedenen Bereichen zu ziehen. Man muss einfach damit starten.

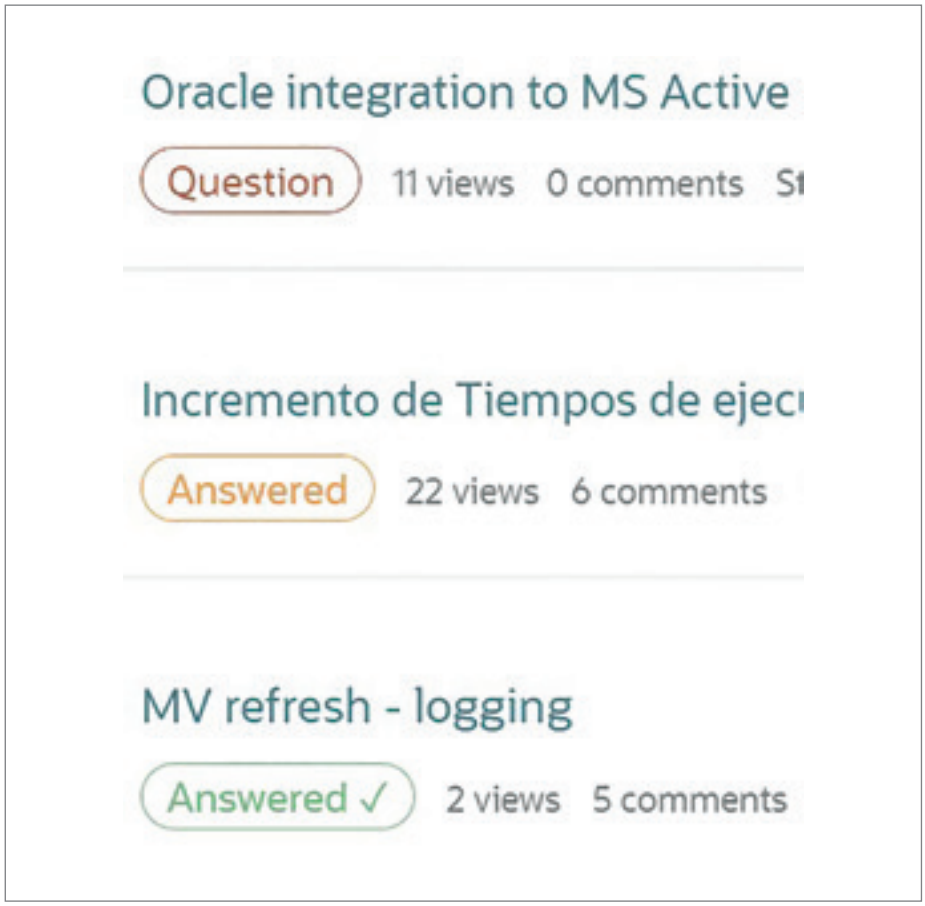


Abbildung 5: Diskussion „Deinstallation“ (Quelle: Jörg Sobottka)

Quellen

[1] *https://community.oracle.com/hub/ (Januar 2023)*

Über den Autor

Jörg Sobottka ist seit 1992 mit Oracle-Technologien beruflich verbunden und derzeit Oracle ACE Pro. Neben Artikeln schreibt er für verschiedene Blogs und wurde seit 2019 viermal als Most Valued Contributors in der My Oracle Support Community für Datenbank und Engineered Systems ausgezeichnet.

Jörg Sobottka
joerg.sobottka@robotron.ch

Oracle Datenbanken Monthly News

Auf dem deutschsprachigen Oracle-Blog ist die März-Ausgabe der News-Serie erschienen.

DOAG Online

Es ist wieder soweit: die neue Ausgabe ist online!
Das sechsköpfige Redaktionsteam von Oracle Deutschland hat wieder Neuigkeiten rund um die Oracle-Datenbank für On-Premises und Cloud-Installation zusammengestellt.

Alles wird wieder in einem Video präsentiert.

In der aktuellen Ausgabe wird wieder ein zusätzliches Quick Link Posting (in Englisch) mit den Links zur Verfügung gestellt, um einen schnellen Zugriff auf die zugehörigen Links zu gewährleisten.

https://www.doag.org/de/home/news/oracle-datenbanken-monthly-news-20/

Ransomware – Chronologie einer Katastrophe

Robert Wortmann, Trend Micro Deutschland

Ransomware ist seit Jahren eine stetige Bedrohung für Unternehmen weltweit. Immer wieder werden Teile oder sogar ganze Unternehmen verschlüsselt – doch wieso richtet diese Art der Attacke immer wieder so immensen Schaden an? In diesem Artikel beleuchten wir eine schwerwiegende Ransomware-Attacke aus dem echten Leben. Dabei gehen wir nicht nur auf den Angriff selbst ein, sondern auch darauf, was in den Jahren vor der eigentlichen Attacke geschah. Was waren in diesem konkreten Fall technische und organisatorische Versäumnisse, die dem Threat Actor die Möglichkeit boten, anzugreifen? Und wie sieht eigentlich das Geschäftsmodell hinter Ransomware aus?

Es ist der Albtraum eines jeden IT-Verantwortlichen: Das Telefon klingelt mitten in der Nacht und ein aufgeregter Administrator teilt mit, dass Teile der IT-Infrastruktur durch eine Ransomware verschlüsselt wurden. Dabei ist das Phänomen Ransomware kein neues, es existiert seit Jahrzehnten. Die Anfänge lassen sich bis in die 1980er Jahre zurückverfolgen, als erste Viren auftauchten, die Daten auf dem infizierten Computer verschlüsselten und die Entschlüsselung nur gegen Zahlung einer Gebühr ermöglichten. Damals waren die Angriffe jedoch noch sehr begrenzt und die meisten Viren konnten leicht entfernt werden. Im Laufe der Zeit wurden Ransomware-Angriffe jedoch immer raffinierter und erreichten im 21. Jahrhundert eine neue Stufe der Bedrohung, als Hacker begannen, Unternehmen und Regierungsbehörden als Ziele auszuwählen und hohe Lösegelder zu fordern.

Früh zielgerichtet investiert

Der beschriebene Fall betrifft ein typisches Unternehmen des produzierenden deutschen Mittelstands. Die ca. 1500 Angestellten befinden sich neben dem Hauptstandort Deutschland über einige andere Standorte weltweit verteilt.

Das Unternehmen investierte vergleichsweise früh große Summen in die Digitalisierung und erkannte auch schnell den Stellenwert von IT-Security – mit entsprechenden Investitionen in präventive Maßnahmen. Im Zuge des Baus eines neuen Rechenzentrums gab es 2015 ein größeres IT-Security-Maßnahmenpaket. Dabei wurden ein sogenannter „Next-Generation" Malware-Scanner, neue Firewalls inklusive Deep Packet Inspection sowie eine E-Mail Sandboxing Appliance beschafft. Da die Verantwortlichen bereits damals Herausforderungen in der Betreibbarkeit dieser Technologien sahen, beauftragten Sie einen Netzwerk-Mitarbeiter, die Hälfte seiner Arbeitszeit primär für den Security-Betrieb aufzuwenden. Verglichen mit zahlreichen Unternehmen ähnlicher Größe und Branche waren diese Maßnahmen nicht nur zeitgemäß, sondern in vielen Bereichen führend.

Mit einem blauen Auge davongekommen

Die weltweite Verbreitung der WannaCry-Ransomware am 12. Mai 2017 war einer der größten und weitreichendsten Ransomware-Angriffe in der Geschichte. WannaCry nutzte eine Schwachstelle im SMB-Protokoll (Server Message Block) aus, die als MS17-010 oder EternalBlue bekannt ist. Diese Schwachstelle ermöglichte es Angreifern, auf entfernte Computersysteme zuzugreifen und Schadcode auszuführen, ohne dass eine Authentifizierung erforderlich war. Obwohl ein offizieller Microsoft-Patch für die Behebung der Schwachstelle bereits am 14. März 2017 verfügbar war, wurde dieser von vielen Organisationen nicht zeitnah installiert.

So traf es auch das in diesem Artikel beschriebene Unternehmen am Abend des 12. Mai 2017: Innerhalb weniger Minuten breitete sich die Malware insbesondere im Client-Netz lateral über das Unternehmensnetzwerk aus und verschlüsselte dabei über 80 Systeme. Nachdem die ersten Meldungen über den Helpdesk eingingen, wurde ein Krisenstab eingerichtet, der Gegenmaßnahmen einleiten sollte. Noch bevor es dazu kam, wurde der Angriff weltweit gestoppt, da ein britischer Cybersecurity-Forscher namens Marcus Hutchins einen Kill-Switch im Schadcode der Malware entdeckt hatte. Dieser Kill-Switch war eine Art Notbremse, die die Ausbreitung der Malware automatisch stoppte, wenn eine bestimmte, nicht registrierte Internet-Domain aufgerufen wurde. Hutchins hatte die Domain registriert und dadurch den Angriff gestoppt.

Nach dem zweitägigen Einsatz eines externen Forensik-Teams wurden die betroffenen Systeme neu aufgesetzt

oder aus dem Backup zurückgespielt. Das interne Fazit war trotz des entstandenen Schadens positiv: Im Vergleich zu anderen betroffenen Unternehmen funktionierten insbesondere die Recovery-Maßnahmen gut. Dass der Angriff allerdings primär durch die Aktionen von Marcus Hutchins gestoppt wurde, kam in der retrospektiven Betrachtung der Ereignisse zu kurz. So wurde beispielweise nie die Frage gestellt, wieso bereits zur Verfügung stehende Patches nie installiert wurden und was passiert wäre, hätte Hutchins keinen Kill-Switch gefunden.

Weiter so!

Die IT-Verantwortlichen nahmen die Ereignisse im Mai 2017 zwar als Warnung, allerdings auch als Bestätigung der bisherigen Strategie wahr. Zudem wurden 2017 weitere Attacken erfolgreich ohne entstandenen Schaden abgewehrt. Bis ins Jahr 2021 erfolgten deshalb nur wenige Änderungen am Sicherheitskonzept: Neben dem Austausch von Firewall-Hardware und einigen Updates der Malware Protection Engines wurden auf Anraten des externen Sicherheitsberaters weitere Awareness-Maßnahmen für Mitarbeiter aufgenommen. Darüber hinaus nahmen die Verantwortlichen keine konzeptionellen Veränderungen vor und auch die Überprüfung von getroffenen Einstellungen, beispielsweise in der Endpoint Protection, wurde hinter andere Projekte zurückgestellt. Besonders schwerwiegend ist die Tatsache, dass trotz der Vorkommnisse im Jahr 2017 keine grundlegende Änderung im Bereich des Schwachstellenmanagements sowie der Ausbringung von Patches erfolgte.

Aufseiten der Angriffe gab es in diesen vier Jahren jedoch zahlreiche Neuerungen. Zwischen 2017 und 2021 veränderte sich Malware und insbesondere Ransomware in Bezug auf Backup und Restore in einigen wichtigen Aspekten:

1. Erweiterung des Angriffsbereichs: Ransomware-Akteure entwickelten ihre Schadsoftware so, dass sie nicht nur die auf dem betroffenen Computer gespeicherten Daten verschlüsselt, sondern auch auf Netzwerk- und Cloud-Speicher zugreift. Dadurch wird es für Opfer schwieriger, auf dort gesicherte Daten zurückzugreifen, um ihre Systeme wiederherzustellen
2. Backup-Verschlüsselung: Einige Ransomware-Varianten entwickelten die Fähigkeit, auch gesicherte Daten auf Backup-Systemen zu verschlüsseln. Dadurch werden die Möglichkeiten für die Wiederherstellung von Daten weiter eingeschränkt.
3. Verschlüsselung: Ransomware entwickelte sich auch in Bezug auf die Verschlüsselungstechnologie weiter: Im Laufe der Zeit nutzten Angreifer immer stärkere Verschlüsselungsmethoden, die es schwieriger machen, die Daten ohne Zahlung der Lösegeldforderungen wiederherzustellen.
4. Ransomware-as-a-Service (RaaS): Ransomware-Autoren begannen damit, ihre Schadsoftware als Dienstleistung anzubieten, wobei andere Personen die Malware verwenden und Lösegeldforderungen an die Opfer weiterleiten können. Dies erleichtert es auch Anfängern, Ransomware-Angriffe durchzuführen.

Irgendwas stimmt nicht

An einem Morgen im Juni des Jahres 2021: Das IT-Operations-Team des Unternehmens blickte auf ein Monitoring voller roter Meldungen. Beinahe alle Serversysteme wiesen extrem hohe Lastwerte auf und nach einer kurzen Phase des Hoffens auf einen „False-Positive"-Fall wurde bereits die erste Ransomware-Mitteilung entdeckt. Darin wurde das Unternehmen zur Zahlung von umgerechnet ca. 295.000 € aufgefordert, zahlbar in Bitcoin.

Die eingesetzte Malware konnte schnell REvil zugeordnet werden. REvil, auch bekannt als Sodinokibi, ist eine Ransomware-Familie, die erstmals im April 2019 auftauchte. Sie zog schnell Aufmerksamkeit auf sich, da sie sich auf Unternehmen spezialisiert hatte und dabei vergleichsweise hohe Lösegeldforderungen stellte. Die Ransomware nutzt in der Regel Schwachstellen in Remote-Desktop-Protokollen, um in das Netzwerk einzudringen und sich auszubreiten. Nach dem Eindringen verschlüsselt sie wichtige Dateien und fordert ein Lösegeld, um die Dateien wiederherzustellen. REvil erwies sich durch seine fortschrittliche Verbreitungstechnologie und seine Fähigkeit, Netzwerke schnell und tiefgreifend zu infizieren, als besonders gefährlich.

Nachdem auch im Jahre 2021 immer noch kein niedergeschriebener Notfallplan existierte, begann das Unternehmen mit allen verfügbaren internen Ressourcen gegen das Problem anzukämpfen. So wurden Serversysteme und andere Services wie E-Mail heruntergefahren und IT-ferne Mitarbeiter nach Hause geschickt. Trotz aller Bemühungen breitete sich die Malware immer weiter im Netzwerk aus und gelangte auch bis an das primäre Backup. Dabei wurde zwar nicht das komplette Backup verschlüsselt, allerdings konnte die Integrität nach Befall des Systems nicht mehr gewährleistet werden.

Nach drei Tagen ohne Schlaf und sichtbare Erfolge begab sich das Team auf die Suche nach externer Unterstützung. Hierbei konnte leider nicht auf einen sogenannten Incident-Response-Retainer zurückgegriffen werden, der einen Einsatz nach Reaktionszeit vertraglich garantiert hätte. Stattdessen wurde ein bisher unbekannter Dienstleister hinzugezogen. Nach erster Klärung der Lage begann die forensische Arbeit des Teams. Diese stellte sich nach kurzer Zeit als äußerst schwierig dar, da die notwendigen historischen Daten nicht in ausreichender Qualität verfügbar waren. Firewall-Logs wurden dabei nur lokal auf den Geräten gespeichert, wobei eine Löschung nach jedem Neustart erfolgte. Auf Basis der Server- und Clientsysteme standen zudem nur lokale Windows-Event-Logs sowie Speicherabbilder zur Verfügung. Weitere Daten, insbesondere Telemetriedaten von Endpunkten, gab es aufgrund fehlender Technologieerweiterung der 2015 eingeführten Endpoint Protection nicht.

Aufgrund der Tatsache, dass keine saubere historische Analyse der Ereignisse möglich war, kommunizierte das Incident-Response-Team dem Unternehmen schnell, dass auch der Restore der Daten vom Backup-Medium-Band als eher unrealistisch zu betrachten sei. Diese Feststellung basierte neben der mangelhaften Übertragungsgeschwindigkeit insbesondere auf der Tatsache, dass durch die fehlende historische Nachverfolgung nicht sichergestellt werden konnte, dass kein bereits kompromittiertes System wiederhergestellt würde.

Während das Incident-Response-Team die Arbeit somit auf das weitere Monitoring konzentrierte, fand sich die Unternehmensleitung zunehmend mit dem Gedanken ab, die Lösegeldsumme zu bezahlen, da dies nach Meinung zweier unabhängiger Quellen die einzige Option auf Wiederherstellung des IT-Betriebs war. Nach einigen eher glücklosen Verhandlungsversuchen mit den Ransomware-Akteuren wurde die Summe übermittelt. Dabei wurde auf ein Verhandlungsteam eines externen Dienstleisters gesetzt, der insbesondere die Authentizität der Key-Übermittlung sicherstellen sollte. Die Übermittlung der Keys geschah wie versprochen und die Entschlüsselung der Daten konnte beginnen.

Trotz der Entschlüsselung musste mehr oder weniger ein Neuaufbau der IT erfolgen. So wurden alle Serversysteme neu aufgesetzt und nur Applikationsdaten wie die Inhalte von Oracle-Datenbanken auf die neuen Systeme zurückgespielt. Während die Kommunikation via Mail nach ca. acht Tagen wieder verfügbar war, benötigte es für den Aufbau anderer Komponenten mehrere Wochen. Die Dauer des kompletten Wiederaufbaus wurde dabei auf etwa fünf Monate bemessen.

Es geht doch!

Bereits während die eigentliche Incident Response noch im Gange war, leitete das Unternehmen erste Maßnahmen ein, um zukünftig für eine höhere Sicherheit sorgen zu können. So wurden technologische Maßnahmen wie Endpoint Detection and Response sowie ein Monitoring von netzwerkbasiertem Lateral-Movement, die beide im Zuge der Incident-Response-Maßnahmen platziert wurden, in den Betrieb übernommen. Auch der 24x7-Monitoring-Service dieser Komponenten sowie die Beauftragung eines vertraglich zugesicherten Incident-Response-Retainers erfolgten wenige Wochen nach Eintreten des Vorfalls.

Neben technologischen Maßnahmen ergriff das Unternehmen insbesondere viele organisatorische Schritte: Wenige Monate später wurde ein erfahrener Chief Information Security Officer (CISO) eingestellt und mit der Implementierung eines Information Security Management System (ISMS) beauftragt. Die ersten Maßnahmen innerhalb des ISMS waren dabei die Implementierung eines robusten Schwachstellenmanagement-Prozesses, die Erstellung von Notfallplänen sowie die Optimierung des Software-Patch-Prozesses. Dabei agiert der CISO als direkter Mitarbeiter des CEO des Unternehmens und unterrichtet den Vorstand regelmäßig über die digitale Unternehmenssicherheit.

Konzepte müssen ständig überdacht werden

Der hier dargestellte Fall zeigt eindrücklich, wie wichtig es ist, vorhandene IT-Security-Konzepte regelmäßig zu evaluieren und gegebenenfalls zu aktualisieren. Immerhin entwickeln sich auch die Bedrohungslage und die Art der Angriffe ständig weiter. Neue Technologien und Anwendungen bringen neue Sicherheitsrisiken mit sich und Angreifer arbeiten an neuen Methoden, um Unternehmen anzugreifen. In der Folge können auch Unternehmen wie das beschriebene schnell von einemvergleichsweise hohen Sicherheitsniveau abfallen und zur „leichten Beute“ von Angreifern werden.

Deshalb ist es besonders wichtig, alle getroffenen Maßnahmen regelmäßig auf ihre Effektivität zu überprüfen. Dazu kann beispielsweise auf ein Red Teaming zurückgegriffen werden, bei dem ein Team von Sicherheitsexperten simulierte Angriffe auf das Unternehmen durchführt, um die Wirksamkeit der bestehenden IT-Security-Maßnahmen zu testen. Diese Tests können sowohl technische als auch soziale Angriffe umfassen, die auf eine realistische Simulation von Angriffsmethoden abzielen, die von echten Angreifern verwendet werden könnten. Bei einem Red-Team-Einsatz werden die Angriffe in enger Absprache mit dem Unternehmen durchgeführt, um sicherzustellen, dass keine Schäden entstehen und eine detaillierte Analyse der Schwachstellen und Erfolge des Unternehmens zu ermöglichen. Das Ziel ist es, realistische Szenarien zu schaffen, die die tatsächlichen Bedrohungen simulieren, denen das Unternehmen ausgesetzt sein kann.

Eine andere Methode ist der Abgleich mit dem MITRE ATT&CK-Framework. Eine Gap-Analyse mit MITRE ATT&CK ermöglicht es IT-Security-Teams, ihre Schutzmaßnahmen gegen bekannte Angriffsmethoden und -techniken zu überprüfen. MITRE ATT&CK ist ein umfassendes Referenzmodell für Angriffe, das verschiedene Angriffsvektoren, Techniken und Prozesse beschreibt, die von Angreifern verwendet werden können. Indem IT-Security-Teams ihre Schutzmaßnahmen mit diesem Modell abgleichen, können sie potenzielle Schwachstellen identifizieren und gezielte Maßnahmen ergreifen, um diese zu schließen. So kann sichergestellt werden, dass die IT-Sicherheitsmaßnahmen auf dem neuesten Stand sind und die Organisation vor bekannten Angriffen besser geschützt ist.

Über den Autor

Robert Wortmann ist ein erfahrener Experte für die Analyse und Planung komplexer Cyber-Defense-Architekturen und -Technologien. Bei Trend Micro Deutschland ist er verantwortlich für Cyber-Defense-Architekturen mit besonderem Fokus auf Advanced Detection und Response.

Robert Wortmann
robert_wortmann@trendmicro.com

Was wir aus den Conti-Leaks lernen können, um uns vor Ransomware und Cyber-Erpressung zu schützen

Anna Collard, SVP Content Strategy and Evangelist KnowBe4 Africa

Nur vier Tage nach dem offiziellen Beginn des Krieges zwischen Russland und der Ukraine im Jahr 2022 veröffentlichte ein ukrainischer Insider der in Russland basierten Conti-RaaS-Gruppe (Ransomware as a Service) Tausende interne Chatprotokolle. Die unter dem Twitter Handle @contileaks veröffentlichen Dokumente enthüllten die täglichen Aktivitäten und die Struktur dieses „Cybercrime-Unternehmens“ und zeigten, dass Conti in vielerlei Hinsicht wie ein normales mittelständiges Unternehmen arbeitete. Im Jahr 2021 war die Conti das führende Ransomware-Syndikat mit über 180 Millionen $ Einkommen.

Aus den Leaks ging hervor, dass die Gruppe bei der Suche nach neuen Mitarbeitern auf seriöse Headhunter-Dienstleister zurückgriff und dass Leistungsbeurteilungen, Weiterbildungsmöglichkeiten und ein „Mitarbeiter des Monats"-Programm bei der Conti zur Tagesordnung gehörten. Am überraschendsten war vielleicht, dass einige Mitarbeiter gar nicht wussten, dass sie für Cyberkriminelle arbeiteten, sondern dachten, dass man Software für Penetrationstests herstellte und Fachleute benötigte, die diskret arbeiten konnten. Wenn einer dieser unwissenden Mitarbeiter erfuhr, für wen sie tatsächlich arbeiteten, wurde ihm ein Bonus angeboten, damit er bleibt und schweigt.

Die Chats deuteten auch darauf hin, dass das Ende von Conti nahe war, schon bevor die Leaks an die Öffentlichkeit kamen. Zum Beispiel gingen aus den Aufzeichnungen hervor, dass die Gehaltszahlungen im Januar 2022 eingestellt und einige Nutzer, die für den Betrieb wichtig waren, inaktiv wurden.

Zum jetzigen Zeitpunkt ist unklar, ob Conti wirklich verschwunden ist oder ob die Gruppe (oder bestimmte Mitglieder) einfach nur die Zeit nutzen, um sich umzustrukturieren und in Zukunft ein Comeback zu feiern.

Im Mittelpunkt der Ransomware-Kriminalität steht der Grundgedanke, dass, wenn man jemandem etwas Einzigartiges und Wertvolles wegnimmt, er dafür bezahlen wird, es zurückzubekommen. Wenn Sie ein Geheimnis von jemandem aufdecken, wird er Sie dafür bezahlen, dass Sie es geheim halten. Der mikrokosmische Markt mit einem Verkäufer und einem verzweifelten Käufer, bei dem das Risiko für den Ransomware-Kriminellen fast gleich null ist, treibt die Erpresserpreise und immense Gewinne für den Kriminellen.

Aus diesem Grund bleibt Ransomware weiterhin an der Spitze der Cyberangriffe, auch in diesem Jahr. Laut dem Data Protection Trends Report 2022 wurden nur 24 % der Unternehmen nicht von Ransomware angegriffen. Die häufigsten Einfallstore sind Schwachstellen in extern erreichbaren Systemen, schlecht konfiguriertes Microsoft Remote Desktop Protocol (MRDP), schwache Benutzerauthentifizierung und Social-Engineering-Attacken wie Phishing-E-Mails. Diese Strategien wurden auch von der Conti benutzt, manchmal sogar in Kombination mit Vishing Campaigns.

Eine aktuelle Analyse von Statista ergab, dass 66 % der Unternehmen weltweit im Jahr 2022 Opfer eines Ransomware-Angriffs wurden, wobei Österreich mit 84 % an der Spitze liegt, gefolgt von Australien mit 80 % und Malaysia mit 79 %.

Die Botschaft ist klar: Ransomware ist eine Form von Cyberangriffen, die nicht verschwinden wird und für die es kein Patentrezept gibt, um sie zu stoppen. Die einzige Möglichkeit, ein Unternehmen vor Cyberangriffen zu schützen, ist die harte Arbeit der „Security in Depth", das Erbauen von Sicherheitsschichten im gesamten Unternehmen.

Der erste Schritt, um die Bedrohung durch Ransomware zu minimieren, ist die Schaffung einer Sicherheitskultur im Unternehmen. Dabei geht es nicht nur um ein paar Plakate und E-Mails, die die Mitarbeiter vor den Risiken warnen und sie zu guten Passwörtern ermuntern. Es geht um eine kontinuierliche Sensibilisierung und Schulung, die Wachsamkeit in Verhaltensweisen und Vorgehensweisen verankert. Die Mitarbeiter müssen verstehen, dass Sicherheit nicht etwas ist, das

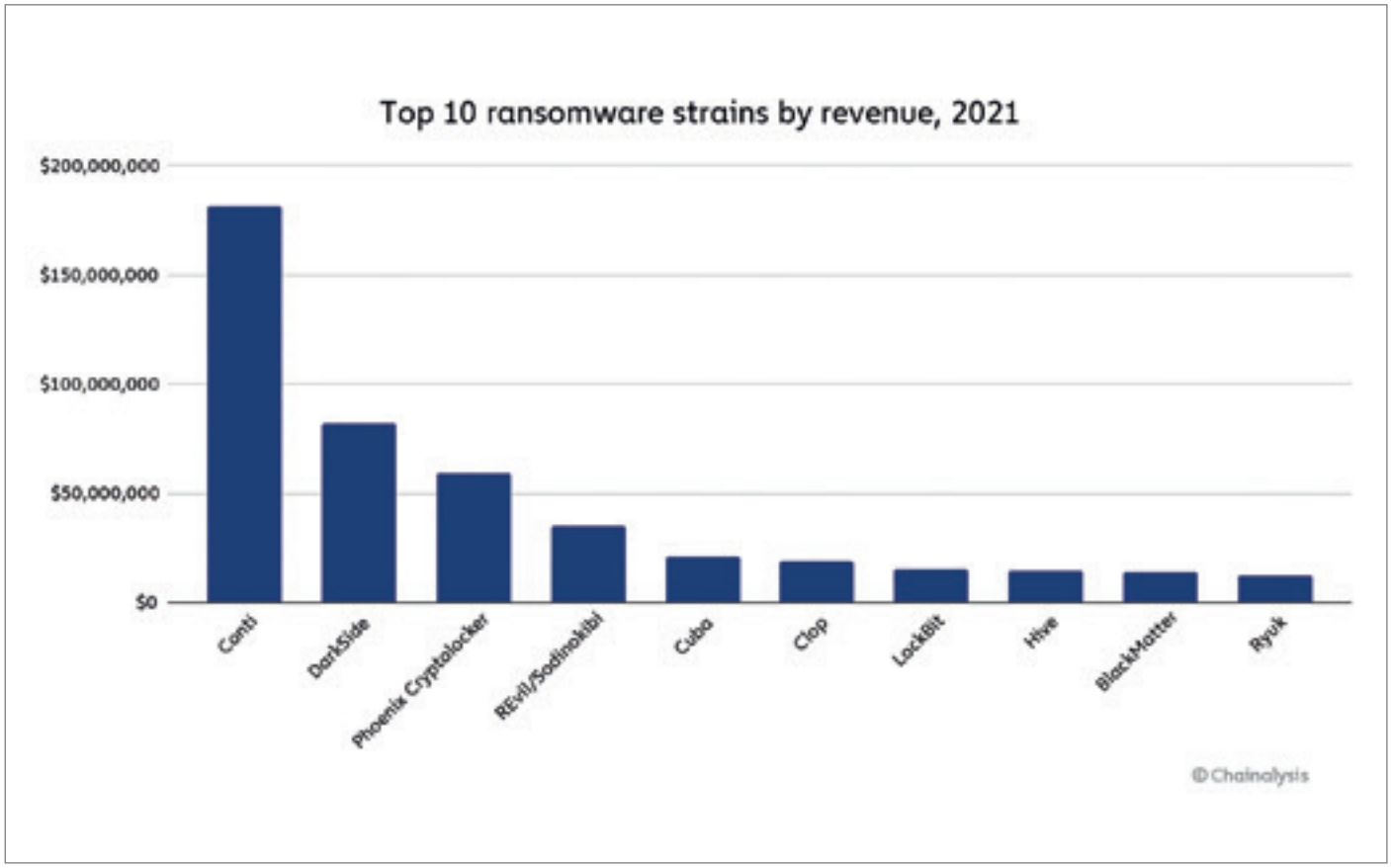


Abbildung 1: Tabelle mit den Top 10 der Ransomware-Stämme nach Umsatz (Quelle: Anna Collard)

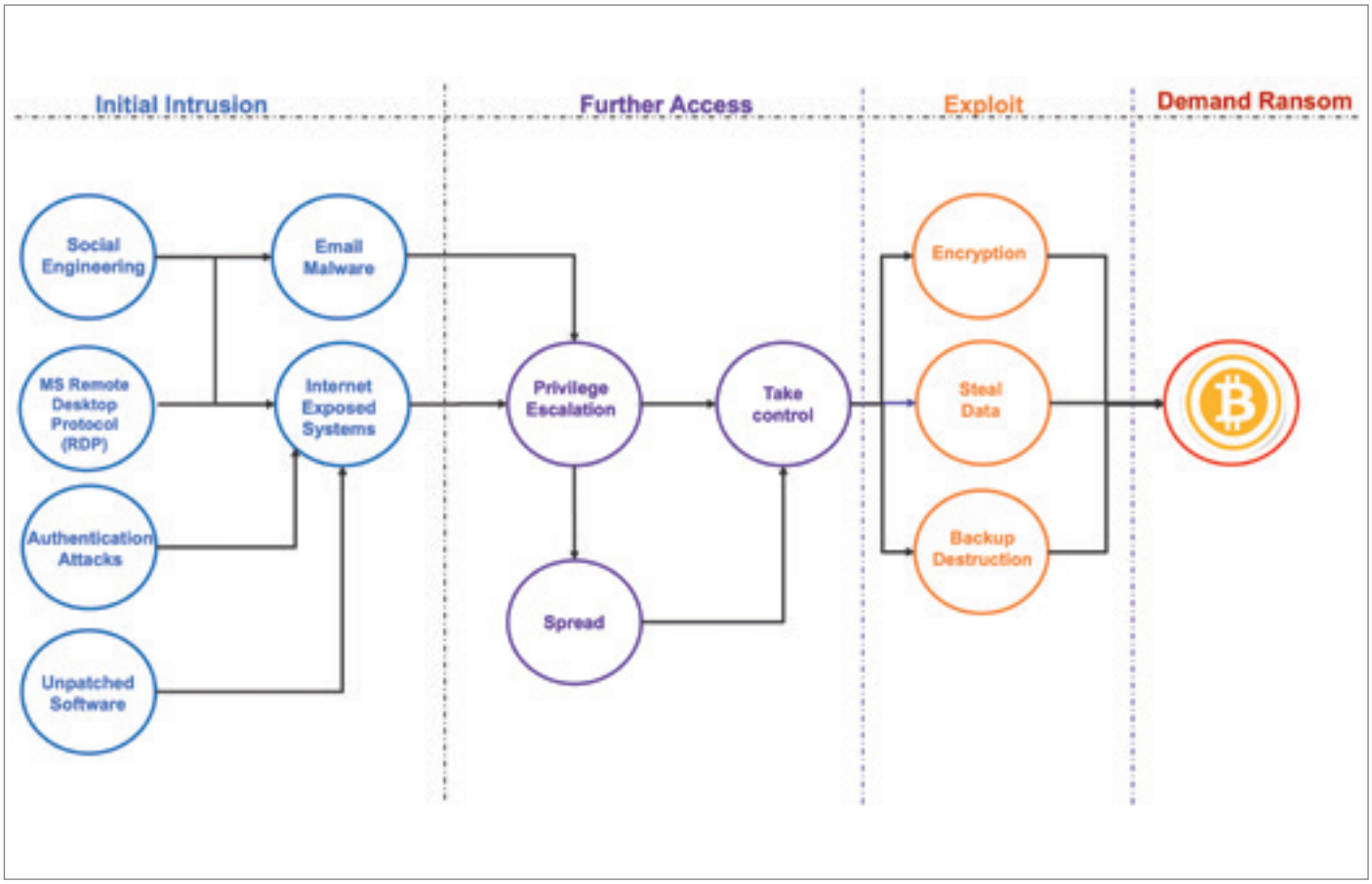


Abbildung 2: Ransomware-Tötungskette (Quelle: Anna Collard)

von einer Person in der IT-Abteilung vorgeschrieben wird, die nicht versteht, wie viel Arbeit sie zu tun haben oder wie frustrierend es ist, unter Zeitdruck über mehrere Sicherheitsschleifen zu springen.

