Nr. 4 | Juni 2023 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP

inklusive BUSINESS NEWS

CONTAINER UND CLOUD-TECHNOLOGIEN

Aus der Praxis

Ein Ziel, verschiedene Ansätze: Performance-Tuning in der Praxis

Im Interview

Frank Prechtel, Softwareentwickler

Business News

Digitale Transformation

#CloudLand2023
www.cloudland.org

CloudLand 2023

DAS FESTIVAL DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY

• Microservices & DDD • CI/CD & Automatisierung
• Container & Cloud-Technologien • DevOps & Methodik

20. - 23. JUNI
im Phantasialand in Brühl

MAIN SPONSORS

MAYFLOWER

MediaMarkt SATURN Technology

QA|WARE SOFTWARE ENGINEERING

REWE DIGITAL

Summer Night powered by

InterFace AG the face of informatics

Eventpartner

Heise Medien

André Sept
Vorstand Cloud Native, Leiter Cloud Native Community

Ausgabe Nr. 4/2023 auf Abruf!

Liebe Mitglieder, liebe Leserinnen und Leser,

Cloud- und Container-Technologien sind mittlerweile kein neuer Trend mehr. Im Interview zum Titelthema gibt Frank Prechtel dazu einen 360-Grad-Einblick. Wie Container mit Leichtigkeit mit dem Tool Oracle Verrazzano in Betrieb genommen werden können, zeigt Marcel Boermann-Pfeifer. Wie eine „Oracle Zero Downtime Migration in die Cloud" bewältigt werden kann, erklärt Sinan Petrus Toma in seinem Artikel. Einen klaren Blick in die Thematik Container bringt Jan Karremans. Jonas Gassenmeyer berichtet über seine Erfahrungen in „Ein Jahr mit PostgreSQL – Das Leben danach". Mit dem Gastkommentar von Günther Stürner „Java – gibt´s auch anderswo" bekommen wir Einblicke rund um das Thema Java. Jürgen Sieben erzählt in seinem Beitrag Spannendes über Packages und ihre Initialisierung. „Das neue Schweizer Taschenmesser für den Oracle DBA mit Ansible, AWX und Co." stellt Jérôme Witt in seinem Beitrag vor. In einem Praxisbericht widmen sich Dani Schnider und Martin Berger ihren verschiedenen Ansätzen im Performance-Tuning. In seinem zweiten Artikel über das unterschätzte Feature Attribute Clustering bringt Randolf Eberle-Geist erneut Licht ins Dunkel. Wie man den Einstieg in autonome Datenbanken über Oracle REST Data Services für autonome Datenbanken schafft, erläutert Timo Herwix. In seinem Artikel über Oracle PL/Scope zeigt Peter van Garsel wie sich das Tool mit seinen Features zu einem Werkzeug in der PL/SQL-Code-Analyse entwickelt.

Diese Ausgabe zeigt einige Facetten der Container- und Cloud-Technologien. Wer noch nicht genug von Cloud-Themen hat, kann gerne zum Cloudland Festival ins Phantasialand kommen, bei dem wir ein Programm mit über 200 Sessions zusammengestellt haben. Wir sehen uns!

Eine perfekte Ergänzung bringt die integrierte Business News mit dem Titel „Digitale Transformation". Genau vor dieser Transformation stehen viele Unternehmen im Zuge ihrer Cloud-Strategie. Im Interview mit Frau Prof. Weibel und Herrn Dr. Schafheitle wird das Thema Changemanagement bei der Einführung kluger HCM-Systeme besprochen. Über „echte" digitale Abrechnungsprozesse schreiben Dr. Thomas Karle, Axel Bröker und Matthias Sauer. Der Artikel von Armin Wildenberg zur digitalen Transformation im Lichte von ChatGPT schafft den Bogen rund um den Hype KI (oder doch schon mehr als ein Hype?), was sicherlich ein Thema in einer der kommenden Ausgaben der Red Stack und Business News sein wird.

André Sept

DER AUFKLEBER

In dieser Ausgabe finden Sie einen Aufkleber „Microservices & Domain-driven Design". Dies ist der korrekte Titel der Print-Ausgabe der Red Stack inkl. Business News Nr. 03/2023 im April. Zu unserem Bedauern war leider der falsche Titel „Container und Cloud-Technologien" abgedruckt. Den können Sie jetzt überkleben. Wir bitten nochmals um Verständnis.

Ein Jahr PostgreSQL – Das Leben danach

Packages und ihre Initialisierung

Einleitung

Cloud

PostgreSQL

PL/SQL

Datenbank

Development

BUSINESS NEWS

Digitale Transformation

52

Leitartikel | *„Es ist bequem, wenn man wichtige Entscheidungen der Technologie überlässt. Aber eben nicht immer klug.”*

89

Ein Einstieg in Oracle REST Data Services für autonome Datenbanken

62

Die digitale Transformation in einer Betrachtung mit ChatGPT

66

Und plötzlich war Lockdown – Lehren aus einer Pandemie-Transformation

Intern

News

TIMELINE

20. April 2023

Im DevTalk mit mit Moritz Klein, Martin Bach und Niels de Bruijn geht es diesmal um das Thema „JavaScript in der Datenbank (MLE)“.

25. April 2023

„Do I use the OCI Database Management Service or Enterprise Manager?” heißt die DOAG IMC WebSession mit Sriram Vrinda and Steven Lemme von Oracle.

25. April 2023

Das Regionaltreffen Osnabrück/Bielefeld/Münster findet in Paderborn statt. Themen der beiden Vorträge sind „Kostenfreie Kundenportale in der Always-Free Oracle Cloud“ und „Verwendung von MySQL REST Service als Datenquelle in Oracle APEX“.

26. April 2023

Das Regionaltreffen Dresden/Sachsen findet in Dresden statt. Folgende Oracle- Infrastruktur-Themen werden behandelt „Neue DB-Version 23c“, „Neue Forms-Version“ und „Neuigkeiten im Umfeld von DB-Automatisierungen (RedHat OpenShift, Kubernetes…)“.

3. und 4. Mai 2023

Die APEX connect 2023 findet im avantgardistischen Hotel nhow Berlin statt. Die über 300 Teilnehmerinnen und Teilnehmer erleben zwei Konferenztage mit zahlreichen Vorträgen und Workshops zu den Themen APEX, JavaScript und PL/SQL, entspanntem Networking sowie eine unvergessliche Party-Nacht voller Popmusik mit JP and Friends und Vanessa Iraci.

9. Mai 2023

In der DOAG IMC WebSession mit Jurgen de Leijer (Oracle) und Jessica Steger erfahren die Teilnehmerinnen und Teilnehmer alles über das Thema „Using End-to-End Distributed Tracing on OCI“.

11. Mai 2023

Im DevTalk mit Ulrike Schwinn, Carolin Krützmann, Jürgen Sieben und Christian Schwitalla heißt das Thema „DB Programming: Performance Aspects in PL/SQL Programming“.

12. Mai 2023

In der WebSession mit Michael Skowasch und Oliver Pyka steht das Thema „Oracle Lizenzen in der Cloud“ auf der Agenda.

24. Mai 2023

Das Regionaltreffen Freiburg mit Jörg Sobottka findet in Freiburg statt.

24. und 25. Mai 2023

Die DOAG 2023 Datenbank mit Exaday findet vom 24. bis 25. Mai 2023 im Van der Valk Airporthotel Düsseldorf statt. Die Besucherinnen und Besucher erwarten zwei Tage Konferenz mit rund 70 Sessions rund um Datenbank und Engineered Systems in Form von Fachvorträgen Erfahrungsberichten und Best Practices. Daneben wird ein Einsteiger-Stream angeboten, zu dem auch Auszubildende und Studenten eingeladen sind. Die Keynote hält Markus Michalewicz von Oracle.

Die Oracle-Anwenderkonferenz

Eventpartner: AOUG SOUG

anwenderkonferenz.doag.org

„Die Vielzahl verbreiteter und gut unterstützter Tools rund um Kubernetes tun ihr Übriges, damit Kubernetes zum aktuellen De-facto-Standard für Neuentwicklungen geworden ist."

Martin Meyer, Redaktionsleiter des Red Stack Magazin, sprach mit dem Softwareentwickler Frank Prechtel über Container- und Cloud-Technologien, Kubernetes, CI/CD und Automatisierung, DevOps-Kultur sowie die Cloud Native Community und das bevorstehende Cloud Native Festival „CloudLand 2023" im Phantasialand.

Bitte stellen Sie sich kurz unseren Lesern vor. Wer sind Sie und mit was beschäftigen Sie sich?

Ich bin freiberuflicher Softwareentwickler aus Nürnberg und bin seit rund einem Vierteljahrhundert in der IT unterwegs. Hauptsächlich beschäftige ich mich aktuell mit der Modernisierung von Legacy-Software aus dem Java-Umfeld und deren Integration in moderne Cloud-Landschaften.

Darauf aufbauend beschäftige ich mich gerne mit dem gesamten Themenkomplex „Cloud Native", von Methodiken über Tools bis hin zu Plattformen, die uns heutzutage bessere Softwareentwicklung ermöglichen.

Wie hat sich die Cloud Native Community seit ihrer Gründung entwickelt?

Die deutschsprachige Cloud Native Community oder kurz DCNC hat den Nerv der Zeit getroffen und die Themen aufgegriffen, die viele (nicht nur) Entwickler heutzutage beschäftigen; dementsprechend Zuwachs hat die DCNC inzwischen bekommen. Neben dem regen Austausch zwischen den einzelnen regionalen Gruppen ist die Organisation des CloudLand-Festivals das Highlight der DCNC-Aktivitäten, auf deren zweite Auflage wir uns im Juni 2023 schon freuen.

Was gehört heute zu „Container- und Cloud-Technologien"?

Mit dem kometenhaften Aufstieg von Linux-Containern (inzwischen standardisiert im Rahmen der „OCI", der „Open Container Initiative" – nicht zu verwechseln mit „Oracle Cloud Infrastructure") ist ein breites Feld an Möglichkeiten eröffnet, die sowohl bei Public-Cloud- Providern als auch im eigenen Rechenzentrum zum Tragen kommen. Eine Anwendung samt all ihren Abhängigkeiten von der Entwicklung bis zur Produktion konsistent in verschiedenen Umgebungen laufen zu lassen, vereinfacht sowohl die Entwicklung (Development) als auch den Betrieb (Operations) ungemein. Vorbei sind die Zeiten von „but it works on my machine".

Nicht zuletzt durch die großen Public Cloud Provider wie Amazon AWS, Microsoft Azure oder Google Cloud hat sich ein riesiges Ökosystem an Tools und Technologien entwickelt, von denen Kubernetes so ziemlich das bekannteste sein dürfte.

Welche Rolle spielt Kubernetes in Bezug auf Containerorchestrierung Ihrer Meinung nach?

Kubernetes ist eine sehr mächtige Plattform, mit allen Vor- und Nachteilen. Zum Preis einer nicht zu unterschätzenden Komplexität beim Betrieb – Kubernetes hat viele bewegliche Teile – löst es zahllose operative Aspekte. Service-Orchestrierung, unterbrechungsfreie Deployments neuer Softwareversionen, Skalierung und vieles mehr sind mit Kubernetes kein Problem.

Zusätzlich hat sich Kubernetes aktuell als der kleinste gemeinsame Nenner etabliert, wenn man containerisierte Anwendungen sowohl im eigenen Rechenzentrum als auch bei einem Cloud-Provider laufen lassen möchte. Die Vielzahl verbreiteter und gut unterstützter Tools rund um Kubernetes tun ihr Übriges, damit Kubernetes zum aktuellen De-facto-Standard für Neuentwicklungen geworden ist.

Wie lässt sich die oft erwähnte DevOps-Kultur erfolgreich in Unternehmen etablieren?

Über Jahrzehnte gewachsene Abläufe lassen sich natürlich nicht über Nacht beseitigen. Für die Etablierung einer DevOps-Kultur gibt es aber DevOps Topologies *(https://web.devopstopo-*

logies.com/ [gut do-kumentierte Ansätze]), wie man organisatorisch dem Ziel Stück für Stück näherkommt.

Die Organisation ist aber nur ein Teil des Ganzen. Aspekte wie ein stärkerer Kundenfokus, eine Fehlerkultur und der Abbau von Silos brauchen auch die Unterstützung auf Management-Ebene.

Selbst wenn eine DevOps-Kultur im Unternehmen noch nicht in Sichtweite ist, kann man auf technischer Ebene schon Themen wie CI/CD und Automatisierung in Angriff nehmen.

Was sollte man über CI/CD und Automatisierung wissen? Wie relevant ist das Thema GitOps?

Mit „Continuous Integration" (die regelmäßige Integration aller Softwarekomponenten samt automatisierten Tests und ebenso automatisierten Softwaremetriken) und „Continuous Delivery" (das automatische Ausspielen der via „CI" erstellten Software in Test-, Integrations- oder sogar Produktionsumgebungen) erspart man sich reichlich fehlerträchtige manuelle Arbeit und schafft kürzere Feedback-Zyklen für die eigene Software.

Sofern man in einer Public Cloud unterwegs ist, kann man mittels „Infrastructure as Code" sogar die für die Anwendung benötigte Infrastruktur bei Bedarf mit Tools wie Terraform *(https://www.terraform.io/)* automatisch provisionieren. Dann ist die die eigene Anwendung betreffende Änderung, egal ob Quellcode oder Infrastruktur, über Git getrieben – daher der Name „GitOps". Sobald man davon gekostet hat, kann man sich den Weg zurück kaum noch vorstellen.

Welche Relevanz kommt dem Thema Observability zu?

Die Softwarelandschaft in Unternehmen wird zunehmend komplexer und unübersichtlicher. Es gibt kaum noch isolierte Systeme, bei denen man mit Logging allein glücklich wird.

Damit man Probleme nicht erst entdeckt, wenn sich Kunden beschweren, sollte man automatisiert „Metriken" wie Reaktionszeit und Fehlerrate der eigenen Anwendungen erfassen und überwachen. Ebenso sollte man via „Tracing"-Mechanismen die anwendungsübergreifende Nachvollziehbarkeit gewährleisten. Mit OpenTelemetry *(https://opentelemetry.io/)* gibt es aber hier zum Glück einen offenen Standard, den man in seine Anwendung integrieren kann.

Was erwartet die Besucher des Cloud-Native-Festivals „CloudLand 2023" dieses Jahr im Phantasialand?

Die Besucher erwarten vier Tage vielfältiges Programm mit Fokus auf interaktiven Formaten, bei denen man in Workshops, Hands-on-Sessions und moderierten Diskussionen seinen Horizont rund um das Thema „Cloud Native" erweitern kann. Im Vordergrund stehen der Erfahrungsaustausch und Wissenstransfer, der im wunderbaren Ambiente eines Freizeitparks auch reichlich Raum für kreative Pausen abseits des offiziellen Programms ermöglicht. Es dürfte im deutschsprachigen Raum kaum eine bessere Möglichkeit geben, sein Wissen rund um das Thema „Cloud" im weitesten Sinne auf Vordermann zu bringen.

FRANK PRECHTEL

Frank Prechtel ist freiberuflicher Softwareentwickler und seit über 20 Jahren in der IT tätig. Wenn er nicht gerade Java-Projekte im Public-Cloud-Umfeld umsetzt, ist er in der DOAG als Vorstand für Querschnittsthemen sowie als Themenverantwortlicher für AWS unterwegs. Er leitet die AWS User Group Nürnberg und unterstützt bei der Organisation der Java User Group Nürnberg und des CloudLand-Festivals.

AUS DER FERNE BETRACHTET:

JAVA – GIBT´S AUCH ANDERSWO

Als Oracle 2010 alle anderen Konkurrenten bei dem Rennen um die SUN-Übernahme überholte, war das für viele eine Überraschung und löste bei nicht wenigen eher ein Unbehagen als einen Freudentaumel aus. Insbesondere die Java-Entwicklergemeinde war wenig amüsiert darüber, dass der „SUN-Style" nun durch den vermeintlich aggressiveren „Oracle-Style" ersetzt werden sollte. Viele gaben keinen Pfifferling mehr auf die Zukunft von Java unter der Regie von Oracle. Auch das berühmte Zitat von Larry Ellison, dass „Java die wichtigste Sache" sei, „die uns seit SQL passiert ist", konnte die Zweifel nur bedingt beseitigen. Als dann die Patentklage gegen Google angestrengt wurde, sahen sich viele Zweifler bestätigt. James Goslings lapidarer Kommentar, dass es „Patentklagen um die Technologie bei SUN nie gegeben hat und solche Klagen auch nicht im generischen Code der Firma gewesen sei", war mehr als eine unverhohlene Kritik des Java-Erfinders.

Rückblickend müssen aber auch die vehementesten Kritiker konstatieren, dass sich die SUN-Technologien fast perfekt in das Oracle-Portfolio eingefügt haben und dass sie auch sehr professionell weiterentwickelt wurden.

Java wurde von Oracle als Open-Source-Projekt weitergeführt und Java 7 als Referenz- Implementierung (2011) freigegeben. Der gesamte Prozess der Weiterentwicklung wurde in Zusammenarbeit mit der sehr aktiven und großen Java-Community – im positiven Sinne – professionalisiert.

Ein wichtiger Schritt wurde 2017 mit der Einführung der halbjährigen Release-Kadenz und der neuen Release-Nummerierung gemacht. Im September 2017 wurde Java 9 freigegeben und seit dieser Zeit wird jeweils im März und September das nächste Release veröffentlicht. Dieser halbjährigen Frequenz neuer Releases wurde eine dreijährige Frequenz überlagert, die dann eine „Langzeitversion" darstellte (LTS = Long Term Support). Die erste LTS-Version – nach diesem neuen Schema – wurde mit Java 11 im September 2018 freigegeben, die wiederum im September 2021 durch Java 17 abgelöst wurde.

Diese Versions- und Release-Strukturierung war wichtig und notwendig. Damit verbunden war jedoch auch die Einführung eines „Preisschildes". Java war immer noch ein Open- Source-Projekt, aber Kunden standen ab diesem Zeitpunkt vor der Frage, wie sie es mit Java in der Zukunft halten sollten: eine kostenlose Open-JDK-Distribution von Oracle oder anderen Anbietern oder eine kostenpflichtige Oracle JDK, die im Kern völlig identisch ist mit Open JDK, aber längere Release-Zyklen, Support und Zusatzprodukte anbietet.

Im Januar 2023 wurde das seit 2017 bestehende Modell nochmals verändert. Oracle bietet drei Varianten desselben Produktes an, die sich jedoch bei der Lizenz und bei der Dauer von Download- und Patch-Verfügbarkeit unterscheiden:

- **Oracle Open JDK** als „GPLv2+CPE"-Lizenz, kostenfrei; das jeweils aktuelle Java-Release kann genutzt werden und hat eine Frequenz von 6 Monaten.
- **Oracle JDK** als „Oracle No Fee Terms and Conditions"-Lizenz (NFTC), die ebenfalls kostenfrei und so lange nutzbar ist, bis eine neue LTS-Version zur Verfügung gestellt wird (plus ein Jahr Karenzzeit). Diese Lizenz bezieht sich immer auf die aktuelle LTS-Version (der dreijährige LTS-Zyklus wurde auf einen zweijährigen Zyklus umgestellt).
- **Oracle JDK** als **„Universal Subscription"**-Lizenz. Diese Lizenz bezieht sich auch auf die LTS-Versionen, ist kostenpflichtig,

bietet eine fünfjährige Laufzeit mit der Möglichkeit, danach einen dreijährigen Extended Support abzuschließen. Alle zwei Jahre wird eine neue LTS-Version freigegeben. Kunden mit dieser Lizenz können zu beliebigen Zeitpunkten auf die zur Verfügung stehenden LTS-Versionen umstellen.

So weit, so gut. Aber insbesondere die „Universal Subscription"-Lizenz bietet Sprengstoff. Es gibt nur eine einzige Metrik, die Oracle für diese Lizenzart bereitstellt: die **Anzahl der Mitarbeiter** des jeweiligen Unternehmens. Unabhängig davon, wie intensiv Java im Unternehmen eingesetzt wird, ergeben sich die Kosten pro Jahr für diese Lizenz gemäß der Formel: ***Anzahl_Mitarbeiter*Kosten_pro_Monat_pro_Mitarbeiter*12.*** Dies ist weit entfernt von dem heute so oft gepriesenen Prinzip des Cloud-Zeitalters: *„Zahle, was Du nutzt"*. Diese Art Flatrate ersetzt zwar das ungeliebte Zählen von Desktops (mit Java) und Server-Cores (mit Java), ist aber ein verdammt grobes Raster und für die meisten Firmen wohl eher uninteressant, weil wesentlich teurer als nach der bisherigen (ebenfalls ungeliebten) Oracle-Preis-Struktur *(siehe auch https://tinyurl.com/JAVAPREIS)* und meist teurer als die kostenpflichtigen Alternativ-Angebote von AZUL und Co.

Damit sind auch die Firmen, die primär auf Oracle-Produkte gesetzt haben (Forms, WLS etc., bei denen die Java-Nutzung kostenfrei ist) und die nur wenige zum Beispiel TOMCAT-Java-Applikationen betreiben, unter massivem Zugzwang, denn speziell für diese Gruppe schlägt die neue Metrik gnadenlos zu.

Anstatt einen Treue-Bonus gibt es einen Treue-Malus. Auch wenn Oracle versucht, die Universal Subscription als Innovation und als Erleichterung zu verkaufen, ist es doch nur ein weiteres Lizenzdesaster.

Aber wie sagt ein altes Sprichwort:

„Chancen präsentieren sich uns mit Vorliebe in der Maske von Unannehmlichkeiten."

GÜNTHER STÜRNER

Oracle Verrazzano: Inbetriebnahme von Containern leicht gemacht

Marcel Boermann-Pfeifer, Oracle Deutschland

Schon seit geraumer Zeit bietet Oracle einige ihrer Kernprodukte als Container-Varianten an. Manches Produkt erfuhr Erweiterungen, um es nativ in eine Microservice-Architektur einzubetten. Und es gibt inzwischen Oracle-Frameworks zum Bau von Microservice-Anwendungen. In der Zwischenzeit sind um die Themen Containerisierung und Microservice-Architekturen valide, bewährte Betriebskonzepte entstanden, die viele wiederkehrende Aufgaben des Anwendungsbetriebes stark automatisieren können und darüber hinaus viele neue Möglichkeiten bieten. Nun ist es an der Zeit, Container mit und ohne Oracle-Software darin in einer umfassenden Unternehmensplattform per Best Practise zu verwalten, zu überwachen und in Anwendungen zusammengefasst auf hybride Umgebungen zu verteilen.

Oracles großer Schritt hin zur Bereitstellung einer umfassenden containerisierten Umgebung für Clouds und On-Premises-Rechenzentren ist Oracle Verrazzano. Groß mag dieser Schritt auch hinsichtlich der gewählten Lizenzform sein: Die Verrazzano-Enterprise-Container-Plattform ist ausschließlich als Open-Source-Komponente verfügbar. Alle Bestandteile von Verrazzano, die enthaltenen Tools und Frameworks, sind ebenfalls Open Source.

Microservice-Architekturen und Anwendungscontainer sind polyglott. Das heißt, es ist egal, mit welcher Programmiersprache die Programme geschrieben wurden, die in den Containern ablaufen. Eine Container-Laufzeitumgebung wie Kubernetes und eine Container-Plattform wie Oracle Verrazzano können mit allen möglichen Containern umgehen. Nutzt man beides in Kombination, reduziert sich der Aufwand und erhöht sich der Komfort enorm. Denn sobald man seine selbst erstellten oder anderweitig bezogenen Container in Betrieb nehmen möchte, kommen recht schnell diverse zusätzliche Tools zum Einsatz, die normalerweise nachzuinstallieren und zu verdrahten wären. Beispielsweise Tools für die Überwachung von Laufzeit-Metriken der Container und der Anwendung selbst, Tools für die Einbindung der Container in das eigene Netzwerk per SSL und über einen Software Load Balancer, Tools für die Analyse von Logging-Informationen jeglicher Art, Tools für Netzwerk-Überwachung innerhalb des Clusters und Visualisierung von Tracing-Informationen. Mit einer Verrazzano-Installation werden die wichtigsten etablierten Tools eingerichtet und verwaltet, also aktualisiert, skaliert, konfiguriert. Zugleich kann die automatisierte, unaufwendige Verdrahtung dieser Tools mit selbsterstellten Containern von Oracle Verrazzano übernommen werden. Dafür sind lediglich die gewünschten Features oder „Traits“, wie es in der mit der Verrazzano-Installation eingerichteten Open-Application-Model-Notation heißt, beim Deployment anzugeben. Neben typischen, gängigen Containern, beispielsweise in Python oder mit Java Frameworks wie SpringBoot, Helidon und Co. erstellt, werden in Verrazzano explizit Container unterstützt, die mithilfe des WebLogic-Migrationskits für Kubernetes generiert wurden oder die eine Coherence-In-Memory-Datenbank enthalten.

Die wichtigsten enthaltenen Tools

- *Betrieb, Continuous Deployment:* Das umfassende Werkzeug **Rancher** wird mitinstalliert, um sich auf einzelne Kubernetes Cluster zu schalten und dort zum Beispiel Ressourcen zu konfigurieren, Anwendungsdeployments beispielsweise via Helm und Git-Integration zu automatisieren, sich auf einzelne Container mit einer Shell zu verbinden, deren Logs einzusehen und vieles mehr.
- *Logging:* Die produktionsreife Installation von **OpenSearch** und **OpenSearch Dashboard**, einer Kombination aus Datenbank und Analyse-Werkzeug für Log-Informationen, ist Teil jeder Verrazzano-Installation. Zusätzliche **fluentd**-Container senden die Log-Informationen aus den Anwendungs-Containern an die OpenSearch- Datenbank. Dazu greift fluentd die Log-Informationen der Container entweder direkt vom Kubernetes Host ab oder bindet sich als sogenanntes „sidecar“ an Container an, die mehr als nur einen Log-Datenstrom befüllen (z. B. WebLogic Server).
- *Monitoring:* Eine Installation der Metrik-Datenbank **Prometheus** samt Auswertungs-Tool **Grafana** steht jeder Verrazzano-Umgebung per Standard zur Verfügung. **Vorinstallierte Dashboards** für Kubernetes Cluster, Coherence, Helidon und WebLogic sind ebenfalls vorhanden. Zusätzliche Dashboards und Datenquellen können nachinstalliert werden, zum Beispiel Oracle Cloud, Oracle-Datenbank und vieles mehr.
- Tracing: Der **Istio** Service Mesh ist integraler Teil der Verrazzano-Installation, er bietet für den gesamten Aufbau einer Anwendung den oft zentralen Einstiegspunkt. Er ermöglicht die Überwachung Kubernetes-interner Netzwerk-Zugriffe und bietet seinerseits Sicherheits-Policies auf Netzwerk-Ebene. Bereitgestellte beziehungsweise im Netzwerk geöffnete Einstiegspunkte in Anwendungen hinein, meist über HTTP/HTTPS Ports, werden Ingresses genannt. Ein Ingress fungiert zusätzlich als Netzwerk-Load-Balancer, falls mehrere Container mit derselben Aufgabe gestartet wurden. Ingresses werden ebenfalls in Form von Traits an eine Anwendungskomponente angehängt. Um interne Netzwerkaufrufe und Call-Stacks im Code der Anwendung analysieren zu können, sind die Tools **Jaeger** und **Kiali** integriert.

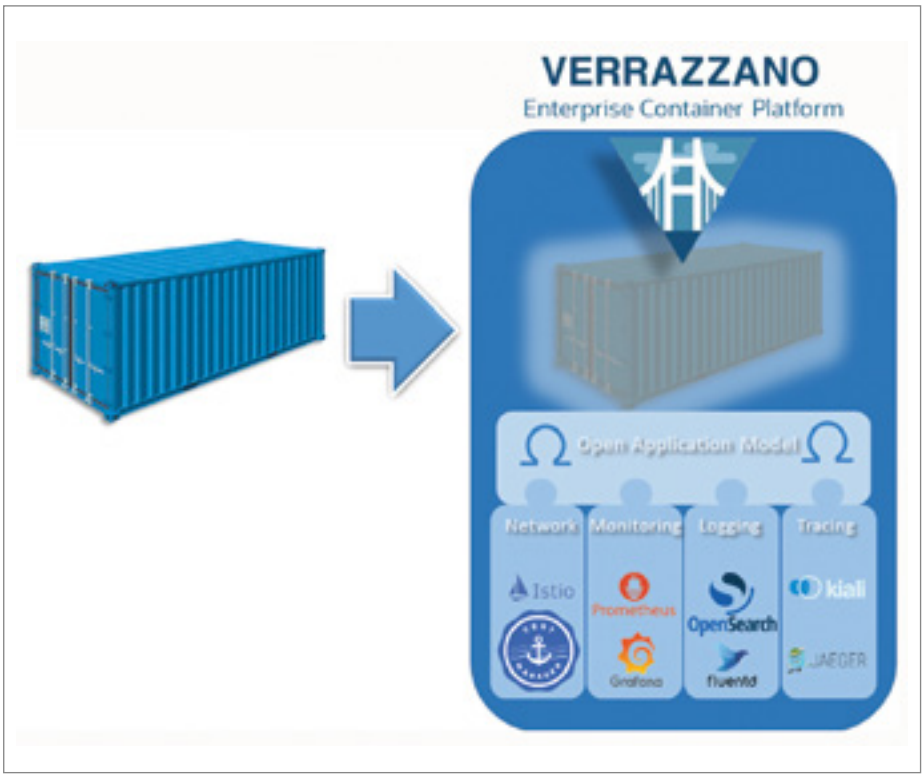


Abbildung 1: Container mit Oracle Verrazzano betreiben (© Oracle Corporation)

- *Security:* Der Netzwerk-Verkehr zwischen den Verrazzano-Komponenten und hin zu den Browser-Clients wird verschlüsselt. Die Single-Sign-on-Komponente **Keycloak** ermöglicht die einheitliche Benutzer- und Rollenverwaltung innerhalb des von Verrazzano verwalteten Clusters. So sind neben eigenen Anwendungen alle Services und Browser-Oberflächen zugriffsgeschützt. Mit Protokollen wie beispielsweise SAML und OpenID Connect oder neben interner Benutzerdatenbank mit einem LDAP-Zugriff auf ein bestehendes Active Directory können Clouds und hauseigene Benutzerverwaltung angeschlossen werden und Anwendungen erhalten auf Wunsch einen einheitlichen, zentralen Login- Bildschirm.
- *Backup:* Das Backup-Tool **Velero** ermöglicht die Sicherung und Wiederherstellung aller Persistent Volumes im Kubernetes Cluster sowie die Sicherung und Wiederherstellung der Kubernetes-Konfigurationsdatenbank, etcd genannt. Ein Plug-in für die Sicherung der OpenSearch-Datenbank wird ebenfalls installiert.
- *Container Registry:* Es ist im Standard-Umfang von Verrazzano **keine Container Registry** enthalten, um Container Images zu verwalten. Der Grund liegt darin, dass im Regelfall in lokalen Rechenzentren und in Cloud-Umgebungen bereits Container Registries beziehungsweise Repositories vorhanden sind, die sehr leicht angebunden werden. Bekannte Beispiele sind in On-Premises-Umgebungen Sonatype Nexus, Harbor oder Artifactory. In Cloud-Umgebungen stehen meist eigene Registries der Cloud-Anbieter zur Verfügung, etwa Oracle Container Registry sowie ähnlich lautende Registries von Google, Azure, Amazon und auch klassische Cloud-Registries wie docker.io.

Die aufgeführten Tools werden vom Verrazzano-Entwicklungsteam miteinander getestet und in das Verrazzano-Paket

Abbildung 2: Verrazzano mit enthaltenen Tools für alle Container-Arten und Umgebungen (© Oracle Corporation)

Abbildung 3: Verrazzano-Versionen und Komponenten-Updates (© Oracle Corporation)

```
apiVersion: core.oam.dev/v1alpha2
kind: Component
metadata:
  name: vz-flask-comp
  namespace: python
spec:
  workload:
    apiVersion: core.oam.dev/v1alpha2
    kind: ContainerizedWorkload
    metadata:
      name: pythonflask
      namespace: python
      labels:
        app: pythonflask
        version: v1
    spec:
      containers:
        - name: app
          image: ilfur/pythonflask:1.0
          ports:
            - containerPort: 5000
              name: pythonflask
          imagePullPolicy: Always
```

Listing 1: Beispiel einer Komponenten-Beschreibung nach der Open-Application-Model-Spezifikation

```
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: pythonflask-appconf
  namespace: python
spec:
      components:
        - componentName: vz-flask-comp
          traits:
            - trait:
                apiVersion: oam.verrazzano.io/v1alpha1
                kind: MetricsTrait
                spec:
                  scraper: verrazzano-system/vmi-system-prometheus-0
                  port: 5000
                  path: "/metrics"
            - trait:
                apiVersion: oam.verrazzano.io/v1alpha1
                kind: IngressTrait
                metadata:
                  name: python-ingress
                spec:
                  rules:
                    - paths:
                        - path: "/"
                          pathType: Prefix
```

Listing 2: Beispiel einer Anwendungs-Beschreibung nach der Open-Application-Model-Spezifikation

integriert. Teilweise werden Tools auch erweitert, zum Beispiel existiert ein Rancher-Plug-in für die Ansicht von Verrazzano-Anwendungen sowie Plug-ins für die Erzeugung neuer Kubernetes Cluster in der Oracle Cloud und in anderen Clouds *(siehe Abbildung 2)*. Wenn vorhanden wird eine aktuelle Patch-Version des jeweiligen Tools einer neuen anstehenden Verrazzano-Version beigefügt, sodass der Verrazzano Kubernetes Operator, die Betriebslogik für alle Verrazzano-Tools, auf Wunsch ein fließendes Update der Umgebung durchführen kann. *Abbildung 3* zeigt den exemplarischen Zusammenhang zwischen Verrazzano-Versionen und Versionen enthaltener Tools.

Verrazzano-Lebenszyklus mit Installation und Patching

Eine **CNCF-zertifizierte Kubernetes-Distribution** *[0]* ab Version 1.19 genügt vollauf als Basis für die Installation. Damit funktioniert Verrazzano beispielsweise auf den Oracle-eigenen Kubernetes-Distributionen für On-Premises, Oracle Container Native Environment (OCNE), und für die Oracle Cloud Infrastructure, Oracle Kubernetes Engine (OKE). Erfolgreich getestet wurde Verrazzano auch mit Minicube, KinD, Azure AKS, Amazon EKS und einigen mehr. Wünscht man den Oracle Premier Support für Verrazzano zu nutzen statt des regen Community Support auf **github.com/verrazzano** *[1]*, ist die Liste unterstützter Kubernetes-Distributionen begrenzt auf die Oracle-eigenen OCNE und OKE, die wiederum auf Oracle Enterprise Linux aufsetzen.

Verwendet man als Basis eine „Cloud-managed" Kubernetes-Distribution, so ist man bereits gewappnet für die Einrichtung von Verrazzano, denn dann sind automatisierte Erzeugung und Anbindung von Storage-Diensten (über vordefinierte Storage Classes) und externe Netzwerk-Anbindung über Load-Balancer-Cloud-Dienste (Services vom Typ „Load Balancer") bereits eingerichtet.

Eine On-Premises-Kubernetes-Installation im eigenen Rechenzentrum sollte hinsichtlich Storage und Netzwerk-Zugriffe von außerhalb zunächst vorbereitet werden. Beispielsweise durch Anlegen einer sogenannten „Default Storage Class" und von ihr verwalteter Persistent Volumes, die von den Verrazzano- Komponenten benötigt werden, um ihre (Meta-) Daten abzulegen. Und durch Installation des „MetalLB"-Load-Balancer, der vorgegebene lokale IP-Adressen verwaltet und sie den Kubernetes-Services zuordnet für den Zugriff von außerhalb.

Die für Kubernetes typische „Helm"-Paketverwaltung muss ebenfalls vorhanden sein, denn alle Verrazzano-Komponenten werden mittels Helm installiert und aktualisiert. Zuletzt ist noch zu beachten, dass

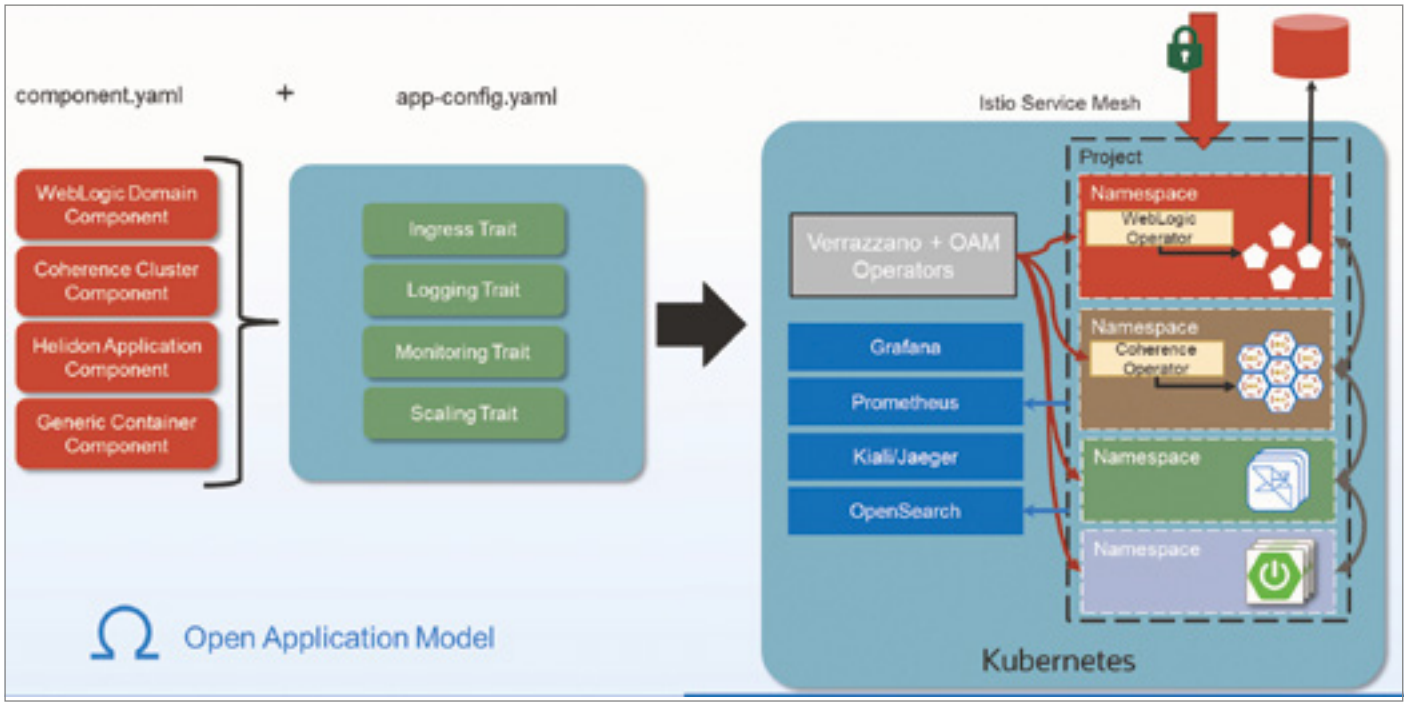


Abbildung 4: Prinzip des Open Application Model (© Oracle Corporation)

nötige Ports für den Zugriff nach außen geöffnet sind, um die installierten Tools über den Load Balancer zu erreichen und auch um Software automatisch herunterzuladen und zu installieren, beispielsweise nach *container-registry.oracle.com*, nach *gcr.io* und auch *github.com/verrazzano*.

Die Installation von Verrazzano läuft in zwei Schritten. Zunächst wird der generische Operator für Verrazzano eingerichtet, der später alle weiteren Komponenten installiert und auch verwaltet, zum Beispiel aktualisiert, was derzeit etwa alle 4 bis 6 Wochen möglich, aber nicht zwingend ist. Im zweiten Schritt wird eine Konfigurationsdatei an den Verrazzano-Operator gesandt (mittels „kubectl apply") mit wichtigen Angaben zur Konfiguration wie beispielsweise Name des Clusters, Netzwerk-Suffix für die Erreichbarkeit der enthaltenen Tools und für bereitzustellende Anwendungen. Dann sind weitere optionale Angaben wie Storage-Größen und SSL- Zertifikate möglich. Daraufhin startet der Operator die eigentliche Installation und lädt Tools herunter, richtet sie ein und verknüpft Load Balancer mit den Standard-Tools. Je nach Netzwerkanbindung und Hardware ist die Prozedur in etwa 7-15 Minuten durchgeführt. Zur Hardware: Es sind zwei, besser drei Worker-Knoten aus Gründen der Hochverfügbarkeit notwendig sowie genügend RAM für Verrazzano-Tools und eigene Container. Beispielsweise 16 GB RAM pro Knoten oder gerne mehr. Vorab werden nur wenige Hundert MB an Storage Volumes reserviert; je nach Lagerungsdauer und täglicher Menge an Logging-Informationen sind in der **Installations-Dokumentation** *[2]* weitere Anweisungen und Größenabschätzungen vorhanden.