Es kann leicht passieren, dass Mitarbeiter bei ihren Passwörtern, der Verwaltung ihrer Zwei-Faktor-Authentifizierungsprotokolle oder der Erkennung von Phishing-E-Mails nachlässig werden. Doch so mühsam und frustrierend Sicherheit auch sein kann, eine Kompromittierung ist es noch mehr. Deshalb sind Schulungen so wichtig.

Natürlich sind Schulung und Sensibilisierung nur eine Seite der Medaille. Entscheidend ist auch, dass das Unternehmen über solide Sicherheitsvorkehrungen in den Bereichen Endpunkt-Erkennung, Bedrohungserkennung, Incident Management, Patch-Management und Cyber-Versicherung verfügt. Um das Risiko von Ransomware zu minimieren, gibt es nicht nur einen einzigen Ansatz, sondern mehrere. Jedes Element spielt eine Rolle bei der Eindämmung der Bedrohung und bei der Minimierung der Gesamtauswirkungen auf das Unternehmen. Wichtig ist auch, sich auf den schlimmsten Fall mit regelmäßig getesteten Inzident-Response-Prozessen vorzubereiten, selbst wenn dieser – hoffentlich – nicht eintritt.

Über die Autorin

Anna Collard ist SVP of Content Strategy & Evangelist bei KnowBe4. Seit 18 Jahren ist sie in der Cybersecurity tätig: als Geschäftsführerin von KnowBe4 Africa und davor als Sicherheitsarchitektin und Beraterin, wo sie Organisationen in Südafrika, Europa und den USA beim Aufbau ihrer Cybersecurity-Abwehr unterstützt hat. Sie gründete die Security-Awareness-Firma Popcorn Training, die 2018 von KnowBe4 akquiriert wurde. Anna gewann den „Women in Tech Innovations Throughout Africa 2020 Award“ für das südliche und zentrale Afrika. Sie wurde in die „Top 50 Women in Cybersecurity – Africa 2020“ sowie in die „Top 100 Women in Cyber 2020 und 2021“ weltweit vom Cyber Defence Magazine aufgenommen. Sie wurde mit dem prestigeträchtigen „ISACA South Africa President Award“ für 2020 ausgezeichnet. Collard ist Inhaberin verschiedener Sicherheitszertifikate wie CISSP, CISA, ISO 27001 Lead Auditor, CIPP/IT und war früher Visa/Mastercard PCI DSS QSA.

Anna Collard
annac@knowbe4.com

Oracle-Lizenzierung in der OCI-, AWS-, Azure- und Google-Cloud

Michael Skowasch, Ordix

Viele Kunden stehen vor der Herausforderung, ihre Oracle-Datenbanken in die Public Cloud zu migrieren. Einige Kunden haben sich bereits für eine Public Cloud entschieden und stellen sich nun die Frage: Können meine Oracle-Datenbanken auch in diese Public Cloud migriert werden? Oder ist eine andere Public Cloud sinnvoller und daher ein Multi-Cloud-Ansatz (mehrere Public Cloud-Anbieter) zu wählen? In diesem Artikel werde ich Ihnen die Vor- und Nachteile dieser Lösungsoptionen aufführen.

Allgemein

Bevor ich Ihnen die einzelnen Public-Cloud-Anbieter vorstelle, möchte ich ein paar allgemeine Begriffe erklären.

In der Public Cloud werden verschiedene Modelle angeboten. Mit dem Modell Infrastructure as a Service (IaaS) können physische oder virtuelle Server beim Cloud-Anbieter exklusiv genutzt werden. Eventuelle Software-Installationen wie die Datenbank-Software und das Erstellen einer Datenbank übernehmen Sie komplett selbst wie auf On-Premises, also wie in Ihrem eigenen Rechenzentrum. Das Modell Platform as a Service (PaaS) stellt nur einen Service zur Verfügung, beispielsweise kann eine Datenbank auf physischen oder virtuellen Servern mit wenigen Klicks erstellt werden. Eine manuelle Software-Instal-

lation und das manuelle Erstellen der Datenbanken entfallen.

Die Lizenzmodelle für den Betrieb von Oracle-Datenbanken in der Public Cloud sind unterschiedlich. Im Modell PaaS kann das Lizenzmodell „Oracle-Lizenzierung bereits im Preis eingerechnet" oder „BYOL (Bring-your-own-licenses)" für PaaS und IaaS gewählt werden.

Das Lizenzmodell „Oracle-Lizenzierung bereits im Preis eingerechnet" ist nur für PaaS verfügbar und beinhaltet alle Oracle-Lizenz- und -Wartungskosten. Mit diesem Modell ist man lizenzsicher unterwegs.

Das Lizenzmodell „BYOL" kann für PaaS und IaaS gewählt werden. Im Modell IaaS ist nur „BYOL" möglich, weil Sie hier die Datenbank-Software und die Datenbank selbst installieren beziehungsweise erstellen und administrieren. Die Oracle-Lizenzen sind bei diesem Modell bereits On-Premises vorhanden und werden in die Cloud übertragen. Zu beachten ist hier, dass die Wartungskosten weiterhin bezahlt werden müssen. Die Kosten für dieses Lizenzmodell sind grundsätzlich günstiger als bei „Oracle-Lizenzierung bereits im Preis eingerechnet".

Die Abrechnung dieser Lizenzmodelle in der Public Cloud erfolgt stundenweise auf „CPU"-Basis. In der Regel werden hier vCPUs (virtuelle CPUs) berechnet. Oft werden die Server in der Public-Cloud mit Hyperthreading betrieben. Hier entsprechen also zwei vCPUs einem physikalischem Core. Die Anzahl der vCPUs pro Server bestimmt also maßgeblich die Kosten. Hinzu kommen Speicher, Backup, Netzwerk- und auch Netzwerkübertragungskosten.

Oracle Cloud Infrastructure (OCI)

In der OCI werden für die Oracle-Datenbanken die Cloud-Modelle IaaS und PaaS angeboten.

IaaS

Bei IaaS können Sie physische oder virtuelle Server exklusiv nutzen. In der OCI nennt man diese Server „Compute-Instanzen". Sie können auf diesen Servern selbst ein OS-Image auswählen, die Oracle-Software installieren und die Datenbank erstellen. Hier kann nur das Lizenzmodell „BYOL" gewählt werden.

PaaS

Mit PaaS können Sie eine Oracle-Datenbank mit wenigen Klicks in der OCI-Console auf sogenannten „Virtual Machines" erstellen. Natürlich können auch mit Terraform oder ähnlichen Tools die Datenbanken per Skript aufgebaut werden. Beide Lizenzmodelle „Oracle-Lizenzierung bereits im Preis eingerechnet" oder „BYOL" können ausgewählt werden. Es kann nur eine Datenbank pro „Virtual Machine" erstellt werden und Sie können standardmäßig nur die letzten vier Oracle-Versionen eines Releases auswählen. Als Editionen sind die Standard Edition 2 (SE2), Enterprise Edition (EE), EE High Performance oder EE Extreme Performance verfügbar. In jeder dieser Editionen können die Optionen und Management-Packs laut *Abbildung 1* verwendet werden. Bei der Wahl des Lizenzmodells „Oracle-Lizenzierung bereits im Preis eingerechnet" sind alle Kosten bezüglich der aufgelisteten Optionen und Management-Packs abgedeckt. Es fallen die vCPU-Kosten der Edition und der ausgewählten „Virtual Machine" an. Werden mit „BYOL" EE-Lizenzen eingebracht, können auch alle aufgeführten Optionen und Management-Packs laut *Abbildung 1* verwendet werden. Doch dazu später mehr. Sie können zwischen der Grid Infrastructure (GI) mit ASM (Automatic Storage Management), bei denen die Datenfiles in Diskgruppen und nicht in Mountpoints liegen, und dem Logical Volume Manager wählen. Wählen Sie den Logical Volume Manager, liegen die Datenfiles in Mountpoints. Bei dieser Auswahl des Logical Volume Manager ist die Datenbank auf der „Virtual Machine" deutlich schneller erstellt. Auch beim späteren Patchen müssen hier nicht erst die GI und dann das Oracle-Home der Datenbank gepatcht werden, sondern nur das Oracle-Home der Datenbank. Die GI muss allerdings gewählt werden, wenn eine RAC-Lösung (Real Application Cluster) mit zwei „Virtual Machines" gewählt wird. Hier wird auch die Edition EE Extreme Performance eingesetzt, weil nur in dieser Variante die RAC-Option inkludiert ist.

Eine frühere Lösung auf physischen Servern (Bare Metal) wird nicht mehr als PaaS angeboten. Dies war eine sehr kostspielige Lösung. Allerdings konnte man hier mehrere Datenbanken pro „Bare Metal Server" erstellen.

Als weiterer PaaS-Ansatz können Oracle-Datenbanken auch auf einer Exadata erstellt werden. Dazu muss die Exadata als Ressource vorher konfiguriert werden.

Es kann auch eine Exadata als „Exadata Cloud@Customer" im Kunden-Rechenzentrum betrieben werden. Hierbei ist die OCI-Virtualisierungs-Software auf der Exadata installiert und konfiguriert, doch die Daten liegen im Kunden-Rechenzentrum. Der Exadata-Service steht in der OCI als exklusiver Service zur Verfügung.

Autonomous Database

Neben dem Exadata-Service bietet die OCI auch die Autonomous Database als exklusiven Service an. Die autonome Datenbank von Oracle ist eine cloudbasierte Datenbank, die auf einer Exadata mit RAC läuft. Optional kann hier noch eine Active-Data-Guard-Datenbank konfiguriert werden. Die Autonomous Database ist deshalb mit einer Verfügbarkeit von 99,995 Prozent ausgelegt. Automatisiert werden Patches und Upgrades eingespielt. Laut Oracle ist sie selbstverwaltend, selbstsichernd und selbstreparierend.

Die Autonomous Database kann für eine Shared-Umgebung (ADB-S) oder Dedicated-Umgebung (ADB-D) ausgewählt werden.

Bei der „ADB-S" ist kein Zugriff auf den Server möglich. Hier kann die Datenbank nur über die OCI-Console angesehen werden. Folgende Workload-Typen sind wählbar: Data Warehouse, Transaction Processing, JSON, APEX.

Mit der „ADB-D" ist ebenfalls kein Zugriff auf Server möglich, allerdings kann man im Gegensatz zu der „ADB-S" das Patch Scheduling verändern. Als Workload-Typen sind Data Warehouse oder Transaction Processing auswählbar.

OCI-Lizenzmodelle

In der OCI wird ein physischer Core als OCPU (Oracle CPU) bezeichnet. Da in der Regel jedoch Hyperthreading aktiviert ist, entspricht eine OCPU zwei vCPUs (virtuellen CPUs). In der Preisliste werden sowohl die OCPUs als Unit Price und die vCPUs aufgeführt. Die Berechnung der Stundenpreise pro Server werden in der Regel bei allen Cloud-Anbietern in vCPUs angegeben.

Mit dem Lizenzmodell „Oracle-Lizenzierung bereits im Preis ein-

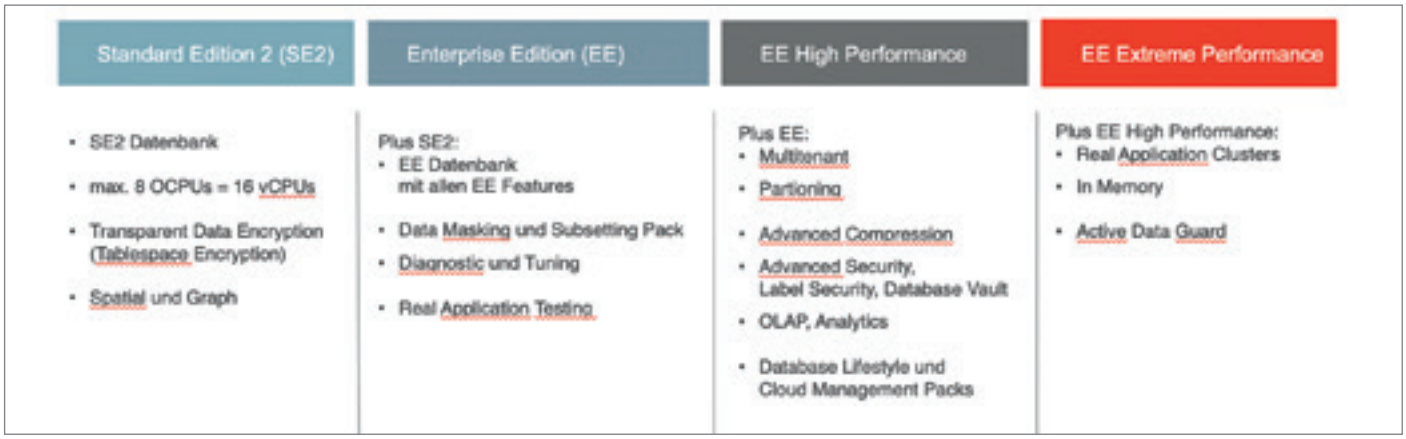


Abbildung 1: Oracle-Editionen in der OCI (Quelle: Oracle)

gerechnet" sind im OCPU- beziehungsweise vCPU-Preis bereits die Oracle-Lizenzkosten und die Wartungskosten inbegriffen. Die Bezahlung erfolgt über die tatsächliche Nutzung: Pay-as-you-go oder es kann mit Oracle eine feste Abnahmemenge, sogenannte Universal Credits beziehungsweise Annual Flex, ehemals als Monthly Flex bezeichnet, vereinbart werden.

Mit dem Lizenzmodell „BYOL" werden vorhandene On-Premises-Oracle-Lizenzen in die OCI eingebracht. Zu beachten ist, dass die Wartungskosten der Oracle-Lizenzen weiterhin bezahlt werden müssen. Dafür ist der OCPU-Preis günstiger als „Oracle-Lizenzierung bereits im Preis mit einberechnet". Werden Oracle-Lizenzen für EE „mitgebracht", kann mit PaaS (gilt nicht für IaaS) auch „Data Masking" und „Subsetting Pack", „Diagnostic und Tuning" und „Real Application Testing" kostenlos verwendet werden *(siehe Abbildung 1)*.

Die folgenden zwei Beispiele zeigen die Ermittlung der Oracle-Prozessor-Lizenzen, die in die OCI eingebracht werden müssen. Hier werden Hyperthreading und Intel-x86-Prozessoren angenommen.

- Edition EE: Server mit 2 OCPUs = 2 phys. Cores = 4 vCPUs = 1 Oracle-Prozessor-Lizenz
- Edition SE2: Server mit max. 8 OCPUs einsetzbar = 16 vCPUs = 2 Oracle-Prozessor-Lizenzen

Weitere Informationen zu „BYOL" können Sie hier nachlesen: *https://www.oracle.com/de/cloud/bring-your-own-license/faq/*

Die allgemeine Preisliste der OCI finden Sie hier: *https://www.oracle.com/de/cloud/price-list.html*

BYOL-Regeln für andere Cloud-Anbieter

Bevor ich zu den anderen Public-Cloud-Anbietern komme, stelle ich die von Oracle definierten „BYOL"-Regeln zum Nachteil für AWS und Azure vor. Warum die Google Cloud hier nicht vorkommt, wird im Abschnitt der Google Cloud klar werden. Das Lizenz-Dokument mit den „BYOL"-Regeln finden Sie hier: *https://www.oracle.com/assets/cloud-licensing-070579.pdf*

EE: Doppelte Anzahl der Lizenzen

Oracle verändert die Anzahl der mitzubringenden Oracle-Lizenzen für die Public-Cloud-Anbieter AWS und Azure. Anbei die relevanten Textzeilen aus dem Lizenz-Dokument:

„Amazon EC2 and RDS - count two vCPUs as equivalent to one Oracle Processor license if multi-threading of processor cores is enabled, and one vCPU as equivalent to one Oracle Processor license if multi-threading of processor cores is not enabled."

„Microsoft Azure – count two vCPUs as equivalent to one Oracle Processor license if multi-threading of processor cores is enabled, and one vCPU as equivalent to one Oracle Processor license if multi-threading of processor cores is not enabled. "

Hat der ausgewählte Server mit Hyperthreading 4 vCPUs (2 phy. Core), so müsste man 2 Prozessor-Lizenzen mitbringen. Ohne Hyperthreading entsprechen 2 vCPUs (2 phy. Cores) ebenfalls 2 Prozessor-Lizenzen.

Hier muss also im Gegensatz zur OCI die doppelte Anzahl von Oracle-Lizenzen mitgebracht werden.

Zum Vergleich: In der OCI gilt die bekannte Regel, zum Beispiel mit Hyperthreading:

2 phys. Cores (4 vCPUs) x 0,5 (Intel-Umrechnungsfaktor) = 1 Prozessor-Lizenz

SE2: Limitierung der vCPUs

Für den Einsatz der SE2 gelten für Amazon EC2 und RDS und Microsoft Azure spezielle Regeln. Auch hier die relevanten Textzeilen:

„Authorized Cloud Environment instances with four or fewer Amazon vCPUs, or four or fewer Azure vCPUs, are counted as 1 socket, which is considered equivalent to an Oracle processor license."

„Oracle Standard Edition One and Standard Edition 2 may only be licensed on Authorized Cloud Environment instances up to eight Amazon vCPUs or eight Azure vCPUs. If licensing Database Standard Edition 2 by Named User Plus metric, the minimums are 10 NUP licenses per 8 Amazon vCPUs or 8 Azure vCPUs. "

Beim Einsatz der SE2 bedeutet dies für einen Server mit Hyperthreading:

4 vCPUs = 2 phys. Cores = 1 Socket = 1 Prozessor-Lizenz.

Ohne Hyperthreading entsprechen 4 vCPUs = 4 phys. Cores = 1 Socket = 1 Prozessor-Lizenz.

Da man bei SE2 nur 2 Sockel pro Server maximal einsetzen darf, ist man hier auf einen Server mit maximal 8 vCPUs

beschränkt. Das heißt, die maximal mögliche Anzahl von 16 CPU-Threads für eine SE2-Datenbank ist damit gar nicht erreichbar.

Zum Vergleich: In der OCI können beim Einsatz der SE2 auf dem Server maximal 16 vCPUs = 8 OCPUs eingesetzt werden.

Amazon Web Services (AWS)

Bei AWS können für Oracle-Datenbanken wie in der OCI auch IaaS und PaaS ausgewählt werden.

IaaS

Bei AWS wird IaaS als Amazon EC2 bezeichnet. Es können physische oder virtuelle Server konfiguriert werden. Für den Betrieb der Oracle-Datenbanken ist hier nur das Lizenzmodell „BYOL" möglich. Hier gelten die vorher genannten nachteiligen „BYOL"-Regeln. Die Oracle-Installation und das Erstellen der Datenbanken müssen hier selbst erfolgen.

PaaS

Das PaaS-Angebot von AWS wird mit Amazon RDS (Relational Database Service) bezeichnet.

Wird das Lizenzmodell „Oracle-Lizenzierung bereits im Preis eingerechnet" ausgewählt, ist das Angebot sehr eingeschränkt:

- Es wird nur die Edition SE2 in den Versionen 19c, 21c angeboten, eine Übersicht finden Sie hier: *https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Oracle.Concepts.Licensing.html*
- Mehrere Instance-Typen können ausgewählt werden. Die Instance-Typen finden Sie hier: *https://aws.amazon.com/de/rds/instance-types/*

Mit dem Lizenzmodell „BYOL" gelten hier die vorher genannten nachteiligen „BYOL"-Lizenzregeln.

- Es können die Editionen SE2 und EE gewählt werden.
- Die Oracle-Lizenzen für Optionen oder Management-Packs müssen im Gegensatz zur OCI auch bereits vorhanden sein, wenn sie verwendet werden.

Die Preise von Amazon RDS können hier berechnet werden: *https://aws.amazon.com/de/rds/oracle/pricing/*

In den folgenden Beispielen zeige ich die Anzahl der mitzubringenden Oracle-Lizenzen und somit nachteiligen „BYOL"-Regeln für Amazon RDS:

Enterprise Edition

- Mit Hyperthreading: 2 vCPUs = 1 phys. Core = 1 Prozessor-Lizenz
- Ohne Hyperthreading, z. B. EC2: T2 Instance-Type : 1 vCPU = 1 phys. Core = 1 Prozessor-Lizenz

Standard Edition 2

- 4 vCPUs = 1 Socket = 1 Prozessor-Lizenz
- Mit Hyperthreading max. 8 vCPUs = 4 phys. Cores = 2 Prozessor-Lizenzen (max. 8 CPU-Threads)
- Ohne Hyperthreading max. 8 vCPUs = 8 phys. Cores = 2 Prozessor-Lizenzen (max. 8 CPU-Threads)

Zum Vergleich „BYOL" in der OCI: Hier gibt es nur „Server" mit Hyperthreading, Intel-x86

Enterprise Edition

- 4 vCPUs = 2 OCPUs = 2 phys. Cores = 1 Prozessor-Lizenz

Standard Edition 2

- SE2: max. 16 vCPUs = 8 OCPUs = 8 phys. Cores = 2 Prozessor-Lizenzen (16 CPU-Threads möglich)

Microsoft Azure

Microsoft und Oracle haben schon längere Zeit eine Kooperation und stellen zwischen ihren Rechenzentren eine Cloud-übergreifende Konnektivität mit niedriger Latenz und hohem Durchsatz bereit. Die Idee dahinter ist, dass die Applikationen in Azure liegen können und die Oracle-Datenbanken in der OCI. Trotzdem ist die Anbindung zwischen den Applikationen und den Oracle-Datenbanken sehr schnell. Seit Juli 2022 werden sogar „Oracle Database Services" in Azure angeboten, um aus Azure heraus Oracle-Datenbanken direkt in der OCI zu erstellen.

Azure bietet für Oracle-Datenbanken keinen PaaS an. Es können in Azure nur im IaaS physische oder virtuelle Server für Oracle-Datenbanken bereitgestellt werden. Zusätzlich werden hier aber Oracle Linux Images angeboten, was die Installation von Oracle erleichtert:

- Oracle Database 12.2 und 18.3 Enterprise Edition
- Oracle Database 12.2 und 18.3 Standard Edition
- Oracle Database 19.3

Eine Übersicht finden Sie hier: *https://docs.microsoft.com/en-us/azure/virtual-machines/workloads/oracle/oracle-overview*

Als Lizenzmodell kann nur „BYOL" gewählt werden. Hier gelten die gleichen nachteiligen Lizenzbedingungen wie bei AWS.

Google Cloud

Die Google Cloud Platform (GCP) bietet im Gegensatz zu AWS mit Amazon RDS oder Oracle mit der OCI keinen PaaS für Oracle an. Dies ist auf eine Oracle-Google-Lizenzbeschränkung im Zusammenhang mit dem Compute Engine Hypervisor zurückzuführen.

Die Google Cloud bietet für den Betrieb von Oracle-Datenbanken zwei Alternativen/Optionen an.

Option 1: Bereitstellen von Oracle mithilfe von GCP-Marketplace-Oracle-Partnern

Eine Oracle-Datenbank kann vom Google Cloud Marketplace bereitgestellt werden; die Dienste eines Google-Partners, der auch als Managed Service Provider (MSP) bezeichnet wird, können erworben werden. Mit dieser Option platzieren Sie Ihre Oracle-Datenbank jedoch tatsächlich in einer Co-Location-Einrichtung, die sich neben der GCP befindet. Die Oracle-Lizenzierung erfolgt zu den Bedingungen der MSPs.

Option 2: Bereitstellen von Oracle mit der Bare-Metal-Lösung von Google

Physische Bare-Metal-Server stehen in einem isolierten lokalen Rechenzentrum. Es besteht kein direkter Zugriff

auf Google-Cloud-Dienste und das Internet. Für die Netzwerkkonnektivität muss mit einem Drittanbieter zusammengearbeitet werden.

Hier ist nur eine „BYOL"-Oracle-Lizenzierung möglich.

Eine Übersicht der Google-Cloud-Modelle finden Sie hier:

https://cloud.netapp.com/blog/gcp-cvo-blg-oracle-on-google-cloud-challenges-and-solutions

Fazit

Die OCI bietet gegenüber anderen Cloud-Anbietern viele Vorteile. So können im Cloud-Modell PaaS mit dem Lizenzmodell „Oracle-Lizenzierung im Preis eingerechnet" alle aktuellen Versionen und Editionen eingesetzt werden. Auch werden Optionen und Management-Packs in den Editionen angeboten, die auf On-Premises oder bei anderen Cloud-Anbietern separat lizenziert werden müssen. Mit der Autonomous Database und dem Exadata-Service stehen exklusive Services in der OCI bereit.

Azure bietet für den Betrieb der Oracle-Datenbanken nur den IaaS an und keinen PaaS.

Bei den Cloud-Anbietern AWS und Azure kann in der Regel nur das Lizenz-Modell „BYOL" ausgewählt werden. Eine Ausnahme bietet hier die AWS, allerdings mit sehr eingeschränkten Oracle-Versionen und -Editionen. Beim Lizenz-Modell „BYOL" müssen bei AWS und Azure doppelt so viele Oracle-Lizenzen eingebracht werden wie in der OCI. Die SE2 kann bei AWS und Azure mit dem Lizenz-Modell „BYOL" nur mit maximal 8 vCPUs pro Server eingeschränkt betrieben werden.

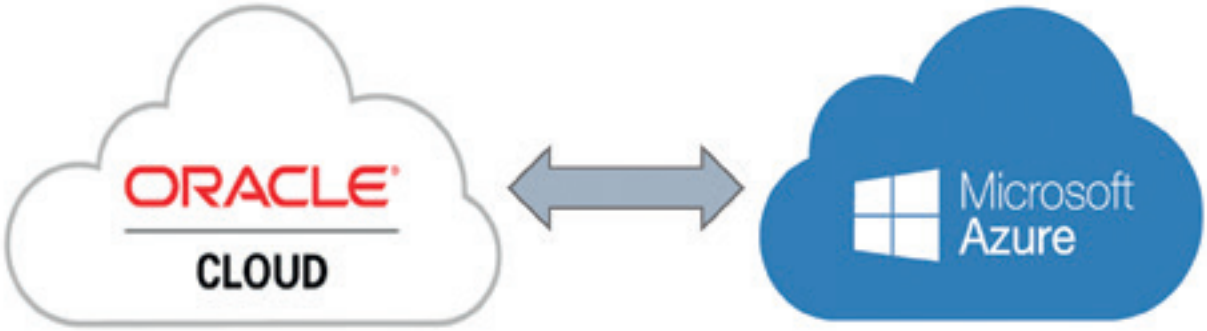
Die Google Cloud ist für die Verwendung von Oracle-Produkten nur mit Marketplace-Oracle-Partnern und auf Bare-Metal-Lösungen eingeschränkt nutzbar.

Haben Sie sich schon für die Azure-Public-Cloud entschieden, dann ist eine ideale Kombination, die Azure-Cloud für Ihre Applikationen und die OCI für Ihre Oracle-Datenbanken zu verwenden. Beide Cloud-Anbieter haben untereinander eine „schnelle" Anbindung und in der Azure-Cloud können Sie mit den „Oracle Database Services" Oracle-Datenbanken direkt in der OCI erstellen. Eine „schnelle" Anbindung zwischen AWS bezieungsweise Google Cloud mit der OCI ist jedoch nicht absehbar.

Über den Autor

Michael Skowasch ist Principal Consultant bei der Ordix AG. Seit über 20 Jahren berät er Kunden in Oracle-Themen, speziell in den Bereichen Hochverfügbarkeit, Lizenz- und Cloudberatung. Zusätzlich zu seiner Beratertätigkeit ist Michael Skowasch auch als Referent im Ordix-Seminarzentrum im Einsatz.

Michael Skowasch
msk@ordix.de



Multicloud 2.0: Oracle Database Service for Azure (ODSA)

Kai-Uwe Fischer

„Garden Walls Tumbling Down“, mit diesen Worten hat Larry Ellison, am 19. Oktober 2022, auf der letztjährigen Oracle Cloud World (OCW) in Las Vegas eine engere Zusammenarbeit mit anderen Cloud-Anbietern angekündigt, ein sogenanntes Internet der Clouds. Alle wichtigen Cloudanbieter sollten miteinander verbunden werden, damit die Kunden die Möglichkeit haben, entsprechende Services über Cloudgrenzen hinweg miteinander zu verbinden. So ist beispielsweise die MySQL-HeatWave-Datenbank nicht nur in OCI, sondern bereits auch in AWS und Azure verfügbar. Darüber hinaus ist seit Juni 2022 der Oracle Database Service for Azure (ODSA) bereitgestellt. Für dieses Jahr hat Oracle einen Oracle Database Service for AWS (April) und einen Oracle Database Service for GCP (November) angekündigt. In diesem Artikel möchte ich den Database Service for Azure (ODSA) vorstellen und die Unterschiede zum OCI-Azure-Interconnect aufzeigen.

Wie Unternehmen von den Vorteilen der Cloud/Multi Cloud profitieren

Skalierbarkeit auf Abruf, flexible Abrechnungsmodelle, bessere Performance und keine Investitionskosten – das sind nur einige der Vorteile für Unternehmen, die sich für eine Public Cloud entscheiden.

Kann die eine Public Cloud alle Kundenanforderungen erfüllen? Wenn nicht, bietet sich der folgende Ansatz an. Warum nicht mehrere Clouds zu einer Multi-Cloud-Lösung verbinden. Genau diesen Ansatz verfolgen Oracle und Microsoft, indem sie die Oracle Cloud Infrastructure (OCI) mit der Microsoft Azure Cloud verbinden.

So kann der Kunde von den Vorteilen des Betriebes seiner Oracle-Datenbank in OCI profitieren und gleichzeitig weiterhin mit Tools wie PowerBI Datenbankabfragen aus Azure heraus gegen die Oracle-Datenbank ausführen.

Einsparpotenzial durch Multi Cloud

Mit dem Multi-Cloud-Ansatz haben Kunden die Möglichkeit, die Kosten für den Datenbankbetrieb zu reduzieren, wenn sie diese in OCI anstatt zum Beispiel in Azure betreiben. Werden anstelle von „Bring your own License“ (BYOL) die Universal Credits verwendet, können die Betriebskosten deutlich gesenkt werden, wenn bei Nichtbenutzung die Datenbank-

Services gestoppt werden. Dies ist vor allem für Test- sowie Development-Datenbanken interessant, die nicht 24/7 zur Verfügung stehen müssen. Des Weiteren besteht durch Skalierung der Datenbank-OCPUs (Scale Up/Scale Down) die Möglichkeit, flexibel auf Anwendungsanforderungen zu reagieren. Im Falle von Scale Down können gleichzeitig auch die Datenbankkosten reduziert werden.

Security First, unter anderem durch den Einsatz von Transparent Data Encryption (TDE)

Die Oracle Cloud Infrastructure (OCI) erfüllt, durch den Defense-in-Depth-Ansatz, höchste Sicherheitsansprüche. Dies beginnt bei der OCI Tenancy und setzt sich über Infrastruktur, Netzwerk, Compute bis hin zur Datenbank fort.

So wird die Oracle-Datenbank „out of the box", sowohl in der Standard- wie auch in der Enterprise Edition, per Transparent Data Encryption (TDE) verschlüsselt. Um TDE bei anderen Cloudanbietern nutzen zu können, muss dort, anders als in der OCI, die Advanced-Security-Option für die Oracle Database Enterprise Edition erworben werden. Eine TDE-Verschlüsselung der Standard Edition ist hier nicht möglich.