Die komfortable Einbindung von Containern: Open Application Model

Um Anwendungs-Container und andere davon benötigte Ressourcen wie Parameter-Listen (ConfigMaps), Dateisysteme (Persistent Volumes), Netzwerkzugänge (Ingresses) usw. in Kubernetes installieren zu können, werden diese, je nach Notwendigkeit detailliert, in Konfigurationsdateien beschrieben. Die Beschreibung erfolgt üblicherweise im spaltenorientierten YAML-Format. Eine vollständige Microservice-Anwendung besteht somit sehr schnell aus sehr zahlreichen einzelnen Ressourcen und vielen YAML-Dateien. Es gibt unter Kubernetes mehrere Möglichkeiten, zusammengehörende Ressourcen zu gruppieren oder sie als derselben Anwendung zugehörig zu kennzeichnen, etwa über Annotationen in den Ressourcen-Beschreibungen. Das sind oft kleine zusätzliche Metadaten im Freitext-Format, die sich später auch durchsuchen, das heißt filtern lassen. Auch die Verbindung der Anwendungskomponenten zur bestehenden Monitoring-, Logging- und Tracing-Infrastruktur erfolgt meist über Annotationen und manchmal über zusätzliche Abschnitte in den Ressourcen-Beschreibungen wie Sidecar-Container, Init-Container usw., die beim Deployment zu berücksichtigen sind.

Um diese Komplexität zu reduzieren, das bedeutet, die Zahl einzeln zu beschreibender oft wiederkehrender Arten von Ressourcen zu minimieren, und um die unterschiedlichen Wege von Verdrahtung und Einbindung in bestehende Infrastruktur zu vereinheitlichen, wurde die **Open-Application-Model-Spezifikation** *[3]* unter anderem von namhaften Mitgliedern wie Microsoft und Alibaba ins Leben gerufen.

Das Open Application Model (OAM) bildet aus Sicht von Kubernetes wenige zusätzliche Ressourcen, um vollständige „Cloud-Native-Applikationen" zu definieren. Diese Anwendungsdefinition bildet eine Klammer um alle von einer Anwendung benötigten Komponenten wie beispielsweise Container-Deployments und Parameter-Listen (ConfigMaps), Kennwörtern (Secrets), Persistent Volumes und vielem mehr. Hinzu kommt eine Wunsch-

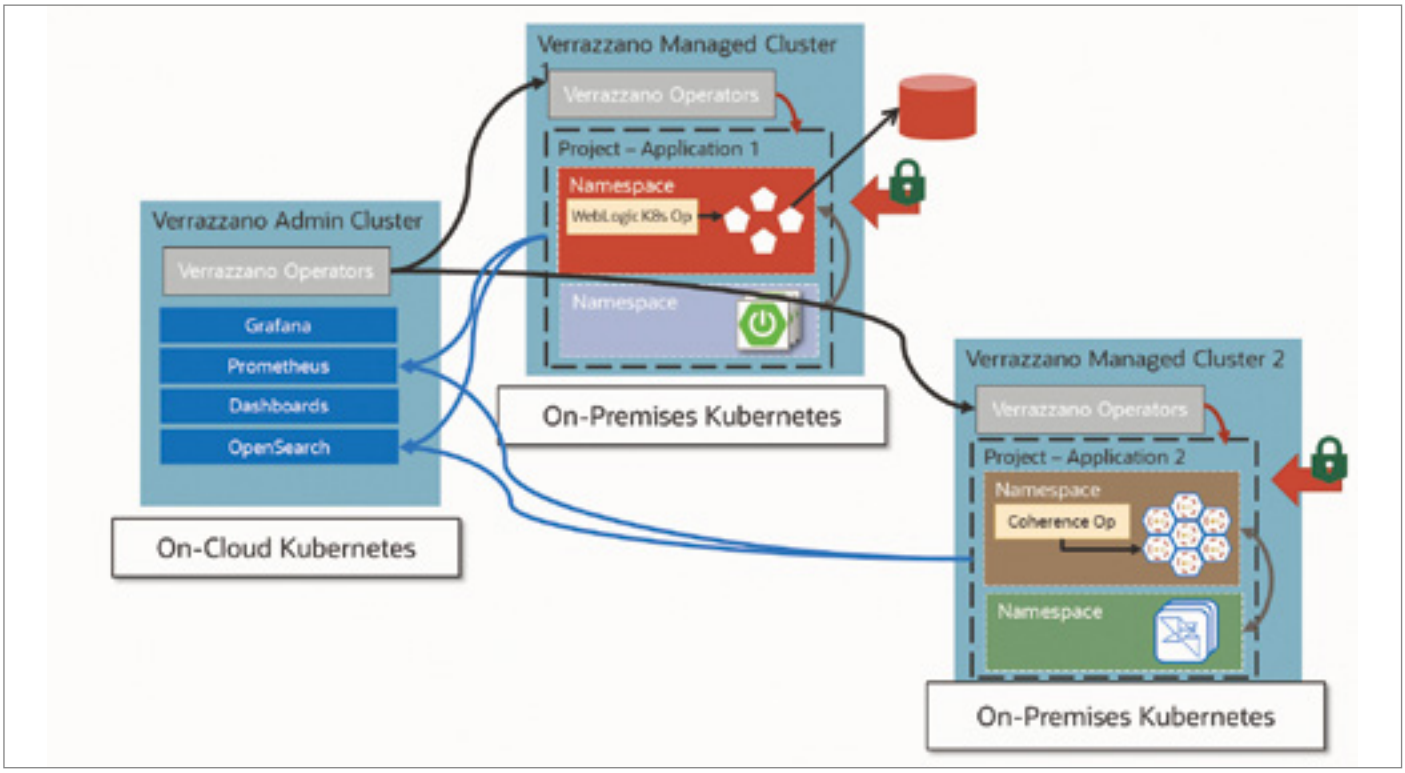


Abbildung 5: Open Application Model und Multi-Cluster-Unterstützung (© Oracle Corporation

Liste an auswählbaren Features für Monitoring, Tracing, Logging und Netzwerk-Zugriffe, sogenannten Traits (Merkmalen/Anhängsel). Mithilfe von Traits wird festgelegt, ob und wie die einzelnen Anwendungskomponenten über einen Load Balancer mit der Außenwelt kommunizieren, ob sie Logging, Tracing und/oder Monitoring benötigen oder eine Skalierungsmöglichkeit. Je nach Art der Komponente, zum Beispiel „WebLogicWorkload", „ContainerizedWorkload", „Deployment", „ConfigMap"..., entscheidet dann die Logik hinter der OAM-Beschreibung, wie sie die Anwendung mit den restlichen vorinstallierten Tools zu verdrahten hat.

Die Logik hinter dem Open Application Model existiert in Form eines Kubernetes-typischen „Operator", einem Container, der sich um die Verwaltung bestimmter Kubernetes-Ressourcen kümmert und sich zu diesem Zweck bei dessen Start im Kubernetes Cluster registriert. Der in Verrazzano enthaltene Operator für Anwendungsbeschreibungen nach der OAM-Spezifikation stammt von Oracle. Es gibt weitere OAM-Operatoren, die nicht im Zusammenhang mit Verrazzano stehen und einzeln betrieben werden können, zum Beispiel **„Crossplane"** *[4]*. Diese bieten jedoch keine Oracle-spezifischen Erweiterungen an wie beispielsweise Workload-Typen für WebLogic, Coherence und Helidon.

Als kleines Beispiel wird in folgenden beiden Beschreibungs-YAMLs ein kleiner in Python geschriebener REST Service in eine Anwendungsbeschreibung mit den Features Netzwerk, Monitoring und Logging verpackt. Zunächst wird der Applikations-Container („docker.io/ilfur/pythonflask:1.0") als Komponente der Anwendung beschrieben *(siehe Listing 1)*.

Darauffolgend wird in der Anwendungsbeschreibung eine Liste zugeordneter Komponenten geführt, mit unserem Python-Container als einziger Komponente. Sie wird im Beispiel „vz-flask-comp" genannt. Im gleichen Zug wird angegeben, welche Betriebs-Features die Komponente nutzen soll *(siehe Listing 2)*.

Wird nun mittels „kubectl apply" erst die Komponenten-, dann die Anwendungsbeschreibung nach Kubernetes gesandt, steht ein vollwertiger, mit SSL abgesicherter und mit Netzwerk-Namen versehener REST Service zur Verfügung, dessen Logs in OpenSearch einsehbar sind und dessen Metriken in Prometheus gesammelt werden. Installiert man eines der frei verfügbaren **Grafana Dashboards für Python und dessen Web Server „Flask"** *[5]* hinzu, sind die Metriken auch sofort visualisierbar.

Alle zugehörigen Ressourcen und Annotationen wie Ingress, Virtual Service, eine sidecar-Konfiguration mit einem fluentd-Container darin zum Abgreifen von Log-Informationen, Annotationen für Prometheus-Server und noch einige Dinge mehr wurden bei Einrichtung der Anwendung vom OAM-Operator generiert. Denn diese normalerweise zusätzlichen Angaben können als immer wiederkehrende Unteraufgaben eines Anwendungs-Deployments automatisiert werden. Das erhöht die Übersichtlichkeit der Anwendung und Zuordenbarkeit ihrer Bestandteile. Auch wäre man nicht gezwungen, alle infrage kommenden Ressourcen im Detail zu beherrschen, um sie richtig zu konfigurieren – das Deployment nach OAM gibt gewissermaßen eine Best Practise für das Anwendungsdeployment vor, die man nachträglich auch wieder anpassen könnte.

Multi-Cluster- und Multi-Cloud-Unterstützung

Obendrein darf jede Anwendung per Beschreibung in einem anderen Kubernetes-Cluster ablaufen, Voraussetzung ist die Erreichbarkeit der Cluster über ein Netzwerk. Ein kleiner Zusatz in der

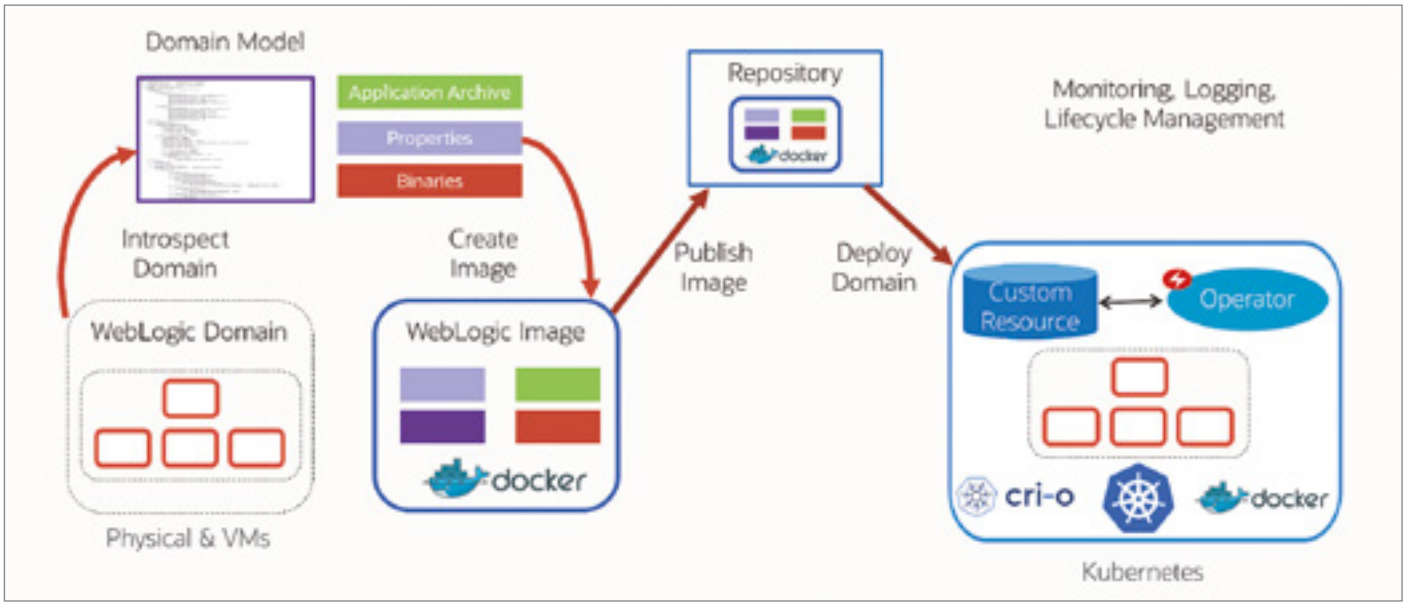


Abbildung 6: Zusammenspiel aller WebLogic Tools bei der (Um-)Konfiguration für den Betrieb unter Kubernetes (© Oracle Corporation)

OAM-Anwendungsbeschreibung zum gewünschten Ziel-Cluster genügt. Logging-Informationen und Metriken werden aus den jeweiligen Clustern zu den zentralen Monitoring-Werkzeugen versandt, die ihrerseits in einem separaten „Management"-Kubernetes-Cluster laufen können. Zu diesem Zweck können Verrazzano-Umgebungen in den Rollen „Admin Cluster" und „Managed Cluster" installiert werden. Ein Managed Cluster enthält nicht alle Betriebstools, sondern registriert sich bei einem zentralen Admin Cluster und schickt seine Metriken und andere Beobachtungsdaten dorthin. In welchem (Cloud-) Rechenzentrum die jeweiligen Cluster betrieben werden, darf frei gewählt werden. Ein kleines Fallbeispiel zwischen einem Cluster auf Azure und einem in der Oracle Cloud wird in einem **Oracle Entwickler-Blog** *[6]* beschrieben *(siehe Abbildung 5)*.

Anwendungsfall: Betrieb von WebLogic-Clustern in Verrazzano

Verrazzano ist eine generische Container-Plattform für alle denkbaren Arten von Container-Workloads, egal in welcher Programmiersprache oder mit welchem Java Framework wie SpringBoot, Micronaut und so weiter sie erstellt wurden. Die enthaltenen Tools und Frameworks lassen sich auf herkömmliche Weise nutzen, aber auch über das Open Application Model bequemer einbinden. Doch speziell für den Einsatz von WebLogic-Applikationsserver-Domänen und Coherence-In-Memory-Clustern bietet Verrazzano eine zusätzliche Unterstützung an, bei dem alle Tools wie OpenSearch und Grafana sowie Frameworks wie das Open Application Model und der WebLogic-Operator für Kubernetes ineinandergreifen.

Im Bereich des WebLogic-Applikationsservers entstanden über die letzten Jahre einige Erweiterungen, die als Open-Source-Komponenten auf *github.com/oracle* zur Verfügung stehen. So ist eine WebLogic Domain, einmal entsprechend strukturiert und aufgegliedert, in der Lage, sich so sauber in eine Microservice-Architektur einzubetten, dass eine eventuelle Umstellung von Java/Jakarta EE auf Microservice-Frameworks eigentlich unnötig wird. Im geringsten Fall kann eine WebLogic Domain von den Skalierungs- und Hochverfügbarkeitsmechanismen einer Kubernetes-Umgebung profitieren sowie rollierende Updates und Patches der Basissoftware und der damit betriebenen Anwendungen durchführen lassen. Dafür ist kein bis kaum noch selbstgeschriebener Code oder Shell-Skript mehr nötig, auch keine zusätzliche Clusterware von Oracle oder anderen Anbietern. Um die Betriebs-Features von Kubernetes mit denen von WebLogic zu verknüpfen und sogar transparent zu machen, wurde ein eigener WebLogic-Kubernetes-Operator erstellt. Er kümmert sich, ganz ähnlich wie der Verrazzano-Operator für das Open Application Model, um eine neue Art von Kubernetes-Ressource. Hier ist es die WebLogic Domain – auch nur „Domain" genannt. Der WebLogic-Operator verwaltet Start und Stopp der WebLogic-Container, führt Skalierungsaufgaben durch und macht rollierende Software-Updates durch fließenden Tausch von bestehenden WebLogic-Containern mit neuen. Alles geschieht im Hintergrund mit Kubernetes-Mitteln, nicht länger mit proprietärem Code. Um WebLogic-Container zu erzeugen und bestehende und auch neue Domain-Konfigurationen in für Kubernetes geeignete Konfigurationen (sogenannte „Models") zu wandeln, existiert das **WebLogic Kubernetes Toolkit (WKT) UI** *[7]*. Dieses wiederum basiert auf dem **WebLogic Deploy Tooling** *[8]*, einer Python-Skriptsammlung zur Erzeugung und zum Patching von WebLogic-Containern, Domain-Beschreibungen und zugehöriger Ressourcen wie Parameter-Listen (ConfigMaps) und Kennwort-Ressourcen (Secrets) für den WebLogic-Operator. **Die WebLogic Remote Console** *[9]* wird vom WKT UI ebenfalls eingebunden, um neue Domain-Konfigurationen im neueren, flexibleren „Model"-Format zu erzeugen oder bestehende Domains einzulesen und in das „Model"-Format zu wandeln. Die WebLogic-Server in den Containern enthalten **Monitoring Exporter** *[10]*, um gängige Monitoring-Werkzeuge wie Prometheus und Grafana zu unterstützen. Alle diese Tools sind unabhängig von Verrazzano einsetzbar mit nahezu beliebigen

Kubernetes Clustern, sind Open Source und im Support für den WebLogic-Server enthalten *(siehe Abbildung 6)*.

Der Vorteil der Verrazzano-Plattform kommt im darauffolgenden Schritt ins Spiel, bei dem eine solche für Kubernetes vorbereitete WebLogic Domain automatisiert mit bestehenden Logging, Monitoring, Tracing Tools zu verdrahten und eine Netzwerk-Infrastruktur automatisch anzubinden ist mit Verschlüsselung, Load Balancing und Einsichten in den anwendungsspezifischen Netzwerk-Verkehr. Hierzu muss die im vorigen Schritt erstellte Domain-Beschreibung für Kubernetes nur noch eingebettet werden in eine Komponenten-Beschreibung und Applikations-Beschreibung im Open-Application-Model-Format. Auch diesen Schritt übernimmt inzwischen das WebLogic Migration Toolkit UI auf Wunsch, wenn man als Ziel des Domain-Deployments statt „Kubernetes" nun „Verrazzano" angibt.

Fazit und Ausblick

Oracle Verrazzano ist eine Open-Source-Container-Plattform für den Betrieb generischer Container unter Kubernetes. Mit Verrazzano werden die wichtigsten, gängigen Tools und Frameworks für den Containerbetrieb gepflegt, miteinander getestet, integriert und in „agilen" Zeitperioden aktuell gehalten. Darüber hinaus besitzt Verrazzano Automatismen und Vereinfachungen beim Deployment von containerisierten und Microservice-basierten Anwendungen. Es müssen nicht alle denkbaren abhängigen Ressourcen auskonfiguriert und ausformuliert werden; das Open Application Model und der zugehörige Operator für Kubernetes erzeugen eigenständig abhängige Ressourcen und übernehmen die oft aufwendige und doch immer wiederkehrende Verdrahtung mit Logging-Monitoring- und Netzwerktools.

Diese Mechanismen und Tools zusammengenommen ergeben eine komfortable Plattform für den Betrieb von Oracle-eigenen, containerisierten und Microservice-basierten Produkten wie WebLogic, Coherence und Helidon, aber eben längst nicht nur die Oracle-eigenen Produkte.

Oracle Verrazzano wird aktuell weiterentwickelt in Richtung einer noch engeren Integration sowohl mit der Oracle Cloud als auch mit dem Oracle Cloud Native Environment (OCNE). Immer mehr bestehende Services der Oracle Cloud sind in Verrazzano anbindbar und können alternativ zu Komponenten in Verrazzano eingesetzt werden. Zum Beispiel lässt sich die OpenSearch-Installation in Verrazzano ersetzen durch Anbindung des entsprechenden OpenSearch-Cloud-Dienstes in der Oracle Cloud. Eine Kopplung des Keycloak Identity Broker mit Identity Domains der Oracle Cloud ist ebenfalls aktuell möglich. Ein Managed Kubernetes (OKE) mit eingerichtetem Verrazzano wird angedacht, wahrscheinlich in Kombination mit einem WebLogic-auf-OKE-Angebot.

Auch die Installation der On-Premises-Kubernetes-Distribution OCNE wird enger verknüpft mit der Einrichtung von Verrazzano. So sollen Überlappungen beziehungsweise doppelt verfügbare Komponenten ausgeräumt werden wie zum Beispiel Istio und Grafana; eine Installation von OCNE und Verrazzano soll ruckelfrei aus einem Guss erfolgen können, voraussichtlich mit Verrazzano 2.0 im Sommer 2023.

Quellen

[0] CNCF-zertifizierte Kubernetes-Distributionen *https://www.cncf.io/certification/software-conformance/*

[1] Oracle Verrazzano auf github.com *https://github.com/verrazzano*

[2] Oracle-Verrazzano-Dokumentation *https://docs.oracle.com/en/cloud/iaas/verrazzano/*

[3] Open-Application-Model-Spezifikatio *https://oam.dev/*

[4] Crossplane OAM Operator *https://blog.crossplane.io/tag/oam/*

[5] Grafana Dashboard für Python und Flask *https://github.com/pilosus/prometheus-client-python-app-grafana-dashboard*

[6] Verrazzano Multicluster zwischen Oracle OKE und Azure AKS *https://blogs.oracle.com/developers/post/multiclustering-between-oci-and-azure-with-verrazzano*

[7] WebLogic Kubernetes Toolkit UI *https://github.com/oracle/weblogic-toolkit-ui*

[8] WebLogic Deploy Tooling *https://github.com/oracle/weblogic-deploy-tooling*

[9] WebLogic Remote Console *https://github.com/oracle/weblogic-remote-console*

[10] WebLogic Monitoring Exporter *https://github.com/oracle/weblogic-monitoring-exporter*

Über den Autor

Marcel Boermann-Pfeifer beschäftigt sich gerne mit diversen Datenbank-nahen APIs für Java und ganz generell mit den Themen rund um Systemintegration, Datenintegration, klassische Middleware und Microservices bzw. Containerisierung. Mit großer Begeisterung unterhält er sich über Familie, Wu Shu, Traditionelle Chinesische Medizin, östliche und westliche Philosophie, kreatives Werken u. a. am KFZ und spielt manches Rhythmus-Instrument wie Bodhrán und Waschbrett.

Marcel Boermann-Pfeifer
marcel.pfeifer@oracle.com

Oracle Zero Downtime Migration (ZDM)

Sinan Petrus Toma, Oracle

Immer mehr Kunden migrieren ihre Oracle-Workloads in die Oracle Cloud oder auf Engineered Systems. Migrationen haben schon immer viele Herausforderungen mit sich gebracht. Dabei kann es um Migration von Datenbank-Workloads von einem System auf ein anderes oder auch um Migrationen in die Cloud gehen. Basierend auf jahrelanger Erfahrung bei der Migration von Oracle-Workloads hat Oracle das Tool Zero Downtime Migration (ZDM) entwickelt. ZDM ist die von Oracle empfohlene Lösung für eine vereinfachte und automatisierte Datenbankmigration, die ohne zusätzliche Kosten nutzbar ist. Mit ZDM können Oracle-Datenbanken direkt und nahtlos auf und zwischen jeder Oracle-eigenen Infrastruktur migriert werden, einschließlich Exadata On-Premises, Exadata Cloud at Customer (ExaC@C) und Oracle Cloud Infrastructure (OCI) inklusive der Oracle Autonomous Database. Oracle ZDM unterstützt eine breite Palette von Oracle-Datenbankversionen und gewährleistet, wie der Name schon sagt, minimale bis keine Ausfallzeit der Produktionsdatenbank während der Migration. Die ZDM-Software kann unter diesem Link kostenfrei heruntergeladen werden: *https://www.oracle.com/goto/zdm*

Das Konzept

Das Design von ZDM ist darauf ausgerichtet, den Migrationsprozess so einfach wie möglich zu gestalten, um eine möglichst geringe Auswirkung auf die Produktions-Datenbanken zu gewährleisten. ZDM automatisiert den gesamten Migrationsprozess und reduziert so das Risiko menschlicher Fehler.

Oracle ZDM führt vor und nach der Migration umfangreiche Prüfungen durch, ermöglicht es, den Migrationsprozess bei Bedarf anzuhalten und fortzusetzen, und enthält einen Evaluierungsmodus, um Problemen während der Datenbankmigration vorzubeugen.

ZDM nutzt in Oracle-Datenbanken integrierte Hochverfügbarkeitstechnologien (HA) wie Oracle Data Guard und Oracle GoldenGate und folgt allen Maximum Availability Architecture (MAA) Best Practices, die sicherstellen, dass es in Produktionsumgebungen zu keinen nennenswerten Ausfallzeiten kommt.

Die Funktionen

Es können sowohl einzelne als auch mehrere Datenbanken gleichzeitig migiert werden. Die Verbindungsdaten zur Quell- und Zieldatenbank sowie weitere Parameter werden in einer Konfigurationsdatei angegeben.

Oracle ZDM bietet unter anderem folgende Funktionsmöglichkeiten:

- Auditing: Alle Aktionen werden auditiert, einschließlich der Aktionen, die durch den Migrationsprozess ausgeführt werden.
- Anpassung des Migrationssablaufs: Der Migrationsablauf (gekennzeichnet durch Migrationsphasen) kann mit eigenen Skripten, die vor oder nach jeder Phase des Migrationsprozesses ausgeführt werden, angepasst werden.
- Job Scheduler: Die Migrationen können geplant und zu einem zukünftigen Zeitpunkt ausgeführt werden.
- Pausen- und Fortsetzungsfunktion: Migrationen können pausiert und bei Bedarf wieder fortgesetzt werden, was zum Beispiel zur Einhaltung eines Wartungsfensters nützlich ist.
- Wiederholung der Migration: Migrationen können von einem Fehlerpunkt aus erneut ausgeführt (fortgesetzt) werden, ohne die vorherigen Migrationsphasen zu wiederholen.
- Migrationvorprüfung: Vorprüfungen können durchgeführt werden, um Fehlern während der Migration vorzubeugen.

Mögliche Quell- und Ziel-Datenbanken

Oracle ZDM unterstützt eine breite Palette von Oracle-Datenbankversionen als Quellen und Oracle Database Cloud Services als Ziele.

Kunden haben eine große Auswahl an Oracle-Datenbank-Quellen für ihre Migration; Oracle ZDM unterstützt Oracle-Datenbanken auf Solaris, Linux und AIX.

Die zu migrierende Quelldatenbank kann On-Premises, in der Oracle Public Cloud Gen 1, in der Oracle Cloud Infrastructure (OCI), auf einer Non-Oracle Cloud oder auf AWS RDS sein.

Die Zieldatenbank kann in einem Base Database Service, Exadata Database Service auf Dedicated Infrastructure, Exadata Database Service auf Cloud@Customer oder Autonomous Database auf Shared oder Dedicated Exadata Infrastructure bereitgestellt werden.

Oracle ZDM unterstützt sowohl Standard-Edition- als auch Enterprise-Edition-Datenbanken. Die Datenbanken können Single-Instanz, Oracle-RAC-One-Node oder Oracle RAC sein.

Migrationsszenarien

Oracle ZDM unterstützt sowohl physische als auch logische Migrationsszenarien.

- Physische Migrationen basieren auf RMAN-Sicherung und -Wiederherstellung sowie Oracle Data Guard. Quell- und Zieldatenbank müssen die gleiche Datenbankversion nutzen und auf einer kompatiblen Plattform laufen.
- Logische Migration hingegen basiert auf Data Pump und Oracle GoldenGate. Quell- und Zieldatenbank können dabei unterschiedliche Versionen und Plattformen nutzen. Dadurch kann gleichzeitig ein Upgrade der Datenbank erreicht werden.

Physische Offline-Migration

Die physische Offline-Migration nutzt RMAN und migriert die Datenbank mithilfe einer Sicherungs- und Wiederherstellungsmethodik *(siehe Abbildung 1)*.

Physische Online-Migration

Die physische Online-Migration mit ZDM nutzt RMAN für die initiale Migration und Oracle Data Guard, um Quell- und Zieldatenbanken synchron zu halten. Die Applikation kann die Quelldatenbank (die Primary) uneingeschränkt nutzen, bis der finale Migrationsschritt durch ein einfaches Data Guard Switchover erfolgt. Danach verbindet sich die Applikation an die Zieldatenbank (die neue Primary) *(siehe Abbildung 2)*.

Kunden können eine vorhandene Sicherung verwenden, um die Last auf

```
MIGRATION_METHOD=ONLINE_PHYSICAL
DATA_TRANSFER_MEDIUM=OSS
HOST=https://swiftobjectstorage.eu-frankfurt-1.oraclecloud.com/v1/tenancy
OPC_CONTAINER=zdmbucket
TGT_DB_UNIQUE_NAME=CDB001_fra26k
PLATFORM_TYPE=VMDB
```

Listing 1: Konfigurationsdatei für die physische Online-Migration

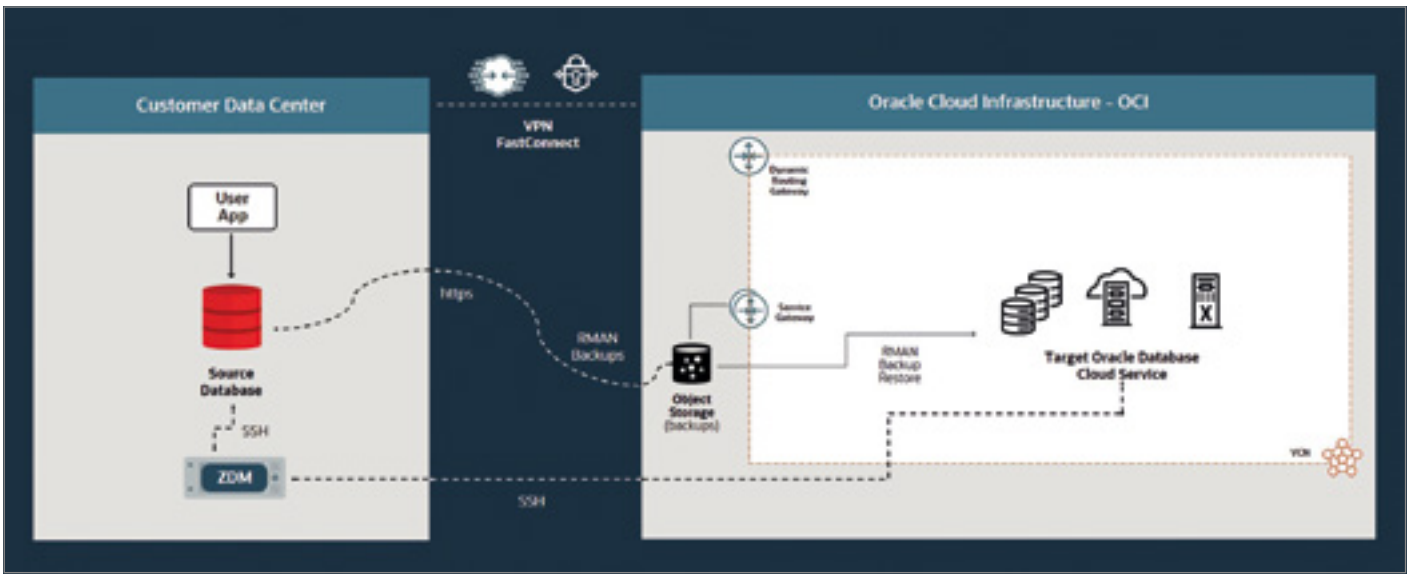


Abbildung 1: Architektur der physischen Offline-Migration von ZDM (Quelle: Sinan Petrus Toma)

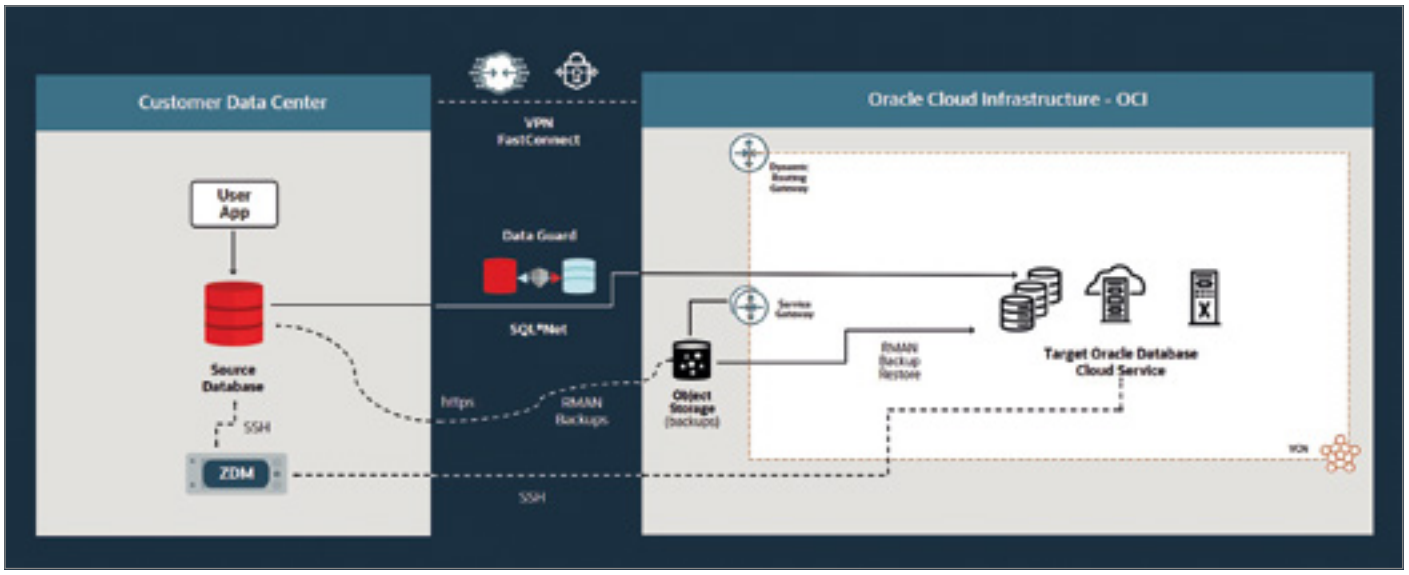


Abbildung 2: Architektur der physischen Online-Migration von ZDM (Quelle: Sinan Petrus Toma)

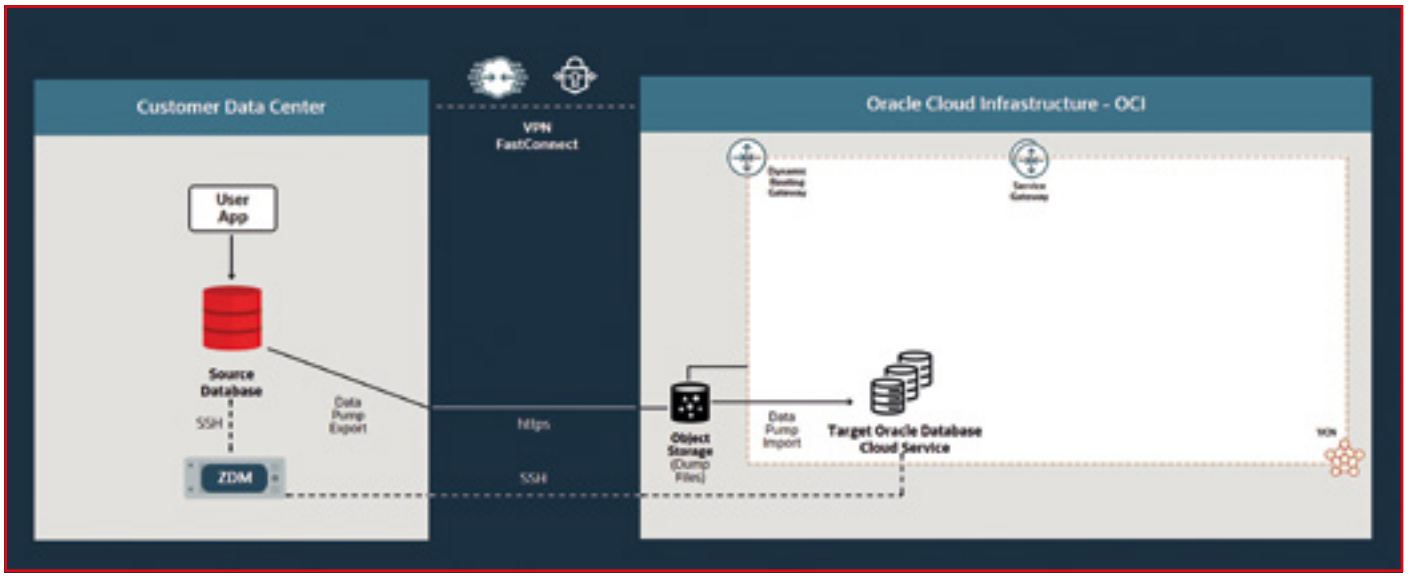


Abbildung 3: Architektur der logischen Offline-Migration von ZDM (Quelle: Sinan Petrus Toma)

```
$ZDMHOME/bin/zdmcli migrate database
-rsp /home/zdmuser/physical_online.rsp
-sourcesid CDB001
-sourcenode source
-srcauth zdmauth
-srcarg1 user:opc
-srcarg2 identity_file:/home/zdmuser/.ssh/id_rsa
-srcarg3 sudo_location:/usr/bin/sudo
-targetnode target
-tgtauth zdmauth
-tgtarg1 user:opc
-tgtarg2 identity_file:/home/zdmuser/.ssh/id_rsa
-tgtarg3 sudo_location:/usr/bin/sudo
-targethome /u01/app/oracle/product/19.0.0.0/dbhome_1
-backupuser "oracleidentitycloudservice/sinan.petrus.toma@oracle.com"
```

Listing 2: Start-Befehl für die Migration

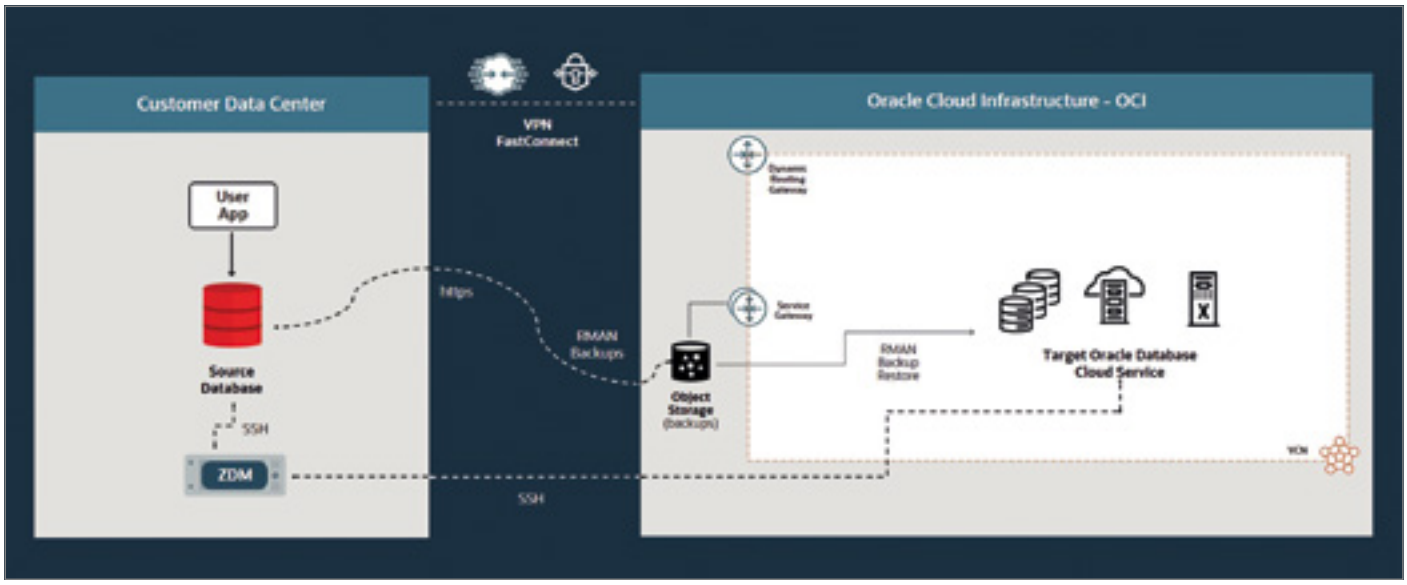


Abbildung 4: Architektur der logischen Online-Migration von ZDM (Quelle: Sinan Petrus Toma)

das Produktionssystem und die Migrationszeit zu minimieren, was vor allem bei sehr großen Datenbanken sehr nützlich sein kann. Oracle ZDM bietet hinzu einen direkten Datentransfer (RESTORE FROM SERVICE), um die Verwendung einer Sicherung zu vermeiden und so eine schnellere und effizientere Migration zu erreichen.

Diese Methode wird empfohlen, wenn eine hochverfügbare Migration und die Minimierung möglicher Auswirkungen Priorität haben.

Logische Offline-Migration

Die logische Offline-Migration mit Data Pump bietet Kunden eine einfache und effiziente Methode, um ihre Datenbanken in die Oracle Cloud zu migrieren, insbesondere in die Oracle Autonomous Database.

Die Daten werden aus der Quelldatenbank extrahiert und in eine Zieldatenbank geladen. Der Sicherungsort für die Data-Pump-Dump-Dateien kann der OCI Object Storage für Oracle-Cloud-Datenbanken oder NFS oder die Recovery Appliance für Exadata Cloud@Customer sein.

Zudem können Database-Links genutzt werden, um eine direkte Verbindung zwischen der Quell- und der Zieldatenbank herzustellen, wodurch die Notwendigkeit eines Sicherungsortes entfällt. Diese Methode wird nur für Quelldatenbanken mit einer Datenbankgröße von weniger als 100 GB empfohlen *(siehe Abbildung 3)*.

Logische Online-Migration

Die logische Online-Migration nutzt Data Pump für die Instanziierung der Zieldatenbank und Oracle GoldenGate für die Synchronisierung von Quell- und Zieldatenbank. Dabei bleibt die Quelldatenbank für Applikationen durch den gesamten Migrationsprozess verfügbar. Da Oracle GoldenGate eine Aktiv-Aktiv-Architektur nutzt, muss lediglich die Applikationsverbindung auf die Zieldatenbank umgeleitet werden *(siehe Abbildung 4)*.

Das Oracle Cloud Marketplace Image für Oracle GoldenGate beinhaltet eine

Bildquelle © akitada31 (www.pixabay.com)

Lizenz für eine Migrationsdauer von bis zu 183 Tagen.

Beispiel

Die Konfigurationsdatei für die physische Online-Migration kann wie in *Listing 1* dargestellt aussehen.

Die Migration wird mit dem in *Listing 2* ersichtlichen Befehl gestartet.

Dabei referenziert `/home/zdmuser/physical_online.rsp` die Konfigurationsdatei.

Zusammenfassung

Mit Oracle ZDM können Oracle-Datenbanken direkt aus verschiedenen Quelldatenbanken in die Oracle Cloud und nach Exadata On-Premises einfach und automatisiert migriert werden.

Oracle ZDM macht manuelle Konfigurationen und Operationen überflüssig und sorgt so für eine fehlerfreie und effiziente Migration zu Oracle Cloud oder Oracle Exadata On-Premises.

Oracle ZDM ist mit der Oracle Maximum Availability Architecture (MAA) kompatibel; die enge Integration mit Oracle-Datenbanktechnologien wie Oracle Data Guard und Oracle GoldenGate stellt sicher, dass Migrationen ohne Ausfallzeiten und ohne Auswirkungen auf die Produktion abgeschlossen werden.

Quellen

[1] *https://www.oracle.com/goto/zdm*

Über den Autor

Sinan Petrus Toma ist Mitglied des Produktmanagement-Teams für Datenbank-Hochverfügbarkeit, Skalierbarkeit, Maximum Availability Architecture (MAA) und Datenbank-Migration bei Oracle. Er begeistert sich für die Oracle-Datenbank und die Oracle Cloud, spricht auf Konferenzen und scheibt gerne Blog-Einträge unter *https://database-heartbeat.com/*, um das Wissen in der Oracle Community zu teilen.

Sinan Petrus Toma
sinan.petrus.toma@oracle.com

www.cloudland.org

CloudLand

DAS FESTIVAL DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY 2023

• Microservices & DDD • CI/CD & Automatisierung
• Container & Cloud-Technologien • DevOps & Methodik

20. - 23. JUNI
im Phantasialand in Brühl

TAG 1

CLOUDCAMP

#CloudLand2023

Eventpartner: Heise Medien

Container entmystifiziert, neue Runde

Jan Karremans, EDB Postgres

Warum?

Warum dieser Artikel, warum dieses Thema? In den nächsten Absätzen würde ich Sie gerne auf eine Reise in die Zukunft mitnehmen. Eine Zukunft, die jetzt noch von einigen für Wahnsinn gehalten wird, trotzdem für viele schon tägliche Realität ist.

Auf unterschiedlichen DOAG-Events haben Sie bestimmt Vorträge gehört, die das Thema erläutert haben. 2019 hat Daniel Westermann mit mir zusammen „Container entmystifiziert" – die erste Runde war schon gemacht. Diese „neue Runde" nun baut auf dieser Grundlage weiter auf, weil sich in den letzten drei bis vier Jahren unheimlich viel geändert hat.

Die Situation

Es ist noch nicht lange her, dass die Aussage „Macht bitte keine Datenbanken mit Containern" stimmte.

Container-Technologie war nicht so weit entwickelt, dass man zuverlässig und risikofrei eine Datenbank fahren konnte. Nicht nur die Geschichte rund um eine

gute Speicheranbindung hat zu grauen Haaren geführt, auch zusätzliche Komplexität mit Bezug zu Docker und der Betrieb einer vernünftigen Container-Management-Plattform haben zu vielen Problemen geführt.

In der ersten Runde haben wir die Geschichte von Containern angeschaut und vielleicht noch nicht direkt die Verbindung zu Datenbanken hergestellt. Das aber hat sich seit 2019 massiv weiterentwickelt, vor allem im Open-Source-Bereich. Der Fokus lag damals auf dem „Wie" von Containern, trotz einer Demo von Postgres in Docker.

„In life, change is the only constant" gilt laut Heraclitus, einem griechischen Philosophen. Okay, eigentlich hat er gesagt: „Στη ζωή, η αλλαγή είναι η μόνη σταθερά", doch das lässt sich nicht so einfach lesen. So sind Unternehmen andauernd auf der Suche nach „Mehr mit weniger". Das Wachstum von „Cloud Deployment" ist ein wesentlicher Treiber für das Wachstum von Container-Technologie. Damit sind nicht nur die öffentlichen Clouds von Anbietern wie Microsoft oder Amazon gemeint, sondern vor allem auch lokale Installationen von sogenannter Hyper Converged Hardware.

Monolithisch zu Agil

Wozu betreiben wir eine IT-Infrastruktur? Die Antwort ist genauso logisch wie auch einfach: Wir betreiben eine IT-Infrastruktur, um den Nutzern Anwendungen anbieten zu können.

Früher arbeiteten Anwendungsbenutzer nur innerhalb des Unternehmens. Ganz einfach und bequem, von 9 Uhr morgens bis 5 Uhr nachmittags. Okay, es gab Ausnahmen für Anwendungen, die vielleicht auch öfter benutzt werden, aber alles schön überschaubar. Überschaubar doch vor allem innerhalb des Kommunikationsbereichs.

Was ich damit meine: Wenn eine Anwendung gewartet werden musste, konnte man darüber Regelungen vereinbaren – Termine für Wartung, Termine für Updates, Termine für Downtime.

Schön bequem organisiert, ohne dass es zu großen Problemen führte.

Mit einem massiven Boost in der Corona-Epoche, ist „online" jetzt der Standard. Von Ihrer Bank bis zu Ihrem Bäcker, von Ihrem Versicherer bis zu Ihrem Autohändler, jeder hat eine Online-Präsenz. Alle Anbieter versuchen, mit Service zu wachsen und gleichzeitig Kosten zu sparen. Mit dem Selbst-Service kann man sein Geld verwalten, Backwaren bestellen, die Lebensversicherung verwalten oder sich in nur fünf Klicks ein neues Auto kaufen.

Das funktioniert allerdings nur dann, wenn die Anwendung da ist, wenn sie gebraucht wird! Nicht von 9 Uhr morgens bis 5 Uhr nachmittags, sondern rund um die Uhr, jeden Tag, von überall zugreifbar. Und das ist nur ein Aspekt! Wie vereinbart man mit Frau Schulz oder mit Herrn Fischer, dass es ein Deployment der App gibt, nächste Woche Mittwoch um 16:30 Uhr?

Frau Schulz und Herr Fischer sind als Benutzer der App nicht einzeln erreichbar. Schlimmer noch: Weil die App, unsere App, am Mittwoch um 16:30 nicht verfügbar ist, klickt der Suchende einfach zur Website und App unseres Wettbewerbers. Das ist jedoch nicht akzeptabel.

Hier hat man dann die Grundlage zur agilen Anwendungsentwicklung. Anwendungen, die für die Einführung neuer Features nicht abgeschaltet werden müssen. Anwendungen, die nicht down gehen für ein Update oder für die Wartung... Immer verfügbar, immer für die Benutzer bereit.

Agile, Microservices, DevOps, CI/CD.

Es war mein Kollege Gabriele Bartolini, der dies als Grafik dargestellt hat. Meiner Meinung nach gibt sie einen wunderschönen Überblick über das, was Cloud Native Mindset bedeutet *(siehe Abbildung 1)*.

Klassische monolithische Anwendungen werden aufgeteilt in kleinere Partikel. Diese sogenannten Microservices sind einfacher zu warten und werden so dargestellt, dass sie meist unabhängig von anderen Microservices im gleichen Bereich sind. So ist man in der Lage, eine Anwendung teilweise und am liebsten online zu erweitern. Wenn man beim Online-Banking zum Beispiel die Funktion zum Hinzufügen eines Zahlungsempfängers ändern muss, funktioniert die Überweisungsfunktion ohne Unterbrechung weiter. Das ist ein anderer Microservice, der nicht von dieser Entwicklung betroffen ist. Alle diese Begriffe werden in einer Mischung benutzt: Agile, Microservices, DevOps und Ähnliches. Aber was bedeutet das eigentlich?

- **AGILE** ist ein iterativer Ansatz für Projektmanagement und Softwareentwicklung, der Teams hilft, ihren Kunden schneller und mit weniger Kopfschmerzen einen Mehrwert zu bieten. Das heißt, Agile beschreibt, wie man das Management seiner Microservices-Entwicklung betreibt, und nicht, was man da so alles macht. Es führt in diesem Artikel zu weit, einen Mini-Kurs zum Thema „Agile Softwareentwicklung" zu schreiben. Es reicht erst mal zu verstehen, dass es mit Containern zunächst gar nichts zu tun hat.
- **MICROSERVICES** sind Anwendungsfunktionen, die mehr oder weniger selbstständig sind. Sie sind einzeln wartbar und erweiterbar, sodass die gesamte Anwendung einfach wächst und gesund bleibt, ohne dafür abgeschaltet werden zu müssen. Es beschreibt, was man baut.
- **DEVOPS** ist die Werkstatt. Geschwindigkeit oder „Velocity", wie die Amerikaner es nennen, ist heute ein Merkmal für Erfolg. Es gibt keine Zeit, eine klassische „Waterfall"-Entwicklung zu machen mit Entwurf, technischem Entwurf, Entwicklung, Testen, Verbesserungen, Akzeptanz und Betrieb. Die Anwendung ist schon veraltet, bevor sie verfügbar ist. Damit geht die Trennung von Entwicklung und Betrieb auch nicht mehr und man kommt zu DevOps. Es beschreibt den Weg, worauf man Anwendungen baut und betreibt.
- **CI/CD** ist kurz für „Continuous Integration/Continuous Delivery" und beschreibt den Status einer Anwendung in Entwicklung. Es ist eigentlich jede Anwendung kontinuierlich in Entwicklung, vor allem in einer Cloud-Native-Welt. Die Schwierigkeit dabei ist nicht Continuous Integration, die fordert, dass man bei jeder Anpassung diese Änderung automatisch testet und in der Anwendung bereitstellt. Die Schwierigkeit ist Continuous Delivery – dies erfordert, dass die Anwendung immer in einem solchen Zustand ist, dass man sie auch immer und sofort produktiv einsetzen kann. Man erreicht diese Situation durch ein gutes Zusammenspiel mit Continuous Integration, dieser Kreislauf ist in der gezeigten *Abbildung 1* auch relativ leicht zu erkennen.

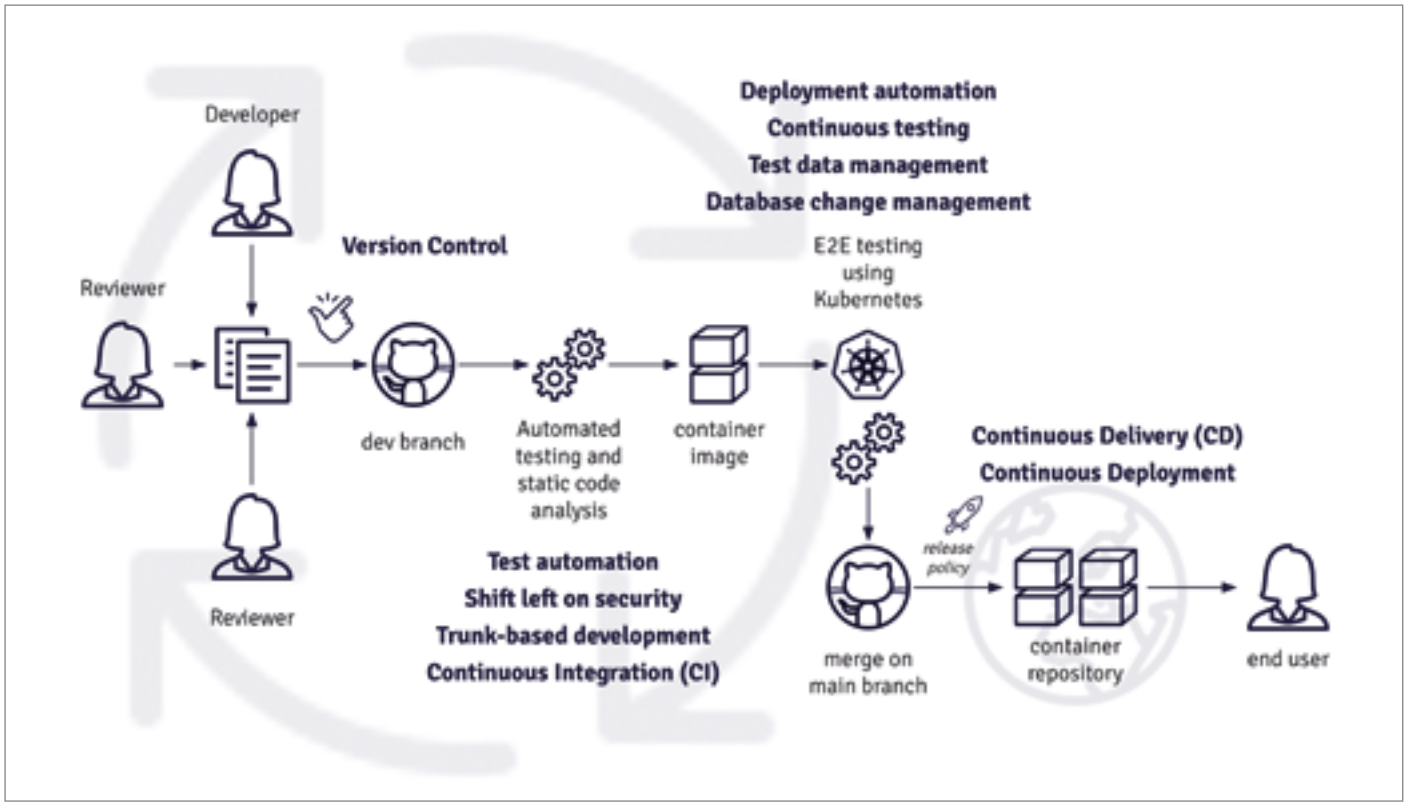


Abbildung 1: Kreislauf im Cloud Native Mindset (Quelle: Gabriele Bartolini)

- **CLOUD NATIVE** ist damit keine Methodik und vor allem hat sie nichts mit Cloud zu tun.

Cloud Native ist eine Denkweise und eine Einstellung, wie man sich Anwendungen vorstellt – anders als bei der Waterfall-Methodik, mit der man früher eine Anwendung baute: Die Anwendung war schon veraltet, bevor man fertig war. In dieser Zeit, mit neuen Anforderungen in Lichtgeschwindigkeit, und Anwendungen, die 24/7 verfügbar sein sollten, gibt es diese ruhige Zeit schon längst nicht mehr.

Microservices

Das Cloud-Native-Mindset und die Entwicklungsvorgabe heißt, dass man eine traditionelle Anwendung aufteilt in eine (vielfältige) Menge von kleineren logischen Teilen. Man kann jeden Teil als eine eigene Anwendung in der Gesamt-Anwendung sehen. Ein Beispiel für die populäre Webanwendung LinkedIn ist die „Group"-Anwendung. Eigentlich ist das schon eine größere Anwendung innerhalb der LinkedIn-Plattform.

Doch in dieser „Group"-Anwendung gibt es eine Menge weiterer Funktionen, die als Microservice betrachtet werden können. Als Beispiel kann man hier die „Create-Group"-Funktion betrachten. Diese kann man dann auch individuell erweitern oder verbessern, ohne dass es dafür ein komplett neues Release der LinkedIn-Plattform erfordert.

Es ist damit auch nicht mehr schwierig sich vorzustellen, wie man eine Anwendung bauen kann, die tatsächlich nie wieder unverfügbar sein sollte! Mit einer vernünftigen Aufteilung der Anwendung in Microservices, mit der Verwendung des Cloud-Native-Mindsets, kann man sogar noch weitere coole Sachen einsetzen als Canary Testing oder ein A/B-Test, wo es die Möglichkeit gibt, unterschiedliche Entwicklungsrichtungen in gezielten praktischen Setzungen auszuprobieren. Persönlich halte ich das für sehr interessant!

DORA

In der heutigen Zeit von „Data Driven" sind KPIs, oder Kennzahlen, ein ganz natürlicher Weg, etwas zu beurteilen.

Glücklicherweise gibt es im Cloud-Native-Umfeld ein festes Merkmal für diese Kennzahlen: den DORA-Report von Google. Die Seite zu DORA findet man hier: *https://cloud.google.com/blog/products/devops-sre/dora-2022-accelerate-state-of-devops-report-now-out*

Es gibt unglaublich viele Facetten und die Messbarkeit von Anwendungsentwicklung. In diesem Moment sind praktisch 4 KPIs führend:

1. **Deployment Frequency** – Häufigkeit der Bereitstellung. Wie oft stellt das Unternehmen Änderungen an der Produktion bereit?
2. **Lead time for Changes** – Vorlaufzeit für Änderungen. Was ist die Vorlaufzeit für Änderungen, wie lange dauert es, bis eine Änderung in die Produktion geht?
3. **Time to restore service** – Zeit, den Dienst wiederherzustellen. Wie lange dauert es im Allgemeinen, einen Dienst wiederherzustellen, wenn ein Vorfall oder Fehler auftritt?
4. **Change failure rate** – Änderung der Ausfallrate. Welcher Prozentsatz von Änderungen führt zu verschlechterten Service-Levels für die Anwendung?

Es gibt in dieser Welt so viel Interessantes zu lernen, womit man die Umstellung von einer eher traditionellen Anwendung

in der Entwicklungspraxis auf eine Microservices-basierte Art und Weise mit einer festen Grundlage beginnen kann. Es gibt keinen Grund mehr, das nicht zu machen.

Kubernetes

Agil, schnell und skalierbar sind Wörter, die in diese neue Welt gehören.

Betriebssystembereitstellung, Rechneraufbau und Netzwerkeinrichtung sind Wörter, die vielleicht nicht direkt frohe Blicke verursachen.

Glücklicherweise ist das auch nicht länger notwendig. Nicht notwendig, obwohl wir über den „Infrastructure-as-a-Code"-Weg vorankommen, weil es noch immer (viel) zu viel Zeit braucht. Sondern, weil wir jetzt über Kubernetes verfügen können oder, wie „GeekWire" schreibt:

„Wir hätten 13 weitere Namen, die wir nicht an der Rechtsabteilung von Google vorbeibringen konnten. Es war der letzte Tag und ich musste etwas auswählen. Ich fuhr zur Arbeit und dachte: ‚Nun, [die Technologie] ist wie das Fahren eines Containerschiffs. Wie würde der Steuermann heißen?' Also versuchte ich, etwas Exotisches zu finden. Ich hatte keine Ahnung, was das griechische Wort dafür war. Ich musste es nachschlagen."

Zwei Kubernetes-Pioniere starteten das Cloud-Tech-Startup Heptio mit einer Investition von 8,5 Millionen US-Dollar.

Zum Namen Heptio erklärte Joe Beda (einer der erfolgreichsten Ingenieure, die Kubernetes mitgestaltet haben): *„Als wir bei der Entwicklung von Kubernetes halfen, stellten wir ihn uns als ‚Seven of Nine' vor, eine ‚Star Trek: Voyager'-Figur, die eine ehemalige Borg-Drohne ist."*

Das war ein Hinweis auf Borg, ein Codename für Googles interne Version von Kubernetes. Das Ding, das seine Suche, Apps und Anzeigen ausführt. Es ist die totale Geek-Kultur.

Wir wollten einen freundlicheren Borg. Aus diesem Namen wurde „Project Seven". Als wir mit Kubernetes an die Öffentlichkeit gingen, wollten wir die „Sieben" nicht aus den Augen verlieren.

Das Kubernetes-Logo hat sieben Seiten. „Hept" ist das griechische Präfix für „Sieben", damit blieb die Sieben im Namen erhalten.

Mittlerweile haben Google und die Linux Foundation die Cloud Native Computing Foundation gegründet, die, genauso wie das Postgres Project, ein von der Community getriebenes oder öffentlich verwaltetes Open-Source-Projekt ist.

Postgres and Kubernetes

Bis jetzt lag der Fokus immer auf der Anwendungssoftware. Quasi Stateless-Programme, die man über Kubernetes skalierbar und automatisch heilend dargestellt hat. Eine unelegante Schwierigkeit ist dabei, dass man Datenmanagement außerhalb dieser Microservices- oder Kubernetes-Umgebung betreiben muss.

Es ist schon irgendwie klar, dass traditionelle Datenmanagement-Systeme wie Oracle, DB2 oder SQL Server sich ganz schwer mit Kubernetes tun. Das Unelegante daran ist, dass eine Kubernetes-basierte Anwendung sich durch eine zentrale Verbindung mit so einer monolithischen Datenbank verbinden muss.

Mit Postgres gibt es jedoch ein Datenmanagementsystem, das in Kubernetes passt, als ob es dafür entwickelt worden ist. Die Entwicklung von Postgres begann 20 Jahre vor der Entwicklung von Kubernetes.

Übrigens ist die Angst vor Datenbanken in Kubernetes die gleiche wie die Angst vor Datenbanken auf virtuellen Maschinen. Eine Diskussion, die wir auch schon vor fast 20 Jahren abgeschlossen haben. Datenbanken in Kubernetes sind kein Thema.

Cloudnative-pg

Veränderung ist konstant, haben wir am Anfang gelesen. Wir machen, was wir tun, nicht, weil es einfach ist, wir machen es, weil es schwierig ist – wenn es einfach wäre, hätten wir es Fußball genannt.

Spaß beiseite, Postgres in Kubernetes soll einfach sein. Ein Anwendungsentwickler sollte nicht auch noch erst Infrastruktur-DBA werden müssen. Dass der Anwendungsentwickler SQL schreiben kann und Datenmanagement versteht, ist schon vielseitig genug!

Das Unternehmen EnterpriseDB hat als dritte Generation von Postgres-Kubernetes-Operatoren den Cloudnative-pg-Operator gebaut. Im April 2022 wurde der Operator als Community-Open-Source-Projekt der Cloud Native Computing Foundation geschenkt und am KubeCon 2022 in Valencia angekündigt. Damit ist das ein Kubernetes-Operator, der grundsätzlich für jeden mit einer Apache-2-Lizenz verfügbar ist. Im Vergleich mit anderen existierenden Postgres-Operatoren: *„Wir haben den gesamten Funktionsumfang durchgespielt, den wir benötigen, und dies ist nicht einmal ein Wettbewerb. In jedem einzelnen Aspekt, den wir untersucht haben, ist Cloudnative-pg viel besser."*

Schließlich ist die nahtlose Integration von Postgres in eine Cloud-Native-Anwendung beim Entwicklungsansatz kein Thema mehr. Der sogenannte „Postgres-DBA-in-a-box"-Ansatz der Cloudnative-pg ermöglicht diesen Schritt. Zusätzlich bringt Postgres als Teil jeder Anwendung eine Vielfalt an Features und Funktionen, um eine reichhaltigere Anwendung zu bauen.

Darum!

Damit haben wir dann auch alle Aspekte gesehen und besprochen.

In der neuen Runde geht es um die Realität der Datenbanken in Kubernetes. Microservices, die eine Mehrzweck-Datenbank als Nucleus-Funktion einsetzen können.

Das alles detailliert in einer Cloud-Native-Ideenwelt. Nicht nächstes Jahr, nicht übernächste Woche, sondern jetzt!

Machen Sie mit auf *cloudnative-pg.io*!

Jan Karremans
jan.karremans@enterprisedb.com

Ein Jahr PostgreSQL – Das Leben danach

Jonas Gassenmeyer, DB Systel

Vor etwas über einem Jahr hatte ich die Möglichkeit, tiefer in die Welt der PostgreSQL-Datenbankentwicklung einzutauchen. Ein treibender Gedanke war: „SQL ist SQL – ich werde mich schon schnell zurechtfinden, SQL bleibt schließlich eine relationale Datenbank". Doch der Teufel steckt bekanntlich im Detail: Die Unterschiede und Herausforderungen, die wirklich auf einen Oracle-Datenbankentwickler warten, waren mir vorher nicht bewusst. Sowohl Oracle als auch PostgreSQL haben schöne Seiten. Es ist durchaus von Vorteil, beide Systeme ein wenig genauer zu kennen. Dieser Artikel gibt meine Erfahrungen und Eindrücke wieder, die ich nach einem Jahr im PostgreSQL-Ökosystem gemacht habe.

Beginnen wir einfach mit einem `create table`-Befehl. In meinem Oracle-Modus lege ich Textspalten mit dem Datentyp `varchar2` an. Wenn ich einen solchen Befehl in PostgreSQL absetze, dann erhalte ich die Fehlermeldung `SQL Error [42704]: ERROR: type "varchar2" does not exist.`

Schon das weist darauf hin, dass man mit den Skripten nicht einfach in „Copy und Paste"-Manier auf eine PostgreSQL-Datenbank wechseln kann. Die korrekte Alternative ist in diesem Fall der Datentyp `text` *(siehe Listing 1)*.

Entgegen der gängigen Praxis in Oracle, dass man die Zeichenketten-Länge möglichst einschränkt, ist das in PostgreSQL eher untypisch *(vgl. Quelle [1])*.
Möchte man anschließend die erstellte Tabelle im Data Dictionary überprüfen, stellt man den nächsten Unterschied fest: Die bewährte Struktur dba|all|user_tables gibt es so nicht. Stattdessen ist das Pendant im sogenannten information_schema-Schema unter dem Namen `pg_class` zu finden. Objektnamen werden außerdem in Kleinbuchstaben hinterlegt *(siehe Listing 2)*.

Wie man sieht, ist die analoge Spalte zu `user_tables.table_name` hier `pg_class.relname`. Mir hat es geholfen, an die relationale Theorie von Edgar F. Codd zu denken, wo ebenfalls von tupels(=row), relations(=table) und Co. die Rede ist. Der Sprachgebrauch ist im Data Dictionary von PostgreSQL wiederzuerkennen.

Als Nächstes soll die Tabelle Demo mithilfe Insert-Befehl befüllt werden. Zunächst werden dazu zwei Zeilen eingefügt *(siehe Listing 3)*.

Anzumerken ist, dass die Spalte col im ersten Befehl mittels leeren Strings und im zweiten explizit mit NULL befüllt wird. In Oracle hat das dieselbe Bedeutung, doch in PostgreSQL ist mit dem leeren String etwas anderes gemeint, sodass eine Abfrage per `where col is null` nur eine Zeile zurückgibt.

Das muss man sich mal auf der Zunge zergehen lassen. Ich möchte nicht wissen, wie viele Applikationen eine explizite NULL-Wert-Behandlung implementiert haben. Um beide Zeilen im `select`-Befehl zu erhalten, könnte man die `where`-Bedingung in `where coalesce(col, '') = ''` umschreiben. Damit ist es allerdings nicht getan. Man muss sich dann auch mit den möglichen Operationen *(z. B. String-Konkatenation – siehe Listing 4)* befassen.

In *Listing 4* mag Ihnen aufgefallen sein, dass die Tabellenangabe `„from dual"` fehlt. Das ist in PostgreSQL valide Syntax – PostgreSQL kennt die Dummy-Tabelle dual nicht. Doch zurück zum eigentlichen Thema: NULL-Werte. Wenn wir in Oracle einen NULL-Wert an eine Zeichenkette hängen, können wir sicher sein, dass die Zeichenkette wieder ausgegeben wird. Das ist in PostgreSQL nicht so. Der gesamte Ausdruck liefert NULL als Ergebnis. Tatsächlich bin ich mehrfach auf kleine Bugs in meinen Abfragen gestoßen, weil ich aus Gewohnheit falsche NULL-Logiken im Kopf hatte. Der Artikel von AWS „Handle empty Strings when migrating from Oracle to PostgreSQL" umfasst 13 Seiten und 3300 Worte. Das zeigt, wie komplex die Unterschiede zwischen beiden Datenbanken allein im Bereich NULL-Werte sind.

Listing 5 verdeutlicht einen weiteren Unterschied (der sich in Oracle 23c in Luft auflösen wird): Der Boolean-Datentyp existiert und kann in PostgreSQL ohne Einschränkungen in SQL verwendet werden.

Bisher ist auf der Demo-Tabelle kein Primärschlüssel definiert. Wenn dieser ergänzt wird und anschließend zwei identische Werte eingefügt werden *(siehe Listing 6)*, kommt es zu einer Fehlermeldung. Im Gegensatz zur generischen Fehlerbeschreibung in Oracle (`SQL Error [1] [23000]: ORA-00001: unique constraint (SYSTEM.xyz) violated`) enthält die Fehlerbeschreibung in PostgreSQL den tatsächlichen Spaltenwert, der zur Verletzung des Constraint geführt hat (im Beispiel ist das „key1"). Gerade bei Bulk-

```
create table logs as

with t as (
select generate_series('2022-07-01 00:00:00'::timestamp
       , '2022-08-19 01:00:00'::timestamp, '1 day') ts


)
, rn as(
select row_number() over(order by random()) r
     , t.ts
 from t
)
, u as (
select generate_series(1, 10) || '_wbesite_url' url


)

select u.url, rn.ts created_at
     , random() * interval '1 minute' request_duration
from rn
join u
on mod(r,10) +1 =
replace(substring(u.url from 1 for 2), '_', '')::bigint
;
```

```
create table logs as (
select * from (
with t as (
select to_date('2022-07-01 00:00:00'
              , 'YYYY-MM-DD HH24:MI:SS') + level ts
  from dual
 connect by level <= 50
)
, rn as(
select row_number() over(order by dbms_random.value) r
     , t.ts
 from t
)
, u as (
select level || '_wbesite_url' url
  from dual
 connect by level <= 50
)

select u.url, rn.ts created_at
     , dbms_random.value * interval '1' minute request_duration
from rn
join u
on mod(r,10) +1 =
replace(substr(u.url, 1, 2), '_'))
)
;
```

Abbildung 1: Erzeugen von Testdaten Oracle- vs. PostgreSQL-Syntax (Quelle: Jonas Gassenmeyer)

Loads macht das die Suche nach der Nadel im Heuhaufen deutlich einfacher. Ein kleiner, aber feiner Unterschied.

Im Kontext von Bulk-Loads und Primärschlüssel-Verletzungen ist in Oracle der `merge`-Befehl hilfreich. Dieser ist seit 9i verfügbar und besitzt somit einen gewissen Reifegrad. In PostgreSQL wurde `merge` erst im Oktober 2022 mit Version 15 eingeführt. Damit ist der Befehl nicht ganz ausgereift und noch etwas limitiert. Genaueres ist dem Syntax-Baum in der offiziellen Dokumentation zu entnehmen *(https://www.postgresql.org/docs/15/sql-merge.html)*. Apropos Dokumentation. Sowohl Oracle als auch PostgreSQL weisen eine sehr gute Doku auf. PostgreSQL verzichtet dabei (absichtlich) auf Code-Beispiele, was manchmal weitere Google-Suchen erforderlich macht. Den Luxus, den Oracle-Base.com bietet, hat man für PostgreSQL nicht. Wer noch nicht PostgreSQL Version 15 verwendet, dem sei die kleine Schwester des `merge`-Befehls empfohlen *(siehe Listing 7)*. Hier lässt sich zumindest eine Update-Aktion (oder „Fehler ignorieren“) bestimmen, wenn eine entsprechende Constraint-Verletzung vorliegt. Nicht ganz so mächtig wie der `merge`-Befehl, doch ich habe häufig darauf zurückgegriffen und fand ihn auch sehr selbsterklärend. Wirklich vermisst habe ich merge nicht.

Man ahnt es schon: An vielen Stellen wird man sich bei PostgreSQL an kleinere Syntax-Änderungen gewöhnen müssen, die mehr oder weniger einschränkende Funktionalität bedeuten. Sollte ich nach einem Jahr Vergleich die aktuell bedeutendste Einschränkung nennen, wären das die Window-Functions. Diese sind zwar auch zahlreich in PostgreSQL vorhanden, allerdings hat mich Oracle in der Vergangenheit mit Details beeindruckt; zum Beispiel wird ein `select count(distinct …) over(partition by …)` mit der Fehlermeldung **„`SQL Error [0A000]: ERROR: DISTINCT is not implemented for window functions`“** zurückgewiesen.

Die Tabelle Demo war bis hierhin ein dankbarer Begleiter zum Aufzeigen der Unterschiede. An dieser Stelle möchte ich ein neues Beispiel einführen, um zu verdeutlichen, dass die Syntax-Differenzen zwischen Oracle und PostgreSQL an vielen Stellen deutlich spürbar sind *(siehe Abbildung 1)*. Man stelle sich folgenden Datensatz vor: Für eine Webanwendung ist in einer Tabelle der jeweilige URL-Aufruf mit Dauer und Zeitstempel des Aufrufs getrackt. Ich habe mit SQL zu Demonstrationszwecken einen solchen Datensatz künstlich erzeugt. Aus Platzgründen erkläre ich die Gegenüberstellung stichpunktartig und nenne die gravierendsten Unterschiede:

- Common Table Expression (with-*Klausel* in Kombination mit *create table*)
- Generierung mehrerer Zeilen (*connect by level* vs. *generate_series*)
- Teilen von Strings (*substr* vs. *Substring*-Funktion; siehe auch Parameter)
- Erzeugen von Zufallszahlen (*dbms* Package vs. *random()*-Funktion)
- Literale beim Erzeugen von Zeitintervallen (*interval '1 minute'* vs. *interval '1' minute*. – Hochkommas beachten!)

Bevor ich mich anderen Aspekten als den Syntax-Unterschieden widme, möchte ich noch erwähnen, dass auch syntaktisch gleiche Konstrukte zu unerwarteten Ergebnissen führen können: Ein `select 5/2` ergibt in PostgreSQL 2 (Zwei). Das ist auch so dokumentiert *(vgl. [2])*, allerdings finde ich den Default *(„for integral types, the devision truncates the result towards zero“)* nach wie vor gewöhnungsbedürftig. Explizite Casts wie `select 5.0/2` oder `select 5::float/2` sind mögliche Lösungen, die dann zum Ergebnis 2,5 führen.

Architekturunterschiede

Im zweiten Teil möchte ich noch etwas ins Detail gehen und ein paar markante architektonische Entscheidungen in PostgreSQL beleuchten. Was eine Datenbank grundlegend ausmacht, ist das MVCC-Prinzip *(Multi Version Concurrency Control)*, das den konkurrierenden Zugriff auf Daten regelt. Wenn eine Transaktion liest, soll sie nicht von einer schreibenden Transaktion geblockt sein und umgekehrt. Hierzu ist es essenziell, dass Datenänderungen versioniert sind. Oracle ändert die Daten „inline“ im Block. Es wird um jeden Preis vermieden, dass die physikalische Adresse einer Zeile im Block nochmal wandert. Das hat zum Beispiel den Vorteil, dass ein Index stets unverändert den Zugriff auf den Block für diese Zeile kennt. Die eigentlichen Änderungen/Versionen sind im UNDO festgehalten. In PostgreSQL wird eine Zeile mehrfach kopiert und eine Transaktions-

```
postgreSQL>cat postgresql.conf | grep vacuum
#autovacuum_work_mem = -1               # min 1MB, or -1 to use maintenance_work_mem
#vacuum_cost_delay = 0                  # 0-100 milliseconds (0 disables)
#vacuum_cost_page_hit = 1               # 0-10000 credits
#vacuum_cost_page_miss = 2              # 0-10000 credits
#vacuum_cost_page_dirty = 20            # 0-10000 credits
#vacuum_cost_limit = 200                # 1-10000 credits
#vacuum_defer_cleanup_age = 0   # number of xacts by which cleanup is delayed
#log_autovacuum_min_duration = -1       # log autovacuum activity;
#autovacuum = on                        # Enable autovacuum subprocess?  'on'
#autovacuum_max_workers = 3             # max number of autovacuum subprocesses
#autovacuum_naptime = 1min              # time between autovacuum runs
#autovacuum_vacuum_threshold = 50       # min number of row updates before
                                        # vacuum
#autovacuum_vacuum_insert_threshold = 1000      # min number of row inserts
                                        # before vacuum; -1 disables insert
                                        # vacuums
#autovacuum_analyze_threshold = 50      # min number of row updates before
#autovacuum_vacuum_scale_factor = 0.2   # fraction of table size before vacuum
#autovacuum_vacuum_insert_scale_factor = 0.2    # fraction of inserts over table
                                        # size before insert vacuum
#autovacuum_analyze_scale_factor = 0.1  # fraction of table size before analyze
#autovacuum_freeze_max_age = 200000000  # maximum XID age before forced vacuum
#autovacuum_multixact_freeze_max_age = 400000000        # maximum multixact age
                                        # before forced vacuum
#autovacuum_vacuum_cost_delay = 2ms     # default vacuum cost delay for
                                        # autovacuum, in milliseconds;
                                        # -1 means use vacuum_cost_delay
#autovacuum_vacuum_cost_limit = -1      # default vacuum cost limit for
                                        # autovacuum, -1 means use
                                        # vacuum_cost_limit
#vacuum_freeze_table_age = 150000000
#vacuum_freeze_min_age = 50000000
#vacuum_failsafe_age = 1600000000
#vacuum_multixact_freeze_table_age = 150000000
#vacuum_multixact_freeze_min_age = 5000000
#vacuum_multixact_failsafe_age = 1600000000
```

Abbildung 2: Mögliche Parameter zum Tunen von Vacuum (Quelle: Jonas Gassenmeyer)

Abbildung 3: Zeitstempel werden stets in UTC gespeichert (Quelle: Jonas Gassenmeyer)

```
create table demo (
  descr text
, col   text
);
```

Listing 1: create table in PostgreSQL

```
select *
from pg_class
where relname = 'demo' ;
```

Listing 2: Data-Dictionary-Abfrage in PostgreSQL

```
insert into demo (descr, col) values ('1 insert', '');
insert into demo (descr, col) values ('2 insert', null);
```

Listing 3: NULL vs. leere Zeichenkette

```
select 'test'||null; --ergibt NULL!
```

Listing 4: Operation mit Zeichenkette und NULL

```
select '' = '' tst; -- liefert TRUE!
```

Listing 5: Booleans

```
alter table demo add primary key (descr);
insert into demo values ('key1', 'i repeat');
insert into demo values ('key1', 'i repeat');

SQL Error [23505]: ERROR: duplicate key value violates SQL
Detail: Key (descr)=(key1) already exists.
```

Listing 6: Primärschlüssel-Verletzung in PostgreSQL

```
insert into demo values ('key2', 'i repeat') on conflict (descr)
do nothing;

insert into demo values ('key2', 'i repeat')
on conflict (descr) --pk or uk
do update set col = 'i''ve been updated';
```

Listing 7: insert on conflict in PostgreSQL

ID steuert, für welche Session die Zeile sichtbar beziehungsweise unsichtbar ist. Je mehr parallele Transaktionen auf demselben Tupel (=Zeile) operieren, desto mehr Kopien derselben Zeile wird es geben. Das sorgt folglich für sogenannten Bloat: Die Tabelle plustert sich auf und verbraucht physikalisch mehr Speicherplatz. Wenn die älteren Versionen nicht mehr benötigt werden, bleiben überflüssige Versionen einer Zeile übrig. Ein Hintergrundprozess muss aktiv werden, um aufzuräumen. Dieser ist unter dem Begriff Vacuum („Staubsauger") bekannt. Ich selbst musste diesen Prozess nie tunen. In OLTP-Systemen mit viel Last sollte man sich jedoch genauere Gedanken machen, wann und wie dieser Hintergrundprozess aktiv werden soll/muss. Hierzu werden in der zentralen Konfigurationsdatei `postgresql.conf` diverse Parameter angeboten *(siehe Abbildung 2)*.

Ein viel gelesener Artikel *[3]*, der die Schwächen von PostgreSQL aufzeigt, behandelt diese „Copy on Write"-Versionierung und es wird angemerkt, dass es ein Limit der zu vergebenen Transaktions-IDs gibt. Bevor ich jemals eine PostgreSQL-Datenbank live und in Farbe verwendet hatte, hat mir der Artikel nicht unbedingt Mut gemacht. Allerdings hat sich das im echten „hands-on"-OLTP-Betrieb nicht bewahrheitet. Laurenz Albe hat eine interessante Gegendarstellung zum Transaktions-ID-Wraparound-Problem verfasst *(vgl. Quelle [4])*. Im Gegenteil – PostgreSQL hat im Bereich Transaktionen mein Herz gewonnen, weil DDL-Befehle (also z. B. *create table*) per Rollback rückgängig gemacht werden können. So macht Skripte-Schreiben wirklich Spaß! Man muss im Fehlerfall nicht aufräumen und komplizierte Workarounds schaffen. Die Objekt-Änderungen werden einfach nicht *commitet*. Das Auto-Commit ist in allen gängigen Clients (DBeaver, psql, pgAdmin,...) Standard. Man muss mittels `begin`-Befehl explizit eine Transaktion aufspannen, um ein implizites Commit zu verhindern. Das wiederum macht das Ausführen von DML, zum Beispiel von einem `delete`-Befehl, erst mal ungewohnt riskant. Wenn man nicht aufpasst, sind die gelöschten Daten wirklich festgeschrieben. Ein Rollback ist nicht mehr möglich.