Weniger Administrationsaufwand durch Cloud Automation

OCI stellt dem Oracle-Datenbank-Administrator (DBA) mittels Oracle Cloud Web-based UI, REST APIs, SDK, CLI oder Terraform ein Cloud Tooling zur Verfügung, mit dem er auf Knopfdruck folgende Tätigkeiten durchführen kann:

- Patchen/Upgrade der Grid-Infrastruktur und Oracle-Datenbank
- Aufbau einer Standby-Datenbank mittels Data Guard
- Backup/Restore von Oracle-Datenbanken
- Skalierung der OCPUs eines DB-Systems
- Skalierung Storage (nur Scale Up) eines DB-Systems
- Aufbau 2-Knoten Real Application Cluster (RAC)

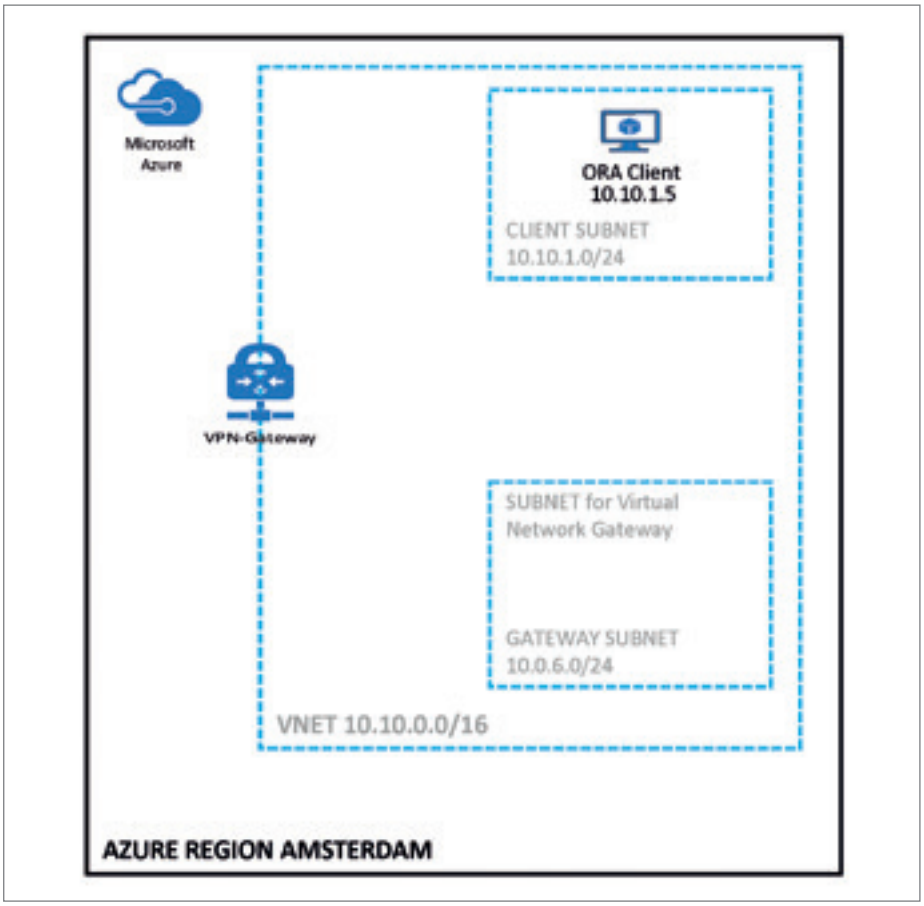


Abbildung 1: Azure VNET (© Kai-Uwe Fischer)

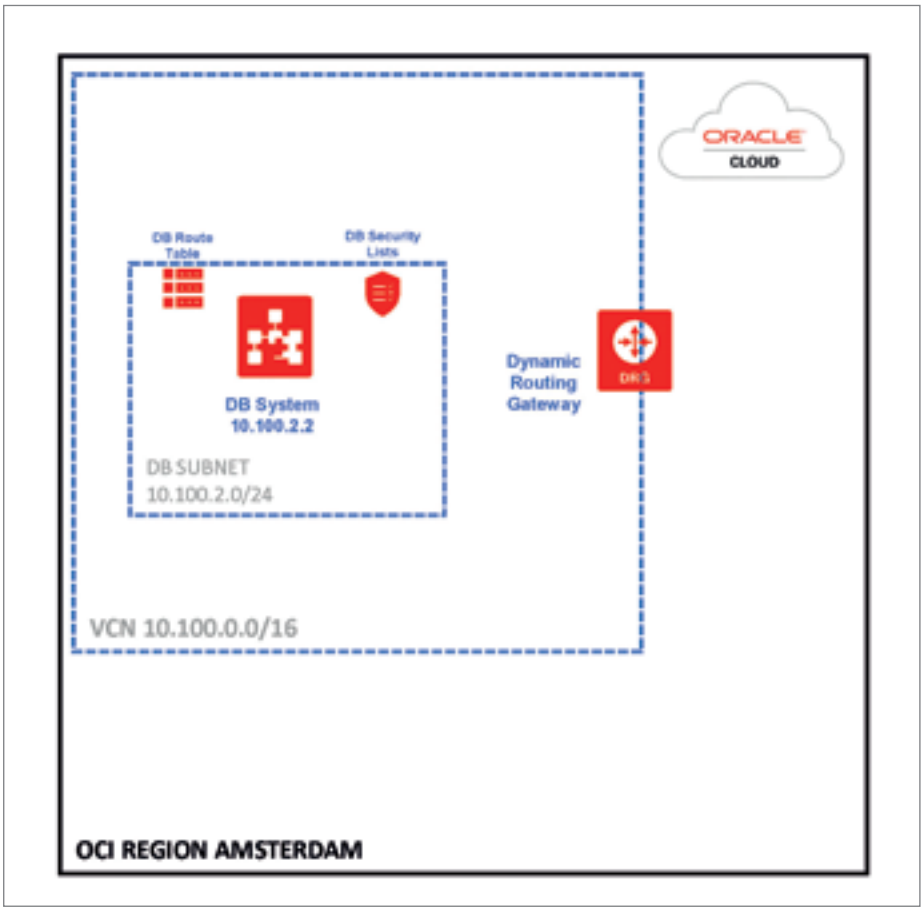


Abbildung 2: OCI VCN (© Kai-Uwe Fischer)

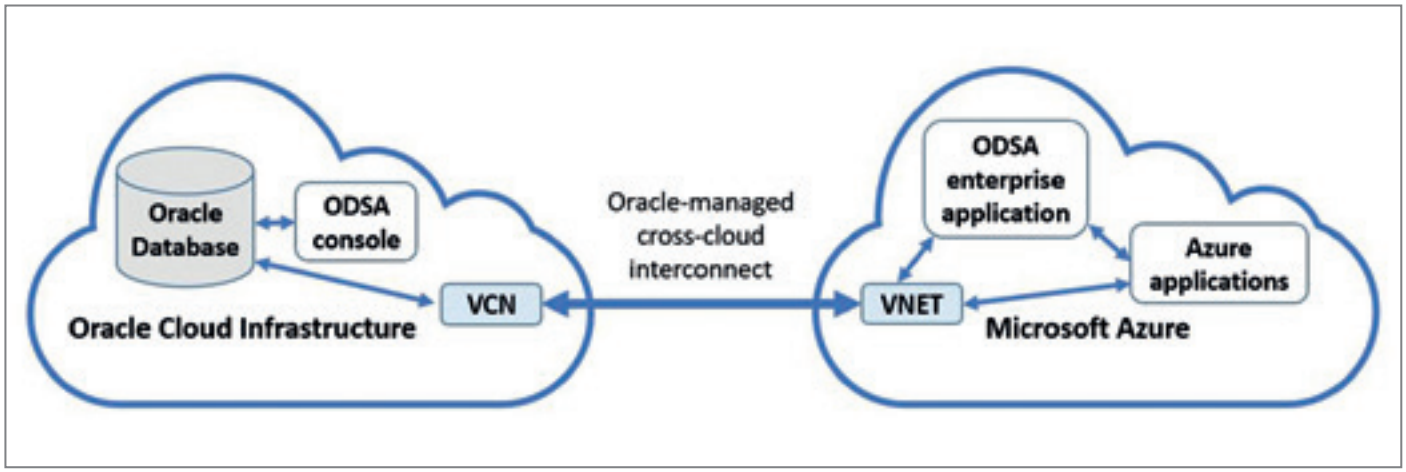


Abbildung 3: ODSA-Struktur (© Kai-Uwe Fischer)

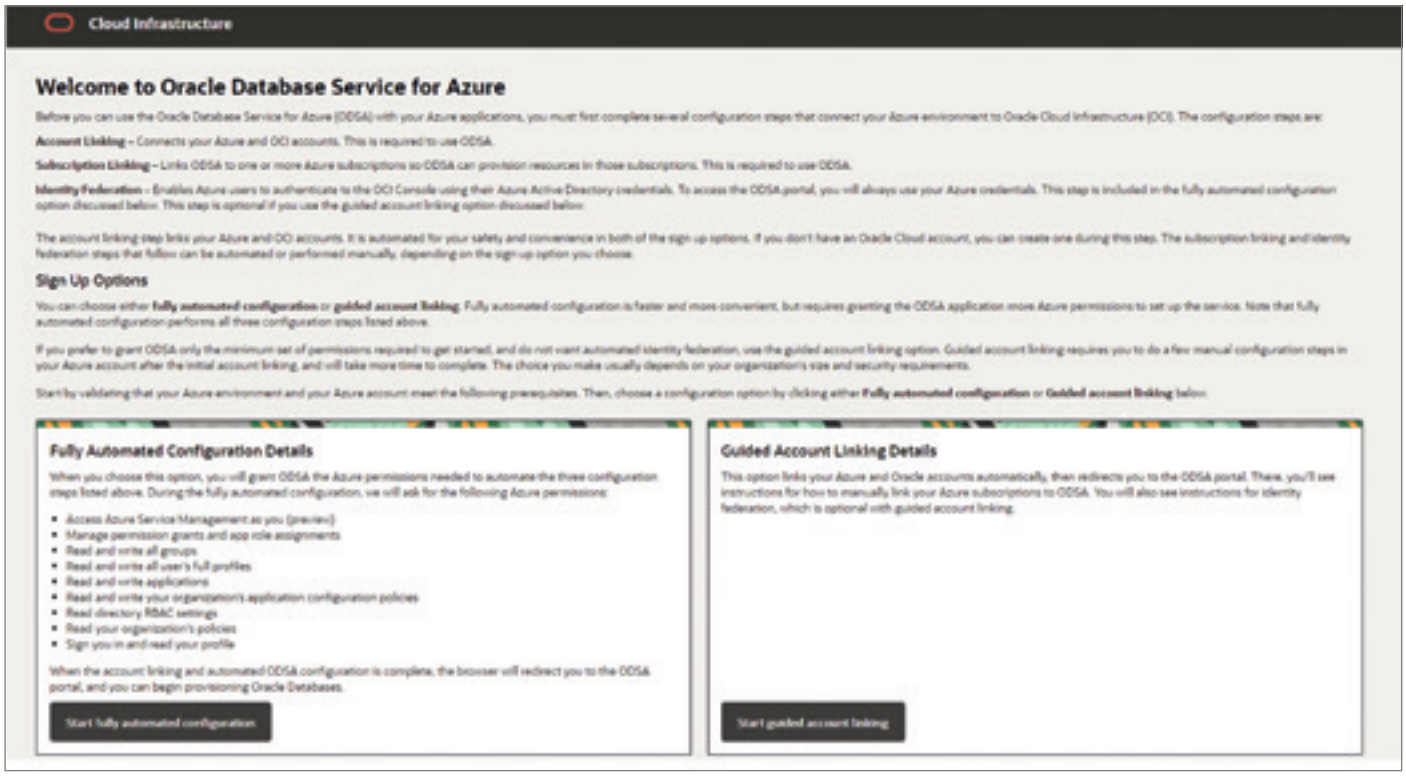


Abbildung 4: ODSA Console (© Kai-Uwe Fischer)

OCI-Azure-Interconnect

Seit dem 5. Juni 2019 existiert eine Partnerschaft zwischen Oracle und Microsoft, die es Unternehmen ermöglicht, geschäftskritische Workloads über Oracle-Cloud- und -Azure-Grenzen hinweg zu migrieren und auszuführen.

Das „Herzstück" ist dabei der OCI-Azure-Interconnect, der eine private, direkte, schnelle und zuverlässige Verbindung zwischen beiden Clouds ermöglicht. Ein 3rd-party-Network-Service-Provider wird hierfür nicht benötigt.

Der Interconnect steht mittlerweile in 12 der insgesamt 41 OCI-Regionen zur Verfügung. Stand Januar 2023 sind dies die folgenden Lokationen:

- Frankfurt
- Amsterdam
- London
- Johannesburg
- Ashburn
- Phoenix
- San Jose
- Toronto
- Vinhedo (Brasilien)
- Singapore
- Seoul
- Tokio

Der Aufbau des OCI-Azure-Interconnect erfolgt in wenigen einfachen Schritten, die ich in der Red-Stack-Magazin-Ausgabe 3 von Mai 2022 detailliert beschrieben habe.

Die wichtigsten Schritte nochmals kurz zusammengefasst:

- Konfiguration einer ExpressRoute in Azure, dabei wird Oracle Cloud FastConnect als Provider ausgewählt. Zusätzlich müssen die Peering Location und die Bandweite festgelegt werden.

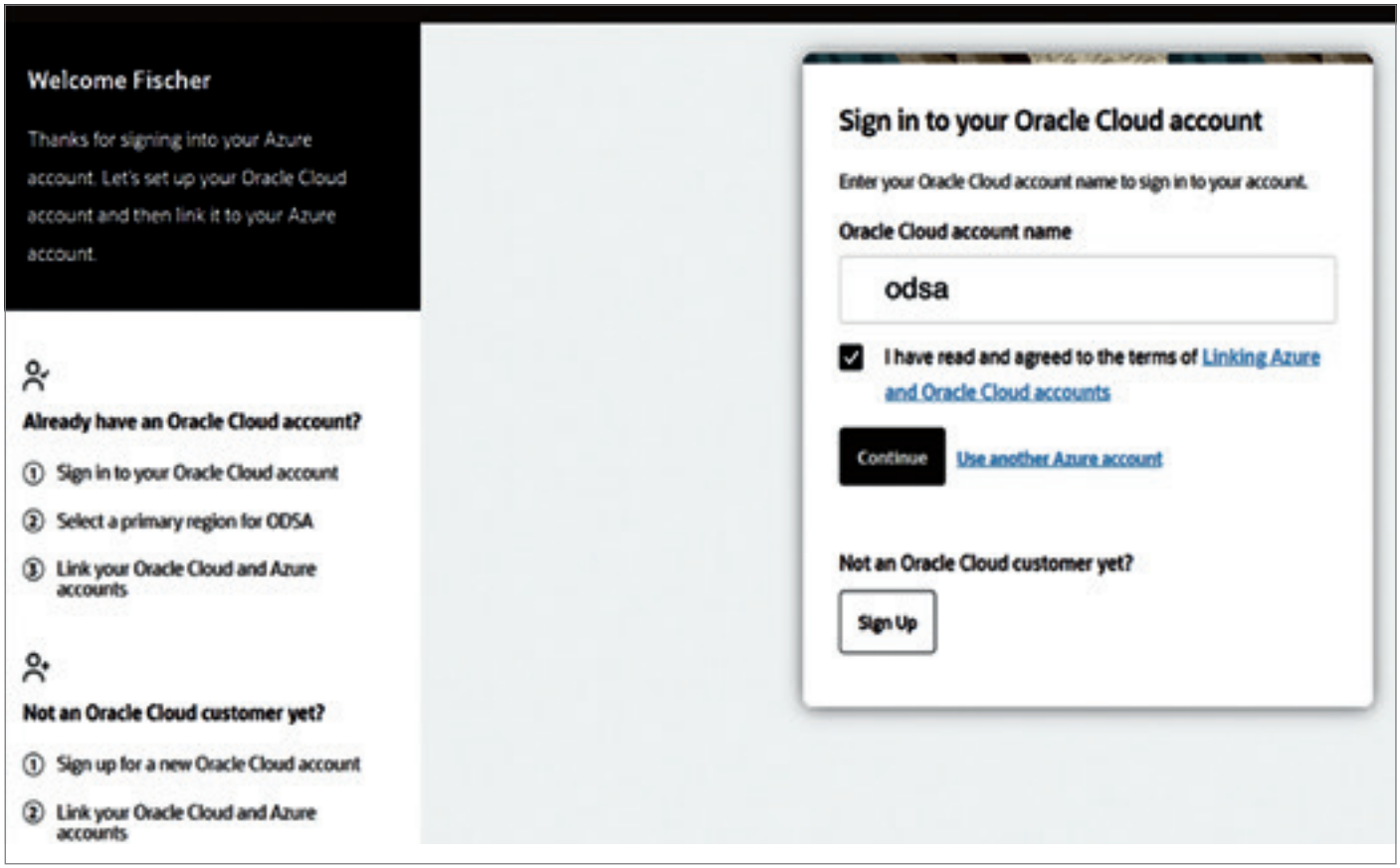


Abbildung 5: OCI-Anmeldung (© Kai-Uwe Fischer)

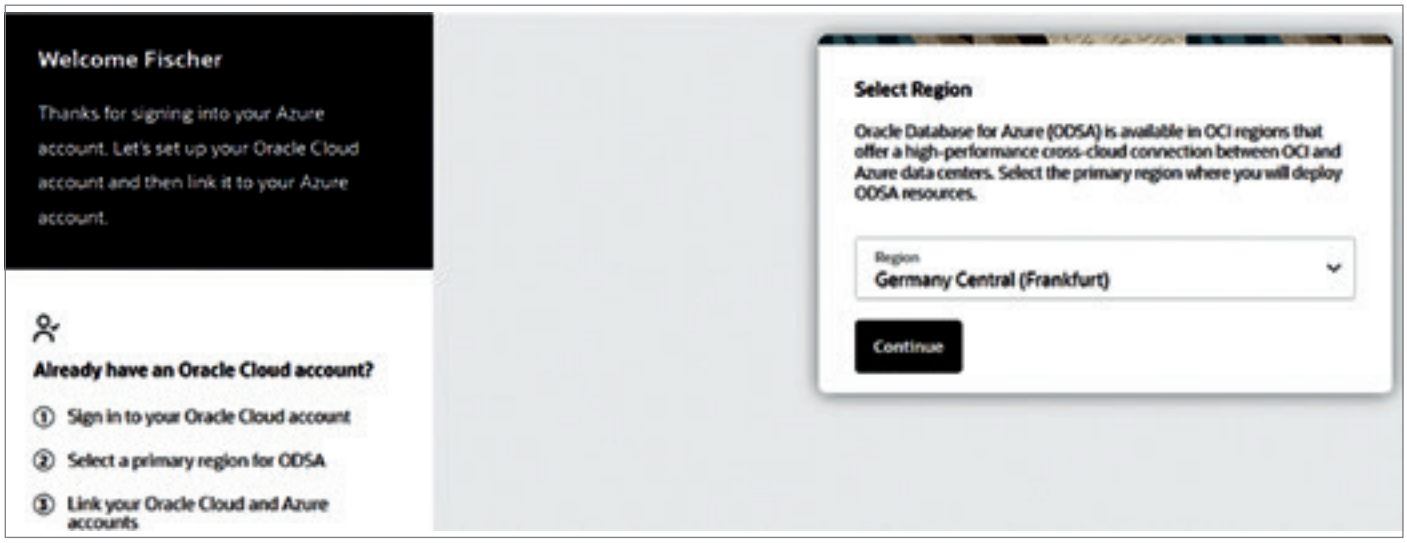


Abbildung 6: Auswahl Region (© Kai-Uwe Fischer)

- Nachdem die Express Route angelegt wurde, muss sich der Nutzer aus der Übersicht den Service Key kopieren.
- Im nächsten Schritt wechselt der Nutzer auf die OCI-Console, erzeugt dort eine FastConnect-Verbindung und wählt als Partner Microsoft Azure ExpressRoute aus. Zudem kopiert er den Azure Service Key in das Feld PARTNER SERVICE KEY.
- Im letzten Schritt muss der Nutzer einen Link zwischen dem Azure Virtual Network und dem ExpressRoute Circuit herstellen.

Der Aufbau des OCI-Azure-Interconnect ist eine Sache, aber was sind die Voraussetzungen und welche Vor- und Nacharbeiten fallen an, wenn ich zwei Clouds (OCI und Azure) miteinander verbinden möchte?

Auf der Microsoft-Seite werden ein beliebiges Azure Virtual Network (VNET) mit mindestens einem CLIENT-Subnetz und einem Subnetz für das Virtual Network Gateway sowie ein VPN-Gateway benötigt *(siehe Abbildung 1)*.

Auf der Oracle-Seite sieht es ähnlich aus, hier werden ein Virtual Cloud Network (VCN) mit mindestens einem privaten Datenbank-Subnetz sowie ein

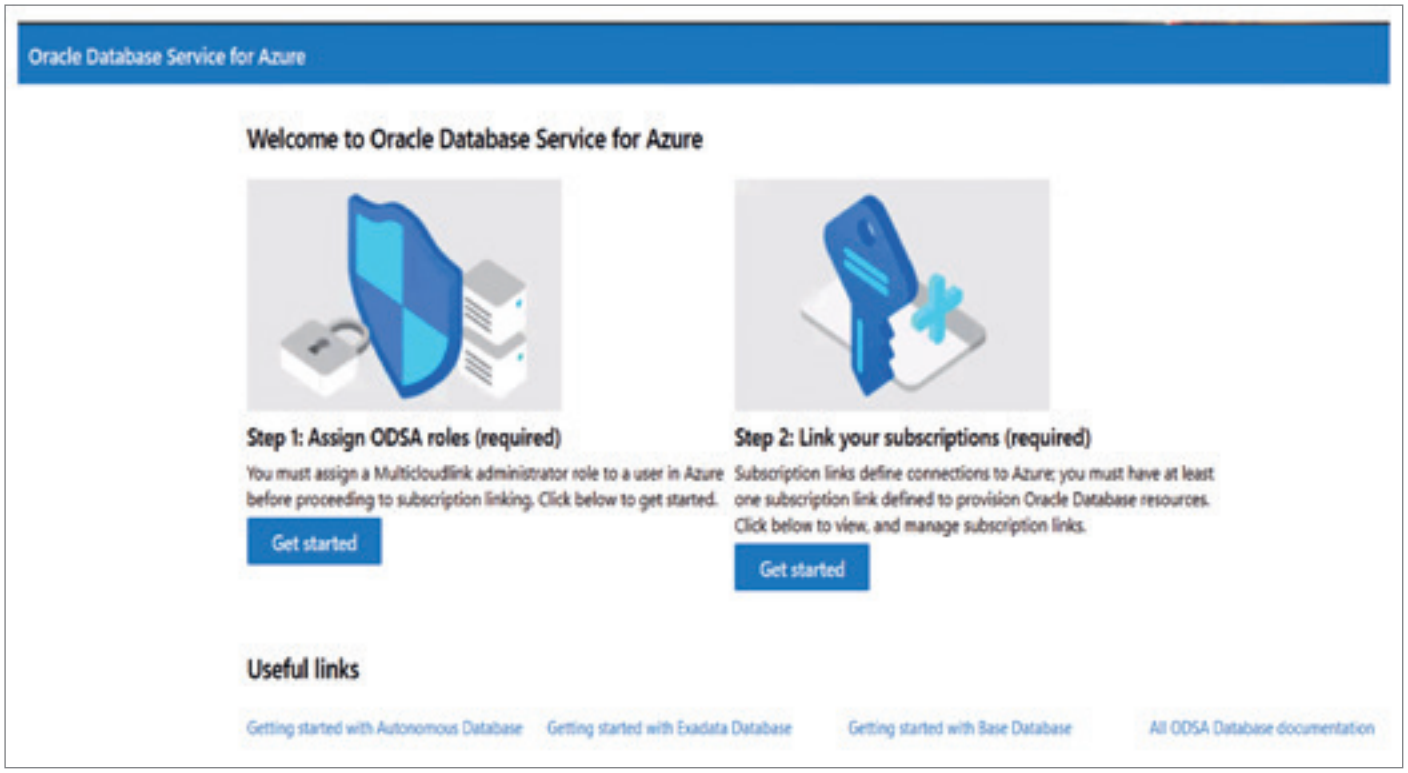


Abbildung 7: ODSA Step 1 (© Kai-Uwe Fischer)

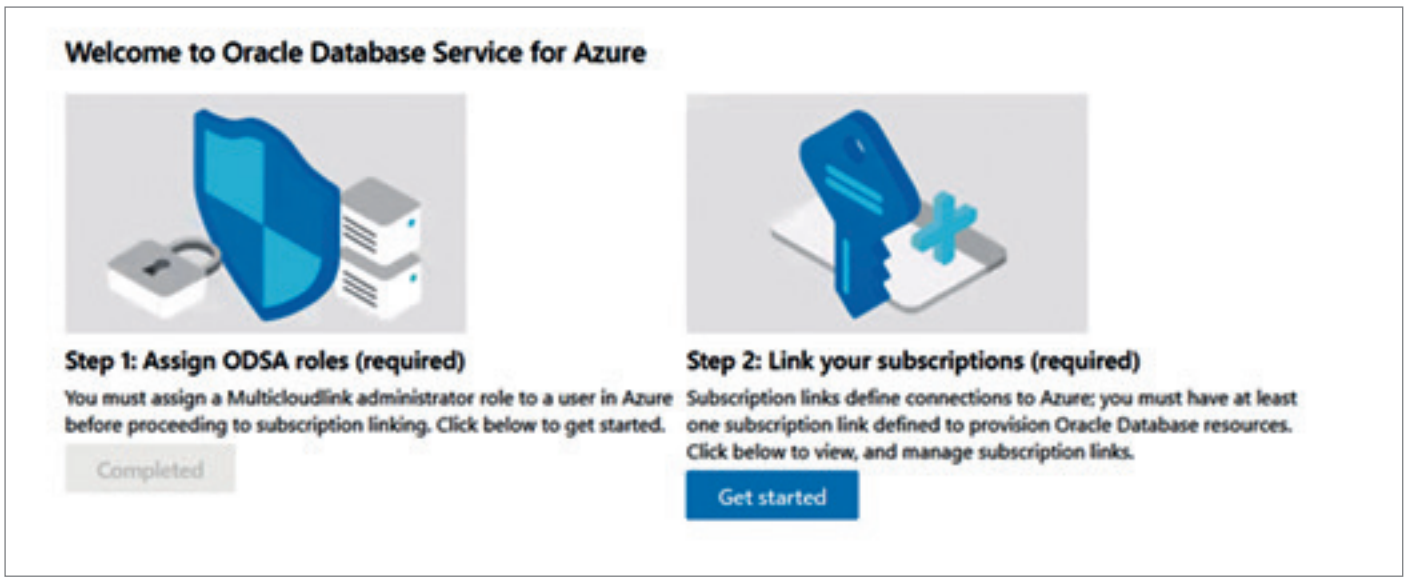


Abbildung 8: ODSA Step 2 (© Kai-Uwe Fischer)

Abbildung 9: Subscription Management (© Kai-Uwe Fischer)

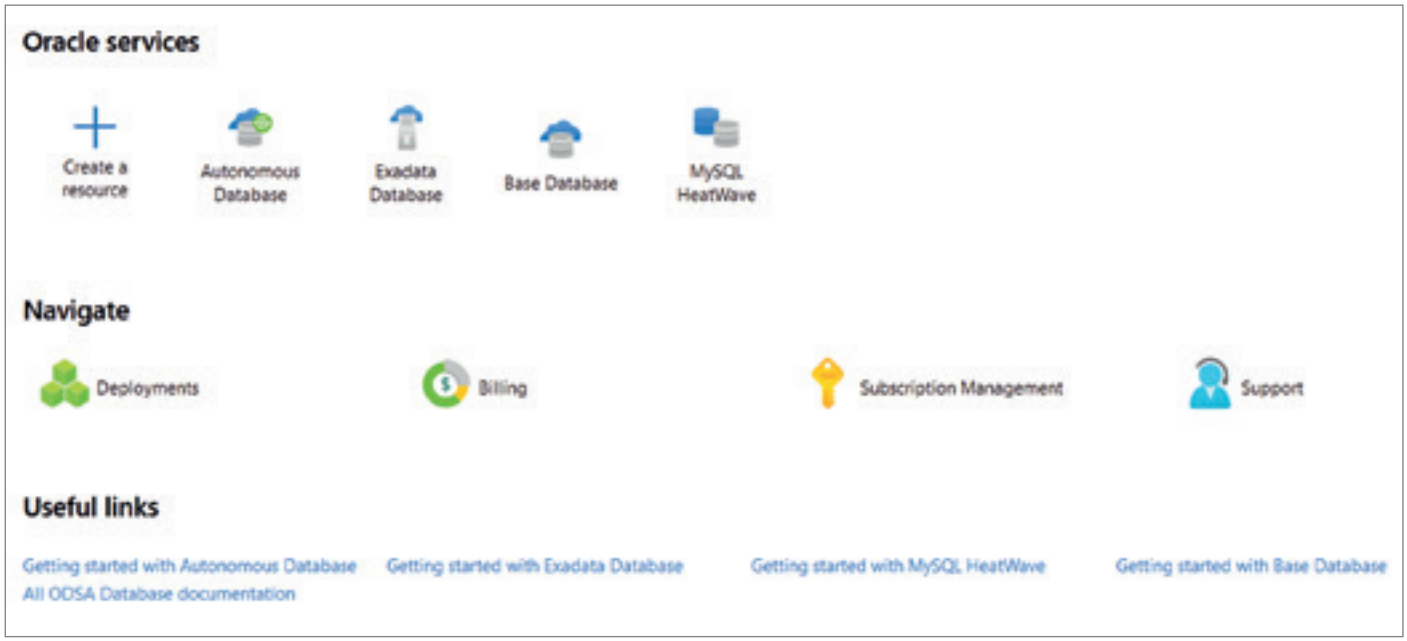


Abbildung 10: Oracle Services (© Kai-Uwe Fischer)

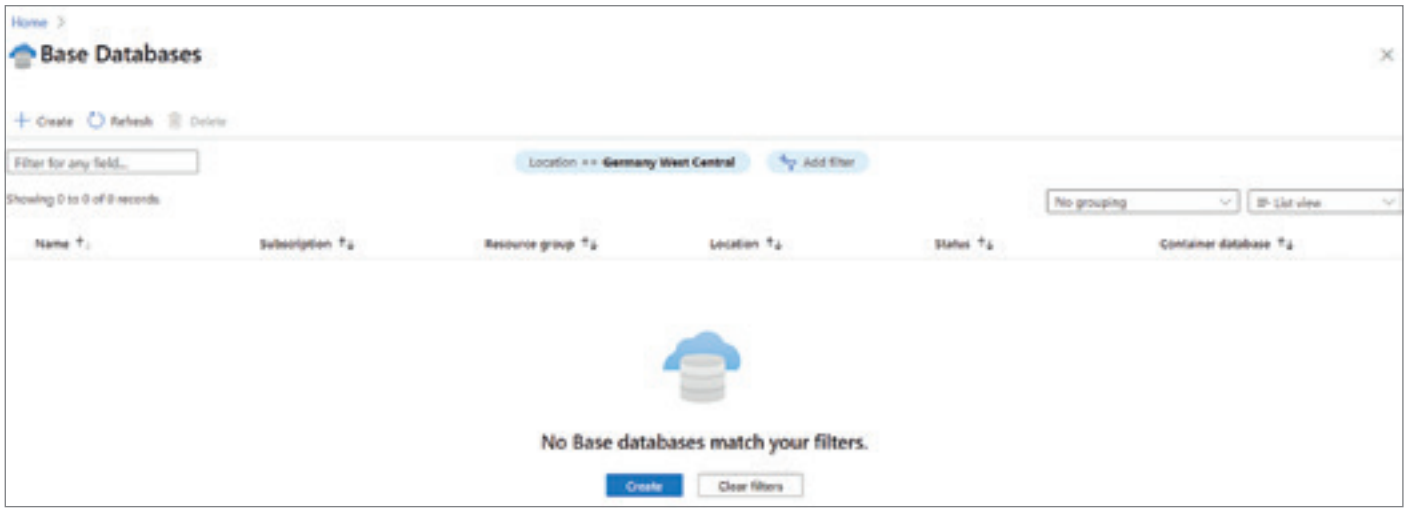


Abbildung 11: Create (© Kai-Uwe Fischer)

Abbildung 12: Basics (© Kai-Uwe Fischer)

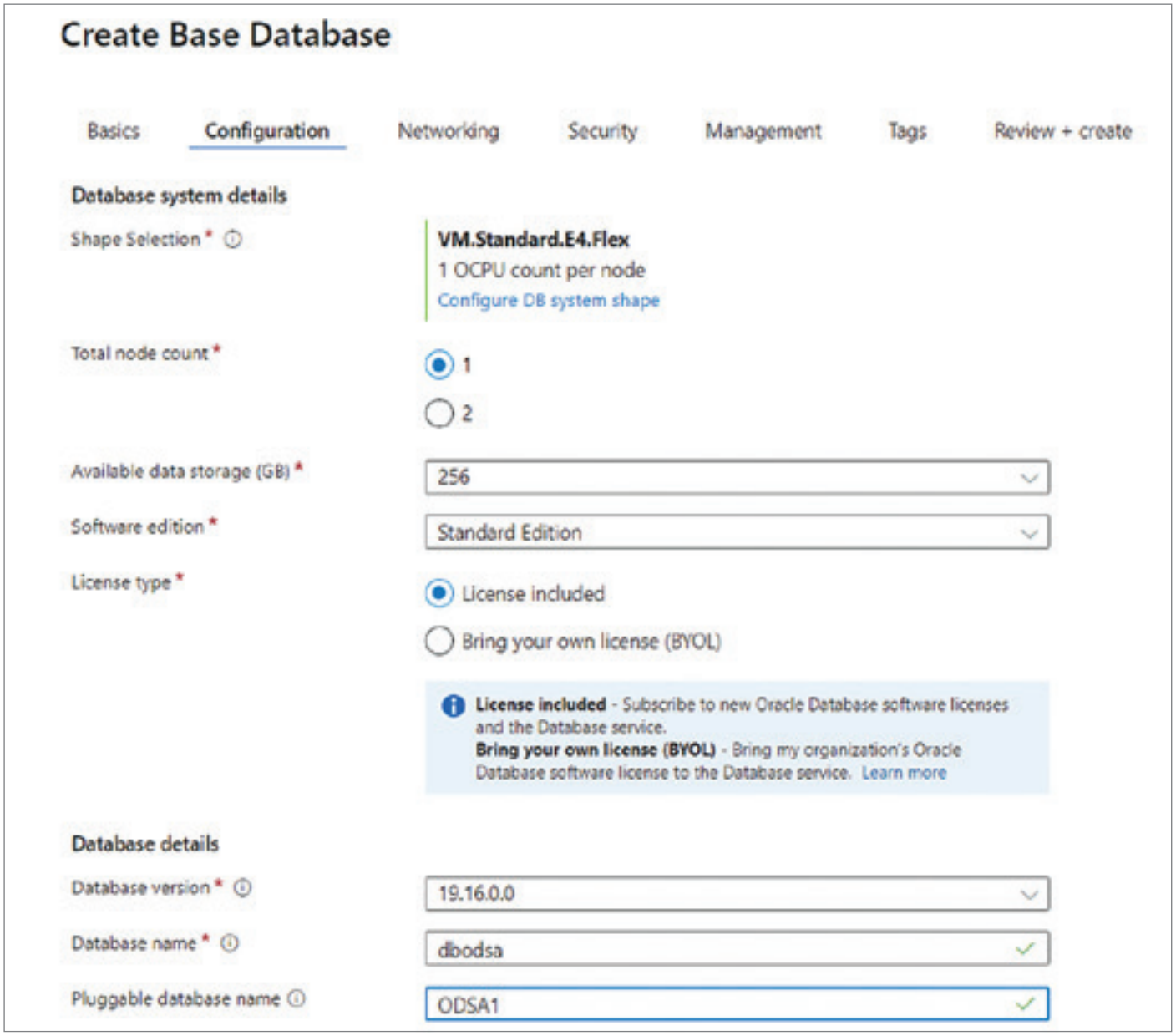


Abbildung 13: Configuration (© Kai-Uwe Fischer)