In meinem Projekt musste ich intensiv mit Zeitstempeln und Zeitzonen hantieren. Vor allem, wenn Client und Server nicht mit der gleichen Zeitzone konfiguriert sind, muss es zwangsläufig zu einer Konvertierung kommen, sodass „hin- und zurückgerechnet" werden kann. Wenn in Oracle ein Zeitstempel im Datentyp `timestamptz` gespeichert wird, gibt es Bytes, die festhalten, in welcher Zeitzone diese Uhrzeitangabe zu verstehen ist. Was Bitgenau gespeichert wird, hängt von den NLS-Settings von Client und Server ab. In PostgreSQL gibt es eine simple Regel: *There is no such thing as a server time zone.* Der Server nimmt den Zeitstempel so an, wie es vom Client angegeben wird (entweder indirekt über `set timezone` in der Session oder direkt am übertragenen String durch die Formatmaske) und konvertiert in jedem Fall in UTC. Ein lesender Prozess findet anschließend immer Bits auf Platte, die den Zeitstempel in **UTC** repräsentieren. Zur Konvertierung nutzt PostgreSQL eine zentrale Datenbank für Zeitzonen – die IANA *(siehe Abbildung 3)*.

Ein wenig anders verhalten sich in PostgreSQL Client und Server übrigens auch beim Austausch des Zeichensatzes (z. B. Unicode). Doch da die NLS-Settings und UTF-8 Encoding in Oracle einen eigenen Artikel wert wären, möchte ich es lediglich erwähnen. Zum Thema Zeiten möchte ich außerdem noch knapp erwähnen: Wer viel mit Zeitstempel-Arithmetik zu kämpfen hat, sollte sich in PostgreSQL auf jeden Fall mit dem `tsrange`-Datentyp vertraut machen *(siehe auch [5])*. Ich würde mir ein Pendant in der Oracle-Datenbank wünschen.

Performance-Troubleshooting

Im letzten Teil widme ich noch ein paar Zeilen einem Thema, das mich im Oracle-Kontext in den Bann gezogen hat. *Performance Troubleshooting* ist mit den Oracle-Bordmitteln ein echtes Zuckerschlecken. Die Art und Weise, wie diese Software instrumentiert ist, ist einzigartig. Dabei gilt vor allem, dass das Logging per Default eingeschaltet ist; vornehmlich beziehe ich mich hier auf das Event Tracing und das AWR=Active Workload Repository, worauf dann auch die ASH=Active Session History basiert. Somit liegen die relevanten Informationen förmlich auf dem Tisch und man ist nur mit der eigentlichen „Detektiv-Arbeit" beschäftigt. Ich weiß nicht, bei wie viel Gigabyte eine Oracle-Installation inzwischen angekommen ist. Zwar ist eine PostgreSQL-Vanilla-Version deutlich schlanker, das macht sich dann aber genau an solchen Features bemerkbar. Der Weg in der PostgreSQL-Welt führt dann über sogenannte Extensions. Diese sind nachträglich zu installieren. Somit ist zumindest eine Vanilla-Installation sorgfältiger zu planen. Ich habe sogar verstärkt wahrgenommen, dass bei Admins und Experten häufig noch der Performance-Nachteil (‚Workloads laufen 2% langsamer') der Gesamtinstallation als Argument aufgeführt wird, weshalb man intensives Logging nicht per Default aktiviert. Hier halte ich es wie Tom Kyte *[6]*. In allen Umgebungen, in denen ich aktiv war, wurden hilfreiche Instrumentations-Mechanismen nicht aktiviert, sodass die Suche bei erstmalig aufgetretenen Performance-Schwierigkeiten knifflig bis unmöglich war. Die Metriken für eine detaillierte Analyse waren schlichtweg nicht vorhanden.

Nach meinem ersten Jahr kann ich folgenden Ansatz empfehlen: Um proaktiv und nachträglich Langläufer im SQL zu identifizieren, sollte die Extension `pg_stat_statements` installiert werden. Es handelt sich dann um eine View, die durch SQL-Abfragen analysierbar ist. Sie hat einige Schwächen, etwa dass statt des echten SQL-Statements eine normalisierte Form gezeigt wird. Literale werden durch Bind-Variablen ersetzt und die Belegung der Binds kann nicht mehr nachvollzogen werden. Das lässt keine Einzelfallbetrachtung zu. Deswegen sollte zusätzlich auch `auto_explain` geladen werden. Damit werden in den Server-Logs mehr Infos rund um ein SQL (inklusive ermitteltem Ausführungsplan) weggeschrieben. Der Schwellenwert, ab welcher Laufzeit ein SQL geloggt wird, lässt sich konfigurieren (`auto_explain.log_min_duration`). Hat man mit der breiten Analyse den Langläufer identifiziert, kann man folglich auf die Server-Logs zurückgreifen, um mehr Infos zu erhalten. Wenn das Problem reproduzierbar ist, kann man in PostgreSQL leider kein echtes Tracing der Session aktivieren. Die Extension `pg_show_plans` erlaubt es, zumindest aus einer zweiten Session heraus, in eine Langläufer-Session reinzuspicken, um den Ausführungsplan der laufenden Abfrage zu sehen. In dem Fall, dass das Problem reproduzierbar ist, gibt es auch kein Sampling, wie man es von der ASH kennt. Ich finde jedoch, dass die Extension `pg_sentinel` vielversprechende ASH-ähnliche Ansätze beinhaltet. Um dann ein Statement wirklich anzupacken und dem Optimizer unter die Arme zu greifen, kann man nur auf Hints zurückgreifen, wenn die Extension `pg_hint_plan` installiert wird. Diese hat auch eine Funktionalität ähnlich zu Baselines.

Baselines werden in Oracle auf Basis der SQL_ID erstellt, die auch im Library Cache abgelegt wird. PostgreSQL hat keine SGA und damit auch keinen Library Cache. Deshalb bringen Bind-Variablen auch nur Performance-Vorteile in der gleichen Session. Session-übergreifend können geparste SQL-Befehle nicht mit einem Softparse wiederverwendet werden. Es gibt die Möglichkeit von Prepared Statements, die das mehrfache Ausführen desselben SQL innerhalb einer Session beschleunigen. In der Red-Stack-Ausgabe 01/2021 hat Hervé Schweitzer einen eigens diesem Thema gewidmeten Artikel geschrieben – sehr empfehlenswert!

Fazit

Der vorliegende Artikel kann nur begrenzt technische Feinheiten beschreiben, die dann für die wirklichen Unterschiede zwischen den beiden Datenbanken sorgen. Es wurden Syntax- und Architektur-Unterschiede betrachtet – das Wort Migration wurde von mir beispielsweise komplett ausgespart. Für mich ist klar, dass ich mich nach einem Jahr nicht für Oracle oder PostgreSQL entscheiden kann/will und für einen Umstieg plädiere! Für mich gilt: „Kenne beide Seiten und bleibe in Übung". Wenn jedoch keine Gründe dagegensprechen, starten Sie neue Projekte mit PostgreSQL und probieren Sie es aus. Ich freue mich auf Ihr Feedback.

Quellen

[1] *https://maximorlov.com/char-varchar-text-postgresql/*
[2] *https://www.postgresql.org/docs/14/functions-math.html*
[3] *https://rbranson.medium.com/10-things-i-hate-about-postgresql-20dbab8c2791*
[4] *https://www.cybertec-postgresql.com/en/transaction-id-wraparound-a-walk-on-the-wild-side/*
[5] *https://www.cybertec-postgresql.com/en/multiranges-in-postgresql-14/*
[6] *https://carymillsap.blogspot.com/2009/02/on-usefulness-of-software.html*

Über den Autor

Jonas könnte den ganzen Tag über relationale Datenbanken reden. Er ist froh, dass er diese Leidenschaft auch zu einem Beruf machen konnte.

Jonas Gassenmeyer
jonas.gassenmeyer@deutschebahn.com

Packages und ihre Initialisierung

Jürgen Sieben, ConDeS

Dieser Artikel basiert auf einer Fundstelle in einem Fachbuch, die mich bei der Lektüre zu Widerspruch gereizt hat. Mir geht es natürlich nicht darum, mit dem Finger auf andere Autoren zu zeigen, sondern dient als Ankerpunkt, um ein Thema zu erläutern, von dem ich hoffe, dass es für viele Leser interessant sein könnte. Die Fundstelle stammt aus meinem eigenen Buch „Oracle PL/SQL – das umfassende Handbuch“ und wirft Fragen zu Packages und ihrer Initialisierung auf.

Die Fundstelle

In meinem PL/SQL-Buch behaupte ich, ein Package könne Initialisierungscode enthalten, der nur beim ersten Öffnen des Packages ausgeführt wird. Das stimmt. Dann jedoch schreibe ich, dass dieser Code ein Problem habe, denn auf Ebene dieses Codes existiere keine Fehlerbehandlung. Wenn der Code fehlerhaft läuft, gilt das Package dennoch als initialisiert. Das stimmt nicht. Woher ich das habe, kann ich gar nicht mehr nachvollziehen, wahrscheinlich habe ich das bei Steven Feuerstein vor Jahren gelesen, sicher bin ich mir aber nicht. Doch ist das ein Beispiel dafür, wie man Wissen, das einem persönlich als gesichert scheint, einfach wiedergibt, ohne es noch einmal zu prüfen. Nun ja, das habe ich dann irgendwann doch gemacht und mich bei der Gelegenheit einmal darum gekümmert, welche Optionen zum Initialisieren eines Packages existieren und welche Themen rechts und links davon hineinspielen.

Das Problem

Ein Package ist das Grundkonstrukt von PL/SQL-Programmen. Es scheint auf den ersten Blick ein einfaches Konstrukt zu sein, denn es bündelt ja nur mehrere Methoden, die thematisch zueinander gehören, und ermöglicht weitergehende Strategien wie die Trennung öffentlicher und privater Methoden sowie die Überladung einer Methode mit mehreren Parametersignaturen.

Weniger im Fokus ist, dass ein Package einen Status besitzen kann. Dies ist immer dann der Fall, wenn Packages auf globaler Ebene (ob privat im Body oder öffentlich in der Spezifikation, spielt keine Rolle) Attribute (Variablen oder Cursor) enthalten. Oft werden diese beim ersten Verwenden des Packages im Arbeitsspeicher mit Werten beladen, damit sie anschließend direkt zur Verfügung stehen. Beispiele könnten sein:

- Der Name eines Mandanten bei mandantenfähiger Software
- Globale Parameter, die hauptsächlich aus PL/SQL genutzt werden
- Kleinere Listen von Referenzwerten

Ein Package, das solche Attribute enthält, nennt Oracle `stateful packages`, sie sind zustandsbehaftet, was nichts anderes bedeutet, als dass sich gleiche Packages (in unterschiedlichen Sessions) durch die Werte ihrer Attribute unterscheiden können. Dieser Zustand muss also separat gespeichert werden.

Um diese Werte zu initialisieren, steht im Package die Möglichkeit offen, in der Implementierung des Packages das Schlüsselwort `begin` zu verwenden und anschließend Code zu schreiben, der diese Aufgaben übernimmt. Doch was passiert, wenn dieser Code nicht fehlerfrei ausgeführt werden kann? In welchem Zustand befindet sich das Package anschließend?

Mein Wissensstand war, dass in diesem Bereich kein `exception`-Block zur Verfügung steht und daher dieser Fehler unbemerkt bliebe. Zudem gelte das Package trotz fehlerhafter Initialisierung als initialisiert und die Funktionalität wäre anschließend letztlich unvorhersehbar. Das stimmt so nicht (und ich habe noch einmal bei Steven Feuerstein nachgelesen und bemerkt, dass ich das Problem vereinfacht in Erinnerung und dadurch falsch wiedergegeben hatte).

Sehen wir uns also das Problem einmal genauer an.

Packages können Fehlerbehandlungen im Initialisierungscode enthalten, behandeln aber nur Fehler, die auch dort ausgelöst werden. Hierzu ein kurzes Beispiel *(siehe Listing 1)*.

Beim ersten Aufruf des Packages wurde der Initialisierungscode aufgerufen und der dort auftretende Fehler korrekt bearbeitet. Steven Feuerstein beschrieb ein Verhalten, das einen Fehler verdeckt, wenn er außerhalb des Initialisierungscodes auftaucht *(siehe Listing 2)*.

Der Fehler wird geworfen, wenn die private, globale Variable `my_internal_var` durch eine zu lange Zeichenkette initialisiert wird. Diese Zuweisung wird allerdings außerhalb des `begin-exception`-Blocks innerhalb des Packages durchgeführt. Früher wurde der Fehler beim ersten Aufruf des Packages geworfen, das Package aber dennoch als korrekt initialisiert gekennzeichnet, der Wert von `my_internal_var` wäre früher NULL gewesen.

Doch hat sich in der Zwischenzeit das Verhalten geändert: Auch hier wird der Fehler mittlerweile korrekt geworfen, kann jedoch immer noch im Initialisierungsteil nicht abgefangen werden (ich kann nicht mehr nachvollziehen, ab welcher Version nicht mehr, habe aber Version 11 im Verdacht, weil hier der Compiler grundsätzlich überarbeitet wurde). Das Package gilt als nicht instanziiert, der Fehler wird bei jedem Aufruf erneut geworfen.

```
SQL> set serveroutput on

SQL> create or replace package test_pkg
  2  as
  3    my_global_var number;
  4
  5  end test_pkg;
  6  /

Package wurde erstellt.

SQL> create or replace package body test_pkg
  2  as
  3  begin
  4    dbms_output.put_line('Initialisierungscode berechnet');
  5    select 1
  6      into my_global_var
  7      from user_objects
  8     where object_name = 'FOO'
  9       and rownum = 1;
 10  exception
 11    when no_data_found then
 12      my_global_var := 0;
 13  end test_pkg;
 14  /

Package Body wurde erstellt.

SQL> begin
  2    dbms_output.put_line('Ergebnis: ' || test_pkg.my_global_var);
  3  end;
  4  /
Initialisierungscode berechnet
Ergebnis: 0

PL/SQL-Prozedur erfolgreich abgeschlossen.

SQL> begin
  2    dbms_output.put_line('Ergebnis: ' || test_pkg.my_global_var);
  3  end;
  4  /
Ergebnis: 0

PL/SQL-Prozedur erfolgreich abgeschlossen.
```

Listing 1: Packages, die Fehlerbehandlungen im Initialisierungscode enthalten, aber nur Fehler, die auch dort ausgelöst werden, behandeln

Lösungsansätze

Eine Möglichkeit, Initialisierungsfehler außerhalb des **begin**-Blocks abzufangen, besteht in der Einführung einer eigenen Initialisierungsmethode, in der alle Zuweisungen durchgeführt werden *(siehe Listing 3)*.

Nun können alle Fehler während der Initialisierungsphase gefangen und bearbeitet werden. Zudem hat die Initialisierungsmethode noch den Vorteil, bei Bedarf veröffentlicht werden zu können. Auf diese Weise könnte das Package auch später noch re-initialisiert werden. Nun, wo wir das Problem grundsätzlich im Griff haben, tauchen weitergehende Fragen auf: Warum machen wir das Ganze eigentlich?

Wir möchten den Package-Zustand initial festlegen und später ändern können

Eine Anwendung könnte sein, dass ein Package in einer globalen Variable festlegt, ob die Verarbeitung im Package protokolliert werden soll oder nicht. Initial könnten wir festlegen, dass keine Protokollierung erfolgt, später könnte der Status jedoch umgestellt werden. Es könnte sein, dass eine entsprechende Variable hierfür im Package verwendet wird.

Problematisch an diesem Ansatz ist, dass ein Package *pro Session* initialisiert wird. Sind also mehrere Sessions für einen Benutzer geöffnet, sind auch mehrere, voneinander unabhängige Package-„Instanzen“ verfügbar. Das meint, dass ein initialisiertes Package mit entsprechenden Variablenwerten pro Session in dessen UGA vorhanden ist: Jede Session verfügt über eine unabhängige Kopie des Packages. Das führt dazu, dass eine Änderung der Packagevariablen immer nur in der jeweiligen Session zu sehen ist. Eine Session kann den Status von Packages in anderen Sessions nicht ohne Weiteres einsehen oder gar ändern. Das mag gewollt sein oder auch nicht. Wenn Sie möchten, dass der Status des Packages in allen Sessions zentral geändert werden können soll, funktioniert dieser einfache Ansatz nicht, weil die Änderung einer Variable im lokalen Package von den anderen Package-Instanzen nicht gesehen wird.

```
SQL> create or replace package body test_pkg
  2  as
  3    char_to_small exception;
  4    pragma exception_init(char_to_small, -6502);
  5    my_internal_var char(1) := 'FOO';
  6  begin
  7    dbms_output.put_line('Initialisierungscode berechnet');
  8    select 1
  9      into my_global_var
 10      from user_objects
 11     where object_name = 'FOO'
 12       and rownum = 1;
 13  exception
 14    when no_data_found then
 15      my_global_var := 0;
 16    when char_to_small then
 17      dbms_output.put_line('Variable zu klein.');
 18  end test_pkg;
 19  /

Package Body wurde erstellt.

SQL> begin
  2    dbms_output.put_line('Ergebnis: ' || test_pkg.my_global_var);
  3  end;
  4  /
begin
*
FEHLER in Zeile 1:
ORA-06502: PL/SQL: numerischer oder Wertefehler: Zeichenfolgenpuffer zu klein
ORA-06512: in "DOAG.TEST_PKG", Zeile 5
ORA-06512: in Zeile 2
```

Listing 2: Verhalten, das einen Fehler verdeckt, wenn er außerhalb des Initialisierungscodes auftaucht

Eine eng verbundene Spielart dieses Problems besteht, wenn Sie zum Beispiel durch ein globales Attribut die Anzahl der Aufrufe von Package-Methoden zählen möchten. Auch ein solcher Zähler wird Session-bezogen zählen und eventuell nicht alle relevanten Aufrufe sehen können. Zudem ist man manchmal doch sehr überrascht, wie viele oder wenige Aufrufe einer Methode gezählt werden, wenn der Aufruf durch SQL erfolgt. Einfache Denkmodelle à la „eine `select`-Anweisung entspricht einem Aufruf" oder ähnlich werden da nicht zum Ziel führen. `update`-Anweisungen können Methoden mehrfach aufrufen, `select`-Anweisung entweder für jede Zeile einmal oder auch nur wenige Male für eine Anweisung. Wird eine Prozedur aus einem Job heraus aufgerufen, läuft sie in einem eigenen Session-Kontext und verfälscht dadurch die Zählung etc.

Zur Lösung vieler dieser Probleme verlagern wir den Zustand des Packages in einen Kontext, der als „globaler Kontext" eingerichtet wurde. Unter einem Kontext verstehen wir bei Oracle eine Datenstruktur, in der auf leichtgewichtige Weise (die Struktur wird nur im Arbeitsspeicher verwaltet und kann ohne Umgebungswechsel aus SQL und PL/SQL angesprochen werden) boolesche Werte und kurze Zeichenketten unter einem Namen abgelegt werden können.

Sie kennen den von Oracle mitgelieferten Kontext USERENV, in dem sich Angaben zum angemeldeten Benutzer befinden und auf den mit der Funktion `sys_context` zugegriffen werden kann. Dieser Kontext ist Session-bezogen, speichert seine Daten also pro Session in der UGA. Ein Kontext kann aber auch global angelegt und durch ein Package unserer Wahl beschrieben werden. Wenn wir den Zustand unserer Variablen dort ablegen und auch stets nur von dort lesen, ist der Zustand des Packages in allen Sessions stets gleich, denn der Zustand wird nicht im Package, sondern im Kontext gespeichert.

Ein Beispiel führt hier ein wenig zu weit, sehen Sie sich Beispiele für globale Sessions unter dem Suchbegriff `context accessed globally` gern im Internet an. Ein besonders gutes Beispiel finden Sie unter *http://www.oracle-developer.net/display.php?id=424* . Die Strategie besteht hier also darin, dass wir lokale Package-Variablen zugunsten einer Session-übergreifenden Speicherstruktur in der SGA aufgeben: Der Zustand wird, außerhalb des Packages, im Kontext in der SGA gespeichert.

SERIALLY_REUSABLE Packages

Packages mit globalen Variablen oder Cursor werden, wie bereits gesagt, von Oracle `stateful packages` genannt: Weil ein Package pro Session unterschiedliche Werte enthält, haben sie einen „Status" und dieser wird in der *UGA* (User Global Area, ein zentraler Speicherbereich auf dem Datenbankserver, der für die Speicherung Session-bezogener Daten genutzt wird) gespeichert. Das führt dazu, dass viele Benutzer den jeweiligen Packages-Status in der eigenen UGA speichern. Da die Packages diesen Status zudem halten, bis die Session beendet wird, kann dieses Verhalten zu einer Belastung des Datenbankservers führen, denn der Zustand vieler instanziierter Packages muss gespeichert werden. Hieraus resultiert zunächst einmal die Empfehlung, die Speichergröße dieser globalen Variablen

```
SQL> create or replace package body test_pkg
  2  as
  3    my_internal_var char(1);
  4
  5    procedure initialize
  6    as
  7      char_to_small exception;
  8      pragma exception_init(char_to_small, -6502);
  9    begin
 10      dbms_output.put_line('Initialisierungscode berechnet');
 11      my_internal_var := 'FOO';
 12      select 1
 13        into my_global_var
 14        from user_objects
 15       where object_name = my_internal_var
 16         and rownum = 1;
 17    exception
 18      when no_data_found then
 19        my_global_var := 0;
 20      when char_to_small then
 21        dbms_output.put_line('Variable zu klein.');
 22    end initialize;
 23  begin
 24    initialize;
 25  end test_pkg;
 26  /

Package Body wurde erstellt.

SQL>
SQL> begin
  2    dbms_output.put_line('Ergebnis: ' || test_pkg.my_global_var);
  3  end;
  4  /
Initialisierungscode berechnet
Variable zu klein.
Ergebnis:

PL/SQL-Prozedur erfolgreich abgeschlossen.
```

Listing 3: Einführung einer eigenen Initialisierungsmethode, in der alle Zuweisungen durchgeführt werden

und Cursor im Hinterkopf zu haben und nicht exzessiv zu vergrößern.

Für eine spezielle Anwendungsform solcher Packages existiert seit Version 10g der Datenbank das Pragma `serially_reusable`. Wenn man weiß, worum es geht, erscheint einem der Name als Programm, wenn nicht – nicht ;-). Worum geht es also? Stellen wir uns vor, ein Package müsste globale Variablen stets nur im Kontext eines „Server-Calls“, wie Oracle das nennt, vorhalten. Ein Server-Call meint einen PL/SQL-Block, der zusammenhängend ausgeführt wird. Er kann weitere Package-Aufrufe enthalten, beginnt aber beim ersten `begin` und endet irgendwann beim entsprechenden end.

Stellen wir uns nun vor, im Kontext dieses Server-Calls würden alle relevanten Package-Attribute auf Werte eingestellt, die in diesem Server-Call benötigt werden. Nach dem Call könnten die Werte verworfen werden, denn der nächste Server-Call setzt eigene Werte, die er für seine Arbeit benötigt. In einem solchen Szenario könnte die Option `serially_reusable` helfen, die Speicherlast zu reduzieren. Denn: In einem herkömmlichen Package würde die Speicherung von Daten in einer globalen Variablen eines Packages dazu führen, dass dieser Wert für die Dauer der Session in der UGA verbleibt, obwohl ein Folgeaufruf des Packages in der gleichen Session diese gespeicherten Werte nicht verwendet, sondern durch neue ersetzt. Das Problem ist also, dass Werte in der UGA vorgehalten werden, obwohl sie durch den Folgeaufruf gar nicht mehr benötigt werden. Eigentlich könnte in diesem Szenario auf die Speicherung dieser Werte verzichtet werden, was die UGA entlasten würde.

In einem solchen Einsatzszenario ist das Package nacheinander, das heißt durch den Folgeaufruf, wiederverwendbar: `serially_reusable`. Dass dies so ist, kann der Compiler nicht erkennen, denn es ergibt sich aus der Nutzung des Packages. Wenn Sie als Entwickler dieses Verhalten jedoch zusagen können und dies dem Package durch das Pragma (also eine Nachricht an den Compiler) `serially_reusable` mitteilen, wird die Datenbank nun Folgendes tun: Der initiale Wert der Package-Variablen wandert in die SGA und dient sozusagen als Seed (Startwert) für den jeweiligen Aufruf durch die einzelnen Packages. Möchten Sie Änderungen an den Startwerten aus der SGA vornehmen, können Sie dies tun; diese Änderungen überleben allerdings den Server-Call nicht, werden also mit Abschluss des äußersten PL/SQL-Blocks verworfen. In der SGA verbleiben nur die

initialen Startwerte, in der UGA wird anschließend nichts gespeichert. Das reduziert die Speicherlast der UGA, hat aber den Nachteil, dass Ihr Code, wenn er dennoch auf den letzten geänderten Wert aus einem früheren Aufruf in der gleichen Session vertraut, nun nicht mehr (korrekt) funktionieren wird.

Finden Sie diese Option interessant, sollten Sie die Einschränkungen kennen, die festlegen, dass solche Packages nicht aus einem Trigger oder aus SQL heraus aufgerufen werden dürfen.

Vermeidung des Problems durch stateless Packages

Ist Ihnen das Ganze zu fummelig und verwickelt, kann man das Problem da und dort auch dadurch lösen, dass überhaupt keine Attribute in Packages verwendet werden. Ein Package ohne private oder öffentliche Variablen und Cursor wird als stateless angesehen und speichert daher auch keine Werte in der UGA. Konstanten, die sich definitionsgemäß in unterschiedlichen Sessions nicht unterscheiden können, stellen ebenfalls keine Attribute in diesem Sinne dar.

Packages können durchaus ohne globale Attribute erstellt werden, im Regelfall wird dies jedoch zur Folge haben, dass Methoden mehr Parameter benötigen als ohne globale Attribute, denn hierfür werden diese ja häufig gebraucht: Um global gültige Informationen nicht von Methode zu Methode durch Parameter weitergeben zu müssen und andererseits nicht ständig die gleichen Initialisierungsroutinen für solche Parameter aufrufen zu müssen. Eine Abwägung also, die mit Bedacht durchgeführt werden sollte. Ich verwende globale Attribute in meinen Packages nur, wenn sie mir einen relevanten Vorteil verschaffen, aber so sparsam wie irgend möglich. Je nach Aufgabenstellung kann es aber durchaus sein, dass eine solche Selbstbeschränkung nicht aufrechterhalten werden kann, dann muss man halt davon abweichen und hat mit den oben beschriebenen Möglichkeiten ein wenig Handlungsspielraum. Wichtiger ist jedoch, sich grundsätzlich klarzumachen, dass das Problem der Initialisierung von Packages existiert, und die Auswirkungen auf den eigenen Code zu kontrollieren.

Zusammenfassung

PL/SQL-Packages sind die Grundbausteine für die Erstellung eines Programms innerhalb der Datenbank und haben große Vorteile in der Verwendung. Überladung, Kapselung von Logik durch private Methoden, Performanz-Optimierung durch einmalige Kompilierung ganzer Programmteile sind nur einige Beispiele hierfür. Die Verwaltung global zugänglicher Attribute von Packages schafft allerdings eine neue Ebene der Komplexität, weil vormals zustandslose Packages nun einen Zustand erhalten. Erhält ein Package einen Zustand, sind Initialisierungen dieses Zustandes erforderlich, die durch Code im Packagekörper oder – besser – durch Initialisierungsmethoden durchgeführt werden.

Dieser Zustand ist Session-bezogen, was manchmal stören mag, und hat einen erhöhten Speicherverbrauch zur Folge, den man manchmal reduzieren kann. Techniken hierfür sind global zugängliche Kontexte und das Pragma `serially_reusable`. Sie stellen keine Basistechnik dar, sondern sind schon etwas abgefahrener, können aber spezielle Aufgabenstellungen hervorragend lösen.

Jürgen Sieben
j.sieben@condes.de

Oracle Datenbanken Monthly News

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

DOAG Online

Es ist wieder so weit: 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-22/

Ansible, AWX und Co. – das neue Schweizer Taschenmesser für den Oracle DBA

Jérôme Witt, dbi services

From zero to hero, unter diesem Motto lässt sich wohl zusammenfassen, wie sich die Cloud in letzter Zeit fast schon viral in der IT-Welt durchgesetzt hat. Als wären die Cloud und alle technischen Vorträge rund um Cloud der heilige Gral für jeden IT-Mitarbeiter, der etwas auf sich hält. Ja, Cloud ist zu einem Standard geworden, der jeden Sektor der IT-Branche prägt. Insbesondere im (Oracle-)Datenbankbereich ergeben sich durch DevOps Werkzeuge, die den Weg in die Cloud vereinfachen oder einfach nur bei Automatisierungsprozessen unterstützen. Dieser Artikel stellt einige dieser Werkzeuge und Konzepte zur modernen, agilen Weggestaltung anhand von Beispielen vor.

#Containers & GitOps Pipelines

Um den Einstieg in die Containerisierung möglichst einfach zu gestalten, bietet Oracle einerseits vorgefertigte Docker Images über die Oracle Container Registry *[1]* oder Docker Files via GitHub *[2]*, die es ermöglichen, einen Container nachzubauen. In beiden Fällen sollte die Oracle-Lizenzierung bereits im Voraus geklärt werden.

Dennoch ist es sinnvoll für einen DBA, sich mit der Container-Technologie tiefergehend auseinanderzusetzen. Gibt es auch in ihrem Unternehmen noch den einen oder anderen Entwickler, der Java-Programme manuell in die Datenbank lädt? Wäre es nicht ideal, ihm dies automatisiert auf einer Testumgebung in einer GitOps-(Integration-)Pipeline zu ermöglichen? Ebenso könnte der DBA mit einer solchen Pipeline die Verteilung einer neuen Programmversion (aka Artifact) in einer Testumgebung verknüpfen (delivery). Eine vereinfachte Version einer solchen Pipeline ist in *Abbildung 1* dargestellt.

Dieses Beispiel eignet sich besonders gut für diesen Artikel, da Oracle weder ein fertiges Image noch ein Docker File mit dem Datenbank-Client-Programm Loadjava liefert.

```
% docker images
REPOSITORY           TAG    IMAGE ID       CREATED      SIZE
ol8_ora_client_19c   jew    1f7a7fe80e61   1 week ago   4.98GB
```

Listing 1: Docker Images Listing

```
% docker images
REPOSITORY           TAG    IMAGE ID       CREATED      SIZE
ol8_ora_client_19c   jew    1f7a7fe80e61   1 week ago   2.3GB
```

Listing 2: Docker Images Listing (multi-layered Build)

```
% curl -s -k -X POST \
    -H "Authorization: Bearer $AWX_ORA_DEVOPS_TOKEN" \
    -H "Content-Type: application/json" \
       http://awx.jew.lab:8080/api/v2/job_templates/11/launch
```

Listing 3: Ansible AWX RestAPI Call

```
"" Generic options (SHELL programming)
syntax on
set tabstop=3
set expandtab

"" Ansible, python
autocmd FileType yaml,python setlocal number autoindent tabstop=2
cursorline cursorcolumn

"" HCL Terraform
"" I do not like the vim hcl syntax highlight
autocmd BufNewFile,BufRead *.tf setlocal number autoindent tabstop=2
cursorline cursorcolumn syntax=yaml
```

Listing 4: ~/.vimrc

Docker Images bauen

In diesem Paragrafen ist Docker-Grund-Know-how vorausgesetzt. Folgende Dokumente sind sowohl für den Einstieg als auch zur Auffrischung geeignet: Docker File Best-Practices *[3]* und Referenz *[4]*. Grundsätzlich ändert sich bei der Installation einer Oracle-Software in einem Container nicht viel. Dennoch gibt es auch hier einige nützliche Tipps:

- Tipp #1: Docker Image Cache für Debugzwecke nutzen
  Um zum Oracle Universal Installer Logfile zu kommen. Bei einem Abbruch wäre es möglich, durch den Docker Image Cache auf einen Schritt zurückzugreifen.

Aber wir müssten nochmals den runInstaller-Befehl anstoßen und Schritt für Schritt vorgehen (zeitaufwendig). In *Abbildung 2* sehen wir, wo wir anhand des Docker Cache auf den Build-Schritt 9 zugreifen können (image cache id: 483b2488b240).

- Tipp #2: runInstaller-Fehler abfangen
  `RUN .../runInstaller -ignoreSysPrereqs && /bin/true`

Keine schöne Lösung, aber wirksam!

In der Containerwelt ist in der Regel kein SWAP-Space vorhanden. Container lassen sich gut mit Bienen vergleichen. Sie sind selten Einzelgänger, leben hingegen meist in Völkern, die sich alle Ressourcen teilen. Daher könnte es das ganze Volk beeinträchtigen, wenn Container anfangen zu swappen. Oracle-Produkte können wir mit Biene Maja vergleichen, da generell diese eine Biene immer SWAP-Space verlangt, obwohl wir alle wissen, dass es nicht gut für den Rest des Volkes ist.

Ich nehme an, dass jeder, der diesen Artikel studiert, bereits mit Oracle-Technologie vertraut ist und die Installationsvoraussetzungen bekannt sind. Die Option „-ignoreSysPrereqs“ unterbricht den Installationsprozess nicht, jedoch bringt der Oracle Universal Install einen Return Code, der ungleich null ist. Daher wird am Oracle-Universal-Installer(runInstaller)-Befehl ein weiterer Betriebssystem-Befehl angehängt für den Fall „/bin/true“, der relevant für Ihren Build ist.

Das Oracle GitHub Repository für Docker Files *[2]* liefert einen guten Spickzettel.

Last but not least, das Buch „Oracle on Docker: Running Oracle Databases in Linux Containers“ von Sean Scott *[5]* ist wirklich empfehlenswert und vertieft alle Aspekte, die oben erwähnt wurden.

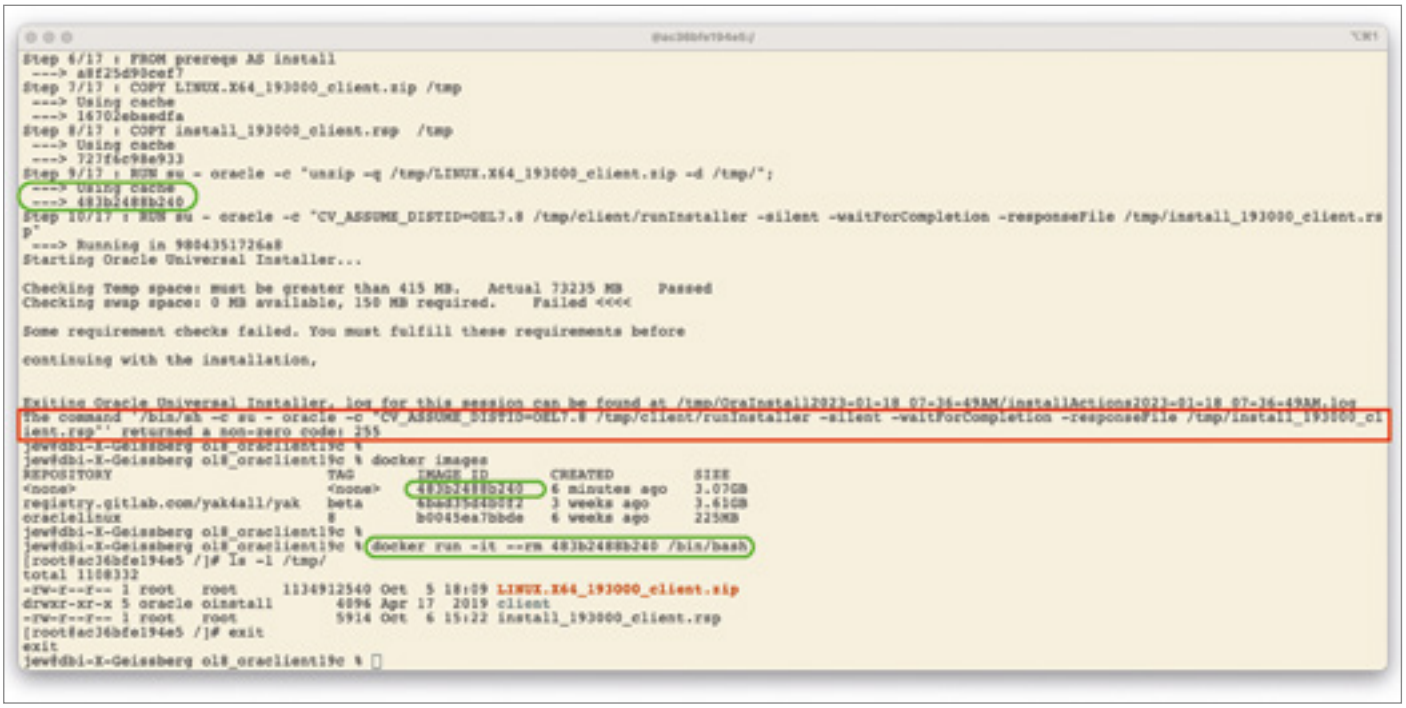


Abbildung 2: Docker-Image-Build-Auszug (Quelle: Jérôme Witt)

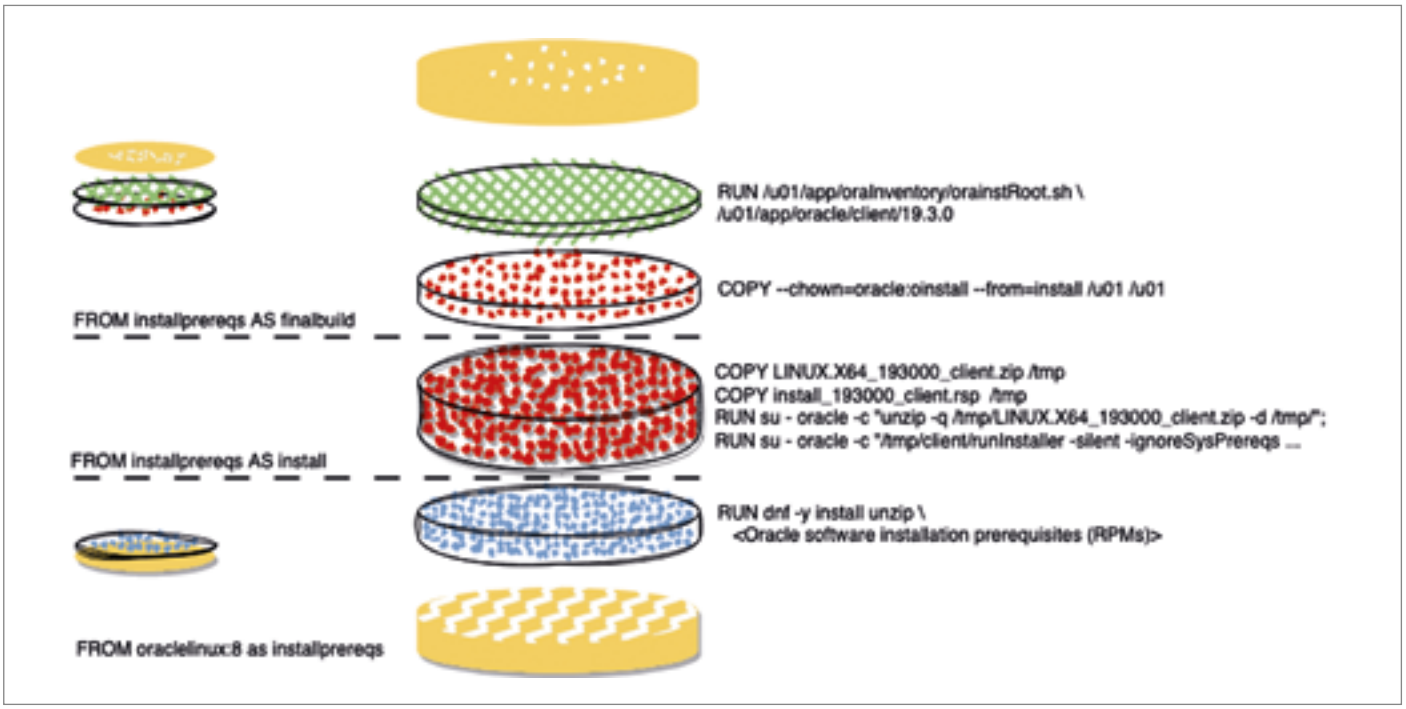


Abbildung 3: Docker multi-stage build (Quelle: Jérôme Witt)

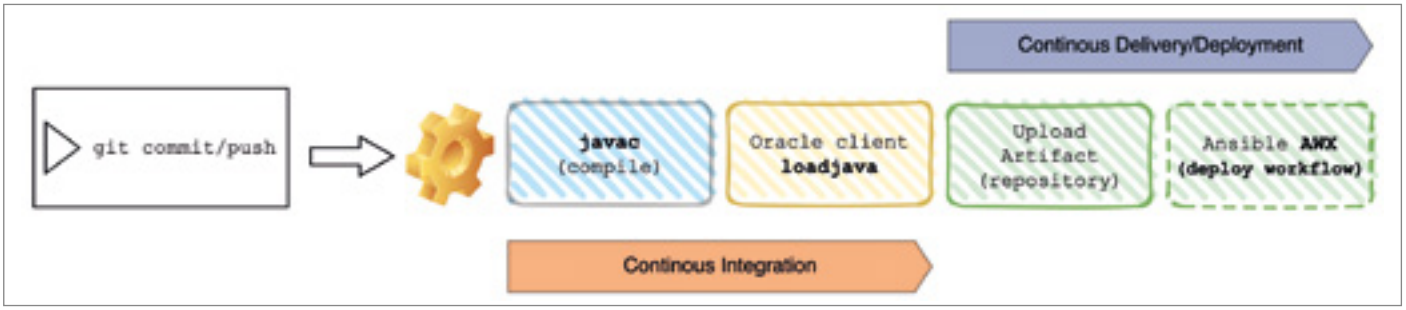


Abbildung 1: CI/CD-Pipeline (Quelle: Jérôme Witt)

```
vi                                                    ⌥⌘1
---
- name: Deploy Best java app ever onto database server
  hosts: ec2-eu1dborat01.jew.lab
  gather_facts: false
  tasks:

    - name: Create destination folder if required
      become: true
      ansible.builtin.file:
        path: "/u01/app/oracle/admin/CDB01/scripts/PDB1"
        state: directory
        owner: oracle
        group: oinstall
        mode: '0750'

    - name: Download artifact from gcloud storage
      become: true
      google.cloud.gcp_storage_object:
        project: "jew-lab"
        bucket: "artifacts_sfw"
        action: download
        src: "best_java_app_ever/HelloWorld.class"
        dest: "/u01/app/oracle/admin/CDB01/scripts/PDB1/HelloWorld.class"

    - name: Validate permissions
      become: true
      ansible.builtin.file:
        path: "/u01/app/oracle/admin/CDB01/scripts/PDB1/HelloWorld.class"
        owner: oracle
        group: oinstall
        mode: '0644'
...
~
"deploy_best_java_app_ever.yml" 32L, 914B written
```

Abbildung 4: Ansible Playbook (Quelle: Jérôme Witt)

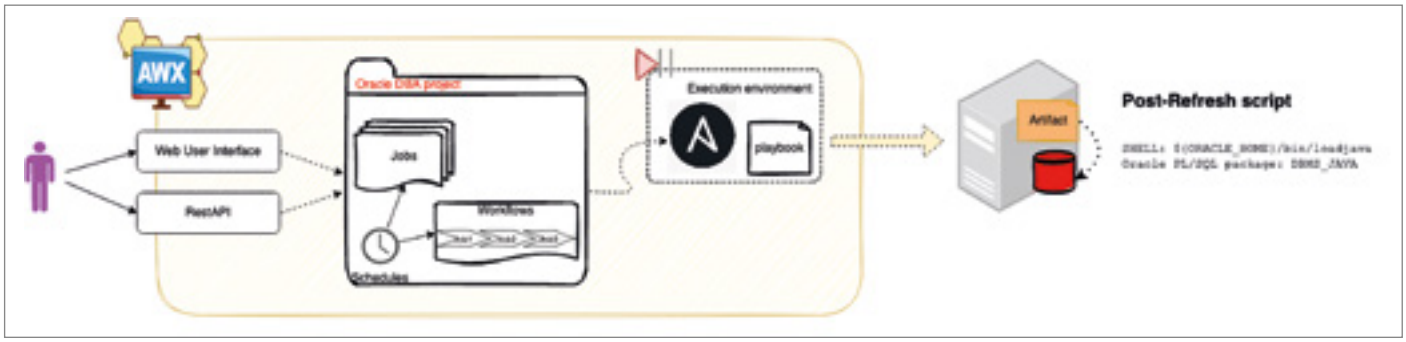


Abbildung 5: Ansible-AWX-Prozessablauf (Quelle: Jérôme Witt)

- Tipp #3: Schlanke Installationskomponente

Der Anwendungsfall, den wir in diesem Artikel beschreiben, benötigt eigentlich nur wenige Oracle-Client-Komponenten. Jedoch wird das Docker Image durch den Installationsprozess stark aufgeblasen *(siehe Listing 1)*.