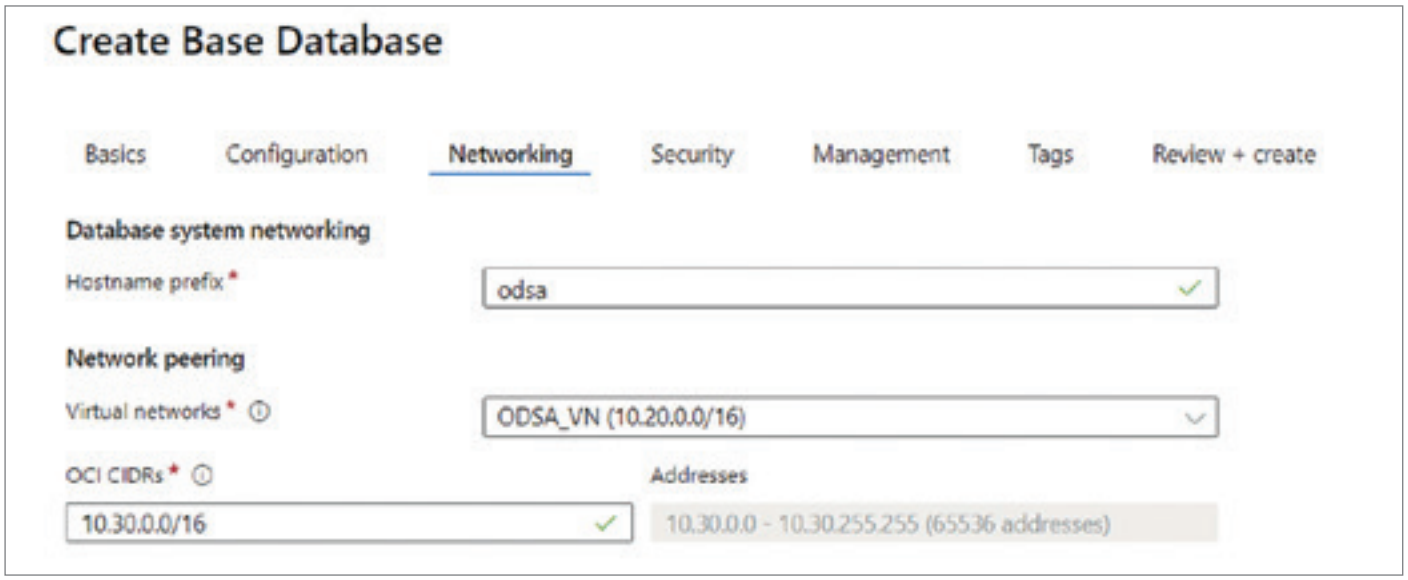


Abbildung 14: Networking (© Kai-Uwe Fischer)

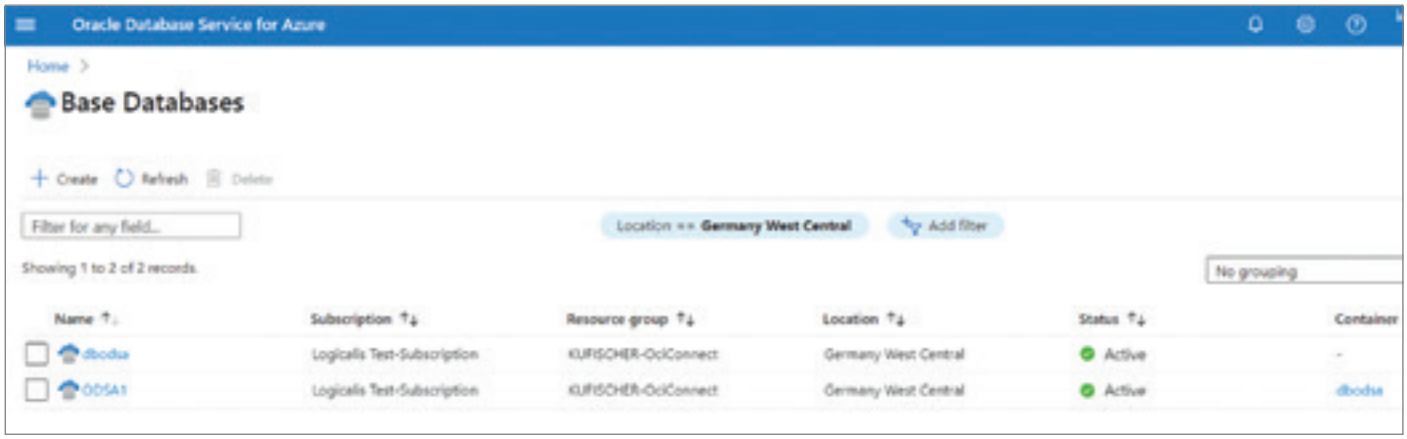


Abbildung 15: Auflistung der Base Databases (© Kai-Uwe Fischer)

Abbildung 16: TNS Names Connection String (© Kai-Uwe Fischer)

Key capabilities	Oracle Database Service for Azure Portal	OCI Console
Manual Backup	✓	✓
Automated Backup	✓	✓
Restore to existing database	✓	✓
License Management	✓	✓
RAC	✓	✓
Service Requests	✓	✓
Data Guard	Coming soon	✓
Scaling Infrastructure	Coming soon	✓
Autoscaling	Coming soon	✓

Abbildung 17: Vergleich ODSCA / OCI Console (© Kai-Uwe Fischer)

Dynamic Routing Gateway benötigt *(siehe Abbildung 2)*.

Eine wichtige Sache gilt es zu beachten: Die IP-Adressen der beiden virtuellen Netzwerke dürfen sich nicht überlappen. Beispielsweise kann für das Azure-Netzwerk der CIDR-Block 10.10.0.0/16 und für Oracle der CIDR-Block 10.100.0.0/16 verwendet werden.

Damit, nach der vollständigen Einrichtung des Interconnect, auch eine Kommunikation zwischen den beiden Cloudwelten stattfinden kann, müssen folgende Nacharbeiten auf beiden Cloud-Seiten ausgeführt werden:

- Anpassen der Route Tables
- Freischalten der notwendigen Ports in den Network Security Groups beziehungsweise Security Lists

Für Kunden, die nur eine geringe Anzahl von Oracle-Datenbanken im Einsatz haben, mag dieser Multicloud-Ansatz zu steinig sein. Bedeutet dies doch für den Kunden, dass er zwei komplette Cloud-Umgebungen aufbauen und verwalten muss. Gerade bei kleineren Unternehmen ist jedoch dafür selten genügend Personal und/oder Know-how vorhanden.

Genau für diese Kunden gibt es jetzt den Oracle Database Service for Microsoft Azure (ODSA).

Oracle Database Service for Microsoft Azure (ODSA)

ODSA wurde am 20. Juli 2022 mit den folgenden Features angekündigt:

- Oracle Database Service for Microsoft Azure (ODSA) ermöglicht Ihnen die einfache Integration des Datenbankdienstes von Oracle Cloud Infrastructure in Ihre Azure-Cloud-Umgebung.
- ODSA verwendet einen servicebasierten Ansatz und ist eine Alternative zur manuellen Erstellung komplexer Cross-Cloud-Bereitstellungen für Ihren Application Stack.

Bei ODSA handelt es sich um eine Oberfläche von Oracle mit dem Look-and-Feel von Azure, mit der Oracle-Datenbanken und die dazugehörigen OCI-Komponenten (Compartment, Virtual Cloud Network (VCN), Subnet, Route Table und Security List) „out of the box“ erzeugt werden. Die Anmeldung an diese Oberfläche erfolgt mit den Azure Credentials.

Aktuell (Stand Januar 2023) können folgende Oracle-Datenbank-Systeme über diese Oberfläche erstellt werden:

- Autonomous Database on shared Exadata Infrastructure
- Oracle Exadata Database
- Base Database incl. RAC
- MySQL Heatwave

Des Weiteren befindet sich Autonomous Database (Dedicated) auf der Roadmap.

Der Oracle Database Service for Azure nutzt ebenfalls den OCI-Azure-Interconnect, allerdings ist dieser nicht dediziert, sondern ein Multitenant Interconnect ohne SLAs und wird von Oracle verwaltet. Die Nutzung des Interconnect ist dafür kostenfrei.

Für den Kunden ist der Interconnect völlig transparent, er muss sich also nicht um den Aufbau oder die Konfiguration kümmern.

Um mit ODSA zu starten, benötigt der Kunde einen Azure Account und ein entsprechendes VNET in seiner Azure-Umgebung, das mit dem VCN auf Oracle- Seite gekoppelt werden soll *(siehe Abbildung 3)*.

Der erste Schritt besteht darin, die folgende URL *https://signup.multicloud.oracle.com/azure* aufzurufen, sich mit seinem Azure Account an der Oracle-Database-Service-for-Azure-Oberfläche *(siehe Abbildung 4)* anzumelden und **„Start guided account linking“** auszuwählen.

Im nächsten Schritt gelangt der Nutzer auf die Oracle-Cloud-Infrastructure-Seite und kann dort – wenn vorhanden – seinen bestehenden Oracle-Cloud-Account-Namen angeben, der mit dem Azure Account verknüpft werden soll. Falls noch kein OCI-Account vorhanden ist, kann dieser über **„Sign Up“** neu erstellt werden. Für diesen Vorgang wird allerdings eine Kreditkarte benötigt, da diese hinterlegt werden muss *(siehe Abbildung 5)*.

Nach erfolgreicher Anmeldung an dem OCI-Account muss der Nutzer die Region auswählen, in der die Verknüpfung mit Azure erfolgen soll. In diesem Fall erfolgt diese in Germany Central Frankfurt *(siehe Abbildung 6)*.

Nachdem nun die gewünschte Region ausgewählt wurde, müssen noch zwei Schritte in der ODSA Console ausgeführt werden *(siehe Abbildung 7)*. Auf der OCI-Seite sind bereits alle Arbeiten abgeschlossen und die OCI-Console wird im Prinzip nicht mehr benötigt.

In Step 1: **Assign ODSA roles** muss die **Multicloudlink Administrator Role** dem ODSA-User zugeordnet werden.

Die Vorgehensweise ist folgende:

1. Sign in to the Azure portal.
2. Under **Azure services**, click **More services**, then click **Azure Active Directory**.
3. Under **Manage**, click **Enterprise applications**.
4. In the list of enterprise applications, click on the name of the **Oracle Database Service** application to load the application’s Overview page.
5. Under **Manage**, click **Users and groups.**
6. Select the user you want to manage by clicking the checkbox beside the user name.
7. Click **Edit**.
8. On the **Edit Assignment** page, under **Select a role**, click **None selected** to open the **Select a role** panel.
9. Select the **ODSA Multicloudlink Administrator** role.
10. Click **Select**. The **Edit Assignment** page displays.
11. Review the assignment information, then click **Assign** to complete the assignment.
12. Sign out and sign back into ODSA.

Nach dem Zuweisen der Multicloudlink Administrator Role muss der Nutzer sich einmal in der ODSA Console abmelden und wieder neu anmelden. Danach folgt der zweite Schritt: **Link your subscriptions** *(siehe Abbildung 8)*.

Nach Auswahl des Buttons **Get started** gelangt der Nutzer in das Subscription-Management-Menü.

Hier wählt er die Subscription aus und klickt auf **„Link Subscription“** *(siehe Abbildung 9)*.

Anschließend gelangt er in das Oracle-Services-Menü *(siehe Abbildung 10)*, von dem aus er die Oracle-OCI-Datenbanken mittels Azure „Look and Feel“ anlegen und administrieren kann, ohne sich an dem Oracle Cloud Account anmelden zu müssen.

Zum Anlegen eines Base-Database-Systems müssen analog zu der ODSA-Multicloudlink-Administrator-Rolle im Azure Active Directory zusätzliche Rollen hinzugefügt werden, zum Beispiel

- ODSA Database Family Administrator

- ODSA BaseDB Infrastructure Administrator
- ODSA BaseDB PDB Administrator

Die Vorgehensweise zur Erstellung eines DB-Systems möchte ich anhand einer Base Database aufzeigen:

- **Base Database** auswählen.
- **Create** auswählen *(siehe Abbildung 11)*.
- Im Tab **Basics** den **Instance name** angeben *(siehe Abbildung 12)*.
- Im Tab **Configuration** ein entsprechendes Shape, Total Node Count, Available data storage in GB, Software Edition, License Type und Database Details auswählen beziehungsweise angeben *(siehe Abbildung 13)*.
- In diesem Fall verwende ich ein **VM.Standard.E4.Flex** Shape mit einer OCPU (2 vCPUS) und einem Knoten mit **256 GB** Plattenplatz. Als Datenbank wird eine **Standard Edition** in der Version **19.16**, eine Containerdatenbank namens **dbodsa** mit einer Pluggable-Datenbank **ODSA1** installiert.
- Im **Tab Networking** werden der Hostname Prefix und das virtuelle Azure-Netzwerk ausgewählt. Außerdem muss der OCI-CIDRs-Block für das OCI VCN angegeben werden, in diesem Fall **10.30.0.0/16**. Dabei ist zu beachten, dass sich die Subnetze nicht überschneiden, da sonst das Routing nicht funktioniert *(siehe Abbildung 14)*.
- Im Tab **Security** wird der SSH Key für den Zugriff auf das DB-System bereitgestellt. Die Anmeldung erfolgt standardmäßig über den User opc. Mit dem sudo-Kommando kann der User gewechselt werden. Folgende User sind per Default definiert:
  - root
  - grid
  - oracle

  Des Weiteren wird hier das Password des DB-Administrators SYS angegeben. Das hier definierte Password ist zudem auch das TDE Wallet Password. Im Tab **Management** kann das automatische Datenbank-Backup aktiviert und die Backup Retention Period eingestellt werden Im **Tab Review + create** die gemachten Eingaben überprüfen und mit einem Klick auf den Create **Button** das DB-System erzeugen.
- Nach dem Erstellen des DB-Systems wird unter Base Databases sowohl die CDB (dbodsa) wie auch die erzeugte PDB (ODSA1) angezeigt *(siehe Abbildung 15)*.
- In der Übersicht der einzelnen CDB/PDB wird im Feld Connection String der dazugehörige TNS-Names-Eintrag angezeigt *(siehe Abbildung 16)*.
- Trägt man nun diesen auf seinem Client in die TNSNAMES.ORA ein, so kann direkt auf die Datenbank zugegriffen werden. Weder das Routing noch die Firewall-Regeln (Security List) müssen dafür händisch freigeschaltet werden. Dies geschieht automatisch beim Erstellen des DB-Systems und kann so eine Menge Zeit und Nerven sparen.
- Ferner überträgt der ODSA-Service die OCI-Datenbankmetriken an Azure Application Insights und stellt Datenbankereignisse für Azure Log Analytics bereit.

Nachdem jetzt ein Base-Database-System in der Oracle Database Service for Azure Console erstellt wurde, möchte ich noch kurz auf die weiteren Möglichkeiten in der Console eingehen. *Abbildung 17* zeigt den aktuellen Stand dazu, welche Features bereits in der ODSA Console implementiert wurden. Die Themen Data Guard und Skalierung stehen aktuell noch auf der Roadmap. Wer aber diese Features nutzen möchte, kann dies im Moment nur über die OCI Console konfigurieren.
Über die ODSA Console können Pluggable Databases (PDBs) erstellt, geklont, verwaltet und gelöscht werden. Des Weiteren können auch manuelle Datenbank-Backups erstellt und wiederhergestellt werden.

Allerdings habe ich auch zwei wichtige Features in der ODSA Console vermisst. Da wäre auf der einen Seite die Möglichkeit, das DB-System zu stoppen, wenn die Ressource nicht mehr benötigt wird, und auf der anderen Seite das Einspielen von Grid-Infrastructure- sowie Datenbank-Patches und Updates. Diese Features funktionieren aktuell nur über die OCI Console.

Zusammenfassung und Fazit:

Was soll ein Kunde nun einsetzen – ODSA oder OCI-Azure-Interconnect? Die Antwort kann leider nicht pauschal beantwortet werden, sondern hängt von den jeweiligen Kundenanforderungen und dem Use Case ab. Wenn dieser Use Case Azure-Ressourcen mit Datenbanken in OCI verbinden möchte, ist dies natürlich mit ODSA möglich. Soll allerdings eine bi-direktionale Kommunikation von Azure- und OCI-Ressourcen möglich sein, so muss in diesem Fall der OCI-Azure-Interconnect verwendet werden. Eine andere Frage, die es zu klären gilt: Wie möchte der Kunde den Interconnect nutzen?

Wenn der Kunde sich nicht selbst um den Interconnect kümmern möchte, keine Netzwerkkosten bezahlen möchte sowie keine Probleme hat, den Interconnect mit anderen Kunden zu teilen, so ist ODSA die erste Wahl. Wenn nicht, bleibt nur die Variante, den Interconnect selbst zu betreiben. Dabei muss der Kunde die Ports für FastConnect und ExpressRoute zahlen und hat dafür auch einen SLA auf die OCI-Azure-Verbindung.

Über den Autor:

Kai-Uwe Fischer ist seit mehr als 23 Jahren im Oracle-Datenbank-Umfeld als Consultant unterwegs und beschäftigt sich seit Jahren intensiv mit der Oracle Cloud (OCI-C und OCI). In dieser Zeit hat er einige Kunden auf dem Weg in die Oracle Cloud beziehungsweise Multicloud begleitet.

Herr Fischer hält mehrere wichtige Cloud-Zertifizierungen wie zum Beispiel Oracle Cloud Infrastructure Certified Architect Professional, Specialist Oracle Database Cloud Migration and Integration sowie Oracle Cloud Database Services Specialist. Seit Januar ist er als Solution Architect Oracle Database & OCI bei der SVA System Vertrieb Alexander GmbH tätig.

Kai-Uwe Fischer
kai-uwe.fischer@sva.de

Da steht ein Elefant im Raum

Markus Flechtner, Ordix

„Wenn ein Elefant im Raum steht, dann sollte man ihn vorstellen." – Der Elefant hört in unserem Fall auf den Namen „Slonik" und ist das Logo der Open-Source-Datenbank PostgreSQL. „Slonik" ist russisch und steht für „kleiner Elefant". Mehr und mehr wird dieser Elefant zu einer Alternative auf dem Datenbank-Markt. Immer mehr und mehr Oracle-Anwender ziehen einen Wechsel Richtung PostgreSQL in Erwägung. Grund genug, einmal einen Blick auf die Unterschiede zwischen den beiden RDBMS, Oracle und PostgreSQL, zu werfen.

Der Hauptgrund für die Beschäftigung mit dem Thema „PostgreSQL" sind sicher die Kosten. Bei Oracle fallen Lizenz- und jährliche Supportkosten an. PostgreSQL ist in der Open-Source-Variante kostenlos verfügbar. Damit haben wir schon den größten Unterschied zwischen beiden Datenbanksystemen. Aber ist PostgreSQL wirklich kostenlos? Wer eine der kommerziellen Varianten einsetzt oder wer kommerziellen Support einkauft, der hat natürlich auch bei PostgreSQL Kosten. Und neben den Software- und Wartungs¬kosten kommen die Ausgaben für Schulungen, gegebenenfalls Migrationskosten usw. noch dazu. Die Aussage „PostgreSQL ist günstiger als Oracle" gilt in dieser Pauschalität nicht, das hat ja auch schon Daniel Westermann im Interview im Red Stack Magazin 6/2022 bestätigt.

Wenn man auf den kommerziellen Support verzichtet, dann hat man „nur" den Support durch die Community via Mailing-Listen. Und da gilt – wenn man fundierte Fragen stellt und sich an die Netiquette hält – durchaus: „Hier werden Sie geholfen". Ob der PostgreSQL-Support – kommerziell oder in der Community – dann besser ist als der Oracle-Support, über den in den DOAG-Umfragen die Mitglieder regelmäßig klagen, das muss dann jeder selbst entscheiden.

Sehr wichtig: die Community

Generell gilt: Ohne die Community geht bei PostgreSQL nichts. Die Community sorgt für Support, die Community entscheidet über die Features in den neuen Versionen usw. Die Community sorgt dafür, dass PostgreSQL lebt und weiterentwickelt wird. Wenn man der Mentalität „OpenSource ist kostenlos" folgt und nichts in und für die Community tut, dann wird man dem Open-Source-Prinzip nicht gerecht. Denn wenn alle nur konsumieren und sich nicht engagieren, dann geht es irgendwann nicht mehr weiter. Die Community lebt vom Mitmachen. Dazu muss man keinen C-Code für PostgreSQL schreiben. Es gibt vielfältige Möglichkeiten: Testumgebungen bereitstellen, neue Features testen, an der Dokumentation mitarbeiten, Wissen in Vorträgen oder Artikeln teilen, Konferenzen organisieren oder unterstützen usw. Kurzum, wie immer gilt auch hier: Wer etwas will, findet Wege!

Und wie sieht es auf technischer Ebene aus? Da gibt es viele Gemeinsamkeiten: Beides sind relationale Datenbank-Systeme und die Relationen-Algebra, die Codd'schen Regeln und die ACID-Regeln sorgen für einen engen Spielraum für die Implementierungen. Auch die SQL-Standards setzen Grenzen. Allerdings

erfüllt keines der beiden Systeme die SQL-Standards komplett. Aber wer eine Datenbank als „dummen Datenspeicher" betreibt und sämtliche Logik in der Applikation hält, für den sind die Unterschiede marginal.

```
postgres=# create user scott_usr password 'tiger';
CREATE ROLE
```

Listing 1: „Create User" wird – im ersten Moment irreführend mit „create role" quittiert

```
postgres=# create schema scott_schema authorization scott_usr;
CREATE SCHEMA
```

Listing 2: Schemaname und der zugehörige Eigentümer können sich unterscheiden

PostgreSQL ist schlanker

Beginnen wir mit der Installation: Eine Oracle-Installation belegt etwa 9 GB. Bei PostgreSQL sind es etwa 70 MB. Allerdings ist PostgreSQL dabei auch recht minimalistisch. Dinge, die bei Oracle dabei sind, sind bei PostgreSQL als Tools oder Erweiterungen verfügbar. Fast alles, was einem fehlen könnte, gibt es als Erweiterung. Diese Erweiterbarkeit ist Fluch und Segen zugleich. Fluch, weil man suchen muss. Weil man sich entscheiden muss: Nehme ich Backup-Tool A oder B? Wenn es dann um ein PostgreSQL-Upgrade geht, dann muss der DBA selbst schauen, ob es für die Erweiterungen neue Versionen gibt, die mit der neuen Datenbank-Version kompatibel sind. Das alles sorgt für Aufwand und damit auch für Kosten. Man hat als Administrator dann auch mehr Verantwortung, als wenn man sich auf ein weitestgehend „vorkonfiguriertes Paket" wie Oracle verlässt. Aber die Medaille hat wie immer zwei Seiten, denn es kann auch ein Segen sein: Man hat mehr Freiheiten und kann sich „sein" PostgreSQL so zusammenstellen, wie man es benötigt.

Wenn man nach der Installation eine Datenbank anlegt, dann braucht man bei Oracle mehr als 4 GB Platz. Ein leerer PostgreSQL-Cluster braucht ca. 40 MB und für eine Datenbank kommen noch 2 MB dazu. Auch wenn man sich die Prozessliste anschaut, dann ist PostgreSQL schlanker. Während es bei Oracle inzwischen mehr als 40 Hintergrundprozesse gibt, sind es bei einem einfachen PostgreSQL weniger als 10. Zugegebenermaßen kommen durch Erweiterungen teilweise noch Prozesse dazu, aber insgesamt gilt, dass PostgreSQL wesentlich ressourcenfreundlicher ist als Oracle.

Wenn man mit verschiedenen Datenbanken arbeitet, dann ist eine Begriffsvielfalt unvermeidlich. So ist es auch beim Vergleich von Oracle mit PostgreSQL. Vergleichbare Funktionalität hat unterschiedliche Namen und die gleichen Begriffe bezeichnen unterschiedliche Funktionalität.

Während man bei Oracle von „Tabellen" und „Indizes" spricht, sind es bei PostgreSQL erst einmal „Relationen". Im Alltag redet man natürlich auch bei PostgreSQL von Tabellen, Indizes, Datensätzen und Spalten, aber in der Dokumentation und in den System-Tabellen und -Views findet man oft die Begriffe „Relation", „Tupel" und „Attribute". Diese Begriffe kommen aus der Datenbanktheorie. PostgreSQL orientiert sich aufgrund seiner universitären Herkunft oft an diesen akademischen Begriffen.

Wenn man bei Oracle bei „Cluster" wohl an den Real-Application-Cluster denkt, dann ist in PostgreSQL ein „Cluster" eine „Sammlung von Datenbanken, die von einer PostgreSQL-Instanz verwaltet wird.

Auch der Begriff „Tablespace" hat bei PostgreSQL eine andere Bedeutung. Bei Oracle gilt: Ein Tablespace besteht aus Datendateien und in diesen Dateien liegen die Objekte. Bei PostgreSQL ist ein „Tablespace" nur ein „Alias" für ein Verzeichnis. In diesem Verzeichnis liegen dann pro Relation je nach Größe eine oder mehrere Dateien. Dieses unterschiedliche Konzept ist auch der Grund dafür, dass zusätzliche Tablespaces bei PostgreSQL seltener verwendet werden als bei Oracle.

Benutzer, Rollen, Schemata

Drei Begriffe, viele Ähnlichkeiten. Es gibt aber Unterschiede. Bei Oracle sind „Schemata" „Benutzer mit Objekten". Getrennt davon gibt es Rollen als Sammlung von Rechten. Bei PostgreSQL gilt das auch, aber Rollen und Benutzer teilen sich den gleichen Namensraum. Ein „Benutzer" ist eine „Rolle mit Login-Recht". Zusätzlich zu Benutzer und Rollen, die clusterweit definiert sind, gibt es „Schemata" auf Datenbank-Ebene. Jedes Schema hat einen Benutzer als Eigentümer. In den meisten Fällen gibt es ein 1:1-Mapping, es sind aber separate Namensräume. Das kann durchaus für Verwirrung sorgen *(siehe Listing 1 und 2)*.

Transaktionen: „ORA-1555" versus „Table Bloat"

Unterschiede gibt es auch bei den Transaktionen. Nicht im ganz großen Sinne, denn beide Datenbanken befolgen die ACID-Regeln, aber bei der Implementierung. Während es bei Oracle die Redolog-Dateien gibt, die zyklisch geschrieben werden, gibt es bei PostgreSQL WAL-Segmente. „WAL" steht für Write-Ahead-Log und meint die gleiche Funktionalität; es sind die Transaktionslogs. Dabei werden diese Dateien nicht zyklisch überschrieben, sondern – grob gesagt – von PostgreSQL in einem gewissen Rahmen (Plattenplatz) verwaltet und auch wiederverwendet.

Auch beim Thema „Undo" gehen beide Datenbanken unterschiedliche Wege. Während wir bei Oracle die „Undo"-Tablespaces haben, löst PostgreSQL das Problem der Lesekonsistenz auf andere, etwas einfachere Weise. Jeder Datensatz hat zwei Spalten, xmin und xmax, die den Gültigkeitszeitraum (in Transaktions-IDs) eines Datensatzes angeben. Beispielsweise wird bei einem Update xmax als Obergrenze gesetzt und eine (geänderte) Kopie des Datensatzes angelegt, die dann die aktuelle Version darstellt. Vorteil dieses Verfahrens ist, dass Postgre-

```
postgres# update scott.emp set name='KONG' where name='KING';
UPDATE 1
postgres# select 1/0;
ERROR:  22012: division by zero
LOCATION:  int4div, int.c:846
postgres# update scott.emp set salary=salary+100 where empno=7839;
ERROR:  25P02: current transaction is aborted, commands ignored until end of transaction block
LOCATION:  exec_simple_query, postgres.c:1116
postgres# commit;
ROLLBACK
```

Listing 3: Verhalten von PostgreSQL bei Fehlern in einer Transaktion – nur ROLLBACK erlaubt

SQL-Benutzer kein Pendant zum Fehler „ORA-1555 Snapshot too old“ kennen, der immer dann kommt, wenn Oracle aus den Undo-Daten das lesekonsistente Bild eines Datensatzes nicht mehr erstellen kann. Nachteil ist, dass DML die Tabellen größer macht. Vereinfacht formuliert verdoppelt ein UPDATE auf alle Datensätze die Größe einer Tabelle. Dieses Verhalten wird auch als „Table Bloat“ bezeichnet. Daher sollte man regelmäßig aufräumen, das bedeutet, alte Datensatz-Versionen, die von keiner Sitzung mehr benötigt werden, zu löschen. Dazu gibt es den VACUUM-Befehl und den Autovacuum-Daemon, der diese Arbeit automatisch im Hintergrund erledigen kann.

Auch bei Transaktionsfehlern gehen beide Datenbanken einen unterschiedlichen Weg. Wenn bei Oracle ein Befehl auf einen Fehler läuft, dann wird nur dieser Befehl nicht ausgeführt. Bei PostgreSQL wird hingegen die ganze Transaktion zurückgerollt und erst nach einem Transaktionsende („ROLLBACK“) geht es weiter. Dieser Unterschied kann dazu führen, dass man bei migrierten Applikationen die Fehlerbehandlung anpassen muss *(siehe Listing 3)*.

Ein manchmal recht angenehmer Unterschied von PostgreSQL zu Oracle ist das „transaktionale DDL“: Man kann DDL mittels ROLLBACK zurückrollen. Das kann zum Beispielbei fehlgeschlagenen Applikationsupgrades sehr helfen.

Datum – mit oder ohne Uhrzeit?

PostgreSQL hat generell mehr Datentypen als Oracle. So gibt es dort beispielsweise den Datentyp boolean, der bei Oracle erst für Version 23c angekündigt ist. Auch der Typ „DOMAIN“, kurz gesagt ein Datentyp mit verknüpftem Constraint, den es in PostgreSQL schon länger gibt, soll mit Oracle 23c kommen. Oder den Typ tsrange, der die Arbeit mit Zeiträumen erleichtert. Einen feinen Unterschied gibt es beim Datentyp DATE: bei Oracle mit Uhrzeit; bei PostgreSQL ohne Uhrzeit. Spätestens, wenn man bei einer Migration die Datentypen 1:1 übernimmt, dann führt das in diesem Fall zu Problemen. Ein Unterschied mit Aha-Erlebnis zeigt sich auch bei numerischen Typen, wenn man sie dividiert *(siehe Listing 4)*.

```
postgres=# select 5/2,5.0/2;
 ?column? |      ?column?
----------+--------------------
        2 | 2.5000000000000000
(1 row)
```

Listing 4: „Integer“ durch „Integer“ ergibt „Integer“, „Float“ durch „Integer“ ergibt „Float“

```
 SQL> select 'DOAG'||NULL from dual;
'DOA
----
DOAG
```

Listing 5: Verknüpfung mit NULL-Strings in Oracle

```
postgres=# select 'DOAG'||NULL;
 ?column?
----------
 [NULL]
(1 row
```

Listing 6: Verknüpfung mit NULL-Strings in PostgreSQL

Bei Oracle ergeben beide Rechnungen „2.5“.

Hochverfügbarkeit

Beim Begriff „Hochverfügbarkeit“ kommen bei Oracle sofort die Begriffe „Real Application Cluster“, „(Active) Data Guard“ und „GoldenGate“ ins Spiel. Bei PostgreSQL basiert Hochverfügbarkeit auf Replikation. Dabei gibt es die gleichen Formen der Standby-Datenbank wie bei Oracle. „Active DataGuard“, bei PostgreSQL „Hot-Standby“ genannt, bekommt man kostenfrei dazu. Für komplexere HA-Systeme wird bei PostgreSQL häufig Patroni *(https://github.com/zalando/patroni)* eingesetzt. Patroni ist aber keine vollständige Anleitung für

PostgreSQL-Hochverfügbarkeit, sondern eine Vorlage; eine Blaupause. Das gibt dem DBA mehr Freiheiten bei der Implementierung einer konkreten Lösung, aber auch mehr Verantwortung.

Der kleine Unterschied

Wenn man sich mit beiden Datenbanken beschäftigt oder sogar eine Datenbank inklusive Applikation von Oracle nach PostgreSQL migriert, dann stößt man schnell auf die Verhaltensunterschiede der beiden Datenbanken. Neben den oben geschilderten Unterschieden im Transaktionsverhalten gibt es noch einige mehr. Hier ein paar Beispiele:

Ein Klassiker dabei ist der unterschiedliche Umgang mit NULL-Strings. Oracle behandelt NULL und leere Strings gleich *(siehe Listing 5)*.

PostgreSQL hingegen verarbeitet NULL-Strings ANSI-SQL-konform *(siehe Listing 6)*.

Da NULL-Strings bei Verknüpfungen sicher nicht der Normalfall sind, ist so ein Unterschied bei Tests nach Migrationen nicht so einfach zu entdecken. Ein kleiner, aber lästiger Unterschied zwischen beiden Datenbanken ist die Tatsache, dass Objektnamen bei Oracle standardmäßig in Großschrift sind, während es bei PostgreSQL Kleinbuchstaben sind. Man muss also mit Anführungszeichen arbeiten.

Ab jetzt nur noch PostgreSQL?

Wie immer gilt in der Realität: Es gibt kein schwarz oder weiß. Es gibt kein „besser oder schlechter", sondern nur: „Für ein Umfeld oder eine Applikation ist diese Datenbank besser geeignet als eine andere." Viele Oracle-Anwender setzen bei neuen Projekten auf PostgreSQL, um so Erfahrungen zu sammeln. Bestehende Systeme werden eher seltener migriert, insbesondere wenn es große und kritische Systeme gibt. Und so wird es ein Miteinander geben. Da es immer mehr Datenbank-Anwendungen gibt, ist in der Datenbank-Welt auch Platz für mehrere Datenbanken. Das ist auch gut für die Anwender, denn wie immer gilt: Konkurrenz belebt das Geschäft.

Über den Autor

Markus Flechtner ist seit mehr als 30 Jahren im Oracle-Umfeld tätig. Nach einer Tätigkeit als Entwickler (Oracle-Forms, Oracle-Reports, PL/SQL) wechselte er in den DBA-Bereich. Seit 2022 ist er als Principal Consultant und Teamleiter bei der ORDIX AG. Er gibt Schulungen zu den Datenbank-Themen und ist häufig Sprecher auf nationalen und internationalen Konferenzen. Seit 2020 ist er Oracle ACE. In der DOAG engagiert er sich als Themenverantwortlicher für Open-Source-Datenbanken.