Eine Möglichkeit, den Oracle Client schlanker zu gestalten, ist beispielsweise, ein eigenes GoldImage zu erstellen. Um den Rahmen dieses Artikels nicht zu sprengen, gehen wir an dieser Stelle nicht weiter auf diese Möglichkeit ein.

- Tipp #4: Multi-layered builds

Eine der größten Challenges beim Bau eines Docker Image ist der Platzverbrauch. Jeder Befehl RUN, COPY, ADD erzeugt eine neue Schicht. Hierbei gilt es zu beachten, dass diese so schlank wie möglich bleibt. Beim Oracle Client handelt es sich lediglich um die Zip-Datei, die zur Installation benötigt wird *(siehe Abbildung 3)*.

Eine detaillierte Anleitung zu multi-layered Builds kann in der Docker-Build-Dokumentation *[6]* gefunden werden. Dank dieser Funktionalität könnte unser Image ohne Weiteres von 4.9 GB auf 2.3 GB verkleinert werden *(siehe Listing 2)*.

Eine weitere Möglichkeit, um noch mehr Platz zu sparen, besteht darin, die Oracle-Softwareinstallationsvoraussetzungen nochmals gründlich durchzugehen. Zum Beispiel sind sysstat, smartmontools, die für Performance-Überwachung benötigt werden, sicherlich irrelevant. Zusätzlich

hierzu können einige Dateien aus dem OracleHome gelöscht werden. Siehe Oracle-GitHub-Skript „installDBBinaries.sh“ *[7]* für die Oracle 19c Installation.

- Tipp #5: CICD Pipeline infrastructure <> local Docker Engine

Eine Docker Engine auf dem Laptop zu installieren, ist der praktische Weg, den viele Entwickler (und auch manche DBAs) nehmen, um erste Entwürfe zu testen. Allerdings kann es durchaus sein, dass das, was lokal erfolgreich gebaut wurde und funktioniert, in der GitOps-Pipeline-Infrastruktur nicht funktioniert. Ein Docker Build passiert komplett im Arbeitsspeicher, was für Oracle-Produkte einiges an Ressourcen benötigt, im Gegensatz zur üblichen „schmalen“ containerisierten Anwendung.

Und wenn die GitOps-Pipeline-Infrastruktur auf Cloud-Computing-Ressourcen läuft, die nur über 2 vCPUs und 512 MBytes Arbeitsspeicher verfügt, schlägt der gute alter Linux Out-Of-Memory Killer zu. Wie soll das 2.4 GB Docker Image, das wir bauen, ins RAM passen? ☺

#Infrastructure-as-Code Ansible

In diesem Paragrafen ist Ansible-Architektur-Grundwissen von Vorteil. Ansible ist im Datenbankumfeld leicht einzusetzen, da es außer einem SSH-Zugang nichts zusätzlich benötigt.

Der in *Abbildung 1* gezeigte Anwendungsfall entlastet den DBA, indem das neue Programm (Artefakt) automatisch auf die Testumgebung verteilt wird. Somit wird sichergestellt, dass die Integration des neuen Programms im Rahmen eines Post-Refresh-Skripts automatisch in die Datenbank einfließt.

Die Programmierung eines solches Ansible Playbook (Skript) lässt sich durch das YAML-Format leicht lesen und warten. Anbei ein Auszug aus dem Playbook, das in der *Abbildung 4* zum Einsatz kommt.

Beim „Programmieren“ muss eigentlich nur auf die Einrückung geachtet werden. Betriebsaufgaben übernehmen die Ansiblemodule inklusive aller benötigten Verifizierungen, zum Beispiel ansible.builtin.file-Module – Attribute, um Dateiberechtigungen entsprechend zu setzen. Zudem sind die Ansiblemodule in der Regel idempotent, das heißt, wenn etwas konfiguriert wird, dann nur, was dem Zustand auf der Zielmaschine (Ansible managed node), nicht dem Inhalt der YAML-Datei entspricht. Dies zugunsten des Ansible-Endbenutzers, da er überhaupt keine Konditionen und Kontrollen in dem „Skript“ einbauen muss (Wartbarkeit).

Orchestration: Red Hat Automation Platform (upstream Ansible AWX)

Ansible AWX ist das Open-Source-Upstream-Projekt *[8]*, das Red Hat unter dem Namen „Ansible Automation Platform“ vertreibt (ehemalig: Red Hat Ansible Tower). Die Plattform bietet die Möglichkeit, Ansible-Playbook-Ausführungen zu orchestrieren und vieles mehr. Im Bezug auf die obige GitOps-Pipeline (*Abbildung 1*) bekommt der Entwickler (oder die Entwickler-Gruppe) gezielt die entsprechenden Ausführungsprivilegien, um diesen einzelnen Job entweder über RestAPI oder direkt im Web GUI anzustoßen, wie in *Abbildung 5 [5]* dargestellt.

Ansible AWX Role Based Access Management ist ziemlich einfach und erlaubt es, gezielt auf bestimmten Ressourcen Ausführungsprivilegien zu setzen. Dies, ohne sicherheitsrelevante Informationen wie beispielsweise den Server-Namen preiszugeben.

Mehr über Ansible-AWX-Sicherheitseinstellungen kann in der Ansible-AWX-Dokumentation *[9]* gefunden werden.

In unserem Beispiel wird der Ansible-AWX-Job durch einen Token über RestAPI Call angestoßen, wie in *Listing 3* beschrieben. Logfiles und Audit stehen per Default für 365 Tage zur Verfügung, was Troubleshooting, Post-mortem- und Auditanalyse erleichtert.

Fazit

Wir leben in einer roten Welt, die mit dem Buchstaben „O“ beginnt. Ein großer Teil unserer DBA-Arbeit dreht sich rund um diese roten Technologien, jedoch sollten wir (DBAs) uns nicht nur in dieser geschlossenen Datenbank-Welt bewegen, sondern uns auch der Außenwelt öffnen. DevOps-Ansätze, wie in dem Artikel oberflächlich beschrieben, sind mittlerweile ein adoptierter Maßstab bei vielen (anderen) Datenbankumgebungen geworden. Das Ziel dieses Artikels war es, Interesse und Neugier für diese offene Welt zu wecken: Scheinbar bin ich daran nicht gescheitert, denn Sie haben sich ihn bereits praktisch bis zum Ende durchgelesen.

Bonus: Muss es zwingend ein YAML-GUI-Editor sein?

NEIN. Der gute, alte Vim-Text-Editor eignet sich hervorragend, um YAML zu „programmieren“. Die *Abbildung 4* entstand durch eine einfache Vim-Konfigurationsdatei (~/.vimrc) *(siehe Listing 4)*.
Have Fun!

Quellen

[1] *https://container-registry.oracle.com*
[2] *https://github.com/oracle/docker-images*
[3] *https://docs.docker.com/develop/develop-images/dockerfile_best-practices/*
[4] *https://docs.docker.com/engine/reference/builder/*
[5] *https://www.oreilly.com/library/view/oracle-on-docker/9781484290330/*
[6] *https://docs.docker.com/build/building/multi-stage/*
[7] *https://github.com/oracle/docker-images/blob/main/OracleDatabase/SingleInstance/dockerfiles/19.3.0/installDBBinaries.sh*
[8] *https://github.com/ansible/awx*
[9] *https://docs.ansible.com/ansible-tower/latest/html/userguide/security.html#rbac-ug*

Über den Autor

#besharing #bepassionate #besuccessful #beresponsible Das sind die Werte, die mich und das Unternehmen, für das ich leidenschaftlich seit über einem Jahrzehnt arbeite, beschreiben.

Jérôme Witt
jerome.witt@dbi-services.com

BERLINER EXPERTENSEMINARE

Die Berliner Expertenseminare sind Expertenschulungen und Weiterbildungen der DOAG, die mit einer Hands-On-Mentalität über zwei Tage geballtes Fachwissen mit praxisnahen Übungen vermitteln. Profis geben in kleiner Runde ihr großes Know-how weiter und sorgen für einen optimalen Wissenstransfer, der unmittelbar danach angewendet und in die täglichen Aufgaben und Herausforderungen fließen kann. Für ein exquisites Buffet ist während des gesamten Seminars ebenfalls gesorgt. Die Schulungen dauern täglich bis 17 Uhr. Im Anschluss an den ersten Seminartag wartet eine Abendveranstaltung auf die Teilnehmer.

05. - 06.07.2023

Oracle APEX und Oracle Rest

Berliner Expertenseminar mit Marco Patzwahl

Sie lernen die Grundlagen für die Benutzung von Rest-Schnitstellen bei Oracle. Es werden die Installation, das Backup und die Konfiguration bzw. das Tuning und Monitoring der Rest-Schnittstelle behandelt. Im Seminar werden viele Beispiele gezeigt und Best Practices besprochen. Auch die JSON-Verarbeitung (Table => JSon , JSon => Table) steht im Fokus.

30. - 31.08.2023

Oracle Cloud Infrastructure – von der Konsole zur Automation – Kickstart!

Berliner Expertenseminar mit Stefan Oehrli und Martin Berger

Es wird gemeinsam eine Umgebung bestehend aus Compute-Instanzen und Datenbanken aufgebaut. Danach konfigurieren Sie in praktischen Übungen Terraform und deployen anschließend unterschiedliche Ressourcen mithilfe von Terraform und OCI Stacks. Dazu gehören Compute Instances, Database Services, Autonomous Database, Load Balancer und mehr.

27. - 28.09.2023

Oracle Datenbank Indexing

Berliner Expertenseminar mit Randolf Eberle-Geist

In diesem Seminar werden die wichtigsten Themen bezüglich Indizierung mit B*Tree und Bitmap Indizes in der Oracle-Datenbank behandelt – Text / XML / JSON / Domain-Indizes werden zwar je nach verfügbarer Zeit erwähnt und beschrieben, der Schwerpunkt liegt eindeutig auf den B*Tree / Bitmap Indizes.

DOAG

DOAG Datenbank mit Exaday 2023

ON DEMAND

DATENBANK 2023 VERPASST?

Jetzt On-demand-Ticket buchen und Vortragsaufzeichnungen anschauen!

Alle Angebote im On-demand-Ticket-Shop

BUSINESS NEWS

04/2023

„Es ist bequem, wenn man wichtige Entscheidungen der Technologie überlässt. Aber eben nicht immer klug."

Interview mit Antoinette Weibel und Simon Schafheitle von Dr. Thomas Karle und Marcos López

Professor Dr. oec. publ. Antoinette Weibel ist Ordinaria für Personalmanagement an der Universität St. Gallen (eine der zehn besten Wirtschaftsuniversitäten Europas). Seit dem 1. April 2016 ist sie auch Direktorin am Forschungsinstitut für Arbeit und Arbeitswelten der Universität St. Gallen. Sie ist Präsidentin des Geschäftsleitenden Ausschusses des Instituts für Systemisches Management und Public Governance der Universität St. Gallen (IMP-HSG), Mitglied des Vorstandes des Instituts für Kommunikations- und Medienmanagement (MCM-HSG) und des Instituts für Wirtschaftsethik (IWE-HSG) der Universität St. Gallen. Nicht zuletzt ist sie Vorstandsmitglied der Schweizerischen Akademie für Geistes- und Sozialwissenschaften (SAGW).

Dr. oec. Simon Schafheitle ist Assistenzprofessor für Personalmanagement und Künstliche Intelligenz an der Universität Twente (NL). Er wurde im Jahr 2020 mit seiner Arbeit über das Zusammenspiel von Algorithmen, Personalsteuerungspraktiken und Vertrauen am Arbeitsplatz von der Universität St. Gallen promoviert. In seiner Forschungstätigkeit widmet er sich der Frage, wie ein digitaler Humanismus am Arbeitsplatz aussehen kann – also wie Technologie am Arbeitspatz eingesetzt werden kann, damit das Vertrauen der Mitarbeitenden, ihre erlebte Sinnhaftigkeit sowie „die menschliche Komponente" am Arbeitsplatz davon profitieren. In ihrem gemeinsamen Projekt „Kluge HR-Technologie und humanzentrierte Führung" behandeln sie, wie Unternehmen und Verantwortliche zunehmend durch neue Technologien im HR gefordert sind, ethische Fragen zu stärken, um eine lebenswertere Zukunft in und von Organisationen zu gewährleisten. Hierbei müssen der Technologie auch Grenzen gesetzt, mehr institutionalisierte Partizipation der Mitarbeitenden geschaffen und eine (moralische) Sensibilisierung aller Unternehmensteilnehmer in den Vordergrund gerückt werden.

Frau Weibel, Herr Schafheitle, was sind die besonderen Herausforderungen für die Organisationen bei den sich aktuell entwickelnden HR-Technologien?

Es stellen sich drei große Herausforderungen: Erstens bringt der Technologiewechsel eine Umwälzung der Arbeitswelt mit sich. Dadurch stehen viele Ängste im Raum: „Behalte ich meinen Job?", „Bin ich den neuen Anforderungen gewachsen?" etc. Wenn Unternehmen (aber auch der Staat) diesen Ängsten nicht wirksam begegnen – etwa umfassende Um- und Weiterbildung anbieten – bröckelt das Vertrauen in das System. Zweitens wächst die Sorge vor der Überwachung durch Technologien. Die Möglichkeiten werden durch die Verknüpfung von Fortschritten in der Datenerhebung und -analyse stetig vergrößert – der verantwortungsvolle Umgang hinkt jedoch teilweise hinterher. Das schürt Misstrauen. Drittens verbinden sich insbesondere mit dem Ausmaß der vermeintlichen „Klugheit" der Technologie weitere durchaus berechtigte Ängste. Im Mittelpunkt steht die Frage der Beziehung zwischen Technologie und Mensch. Degradieren uns neue Technologien (wie auch in den letzten großen Technologieinnovationen) zum „verlängerten Arm der Maschine" und erhöhen damit das Potenzial einer Dehumanisierung? Oder werden wir zum „besseren" Menschen – im Raum steht etwa die Vision des Transhumanismus (wir werden durch Maschinen zu „Super-"Menschen) oder des Neohumanismus (wir besinnen uns darauf, was wir besser als Maschinen können, und schärfen diese Fähigkeiten).

Was können Organisationen tun, um diesen Herausforderungen wirksam zu begegnen?

Organisationen müssen sich umfassend vorbereiten. Zum einen müssen technologische und ethische Kompetenzen aufgebaut werden. Es gilt nämlich die Technologien kritisch auszuwählen, benutzerorientierte Änderungen zu verlangen und darüber nachzudenken, wo man Technologien einsetzt und wo nicht. Zweitens muss man die Mitarbeitenden mit ins Boot nehmen. Es zeigt sich, wie bei allen Änderungen, dass man Betroffene zu Beteiligten machen sollte. Gute Technologien sollten immer nutzenstiftend, arbeitserleichternd und vertrauensbewahrend wirken. Deshalb ist eine Partizipation von Mitarbeitenden sowohl im Pilotprojekt als auch für Verbesserungen im Verlauf der Nutzung erforderlich. Schließlich sollte der Umgang mit der Technologie kritisch-konstruktiv und lebendig bleiben: Will heißen, wir müssen vermeiden, dass wir unser Gehirn abschalten und alles nur noch auf Autopilot stellen. Oder wie es ein Kollege ausdrückt: Die Gefahr der funktionalen Dummheit ist groß. Es ist bequem, wenn man wichtige Entscheidungen der Technologie überlässt. Aber eben nicht immer klug.

Was verstehen Sie unter „klugen" HR-Technologien? Inwieweit spielen hier auch KI-Ansätze eine Rolle?

Gemeinhin empfindet man eine Technologie dann als intelligent, wenn sie eigenständig kognitive und/oder motorische Aufgaben erledigen kann, die vormals ausschließlich dem Menschen vorbehalten waren. Kognitive Aufgaben reichen von einfachen Vorhersagen (z. B. bezüglich Mitarbeiter-Retention oder Attrition), über Klassifizierungsleistungen (z. B. Wissensmanagement – „Wer weiß was?" – oder Silo-Überbrückung), die Vereinfachung von Datenstrukturen (z. B. Kompetenzmodellierungen, virtuelle Lern- und Karriereassistenten) bis hin zum sogenannten „Reinforcement Learning", das jeder, der schon einmal Berührungspunkte mit OpenAI/ChatGPT hatte, direkt erlebt konnte. Motorische Anwendungen kommen zugegebenermaßen im HR gegenwärtig eher weniger zum Einsatz – die Musik spielt hier, unter anderem, in der Pflege, im Facility-/Hospitality-Management (Pflege-/Putz-/Serviceroboter) oder in verschiedenen Bereichen des öffentlichen Lebens (Stichwort: selbstfahrende Autos).

Technisch gesehen geht es bei den meisten Algorithmen um die Minimierung des Vorhersagefehlers und die zugrunde liegende Frage lautet vereinfacht: Wie viel muss der Mensch zu diesem Prozess beitragen beziehungsweise wie viele Annahmen muss er im Vorfeld treffen, damit dieser Fehler minimal wird. Beim sogenannten „Reinforcement Learning", das gemeinhin als intelligentestes Set von Algorithmen angesehen wird, braucht es kaum Annahmen und menschliches Zutun, sondern der Algorithmus gibt sich über einen Belohnungskreislauf am Ende selbst Feedback, ob ein Output gut oder schlecht war. Im Fall eines (HR-)Chatbots heißt das, der Algorithmus „weiß", dass er dann eine Belohnung bekommt, wenn die Wahrscheinlichkeit, dass das geschätzte Wort nahezu perfekt zum restlichen Satz passt, hoch ist. Und technisch gesehen „will" der Algorithmus seine Belohnungen maximieren, das Lernen kommt also und bleibt in Gang. Notabene: Damit die (HR-)Technologie „klug" wird, braucht es die Arbeit unzähliger sogenannter Crowd Worker. Das sind Personen, die teils unter widrigsten Bedingungen, bei schlechter Bezahlung und zulasten des eigenen mentalen Wohlbefindens nichts anderes tun, als (verstörende) Bilder zu klassifizieren, belästigende und beleidigende Wortsequenzen zu identifizieren oder, noch schlimmer, Videos von Hinrichtungen anzuschauen, um dem Algorithmus zurückzumelden, dass es dafür „keine" Belohnung geben soll.

An welchen Stellen müssen der Technologie Grenzen gesetzt werden und wie kann dies erfolgen?

Was wir gerade eben sehr technisch beschrieben haben, lässt sich über die folgenden drei Kriterien in die HR-Managementpraxis übersetzen: (1) Black-box functioning, (2) Function creep und (3) Prescriptive capabilities. Bitte entschuldigen Sie das Denglisch und lassen Sie uns kurz erläutern, wie sich diese Übersetzung eignet, um der Technologie im Sinne des Mitarbeiterwohls Grenzen zu setzen. Das (1) „Black-box functioning" meint nichts anderes als „hex hex", das heißt, man kann oft nicht oder nur schwer nachvollziehen, wie eine Vorhersage, eine Klassifizierung oder eine Vereinfachung zustande gekommen ist. Hier zeigt die Evidenz: Lassen Sie es die Mitarbeitenden spielerisch ausprobieren, erklären Sie die Funktionsweise, so gut es geht, und – vor allem mit Blick auf das so wichtige Vertrauen am Arbeitsplatz – geben Sie als Chef ruhig zu, dass Sie es auch nicht besser wissen als Ihre Mitarbeitenden.

(2) „Function creep“ meint im Kern: „Der Zweifel ist erhaben“, man hat also eine ungefähre Ahnung vom Funktionsspektrum der Technologie, aber vollends kennt man es erst, wenn man sie eingesetzt hat (und dann kann das Kind natürlich schon in den Brunnen gefallen sein). Hier zeigt die Evidenz: Leisten Sie in Ihrem Unternehmen Kulturarbeit eines kontinuierlichen Ausprobierens, schaffen Sie Routinen, die Technologie kontinuierlich auf den Prüfstand stellt und – gestatten Sie uns diesen persönlichen Kommentar – lassen Sie die Finger von „trustworthy AI-Zertifikaten“.

Schließlich meint „Prescriptive capabilities“ die Fähigkeit von Algorithmen, einen Outcome mit einer gewissen Eintrittswahrscheinlichkeit vorherzusagen, also eine (mehr oder weniger) qualifizierte Entscheidungsgrundlage zu liefern. Mit Blick auf die vorhin angesprochenen Ängste legt die Evidenz hier nahe, das Thema „Automatisierung von Führung“ anzugehen. Trainieren Sie ihre Führungskräfte, Daten richtig zu interpretieren, in einen sinnstiftenden Kontext einzubetten und empathisch zuzuhören. So fühlt sich die/der Mitarbeitende als Mensch und nicht als dehumanisiertes Datensubjekt im besten Sinn Taylors.

Wie sehen Sie die Bereitschaft deutscher Unternehmen, sich in den zuvor genannten Punkten zu verändern?

Puh, das ist eine Frage, deren Antwort wir nur annähern können. Eine vielversprechende Möglichkeit ist es, die Verbreitung der sogenannten „digital literacy“ in den Unternehmen anzuschauen; sie beschreibt die Fähigkeit von Personen (also auch HR-Managern), einfach mit Technologie zu interagieren, sie effektiv einzusetzen sowie auch resultierende ethisch-moralische Dilemmata zu antizipieren und anzugehen. Eurostat nutzt hierfür ein sehr aussagekräftiges Maß, da es die folgenden Kompetenzen umfasst: Informations- und Datenkompetenz, Kommunikation und Zusammenarbeit, Erstellung digitaler Inhalte, Sicherheit und Problemlösungskompetenz. Während sich die Bevölkerungen der Niederlande, Finnlands und Irlands die Top-3-Plätze aufteilen, liegt Deutschland auf Platz 22 (der EU-Schnitt ist auf Rang 16). Eine vielleicht etwas positivere Aussicht liegt im deutschen Mitarbeitermitbestimmungsgesetz begründet, das ihre Teilhabe im Einsatz kluger HR-Technologie vorsieht; so muss qua Gesetz eine Interessensharmonisierung herbeigeführt und über Grenzen der Technologie gesprochen werden.

Am Markt gibt es aktuell zahlreiche Unternehmenssoftware-Hersteller, die umfassende HR-Pakete anbieten. Gibt es Aspekte bei der Auswahl und auch Einführung kluger HR-Systeme, die besonders beachtet werden sollten?

Das ist in der Tat eine knifflige Frage und ich möchte keinesfalls die zahlreichen Best-Practice-Ratgeber zu People Analytics wiedergeben, getreu dem Motto: Es ist alles gesagt, nur noch nicht von uns! Spaß bei Seite – mit Blick auf die Technologie an sich fallen uns zwei Punkte ein: (1) Maßanfertigung statt „von der Stange“ und (2) Prinzip der Datensparsamkeit. Ersteres bedeutet, die Funktionalitäten der Technologie auf den Prüfstand der HR Value Proposition zu stellen und sich zu fragen: Passt das zusammen? Datensparsamkeit meint ganz praktisch eine grobe Granularität von Datensammlung und -analysen, damit, neben rechtlichen Einschränkungen, auch nicht das Gefühl entsteht, der Mitarbeitende müsse die Hosen runterlassen.

@rawpixel.com Quelle: freepik.com

Was sich als zusätzliche Daumenregel nützlich erweist, ist eine kurze Recherche zum Anbieter der Technologie, also etwa ob die Firma ein absolutistischer Chauvi-Laden im Musk'schen Sinn ist oder ob Werte wie Diversität, Inklusion und faire Bezahlung authentisch gelebt und praktiziert werden; oder eben, wie sie ihre Crowd Worker behandelt.

Wie definieren Sie einen digitalen Humanismus am Arbeitsplatz? Welche Empfehlungen leiten Sie daraus für die moderne Unternehmensführung ab?

Wie schon erwähnt: Der digitale Humanismus setzt voraus, dass sich Organisationen bewusst mit den Anwendungsmöglichkeiten der Technologie auseinandersetzen. Es geht also vorerst darum, Technologie so einzusetzen, dass sie Arbeit erleichtert, den Menschen unterstützt und Vertrauen im und in das Unternehmen bewahrt. In einem zweiten – aber viel weitreichenderem Schritt – muss die Frage gestellt werden: Was ist die gute Arbeit und das gute Unternehmen der Zukunft? Wie können wir „lebendige" Unternehmen schaffen, mit sinnhafter Arbeit, die Menschen ermöglicht, ihr Potenzial für und mit anderen Menschen zum „Wohle der Menschheit" einzubringen. Konkret geht es darum, ein neues Geschäftsmodell zu schaffen: Maschinen, die uns unterstützen; Menschen, die mit Herz und Kreativität neue Ideen, neue Formen der Zusammenarbeit und neue Produkte auf den Weg bringen für eine Gesellschaft, die nachhaltig und generativ ist.

Wenn kluge HR-Systeme die Arbeitswelt menschlicher machen (sollen), wie bemisst man den Erfolg dieses transformatorischen Prozesses? Interviews, Fragebögen, wirtschaftlicher Erfolg des Unternehmens?

Hmm – ja klar, das sind die üblichen Messinstrumente. Wir würden aber meinen, wenn wir nach „Lebendigkeit" streben, sollten wir in glänzende Augen sehen können. Und wir sollten spüren, dass Mitarbeitende mitdenken und mitfühlen. Im Kern bedeutet das, dass wir unser übliches Messarsenal ergänzen müssen. Was sind die Geschichten, die man im Unternehmen erzählt? Wo wird gelacht, wo spürt man Inspiration, wo sind alle dabei? Vielleicht ist es an der Zeit, eine Organisationsethnographin anzustellen – oder vielleicht müssen wir alle wieder unsere Sinne schärfen. Übrigens sollten die neuen Produkte auch zu glänzenden Augen bei unseren Kunden und Stakeholdern führen.

Wie kann man der Versuchung widerstehen, kluge HR-Technologie dumm einzusetzen?

Indem wir dafür sorgen, dass dies keine Versuchung für uns ist. Weil wir andere Dinge schätzen – etwa Vertrauen und Lebendigkeit. Und weil wir Gespür für das Gute und Schöne entwickelt haben. Und weil wir uns das Denken – die kritische Betrachtung unseres Tuns – wieder angewöhnt haben.

Frau Weibel, Herr Schafheitle, herzlichen Dank für das Interview.

„Echte“ digitale Abrechnungsprozesse auf Basis der XRechnung

Axel Bröker, Dr. Thomas Karle, Matthias Sauer, PROMATIS software GmbH, Ettlingen (TechnologieRegion Karlsruhe)

Die Transformation des Abrechnungsprozesses in eine durchgehend digital ausgeführte Verarbeitung ist eine Aufgabe, die bei Digitalisierungsprojekten in nahezu jeder Organisation umzusetzen ist. Das Ziel ist hierbei nicht das Erstellen und Versenden von PDFs, sondern die Umsetzung eines komplett digitalisierten Prozesses ausgehend von der Rechnungserstellung, über die elektronische Übermittlung und Zustellung bis hin zum Import der Rechnungsdaten bei der Zielorganisation zur weiteren Bearbeitung. Für diesen wichtigen Digitalisierungsfall haben Bund und Länder im Juni 2017 eine Standardisierung beschlossen – die sogenannte XRechnung – und diese ab dem 27. November 2020 für Unternehmen zur Pflicht erklärt, wenn Rechnungen an Bundesbehörden übermittelt werden müssen. Der Artikel beschreibt das generelle Vorgehen beim Einsatz der XRechnung. Hierbei werden mögliche Einsatzszenarien, aber auch die Grenzen beschrieben.

Einleitung

Die Pflicht zur Übermittlung elektronischer Rechnungen bei Bundesbehörden im Standard XRechnung setzt viele deutsche Unternehmen unter akuten Zugzwang. Dies kann jedoch als guter Startpunkt für ein weiterreichendes Digitalisierungsprojekt im Unternehmen genutzt werden. Denn die XRechnung kann auch generell für die digitale Rechnungsabwicklung zwischen Unternehmen genutzt werden. In vielen europäischen Ländern werden elektronische Rechnungen bereits großflächig im Business2Buisness-Bereich (kurz: B2B-Bereich) eingesetzt. Gerade der in Deutschland stark vertretene Mittelstand könnte davon profitieren und den Order2Cash-Prozess in den Unternehmen dadurch deutlich verbessern. Für die Umsetzung eines auf einer elektronischen Rechnung basierenden Abrechnungsprozesses stehen verschiedene Mechanismen zur Verfügung. Nachfolgend werden diese Mechanismen erläutert und die Einbettung in ERP-Systeme – insbesondere auch in die Oracle-ERP-Systeme – beschrieben.

Was ist eigentlich eine elektronische Rechnung und was nicht?

Eine elektronische Rechnung (kurz: E-Rechnung) ist ein elektronisches Dokument mit gleichem Inhalt und den gleichen Rechtsfolgen wie eine Rechnung in Papierform *[1]*. Sie besteht aus einer nach genauen Vorgaben aufgebauten Datenstruktur, die in einem elektronischen Format erstellt, übermittelt, empfangen und weiterverarbeitet werden kann. Inhalt und Format der Datenstruktur werden europaweit einheitlich durch die europäische Norm EN 16931 festgelegt. Mit einer E-Rechnung können dadurch Rechnungsinformationen in einem durchgängig digitalen Abrechnungsprozess von der Erstellung bis hin zur Zahlung der Rechnungsbeträge automatisiert verarbeitet werden.

Ein grundlegendes Kriterium für eine E-Rechnung ist die Verwendung einer definierten maschinenlesbaren Datenstruktur *[1]*. Dadurch stellt sie ein elektronisches Dokument dar, das rechtlich wie eine Papierrechnung behandelt wird (siehe EU-Richtlinie 2014/55/EU und E-Rechnungsverordnung – E-RechV). Eine Bilddatei, ein PDF oder auch eine eingescannte Papierrechnung erfüllen diese gesetzlichen Anforderungen dadurch nicht.

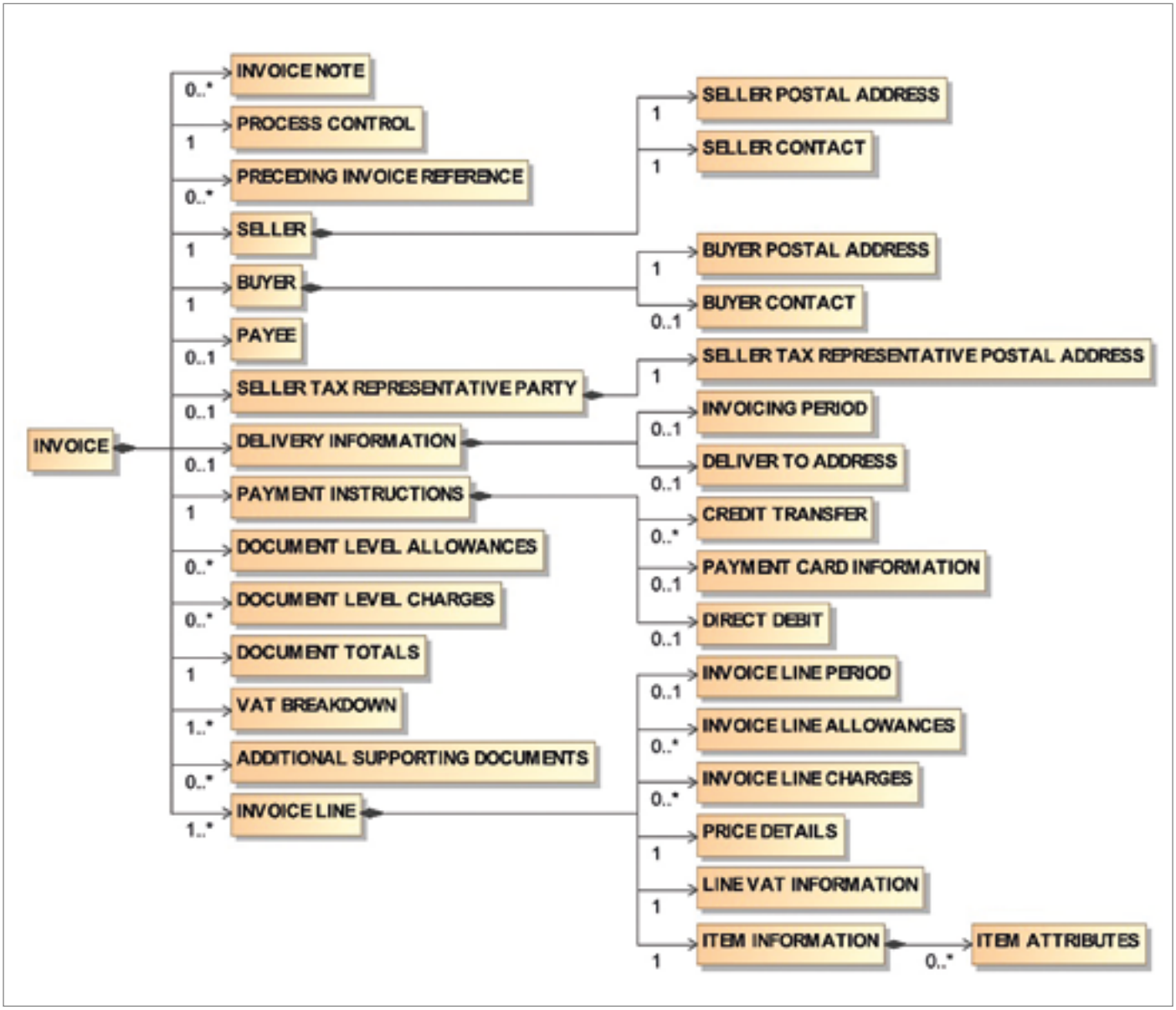


Abbildung 1: Grundstruktur einer XRechnung [3]

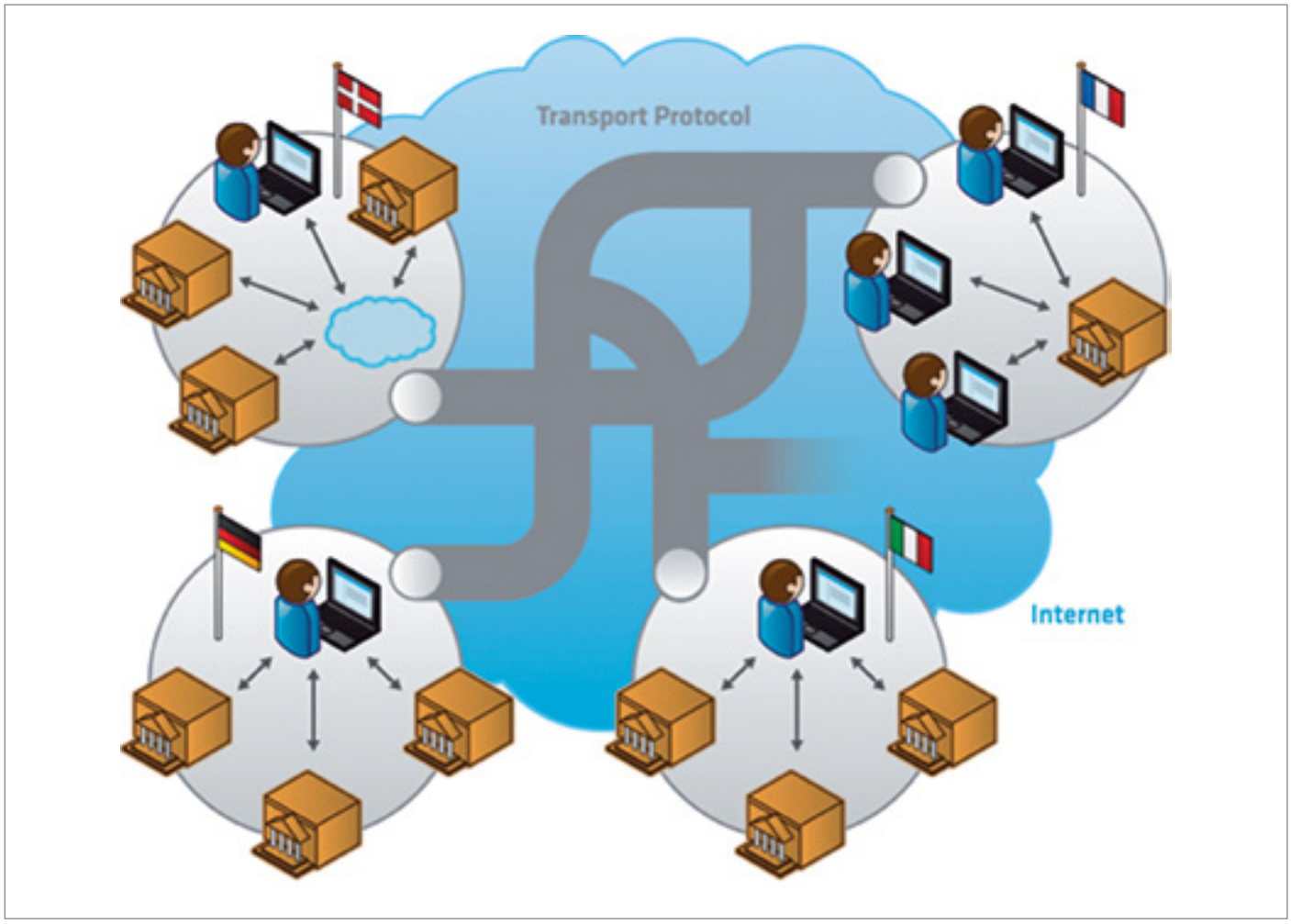


Abbildung 2: Peppol-Netzwerk [4]

XRechnung – Mögliche Datenstruktur für eine E-Rechnung

Wie sieht nun eine solche vordefinierte Datenstruktur aus? Hierzu wurde in Deutschland die XRechnung als ein XML-basiertes semantisches Datenmodell definiert, das als Standard für elektronische Rechnungen dienen soll und insbesondere im Rechnungsaustausch mit öffentlichen Auftraggebern in Deutschland verwendet wird *[2]*. Der XRechnung-Standard wurde 2017 durch den IT-Planungsrat für Bund und Länder in der Version 1.0 beschlossen. Die XRechnung ist eine nationale Ausgestaltung der europäischen Norm EN 16931 (eine sogenannte CIUS) *[2]*. Die XRechnung ist ein Standard für die Art und die technische Zusammensetzung der Rechnungsinformationen in einem XML-Datensatz im Sinne einer E-Rechnung. Er ermöglicht den Empfang und die Weiterverarbeitung durch Softwaresysteme. Eine XRechnung enthält neben den umsatzsteuerrelevanten Bestandteilen weitere Informationen wie Zahlungsbedingungen, Bankverbindungen und Lieferantendaten. Dabei werden die Grundsätze zur ordnungsmäßigen Führung und Aufbewahrung von Büchern, Aufzeichnungen und Unterlagen in elektronischer Form sowie zum Datenzugriff (GoBD) eingehalten.