Markus Flechtner
mfl@ordix.de

Die Oracle-Anwenderkonferenz

Eventpartner: AOUG SOUG

anwenderkonferenz.doag.org

Ein Trio zum Early-Bird-Preis: Kräftig sparen bei den nächsten DOAG-Konferenzen

In der ersten Jahreshälfte 2023 finden noch drei DOAG-Konferenzen statt, für die ein attraktiver Frühbucherrabatt angeboten wird.

Marcos López

APEX connect 2023

Der bis zum 30. März 2023 laufende Frühbucherpreis bietet eine satte Ersparnis von über 250 EUR auf das reguläre Zwei-Tages-Ticket der APEX-Konferenz am 3. und 4. Mai in Berlin, inklusive Community-Abend und Verpflegung an allen Konferenztagen. *https://shop.doag.org/shop/prd.509.apex-connect--2-tage/*

DOAG 2023 Datenbank mit Exaday

Am 24. und 25. Mai dreht sich in Düsseldorf alles um die Datenbankthemen Oracle Database, PostgreSQL und MySQL sowie die Engineered-Systems-Schwerpunkte Exadata und Oracle Database Appliance (ODA). Das reguläre Ticket ist noch bis zum 19. April 2023 anstatt für 950 EUR für 800 EUR zu haben. *https://shop.doag.org/shop/prd.593.doag-2023-datenbank-konferenzpass/*

CloudLand 2023

Die Early-Bird-Aktion des Cloud Native Festivals, das vom 20. bis 23. Juni 2023 wieder im Phantasialand stattfindet, ermöglicht einen rabattierten Ticketkauf für drei verschiedene Ticketvarianten. Der Frühbucherpreis für die CloudLand ist noch bis zum 9. Mai 2023 verfügbar. *https://shop.doag.org/events/cloudland/shop/*

PremiumCard

An dieser Stelle möchten wir noch auf ein besonderes Angebot hinweisen: die DOAG PremiumCard. Sie ermöglicht eine Event-Flatrate für die meisten DOAG-Veranstaltungen. Alle weiteren Infomationen dazu finden Sie hier: *https://www.doag.org/de/verein-mitgliedschaft/mitgliedschaft/#c7321*

Mit einem Katzensprung von Oracle zu PostgreSQL?

Wie die Migration in die AWS-Cloud besser gelingt

Christian Ballweg, Opitz Consulting Deutschland

„Das Unerwartete zu erwarten, zeigt einen durch und durch modernen Geist." (Oscar Wilde). Die Migration von Oracle auf Open-Source-Alternativen – besonders PostgreSQL – ist derzeit in aller Munde. Dieser Artikel gibt eine Übersicht über eine gute Vorbereitung, den Migrationsweg und mögliche Herausforderungen.

Eine Migration von Oracle zu PostgreSQL wird besonders von den Anbietern der Tools und Zielumgebungen oft als recht einfach dargestellt. Das dient der Übersicht und schafft auch Vertrauen in die Darstellung. Denn wer will hier schon mit Problemen empfangen werden? Leider stellt sich oft erst im Verlauf einer Migration zu dem Open-Source-Datenbanksystem heraus, was man noch hätte untersuchen können, was vergessen oder falsch verstanden wurde – oder dass eine Migration durch falsch oder nicht kalkulierte Aufwände und Risiken am Ende komplexer und damit teurer wird als ursprünglich angenommen. Deshalb sollten der Migrationsweg gut vorbereitet sein und mögliche Herausforderungen eingeplant werden.

In diesem Artikel will ich auf einzelne Schritte und interessante Aspekte einer Migration eingehen. Dadurch erhalten Sie eine empfohlene Methode, wie Sie eine Migration von Oracle nach PostgreSQL planen und durchführen können.

Tatsächlich kann alles glatt laufen. Wir erleben bei unserer Kundschaft immer wieder zwei Extrempunkte bei der Migration:

1. Einfach und schnell: Manchmal gelingt der Übergang zu PostgreSQL automatisiert und ohne Probleme, wenn wir mithilfe des AWS Schema Conversion Tool (SCT) das Zielschema aus der Oracle-Quelle nach PostgreSQL übersetzen. Wir schauen noch einmal über die vorgeschlagenen Datentypen und schließen die Migration mit der Befüllung des Zielschemas durch den AWS Database Migration Service (DMS) ab. Anschließend aktivieren wir noch die Trigger und Foreign-Key Constraints, die eine parallele Befüllung per DMS gegebenenfalls verhindert hätten. Die Applikation ist dabei einfach gehalten und bedarf keiner weiteren Anpassung als einer minimalen Umstellung des „DB-Dialekts".
2. Unerwartet kompliziert: Die Datenbank besteht aus vielen nicht automatisch konvertierbaren Codebestandteilen und komplizierter Logik, deren Analyse und Umstellung viele Personentage, Wochen oder sogar Monate verschlingen kann – und schlimmstenfalls durch unsachgemäße Umstellung unvollständige oder falsche Ergebnisse in schlechter Performance liefert.

Ob es einfach oder kompliziert wird, können Sie im Vorfeld nicht sicher wissen. Doch das heißt nicht, dass Sie einen Blindflug riskieren sollten. Denn der kann unter Umständen teuer werden. Deshalb ist es hilfreich, die wichtigsten Herausforderungen zu kennen und Ihre Migration bewusst zu steuern.

Doch bevor wir nun direkt in Details springen und uns über mögliche Probleme und deren Lösung Gedanken machen, sollten wir einen Schritt zurücktreten und uns das Gesamtvorhaben ansehen.
Im Sinne von

„Wenn ich eine Stunde Zeit hätte, um ein Problem zu lösen, würde ich 55 Minuten damit verbringen, über das Problem nachzudenken und 5 Minuten über die Lösung."
Albert Einstein

Stellen Sie die Frage nach dem Warum

Eine Migration von einer zur anderen DB-Engine und dazu noch vom eigenen Rechenzentrum in die Cloud ist selten trivial. Zwei Fragen sollten Sie sich im Vorfeld stellen: Welche Ziele verfolgen Sie? Und: Welche Aufwände wollen Sie investieren, um diese Ziele zu erreichen?

Mit der STAR-Methode *[1]* können Sie sich den Antworten Schritt für Schritt annähern:

S **Situation:** Warum widmen Sie sich diesem Vorhaben? Welches Umfeld ist betroffen und wie sieht es aus?
T **Task:** Welche Probleme möchten Sie lösen? Welche (kurz- und langfristigen) Ziele verfolgen Sie?
A **Action:** Was müssen Sie zur Zielerreichung tun? Welche Werkzeuge können Sie dabei unterstützen?
R **Result:** Wann sind diese Ziele erfüllt? Wann würden Sie die Reißleine ziehen?

Wann ist PostgreSQL eine Alternative?

Diese drei Auslöser für die Migration zu PostgreSQL erleben wir in der Praxis am häufigsten:

- Wenn Upgrades nötig werden … stehen oft größere Veränderungen ins Haus, zum Beispiel wenn ein Produkt wie Oracle <19c nicht mehr supportet wird und viele beginnen, an einen Wechsel zu denken.
- Wenn die Lizenzkosten explodieren … überlegen viele, zu Open-Source-Produkten zu wechseln.
- Wenn Wildwuchs eingedämmt werden soll …, weil beispielsweise zu viele verschiedene Systeme unterschiedlicher Anbieter im Einsatz sind, kann die Kon-

Abbildung 1: Vorhaben unterteilen (Quelle: Christian Ballweg)

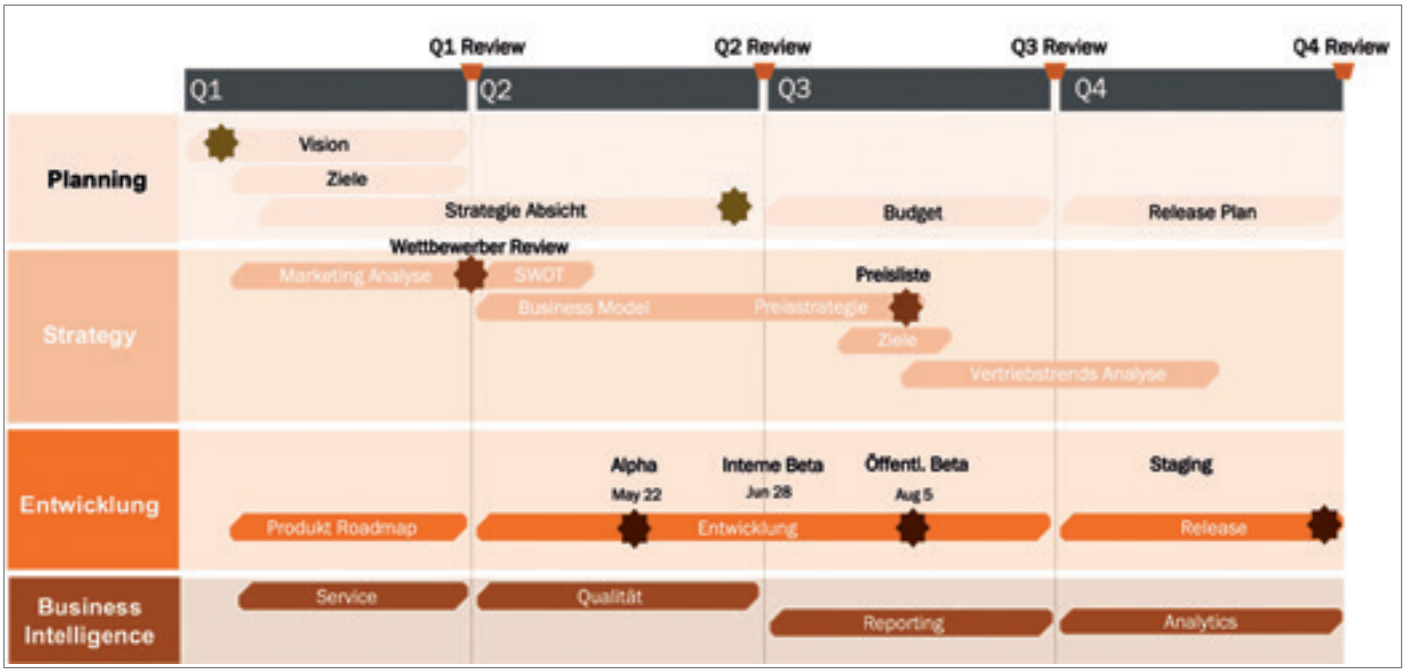


Abbildung 2: Roadmap-Beispiel aus ©projekte-leicht-gemacht.de [2]

solidierung auf ein anderes Produkt erwogen werden.

Mit einer Migration zu PostgreSQL verfolgen Sie zudem möglicherweise folgende strategischen Ziele:

- Modernisierung und Einführung einer skalierbaren Architektur
  - Virtualisierung oder Konsolidierung auf Hyper Converged Infrastructure
  - Containerisierung mit Docker/Kubernetes
  - Automatisierung und Dynamisierung (CI/CD Deployment inklusive DB)
- Datenbankmigration in die Cloud
  - als Managed Service
  - auf selbst verwaltete (virtuelle) Server

Warum PostgreSQL für die Erreichung dieser Ziele eine gute Wahl darstellen kann, ist bereits vielen DOAG-Vorträgen und Blog-Artikeln zu entnehmen. Exemplarisch erwähnen lassen sich neben Lizenzkosten-Ersparnissen auch die vielfältige Einsetzbarkeit, Erweiterbarkeit und Robustheit, bei gleichzeitig schlankem und Ressourcen-schonendem Aufbau und damit einhergehender guter Performance von PostgreSQL.

Um nicht im erwähnten Blindflug durch ein Projekt zu stolpern, ist eine gute Vorbereitung nötig. Immerhin sind nicht nur DBAs daran beteiligt, sondern auch Zuständige für Development, Architecture und so weiter bis hin zur Fachabteilung, mit Mitarbeiter:innen, die Ihre Dateninhalte kennen und nutzen. *Abbildung 1* zeigt die drei Bereiche, in die Sie ein Migrationsvorhaben unterteilen können.

1. Roadmap

Wie bei jeder Vision, die Sie in die Praxis umsetzen wollen, erstellen Sie auch bei der Migration zu PostgreSQL zunächst eine Roadmap *[2]*. Darin visualisieren Sie die einzelnen Meilensteine *(Abbildung 2* zeigt hierzu ein Beispiel), die später im Projektplan mit den notwendigen Details konkretisiert und fortgeschrieben werden.

Die Roadmap gibt einen Überblick über das gesamte Projekt, ohne zu sehr ins Detail zu gehen. Sie hilft Ihnen, Zwischenziele zu formulieren und sich einen ersten Überblick über den zeitlichen Ablauf zu verschaffen.

Vielleicht fragen Sie sich, warum es so wichtig ist, einen einfachen Überblick zu bekommen?

Hier geht es um mehr als die Abläufe der Migration selbst. Von der Datenbank, die Sie migrieren, sind viele weitere Systeme abhängig, zum Beispiel

- Anwendungen, die die Datenbank direkt nutzen
- ETL-Prozesse, die Daten in die Datenbank laden oder aus ihr extrahieren
- andere Anwendungen, die etwa über Datenbanklinks die Datenbank als Integrationsschnittstelle nutzen

Außerdem sind die Modernisierungsaktivitäten der einzelnen Stakeholder zu koordinieren. Deshalb ist die Erarbeitung einer Roadmap eine nicht zu unterschätzende Aufgabe.

Die Roadmap dient also als Grundlage für eine detaillierte Programm- und Projektplanung. Dieser mit den zuvor gesammelten Fakten auszuarbeitende Projektplan ist detaillierter. Er enthält konkret benannte Aufwände, die Ressourcenplanung sowie Termine für die Umsetzung.

Wie wir bei Opitz Consulting vorgehen, wenn wir eine Roadmap erstellen, zeigt das „OC-Vorgehensmodell für die Erstellung einer Roadmap“ in *Abbildung 3*.

Da technische und fachliche Analysen gegenseitig aufeinander einwirken, sind für eine detaillierte Erarbeitung im Rahmen der Projektplanung und späteren Durchführung mehrere Iterationen notwendig (vgl. *Abbildung 4*).

Was brauchen Sie, um herauszufinden, ob eine Migration generell durchführbar ist und welche Werkzeuge und Zielarchitektur Sie wählen sollten? Darum geht es im Pre-Assessment. Werkzeuge wie der „Migration Assessment Report“ des SCT sowie vorbereitete Aufgabenpakete und Fragen helfen Ihnen, die Antworten zu finden.

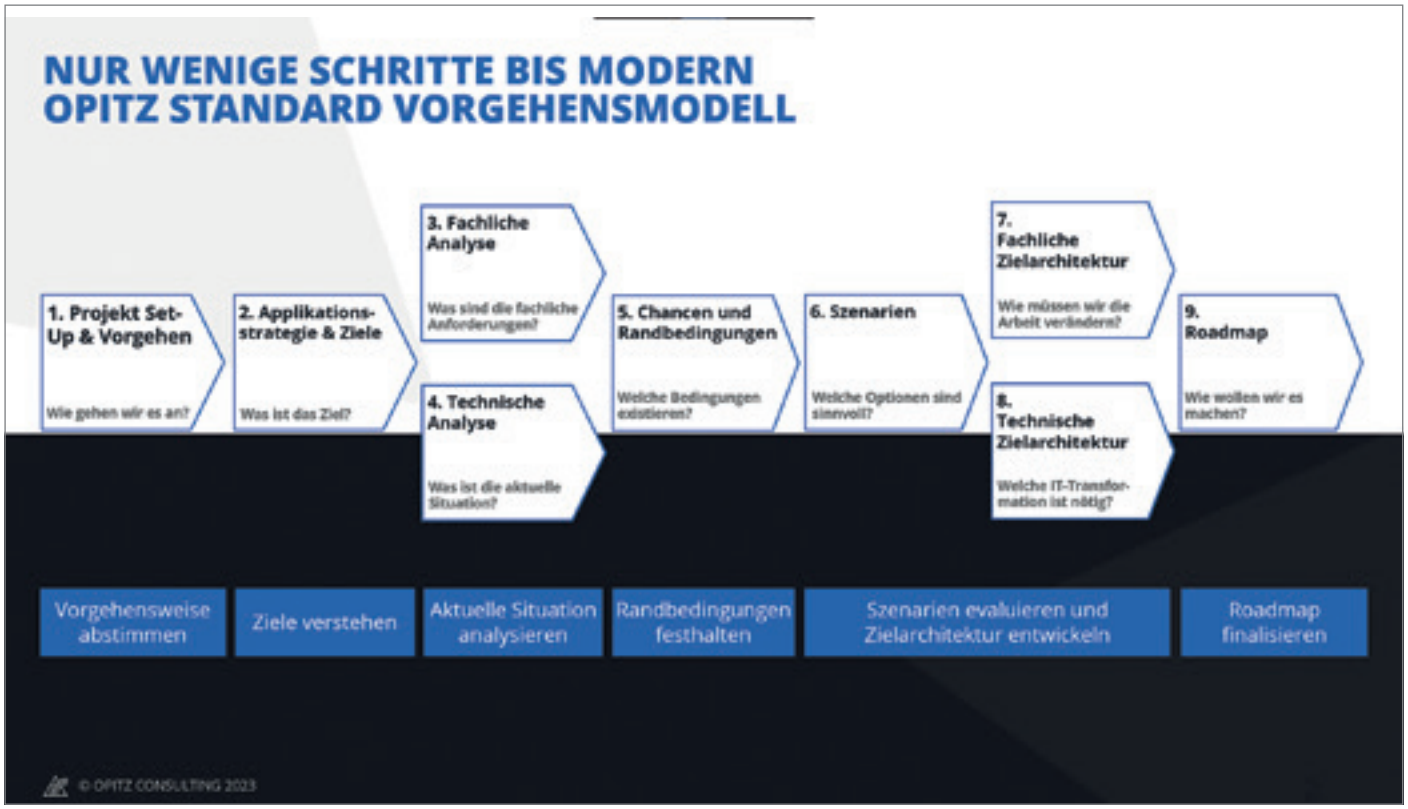


Abbildung 3: OC-Vorgehensmodell (©Opitz Consulting)

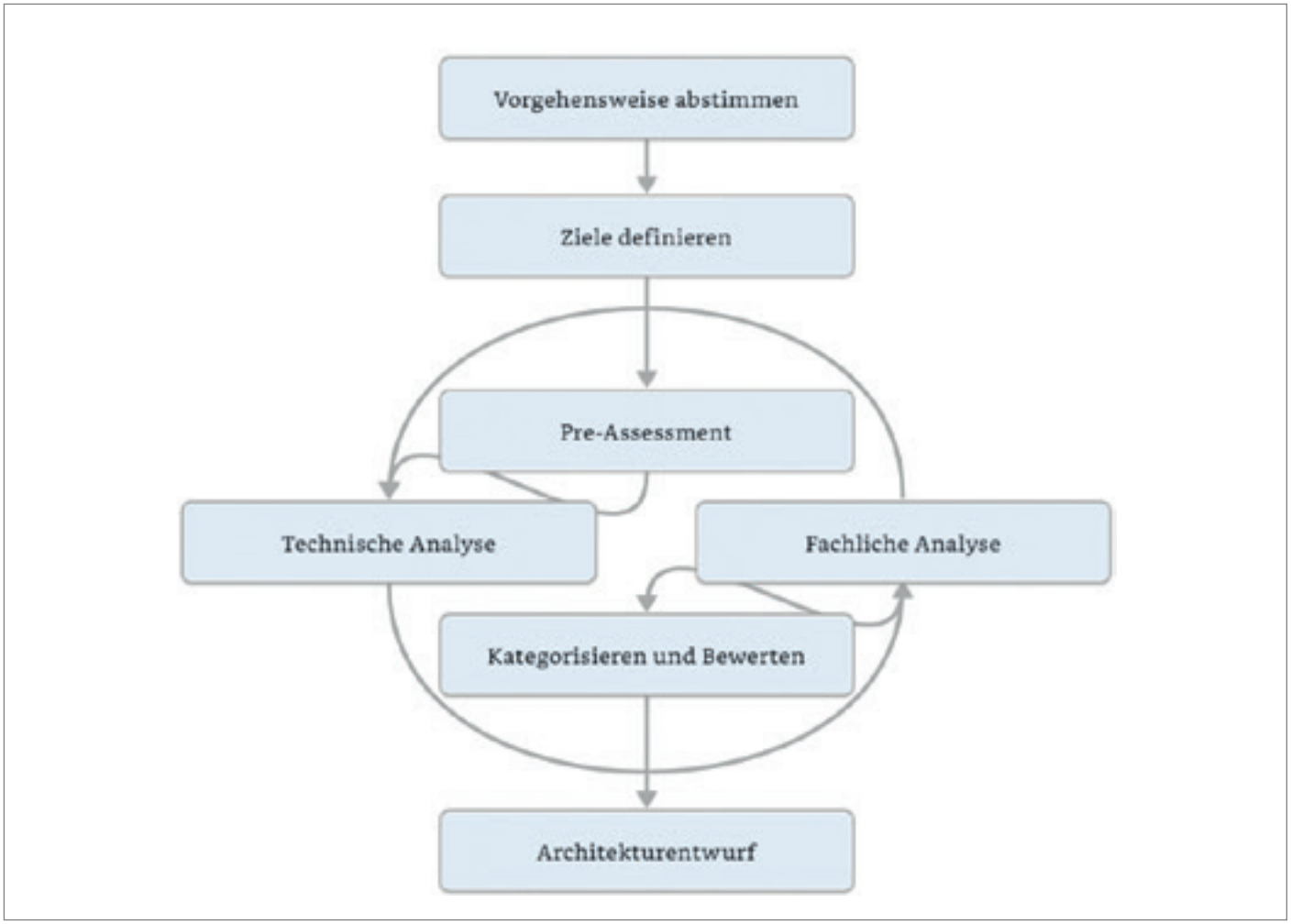


Abbildung 4: Architekturanalyse (©Opitz Consulting)

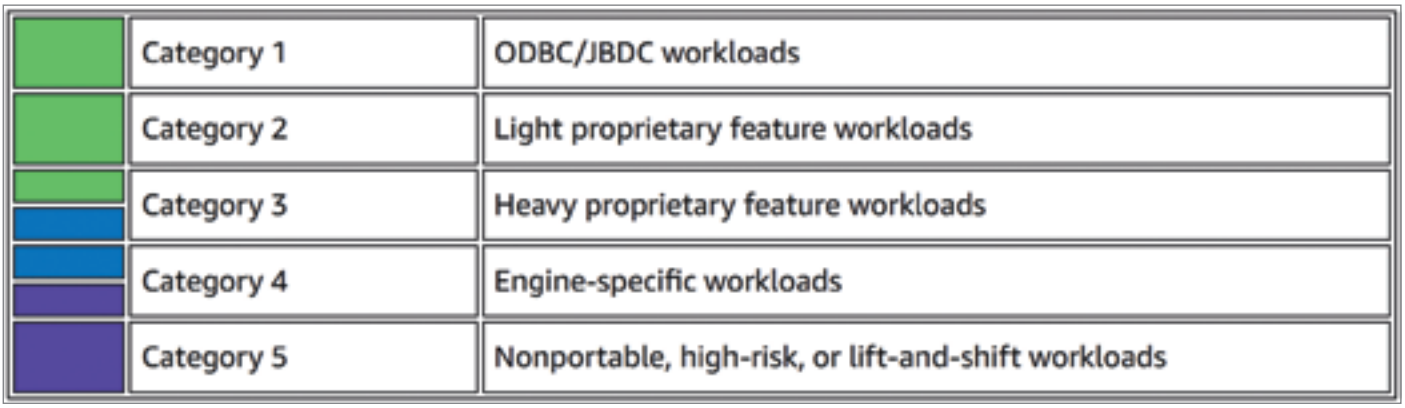


Abbildung 5: AWS Workload Categories (©Amazon Web Services [3])

Wo die Roadmap aufhört – häufig nach dem Pre-Assessment oder der ersten Iteration –, geht es im Laufe der Projektumsetzung weiter ins Detail. Wie schon erwähnt, müssen für die Roadmap nur die wichtigsten Kernpunkte identifiziert werden.

2. Projektplanung

In diesem Abschnitt der Migration geht es darum, Fragen aus dem Pre-Assessment zu analysieren und zu konkretisieren:

- Wieviel Code ist in der DB vorhanden?
- Zu welchem Prozentsatz ist eine automatische Konvertierbarkeit gegeben?
- Welche Codeteile müssen neu geschrieben werden?
- Welche Datentypen und internen Funktionen müssen ersetzt werden?
- Müssen weitere generelle und spezifische Probleme gelöst werden?
- Welche Dokumentation kann hierbei unterstützen?

Bei der Beantwortung dieser Fragen können entsprechende Analysewerkzeuge helfen, etwa das AWS Schema Conversion Tool (SCT).

Analog zu *Abbildung 5* folgt die Klassifizierung der Komplexität (im Folgenden abgekürzt mit „Cat“).

Die Kategorien lassen sich nicht-technisch wie folgt zusammenfassen:

- Cat 1:
  - „simpel, leicht und schnell“
  - „wenig Know-how erforderlich“
- Cat 2:
  - „viele Änderungen geringer Komplexität“
  - „wenige Änderungen hoher Komplexität“
  - „Know-how vorhanden“
- Cat 3:
  - „viele Änderungen hoher Komplexität“
  - „Know-how muss auf-/ausgebaut werden“

Eine ausführlichere Darstellung liefert AWS in der Dokumentation **„Migration strategy for relational databases“** *[3]*. In den Kapiteln **„Qualify workloads“** und **„Choose a migration strategy“** wird auch die zitierte Kategorisierung dargestellt.

Sind die Kategorien „Engine-specific workloads“ (Cat 4) und „Non-portable, high-risk, or lift-and-shift workloads“ (Cat 5) vorhanden, gilt dies als Ausschlusskriterium für eine Migration von Oracle nach PostgreSQL.

Achtung:

Die detaillierte Bewertung dieser Kriterien wird durch das dort ebenfalls zitierte „Workload Qualification Framework“ (WQF) vorgenommen, das nur noch über eine spezielle Anfrage an AWS zu bekommen ist.

Ist dies abgeschlossen, können wir drei grundlegende Fragen beantworten. Diese Fragen sagen uns, ob das Projekt realisierbar ist:

- Ist es technologisch sinnvoll?
- Ist es durchführbar?
- Ist es finanziell attraktiv?

Tipp:

Wir empfehlen dringend, mit einer „Cat 1“-Aufgabe oder einer (bereits sehr gut durchdrungenen) „Cat 2“-Aufgabe zu beginnen. Eine zu komplexe Aufgabe braucht zu viel Abstimmungsaufwand und zieht sich schnell über Monate hin. Damit verspielen Sie nicht nur das Vertrauen des Managements und der beteiligten Stakeholder, sondern auch die Motivation aller Beteiligten.

Die Aufgabe, eine Oracle-Datenbank nach PostgreSQL zu migrieren, *kann,* aber *muss nicht* die Migration der gesamten Datenbank bedeuten. Gerade in einer stark konsolidierten Umgebung, die (um Lizenzkosten zu sparen) viele Anwendungen *ohne* Bezug *zueinander* (!) auf einer leistungsfähigen und verfügbaren Datenbank wie etwa einem Oracle Real Application Cluster vereint, vielleicht sogar auf einer sehr leistungsfähigen Hardware wie einer Exadata, finden sich nicht selten solche voneinander unabhängigen Anwendungen.

Hier kann es sinnvoll sein, wenn möglich, einzelne Anwendungen aus diesem Konstrukt herauszulösen und separat nach PostgreSQL zu migrieren. Das begrenzt die Komplexität in der Gesamtbetrachtung, setzt aber eine sehr gute Analyse im Vorfeld voraus, die sehr wahrscheinlich nicht vom DBA allein geleistet werden kann.

Der DBA kann jedoch viele Details ermitteln und bei der Analyse unterstützen, zum Beispiel durch die Analyse der Datenbankgröße, des Schemas, der Anwendungstabellen sowie von weiteren Punkten, wie

- verwendete Zeichensätze
- problematische Datentypen
- Durchsatz (Redolog-Volumen pro Tag, leider nur bezogen auf die gesamte DB)
- Größe der Large Objects (Performance-Faktor bei der Datenmigration)
- Größe der Indizes (Performance-Faktor bei der Bereitstellung nach der Datenmigration)

3. Umsetzung

Abbildung 6 zeigt den Zyklus, der nun zu Dokumentations- und Testzwecken – vielleicht auch über mehrere Stages mehrfach wiederholt – durchlaufen wird.

Wie bereits erwähnt, sind viele Bereiche einer Migration betroffen. Sie alle müssen an der Umsetzung beteiligt werden. Die folgende Liste zeigt, welche Bereiche hierbei unterstützen können. Je nach Komplexitätskategorie werden mindestens die ersten beiden Bereiche benötigt, gegebenenfalls auch weitere:

- Datenbankadministration (Mitarbeiter:innen mit Expertise zu den Quell- und Zieldatenbanken)
  - Prüfung/Bewertung der Quelle
  - Umsetzung: Schema-Migration
  - Umsetzung: Data-Migration
  - Ausgestaltung der Zielarchitektur
  - Unterstützung bei Performancevergleich und Analyse
- Development (Mitarbeiter:innen mit Anwendungs- und Programmiererfahrung)
  - Codeanpassungen der Applikation
  - Tests
  - Festlegung der Akzeptanzkriterien (mit der Fachabteilung, s. u.)
- Cloud- und Infrastruktur-Architekt:innen
- Netzwerk- und Security-Fachkräfte
- Storage-Fachkräfte
- … und natürlich die Fachabteilung mit Mitarbeiter:innen, die die Inhalte verstehen und Ergebnisse bewerten können.

Damit die Migration nicht zum Selbstzweck gerät, ist es wichtig, sich auch über folgende Punkte Klarheit zu verschaffen:

a. Wie sieht das gewünschte oder erwartete Endergebnis aus? Wann war die Migration für Sie erfolgreich?
b. Wann stoppen Sie das Migrationsvorhaben? Nehmen Sie ein Teilergebnis mit? Schwenken Sie komplett zurück?
c. Welche Risiken haben Sie zu erwarten?
d. Wie können diese Risiken minimiert werden?

In diesem Artikel auf jedes Detail einzugehen, würde den Rahmen sprengen. Eine Übersicht der möglichen Probleme bei einer Migration von Oracle zu PostgreSQL mit SCT enthält die Dokumentation „Oracle to Aurora PostgreSQL Migration Playbook: AWS SCT Action Code Index“ *[4]*. Sie listet zum Beispiel mögliche Probleme auf hinsichtlich

- SQL, Cursors, Merge, Query Hints
- DDL für Tables, (Materialized) Views, Triggers, Data Types
- Transaction Isolation, Stored Procedures, Partitioning, DB Links

Ein sehr triviales Beispiel mit möglicherweise großer Wirkung ist die unterschiedliche Behandlung von NULL, die im EnterpriseDB-Artikel *„How NULL and empty strings are treated in PostgreSQL vs Oracle“ [5]* dargestellt ist.

Durch die unterschiedliche Behandlung von NULL kann es vorkommen, dass eine andere als die erwartete Ergebnismenge zurückgeliefert wird. Der Grund: Oracle setzt einen String mit 0 Zeichen ('') mit NULL gleich, während PostgreSQL diesen als „leer“ und ungleich NULL bewertet. Letzteres entspricht dem ANSI/ISO-SQL-Standard.

Wenn Oracle-Kompatibilität bei möglichst wenig Code-Anpassungen gewünscht ist, kann EnterpriseDB eine gute Wahl sein. EnterpriseDB vertreibt neben Tools für das freie PostgreSQL auch eine eigene PostgreSQL-Variante, die Oracle Features und Packages wie zum Beispiel UTL_FILE, UTL_MAIL oder DBMS_* Packages enthält.

Um einen „Hersteller-Support“ wie bei Oracle zu bekommen, bietet EnterpriseDB zudem einen professionellen Troubleshooting- und insbesondere auch Break/Fix-Support sämtlicher PostgreSQL-Varianten, Tools und Extensions durch eigene PostgreSQL-Entwickler.

Zur Bewertung analog SCT verwendet EnterpriseDB ein Webportal, das EDB Migration Portal *[6]*.

Diese beiden Blog-Artikel enthalten umfängliche Darstellungen von Unterschieden und Herausforderungen bei der Migration von Oracle zu PostgreSQL:

- *EnterpriseDB: „The Complete Oracle to Postgres Migration Guide: Move and Convert Schema, Applications and Data, EnterpriseDB“ [7]*
- *AWS: „Challenges When Migrating from Oracle to PostgreSQL—and How to Overcome Them“ [8].*

All diese kleinen und größeren Unterschiede sollten idealerweise bekannt und berücksichtigt worden sein.

Da meist nicht ausreichend viel Zeit zur Verfügung steht, brauchen Sie einen Mix aus kenntnisreicher Unterstützung, genauem Testen und sachgerechter Prüfung.

Fazit

Zum Schluss noch einmal der wichtige Hinweis:

„Testen, testen, testen! Lieber einmal zu viel getestet als einmal zu wenig.“

Denn ist die Applikation einmal umgeschaltet und zur Verwendung freigegeben worden, kann es schwierig werden, die in PostgreSQL geänderten Daten zurück nach Oracle zu spielen.

Wer im Vorfeld auf die Mahnung und den Vorschlag gehört hat, die Daten aus der Zielumgebung mittels AWS Database Migration Service, Oracle GoldenGate, Quest Shareplex oder anderer Methoden zurück in eine Kopie der Oracle-Datenbank zu replizieren, kann bei Bedarf mit bestenfalls kurzer Unterbrechung auf die Quelldatenbank zurückschalten, ohne neue Daten zu verlieren.