Abbildung 1 zeigt die Datenstruktur der XRechnung *[3]*. Die Grundstruktur ist hierarchisch aufgebaut und stellt dadurch mehrere Ebenen bereit. Hier findet man die beiden groben Ebenen Rechnungskopf (Invoice) und Rechnungszeile (Invoice Lines) wieder. Darüber hinaus sind dann jeweils Details als Informationselemente mit Codelisten zugeordnet. Es dürfen einzelne Aspekte im Rahmen dieser Struktur auch enger gefasst werden: Hierzu gibt es die Möglichkeit, optionale Informationselemente als Pflichtelemente zu deklarieren. Neue Geschäftsregeln dürfen zu existierenden Informationselementen ergänzt werden und verwendete Codelisten können auf eine Teilmenge von Werten beschränkt werden. Darüber hinaus darf die Struktur auch nach bestimmten Regeln im Rahmen einer sogenannten Extension erweitert werden: Hierbei sind das Hinzufügen neuer Informationselemente, die Erweiterung von Codelisten und die Ergänzung semantischer Bedeutungen von Feldern oder Regeln möglich.

Wie wird eine XRechnung übermittelt?

Neben den zuvor beschriebenen inhaltlichen Aspekten sind für die XRechnung auch Mechanismen für die Übermittlung zu einem Rechnungsempfänger vorgesehen. Hierzu werden mit den Rechnungseingangsplattformen ZRE und OZG-RE den Rechnungssendern verschiedene Übertragungskanäle zur Übermittlung von E-Rechnungen an die Rechnungsempfänger zur Verfügung gestellt, zum Beispiel über Weberfassung, Upload oder E-Mail *[2]*.

Eine deutlich elegantere Möglichkeit steht mit Peppol (Pan-European Public Procurement OnLine) zur Verfügung *[4]*. Unternehmen sollen durch das Peppol-Netzwerk

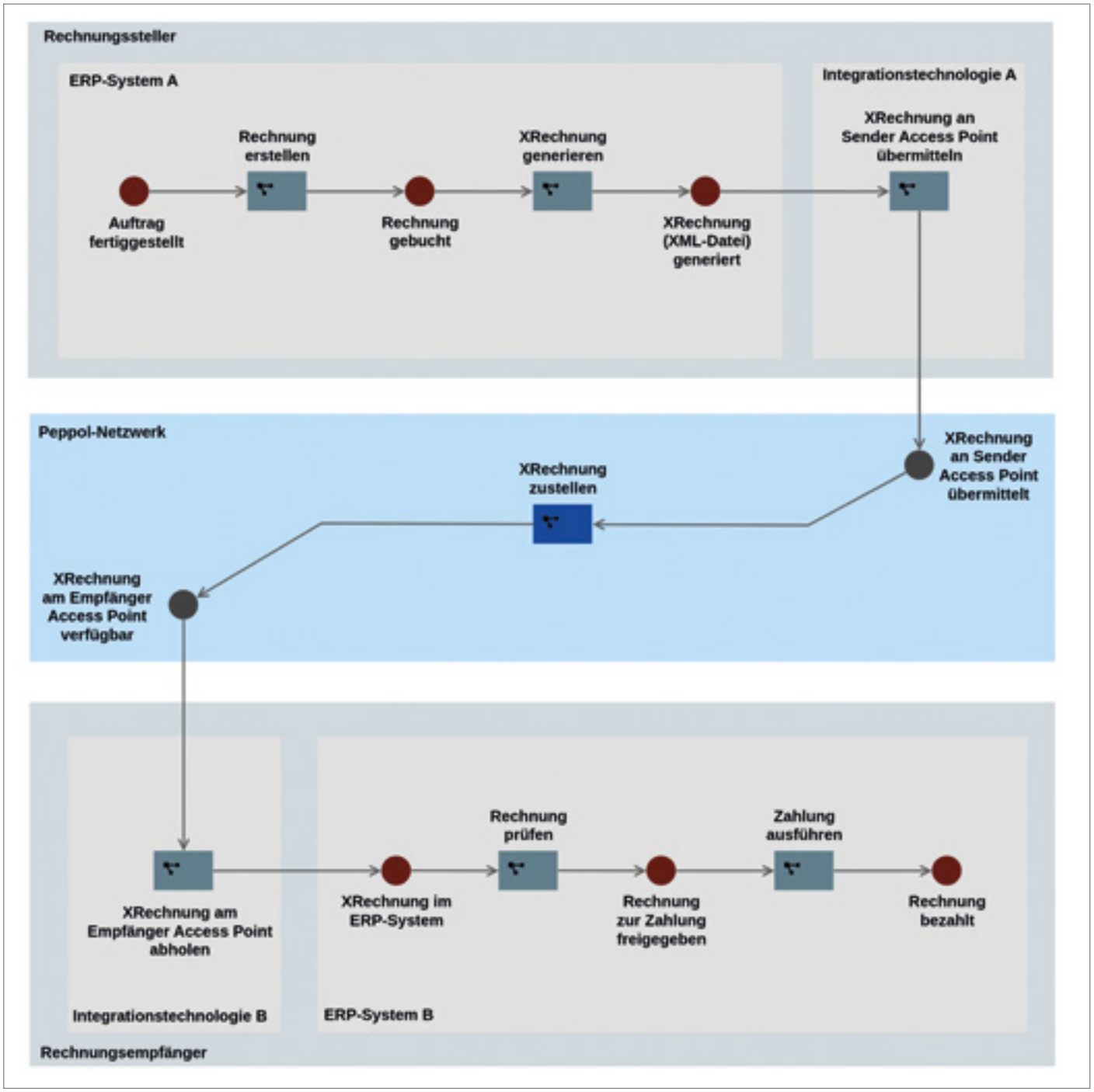


Abbildung 3: ERP/Peppol-basierter vollautomatischer Abrechnungsprozess (© 2023 PROMATIS software GmbH)

in die Lage versetzt werden, elektronisch mit allen europäischen Regierungsinstitutionen im Rahmen von Beschaffungsprozessen – dem sogenannten Business-2Government-Bereich (kurz: B2G-Bereich) – zu kommunizieren (*Abbildung 2*). Darüber hinaus soll es jedoch auch im B2B-Bereich zur Harmonisierung und Verbesserung von Prozessen genutzt werden. Peppol stellt hierzu eine Sammlung von Komponenten und Spezifikationen bereit, mit deren Hilfe innerhalb des Peppol-Netzwerks Dokumente auf standardisierte Art und Weise zwischen unterschiedlichen Business-Partnern ausgetauscht werden können. Die bereitgestellten Übermittlungsmechanismen können innerhalb von Europa, auf Bundes-, Landes- und Kommunalebene eingesetzt werden. Explizit ist auch die Verwendung im gesamten B2B-Bereich vorgesehen. Die Anbindung beteiligter Partner erfolgt über sogenannte Peppol Access Points.

Verarbeitung in ERP-Systemen

Für die Umsetzung eines digitalisierten Abrechnungsprozesses über ERP-Systeme muss einerseits für die Erstellung einer entsprechend der XRechnung konformen XML-Datei gesorgt werden und andererseits der Übermittlungsprozess implementiert werden. Die Erstellung der XML-Datei kann in der Regel entweder durch Konfiguration des ERP-Systems oder die Erstellung eines Reports mit einem entsprechenden Tool umgesetzt werden. Für den Übermittlungsprozess kann das Senden der XRechnung an einen Access Point des Peppol-Netzwerks ausgehend vom ERP-System mit Integrationstechnologien umgesetzt

werden. Die Zustellung an den Rechnungsempfänger erfolgt dann über das Peppol-Netzwerk. Je nach vorhandener IT-Infrastruktur können somit passende Lösungen umgesetzt werden. *Abbildung 3* zeigt einen komplett automatisierten Gesamtprozess ausgehend vom ERP-System A des Rechnungsstellers bis zur Ausführung der zugehörigen Zahlung im ERP-System B des Rechnungsempfängers.

Mögliche Umsetzungen am Fallbeispiel Oracle NetSuite

Eine einfache Möglichkeit für Oracle NetSuite ist beispielsweise die Verwendung ergänzender 3rd-Party-Produkte, die eine XRechnung out of the box liefern. Ein zweiter Ansatz ist die Nutzung des Electronic-Invoicing-Moduls von NetSuite. Die Umsetzung der Erstellungsfunktion für eine XRechnung kann hier dann durch Konfiguration dieses Moduls erfolgen, das heißt ohne Entwicklung und komplett durch Standardfunktionalität. Die XRechnung kann dann entweder in einem der beiden Portale ZRE oder OZG-RE hochgeladen oder per E-Mail versandt werden. Für eine vollständig automatisierte Verarbeitung ist dann die Entwicklung einer Anbindung an Peppol erforderlich. Dies kann beispielsweise durch Verwendung der Oracle Integration Cloud umgesetzt werden. Im Peppol-Netzwerk kann dann der kostenlos verfügbare Access Point der Bundesdruckerei verwendet werden, um die Übermittlung durchzuführen.

Einsatzgebiete

Der Hauptanwendungsfall der XRechnung ist aktuell noch die digitalisierte Umsetzung von Abrechnungsprozessen im Zusammenhang mit öffentlichen Auftraggebern. Jedoch kann der Ansatz auch im B2B-Bereich angewendet werden. Hier sind der Fantasie bezüglich Format und Übermittlung von E-Rechnungen aktuell keine Grenzen gesetzt. Dies ist allerdings eher Fluch statt Segen, denn die austauschenden Parteien müssen sich diesbezüglich einigen. Der Einsatz der XRechnung scheint naheliegend, ist jedoch im B2B-Umfeld noch nicht weit verbreitet. Es können auch Rechnungen, die nicht dem Format der XRechnung entsprechen, über einen Peppol Access Point verschickt werden. Zu beachten ist hier jedoch, dass der kostenfreie Peppol Access Point der Bundesdruckerei nicht für die B2B-Kommunikation genutzt werden kann. Alternative Formate sind beispielsweise ZUGFeRD, bei dem die E-Rechnung in ein PDF eingebettet wird, oder der E-Kommunikations-Dinosaurier EDI.

Fazit

Mit der XRechnung wurde ein wichtiger Schritt in Richtung Digitalisierung im Umgang mit öffentlichen Behörden unternommen. Die Vorteile für alle Beteiligten sind klar: Reduzierung der Kosten, Minimierung der zeitlichen Abläufe und somit eine Erhöhung der Unternehmensliquidität durch standardisierte Datenformate und Prozesse. Darüber hinaus können die XRechnung und die damit zusammenhängenden Mechanismen zur Übermittlung und Verarbeitung jedoch auch zur Implementierung einer vollautomatisierten Rechnungsabwicklung zwischen Unternehmen mit den gleichen zuvor genannten Vorteilen genutzt werden. Hier besteht in Deutschland im Business2Buisness-Bereich aktuell noch einiges an Potenzial, vor allem bei den KMU. Die erforderliche Infrastruktur und die Mechanismen sind vorhanden, sie müssen nur noch konsequent genutzt werden.

Literatur

[1] E-Rechnung in der Bundesverwaltung: *https://www.e-rechnung-bund.de*

[2] XRechnung: *https://www.e-rechnung-bund.de/faq/xrechnung*

[3] XRechnung Spezifikation: *https://xeinkauf.de/app/uploads/2022/11/220-XRechnung-2022-01-27-2.pdf*

[4] Peppol: *https://peppol.eu*

Axel Bröker
axel.broeker@promatis.de

Axel Bröker ist IT Service Manager bei PROMATIS und berät Kunden hierbei als Finance-Experte sowohl bei technischen Fragen als auch bei der Umsetzung gesetzlicher Anforderungen im Rahmen von ERP-Systemen.

Dr. Thomas Karle
thomas.karle@promatis.de

Dr. Thomas Karle verantwortet den Bereich Business Process Management in der PROMATIS Unternehmensgruppe und ist als Strategieberater und Projektleiter in der geschäftsprozessorientierten Implementierung von Unternehmenssoftware-Lösungen aktiv.

Matthias Sauer
matthias.sauer@promatis.de

Matthias Sauer ist Vice President Application Development bei PROMATIS und als technischer Projektleiter und Systemarchitekt im Rahmen der Konzeption und Umsetzung von Zusatz- und Integrationskomponenten in Unternehmenssoftware-Projekten tätig.

Die digitale Transformation in einer Betrachtung mit ChatGPT

ChatGPT, OpenAI und Co-Autor Armin Wildenberg, DOAG e.V.

Die digitale Transformation stellt die meisten, wenn nicht alle Unternehmen und die öffentliche Verwaltung auf sämtlichen Ebenen vor eine große noch zu bewältigende Herausforderung. Obwohl während der Pandemie viele Unternehmen und Verwaltungen aus der Notwendigkeit heraus Schritte in Richtung der Transformation vorgenommen haben, bleibt noch viel zu realisieren. ChatGPT beschleunigt diese Transformation seit Beginn des Jahres 2023 zusätzlich.

Die rasante Entwicklung der Nutzung, die vielfältige Verwendung und die notwendige Diskussion über ChatGPT in den ersten Monaten des Jahres werden an keinem in der IT vorbeigegangen sein. Insbesondere der Gedanke an die umfangreichen Informationsmengen im Zugriff von ChatGPT und die schnelle Generierung von Texten durch ChatGPT legten mir den Gedanken nahe, doch diese KI nach der digitalen Transformation zu fragen. Im folgenden Text hat ChatGPT die Frage nach einer Definition der digitalen Transformation beantwortet sowie zu den Auswirkungen und Herausforderungen an die Unternehmen und zur aktuellen Umsetzung in Deutschland Stellung genommen. Des Weiteren waren die Auswirkungen auf die Mitarbeiter und die Maßnahmen zur Vorbereitung der Mitarbeiter auf die digitale Transformation Fragestellungen, die ChatGPT beantworten sollte. Das Ergebnis folgt im Anschluss an diesen Abschnitt. Alle Passagen mit Ausnahme der Überschriften – die

wurden vom Co-Autor teils aufgehübscht – lieferte Chat GPT.

Die digitale Transformation: Definition und Auswirkungen auf Unternehmen

In der heutigen digitalen Welt ist die digitale Transformation ein Begriff, der immer wieder auftaucht. Aber was genau bedeutet dieser Begriff und wie beeinflusst er Unternehmen?

Definition der digitalen Transformation

Die digitale Transformation bezieht sich auf den Prozess, bei dem Unternehmen ihre Geschäftsprozesse und -modelle durch den Einsatz digitaler Technologien optimieren und verbessern. Dies kann eine Vielzahl von Veränderungen beinhalten, wie beispielsweise die Einführung neuer Softwarelösungen, die Automatisierung von Prozessen oder die Umstellung auf Cloud-Computing.

Auswirkungen auf Unternehmen

Die digitale Transformation hat erhebliche Auswirkungen auf Unternehmen, insbesondere in Bezug auf ihre Geschäftsprozesse, ihre Kundenbeziehungen und ihre Mitarbeiter. Im Folgenden werden einige der wichtigsten Auswirkungen genauer erläutert:

- ***Effizienzsteigerung***

Eine der Hauptwirkungen der digitalen Transformation ist die Steigerung der Effizienz von Geschäftsprozessen. Durch den Einsatz digitaler Technologien können Unternehmen ihre Prozesse automatisieren und somit Zeit und Ressourcen sparen. Darüber hinaus können sie durch die Analyse von Daten und die Nutzung von Business-Intelligence-Lösungen fundierte Entscheidungen treffen.

- ***Verbesserte Kundenbeziehungen***

Die digitale Transformation ermöglicht es Unternehmen, ihre Kundenbeziehungen zu verbessern. Durch den Einsatz von Technologien wie Social Media, Chatbots und künstlicher Intelligenz können Unternehmen eine personalisierte Erfahrung bieten und Kundenbedürfnisse besser verstehen. Darüber hinaus können Unternehmen durch die Analyse von Daten besser auf die Bedürfnisse ihrer Kunden eingehen und ihnen genau das bieten, was sie suchen.

- ***Veränderungen der Arbeitsweise***

Die digitale Transformation führt auch zu Veränderungen in der Arbeitsweise von Unternehmen. Durch den Einsatz von Technologien wie Collaboration-Tools und Cloud-Computing können Mitarbeiter flexibler arbeiten und von überall aus auf Informationen zugreifen. Dies kann auch dazu beitragen, die Zusammenarbeit zwischen Mitarbeitern zu verbessern und Innovationen zu fördern.

- ***Neue Geschäftsmodelle***

Die digitale Transformation kann auch dazu führen, dass Unternehmen neue Geschäftsmodelle entwickeln. Durch den Einsatz digitaler Technologien können Unternehmen neue Produkte und Dienstleistungen einführen und neue Märkte erschließen. Darüber hinaus können sie durch die Analyse von Daten neue Geschäftsmöglichkeiten identifizieren und diese nutzen.

Fazit

Die digitale Transformation ist ein wichtiger Prozess, der erhebliche Auswirkungen auf Unternehmen hat. Durch den Einsatz digitaler Technologien können Unternehmen ihre Effizienz steigern, ihre Kundenbeziehungen verbessern, Veränderungen in der Arbeitsweise umsetzen und neue Geschäftsmodelle entwickeln. Unternehmen, die die digitale Transformation erfolgreich umsetzen, werden in der Lage sein, sich an die sich schnell verändernde digitale Welt anzupassen und wettbewerbsfähig zu bleiben.

Der Stand der Digitalisierung in den deutschen Unternehmen

Es gibt in Deutschland große Unterschiede im Digitalisierungsgrad der Unternehmen, abhängig von Branche und Unternehmensgröße. Allerdings hat die Covid-19-Pandemie im Jahr 2020 gezeigt, dass viele Unternehmen in Deutschland noch nicht ausreichend auf die Herausforderungen der Digitalisierung vorbereitet waren.

Laut einer Studie des Digitalverbands Bitkom aus dem Jahr 2020 hatten beispielsweise nur 39 Prozent der Unternehmen in Deutschland eine Digitalstrategie. Besonders kleinere Unternehmen sind oft noch nicht ausreichend digitalisiert und haben Schwierigkeiten, die notwendigen Investitionen in digitale Technologien und Prozesse zu tätigen.

Es gibt jedoch auch viele deutsche Unternehmen, die in der digitalen Transformation bereits weit fortgeschritten sind und innovative digitale Lösungen einsetzen, um ihre Geschäftsprozesse und -modelle zu verbessern. Insbesondere in Branchen wie der Automobilindustrie, dem Maschinenbau oder der Chemieindustrie gibt es viele erfolgreiche Beispiele für Unternehmen, die sich auf die Digitalisierung konzentrieren und digitale Innovationen vorantreiben.

Insgesamt ist die digitale Transformation in Deutschland ein wichtiger Bereich, in dem viele Unternehmen noch viel Potenzial haben, um wettbewerbsfähig zu bleiben und zukunftsfähig zu sein.

Die Wirkung der digitalen Transformation auf die Mitarbeiter in den Unternehmen.

Die digitale Transformation hat erhebliche Auswirkungen auf Mitarbeiter und verändert ihre Arbeitsweise sowie ihre Rolle im Unternehmen. Hier sind einige der wichtigsten Auswirkungen und Veränderungen:

- ***Automatisierung von Arbeitsprozessen***

Durch den Einsatz von digitalen Technologien können viele Arbeitsprozesse automatisiert werden. Routinetätigkeiten, die früher manuell durchgeführt wurden, werden zunehmend von digitalen Systemen übernommen. Dies führt zu einer Veränderung der Tätigkeiten und Aufgaben der Mitarbeiter. Sie müssen sich mit neuen Systemen und Prozessen auseinandersetzen und sich gegebenenfalls in neuen Bereichen weiterbilden.

- ***Flexibilität und neue Arbeitsmodelle***

Die digitale Transformation ermöglicht auch neue Arbeitsmodelle, wie beispielsweise Remote Work oder Home Office. Durch den Einsatz von Collaboration-Tools und Cloud-Technologien können Mitarbeiter von überall aus arbeiten und auf Informationen zugreifen. Dies ermöglicht mehr Flexibilität in der Arbeitsweise und eine bessere Vereinbarkeit von Beruf und Privatleben.

- ***Notwendigkeit neuer Fähigkeiten***

Mit der zunehmenden Digitalisierung und Automatisierung wird auch die Nachfrage nach bestimmten Fähigkeiten und Kennt-

nissen steigen. Mitarbeiter müssen sich auf neue Arbeitsprozesse und -systeme einstellen sowie in der Lage sein, mit neuen Technologien umzugehen. Auch die Fähigkeit, Daten zu analysieren und auszuwerten, wird in vielen Bereichen immer wichtiger.

- ***Veränderung der Unternehmenskultur***

Die digitale Transformation kann auch zu einer Veränderung der Unternehmenskultur führen. Unternehmen müssen eine Kultur der Innovation und des lebenslangen Lernens fördern, um sich an die sich schnell verändernden Anforderungen anzupassen. Dies erfordert eine offene Kommunikation und eine Bereitschaft, neue Ansätze auszuprobieren.

- ***Neue Arbeitsplatzsicherheit***

Die Automatisierung und Digitalisierung können auch Auswirkungen auf die Arbeitsplatzsicherheit haben. In einigen Fällen können bestimmte Tätigkeiten durch digitale Systeme ersetzt werden, was zu einer Reduzierung des Personalbedarfs führen kann. Andererseits können durch die Digitalisierung auch neue Jobs entstehen und Mitarbeiter können sich auf neue Bereiche spezialisieren.

Insgesamt hat die digitale Transformation erhebliche Auswirkungen auf Mitarbeiter und erfordert eine Anpassung an neue Arbeitsprozesse und -modelle. Unternehmen sollten die Mitarbeiter aktiv einbeziehen und unterstützen, um eine erfolgreiche Digitalisierung zu erreichen.

Einbindung und Vorbereitung von Mitarbeitern auf die digitale Transformation

Um ihre Mitarbeiter auf die digitale Transformation vorzubereiten, können Unternehmen verschiedene Maßnahmen ergreifen. Hier sind einige wichtige Ansätze:

- ***Schulungen und Trainings***

Eine der wichtigsten Maßnahmen ist die Schulung der Mitarbeiter auf neue Technologien und Arbeitsprozesse. Unternehmen können Schulungen und Trainings anbieten, um das technische Know-how und die Fähigkeiten der Mitarbeiter zu verbessern. Hierbei können auch digitale Lernformate, wie etwa E-Learning-Plattformen oder Webinare, zum Einsatz kommen.

- ***Förderung einer Lernkultur***

Unternehmen sollten eine Kultur der kontinuierlichen Weiterbildung und des lebenslangen Lernens fördern. Dies kann durch die Etablierung von internen Lernplattformen oder durch regelmäßige Trainings- und Schulungsprogramme unterstützt werden.

- ***Digitalisierung der Arbeitsumgebung***

Die Arbeitsumgebung kann auch durch den Einsatz von digitalen Technologien verbessert werden. Unternehmen können beispielsweise Collaboration-Tools und Cloud-Technologien einführen, um die Zusammenarbeit und den Informationsaustausch zwischen den Mitarbeitern zu verbessern.

- ***Change Management***

Die digitale Transformation erfordert oft auch eine Veränderung der Arbeitsprozesse und -modelle. Hier ist es wichtig, die Mitarbeiter aktiv in den Veränderungsprozess einzubeziehen und ihnen die Gründe und Vorteile der Veränderungen zu vermitteln. Ein strukturiertes Change Management kann hierbei helfen.

- ***Förderung von Innovationen***

Unternehmen sollten auch eine Kultur der Innovation fördern, um die Kreativität und das Engagement der Mitarbeiter zu fördern. Hierbei können auch interne Innovations- oder Ideenwettbewerbe helfen.

Insgesamt ist es wichtig, die Mitarbeiter aktiv in die digitale Transformation einzubeziehen und ihnen die notwendigen Ressourcen und Fähigkeiten zur Verfügung zu stellen, um eine erfolgreiche Digitalisierung zu erreichen.

Schlusswort des „Co-Autors“

So sieht also eine aktuelle, umfassende Betrachtung des Themas „Digitale Transformation“ durch die KI aus. Dieser Artikel entstand schneller als die Idee und die Ausformulierung der Fragen. Beeindruckend. Zukünftig werden wir möglicherweise nicht mehr erkennen können, wer solche Ausarbeitungen verfasst hat. Beide Themen, digitale Transformation und KI, werden uns sicherlich in den nächsten Ausgaben unseres Magazins wieder begegnen – und dann selbstverständlich mit menschlichen Autoren als Verfassern.

Armin Wildenberg
armin.wildenberg@doag.org

Armin Wildenberg ist gelernter Entwickler und war seit 1984 in der IT der IKB Deutschen Industriebank AG in Düsseldorf tätig. In den Jahren 2005 bis 2018 war er als Teamleiter und Direktor zuständig für Entwicklung des Kreditsystems KreDa der Bank. Die Basis dieser Anwendung bilden ORACLE DB, ORACLE FORMS und Reports sowie seit 2013 ORACLE ADF. 2019 wechselte er in den Fachbereich Competence Center Kredit. Über die Development Community ergaben sich die ersten Berührungspunkte mit der DOAG. Er war seit der Gründung 1988 bis 2020 Mitglied im DOAG Anwenderbeirat. Seit 2020 gehört er als Leiter der Data Analytics Community dem Vorstand der DOAG an und bringt sich in seinem Vorruhestand in die Gestaltung der DOAG ein. Zeitweise arbeitet er als Trainer für INNODREI. Der aktuelle persönliche Schwerpunkt liegt auf den Themen Strategie und Innovation sowie in den agilen Methoden.

#CloudLand2023
www.cloudland.org

CloudLand

DAS FESTIVAL DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY 2023

• Microservices & DDD • CI/CD & Automatisierung
• Container & Cloud-Technologien • DevOps & Methodik

20. - 23. JUNI
im Phantasialand in Brühl

TAG 2

Summer Night

Summer Night powered by InterFace AG the face of informatics

Eventpartner Heise Medien

Und plötzlich war Lockdown – Lehren aus einer Pandemie-Transformation

Christian Linck, DKV Mobility/Customer Product Services

Der Anfang des Jahres 2020 hat die Welt erschüttert. Im Traum hätte niemand daran gedacht, dass es eine Pandemie geben könnte und welche Konsequenzen das für unser aller tägliches Leben hat. Auch wenn es langsam Zeit für ein generelles Fazit zu Covid 19 ist, glaube ich nicht, dass ich der richtige dafür bin. Allerdings kann ich zusammenfassen, was sich in der Quarantänezeit für mich und auch für meine Mitarbeiter geändert hat.

Mitarbeiter finden, noch nie so einfach und doch so schwer

Am Anfang ist immer das Recruiting. In unserem Fall eher klassisch organisiert: zwei Vor-Ort-Termine sowie ein Assessment passend zur Position mit vier bis fünf Personen in einem Raum. Am Ende hat man zwei Stunden mit einem Kandidaten verbracht und trifft eine Entscheidung über einen gemeinsamen weiteren Weg. Bis zuletzt war das der erwartete Standard. Corona hat diesen jedoch nachhaltig verändert. Erste Interviews finden heute per Videokonferenz statt und sind wesentlich schneller organisiert, weil nun nicht mehr alle Beteiligten – auf beiden Seiten – dafür am selben Ort sein müssen. Die darauffolgenden Assessments mussten wir auf Formate umbauen, die ein digitales Zusammenarbeiten zulassen, indem zum Beispiel Livecoding-Plattformen und andere Tools während des Auswahlprozesses genutzt werden. Auch ist der zeitliche Abstand zwischen diesen Terminen kürzer als vorher und sie finden wesentlich schneller hintereinander statt. Der gesamte Prozess der Mitarbeiterfindung ist im Vergleich schneller geworden und würde ohne die notwendigen gesetzlichen Vorgaben noch erheblich weniger Zeit in Anspruch nehmen.

Aber auch auf der Bewerberseite gibt es Änderungen. Ich erinnere mich noch an ausgeklügelte, fein formulierte Anschreiben und Lebensläufe. Heute sind diese eher kein Thema mehr – die meisten Bewerbungen kommen mit einem Profil aus einer sozialen Plattform in den Prozess, wie etwa LinkedIn. Der häufigste Eingangskanal ist auch nicht mehr die Post, sondern unser Bewerberportal oder daran angebundene digitale Wege. Bewerbungen auf den Weg zu bringen, ist somit eine Angelegenheit von wenigen Klicks geworden. Ich finde diese Veränderung zweischneidig. Ich erhalte wesentlich schneller Feedback zu ausgeschriebenen Stellen, eine größere Anzahl von Profilen und auch die Diversität der Profile erscheint mir größer. Auf der anderen Seite konsumiere ich wesentlich mehr Profile und muss wesentlich schneller Entscheidungen zu Bewerbern treffen. Es gibt Fälle, bei denen ich mir mehr Zeit wünsche, um die Menschen besser kennenzulernen.

Menschen willkommen heißen, aber nur per Video

Nach der Bewerbung kommt die Begrüßung im Unternehmen. Diese ist mir persönlich enorm wichtig. Wenn ein neuer Mitarbeiter ankommt, ist alles vorbereitet: Zugangskarte ist vorhanden, Hardware ist eingerichtet, Blumen stehen parat – aber Moment: Wenn der neue Mitarbeiter gar nicht ins Office kommt, wie kriegt er dann seine Zugangskarte? Wie lange überlebt so ein Blumenstrauß eigentlich? Und wie schüttelt man jemandem die Hand zur Begrüßung, wenn man ihn nur auf einem Bild-

schirm sieht? Hier war größeres Umdenken gefragt. Für die Logistik wurden Lösungen mit Kurieren gefunden und Unterschriftenprozesse wurden digitalisiert. Damit hatten wir das Gröbste erledigt. Am Ende geht es aber darum, den Menschen zu begrüßen. Das Einzige, das hier wirklich geholfen hat, war viel Zeit und selbst Mensch sein.

In dieser für viele Menschen von Unsicherheit geprägten Pandemie haben wir, mich eingeschlossen, am meisten in den Aufbau der neuen Beziehungen investiert und aktiv das Netzwerk in der Abteilung aufgebaut – was sonst eher nebenläufig passiert ist. Geholfen haben uns vor allem neue Rituale. So trinke ich mit meinen Teammanagern montags morgens als Erstes einen Kaffee; dabei ist es egal, wo wir zurzeit arbeiten. Unsere Teams hingegen beenden gemeinsam ihren Freitagnachmittag mit einem Teammeeting. Aber auch in unserer Chatplattform haben sich neue Gruppen gefunden, die gemeinsame Hobbys oder Interessen teilen, es wurden während der Lockdowns gemeinsame virtuelle Sportveranstaltungen oder Quiz Nights organisiert und die gute alte Weihnachtsfeier hat einen neuen Anstrich bekommen, indem wir einen virtuellen Breakout Room gemeinsam bestanden haben oder ein virtuelles Kochevent erleben durften.

Einfach war es jedoch nicht, dort hinzukommen. Wo ich zuvor im Office die neuen Mitarbeiter begrüßen, dem Team vorstellen und bei den ersten Schritten begleiten konnte, so musste dies nun über virtuelle Meetings geschehen. Anstatt eines ganzen Tages an der Seite des neuen Mitarbeiters sollte nun ein einstündiges Begrüßungsmeeting reichen sowie die Weiterleitung sämtlicher bereits für die Teams bestehender (Serien-)Termine. Während vorher Gespräche natürlich entstanden sind, mussten sie nun aktiv initiiert werden. Kleinere Startschwierigkeiten, die immer entstehen, waren so schwieriger zu erkennen und zu klären. Dadurch entstanden neue Situationen, auf die ich noch keine Antwort hatte. Hier haben mir ein offener Umgang und auch das Eingestehen der eigenen Schwierigkeiten mit der neuen Situation geholfen, gemeinsam mit unseren „Neuen" den Start im Unternehmen erfolgreich zu meistern. Heute habe ich einen Werkzeugkoffer voll Informationspaketen, Anleitungen, neuen Meetingstrukturen und kleinen Workarounds, die mir helfen, die Startphase mit neuen Mitarbeitern erfolgreich zu gestalten. Ich bin allerdings auch dankbar, dass wir nach zwei Jahren wieder die Möglichkeit haben, unsere neuen Mitarbeiter persönlich zu begrüßen.

Apropos Video, man sieht nur, was man gezeigt bekommt

Ein ganz eigenes Thema: Eine aktivierte Kamera zeigt einem das Gesicht des Gesprächspartners und zumindest einen Teil der nonverbalen Kommunikation. Für mich stellte am Anfang die Umstellung auf Videokonferenzen eine Herausforderung dar, da Körpersprache und Mimik nur eingeschränkt zur Verfügung standen und Gespräche dadurch für mich anstrengend waren. Hinzu kommt ein erheblicher Teil von deaktivierten Kameras. War es am Beginn der Umstellung häufig noch technisch bedingt, so haben sich über die Zeit verschiedenste Gründe ergeben, die Kamera nicht zu aktivieren: Morgens läuft vielleicht noch die Familie durch die Wohnung, ab Mittag kommen die Kinder von der Schule oder der Mitarbeiter sitzt vielleicht gar nicht in seiner Wohnung, sondern im Sommer auf seinem Balkon in der Sonne. Die Teilnehmer steuern, wer sie wann sieht. Anders als bei einem Office-Anwesenheitsmodell kann man sich nicht mehr drauf verlassen, dass man wirklich sieht, was mit dem Mitarbeiter vor sich geht, da dieser nur zeigt, was er zeigen möchte. Geht es dem Mitarbeiter gut, ist dies kein Problem. Es kann aber zu einem werden, wenn die Umstände im Homeoffice dem Mitarbeiter oder auch seinen Aufgaben schaden. Ist dies der Fall, muss ich mich darauf verlassen können, dass sich mir meine Mitarbeiter anvertrauen. Für mich war es daher wichtig, mit meinen Mitarbeitern ohne Zwang ein Umfeld zu erschaffen, in dem sie sich öffnen und ich regelmäßig ihre Gesichter zu sehen bekomme. Dies hat Zeit und Überzeugungsarbeit gekostet. Daher bin ich nun umso froher, dass meine Mitarbeiter ihre Kamera heute die meiste Zeit aktiviert haben und sich wohl damit fühlen, wenn ich als ihr Vorgesetzter sehe und erlebe, was im Homeoffice bei ihnen passiert.

Das Büro nicht mehr der Arbeitsmittelpunkt, mehr eine Option

Auch der tägliche Arbeitsort ist im Zuge der Veränderungen zu einem Verhandlungspunkt geworden. Vor Pandemie und aller damit verbundenen Veränderung war das Arbeiten im Büro für jedermann normal. Mobiles Arbeiten war etwas für den Außendienst und wurde vielleicht verwendet, um Notsituationen unserer Mitarbeiter zu lösen. Diese Perspektive hat sich von einem Tag auf den anderen vollständig verändert. Wir sind mit mehr oder weniger Gewalt in ein 100 % Homeoffice Setup gegangen, haben uns durch die Anfangshürden gekämpft und alles ans Laufen gebracht. Bis auf kleinere Reibungsverluste war die Umstellung erfolgreich und ist es noch. Nun enden die Coronamaßnahmen mehr und mehr, sodass eine anvisierte Rückkehr ins Office ansteht. Aber warum? Was ist der Mehrwert eines solchen Beschlusses? Ohne ein zu großes Fass aufzumachen, es gibt ihn, diesen Mehrwert. Er ist nicht hart messbar, aber vorhanden; genauso gibt es den Mehrwert, ein Homeoffice Setup zu haben. Der Punkt und die harte Arbeit liegt darin, den Sweetspot zu finden, unter dem das Unternehmen erfolgreich operiert und die Mitarbeiter flexibel genug mit ihrer Arbeit umgehen. Was das jeweils für die Unternehmen und die Mitarbeiter genau heißt, ist individuell zu ermitteln.

Eines lässt sich allerdings klar festhalten: Wer sich hier sperrig zeigt, zieht den Kürzeren. Beide Seiten der Gleichung brauchen eine Einigung und müssen aufeinander zu gehen.

Wenn Kind und Kegel mitarbeiten

Wenn Kind und Kegel plötzlich Bedürfnisse haben, ist man als Elternteil, womöglich noch alleinerziehend, schnell auf Probleme gestoßen. Wir haben Telefonkonferenzen erlebt mit Kleinkindern auf dem Schoß und Teenagern mit ausgewachsenem Wutausbruch. Das alles ist vor dem Hintergrund von DSGVO und Datenschutz eigentlich nicht möglich, aber Realität. Hier kann man eigentlich auch nur wieder Mensch sein, denn ein schreiendes Baby auf dem Schoß wird immer die Priorität haben, egal was gerade im Büro zu erledigen ist. Mich haben die vielen Stunden mit unserem Abteilungsnachwuchs dazu gebracht, die eine oder andere Priorität und Wichtigkeit zu hinterfragen. Den Kollegen habe ich hingegen bewusst mehr Freiheiten eingeräumt, ihre Terminpläne an die eigenen Bedürfnisse und die ihrer Familie anzupassen – zu allseitigem Vorteil. Für mich hieß dies auch, mehr Flexibilität in meinen Arbeitsalltag zu integrieren. Von einem Tag auf den anderen waren wir zu viert im Homeoffice und Homeschooling. Anstatt Führungskraft war ich plötzlich auch noch Aushilfslehrer, Koch und IT-Support. Mir haben klare Grenzen geholfen, durch

die ich mich zumindest einen Teil des Tages vom Rest der Familie abgezwackt habe, um meinen Job zu erledigen. Die Haushaltsführung, wie etwa den Einkauf oder die Planung der Mahlzeiten, haben uns digitale Assistenten erleichtert und dennoch war es für alle eine Herausforderung.

Bis zur Freiheit und noch viel weiter

Ein persönliches Fazit für die letzten zwei Jahre und der darin durchlebten Transformation ist gar nicht so einfach. Wir haben alle unter besonderen Bedingungen erlebt, wie schnell sich die eigene Arbeitswelt verändern kann und welche Freiheiten und auch neue Pflichten dies mit sich bringt. Flexibilität, schon immer gefordert von jedermann, wurde hier hart auf die Probe gestellt. Zu unserer aller Freude hat es funktioniert – Mitarbeiter brauchen keinen „Stallgeruch", um sich mit dem Unternehmen zu identifizieren, Recruiting funktioniert auch ohne Vor-Ort-Termine, Mitarbeiter im Unternehmen willkommen heißen ebenso, mit Familie und ihren Anforderungen kommen wir auch klar. Und doch wünschen wir uns immer wieder zurück ins Office, weil es doch manchmal einfacher ist, schöner ist oder nicht jeder Jeck gleich ist. Auch der Abend mit den Kollegen im Biergarten hat uns weitergebracht. Aber was bleibt dann von all dem übrig? Die Freiheit, selbst zu entscheiden, wo ich arbeite, und die eigene Verantwortung, meinen Job bestmöglich zu erledigen, sind für mich die wesentlichen Dinge. Denn mit beidem kann ich Leben und Arbeiten, genauso wie meine Mitarbeiter.

Christian Linck
christian.linck@dkv-mobility.com

Christian Linck arbeitet seit über 20 Jahren in der Softwareentwicklung in verschiedenen Branchen. Nach 10 Jahren als Softwareentwickler hat er den Wechsel zum Coaching und zur Personalführung gemacht und hilft aktuell vier Produktentwicklungsteams, die Zukunft der Mobilität neu zu gestalten.

Ein Ziel, verschiedene Ansätze: Performance-Tuning in der Praxis

Dani Schnider, Callista und Martin Berger, Trivadis – Part of Accenture

Dani und Martin haben über Jahre hinweg gemeinsam Performanceprobleme eines Kunden analysiert und behoben. Bei dieser Arbeit haben sie erlebt, wie die gemeinsamen Ziele über unterschiedliche Wege erreicht wurden. Einige der Gemeinsamkeiten und Unterschiede, deren Hintergründe und Ansätze erklären sie hier.

Mehr als zwei Jahre lang haben wir zusammen mit weiteren Kollegen bei einem Kunden im Team Performance-Analysen und -Optimierungen auf verschiedenen Oracle-Datenbanken und Applikationstypen durchgeführt. Die Besonderheit dabei war, dass das Team zeitlich versetzt gearbeitet hat: Der Kunde hatte eine umfassende Abdeckung, jedes Team-Mitglied aber durchaus wochenlang keinen Kontakt zu diesem Projekt.

Dabei ist uns recht schnell klar geworden, dass wir konkrete Probleme oft nicht allein lösen können, sondern sie durch zeitliche oder organisatorische Limitierungen an Kollegen übergeben müssen. Dadurch ist uns aufgefallen, dass jedes Teammitglied, obwohl alle erfahrene Spezialisten mit langjähriger Erfahrung sind, doch jeweils seine eigene Vorgehensweise, Ansätze und Methoden hat.