Wer mehr erfahren möchte kann im DOAG-Archiv weitere Beiträge von mir zu diesem Thema finden:

- DOAG-Webinar: „Aufgezeichnet: Migration Factory – Oracle to AWS RDS for PostgreSQL“ [9], bei dem ich näher auf die Bedienung von SCT eingehe
- Vortrag auf der DOAG-Konferenz 2022: *„Migrationen von Oracle (on-premises) nach PostgreSQL und Oracle RDS in der Cloud“ [10]*

Viel Spaß und Erfolg bei Ihrer Migration!

Quellen

[1] Star-Methode, Wikipedia, *https://de.wikipedia.org/wiki/Star-Methode*

[2] Andrea Windolph (2022): Roadmapping: Was und wofür? Ein kompakter Überblick, projekte-leicht-gemacht.de, *https://projekte-leicht-gemacht.de/blog/business-wissen/roadmapping/* Kapitel: „Unterschied zwischen Roadmapping und Projektplanung“ und Bildreferenz: *https://projekte-leicht-gemacht.de/blog/*

business-wissen/roadmap-beispiele/#vorlage
[3] Amazon Web Services AWS (2020): AWS Prescriptive Guidance – Migration strategy for relational databases, *https://docs.aws.amazon.com/prescriptive-guidance/latest/strategy-database-migration/qualify-workloads.html*
[4] Amazon Web Services AWS: Oracle to Aurora PostgreSQL Migration Playbook, *https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-postgresql-migration-playbook/chap-oracle-aurora-pg.tools.actioncode.html*
[5] Thom Brown (2023): How NULL and empty strings are treated in PostgreSQL vs Oracle, EnterpriseDB, *https://databaserookies.wordpress.com/2023/01/09/substr-functionality-differences-between-oracle-and-postgresql-what-you-need-to-know/*
[6] EDB Migration Portal, EnterpriseDB, *https://migration.enterprisedb.com/*
[7] Raghavendra Rao (2020): The Complete Oracle to Postgres Migration Guide: Move and Convert Schema, Applications and Data, EnterpriseDB, *https://edb.prod.acquia-sites.com/blog/the-complete-oracle-to-postgresql-migration-guide-tutorial-move-convert-database-oracle-alternative*
[8] Silvia Doomra (2018): Challenges When Migrating from Oracle to PostgreSQL—and How to Overcome Them, Amazon Web Services AWS, *https://aws.amazon.com/de/blogs/database/challenges-when-migrating-from-oracle-to-postgresql-and-how-to-overcome-them/*
[9] Christian Ballweg (2020), Aufgezeichnet: Migration Factory – Oracle to AWS RDS for PostgreSQL, DOAG, *https://www.doag.org/de/home/news/aufgezeichnet-migration-factory-oracle-to-aws-rds-for-postgresql*
[10] Christian Ballweg (2020), Migrationen von Oracle (on-premises) nach PostgreSQL und Oracle RDS in der Cloud, DOAG, *https://en.shop.doag.org/download.1db52e-a32027415831e77c86c82f1592_1/*

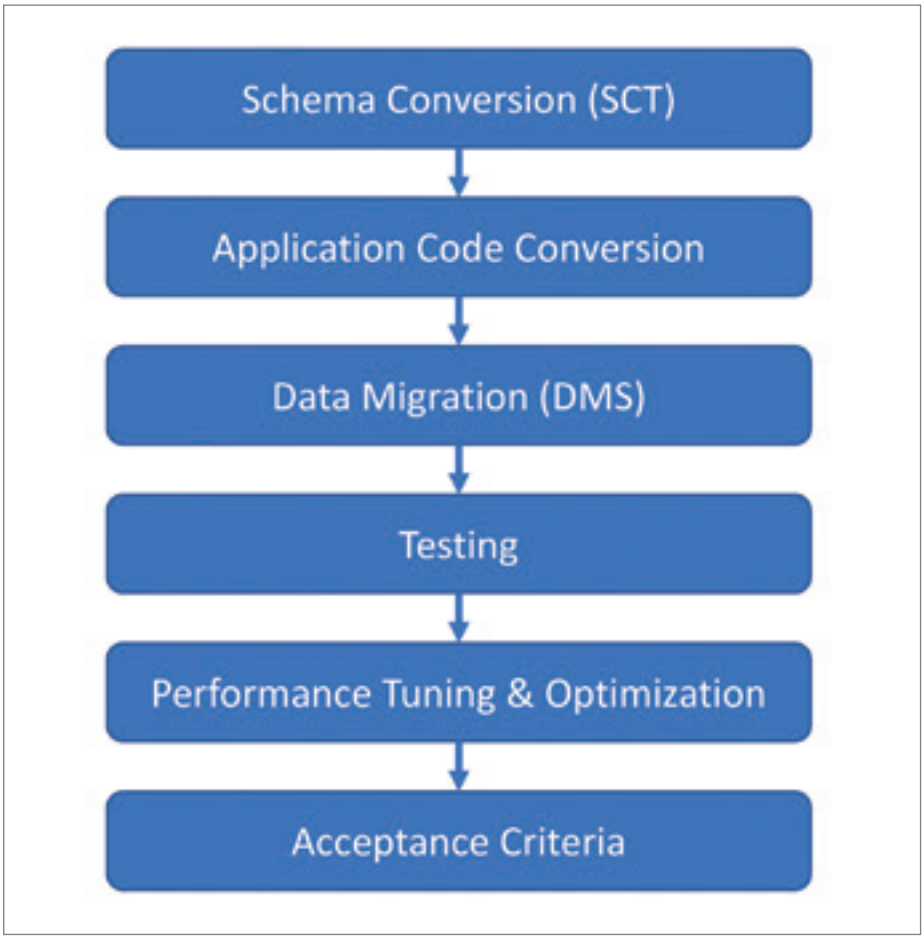


Abbildung 6: Migrationszyklus (Quelle: Christian Ballweg)

Über den Autor

Christian Ballweg ist seit mehr als 10 Jahren als Solution Architect bei der Opitz Consulting Deutschland GmbH beschäftigt. Seine über 20 Jahre Erfahrung in der IT erwarb er im Bereich Design, Aufbau und Betrieb von Oracle-Datenbankumgebungen mit den Schwerpunkten Data Warehouse, Hochverfügbarkeit, Performance Tuning, Upgrades und Minimal-Downtime-Migrationen sowie bei Konsolidierungs- und Lizenzoptimierungsprojekten. Er liebt es, sein Wissen als Berater, als Referent bei Fachkonferenzen oder als Autor von Fachartikeln weiterzugeben.

Christian Ballweg
Christian.Ballweg@opitz-consulting.com

Dynamisches SQL – Erfahrungen und Beispiele

Wilfried Eigl, selbstständig

Dynamisches SQL kann gefährlich werden, ist aber auch gelegentlich notwendig. Hier möchte ich meine Erfahrungen damit weitergeben und auch einige Beispiele erwähnen, bei denen ich intensiv dynamisches SQL eingesetzt habe.

Wenn man kein dynamisches SQL verwendet, hat man einige Vorteile. So werden die notwendigen Datenbankobjekte und Berechtigungen bei der Kompilierung überprüft und Abhängigkeiten validiert. Der Einsatz von dynamischem SQL wird notwendig, wenn zum Zeitpunkt der Kompilierung der Text des Statements noch unbekannt ist. Das ist der Fall, wenn Tabellen- oder Spaltennamen nicht bekannt sind oder die Anzahl der Bedingungen in der WHERE-Clause nicht feststehen.

Insbesondere SQL-Injection ist ein häufiges pauschales Argument gegen den Einsatz von dynamischem SQL. Wobei ich einige Szenarien sehe, in denen ich darin keine Probleme erkennen kann:

1. Tools, die nur von Administratoren genutzt werden, sehe ich als unkritisch, da dieser Personenkreis auch andere Methoden kennt, um an Informationen über die Datenbank zu kommen.
2. Bei Batch-Verarbeitungen, die keine Schnittstellen nach außen haben, sollte man in der Lage sein, über eine entsprechende Berechtigungsvergabe die Implementierung abzusichern.

Generell sollte man allerdings jeden Einzelfall darauf überprüfen, ob der Einsatz der erstellten Programme zu einer

Gefährdung führen kann. Doch es gibt auch Fälle, in denen man aus technischen Gründen nicht auf dynamisches SQL verzichten kann.

Wie dynamisches SQL erstellt und ausgeführt wird

Die Erstellung zusammengesetzter DDL-Statements ist unproblematisch. Bei den DML-Statements ist der SELECT-Befehl ein Problem, wenn man nicht weiß, wie viele Spalten zurückgegeben werden und welchen Datentyp diese haben werden. Hierbei kann man sich zum Teil damit helfen, per Create-Table-As-Select eine Tabelle als Ergebnis zu erstellen. Kennt man die Anzahl der Spalten und deren Datentyp, kann man eine entsprechende Variablenstruktur erstellen, die den Rückgabewert abspeichert. Enthält die Ergebnismenge mehrere Zeilen, so ist als Zielstruktur eine PL/SQL-Tabelle zu empfehlen, die per BULK LOAD (z. B. *siehe Abbildung 5*) befüllt wird. Man kann allerdings auch wie in *Abbildung 2* jeden Datensatz einzeln verarbeiten.

Zur Ausführung von dynamischem SQL gibt es mehrere Möglichkeiten.

Eine Möglichkeit bietet das ORACLE Package DBMS_SQL *(siehe Abbildung 1)*. In der Reihenfolge einer Ausführung enthält dieses Package folgende Funktionen:

- OPEN_CURSOR – erstellt einen Cursor und gibt als Funktionswert eine Cursor-ID zurück.
- PARSE – der Cursor-ID wird ein erstelltes SQL-Statement zugeordnet und beim Parsen des Statements Datenbankobjekte und Berechtigungen geprüft.
- BIND_VARIABLE, BIND_VARIABLE_PKG, BIND_ARRAY – Ersetzung von Bind-Variablen.
- DEFINE_COLUMN, DEFINE_COLUMN_LONG, DEFINE_ARRAY – Definition der Zielvariablen.
- EXECUTE – Ausführung des SQL-Statements.
- CLOSE_CURSOR – den Cursor wieder schließen.

Nicht immer werden alle Schritte benötigt. So werden BIND-Funktionen überflüssig, wenn keine Bind-Variablen benutzt werden. Bei DDL-Statements findet die Ausführung bereits beim PARSE statt, sodass alle weiteren Schritte obsolet sind.

Eine weitere Möglichkeit besteht in der Nutzung eines REF-Cursors in PL/SQL *(siehe Abbildung 2)*. Aber auch bei einem normalen expliziten Cursor oder auch dem impliziten Cursor einer FOR-Schleife kann das SQL-Statement aus mehreren Komponenten zusammengesetzt und ausgeführt werden.

Eine dritte Möglichkeit haben wir mit der Nutzung des PL/SQL-Statements EXECUTE IMMEDIATE *(siehe Abbildung 3)*. Damit kann ein SQL-Befehl oder ein PL/SQL-Block, der in einer Variable abgespeichert wurde, ausgeführt werden. Es können Bind-Variablen genutzt werden (USING); auch Rückgabewerte sind möglich in einzelnen Variablen (INTO) oder auch PL/SQL-Tabellen (BULK COLLECT) (*siehe* ORACLE Database PL/SQL Language Reference).

Schreibweisen im Code

Bei der Zusammensetzung des SQL-Statements kann man verschiedene Schreibweisen nutzen *(siehe Abbildung 4)*, die unterschiedliche Vor- und Nachteile haben.

Die einfachste Methode ist die Zusammensetzung eines Strings durch Konkatenieren (stmt1). Dabei besteht allerdings der Nachteil, dass man Hochkommas, die im Statement vorkommen, verdoppeln muss. Je nach Anzahl kann das die Lesbarkeit deutlich verschlechtern und es ist auch immer wieder eine beliebte Fehlerquelle. Einfacher wird die Handhabung von Hochkommas mit Alternative Quoting *(siehe Abbildung 4: stmt2, stmt3, stmt4)*. Hierbei wird der Text mit einem alternativen Zeichen (hier {...}) geklammert, sodass innerhalb des Textes das Hochkomma normal benutzt werden kann. Diese Vorgehensweise kann man auch

```
DECLARE
  e_tab varchar2(30) := 'hr.employees';
  emp_name varchar2(20) :=  'Ernst';
  cursor_id number;
  rows_processed number;
BEGIN
  cursor_id := DBMS_SQL.OPEN_CURSOR;
  DBMS_SQL.PARSE(cursor_id,'delete from '||e_tab||' where last_name = :x',1);
  DBMS_SQL.BIND_VARIABLE(cursor_id, ':x', emp_name);
  rows_processed := DBMS_SQL.EXECUTE(cursor_id);
  DBMS_SQL.CLOSE_CURSOR(cursor_id);
  commit;
  dbms_output.put_line('Alles OK! '|| rows_processed ||' Zeilen deleted');
END;
```

Abbildung 1: Dynamisches. SQL mit DBMS_SQL (Quelle: Wilfried Eigl)

```
DECLARE
  type EmpCurTyp is ref cursor;
  emp_cursor EmpCurTyp;
  e_tab varchar2(30) := 'hr.employees';
  emp_record hr.employees%ROWTYPE;
  stmt_str varchar2(200);
BEGIN
  stmt_str := 'select * from '||e_tab||' where job_id = :j';
  OPEN emp_cursor FOR stmt_str USING 'SA_MAN';
  LOOP
    FETCH emp_cursor INTO emp_record;
    dbms_output.put_line(emp_record.last_name||','||
   emp_record.first_name);
    EXIT WHEN emp_cursor%NOTFOUND;
  END LOOP;
CLOSE emp_cursor;
END;
```

Abbildung 2: Dynamisches SQL mit REF-Cursor (Quelle: Wilfried Eigl)

```
DECLARE
  e_tab varchar2(30) := 'hr.employees';
  emp_name varchar2(20) :=  'Ernst';
  stmt varchar2(200);
  rows_processed number;
BEGIN
  stmt := 'delete from '||e_tab||' WHERE last_name = :x';
  dbms_output.put_line(stmt);
  execute immediate stmt using emp_name;
  commit;
END;
```

Abbildung 3: Dynamisches. SQL mit EXECUTE IMMEDIATE (Quelle: Wilfried Eigl)

```
DECLARE
...
  tab varchar2(30) := 'hr.employees';
  nam varchar2(30) := '''Smith''';
BEGIN
  stmt1 := 'delete from '||tab||' where last_name=''Smith''';
  stmt2 := q'{delete from employees where last_name='Smith'}';
  stmt3 := 'delete from '||tab||q'{ where last_name ='Smith'}';
  stmt4 := q'{delete from #tab# where last_name = 'Smith'}';
  stmt4 := replace(stmt4,'#tab#',tab);
  stmt5 := 'delete from #tab# where last_name = ''Smith''';
  stmt5 := replace(stmt5,'#tab#',tab);
  stmt6 := 'delete from #tab# where last_name = #nam#';
  stmt6 := replace(stmt6,'#tab#',tab);
  stmt6 := replace(stmt6,'#nam#',nam);
...
END;
```

Abbildung 4: Verschiedene Schreibweisen (Quelle: Wilfried Eigl)

```
-- SQL-Injection liefert alle Datensätze;
DECLARE
  type erg_type is table of varchar2(20);
  erg erg_type;
  stmt varchar2(200);
--  param varchar2(20) := 'Zlotkey';
  param varchar2(20) := '1'' or (1=1) --';
BEGIN
  stmt := 'select first_name
              from hr.employees
             where last_name='''||param||'''';
  execute immediate stmt bulk collect into erg;
END;
```

Abbildung 5: SQL-Injection ohne Bind-Variable (Quelle: Wilfried Eigl)

```
--- SQL-Injection liefert kein Ergebnis
DECLARE
  type erg_type is table of varchar2(20);
  erg erg_type;
  stmt varchar2(200);
  param varchar2(20) := '1'' or (1=1) --';
BEGIN
  stmt := 'select first_name from hr.employees where last_name=:n';
  execute immediate stmt bulk collect into erg using param;
END;
```

Abbildung 6: SQL-Injection mit Bind-Variable (Quelle: Wilfried Eigl)

```
select SYS_CONTEXT ('USERENV', 'SESSION_USER')
      ,SYS_CONTEXT ('USERENV', 'OS_USER')
  from  dual;
```

Abbildung 7: Aktuellen User abfragen (Quelle: Wilfried Eigl)

```
-- Function EXEC_STMT - führt ein SQL-Statement aus
  v_rc := LOG.EXEC_STMT(v_sql,$$PLSQL_UNIT_OWNER
                             ,$$PLSQL_UNIT
                             ,$$PLSQL_LINE
                             ,$$PLSQL_UNIT_TYPE
                             ,USER);
```

Abbildung 8: Funktionsaufruf (Quelle: Wilfried Eigl)

```
-- Function EXEC_STMT_INTO_VARCHAR2
-- SQL_Statement mit Rückgabewert
  v_varchar := LOG.EXEC_STMT_INTO_VARCHAR2(v_sql
                                          ,$$PLSQL_UNIT_OWNER
                                          ,$$PLSQL_UNIT
                                          ,$$PLSQL_LINE
                                          ,$$PLSQL_UNIT_TYPE
                                          ,USER);
--(auch für NUMBER,CLOB,VARCHARLIST,NUMBERLIST möglich)
```

Abbildung 9: Funktionsaufruf mit Rückgabewert (Quelle: Wilfried Eigl)

nur auf Teilstrings anwenden *(siehe Abbildung 4: stmt3)*. Die Lesbarkeit des SQL-Statements kann auch dadurch verbessert werden, dass man auf Konkatenieren verzichtet und stattdessen Platzhalter (hier #tab#) einführt, die dann mithilfe einer REPLACE-Funktion durch den korrekten Wert ersetzt werden *(siehe Abbildung 4: stmt5, stmt6)*.

Ich habe es immer als angenehm empfunden, wenn nicht zu viele verschiedene Schreibweisen innerhalb eines Projektes benutzt werden, da man sich dann beim Lesen des Codes nicht immer wieder umstellen muss. Dafür kann ein kurzer Style-Guide nützlich sein.

SQL-Injection

Bei SQL-Injection wird ein Parameter für die Ausführung eines dynamischen SQL-Statements so geschickt ausgewählt, dass die Anwendung zusätzliche Informationen über die Datenbank zurückliefert. In *Abbildung 5* sehen wir dazu ein Beispiel. Eigentlich ist der Code dafür gedacht, Vornamen zum Nachnamen eines Mitarbeiters zu liefern (der Einfachheit halber habe ich hier den Parameterwert im Code definiert, er wird jedoch in der Regel über eine Anwendungsoberfläche von einem Benutzer eingegeben werden). Die auskommentierte Zeile enthält einen Namen, der im HR-Schema tatsächlich vorhanden ist und daher ein Ergebnis liefern würde. Wird nun der Parameter verändert wie in diesem Beispiel, so werden alle Namen der Tabelle ausgegeben.

Diese Manipulation kann man verhindern, indem man mit Bind-Variablen arbeitet *(siehe Abbildung 6)*. In diesem Fall wird nach einem Namen gesucht, der dem manipulierten String entspricht und kein Ergebnis liefert.

Ausführung zentralisieren

Die Ausführung der dynamisch erstellten SQL-Statements implementiere ich in der Regel in ein eigenes Package. Diese Zentralisierung soll mehrere Bedingungen erfüllen.

- Einheitliche Ausführung der Statements
- Führung einer LOG-Tabelle des ausgeführten Codes
- Zentrale und einheitliche Fehlerbehandlung
- Speicherung des Ausführungsorts (Package/Function/Zeile)
- Speicherung der Ausführungszeit (Beginn/Ende/Dauer)
- Speicherung des ausführenden Users
- Möglichkeit einer Abschaltung der LOG-Funktionalität
- Rückgabewerte von SELECT-Statements sollen möglich sein

Um die Informationen zum Ausführungsort und ausführenden User im zentralen Package verarbeiten zu können, müssen diese beim Aufruf der Funktion als Parameter übergeben werden. Dazu kann man Predefined Inquiry Directives nutzen, die Informationen über die aktuelle Position der Codeausführung enthalten. Die Bezeichner von Inquiry Directives beginnen immer mit $$.

- $$PLSQL_UNIT_OWNER – Owner des aktuellen Codes
- $$PLSQL_UNIT – Name der aktuellen Codeeinheit (Package/Function)
- $$PLSQL_LINE – Zeilennummer innerhalb der Codeeinheit
- $$PLSQL_UNIT_TYPE – Typ der Codeeinheit (Package/Function ...)

Den Namen des ausführenden Users erhält man entweder über die SQL-Konstante USER wie zum Beispiel im Statement „select USER from dual;“ oder über die Systemfunktion SYS_CONTEXT *(siehe Abbildung 7)*. Dabei bietet die Systemfunktion noch weitere Informationen über die aktuelle Session, die man ebenfalls nutzen könnte wie zum Beispiel den Betriebssystem-User. Bei der Nutzung von technischen Usern zur Anmeldung an der Datenbank kann der Betriebssystem-User nützlicher sein.

Damit können wir die Funktionsaufrufe für die Ausführung des erstellten SQL-Statements einheitlich gestalten *(siehe Abbildungen 8 und 9)*.

Leider liefert der Wert von $$PLSQL_UNIT nur den Namen des aktiven Packages, aber nicht den Namen der Funktion oder Prozedur, die aktuell ausgeführt wird. Dieser muss selbst ermittelt werden mithilfe der Werte $$PLSQLOWNER, $$PLSQL_UNIT und $$PLSQL_LINE. Entweder man nutzt die DD-View ALL_SOURCE oder die etwas unbekanntere View ALL_IDENTIFIERS. Letztere View enthält auch nicht immer Werte. Voraussetzung dafür ist das Setzen des Parameters PLSCOPE auf den Wert IDENTIFIERS: ALL. Prüfen kann man diese Einstellung mit dem Statement:

```
select * from user_plsql_object_
settings;
```

Gesetzt wird der Wert mit dem Befehl:

```
alter session set PLSCOPE_
SETTINGS='IDENTIFIERS:ALL';
```

Wenn der Parameter gesetzt ist, dann werden bei jeder Kompilierung die Identifier des Codes registriert.

Hier sehen wir, dass ein Package mit dem Namen PKG existiert. In diesem Package wird in Zeile 3 eine Procedure definiert mit dem Namen PROC. Danach eine Variable V_RC etc. Die Zeilennummern der Definitionen können genutzt werden, um mithilfe der aktuellen Zeilennummer die aktuell ausgeführte Funktion oder Procedure innerhalb des Packages zu ermitteln.

Nun fehlt noch die Möglichkeit, die LOG-Funktion abschalten zu können, wenn sie nicht benötigt wird. Dazu kann man die bedingte Kompilierung ($IF ...) nutzen. Falls generell zum Beispiel auf Produktion das LOG abgeschaltet werden soll, dann können wir den Namen mit SYS_CONTEXT ermitteln und prüfen.

Eine zweite Möglichkeit besteht in der Definition einer User-defined Inquiry Directive, deren Wert dann bei der bedingten Kompilierung abgefragt wird. Erstellt wird diese Variable mit dem Befehl:

```
Alter session set PLSQL_
CCFLAGS='SQLLOG:TRUE';
```

Mit diesem Statement erstellt man eine Variable mit dem Namen $$SQLLOG und weist dieser den booleschen Wert TRUE zu. In Verbindung mit IF$ kann man einen Bereich im PL/SQL Package definieren, der nur dann kompiliert wird, wenn der Wert der Variable TRUE ist.

Den aktuellen Wert der User-defined Inquiry Directives können wir in der View ALL_PLSQL_OBJECT_SETTINGS View abfragen.

```
select object_name,name,type,usage,line
  from all_identifiers
 where object_name like 'PKG'
   and object_type='PACKAGE BODY'
 order by line;
```

Abbildung 10: Abfrage ALL_IDENTIFIERS (Quelle: Wilfried Eigl)

OBJECT_NAME	NAME	TYPE	USAGE	LINE
PKG	PKG	PACKAGE	DEFINITION	1
PKG	PROC	PROCEDURE	DEFINITION	3
PKG	V_RC	VARIABLE	DECLARATION	5
PKG	NUMBER	NUMBER DATATYPE	REFERENCE	5
PKG	LOG	PACKAGE	REFERENCE	8
PKG	LOGTEXT	FUNCTION	CALL	8
PKG	USER	FUNCTION	CALL	9

Abbildung 11: View ALL_IDENTIFIERS (Quelle: Wilfried Eigl)

```
select SYS_CONTEXT ('USERENV', 'DB_NAME')
      ,SYS_CONTEXT ('USERENV', 'DB_UNIQUE_NAME')
      ,SYS_CONTEXT ('USERENV', 'HOST')
      ,SYS_CONTEXT ('USERENV', 'SERVER_HOST')
      ,SYS_CONTEXT ('USERENV', 'INSTANCE_NAME')
  FROM DUAL;
```

Abbildung 12: Bezeichner einer DB (Quelle: Wilfried Eigl)

```
if$ $$SQLLOG then
...
end if;
```

Abbildung 13: Bedingte Kompilierung (Quelle: Wilfried Eigl)

```
1 select name,type,plsql_warnings,plsql_ccflags,plscope_settings
2   from all_plsql_object_settings where name='PKG';
```

Abfrageergebnis

Alle Zeilen abgerufen: 2 in 0,23 Sekunden

	NAME	TYPE	PLSQL_WARNINGS	PLSQL_CCFLAGS	PLSCOPE_SETTINGS
1	PKG	PACKAGE	DISABLE:ALL	SQLLOG:TRUE	IDENTIFIERS:ALL
2	PKG	PACKAGE BODY	DISABLE:ALL	SQLLOG:TRUE	IDENTIFIERS:ALL

Abbildung 14: PLSQL_OBJECT_SETTINGS (Quelle: Wilfried Eigl)

USER	UNIT_NAME	UNIT_LINE	UNIT_PROC	SQLTEXT	STATUS	...	ANFANG	ENDE	DAUER
WILLI	PROC1	5	TYPE: PROCEDURE	(null)	0	...	14.09...	14.09...	+0...
WILLI	PKG	5	GET PROCNAME	select att1 from...	0	...	14.09...	14.09...	+0...
WILLI	PKG	5	PROC	select att1	0	...	14.09...	14.09...	+0...
WILLI	PKG	5	PROC	insert into	1	...	14.09...	14.09...	+0...
WILLI	PKG	8	PROC	insert into	0	...	14.09...	14.09...	+0...

Abbildung 15: LOG-Tabelle (Quelle: Wilfried Eigl)

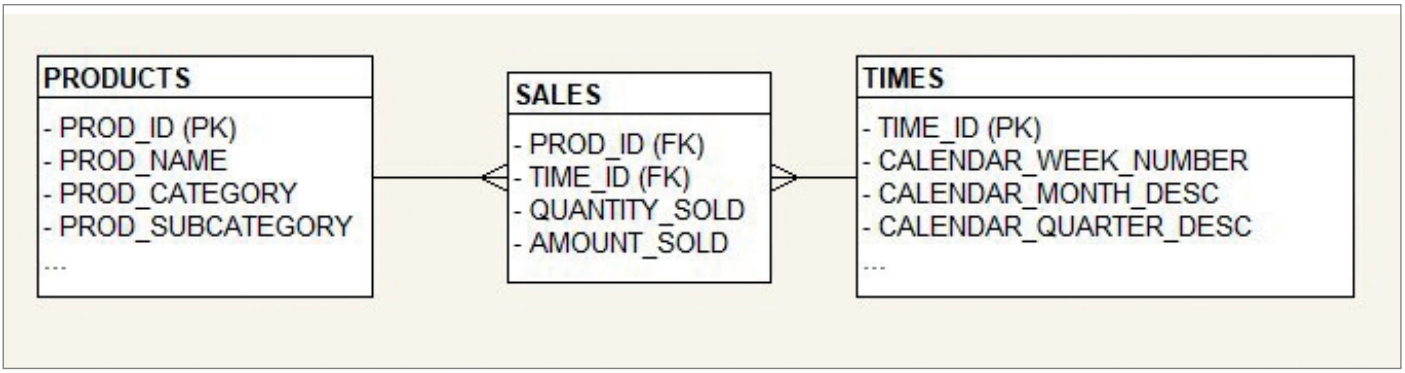


Abbildung 16: Beispieltabellen aus dem Schema SH (Quelle: Wilfried Eigl)

```
select p.prod_name
      ,sum(s.quantity_sold) quantity_98_01
  from sh.products p
      ,sh.sales s
      ,sh.times t
 where s.prod_id=p.prod_id
   and s.time_id=t.time_id
   and t.calendar_month_desc = '1998-01'
 group by p.prod_name;
```

Abbildung 17: Zu generierendes SQL-Statement (Quelle: Wilfried Eigl)

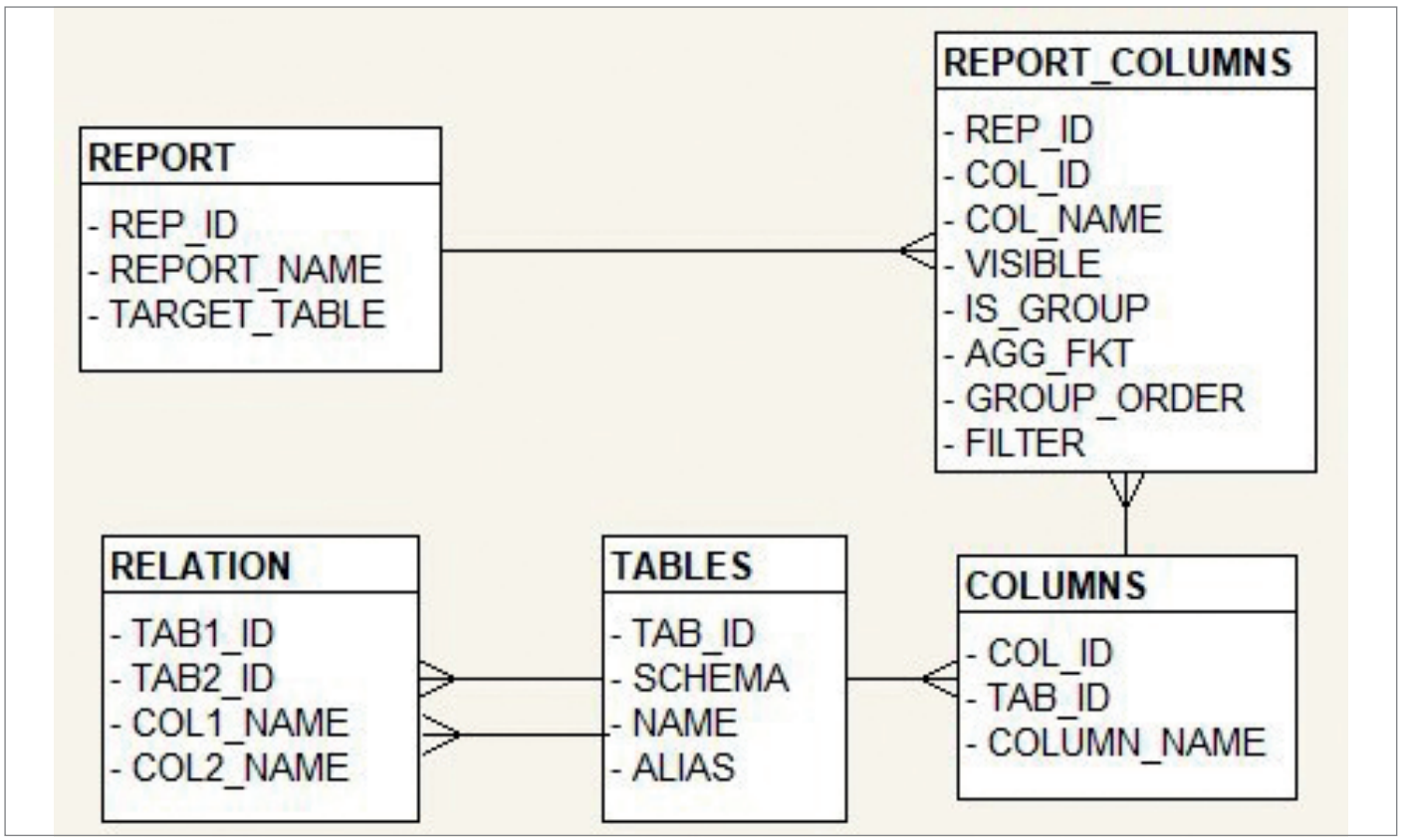


Abbildung 18: Tabellen zum Beispiel SQL-Abfrage (Quelle: Wilfried Eigl)

TABLES

TAB_ID	SCHEMA	NAME	ALIAS
1	sh	products	p
2	sh	sales	s
3	sh	times	t

COLUMNS

COL_ID	TAB_ID	COLUMN_NAME
1	1	prod_id
2	1	prod_name
3	1	prod_category
4	1	prod_subcategory
10	2	prod_id
11	2	time_id
12	2	quantity_sold
13	2	amount_sold
20	3	time_id
21	3	calendar_week_number
22	3	calendar_month_desc
23	3	calendar_quarter_desc

RELATION

TAB1_ID	TAB2_ID	COL1_NAME	COL2_NAME
2	1	PROD_ID	PROD_ID
2	3	TIME_ID	TIME_ID

REPORT

REP_ID	REPORT_NAME	TARGET_TABLE
1	Test	Test_Ergebnis

REPORT_COLUMNS

REP_ID	COL_ID	COL_NAME	VISIBLE	IS_GROUP	AGG_FKT	GROUP_ORDER	FILTER
1	2	(null)	J	(null)	(null)	1	(null)
1	12	quantity_98_01	J	J	SUM(#tab#)	(null)	(null)
1	22	(null)	N	(null)	(null)	(null)	1998-01

Abbildung 19: Befüllte Tabellen (Quelle: Wilfried Eigl)

```
-- Liste der COL_IDs die benötigt werden
select '('||listagg(col_id,',')||')' from report_columns where rep_id=1;
-- (2,12,22)
--
-- Spaltenliste
select listagg(t.alias||'.'
                    ||decode(is_group,'J',replace(agg_fkt,'#tab#',column_name),column_name)
                    ||' '
                    ||rc.col_name,',')
  from columns c
      ,tables t
      ,report_columns rc
 where c.col_id in (2,12,22)
   and c.col_id=rc.col_id
   and t.tab_id=c.tab_id
   and rc.visible='J'
;
-- p.prod_name ,s.SUM(quantity_sold) quantity_98_01
--
-- Tabellenliste
select listagg(t.schema||'.'||t.name||' '||t.alias,',')
  from columns c
      ,tables t
 where c.col_id in (2,12,22)
   and t.tab_id=c.tab_id
;
-- sh.products p,sh.sales s,sh.times t
--
-- Filter-Kriterium
select listagg(t.alias||'.'||column_name||'='''||rc.filter||'''',',')
  from columns c
      ,tables t
      ,report_columns rc
 where c.col_id in (2,12,22)
   and c.col_id=rc.col_id
   and t.tab_id=c.tab_id
   and rc.filter is not null
;
-- t.calendar_month_desc='1998-01'
```

Abbildung 20: Bestandteile des Select-Statements 1 (Quelle: Wilfried Eigl)

Jetzt haben wir alle Informationen zusammen, um die Funktionen für das zentrale Package zu erstellen. Die Bestandteile werden hier in Pseudo-Code aufgelistet.