Wir mussten also bei den Analysen nicht nur auf die konkreten Probleme eingehen, sondern auch unsere jeweiligen Methoden, konkrete Analysen und Ergebnisse austauschen. In diesem Austausch konnten wir alle voneinander lernen, unser Wissen und auch unsere eigene Toolbox erweitern. Über einige der Themenbereiche haben wir unsere Erfahrungen ausgetauscht.

Kommunikation mit den Kunden

Dani: Unsere Kollegen vom 2nd Level Support haben eine Liste von Standardfragen, die sie in jedem Ticket stellen, wenn es um Performanceprobleme geht *(siehe Abbildung 1)*. Was machst du jeweils mit den Antworten auf diese Fragen, Martin?

Martin: Einerseits ist es toll vom 2nd Level, die Fragen zu stellen, um den Kunden zu zeigen, dass jemand am Ticket arbeitet. Es gibt für den Kunden nichts Schlimmeres, als wenn er das Gefühl hat, er habe ein für ihn wichtiges Problem, aber niemand kümmert sich darum. Auf der anderen Seite werden gleich einige Details, die für unsere Analyse wichtig sein können, abgefragt. Viele sind generell gültig, andere können kundenspezifisch sein. Eine generelle Liste an Einstiegsfragen ist sicher sinnvoll, um eine Basis zu schaffen.

Dani: Ich muss immer schmunzeln bei der Frage 5: „Was habt ihr geändert". Die Antwort ist immer, sie hätten nichts geändert. Die Frage 6 nach den Statistiken finde ich gut, aber meiner Erfahrung nach sind veraltete oder unvollständige Statistiken heute nur noch selten die Ursache für Performanceprobleme. Bei den meisten Kunden werden die Statistiken mit dem Default-Job berechnet und sind somit meistens aktuell. Aber ein kurzer Check, wann die Statistiken das letzte Mal aktualisiert werden, ist sicher gut.

Martin: … und dann im Ticket dokumentieren, dass wir das geprüft haben. Überhaupt ist es wichtig, möglichst gut zu dokumentieren, was wir untersucht haben. Auch, und besonders, Fehlschläge und Sackgassen. Das erspart Kollegen, diese Arbeit nochmals zu machen.

Dani: Apropos Kommunikation – ich stelle mir oft die Frage: „Wie sag ich's dem Kunden?" Bei einigen Tickets wollte ich am liebsten schreiben: „Eure Applikation ist Schrott" – aber das kommt nicht so gut an. Wie sage ich einem Entwickler, dass sein Datenmodell oder seine mangelnden SQL-Kenntnisse der Hauptgrund für die Performanceprobleme sind?

Martin: Dieses Ticket-System hat den Vorteil, dass wir zwei Ebenen haben: Aus unserer Sicht können und sollen wir recht klar schreiben, was wir sehen, und unsere technischen Aktivitäten und Ergebnisse dokumentieren.

Dem Kunden gegenüber müssen wir natürlich eine andere Sprache wählen: Auf den jeweiligen Kunden angepasst und inhaltlich so detailliert wie notwendig. Wir kennen einige der „Stammkunden" gut und können unsere Sprache an ihre Fähigkeiten und Umgangsformen anpassen. Das Ziel ist also, freundlich, aber klar zu formulieren und zu zeigen, dass es nicht genügt, einfach ein paar Parameter auf der Datenbank zu ändern. Wichtig ist auch, mögliche Lösungswege oder alternative Implementierungsvarianten aufzuzeigen.

Identifikation des Performanceproblems

Dani: Jetzt sind wir schon bei den Lösungen. Aber die erste Herausforderung für uns ist ja oft, zuerst mal herauszufinden, wo genau die Ursache für ein Performanceproblem liegt. Wie können wir identifizieren, wo am meisten Zeit verloren geht?

Nützlich finde ich, dass der 2nd Level Support bei jedem Performance-Ticket jeweils gleich einen AWR-Report für die kritische Zeitperiode erstellt. Ich muss zwar ehrlicherweise gestehen, dass ich diese AWR-Reports nur brauche, um einen ersten Überblick zu bekommen. Für die Detailinformationen gehe ich dann relativ schnell auf die Datenbank. Wie siehst du das? Arbeitest du oft mit dem AWR-Report oder hast du da andere Varianten?

Martin: Ich gebe dir vollkommen recht: Ein AWR-Report ist immer nur ein Durchschnitt über einen gewissen Zeitraum zwischen Anfangs- und End-Snapshot. Er gibt uns Basisinformationen über das System wie Datenbankversion, Host- oder Clustername. Auch Durch-

1. **How long it is since you've started experiencing the performance degradation?**
2. **Does performance degradation affect the entire application or only one sql statement?**
3. **Please provide timestamp when query worked faster/slower and if it's the case, a data base where it works fine.**
4. **Please make sure to provide database name, username and a sql id or sql statement as well as a timestamp of when it ran.**
5. **Have you recently changed something on the application side or at the data level?**
6. **Have you gathered statistics recently? - If not, please run gather stats on schema and let us know how the query performs afterwards**

Abbildung 1: Standardfragen in jedem Performanceticket (Quelle: Dani Schnider)

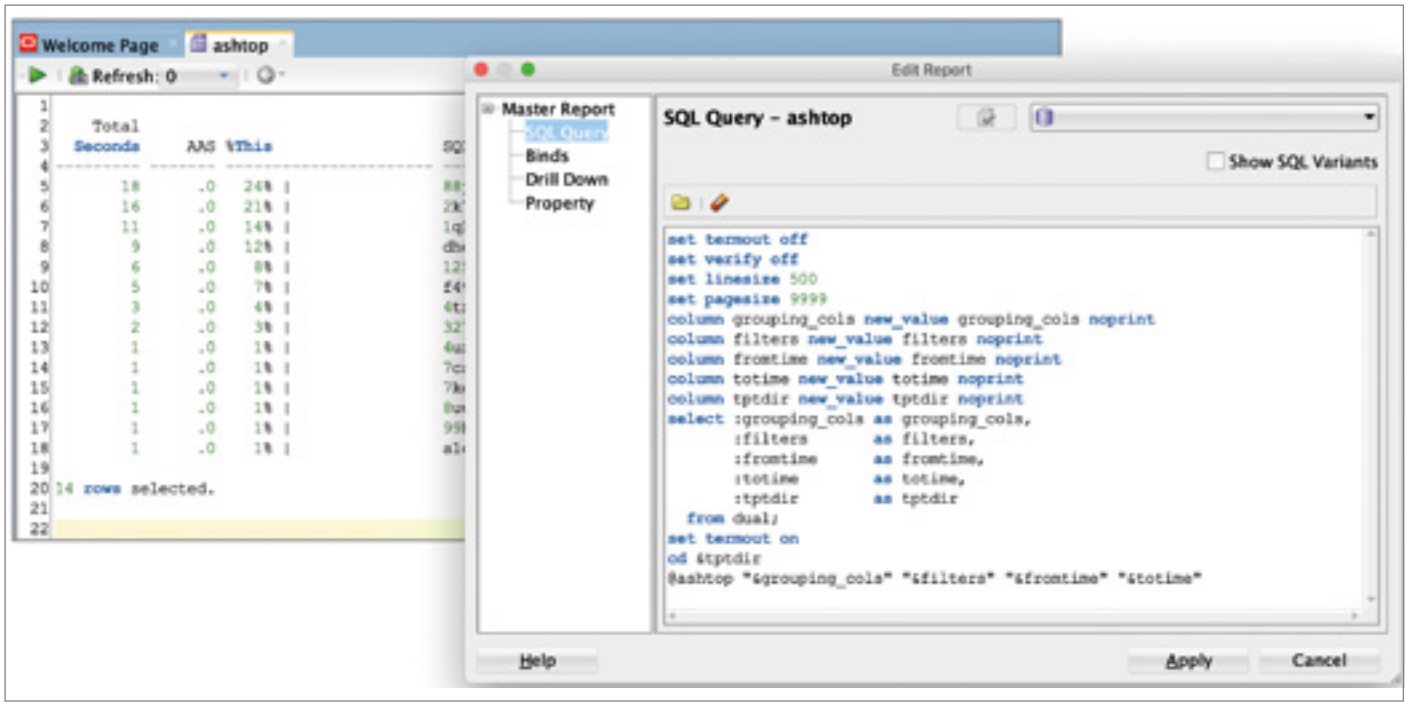


Abbildung 2: Script ashtop.sql als User Defined Report in SQL Developer (Quelle: Dani Schnider)

schnitte über den Report-Zeitraum wie die Gesamtlast oder die diversen TOP-Reports. Ob das konkrete Problem unter diesen zu finden ist, sagt uns der AWR-Report allein noch nicht.

Dani: Heißt das, dass du neben dem AWR-Report noch weitere Informationsquellen brauchst?

Martin: Es gibt Tools, die mehr Informationen als AWR sammeln, beispielsweise SQLdb360 *[1]*. Hier gibt es Hunderte von Tests, die Detailinformationen sammeln, beispielsweise die Anzahl der Archive-Logs auf Stundenbasis, gruppiert über Tage. Damit kann man die Transaktionslast gut sehen. Ein großer Vorteil dieses Tools ist, dass bei jedem einzelnen Check auch das SQL angegeben ist, mit dem er durchgeführt wurde. Wer also darauf aufbauend noch tiefer analysieren will, hat einen sehr guten Startpunkt.

Welche Werkzeuge verwendest Du, um weiter ins Detail zu gehen?

Dani: Gelegentlich verwende ich die Troubleshooting Scripts von Tanel Põder *[2]*. Ich kenne längst nicht alle Scripts aus seiner Library, aber sehr häufig verwende ich ashtop.sql beziehungsweise dashtop.sql. Damit können Informationen aus der Active Session History (ASH) angezeigt und nach unterschiedlichen Kriterien gruppiert werden. Wenn ich schon weiß, welches SQL betroffen ist, dann kann ich mit dem Script die häufigsten Wait-Events für dieses SQL ermitteln. Oder ich kann die langsamsten SQL-Statements ermitteln, die im fraglichen Zeitraum unter einem bestimmten Usernamen ausgeführt wurden. Viele dieser Informationen findet man natürlich auch im AWR-Report, aber hier habe ich mehr Möglichkeiten, meine Fragenstellungen über Parameter genauer einzuschränken.

Als typischer Entwickler arbeite ich in der Regel mit Tools wie SQL Developer oder Toad. Die Scripts von Tanel sind jedoch für SQL*Plus ausgelegt. Unser Kollege Philipp Salvisberg hat einmal einen Blog-Post darüber geschrieben, wie man SQL*Plus Scripts in SQL Developer integrieren kann *[3]*. Als Beispiel hat er genau mein „Lieblings-Script" ashtop.sql verwendet. Das habe ich in meiner Umgebung beim Kunden genauso nachgebaut für die Scripts von Tanel Põder, die ich regelmäßig verwende. Ich kann sie dann als „User Defined Reports" im SQL Developer nutzen *(siehe Abbildung 2)*.

Martin: Ich sehe, wir sind da etwas unterschiedlich unterwegs mit der Wahl unserer Werkzeuge. Aber bei all diesen Tools geht es ja in einem ersten Schritt meistens darum, das Performanceproblem, also die „problematische" SQL_ID, den PL/SQL- Block oder den Business-Prozess zu identifizieren.

Dani: Manchmal geht es sogar noch einfacher. Wenn bereits im Ticket steht, welches SQL das Performanceproblem verursacht, können wir die SQL_ID natürlich auch im Shared Pool (gv$sql) oder über die AWR History Views (dba_hist_sqltext) suchen *(siehe Listing 1)*.

Instrumentierung

Martin: Aber die Werkzeuge, die wir bis jetzt haben, helfen uns nicht direkt, um in einer Instanz das Problem von allen anderen zeitgleichen Nicht-Problem-Aktivitäten unterscheiden zu können. Das erfordert oft einiges an manueller Arbeit.

Ein erfolgreicher Ansatz dazu ist im Allgemeinen die Instrumentierung. Dabei wird ein konkreter Prozess oder Ablauf klar markiert und dadurch identifizier- und messbar gemacht. In Oracle ist das sehr einfach und sogar gratis möglich: Instrumentierung ist in allen Versionen und Editionen verfügbar, in vielen Tools schon fix integriert und direkt verwendbar: In der v$session, in SQL Trace, in AWR oder in Statspack. Der Enterprise Manager greift auch darauf zurück. Zusätzlich gibt es keinen Performance-Overhead, es muss nur programmatisch aktiviert werden. Das ist im Prinzip in allen Umgebungen möglich: sei es in PL/SQL, beliebigen Clients wie jdbc thin/thick, OCI, .NET, php. Sogar in der Cloud wird die Instrumentierung verwendet. Sie ist also aus Oracle-

Sicht die optimale Vorbereitung, um Probleme aller Art identifizieren zu können.

Dani: Als Entwickler versuche ich auch, Informationen über den Ablauf von Prozeduren und Funktionen zu protokollieren, beispielsweise mit dem Ora-OpenSource PL/SQL Logger *[4]*. Das funktioniert, wenn ich mit PL/SQL oder einer anderen Programmiersprache arbeite. Aber viele Tools haben keine oder proprietäre Logging-Mechanismen. Unser Kunde verwendet Informatica PowerCenter als ETL-Tool. Soviel ich weiß, haben wir keinen Zugriff auf die Logfiles von Informatica? Das wäre vielleicht nützlich?

Martin: Informatica liefert von Haus aus nichts zur Instrumentation mit. Hier sind alle Beteiligten gefordert: Die DBAs, Developer und Architekten müssen kreativ sein. Du hast eine Logging-Funktion angesprochen. In dieser kann man am Beginn und am Ende jedes Prozesses die Instrumentierung zum Beispiel mit einem Trigger oder direkt im Package steuern.

Bei diesem Kunden haben wir einen anderen Weg gewählt: Informatica selbst bietet ein sogenanntes Transaction Environment an. Bei diesem können wir am Anfang der Transaktion ein zusätzliches SQL ausführen und den Workflow-Namen und die Workflow Run Instance – die konkrete Ausführung – als Module und Action definieren.

Damit ist es sehr leicht möglich, den gesamten Workflow, den der Kunde als Business-Prozess ansieht, einfach zu identifizieren.

Anzeigen von Execution Plans

Martin: Jetzt haben wir die Stelle in der Applikation identifiziert, die für die Performanceprobleme verantwortlich ist – entweder durch Instrumentierung oder mit einer der vorher diskutierten Methoden. Die nächste Frage wäre nun: Wie schaust du dir die Ausführungspläne jeweils an?

Dani: Meistens ganz klassisch mit dem Package *dbms_xplan.* Wenn ich das SQL direkt ausführen kann, verwende ich den Hint */*+ *gather_plan_statistics* */* und schaue mir dann mit *dbms_xplan.display_cursor* den ausgeführten Execution Plan an. Der Vorteil ist hier, dass ich nicht nur die geschätzte Kardinalität des Optimizers sehe (Spalte „E-Rows"), sondern auch die tatsächliche Anzahl von Rows, die in jeder Zeile ausgeführt wurden (Spalte „A-Rows"). Für Statements, die länger laufen und die ich nicht direkt ausführen kann, verwende ich *dbms_xplan.display_awr.*

Martin: Wichtig ist hier noch zu ergänzen, dass die Adaptive Plans auch angezeigt werden sollten – bei *dbms_xplan* mit dem Parameter *„format* => '+adaptive' ".

Wer nicht die Command-Line-Methode verwenden will, kann natürlich auch Enterprise Manager oder SQL Developer verwenden.

Dani: Ja genau. Wenn ich interaktiv am Testen bin, wähle ich oft auch die bequeme Lösung und zeige den Execution Plan mit „Autotrace" (F6) im SQL Developer an. Wie bei *dbms_xplan.display_cursor* wird auch hier der tatsächlich ausgeführte Plan angezeigt, leider jedoch standardmäßig nicht mit den Angaben zur Anzahl Rows pro Zeile. Die Konfiguration des SQL Developer kann aber so geändert werden, dass die zusätzliche Spalte LAST_OUTPUT_ROWS (entspricht „A-Rows" in *dbms_xplan*) ebenfalls angezeigt wird *(siehe Abbildung 3)*.

```
SELECT sql_id, sql_text
  FROM gv$sql
 WHERE UPPER(sql_text) LIKE '%MV_CURIOUS%';

SELECT sql_id, sql_text
  FROM dba_hist_sqltext
 WHERE UPPER(sql_text) LIKE '%MV_CURIOUS%';
```

Listing 1: Ermitteln von SQL_ID aus Shared Pool bzw. AWR History

```
select sql_full_plan_hash_value, sql_plan_hash_value,
       sql_plan_line_id, count(*) cnt
from gv$active_session_history ash
where ash.sql_id='gdtppzzfkmr1w'
and sample_time between '2022-01-25 02:27:26'
                    and '2022-01-31 02:27:26'
group by sql_full_plan_hash_value,
      sql_plan_hash_value, sql_plan_line_id
order by sql_full_plan_hash_value,
      sql_plan_hash_value, sql_plan_line_id;
```

Listing 2: Detailauswertung von Execution Plan aus Active Session History

```
SELECT stat.sql_id
     , stat.plan_hash_value
     , TO_CHAR(snap.end_interval_time, 'dd.mm.yyyy') exec_date
     , ROUND(SUM(elapsed_time_delta)/1000000) elapsed
     , ROUND(SUM(cpu_time_delta)/1000000) cputime
     , ROUND(SUM(iowait_delta)/1000000) iotime
     , SUM(stat.rows_processed_delta) num_rows
     , SUM(stat.executions_delta) execs
     , ROUND(SUM(elapsed_time_delta)
       / GREATEST(SUM(stat.executions_delta), 1)/1000000, 2) sec_per_
exec
  FROM dba_hist_sqlstat stat
  JOIN dba_hist_snapshot snap
    ON snap.dbid            = stat.dbid
   AND snap.instance_number = stat.instance_number
   AND snap.snap_id         = stat.snap_id
WHERE stat.sql_id           = '&sql_id'
   AND stat.elapsed_time_delta > 0
GROUP BY stat.sql_id
    , stat.plan_hash_value
    , TO_CHAR(snap.end_interval_time, 'dd.mm.yyyy')
ORDER BY TO_DATE(exec_date, 'dd.mm.yyyy');
```

Listing 3: Code Snippet für „SQL History" einer SQL_ID

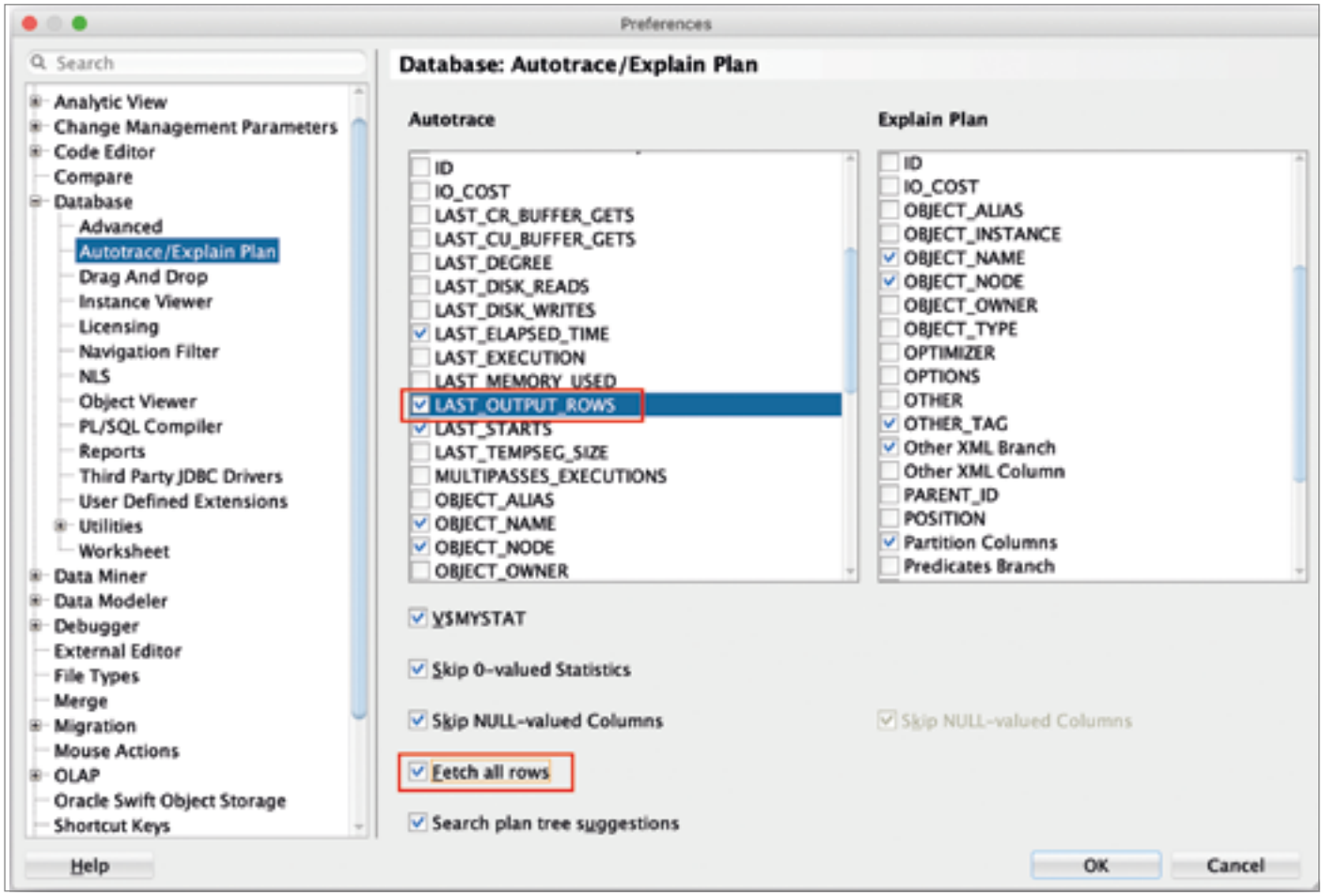


Abbildung 3: Empfohlene Einstellungen für Ausführungspläne im SQL Developer (Quelle: Dani Schnider)

Martin: Im SQL Developer erhalten wir mit der Autotrace-Funktion auch gleich ein Delta der Session-Statistiken. Diese zusätzlichen Informationen zeigen, was alles zwischen Beginn und Ende der Ausführung des SQL in der Session passiert ist. Leider ist „Fetch all rows" standardmäßig nicht aktiviert und nur die ersten 50 Zeilen werden gelesen und bewertet. Damit sich die Laufzeitstatistiken auf die ganze Ausführung beziehen, muss „Fetch all rows" in den Einstellungen ebenfalls aktiviert werden *(siehe Abbildung 3)*.

Doch jetzt noch eine generelle Frage: Du hast verschiedene Möglichkeiten erwähnt, wie der Execution Plan angezeigt wird, aber ich weiß immer noch nicht, wo genau im Plan die Zeit verloren gegangen ist.

Dani: Ja, das ist bei komplexen Plänen nicht immer einfach zu sehen. Wenn es sich um ein SQL handelt, das nicht gerade stundenlang läuft, verwende ich auch gerne SQL-Real-Time-Monitoring – eine meiner Meinung nach sehr coole Möglichkeit, um die wichtigsten Informationen zur Ausführung eines SQL-Statements anzuzeigen. Hier wird grafisch angezeigt, wie viel Zeit in jeder Zeile des Execution-Plans benötigt wurde.

Martin: Ich bin auch ein großer Fan von SQL-Real-Time-Monitoring, besonders der interaktive HTML-Report hat es mir angetan. Hier sind die meisten Informationen auf einen Blick verfügbar und auch in einem Gespräch leicht erklärbar.

Wenn wir das SQL nicht mit SQL-Monitoring aufzeichnen können, kommen wir trotzdem zu den meisten dieser Informationen. In der View gv$active_session_history gibt es eine Spalte sql_plan_line_id, die genau zur Zeilennummer im Execution Plan passt. Wenn wir die Informationen, wie in *Listing 2* gezeigt, aggregieren, sehen wir über die Information count(*), wo proportional am meisten Zeit verwendet wurde. Dies ist ein bisschen aufwendiger als der Report, aber wir kommen an dieselben Informationen heran.

Dani: Früher habe ich oft noch SQL Trace und TKPROF verwendet, da dort die Anzahl verarbeiteter Rows und die Ausführungszeiten von Parse-, Execute- und Fetch-Phase direkt angezeigt werden. Aber bei vielen Kunden ist das recht mühsam, da ich meistens keinen Zugriff auf OS-Ebene auf den Datenbankserver habe.

Martin: In Cloud-ähnlichen oder sehr restriktiven Umgebungen haben auch wir als DBAs oft keinen Zugriff auf das Betriebssystem beziehungsweise das Diagnostic Repository. Doch wir haben für diesen Zweck eigene Views: In V$DIAG_TRACE_FILE sehen wir eine Liste der Trace-Filenamen – ein Äquivalent zu ls oder dir.

In V$DIAG_TRACE_FILE_CONTENTS ist dann der Inhalt des jeweiligen Trace-Files sichtbar: Jede Zeile des Files ist eine Zeile der View, schön mit einer Zeilennummer sortiert.

In diesem Fall war ich einmal so faul und habe mir eine Lösung für den SQL Developer geschrieben: Einen kleinen Report, der mir den Trace-File-Namen angibt und in einem zweiten Schritt die Information als LOB ausgibt, das ich einfach dann lokal mit einem rechten Maus-Klick speichern kann *(siehe [5])*.

Warum haben wir unterschiedliche Pläne?

Dani: Oft haben wir es ja mit Situationen zu tun, in denen für das gleiche SQL unterschiedliche Ausführungspläne verwendet werden. Die Gründe können vielfältig sein ...

Martin: Ja, zum Beispiel neue Statistiken nach größeren Änderungen der Datenmengen beziehungsweise Datenverteilung. Oder es wurden neue Datenbankobjekte, zum Beispiel Indizes, angelegt ...

Dani: ... aber die Kunden antworten doch immer, sie hätten nichts geändert. Neue Features nach Datenbank-Upgrades oder Patches können auch ein Grund sein. Hier tue ich mich oft schwer herauszufinden, welches Feature genau verantwortlich dafür ist, dass ein Plan „kippt".

Martin: Ja, Gott sei Dank gibt es in den meisten Bereichen schon die Vorarbeit anderer Leute, beispielsweise das Tool Pathfinder *[6]*. Die Idee dahinter ist, alle Möglichkeiten des Optimizers einfach iterativ durchzuprobieren, um herauszufinden, welche Parameter-Einstellungen welchen konkreten Pfad erzwingen.

Es werden alle Optimizer-Parameter, relevanten Session-Parameter und _fix_control-Nummern durchprobiert, um daraus abzuleiten, welche Settings zu welchem konkreten Plan führen. Auf einer modernen Datenbankversion sind das über 2000 Tests. Je nach Parse- oder Execute-Zeit ist das nichts für Ungeduldige.

Dani: Oft hören wir die Aussage vom Kunden: „Bis gestern war die Abfrage schnell, heute dauert sie viel länger". Um solche Aussagen zu verifizieren – oder zu widerlegen –, verwende ich meistens eine Query (basierend auf der View dba_hist_sqlstat), die ich als „SQL History" in meinen Code Snippets im SQL Developer gespeichert habe *(siehe Listing 3)*. Damit kann ich rasch feststellen, ob und wann sich der Execution Plan in den letzten 30 Tagen geändert hat. So kann ich einschränken, ob wir es mit einem generellen Problem im SQL zu tun haben oder ob sich der Plan durch einen der genannten Gründe verändert hat.

Martin: Ich stelle fest, dass wir in vielen Punkten ähnlich vorgehen, obwohl wir dazu unterschiedliche Tools und Methoden verwenden.

Schlussendlich haben wir verschiedenste Methoden zur Hand, um unser gemeinsames Ziel zu erreichen: ein performantes System.

Quellen

[1] *https://github.com/sqldb360/sqldb360*
[2] *https://github.com/tanelpoder/tpt-oracle*
[3] *https://www.salvis.com/blog/2019/10/24/integrate-sqlplus-scripts-in-sql-developer/*
[4] *https://github.com/OraOpenSource/Logger*
[5] *https://github.com/berx/oracle_scripts/blob/master/getTracefile.xml*
[6] *https://mauro-pagano.com/2015/10/26/introducing-pathfinder-is-there-a-better-plan-for-my-sql*

Dani Schnider
dani.schnider@callista.ch

Martin Berger
m.a.berger@accenture.com

Quadratur des Kreises – Attribute Clustering, ein weithin unterschätztes Feature der Oracle-Datenbank – Teil 2

Randolf Eberle-Geist, Unabhängiger Berater

Im ersten Teil des Artikels wurde anhand eines einfachen Beispiels erläutert, um was es beim Thema „Clustering“ geht und wie die physische Organisation von Daten in einer Tabelle die Effizienz von Zugriffen auf die Daten beeinflussen kann. Im ersten Teil wurde gezeigt, wie man die Beispieltabelle als „Index Organized Table“ (IOT) in Oracle anlegen konnte – hier nun im zweiten Teil wird auf das Feature „Attribute Clustering“ eingegangen, mit dem Ähnliches erreicht werden kann.

Optimierung mittels „Attribute Clustering“

Seit Oracle 12c hat Oracle auch noch ein weiteres Feature bezüglich „Clustering von Daten“ im Angebot – „Attribute Clustering“ – es bietet die Möglichkeit, die physische Organisation der Daten einer „Heap“-Tabelle eben doch zu beeinflussen. Daher auch der Titel des Artikels „Quadratur des Kreises“ – denn „Attribute Clustering“ ermöglicht es, für die eigentlich „organisationslose“ „Heap“-Tabelle doch eine Vorgabe zu machen, wie die Daten physisch organisiert sein sollen.

Wie soll das gehen, denn definitionsgemäß heißt doch eine „Heap“-Tabelle genau deswegen so, weil es keine Vorgaben gibt, wo Daten darin abzulegen sind? Das „Attribute Clustering“ bei Oracle wird daher nur bei bestimmten Operationen auf der „Heap-Tabelle“ berücksichtigt – immer dann nämlich, wenn direkt neue Blöcke mit Daten geschrieben werden. Das ist zum Beispiel bei dem in Teil 1 erwähnten „APPEND“-Modus des Inserts der Fall oder auch bei einer „CREATE TABLE AS SELECT“-Operation oder auch bei einem ALTER TABLE MOVE, wenn die Tabelle reorganisiert und dabei komplett neu geschrieben wird.

```
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)
)
CLUSTERING BY LINEAR ORDER (ticker_code, trade_date)
;
```

Listing 1: Erzeugung der Beispieltabelle mit „Attribute Clustering"-

Währenddessen ignoriert die Datenbank das „Attribute Clustering" bei konventionellem DML – wird also ein normales Insert in die „Heap"-Tabelle gemacht, gelten auch die in Teil 1 beschriebenen, normalen Regeln – dort wo Platz ist, wird die Zeile hingeschrieben, ungeachtet irgendwelcher Vorgaben durch „Attribute Clustering", wie die Daten organisiert sein sollen.

Um also das „Attribute Clustering" effektiv auf eine „Heap-Tabelle" anzuwenden, müssen die Daten per „APPEND"-Modus eingefügt oder per „CREATE TABLE AS SELECT"-Operation erzeugt werden – dies ist häufig in Data-Warehouse- Umgebungen sowieso der Fall, wenn Daten im Rahmen von ETL-Transformationen in Batches oder per Exchange-Partition-Operationen verarbeitet werden.

Ist dies nicht der Fall, müssen neu erzeugte Daten regelmäßig per ALTER TABLE MOVE reorganisiert werden – auch dies ist in neueren Versionen von Oracle in vielen Fällen inzwischen per ONLINE-Operation möglich, also auch, während DML auf der Tabelle aktiv ist. Idealerweise kann man das mit einer geeigneten Partitionierung (separate Lizenz bei Oracle notwendig) verknüpfen, sodass alte Daten, die bereits reorganisiert wurden, nicht mehr berücksichtigt werden müssen.

Eigenschaften und Auswirkungen des neuen „Attribute Clustering"-Features

„Attribute Clustering" bedeutet, dass ich der Datenbank per Metadatendefinition mitteile, dass die Daten in einer „Heap"-Tabelle gemäß eines oder mehrerer Kriterien organisiert sein sollen – Daten mit gleicher Ausprägung dieser Kriterien also physisch zusammenhängend in der Tabelle abgelegt werden, was beim lesenden Zugriff bedeutet, dass nur auf so viele Blöcke zugegriffen werden muss, wie diese zusammenhängend gespeicherte Daten allokieren. Beim Einsatz von Exadata „Storage-Indizes" im Rahmen der sogenannten „Smart Scans" (Full Table Scan wird an die Storage Cells von Exadata ausgelagert), InMemory-Column-Store-basierten Full Table Scans oder Zonemaps (leider auch nur in Exadata-Umgebungen aus lizenztechnischen Gründen verfügbar) gilt dies sogar für „Full Table Scans", also auch diese können durch „Attribute Clustering" signifikant beschleunigt/optimiert werden, da nur die relevanten Bereiche der Tabelle verarbeitet werden und der Rest „ignoriert"/übersprungen wird – ansonsten eben insbesondere bei indexbasierten Zugriffsmustern.

Wie würde also nun in dem konkreten Beispiel die Verwendung von „Attribute Clustering" aussehen? Das „CREATE TABLE"-Kommando zum Erzeugen einer entsprechenden „Heap"-Tabelle mit „Attribute Clustering" könnte so aussehen *(siehe Listing 1)*.

Anstelle der „ORGANIZATION INDEX"-Option ist nun also die „CLUSTERING BY"-Angabe verwendet worden, die der Datenbank mitteilt, dass die Daten in der „Heap"-Tabelle gemäß der angegebenen Kriterien zusammenhängend physisch abgelegt werden sollen. Dies kann auch nachträglich für bereits existierende Tabellen mittels ALTER TABLE definiert oder auch nachträglich wieder entfernt und durch eine andere Clustering-Definition ersetzt werden.

LINEAR ORDER ist der Standardfall und bedeutet, dass bei Angabe von mehreren Kriterien – hier TICKER_CODE und TRADE_DATE – die Daten zuerst primär nach TICKER_CODE zusammenhängend gespeichert werden und, falls es mehrere Zeilen mit dem gleichen TICKER_CODE gibt, dann gemäß des TRADE_DATE zusammenhängend sind. Das heißt, dass diese Art des Clusterings nur für Abfragen von Vorteil ist, die primär nach dem TICKER_CODE abfragen und eventuell noch zusätzlich nach dem TRADE_DATE, aber nicht für Abfragen, die nur nach dem TRADE_DATE abfragen, denn das primäre Kriterium ist nun mal der TICKER_CODE – die Daten sind eben nicht zusammenhängend nach TRADE_DATE abgelegt, sondern nur bei gleichem TICKER_CODE innerhalb dieser Menge nach TRADE_DATE zusammenhängend.

Diese Art des Clusterings von „Heap"-Tabellen könnte ich auch manuell umsetzen – also ohne die „CLUSTERING BY"-Definition – wenn ich die Tabelle per CREATE TABLE AS SELECT … ORDER BY oder INSERT /*+ APPEND */ INTO TABLE SELECT … ORDER BY befüllen würde. Ich könnte also auch diesen Effekt in der „Standard Edition" von Oracle ohne das „Attribute-Clustering"-Feature erreichen, wenn ich den Aufwand dafür in Kauf nehme, die Tabelle entsprechend physisch sortiert abzulegen.

Wie bereits beschrieben, wird diese Angabe jedoch nur bei speziellen Operationen auf der „Heap"-Tabelle von Oracle berücksichtigt. Verwende ich den gleichen PL/SQL- Block wie bisher zum Befüllen der Tabelle, um das Hinzufügen von täglichen Kursständen zu simulieren, kommen dort konventionelle INSERT-Befehle zum Einsatz – und nicht die Spezialform INSERT APPEND – sodass das Clustering eben nicht angewendet werden wird und de facto erst mal keine Auswirkung hat. Erst wenn ich zum Beispiel ein ALTER TABLE MOVE auf die befüllte Tabelle durchführe, werden dabei die Clustering-Definition berücksichtigt und die Daten bei der Reorganisation der Tabelle entsprechend abgelegt – ein einfaches, seriell ausgeführtes SELECT * FROM STOCK_HISTORY zeigt die Daten in der nun geänderten physischen Reihenfolge *(siehe Listing 2)*.

Wie man jetzt sehen kann, sind die Daten nach dem Befüllen per PL/SQL-Block und Durchführen von ALTER TABLE MOVE tatsächlich physisch anders angeordnet –

```
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_000                    20000102 00:00:00           0            0          0          0
COMPANY_000                    20000103 00:00:00           0            0          0          0
COMPANY_000                    20000104 00:00:00           0            0          0          0
COMPANY_000                    20000105 00:00:00           0            0          0          0
.
.
.
COMPANY_000                    20020922 00:00:00           0            0          0          0
COMPANY_000                    20020923 00:00:00           0            0          0          0
COMPANY_000                    20020924 00:00:00           0            0          0          0
COMPANY_000                    20020925 00:00:00           0            0          0          0
COMPANY_000                    20020926 00:00:00           0            0          0          0
COMPANY_001                    20000101 00:00:00           1            1          1          1
COMPANY_001                    20000102 00:00:00           2            2          2          2
COMPANY_001                    20000103 00:00:00           3            3          3          3
COMPANY_001                    20000104 00:00:00           4            4          4          4
COMPANY_001                    20000105 00:00:00           5            5          5          5
.
.
.
```

Listing 2: Geänderte physische Speicherung der Zeilen in der Beispieltabelle aufgrund der „Attribute-Clustering“-Definition

```
SQL> 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 |    11   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| STOCK_HISTORY    |   732 | 29280 |    11   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | STOCK_HISTORY_PK |   732 |       |     6   (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
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         16  consistent gets
          0  physical reads
        132  redo size
      30304  bytes sent via SQL*Net to client
        382  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 der Abfrage über zwei Jahre eines Aktienkurses auf der per „Attribute Clustering“ anders organisierten „Heap“-Tabelle

```
REM Create the SALES_SOURCE table
REM This will provide us with a consistent dataset
REM for any fact tables we choose to create later on
--
CREATE TABLE sales_source
  (
    order_id          NUMBER(20)    NOT NULL ,
    order_item_number NUMBER(3)     NOT NULL ,
    sale_date         DATE          NOT NULL ,
    delivered         DATE          ,
    sale_agent        VARCHAR2(100) NOT NULL ,
    product_id        NUMBER(10)    NOT NULL ,
    amount            NUMBER(10,2)  NOT NULL ,
    quantity          NUMBER(5)     NOT NULL ,
    location_id       NUMBER(20)    NOT NULL ,
    warehouse         VARCHAR2(100) NOT NULL
  )
/

REM     Create the LOCATIONS table
CREATE TABLE locations
  (
    location_id NUMBER(20) ,
    state       VARCHAR2(100) NOT NULL ,
    county      VARCHAR2(100) NOT NULL ,
    description VARCHAR2(1000) NOT NULL ,
    PRIMARY KEY (location_id)
  )
/

REM     Drop the PRODUCTS table
DROP TABLE products
/

REM     Create the PRODUCTS table
CREATE TABLE products
  (
    product_id          NUMBER(20) ,
    product_name        VARCHAR2(20) ,
    product_description VARCHAR2(100) ,
    PRIMARY KEY(product_id)
  )
/
```

Listing 4: Erzeugung der Beispieltabellen für Demonstration „Interleaved Order Attribute Clustering“

zuerst kommen alle TRADE_DATEs zum gleichen TICKER_CODE, dann folgt der nächste TICKER_CODE und dafür wieder alle TRADE_DATEs und so weiter.

Führe man nun also wieder die gleiche Abfrage wie bisher aus, die die Historie von zwei Jahren für einen bestimmten TICKER_CODE selektiert, sind die gesuchten Daten in der Tabelle zusammenhängend abgelegt und nicht jeweils 1000 Zeilen voneinander entfernt – entsprechend wird der Zugriff per Index immer wieder in die gleichen Tabellenblöcke springen, in denen die gesuchten Zeilen zusammenhängend gespeichert sind, und entsprechend weniger logisches I/O – und als Konsequenz daraus bei größeren Tabellen auch weniger physisches I/O ist notwendig, um die 731 Zeilen zu selektieren *(siehe Listing 3)*.

Ähnlich wie beim „Index Organized Table“ habe ich also diese Art der Abfrage bezüglich des logischen und potenziell auch physischen I/Os um viele Faktoren effizienter gemacht – es gibt zwar immer noch einen Tabellenzugriff im Ausführungsplan (TABLE ACCESS BY INDEX ROWID), aber es wird nur auf acht Blöcke der Tabelle dafür zugegriffen, in denen die gesuchten Zeilen zusammenhängend abgelegt sind, anstatt für jede gesuchte Zeile in einen anderen Block der Tabelle zu springen, sodass in Summe nur 16 logische I/Os benötigt werden, um die Abfrage auszuführen.