- PRAGMA AUTONOMOUS_TRANSACTION -- wie bei jeder LOG-Funktion
- $IF $$SQLLOG -- LOG-Tabelle wird nur beschrieben, wenn gewünscht
- INSERT into LOG-Tabelle
  SQL-Statement
  UNIT_OWNER, UNIT_NAME, UNIT_LINE, USER, STARTZEIT
  Prozedur- oder Funktionsname wird ermittelt aus OWNER, NAME und LINE
  Rückgabe der ROWID
- Ende der bedingten Kompilierung
- EXECUTE IMMEDIATE SQL-Statement
- Bedingte Kompilierung – UPDATE der ROWID mit ENDZEIT, DAUER, STATUS=0
- Bei Fehler; Bedingte Kompilierung: UPDATE der ROWID mit ENDZEIT, DAUER, STATUS=1, MELDUNG, DBMS_UTILITY.CALL_STACK, DBMS_UTILITY.ERROR_STACK
- Bei Fehler: sonstiges: RAISE APPLICATION ERROR

Beispiel SQL-Abfrage

Es soll eine Anwendung erstellt werden, die einem Anwender erlaubt, einen Report zu erstellen und abzuspeichern, indem lediglich die benötigten Attribute aus einer zur Verfügung gestellten Liste ausgewählt werden. Das zur Erstellung des Reports benötigte SQL-Statement wird automatisch generiert. Als Basis dafür nehmen wir hier das Schema SH mit den Tabellen SALES, TIMES und PRODUCTS *(siehe Abbildung 16)*.

Sehen wir uns ein einfaches SELECT-Statement auf diese Struktur an. Es soll die Verkäufe eines Quartals über alle Produkte ermitteln. Dazu benötigen wir Informationen vom Anwender über die gewählten Attribute sowie darüber, ob eine Gruppierung vorgenommen werden soll. Doch ebenso werden Informationen über die Struktur, wie einen Alias für die gewählten Tabellen und die Verbindungen der Tabellen

```
-- Where-Clause
--select listagg(t.alias||'.'||c.column_name||' '||t.alias,',')
select listagg(t1.alias||'.'||t1.name||'='||t2.alias||'.'||t2.name,' and ')
  from columns c
      ,tables t1
      ,tables t2
      ,relation r
 where c.col_id in (2,12,22)
   and t1.tab_id=c.tab_id
   and r.tab1_id=t1.tab_id
   and r.tab2_id=t2.tab_id
;
-- s.sales=p.products and s.sales=t.times
--
-- Group by Expression
select listagg(t.alias||'.'||column_name,',')
  from columns c
      ,tables t
      ,report_columns rc
 where c.col_id in (2,12,22)
   and c.col_id=rc.col_id
   and t.tab_id=c.tab_id
   and rc.group_order is not null
 order by rc.group_order
;
-- p.prod_name
```

Abbildung 21: Bestandteile des Select-Statements 2 (Quelle: Wilfried Eigl)

```
select uc.constraint_name
      ,'ALTER TABLE '||uc.table_name||' ADD CONSTRAINT '||uc.constraint_name
      ||' primary key ('
      ||LISTAGG(ucc.column_name, ',') WITHIN GROUP (ORDER BY ucc.position)
                                      OVER (partition by uc.constraint_name)
      ||')' SQL_Statement
  FROM user_constraints uc
      ,user_cons_columns ucc
 WHERE uc.constraint_type='P'
   and uc.constraint_name=ucc.constraint_name;
```

ergebnis

SQL | Alle Zeilen abgerufen: 9 in 0,021 Sekunden

CONSTRAINT_NAME	SQL_STATEMENT
CHANNELS_PK	ALTER TABLE CHANNELS ADD CONSTRAINT CHANNELS_PK primary key (CHANNEL_ID)
COUNTRIES_PK	ALTER TABLE COUNTRIES ADD CONSTRAINT COUNTRIES_PK primary key (COUNTRY_ID)
CUSTOMERS_PK	ALTER TABLE CUSTOMERS ADD CONSTRAINT CUSTOMERS_PK primary key (CUST_ID)
PRODUCTS_PK	ALTER TABLE PRODUCTS ADD CONSTRAINT PRODUCTS_PK primary key (PROD_ID)
PROMO_PK	ALTER TABLE PROMOTIONS ADD CONSTRAINT PROMO_PK primary key (PROMO_ID)

Abbildung 22: Generierung der PK-Constraint-Befehle (Quelle: Wilfried Eigl)

über Fremdschlüssel, benötigt *(siehe Abbildung 17)*.

Die Definition eines Alias für jede Tabelle ergibt Sinn, damit die Statements nicht zu groß werden. Die Erstellung der Attributliste kann mit der Funktion LISTAGG recht einfach gemacht werden, allerdings nur bis zu einer bestimmten Länge. Wird der zu erzeugende String länger, so muss man sich mit der Funktion XMLAGG behelfen, was weitaus komplizierter ist.

In den Tabellen RELATION, TABLES und COLUMNS wird die Struktur der Daten abgespeichert. Ähnliche Informationen könnte man auch aus dem Data Dictionary holen. Allerdings ist hier noch ein ALIAS für jede Tabelle abgespeichert und die Verbindung zwischen den Tabellen ist wesentlich einfacher in der Tabelle RELATION abgelegt.

Die Informationen über den Report und das zu generierende SELECT-Statement befinden sich in den Tabellen REPORT und REPORT_COLUMNS. In REPORT befindet sich der Name einer Zieltabelle für ein CTAS-Statement. Die Spalten für den Aufbau des SELECT-Statements befinden sich in der Tabelle REPORT_COLUMNS. Hier ist zu jedem ausgewählten Attribut abgespeichert, wie damit umgegangen werden soll *(siehe Abbildung 18)*.

In *Abbildung 19* sieht man die Befüllung der Tabellen für dieses kurze Beispiel. Daraus können nun die Bestandteile des SQL-Statements in *Abbildung 17* erstellt werden. Der Code für die einzelnen Bestandteile befindet sich in den *Abbildungen 20 und 21*.

Beispiel Constraints verwalten

Die Idee zu einer eigenen Verwaltung von Constraints kam mir in einem Projekt, als immer wieder Constraints verschwunden sind. Offensichtlich gab es mindestens eine Person, die solche Dinge als lästig empfand und es als einen pragmatischen Ansatz sah, diese einfach kommentarlos zu entfernen. Ich habe dies immer erst dann bemerkt, wenn es Fehlermeldungen bei einem Batchlauf gab. Als Konsequenz daraus habe ich mir SQL-Befehle zur Erstellung der Constraints aus dem Data Dictionary erzeugt und in einer eigenen Tabelle abgespeichert. Damit besitzt man dann die Möglichkeit, vor dem Start einer größeren Verarbeitung die Inhalte des Data Dictionary gegenüber der eigenen Tabelle zu prüfen und damit die Existenz der Constraints zu testen. In *Abbildung 22* sehen wir ein SQL-Statement, mit dem die SQL-Befehle zur Erstellung der bereits existierenden Primary Keys erzeugt werden können. Da die Informationen dazu aus dem bestehenden System kommen, erübrigt sich hier eine Prüfung der Sicherheit.

Fazit

Dynamisches SQL zu nutzen kann Vorteile bringen unter der Voraussetzung, dass man sich zuvor auch über die Sicherheit Gedanken macht. Das sollte jedoch für die gesamte Programmentwicklung gelten. Ich sehe keinen Grund dafür, diese Möglichkeit generell zu verteufeln. Denn: *„Es kommt darauf an, was man daraus macht!“*

Wilfried Eigl
info@ithood.de

BUSINESS

NEWS
03/2023

Postpandemische Arbeitswelten

Jetzt alles besser? Arbeitswelten nach Corona

Steffen Reißig, Robotron

Was haben wir in der Arbeitswelt aus Corona gelernt? Wie wurden die damit verbundenen Chancen genutzt? Welche Änderungen haben sich ergeben? Eine Retrospektive aus Sicht des Softwareherstellers und Systemhauses Robotron Datenbank-Software GmbH.

Als langjähriges DOAG-Mitglied und ebenso langjähriger Oracle-Partner war und ist uns Online-Arbeit mit Kunden, ob im Projekt oder im Service, schon viele Jahre vertraut. Diese Erfahrung und eine frühzeitige Serviceorientierung hin zu unseren Kunden hat die aus der Corona-Pandemie folgenden Änderungen deutlich vereinfacht. Aber der Reihe nach:

Neue Realitäten – Corona, Bazooka und Homeoffice

Anfang 2020 kamen die ersten Meldungen aus China, dass sich dort eine neue Infektionskrankheit sehr schnell ausbreitete und viele Menschen teilweise schwer erkrankten. Im März 2020 erfasste Corona Deutschland. Immer mehr Infektionen, schnellere Ansteckungen, die Krankhäuser füllten sich mit Patienten und erste Todesfälle wegen Corona verunsicherten vollends. Das öffentliche Leben wurde schrittweise eingeschränkt, erste Lockdowns mit Arbeit im Homeoffice folgten, Lieferketten und die Wirtschaft begannen einzubrechen und Bundesfinanzminister Scholz kündigte Mitte März 2020 ein umfassendes Corona-Hilfspaket an: „Das ist die Bazooka, mit der wir das Notwendige jetzt tun." Grenzen wurden geschlossen und von Kontaktverboten war in dramatischen Medieninformationen zu lesen. Hamsterkäufe folgten, in Deutschland wurde das Toilettenpapier knapp (was ich meiner amerikanischen Verwandtschaft nicht erklären konnte) und in Frankreich sollen Rotwein und Kondome schwer erhältlich gewesen sein (das musste ich der amerikanischen Verwandtschaft nicht erklären).

Erste Umstellungen

Die ersten Einschränkungen im Arbeitsleben kamen ebenfalls schnell. In unserem Unternehmen entschieden wir uns im März 2020 für Homeoffice als in dieser Situation bevorzugte Arbeitsform. Nur wenige Mitarbeiter der eigenen ITK arbeiteten noch in ihren Büros und das mit sehr viel persönlichem Einsatz. Galt es doch, die eigene IT-Infrastruktur innerhalb kürzester Zeit für mehr als 500 Mitarbeiter unserer Firmengruppe in Deutschland, aber auch in der Schweiz, in Tschechien und Neuseeland auf vollständigen Remote Access umzustellen – ein beispielloser und zum Glück sehr erfolgreicher Kraftakt!

Mit einem Mal waren Kundentreffen vor Ort nicht mehr möglich und Videokonferenzen versuchten anfangs mehr schlecht als recht, diese zu ersetzen. Bisher weniger beachtete Softwarepakete wie Zoom und Webex wurden populär. Galt es doch, die Verbin-

Abbildung 1: Robotron Haus (Quelle: Steffen Reißig, Robotron)

Abbildung 2: Robotron Kantine (Quelle: Steffen Reißig, Robotron)

dung zu Kunden und eigenen Mitarbeitern zu halten, was organisatorisch anfangs schwierig war. Schnell zeigte sich, dass die reduzierte und neue Kommunikation eine der größten Herausforderungen der kommenden Monate und Jahre werden würde.

Überall dort, wo persönlicher Kontakt und der Aufbau von Beziehungen gefragt waren, bedurfte die Umstellung besonderer Aufmerksamkeit. Bei Bewerbungsgesprächen und beim Onboarding neuer Mitarbeiter war es uns wichtig, die neuen Teammitglieder durch Mentoren sowie eine schnelle virtuelle und persönliche Integration in die Arbeitswelt unserer Teams zu unterstützen und zu begleiten. Im Verlauf der Zeit gelang dies immer besser.

Arbeiten in Krisenzeiten

Die folgenden Monate bis zum Jahresende 2022 waren geprägt von Lockdowns, unterbrochen von zaghaften Lockerungen, ersten wieder persönlichen Kundenkontakten, und das immer mit der Motivation, sich den Gegebenheiten anzupassen und dennoch ein Stück Normalität zurückzuerlangen. Kennzeichnend für diese Zeit war die Freude und Zuversicht über jeden noch so kleinen Schritt wieder in Richtung persönlicher Kunden- und Geschäftskontakte. Gleiches traf auch auf den direkten Austausch in den Teams zu.

Neue Arbeitswelten und Herausforderungen

Wir testeten mehre etablierte Softwarepakete für Kommunikation und Kollaboration, bis sich eine Kombination aus integrierten Lösungen und Eigenentwicklungen für uns als Mittel der Wahl für die Kommunikation und Zusammenarbeit intern und mit Kunden durchgesetzt hatte. Parallel wurde Kommunikation in fachlich spezialisierten Online Communities und geschäftlich orientierten sozialen Netzwerken wie LinkedIn immer wichtiger. Heute sind diese Communities eine weitere gute Methode geworden, um Geschäftskontakte aufzubauen und zu pflegen sowie die notwendige Sichtbarkeit auf die eigenen geschäftlichen Leistungen zu generieren. Auch für die Personalgewinnung sind Online-Kanäle und soziale Netzwerke nahezu unverzichtbar geworden.

Mit den sich in schneller Abfolge verbessernden Kommunikations- und Kollaborations-Tools veränderten sich auch die Anforderungen an die Arbeitsorganisation. Schnell waren die Kalender von früh bis in den späten Nachmittag mit nahtlos aufeinander folgenden Terminen gefüllt, ohne dass Zeitfenster zur Vor- und Nachbereitung oder Bearbeitung berücksichtigt wurden. Hier musste dringend nachgebessert und Regelwerke für Online-Meetings geschaffen werden, die Ablauf, Aufzeichnung und Verbindlichkeit gewährleisteten.

Eine besondere Herausforderung stellte beziehungsweise stellt die technische und organisatorische Absicherung der Online-Kommunikation und -Arbeit dar. Es zeigte sich auch, dass Security-Themen bei uns selbst, wie auch bei unseren Kunden, immer mehr Aufmerksamkeit benötigen, woraus sich in unserem Unternehmen ein eigenes Leistungsspektrum entwickelt hat.

Neue soziale Herausforderungen

Wir haben schon immer viel Wert auf eine gute Arbeitsatmosphäre, ein starkes Teamgefühl, auf Kooperation und Wissensaustausch bis hin zu gemeinsamen Aktivitäten außerhalb des Arbeitslebens gelegt. Das Gespräch in der Kaffeeküche, gemeinsame Grillabende, Firmenevents und -feiern gehören genauso dazu wie das große Angebot im eigenen Robotron-Sportverein mit verschiedensten Sektionen.

Das war natürlich 2020 bis Mitte 2022 stark behindert. Neue Ideen waren also gefragt und wurden auch schnell umgesetzt. Die virtuelle, gemeinsame Kaffeepause gehört ebenso wie das virtuelle Team Event dazu. Ich kann mich noch gut an die von lokalen Wein- und Whiskey-Händlern durchgeführten abendlichen Verkostungen mit Videokonferenz und Verkostungspaketen erinnern. Wer in mehreren Teams aktiv war, konnte sich so schnell zum Wein- oder Whiskey-Experten entwickeln. Ebenso gut in Erinnerung geblieben sind mir kurzweilige, virtuelle Spielabende.

Auch der Sportverein mit seinen ca. 250 Mitgliedern musste das Angebot zeitweise ein- beziehungsweise umstellen. So wurden Yoga, funktionales Training und Rückenschule kurzerhand ins Freie verlegt oder live online angeboten. Da wir nunmal den größten Teil unserer Arbeit sitzend er-

ledigen, war das Bewegungsangebot in den Lockdown-Zeiten besonders wichtig. Inzwischen wird wieder im hauseigenen Sportraum geturnt, teilweise sogar hybrid.

Nach zwei Jahren Konzentration auf Online-Events zeigte sich, dass persönliche Treffen und Gespräche damit nicht zu ersetzen waren oder zu ersetzen sind.

Vor ganz besonderen Herausforderungen standen natürlich Mitarbeitende mit Kindern in dieser Zeit. Kindereinrichtungen und anfangs auch Schulen waren schnell geschlossen, sodass neben den Arbeitsaufgaben die häusliche Betreuung der Jüngsten und für die älteren Kinder der virtuelle Hausunterricht zu organisieren waren. Für Familien war es äußert schwer, alles zugleich und dann auch in guter Qualität zu bewerkstelligen. Wir haben dabei die betroffenen Mitarbeiterinnen und Mitarbeiter mit Flexibilität bei Zeit, Arbeitsaufgaben und Organisation unterstützt, wo es nur möglich war. Es zeigte sich, dass nicht nur unsere Mitarbeitenden, sondern natürlich auch unsere Kundschaft vor dieser Herausforderung stand und es gelang so, mit Toleranz und viel Einsatz gute, manchmal ungewöhnliche Lösungen, zum Beispiel mit „Kinderdienst“ und „Spielbüros“, zu finden.

Neue Arbeitsumgebungen

Leere Büroräume während des Lockdowns, gute Erfahrungen mit mobilem Arbeiten sowie damit verbundene Freiräume führten bei uns zu Überlegungen, wie man die eigene Arbeitswelt dauerhaft umgestalten kann. Durch die getroffene Entscheidung gegen Großraumbüros bieten wir unseren Teams Möglichkeiten zum konzentrierten, störungsarmen Arbeiten und dennoch einem guten Austausch im Team. Die sehr großzügigen, räumlichen Gegebenheiten an unserem Hauptsitz in Dresden erlauben es unseren Mitarbeitenden, zu zweit, maximal zu viert in den Büros zu arbeiten. Eine Kombination von Präsenzarbeit und mobilem Arbeiten (Homeoffice) hat sich bewährt. Durch die Möglichkeit, auch mobil zu arbeiten, wird zeitlicher Freiraum geschaffen und die Mitarbeiterbindung gestärkt. Nicht nur Kollegen, die einen weiteren Arbeitsweg haben, nutzen gern diese Möglichkeit.

Größere Veränderungen ergaben sich durch diese Zeit im Dienstreiseverhalten. Die über Corona gestiegene Akzeptanz von Projekt- und Servicearbeit per Fernzugang sowie von Videokonferenzen zur begleitenden Kundenkommunikation machen auch weiterhin viele Dienstreisen mit unproduktiver Reisezeit überflüssig. Das gestattete eine Reduzierung und teilweise Elektrifizierung des Fuhrparkes auf eine jetzt sinnvoll verkleinerte Anzahl. Dabei setzt Robotron zunehmend auch auf die Nutzung öffentlicher Verkehrsmittel durch Jobtickets und Bahntickets 1. Klasse für ruhige Dienstreisen. Das sind Themen, bei denen sich betriebswirtschaftlich Sinnvolles mit Mietarbeiterzufriedenheit und Nachhaltigkeit verbinden lassen.

Kulinarisches

Wenige Dinge sind im Homeoffice so störend wie ein leerer Kühlschrank zur Mittagszeit. Dabei sind wir doch an unserem Hauptsitz in Dresden stolz auf eine sehr gute Mittags- und Eventversorgung mit einem lokalen, ausgezeichneten Catering-Anbieter für eigene Mitarbeitende und Gäste. Mitarbeiterumfragen haben bestätigt, wie wichtig eine gesunde und schmackhafte Essenversorgung ist. Trotz weniger Essensportionen während und teilweise nach Corona ist es gemeinsam und mit direkter Unterstützung gelungen, Angebot und Qualität der Versorgung weiterhin auf hohem Standard aufrechtzuerhalten.

Digitalisierung

Die in den letzten zehn Jahre ständig an Tempo und Umfang zunehmende Digitalisierung bringt auch für einen IT-Anbieter permanent neue Aufgaben mit sich. Corona war hier ein Katalysator, der Tempo, Umfang und Qualität der Digitalisierung beschleunigt hat. So sind Cloud Computing und Data Analytics verbunden mit künstlicher Intelligenz industrielle Praxis gewor-

Abbildung 3: Robotron Kaffeekueche (Quelle: Steffen Reißig, Robotron)

Abbildung 4: Robotron Nostalgiekueche (Quelle: Steffen Reißig, Robotron)

Abbildung 5: Robotron Sportraum (Quelle: Steffen Reißig, Robotron)

den und werden schrittweise auch in regulierten beziehungsweise konservativ agierenden Geschäftsfeldern eingeführt.

Mobile Arbeit, Fachkräftemangel im IT-Bereich gepaart mit einem sehr attraktiven Arbeitsmarkt für Jobsuchende sind Rahmenbedingungen, denen wir uns auch im Kontext der Digitalisierung stellen müssen. Das Homeoffice entkoppelt Arbeits- und Wohnort immer weiter und lockert das Verbundenheitsgefühl von Mitarbeitern zum Unternehmen. Zudem eröffnet mobiles Arbeiten auch internationalen IT-Firmen den regionalen Arbeitsmarkt. Der Kampf um kluge Köpfe wird immer härter. Ich glaube dennoch, dass regionale Anbieter in wirtschaftlich starken Regionen bei Beachtung von Mitarbeiterzufriedenheit verbunden mit marktgerechter Bezahlung hier gut mithalten können. Denn auch die Kundenbeziehungen werden durch die Digitalisierung umfangreicher und vielfältiger und neue Geschäftsfelder kommen hinzu. Schlüsselelemente für das wirtschaftlich erfolgreiche Agieren sind dabei primär Serviceorientierung und Branchenkenntnis.

Größere Änderung haben sich in den letzten drei Jahren im IT-Schulungsverhalten bei Kunden und bei der Qualifikation der eigenen Mitarbeitenden ergeben. Es zeigt sich, dass Verwaltungen und Firmen verstärkt Wert auf die Qualifikation ihrer Mitarbeitenden für digitalisierte Prozesse legen. Robotron verfügt über ein Schulungszentrum für eigene Produkte und Lösungen sowie für verwendete Partner-Technologien, einschließlich des von Oracle zertifizierten Schulungszentrums. Innerhalb weniger Wochen war es hier notwendig, das vor Corona maßgeblich durch Präsenzschulungen bestimmte Schulungsangebot auf hybride Modelle umzustellen. Mit hohem Einsatz aller Trainer und technologischer Unterstützung der Oracle University sowie der eigenen ITK konnte die Umstellung erfolgreich in wenigen Wochen gemeistert werden. Jetzt ist unser Schulungsangebot hybrid und kann jederzeit von reiner Präsenzteilnahme bis zu einer hundertprozentigen Online-Durchführung an das Buchungsverhalten der Kunden angepasst werden. So ganz nebenbei hat der Schulungsort Dresden aber als Stadt mit vielen kulturellen Möglichkeiten, dem Leben an der Elbe, einer großen Menge an vielseitigen Ausflugszielen, Gaststätten und Biergärten jede Menge zu bieten. Das allein wäre also jedenfalls auch so eine Reise wert.

Fazit

Corona hat die Digitalisierung des geschäftlichen und persönlichen Lebens stark beschleunigt. Demzufolge haben viele in dieser Zeit eingeführte und bewährte Änderungen Bestand. Das betrifft besonders die Kommunikation sowohl mit den Mitarbeitenden als auch mit der Kundschaft und damit direkt verbundenen Themen wie die zugehörige Arbeitsorganisation. Flexible Regelungen für mobiles Arbeiten sind nahezu in allen Bereichen von Industrie und Verwaltung eingeführt und erhalten geblieben. Mit Nachhaltigkeit und Kostenbewusstsein werden aufwendige Dienstreisen vereinfacht und manchmal überflüssig. Wissen und Daten allgemein und so auch spezielle IT-Qualifikationen sind nochmals wichtiger geworden. Besondere Bedeutung hat speziell für IT-Firmen, in denen Flexibilität und Leistung von Mitarbeitenden erwartet wird, eine ausgeprägte und authentisch praktizierte Mitarbeiterorientierung.

Beim Verbinden von Arbeit und Alltag haben die Veränderungen während der Coronazeit nicht nur bei Robotron positive Spuren hinterlassen und den Horizont erweitert.

Steffen Reißig
Steffen.Reissig@robotron.de

Steffen Reißig ist als Leiter für den Geschäftsbereich Technologie & Services bei Robotron verantwortlich. Er beschäftigt sich seit vielen Jahren mit Datenbanktechnologien und zugehörigen Services, dabei besonders mit Oracle-Technologie. Als interner und externer Dienstleister ist der von ihm geleitete Bereich für Plattform-Services, Data Services und Oracle-Schulungen im Zusammenhang mit großen Datenmengen unterwegs und bekannt. Darüber hinaus ist das Unternehmen als Softwarehersteller für die Energiewirtschaft, die öffentliche Verwaltung und die Industrie bekannt, sehr häufig dabei unter Verwendung von Oracle-Infrastrukturen.

Margarete Scheffler, Kontron AIS GmbH

Postpandemische Arbeitswelten – Warum ein Zugehörigkeitsgefühl zum Unternehmen immer wichtiger wird

Margarete Scheffler, Kontron AIS GmbH

Die Corona-Pandemie hat die Arbeitswelt aufgerüttelt. Inzwischen hat sich „New Normal" etabliert. Mit der digitalisierten Arbeit stehen die Unternehmen allerdings neuen Herausforderungen gegenüber. Das Zugehörigkeitsgefühl in den eigenen Reihen nicht zu verlieren und weiter auszubauen, gewinnt immer mehr an Bedeutung. In diesem Artikel geht es um Übergriffigkeit, neuronales Schmerzempfinden, die Bedeutung der Cafeteria sowie darüber, was dies alles mit dem Thema Verbundenheit zu tun hat und was wir von Alexa lernen können.

Remote Work, Home Office & Co sind mittlerweile aus unserem Alltag nicht mehr wegzudenken. Die Arbeit wurde weitestgehend digitalisiert und die Vorteile der flexiblen Arbeitszeitgestaltung überwiegen. „Wer weniger Zeit auf dem Weg zwischen Home und Office verbringt, hat mehr Zeit für Work und Life" hat Marcus Raitner so treffend formuliert. Das sind alles keine bahnbrechenden Erkenntnisse und doch ist es für die meisten Unternehmen eine ganz neue Erfahrung. Die Betriebe waren zu Beginn der Pandemie gezwungen, die benötigte Infrastruktur schnellstmöglich bereitzustellen. Und auf einmal erleben wir täglich, dass das, was für viele seit Jahren undenkbar schien, wirklich funktioniert. Die meisten von uns können mittlerweile überall dort arbeiten, wo es Internet gibt.

Herausforderungen im neuen Arbeitsalltag

Das Office als Ort, an dem Arbeit stattfindet, hat für viele Mitarbeitende kaum noch Bedeutung. Doch wie sieht es mit der Kaffeeküche aus und mit all den anderen Orten, an denen menschliche Begegnungen vor Corona stattgefunden haben? Wie wichtig ist es dem Mitarbeitenden noch, welches Firmenlogo den Bildschirmhintergrund ziert? Ist es mittlerweile allesentscheidend, für wie viel Gehalt pro Stunde der Mitarbeitende früh seinen Laptop aufklappt? Was hält ihn davon ab, ein höheres Angebot anzunehmen? Es gehört zur Aufgabe von Management und HR, Antworten auf diese Fragen zu finden.

Was wir vom virtuellen Assistenten Alexa lernen können

Schon immer konnten PersonalerInnen vom Marketing lernen: Das Company Image gab es vor Employer Branding, Produktwerbung lange vor Personalmarketing und Customer Experience vor Employee Journey. Beiläufig erwähnt kann dies ein Indiz dafür sein, dass die Bedeutung des Human Capital noch immer nicht den Stellenwert eingenommen hat, der nötig ist. Amazon ist die Einbettung des Unternehmens in das Alltagsleben der Kunden, oder kurz: „Ambient Embeddedness", mit dem virtuellen Assistent Alexa bereits gelungen. Die gleichartige Idee verfolgt „Belonging", zu Deutsch: Firmenzugehörigkeitsgefühl, nur mit umgekehrtem Klebeeffekt.

Die Bedeutung von Belonging

Sich zugehörig zu fühlen, ist eine evolutionär entstandene Überlebensstrategie und gehört seit jeher zu den menschlichen Grundbedürfnissen. Als soziale Wesen nehmen wir Ausgrenzung wie physischen Schmerz wahr, wie neurowissenschaftliche Studien belegen. Ein Zugehörigkeitsgefühl zum Unternehmen zu fördern, kann eine Antwort auf die vorangegangenen Fragestellungen sein und hat dadurch in seiner Bedeutung eine völlig neue Ebene erreicht.

Soziale Ausgrenzung wird vom Körper wie physischer Schmerz empfunden und führt zu einem sofortigen Rückgang des logischen Denkens um 30 Prozent und des IQ um 25 Prozent.[1]

Ein Zugehörigkeitsgefühl ist kein Nice-to-have

Fehlt die Bindung zum Unternehmen, stellt das ein großes Risiko dar. Vor allem in Zeiten, in denen sich Unternehmen bei ihren potenziellen Mitarbeitenden bewerben dürfen und nicht umgekehrt. Man könnte es auch so formulieren, dass sich die Zeiten der Company-Monogamie zum Belonging-Hopping geändert haben. Höchste Zeit also, diesem Thema die nötige Aufmerksamkeit zu widmen. Denn 70 Prozent der Angestellten haben nicht das Gefühl, vollständig in die Unternehmensangelegenheiten einbezogen zu sein und geachtet zu werden. Und: Mitarbeitende, die sich nicht als Teil des Ganzen fühlen, spielen sechsmal häufiger mit dem Gedanken zu kündigen als Mitarbeitende, die sich ihrem Arbeitgeber zugehörig fühlen. Das ergab im März 2022 eine weltweite Studie der internationalen Unternehmensberatung Bain & Company.

70 % der Mitarbeitenden fühlen sich ihrem Unternehmen nicht zugehörig und spielen 6 x häufiger mit dem Gedanken zu kündigen.[2]

Grenzen des Kollektivgedankens

Ist das Thema nicht etwas übergriffig? Schließlich bleibt es doch jedem selbst überlassen, wie man sich dem Unternehmen gegenüber verbunden fühlt. Und für einige von uns ist und bleibt die Arbeitsstätte der Ort, um Geld zu verdienen. Meine Haltung dazu ist klar: Belonging stellt kein privates Thema dar – nicht ausschließlich jedenfalls. Wenn Angebote und Strukturen geschaffen werden, bleibt es noch immer die Entscheidung jedes einzelnen, diese auch wahrzunehmen. Wir leben mittlerweile in einer Welt, in der wir frei über unsere Ressource Arbeitskraft entscheiden können, und das ist gut so. Maßgebend ist, dass sich Arbeitgebende nicht als „Heimat" aufbauschen und ihren Mitarbeitenden das Gefühl der Vereinnahmung geben.

Welche Faktoren Bindung fördern

Es bleibt also die entscheidende Frage: Wie können Arbeitgebende ein Gefühl der Verbundenheit etablieren? Meiner Ansicht nach geht es dabei um vier entscheidende Punkte:

- Communication
- Appreciation
- Community
- Purpose

Eine offene und bewusste Kommunikation ist die Basis von Transparenz und schafft eine Vertrauenskultur. Jeden einzelnen Mitarbeitenden wirklich zu sehen, gute Leistungen so häufig wie möglich zu loben, ihn wertzuschätzen und ihm zuzuhören sowie Optionen der Beteiligung und des Feedbacks zu ermöglichen, sind wichtige Schlüssel. Es ist entscheidend, sich innerhalb einer integrativen Kultur als Teil der Gemeinschaft wahrzunehmen, die mit einer klaren Haltung Responsibility, Fairness und Gerechtigkeit wirklich lebt. Nicht zuletzt ist es elementar, gemeinsam eine Vision zu verfolgen. Von dieser gemeinsamen Idee lässt sich dann das „Warum?", im Sinne von Simon Sineks „Find Your Why", für jeden Einzelnen ableiten.

Inklusion schaffen

Das klingt nach großen, tiefgreifenden und langfristigen Aufgaben. Bedeutet dies, dass wir Belonging deshalb erst einmal auf die lange Bank schieben? Hoffentlich nicht. Denn all die genannten Faktoren sind kein Neuland und bestenfalls bereits – zumindest teilweise – im Unternehmensalltag integriert. Es gilt an dieser Stelle, sich deren Bedeutung bewusst zu werden sowie eine unternehmensseitige Umsetzung zu forcieren. Und wie können wir kurzfristig Veränderungen in Gang setzen? Ganz praktisch können wir auch schon heute das „Wir-Gefühl" zu stärken, indem wir informelle Meeting-Points, digitale Firmenveranstaltungen und interaktive Veranstaltungsformate ins Leben rufen.

Quelle: www.canva.com

Kommunikation fördern: Ansätze für die Praxis

Belegschaftsversammlungen sollten quartalsweise digital als „UnternehmenUnplugged"-Meetings im Open-Mic-Stil veranstaltet werden, bei denen die Geschäftsführung in einer lockeren Atmosphäre den Fahrplan für die kommende Zeit vorstellt und anschließend Fragen der Mitarbeitenden beantwortet. Abteilungsleiter können „offene-Tür"-Zeiträume blocken, in denen ihre Mitarbeitenden die Möglichkeit haben, mit Problemen, Wünschen und Anregungen auf sie zuzukommen. Denkbar ist auch ein wöchentliches, offenes Meeting „Auf ein Kaffee mit HR", bei dem alle Beschäftigten die Möglichkeit haben, mit den PersonalerInnen ins Gespräch zu kommen.

„Im Grunde fühle ich mich meinem Arbeitgeber ähnlich verbunden wie vor Corona-Zeiten. Es ist weniger der Standort, sondern die Verbundenheit mit den KollegInnen und spannenden Aufgaben. Allerdings fehlen die spontanen, oft persönlichen Gespräche. Die Messenger-Dienste oder Online-Konferenzen können das nicht vollständig leisten. Von meinem Arbeitgeber wünsche ich mir deshalb Formate und die nötige Infrastruktur, um mit meinen KollegInnen leichter in einen ungezwungenen Austausch zu kommen, welche wir dann selbst mit Leben füllen können."

Vanessa Kluge – arbeitet als Produktmanagerin seit einem Jahr aus Lissabon und Stuttgart für die Kontron AIS

„Wir-Gefühl" stärken: Best Practice

Einige Firmen nutzen bereits erfolgreiche Formate wie „Blind-Dating-Tools", bei denen Mitarbeitende per Zufallsprinzip abteilungsübergreifend für einen virtuellen Kaffee zugelost werden. Ein ähnliches Prinzip verfolgt der „Mystery Lunch", bei dem sich noch unbekannte KollegInnen zum Mittagessen verabreden können. Es sind Yoga-Breaks und digitale Kochevents zum Thema „Nachhaltig genießen" denkbar, auch Meet-ups für einen ungezwungenen Austausch über Zeitmanagement oder Stressbewältigung sowie Workshops rund um das Thema Gesundheit am Arbeitsplatz. Um den interdisziplinären Austausch zu fördern, können lockere Runden zu spannenden, fachlichen Themen helfen, die in regelmäßigen Abständen abgehalten werden. Es könnte dabei um Best-practice-Ansätze zu selbstorganisierten Teams, Mitarbeiterfüh-


Quelle: www.canva.com

rung oder Tipps und Tricks zu bestimmten Software-Lösungen gehen.

Kompensieren digitale Veranstaltungen reale Interaktionen vollständig?

Je nach Regionalität der Firma sind auch Formate im „real life" denkbar, die die Angestellten gezielt in die Firma locken können. In unserem Unternehmen haben wir beispielsweise sehr gute Erfahrungen mit digitalen Abteilungsvorstellungen und der Möglichkeit, anschließend auf ein Barbecue zusammenzukommen. Auch Weihnachtsfeiern und Sommerfeste planen wir bewusst als nicht-virtuelle Highlights, da wir überzeugt sind, dass menschliche Interaktionen nicht vollständig substituiert werden können.

„Spontane Diskussionen und der abteilungsübergreifende Austausch sind weniger geworden. Gerade um Grüppchenbildung und dem Gefühl, auf sich gestellt zu sein, entgegenwirken zu können, ist Integration wichtig. Ich versuche, den Effekt aktuell durch Aufmerksamkeit und Proaktivität zu kompensieren. Regelmäßige Möglichkeiten für private Interaktionen brauche ich dafür nicht unbedingt. Allerdings würden mich Formate für einen interdisziplinären Austausch zu spannenden Fachthemen interessieren."
Enrico Hüttig – ist Product Owner in der Kontron AIS und verbringt aktuell zwei Monate als Digital Nomade in Kroatien

Umsetzung

Zur ganzen Wahrheit gehört auch, dass die Maßnahmen Zeit benötigen, bis sie sich etablieren können. Ein Blind-Dating-Tool einzuführen, indem man einen entsprechenden Link und die Rahmenbedingungen an die Belegschaft schickt und damit die Arbeit als getan ansieht, wird mit großer Sicherheit floppen. Jedes Event bedarf einer Anlaufphase, in der es von mindestens einer/einem OrganisatorIn betreut wird, um Inhalte, Ideen und vor allem Anwesenheit zu gewährleisten. Deshalb ist mein Rat, nicht zu viel auf einmal umsetzen zu wollen, sondern sich fokussiert einem Format nach dem anderen zu widmen.

Ausblick

Schließlich sind der Kreativität hier, wie so oft, keine Grenzen gesetzt. Und die Unternehmen, die das Thema Belonging verstanden haben, fragen am besten ihre MitarbeiterInnen danach. Gut möglich, dass man bei dem Thema so unterschiedliche Meinungen erhält, wie es Angestellte gibt. Diejenigen, die sich über den positiven Nebeneffekt „Ruhe" ihrer Home-Office-Tätigkeit freuen, weil soziale Interaktion auch mit gewissem Stress verbunden sein kann, werden derartige Maßnahmen wohl weniger erreichen. Es gilt also wie so oft die Balance zu finden aus einem Pool unterschiedlicher Angebote und der Machbarkeit auf der anderen Seite.

Quellen

[1] Baumeister, R. F., Twenge, J. M., & Nuss, C. K. (2002). Effects of social exclusion on cognitive processes: Anticipated aloneness reduces intelligent thought. Journal of Personality and Social Psychology, S. 817-827, Ohio.

[2] Coffman, J., Bax, B., Noether, A., & Blair, B. (2022). The Fabric of Belonging: How to Weave an Inclusive Culture. Bain & Company, Inc., Boston.

Margarete Scheffler
margarete.scheffler@kontron-ais.com

Margarete Scheffler ist als HR Business Partner tätig. Nach ihrem Masterstudium „Business Administration" in Mittweida ist sie seit 2018 bei der Kontron AIS GmbH beschäftigt, einem IT-Dienstleister für Automatisierungssoftware mit über 200 Mitarbeitenden in Dresden. Sie vertieft sich insbesondere in die Themengebiete Employer Branding, Mitarbeiterbindung und Talentmanagement.


Living room
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
20,5°

Raumbuchung trifft smarte Thermostate

Benjamin Klatt, Buuky / viadee AG

Flexible Arbeitsplätze, ein Hang zu guter User Experience und eine Vorliebe für Nachhaltigkeit: Kurzerhand ist ein neues Potenzial zur Energie-Einsparung entstanden. Durch die Integration unseres Arbeitsplatzbuchungssystems Buuky mit unserer Gebäudeautomation ist es uns gelungen, unsere Heizkosten in diesem Winter um voraussichtlich knapp 50% zu reduzieren. Wie dieses System aussieht und auf welche Herausforderungen und Lösungen wir gestoßen sind, berichten wir in diesem Artikel. Um dies vorwegzunehmen: Es hat sich gelohnt!

Wie sieht dieses Energiesparpotenzial aus?

Wie viele Unternehmen arbeiten wir bei der viadee mit einem Desk-Sharing-Prinzip, also einer flexiblen Arbeitsplatzwahl je nach Projekteinsatz, Vorlieben oder Raumbedarf. Zur leichteren Koordination der Arbeitsplätze nutzen wir ein Raumbuchungssystem, das wir Buuky (abgeleitet vom finnischen „Buukata“ / „Buchen“) getauft haben *[1]*. Unsere Heizkörper in Münster und Köln haben wir vor Kurzem mit smarten Thermostaten von Homematic *[2]* ausgestattet. Mit ihnen konnten wir zunächst alle Räume tagsüber auf 19° Grad und sonst auf 16° Grad heizen. Durch eine Integration der Heizungssteuerung mit unserem Arbeitsplatz-Buchungssystem haben wir das Ganze nun noch eine Stufe weitergebracht: Jetzt werden nur die Büros geheizt, in denen sich Kolleg:innen eingebucht haben. Natürlich bleibt die manuelle Heizungseinstellung weiterhin möglich.

Was steckt hinter Buuky?

Als Berater:innen sind wir bei vielen verschiedenen Kunden unterwegs und haben dadurch diverse Arbeitsplatz-Buchungssysteme gesehen. Aus den gesammelten Erfahrungen entstand der Impuls: Das geht besser! Das war der Grundstein dafür, Buuky zu entwickeln und sich das Ziel einer bestmöglichen User Experience (UX) zu setzen. *Abbildung 1* zeigt einen Screenshot von Buuky mit der Möglichkeit, Plätze visuell über einen Raumplan zu wählen.

Inzwischen ist Buuky weit über die erste Version hinaus. So gibt es beispielsweise Analytics zur Raumoptimierung, wie bevorzugte technische Ausstattung oder Energiesparpotenzial, Serienbuchungen oder eine Integrationsschnittstelle, durch die Anwendungen auf Buuky-Informationen zugreifen können. Was geblieben ist, ist der starke Fokus auf die einfache und intuitive Bedienung. Im Kern ist es simpel: Funktioniert die Arbeitsplatzbuchung ein-

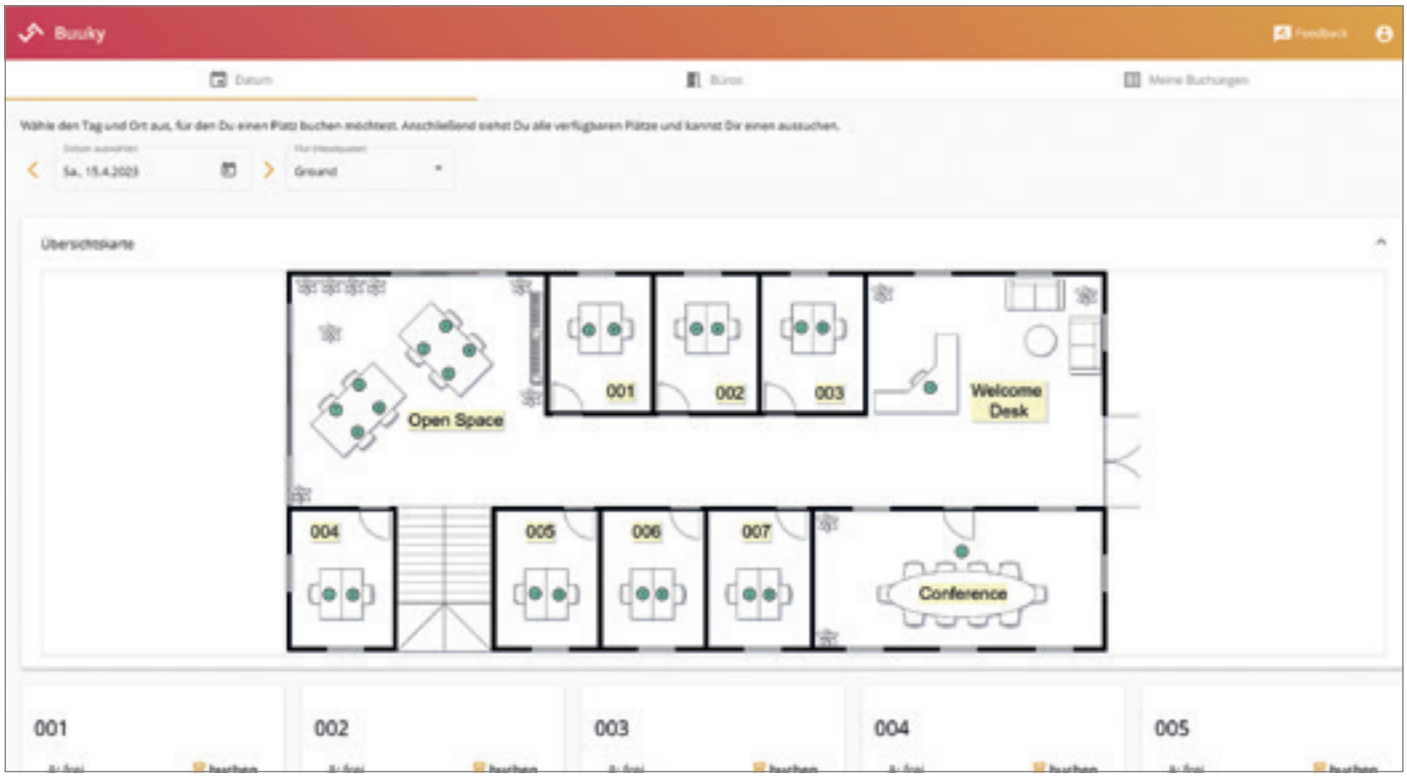


Abbildung 1: Buuky Buchungsübersicht (© buuky.app)

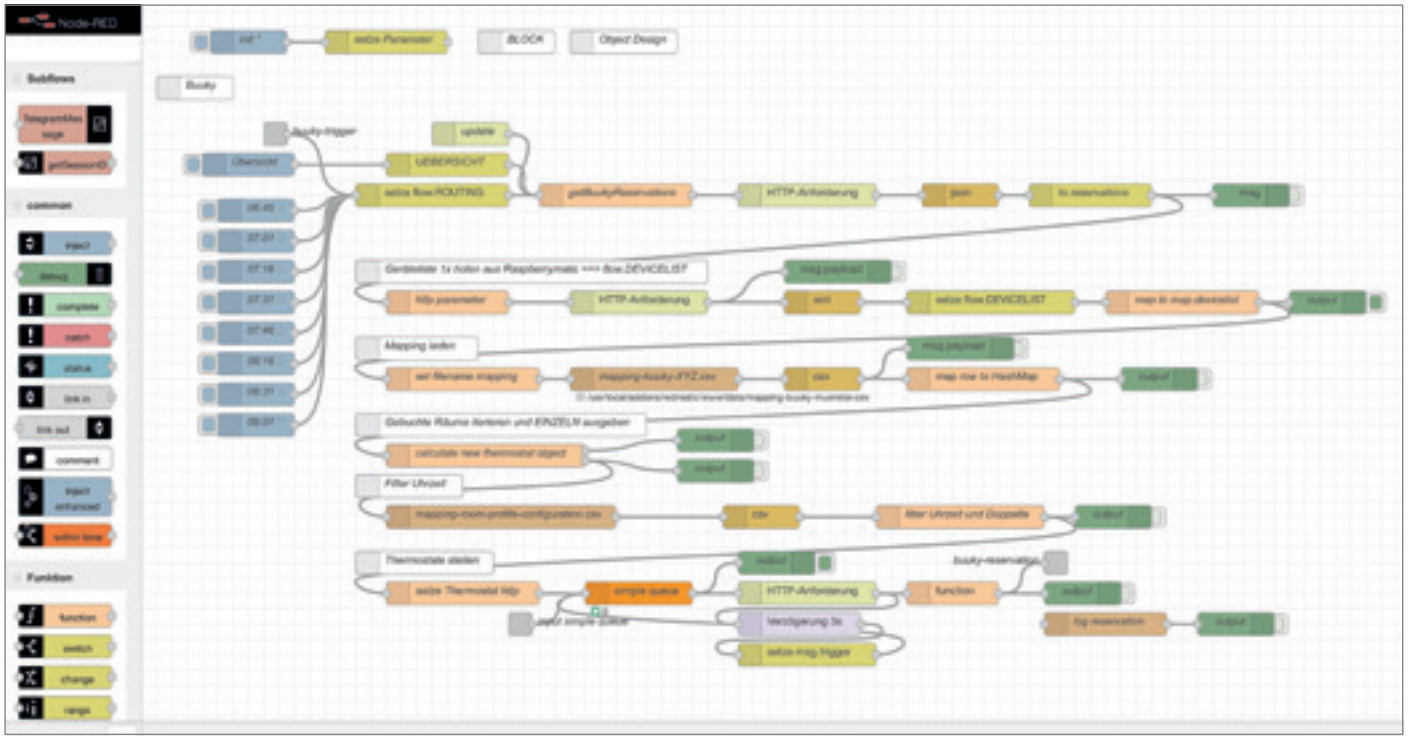


Abbildung 2: NodeRed Steuerungsübersicht (© viadee)

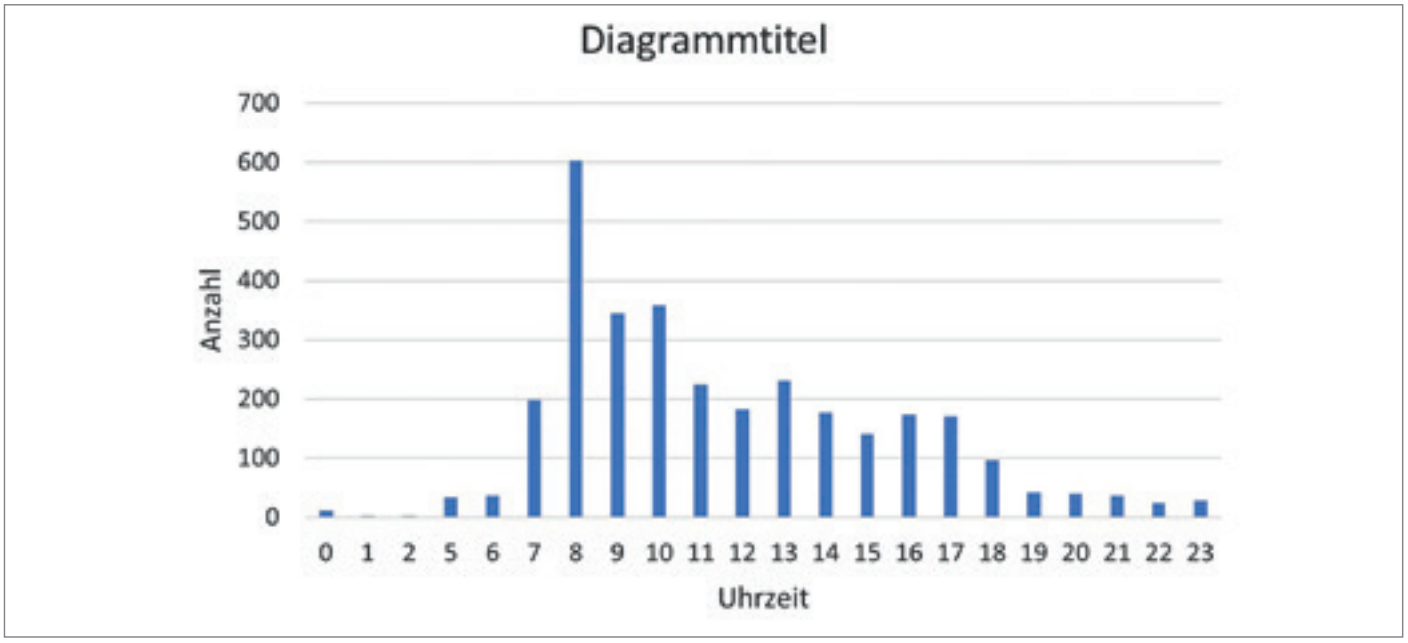


Abbildung 3: Buchungsverhalten über den Tag verteilt (© viadee)

fach und schnell, wird sie stärker genutzt und es entsteht eine Informationsgrundlage, mit der man ganz neue Szenarien erschließen kann. So auch eine optimierte Heizungssteuerung.

Wie funktioniert die Heizungssteuerung?

Die Heizungsthermostate in unseren Geschäftsstellen in Münster und Köln sind inzwischen alle durch smarte Thermostate von Homematic ersetzt wurden. Sie können remote über Funk gesteuert werden. Für die eigentliche Steuerungslogik setzen wir die Open-Source-Lösung NodeRed ein, mit der sich komfortabel Steuerungsflüsse entwickeln lassen. *Abbildung 2* zeigt den NodeRed Flow für die Integrationslogik mit Buuky.

Wie in dem Screenshot in *Abbildung 2* zu sehen ist, werden die Reservierungsdaten per JSON/http-Aufruf von Buuky über dessen Rest-Schnittstelle abgerufen (Integration API). Die Möglichkeit eines JSON-Aufrufs gehört zum Standard von NodeRed. Mit der Information, welche Plätze an dem jeweiligen Standort gebucht sind, werden die zugehörigen Thermostate entweder hoch- oder heruntergeregelt.

Die Herausforderungen bei dem Ansatz

Die funkbasierten Geräte von Homematic haben es uns sehr einfach gemacht, un-

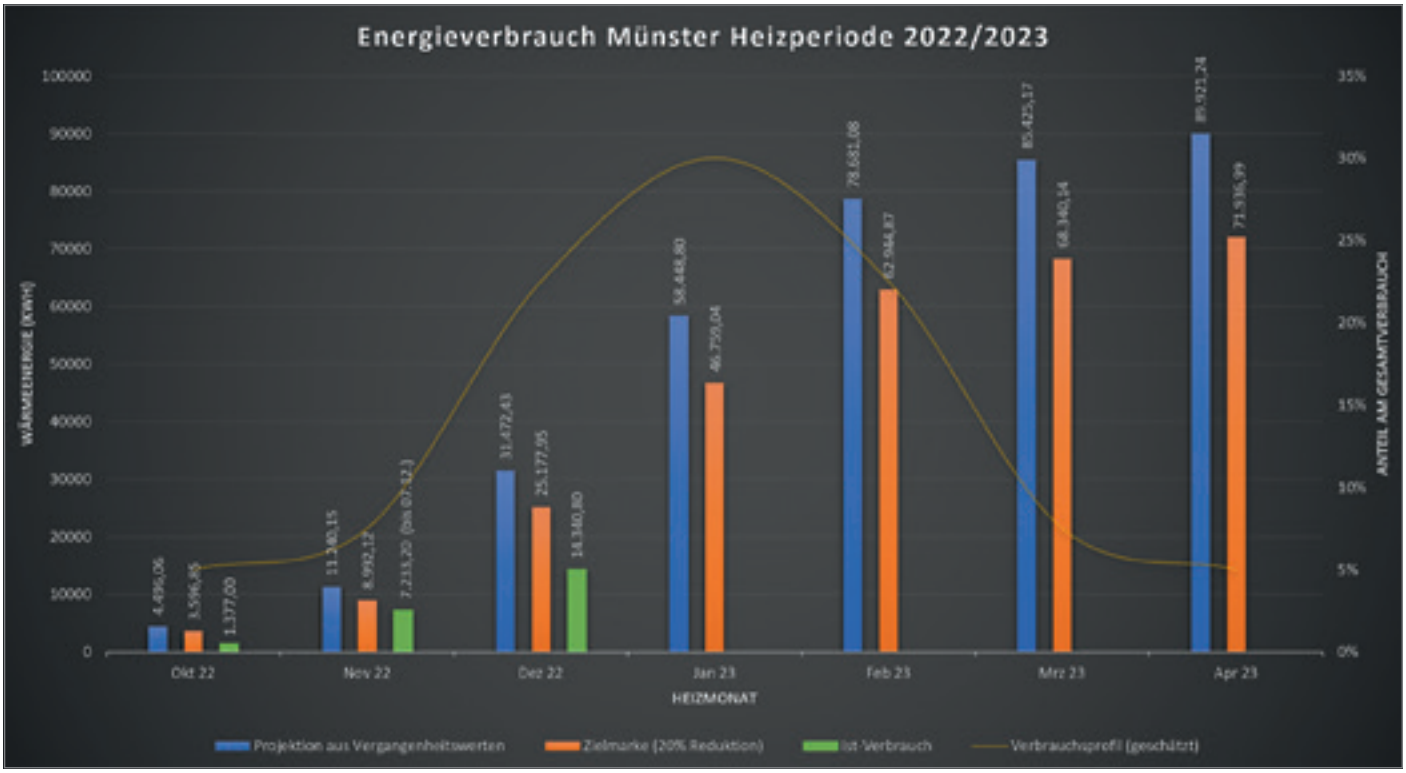


Abbildung 4: Entwicklung Wärmeverbrauch Heizperiode 22/23 in der Geschäftsstelle Münster (© viadee)

sere Büros in kürzester Zeit umzurüsten. Gleichzeitig gibt es aber natürliche Grenzen für den Funkverkehr. So darf über die Protokolle, die auch Homematic verwendet, nur eine begrenzte Anzahl von Signalen pro Minute gesendet werden. Bei unseren derzeit 140 Arbeitsplätzen haben wir diese Grenzen immer wieder erreicht, sodass zum Beispiel Thermostate ihre Statusinformationen nicht immer übermitteln konnten. Durch den guten Support von Homematic ließ sich das Problem aber mit einer passenden Konfiguration schnell lösen.

Darüber hinaus haben wir recht schnell gelernt, dass Arbeitsplatz-Buchungen zu unterschiedlichsten Zeiten stattfinden, wie auch die Verteilung in *Abbildung 3* zeigt. Sei es, dass Kolleg:innen verschiedene Gewohnheiten haben, oder auch Stornierungen oder Umbuchungen, die zu jeder Tageszeit durchgeführt werden. Die Lösung war hier, einfach mehrmals am Tag die Buchungsdaten zu den für unser Unternehmen typischen Zeiten abzurufen und Thermostate zu justieren.

Unsere positiven Erfahrungen

Die gute User Experience des Buchungssystems hat sich positiv auf das Buchungsverhalten ausgewirkt und damit überhaupt erst die Datenbasis für unser Energiesparpotenzial geschaffen. Ebenso wichtig ist der Mobile-First-Ansatz, da das Buchen wie ein Automatismus immer und überall, selbst morgens beim Zähneputzen, möglich sein muss.

Unser ausgesprochenes Ziel für die Heizperiode im Winter 2022/2023 war es, 20 % weniger Wärmeenergie als im Vorjahreszeitraum zu verbrauchen. Wie in *Abbildung 4* beispielhaft für unseren Standort in Münster zu sehen ist, liegen wir zum Jahreswechsel mit knapp 50 % Einsparung deutlich darunter.

„Einfach machen“

Unser Fazit der letzten Monate ist, dass sich ein solches System leichter umsetzen lässt, als wir erwartet haben. Insbesondere hat es sich bewährt, klein anzufangen. Die Steuerung war im ersten Schritt ein Prototyp in einem einzigen Büro.

Daher können wir nur empfehlen: „Einfach machen“. Und wer hierzu etwas Austausch sucht oder mal einen Blick auf die Lösung werfen möchte, kann sich gern mit einem Hinweis auf diesen Artikel bei uns melden. Das gilt ebenso für diejenigen, die gerne einen Buuky Account für das eigene Unternehmen oder Team nutzen möchten.

Quellen

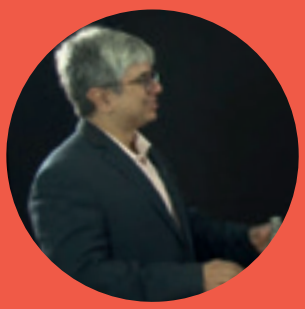
[1] Buuky Website
https://www.buuky.app
[2] Homematic Website
https://homematic-ip.com
[3] NodeRed Website
https://nodered.org

Dr. Benjamin Klatt
benjamin.klatt@viadee.de

Benjamin ist für die viadee als IT-Architekt und Coach im Bereich Organisationsentwicklung unterwegs. Er begleitet Unternehmen bei der Digitalisierung von Produkten und Prozessen sowie der Erschließung neuer Technologien, Arbeitsweisen und Innovationen. Zudem arbeitet er in der Forschung und Entwicklung der viadee und im Buuky-Team.

BEST OF DOAG ONLINE

Eine Auswahl der besten DOAG News Februar/März 2023



DOAG.tv mit Tirthankar Lahiri – über Hochverfügbarkeit und Converged Database

Martin Klier, DOAG Themenverantwortlicher Hochverfügbarkeit, interviewt Oracles Senior Vice President und stellt dabei auch grundsätzliche Fragen.

DOAG Datenbank Kolumne: Warum man die Nadel nie im Heuhaufen verstecken sollte

Es war einmal eine brandneue Oracle-Datenbank (Standard Edition), in der Produktionsdaten gespeichert werden sollten.

DOAG gestaltet Track bei der TDWI München 2023

Das TDWI-Event markiert den Start für eine noch engere Zusammenarbeit zwischen dem TDWI und der DOAG.

DOAG Datenbank Kolumne: Open AI, ChatGPT und Neusprech – Kundenverwirrung mit "Multicloud"

Der mit OpenAI realisierte Chatbot "ChatGPT" ist derzeit in aller Munde.

Das Endergebnis der Wahl zur DOAG Delegiertenversammlung 2023

Der Wahlausschuss hat heute um 12 Uhr das Endergebnis der Wahl zur DOAG Delegiertenversammlung festgestellt.

CloudLand 2023 eröffnet mit einem Barcamp

Die zweite Ausgabe des deutschsprachigen Cloud Native Festivals startet am 20. Juni mit einem ganztägigen Barcamp namens "CloudCamp". Was ist das?

Wir begrüßen unsere neuen Mitglieder

Natürliche Mitglieder:

- Thomas Tschirner
- Thorsten Pfeiffer
- Svetlana Makhlyaid
- Robin Burkard
- Sven Jensen
- Diwaker Nithyanandan
- Christian Bonk
- Karsten Lück
- Michael de la Porte
- Jacqueline Haefke
- Christian Wiegele
- Volker Demel
- Knut Göttling
- Klaus Alberts
- David Fröhlich
- Andreas Monscha
- Stefan Wehling
- Frank Schürer
- Alexander Paulczynski

Korporative Mitglieder:

- ELARA Leitstellentechnik GmbH, Repräsentant: Iuri Olari
- FLYERALARM BitLabs GmbH, Repräsentant: Michel Hartmann
- ImpressSol GmbH, Repräsentant: Victor Getz
- Hiqs GmbH, Repräsentant: Robin Burkard

Termine

April 04

13. bis 15.04.2023
DOAG Leitungskräfteforum und Delegiertenversammlung
Berlin

14.04.2023
Praxiserfahrungen mit Multitenant DB WebSession mit Marco Pachaly-Mischke
Online

20.04.2023
JavaScript in der Datenbank (MLE) DevTalk mit Moritz Klein und Martin Bach
Online

25.04.2023
Do I use the OCI Database Management Service or Enterprise Manager? Infrastruktur & Middleware Community WebSession mit Sriram Vrinda und Steven Lemme
Online

Mai 05

03. und 04.05.2023
APEX connect 2023
Zwei Konferenztage mit zahlreichen Vorträgen und Workshops zu den Themen APEX, JavaScript und PL/SQL
Berlin

09.05.2023
Using End-to-End Distributed Tracing on OCI Infrastruktur & Middleware Community WebSession mit Jurgen de Leijer
Online

11.05.2023
DB-Programmierung: Performance-Aspekte in PL/SQL Programmierung DevTalk mit Ulrike Schwinn, Carolin Hageman und Jürgen Sieben Moderation: Christian Neumüller
Online

12.05.2023
Oracle Lizenzen in der Cloud DB WebSession mit Michael Skowasch
Online

Juni 06

24. und 25.05.2023
DOAG 2023 Datenbank mit Exaday
Die Konferenz zu den Themen Datenbanken (Oracle Database, PostgreSQL, MySQL) und Engineered Systems (Exadata & ODA)
Düsseldorf

13.06.2023
Oracle Exadata Insights: AIOps for Optimizing Resources. Infrastruktur & Middleware Community WebSession mit Murtaza Husain
Online

20. bis 23.06.2023
CloudLand 2023 - Das Cloud Native Festival
Community-Veranstaltung rund um die Themen Cloud- und Container-Technologien, Continuous Delivery, Microservices und DevOps
Phantasialand, Brühl

Impressum

Red Stack Magazin inkl. Business News 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 inkl. Business News 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 inkl. Business News 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: Björn Bröhl. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führt einen konstruktiv-kritischen Dialog mit Oracle.

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
ViSdP: Fried Saacke
Redaktionsleitung Red Stack Magazin:
Martin Meyer
Redaktionsleitung Business News:
Marcos López
Kontakt: redaktion@doag.org

Weitere Redakteure (in alphabetischer Reihenfolge): Andreas Badelt, Christian Ballweg, Martin Berger, Sven Bernhardt, Anna Collard, Ralf Durben, Randolf Eberle-Geist, Wilfried Eigl, Kai-Uwe Fischer, Markus Flechtner, Peter Hoffmann, Susanne Jahr, Dr. Benjamin Klatt, Marcos López, Christoph Lüttig, Bojan Miljias, Kilian Müller, Steffen Reißig, Margarete Scheffler, Michael Skowasch, Jörg Sobottka, Dani Schnider, Markus Schwabeneder, Robert Wortmann.

Titel, Gestaltung und Satz:
Diana Tkach
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

Fotonachweis:
Titel: © fullvector | *www.freepik.com*
S. 6: © Brooke Cagle | *www.unsplash.com*
S. 8: © 3093594 | *www.pixabay.com*
S. 14: © Daria-Yakovleva | *www.pixabay.com*
S. 20: © honeyghumaan01 | *www.pixabay.com*
S.24: © fietzfotos | *www.pixabay.com*
S. 28: © sasint | *www.pixabay.com*
S. 38: © oladimeji-ajegbile | *www.unsplash.com*
S. 48: © josh-calabrese | *www.unsplash.com*
S. 54: © TheDigitalArtist | *www.pixabay.com*
S. 58: © madartzgraphics | *www.pixabay.com*
S. 61: © Tumisu | *www.pixabay.com*
S. 66: © Kai Uwe Fischer | *Oracle*
S. 76: © Nam Anh | *www.pixabay.com*
S. 83: © RickJbrown | *www.pixabay.com*
S. 90: © Wilfried Eigl

Titel S. 102: © vollume | *www.unsplash.com*
S. 108: | *www.canva.com*
S. 112: © freepick | *www.freepik.com*

Anzeigen:
sponsoring@doag.org

Mediadaten und Preise:
www.doag.org/go/mediadaten

Druck:
WIRmachenDRUCK GmbH,
www.wir-machen-druck.de

Inserentenverzeichnis

JavaLand

on demand

JavaLand 2023 verpasst?

Jetzt On-demand-Ticket buchen und Vortragsaufzeichnungen anschauen!

Alle Angebote im On-demand-Ticket-Shop

Präsentiert von: iJUG Verbund · Heise Medien · DOAG

Veranstalter: JavaLand

APEX connect

by DOAG

3. - 4. Mai 2023

IN BERLIN

PROGRAMM ONLINE

apex.doag.org

DOAG