```
insert into products values (1,'DECKING','Decking Description');
insert into products values (2,'GARDENLIGHTING','Gardenlighting Description');
insert into products values (3,'GNOME','Gnome Description');
insert into products values (4,'LAMP','Lamp Description');
.
.
.
insert into locations values (1,'Alabama','Autauga County','Alabama Autauga County Description');
insert into locations values (2,'Alabama','Baldwin County','Alabama Baldwin County Description');
insert into locations values (3,'Alabama','Barbour County','Alabama Barbour County Description');
insert into locations values (4,'Alabama','Bibb County','Alabama Bibb County Description');
insert into locations values (5,'Alabama','Blount County','Alabama Blount County Description');
insert into locations values (6,'Alabama','Bullock County','Alabama Bullock County Description');
.
.
.
commit;

--
REM Gather statistics for our dimension tables
--
EXECUTE dbms_stats.gather_table_stats(ownname=>NULL,tabname=>'locations');
EXECUTE dbms_stats.gather_table_stats(ownname=>NULL,tabname=>'products');
```

Listing 5: Befüllung der Dimensionstabellen

```
CREATE sequence nseq CACHE 10000
/

--
REM Utility procedure to fill SALES_SOURCE table
--

CREATE OR REPLACE
PROCEDURE filltab(
    p_start_date DATE,
    p_number_of_rows NUMBER,
    p_number_of_days NUMBER)
AS
TYPE sa_tab_type
IS
  TABLE OF sales_source.sale_agent%TYPE INDEX BY BINARY_INTEGER;
  sa_tab sa_tab_type;
TYPE wh_type
IS
  TABLE OF sales_source.warehouse%TYPE INDEX BY BINARY_INTEGER;
  wh_tab          wh_type;
  sale_date       DATE;
  num_order_items NUMBER(2);
  sa              sales_source.sale_agent%TYPE;
  product         sales_source.product_id%TYPE;
  location        sales_source.location_id%TYPE;
  wh              sales_source.warehouse%TYPE;
  order_id        sales_source.order_id%TYPE;
  num_products    NUMBER(5);
  num_locations   NUMBER(5);
  max_order_items NUMBER(3)  := 20;
  num_inserted    NUMBER(10) := 0;
  loop_count      NUMBER(10) := 0;
  counter         NUMBER(10);
  deliv_days      NUMBER(3);
BEGIN
  sa_tab(1)  := 'MARK';
  sa_tab(2)  := 'CLARE';
  sa_tab(3)  := 'ANDREW';
  sa_tab(4)  := 'LUCY';
  sa_tab(5)  := 'JENNY';
  sa_tab(6)  := 'JOHN';
  sa_tab(7)  := 'BRIAN';
  sa_tab(8)  := 'JANE';
  sa_tab(9)  := 'ED';
  sa_tab(10) := 'SIMON';
  sa_tab(11) := 'SALLY';
  wh_tab(1)  := 'ALBUQUERQUE';
  wh_tab(2)  := 'WINSTON SALEM';
  wh_tab(3)  := 'NEWPORT';
  wh_tab(4)  := 'BIRMINGHAM';
  wh_tab(5)  := 'OCOEE';
  wh_tab(6)  := 'PRINCETON';
  order_id   := nseq.nextval;
  sale_date  := p_start_date;
  SELECT COUNT(*) INTO num_products FROM products;
  SELECT COUNT(*) INTO num_locations FROM locations;
  LOOP
    num_order_items:= dbms_random.value(1,max_order_items+1);
    order_id       := nseq.nextval;
    sale_date      := p_start_date + dbms_random.value(0,floor(p_number_of_days+1));
    wh             := wh_tab(floor(dbms_random.value(1,7)));
    sa             := sa_tab(floor(dbms_random.value(1,12)));
    deliv_days     := dbms_random.value(2,30);
    INSERT INTO sales_source
    SELECT order_id ,
      rownum ,
      sale_date ,
      sale_date + deliv_days ,
      sa ,
      dbms_random.value(1,floor(num_products))
```

```
      dbms_random.value(1,2000) ,
      dbms_random.value(1,3) ,
      dbms_random.value(1,floor(num_locations)) ,
      wh
    FROM dual
      CONNECT BY rownum    <= num_order_items;
    num_inserted           := num_inserted + num_order_items;
    loop_count             := loop_count   + 1;
    IF mod(loop_count,1000) = 0 THEN
      COMMIT;
    END IF;
    EXIT WHEN num_inserted >= p_number_of_rows;
  END LOOP;
  COMMIT;
END;
/
show errors

--
REM Fill the SALES_SOURCE table with data for 2000 and 2009
REM This may take several minutes...
--

EXECUTE filltab(to_date('01-JAN-2000','DD-MON-YYYY'),1452090,364);
EXECUTE filltab(to_date('01-JAN-2009','DD-MON-YYYY'),500000,364);

--
REM Gather table statistics...
--
EXECUTE dbms_stats.gather_table_stats(ownname=>NULL,tabname=>'sales_source')
```

Listing 6: Vorbereiten der Befüllung der Faktentabelle

Wichtig ist dabei auch zu verstehen, dass man sich bei dieser grundlegenden Variante für ein primäres Clustering-Attribut entscheiden muss – ein Zugriff auf mehrere Zeilen per TRADE_DATE wäre bei einem primären Clustering nach TICKER_CODE eben nicht mehr so effizient, wie es das natürliche Clustering der Tabelle eigentlich ermöglichen würde. Es kommt also darauf an, welche Zugriffsmuster am wichtigsten/häufigsten/kritischsten sind.

Weitere Features von „Attribute Clustering" – „Interleaved Ordering"

Das „Attribute Clustering" von Oracle bietet über die gerade beschriebene, grundlegende Variante hinaus noch weitere Möglichkeiten. Der Standardfall, der auch mittels „Clustered Index"/„Index Organized Tables" (IOTs)/Oracle-Cluster-Strukturen/ORDER BY-Klausel abgebildet werden kann, sieht – wie gerade erklärt – ein primäres Kriterium (eine Spalte der Tabelle) vor, nachdem die Daten organisiert werden. Das heißt, beim Design muss man sich für ein primäres Kriterium entscheiden – nur Zugriffe über dieses primäre Kriterium werden optimal durch das zusammenhängende Ablegen der Daten gemäß dieses Kriteriums unterstützt – bei Oracle IOTs kann dies nur der führende Teil oder der gesamte Primärschlüssel sein.

Das „Attribute Clustering" für „Heap"-Tabellen unterstützt jedoch zum Beispiel auch eine sogenannte „Interleaved Order" – hier können innerhalb bestimmter Grenzen sogar mehrere voneinander unabhängige Kriterien angegeben werden, nach denen die Daten organisiert werden sollen. Möglich macht das Oracle intern durch einen sogenannten „Z-Ordering"-Algorithmus. Haben diese unterschiedlichen Kriterien nicht zu viele verschiedene Ausprägungen, funktioniert das recht gut. Der Zugriff pro Kriterium ist dann unter Umständen nicht ganz so optimal wie bei der „Linear Order", bei der es nur ein primäres Kriterium gibt, dafür kann ich den Zugriff auf die Tabelle aber eben für mehrere, voneinander unabhängige Zugriffswege optimieren – diese Möglichkeit gibt es bei den anderen, alternativen Speicherformen so nicht.

Das funktioniert bei dem bisherigen Beispiel oben nicht – dafür gibt es zu viele Ausprägungen sowohl von TICKER_CODE als auch von TRADE_DATE (1000 mal 1000) in der Tabelle STOCK_HISTORY beziehungsweise die Kombination aus TICKER_CODE und TRADE_DATE ist eindeutig – man kann also nicht sowohl für den Zugriff per TICKER_CODE als auch per TRADE_DATE unabhängig voneinander optimieren. Das ist mit „Attribute Clustering" nur möglich, wenn die Anzahl der Kombinationen der Werte der unterschiedlichen Attribute deutlich unter der Anzahl der Zeilen der Tabelle liegt.

Daher hier ein anderes Beispiel, bei dem dies funktionieren kann.

Hier haben wir eine Faktentabelle sowie zwei Dimensionstabellen, also ein klassisches Star-Schema *(siehe Listing 4)*.

In die Dimensionen werden entsprechende Daten eingefügt *(siehe Listing 5)*.

```
REM Create the SALES fact table
REM This table will not have attribute clustering
REM or zone maps. We will use it to compare with
REM an attribute clustered table.
--
CREATE TABLE sales
AS
SELECT * FROM sales_source
WHERE 1 = -1
/

--
REM Create a SALES_AC fact table
REM The data will be the same as SALES
REM but it will be used to demonstrate
REM attribute clustering
REM in comparison to the standard SALES table.
--
CREATE TABLE sales_ac
AS
SELECT * FROM sales_source
WHERE 1 = -1
/

--
REM Here we enable interleaved ordered attribute clustering
--
ALTER TABLE sales_ac
ADD CLUSTERING BY INTERLEAVED ORDER (location_id, product_id)
WITHOUT MATERIALIZED ZONEMAP
/

set timing on
--
REM Insert data into standard table
--
INSERT /*+ APPEND */ INTO sales SELECT * FROM sales_source
/
--
REM Observe that insert plan is a simple insert
--
SELECT * FROM TABLE(dbms_xplan.display_cursor)
/
COMMIT
/

--
REM Insert data into attribute clustered table.
REM We must use a direct path operation to make
REM use of attribute clustering.
REM In real systems we will probably insert in
REM multiple batches: each batch of inserts will be
REM ordered appropriately. Later on,
REM if we want to re-order all rows into
REM tightly grouped zones we can, for example, use
REM partitioning and MOVE PARTITION to do this.
REM
REM Increased elapsed time is likely due
REM to the sort that is transparently performed to cluster
REM the data as it is inserted into the SALES_AC table.
--
INSERT /*+ APPEND */ INTO sales_ac SELECT * FROM sales_source
/
--
REM Observe the addition of "SORT ORDER BY" in the execution plan
--
SELECT * FROM TABLE(dbms_xplan.display_cursor)
/
COMMIT
/

set timing off

REM Gather table statistics
EXECUTE dbms_stats.gather_table_stats(ownname=>NULL,tabname=>'sales')
EXECUTE dbms_stats.gather_table_stats(ownname=>NULL,tabname=>'sales_ac')
```

Listing 7: Befüllung der Faktentabelle – zwei Varianten ohne und mit „Attribute Clustering“

```
SQL> REM Create indexes on location id for the standard SALES
SQL> REM table and the attribute clustered SALES_AC table
SQL>
SQL> CREATE INDEX sales_loc_i ON sales (location_id, product_id)
  2  /

Index created.

SQL> CREATE INDEX sales_ac_loc_i ON sales_ac (location_id, product_id)
  2  /

Index created.

SQL> CREATE INDEX sales_prod_i ON sales (product_id)
  2  /

Index created.

SQL> CREATE INDEX sales_ac_prod_i ON sales_ac (product_id)
  2  /

Index created.

SQL> REM Observe the improved value of "Average Blocks Per Key"
SQL> REM for the attribute clustered table. This will
SQL> REM result in fewer consistend gets for table lookups from
SQL> REM index range scans.
SQL>
SQL> SELECT index_name, clustering_factor,avg_data_blocks_per_key
  2  FROM   user_indexes
  3  WHERE  index_name LIKE 'SALES%LOC%'
  4  ORDER BY index_name
  5  /

INDEX_NAME                     CLUSTERING_FACTOR AVG_DATA_BLOCKS_PER_KEY
------------------------------ ----------------- ------------------------
SALES_AC_LOC_I                            102182                        1
SALES_LOC_I                              1950703                       22

2 rows selected.
```

Listing 8: Erzeugen passender Indizes für Beispielabfragen auf den beiden Faktentabellen

Dabei gibt es nur 28 Produkte und etwas mehr als 3000 Lokationen, ausmultipliziert also ca. 90.000 Kombinationen.

Die Daten für die Faktentabelle werden wieder per PL/SQL-Prozedur erzeugt – dabei werden Daten für die Jahre 2000 und 2009 exemplarisch generiert. Pro Jahr entstehen ca. eine Million Zeilen, sodass die Faktentabelle für diese zwei Jahre dann ca. zwei Millionen Zeilen enthält *(siehe Listing 6)*.

Zur besseren Vergleichsmöglichkeit erzeugen wir zwei Kopien der eigentlichen Faktentabelle aus den Daten in der gerade befüllten Tabelle SALES_SOURCE, eine ohne „Attribute Clustering" (SALES), die andere mit (SALES_AC) *(siehe Listing 7)*.

Führt man den Beispielcode aus, kann man sehen, dass das Schreiben mittels INSERT APPEND in die Zieltabelle bei der SALES_AC-Tabelle messbar länger dauert, da im Ausführungsplan des INSERT eine entsprechende SORT-ORDER- BY-Operation enthalten ist, die das „Attribute Clustering" abbildet – das Feature kostet also mehr Zeit beim Schreiben der Daten. Es benötigt – wie schon zuvor erwähnt – diese Spezialoperation INSERT APPEND, damit das „Attribute Clustering" dabei berücksichtigt wird – ein konventionelles INSERT würde dies nicht tun, wie beim ersten Beispiel gezeigt.

Das „Attribute Clustering" wurde hier nun für die zwei Fremdschlüssel LOCATION_ID und PRODUCT_ID zu den beiden vorab erzeugten Dimensionen angelegt – und in diesem Fall als „Interleaved Order" – was bedeutet, dass gemäß beider Attribute unabhängig voneinander die Daten zusammenhängend in der Tabelle abgelegt werden. Es werden nun also Abfragen sowohl nur auf der LOCATION_ID als auch nur auf der PRODUCT_ID durch das „Attribute Clustering" optimiert – dies ist, wie bereits erwähnt, mit den anderen Methoden, wie zum Beispiel einer „Index Organized Table" so nicht möglich und ein Alleinstellungsmerkmal des „Attribute-Clustering"- Features.

Führt man nun entsprechende Abfragen auf den beiden Tabellen aus, die auf die genannten Attribute filtern, kann man den Unterschied zwischen den

```
SQL> SET AUTOTRACE ON
SQL>
SQL> REM Observe the IO differences for clustered table.
SQL>
SQL> set termout off
SQL>
SQL> REM Nonclustered
SQL>
SQL> SELECT SUM(amount)
  2  FROM   sales
  3  WHERE  location_id = 50
  4  /

SUM(AMOUNT)
-----------
  604281,39

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1930106867

-------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |             |     1 |     9 |   625   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE                     |             |     1 |     9 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| SALES       |   621 |  5589 |   625   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                 | SALES_LOC_I |   621 |       |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   3 - access("LOCATION_ID"=50)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        598  consistent gets
          0  physical reads
          0  redo size
        352  bytes sent via SQL*Net to client
        372  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL> REM With attribute clustering
SQL>
SQL> SELECT SUM(amount)
  2  FROM   sales_ac
  3  WHERE  location_id = 50
  4  /

SUM(AMOUNT)
-----------
  604281,39

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 311709933
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                |     1 |     9 |    37   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE                      |                |     1 |     9 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| SALES_AC       |   621 |  5589 |    37   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                  | SALES_AC_LOC_I |   621 |       |     4   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

   3 - access("LOCATION_ID"=50)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         24  consistent gets
          0  physical reads
          0  redo size
        352  bytes sent via SQL*Net to client
        372  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

Listing 9: AUTOTRACE-Ergebnisse von Abfragen per LOCATION_ID auf beiden Faktentabellen

```
SQL> REM In this case we have used an attribute cluster with Interleaved ordering
SQL> REM so we can use predicates on location_id or product_id (or both)
SQL>
SQL> REM Forcing index usage here because otherwise the optimizer would use
SQL> REM a full table scan on the table without clustering
SQL> REM and observe the reduced number of consistent
SQL> REM gets for the attribute cluster example.
SQL>
SQL> REM Nonclustered
SQL>
SQL> SELECT /*+ index(sales) */ SUM(amount)
  2  FROM   sales
  3  WHERE  product_id = 10
  4  /

SUM(AMOUNT)
-----------
 72522121,4

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 873694340

-----------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |              |     1 |     8 | 16640   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE                      |              |     1 |     8 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| SALES        | 72684 |   567K| 16640   (1)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                  | SALES_PROD_I | 72684 |       |   145   (1)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

```
   3 - access("PRODUCT_ID"=10)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
      16069  consistent gets
        144  physical reads
          0  redo size
        353  bytes sent via SQL*Net to client
        372  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL> REM With interleaved attribute clustering
SQL>
SQL> SELECT /*+ index(sales_ac) */ SUM(amount)
  2  FROM   sales_ac
  3  WHERE  product_id = 10
  4  /

SUM(AMOUNT)
-----------
 72522121,4

1 row selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 266287893

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                 |     1 |     8 |   796   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE                     |                 |     1 |     8 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| SALES_AC        | 72684 |   567K|   796   (1)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                 | SALES_AC_PROD_I | 72684 |       |   145   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

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

   3 - access("PRODUCT_ID"=10)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        795  consistent gets
          0  physical reads
          0  redo size
        353  bytes sent via SQL*Net to client
        372  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

Listing 10: AUTOTRACE-Ergebnisse von Abfragen per PRODUCT_ID auf beiden Faktentabellen

beiden Tabellen deutlich sehen in der Anzahl der notwendigen „logischen" I/Os. Zuerst werden die dafür benötigten Indizes erzeugt *(siehe Listing 8)*.

Dabei wird schon anhand der unterschiedlichen Kennzahl „Clustering Factor" der Indizes auf den beiden Tabellen ersichtlich, dass die Reihenfolge der Daten in der Tabelle mit „Attribute Clustering" deutlich besser der Sortierung des Index entspricht als bei der Tabelle ohne „Attribute Clustering" – der Wert ist ca. 100.000 bei der einen und fast zwei Millionen bei der anderen Tabelle. Je höher der Wert und je mehr er sich der Anzahl der Zeilen in der Tabelle annähert, desto schlechter stimmen Tabelle und Index von der Sortierung her überein. Der Wert sagt im Grunde aus, wie viele verschiedene Blöcke der Tabelle angesprochen werden müssen, wenn man alle Zeilen der Tabelle in der Reihenfolge des Index ausliest. Im Idealfall – wenn also Tabelle und Index die exakt gleiche Sortierung haben – entspricht der Wert der Anzahl der Blöcke der Tabelle, da dann jeder Block genau nur einmal angesprochen werden muss. Im schlechtesten Fall entspricht der Wert der Anzahl der Zeilen der Tabelle, die über den Index angesprochen werden können, da dann für jede Zeile auf einen anderen Block der Tabelle zugegriffen werden muss.

Nun können die eigentlichen Abfragen auf den beiden Tabellen ausgeführt werden *(siehe Listing 9 für den Zugriff per LOCATION_ID und Listing 10 für den Zugriff per PRODUCT_ID)*.

Wie man sehen kann, benötigt die Tabelle mit „Attribute Clustering" in allen Beispielen – egal ob auf LOCATION_ID oder PRODUCT_ID (auch beides wäre möglich) gefiltert wird – deutlich weniger logisches I/O („consistent gets"-Statistik) als die Tabelle ohne „Attribute Clustering". Das Besondere an dem Beispiel ist eben, dass beide Attribute LOCATION_ID und PRODUCT_ID unabhängig voneinander als Filter verwendet werden können und beide Zugriffswege durch das „Interleaved Ordering" des „Attribute Clustering" optimiert werden.

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
msk@ordix.de

Ein Einstieg in Oracle REST Data Services für autonome Datenbanken

Timo Herwix, MT

Oracle REST Data Services (ORDS) erleichtert die Entwicklung von REST-Schnittstellen für relationale Daten in einer Datenbank. ORDS ist eine Mid-Tier-Java-Anwendung, die HTTP(S)-Verben wie GET, POST, PUT, DELETE auf Datenbanktransaktionen abbildet und alle Ergebnisse als JSON-Daten zurückgibt. Die Anwendung ist für autonome Datenbanken vorkonfiguriert und wird vollständig in der Oracle Cloud verwaltet. In der „Database Actions"-Benutzeroberfläche, die bei allen ORDS-Installationen und der autonomen Datenbank in der Oracle Cloud Infrastructure verfügbar ist, kann ein REST-Service erstellt werden. Dabei werden Schritte wie das Anlegen eines Anwendungsbenutzers und das Erstellen einer Tabelle in der autonomen Datenbank durchgeführt, bevor die Tabelle für REST aktiviert wird. Auf diese Weise können Endpunkte für wichtige CRUD-Operationen wie Erstellen, Aktualisieren, Abfragen und Löschen angefertigt werden. Weiterhin können die Endpunkte des REST-Dienstes mittels OAuth2 gesichert werden.

Oracle REST Data Services überbrückt HTTPS und die Oracle-Datenbank. Als Mid-Tier-Java-Anwendung bietet sie ein REST-API für das Datenbankmanagement, SQL Developer Web, ein PL/SQL-Gateway, SODA für REST und die Möglichkeit, RESTful-Webservices für die Interaktion mit den Daten und gespeicherten Prozeduren in einer Oracle-Datenbank zu veröffentlichen. ORDS ist außerdem durch die Unterstützung von Deployments mit Oracle WebLogic Server, Apache Tomcat und einem Stand-alone-Modus äußerst flexibel. Es vereinfacht den Deployment-Prozess weiter, da kein Oracle Home erforderlich ist und die Konnektivität über einen eingebetteten JDBC-Treiber bereitgestellt wird.

Wird Oracle Application Express verwendet, ist hierfür ein Webserver notwendig, um Anfragen zwischen einem Webbrowser und der Oracle-Application-Express-Engine weiterzuleiten. Diese Anforderung erfüllt ebenfalls Oracle REST Data Services *(siehe Abbildung 1)*.

Representational State Transfer (REST)

REST steht für Representational State Transfer und ist eine Architektur, die zur Entwicklung von Webdiensten verwendet wird. Mit RESTful-Services kann eine Darstellung an angeforderten Daten, die in einer Datenbank gespeichert sind, abgefragt oder bearbeitet werden. REST verwendet die http-Methoden POST, PUT, GET und DELETE, um Create-, Read-, Update- oder Delete-Operationen auf Daten durchzuführen. Die Ressourcen werden durch URLs identifiziert und lösen immer eine Antwort aus. Diese Antwort liegt in Form von XML, JSON, HTML oder einem anderen definierten Format vor *(siehe Abbildung 2)*. Je nach Vorgang enthalten die Antworten Einzelheiten zu einer Änderung der zugrunde liegenden Daten, Fehlermeldungen und Hypertext-Links zu anderen verwandten Ressourcen. REST-Services sind zustandslos, da keine Verbindung zwischen dem Client und dem Computersystem aufrechterhalten wird und kein Client-Kontext zwischen Anfragen gespeichert wird.

Dadurch sind sie für schnelle Leistung, Zuverlässigkeit, Portabilität und Wachstum optimiert.

Oracle Database Actions (SQL-Developer Web)

Oracle selbst bezeichnet die Database Actions als bestes Datenbankmanagement-Tool im Browser. Um Database Actions nutzen zu können, muss vom Anwender nichts Weiteres installiert oder unternommen werden. Mit jeder Installation von ORDS werden die Database Actions zur Verfügung gestellt. Nicht einmal eine Connection zur Datenbank muss hergestellt werden. Zudem bieten die Database Actions viele Möglichkeiten, um mit der Datenbank arbeiten zu können. Der dabei wichtigste Punkt ist wohl, dass keinerlei Kosten entstehen. Zudem sind regelmäßige Wartungen und Updates gewährleistet. Als Features bieten die Database Actions Anwendern einen umfangreichen SQL-Developer, womit im Browser direkt Abfragen geschrieben und ausgeführt werden können. Ohne großen Aufwand können Daten in eine Datenbank importiert oder exportiert werden. Und natürlich gibt es einen kompletten Überblick über REST-APIs. Monitoring und Logging gehören ebenfalls zu den wichtigsten Features. Das Schöne in der Oracle-Cloud und damit verbundenen autonomen Datenbanken ist, dass ORDS automatisch installiert wird, sobald eine ADB aufgesetzt wird. Somit stehen die Database Actions direkt zur Verfügung *(siehe Abbildung 3)*.

Fünf Gründe, warum die Database Actions genutzt werden sollten

- Es wird aktiv weiterentwickelt und regelmäßig aktualisiert
- Security! Alle REST-APIs können durch OAuth2-Clients geschützt werden
- REST-APIs können mittels cURLS schnell getestet werden

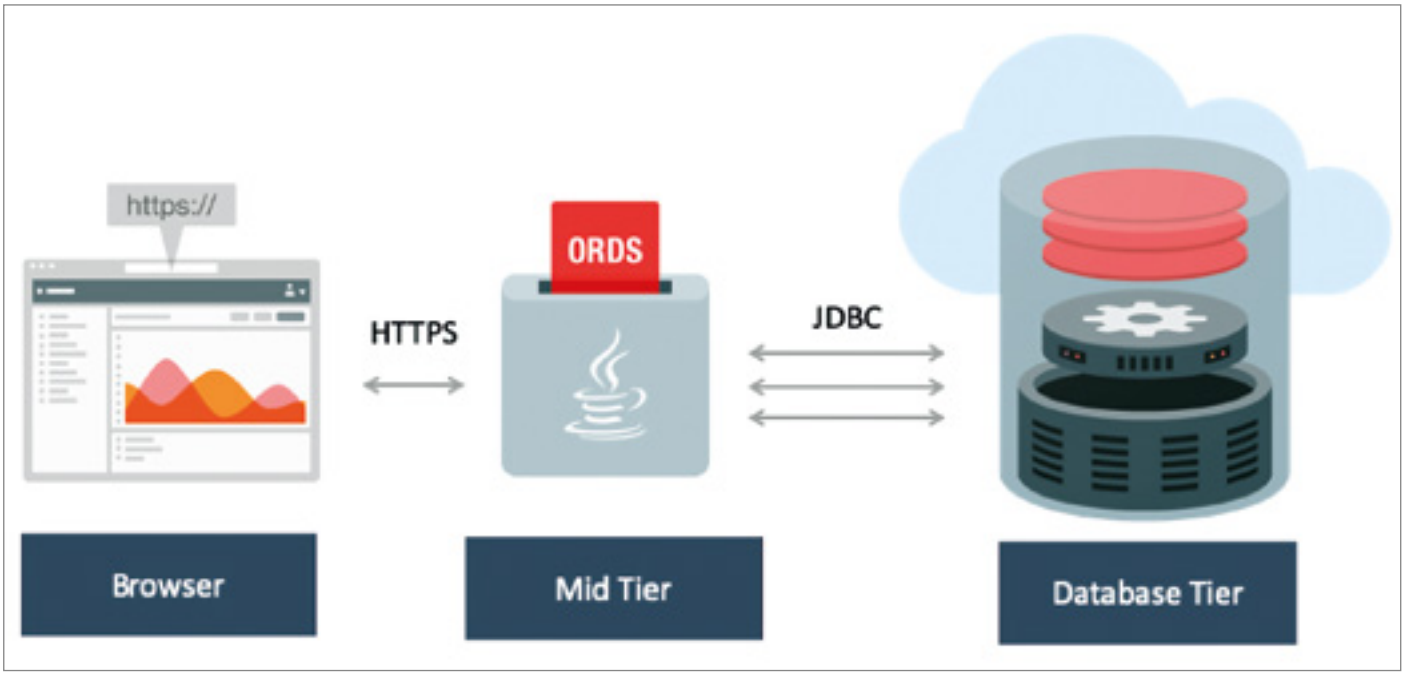


Abbildung 1: Architektur ORDS (Quelle: JMJ-Cloud [1])

Abbildung 2: ORDS REST Request & Response Workflow (Quelle: Jeff Smith [2])

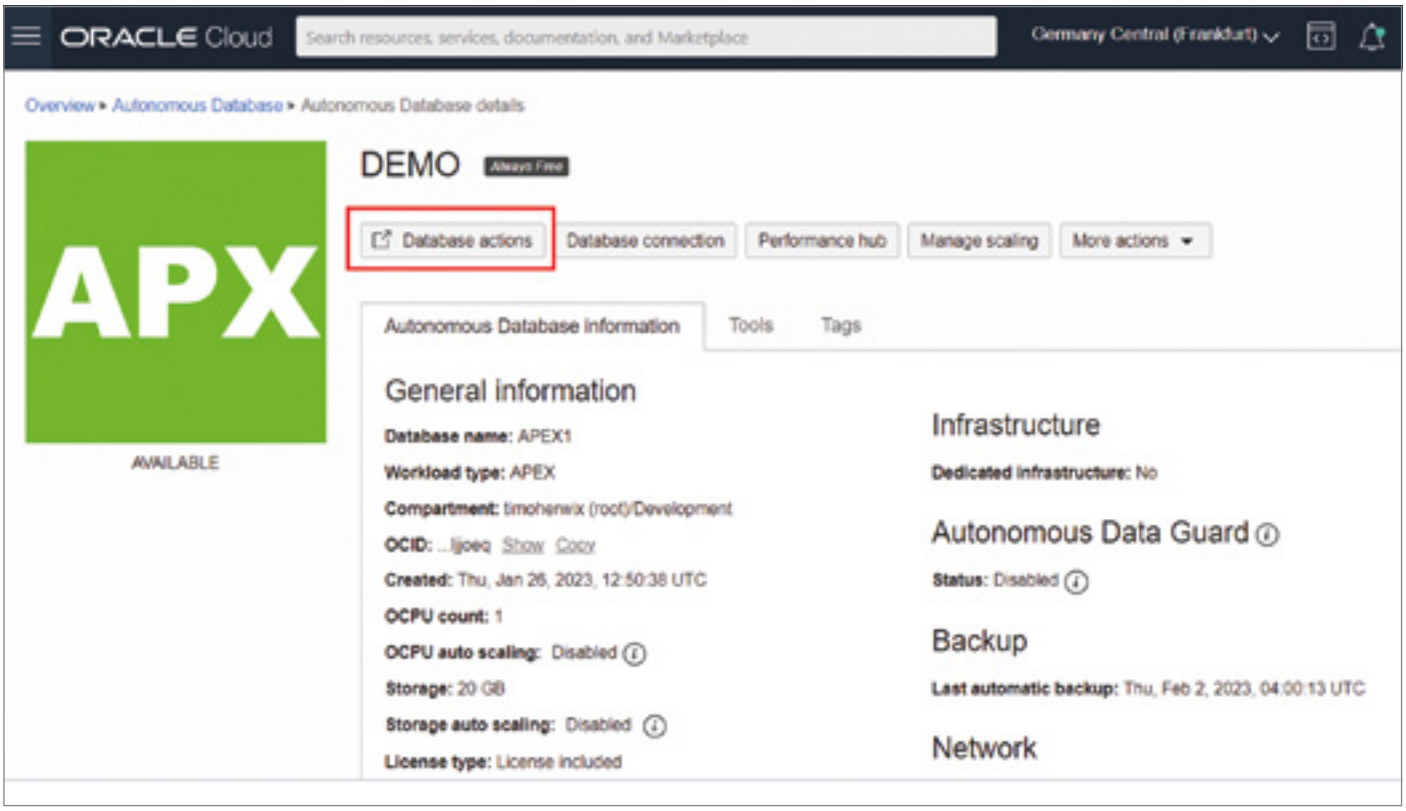


Abbildung 3: Oracle Cloud – Autonomous Database (Quelle: Timo Herwix)

- OpenAPI 3 Support, um REST-APIs zu dokumentieren
- Es ist keine Installation notwendig, auch wird kein REST-Client wie Postman oder Insomnia benötigt

Von der Theorie zur Praxis – eine Anleitung

Um die REST-Services für autonome Datenbanken nutzen zu können, kann die mitgelieferte Database-Actions-Oberfläche genutzt werden. Diese kann aus der Oracle-Cloud-Konsole aufgerufen werden.

Hinweis: Auf diesem Wege wird man direkt als „Admin" angemeldet und hat vollen Zugriff.

Als Erstes sollte ein neues Datenbankschema für Datenbankobjekte und Daten erstellt werden. Demnach muss ein neuer User erstellt werden. Damit der neu erstellte User die REST-Services nutzen kann, muss „Web-Access" aktiviert werden. Zusätzlich wird auch Plattenspeicher für das Schema benötigt, weshalb hier dem User noch Speicher zugewiesen werden kann *(siehe Abbildung 4)*.

Mit den User-Daten erfolgt die Anmeldung auf die Database-Actions-Oberfläche. Nun ist das Erstellen von Tabellen und Einfügen von Daten möglich. Am einfachsten und schnellsten lässt sich das durchführen, indem als Basis ein Excelsheet mit Daten verwendet wird und diese importiert werden. Hierzu wird im SQL-Developer-Web ein „Data Loading"-Wizzard bereitgestellt *(siehe Abbildung 5)*.

Da jetzt eine Tabelle existiert, kann für diese ein REST-Service erstellt werden. Hierzu kann die „Auto-REST-Aktivierung" genutzt werden. Die soeben erstellte (oder gewünschte) Tabelle befindet sich im Navigator auf der linken Seite des SQL-Developers. Dort wird „REST" durch einen Klick mit der rechten Maustaste auf den Tabellennamen und dann unter „REST aktivieren" genutzt *(siehe Abbildung 6)*.

Auf der rechten Seite öffnet ein Slider, der Einstellungen für den REST-Service dient. Im ersten Schritt können hier die Standardeinstellungen verwendet und die Vorschau-URL in die Zwischenablage kopiert werden *(siehe Abbildung 7)*.

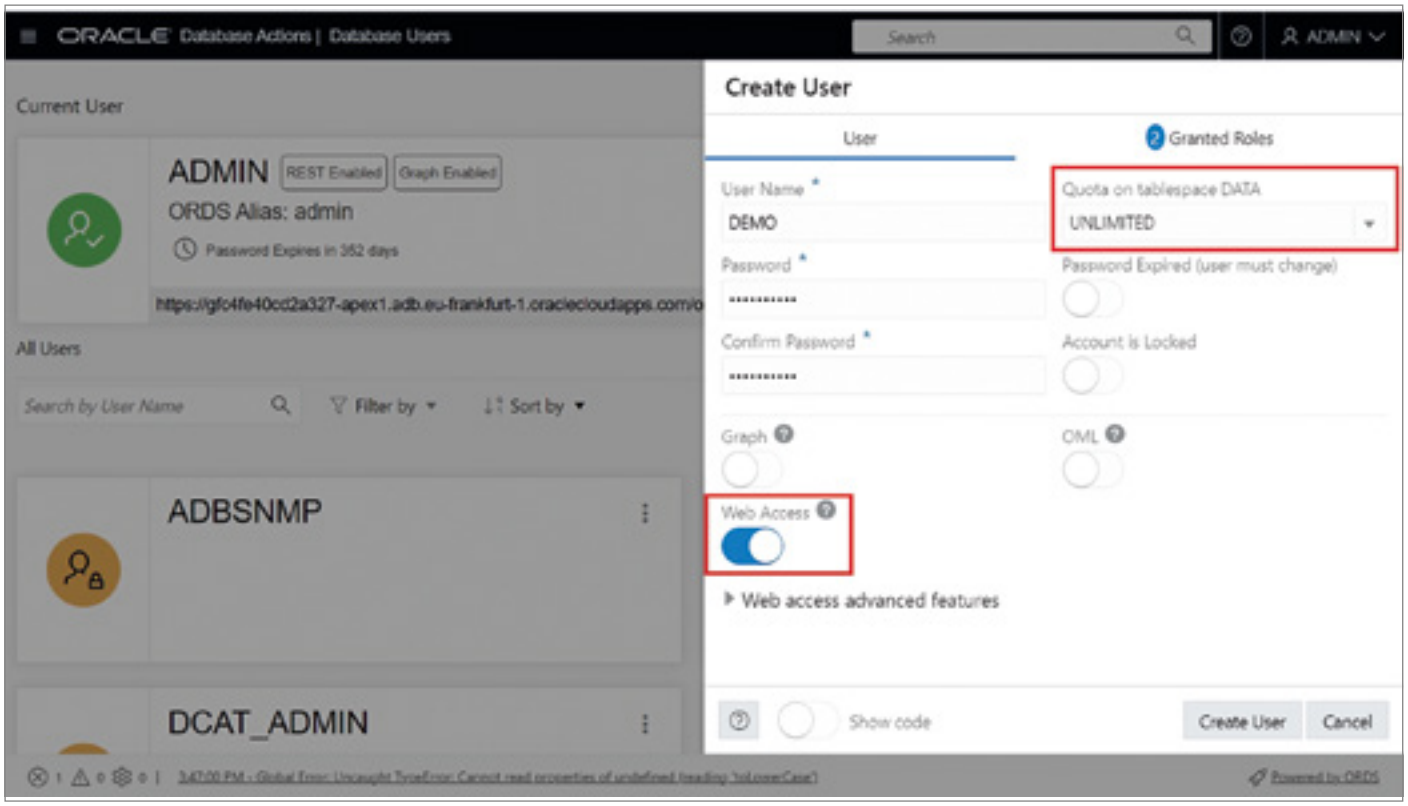


Abbildung 4: Neuen User erstellen (Quelle: Timo Herwix)

Das wars! Die Tabelle ist ab sofort REST-fähig und kann verwendet werden. Um den Zugriff zu testen, kann beispielsweise in einem neuen Browserfenster die Vorschau-URL eingefügt werden, die im vorherigen Schritt in die Zwischenablage kopiert wurde. Als Ergebnis erscheinen die Tabellendaten im JSON-Format *(siehe Abbildung 8)*.

Durch die Auto-Rest-Aktivierung wurden noch mehr REST-Endpoints erstellt, die ab sofort genutzt werden können. Bei einem Wechsel des SQL-Developer-Webs zurück zur Übersicht der Database Actions und der Nagivation zu „REST" können die REST-APIs mithilfe der OpenAPI View genauer betrachtet werden. Mit dieser automatisch generierten Dokumentation wird das API beschrieben und es wird sichtbar, welche Aufrufe zur Verfügung stehen *(siehe Abbildung 9)*.

Security

Die Tabelle ist jetzt REST-fähig und kann von Anwendungen genutzt werden. Jedoch sollte sichergestellt werden, dass diese gegen einen allgemeinen Zugriff abgesichert sein sollte. Hierzu wird die Absicherung via OAuth2-Client seitens der Database Actions angeboten. Ein OAuth2-Client kann in „Security > OAuth-Clients" erstellt werden. Rechts auf der Seite erscheint ein Slider „OAuth-Client erstellen", in dem der Client definiert werden kann. Im Tab „Role" muss die zuvor erstellte Auto-REST-Rolle der Tabelle hinzugefügt werden, wodurch alle REST-Services mit dieser Rolle abgesichert werden können.

Jetzt ist es an der Zeit, den REST-Service zu sichern. In „AutoREST" wird das Objekt durch Anklicken des Menü-Symbols und „Bearbeiten" verändert. Durch das Aktivieren der Schaltfläche „Require Authentication" wird der Service gesichert *(siehe Abbildung 10)*.

Um erneut den Zugriff zu testen, kann beispielsweise wieder die Vorschau-URL in einem neuen Browserfenster eingefügt werden. Nun sieht man, dass man keinen Zugriff mehr auf den REST-Service hat *(siehe Abbildung 11)*.

Um den abgesicherten REST-Service dennoch zu testen, kann zum Beispiel die OCI-Cloud-Shell verwendet und dort mittels cURL-Befehl das API aufgerufen werden. Der cURL-Befehl wird ebenfalls automatisch generiert und kann über das Menü des REST-Service aufgerufen werden, indem dort „Get cURL" aktiviert wird. Hier lässt sich erkennen, dass der cURL-Befehl nun seine Header-Informationen enthält, indem ein Bearer Token angegeben werden muss *(siehe Listing 1)*.

Den Bearer Token erhält man, indem wieder zu „Security > OAuth-Clients" navigiert wird und dort auf den entsprechenden Client „Get Bearer Token" ausgewählt wird *(siehe Abbildung 12)*.

Über Oracle-Cloud-Konsole wird die Cloud Shell gestartet. Wird jetzt der entsprechende cURL-Befehl samt Bearer Token ausgeführt, wird das Ergebnis des REST-Service angezeigt *(siehe Abbildung 13)*.

Fazit

Oracle REST-Services für autonome Datenbanken können sehr schnell und einfach mit ausschließlich SQL- und PL/SQL-Kenntnissen erstellt werden. Dabei kümmert sich ORDS um die Security und stellt einen OAuth2-Client zur Verfügung, womit alle REST-APIs schnell und sicher geschützt werden können. Nutzt man die Auto-Rest-Funktion, muss erst gar kein Code geschrieben werden und alle Standardfunktionen stehen mit wenigen Klicks zur Verfügung. Da ORDS, und somit auch die Database Actions, automatisch installiert werden, sobald

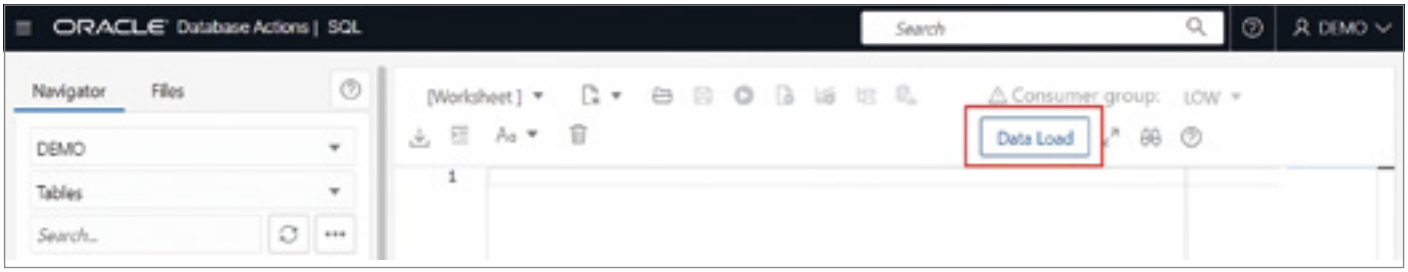


Abbildung 5: Data Loader (Quelle: Timo Herwix)

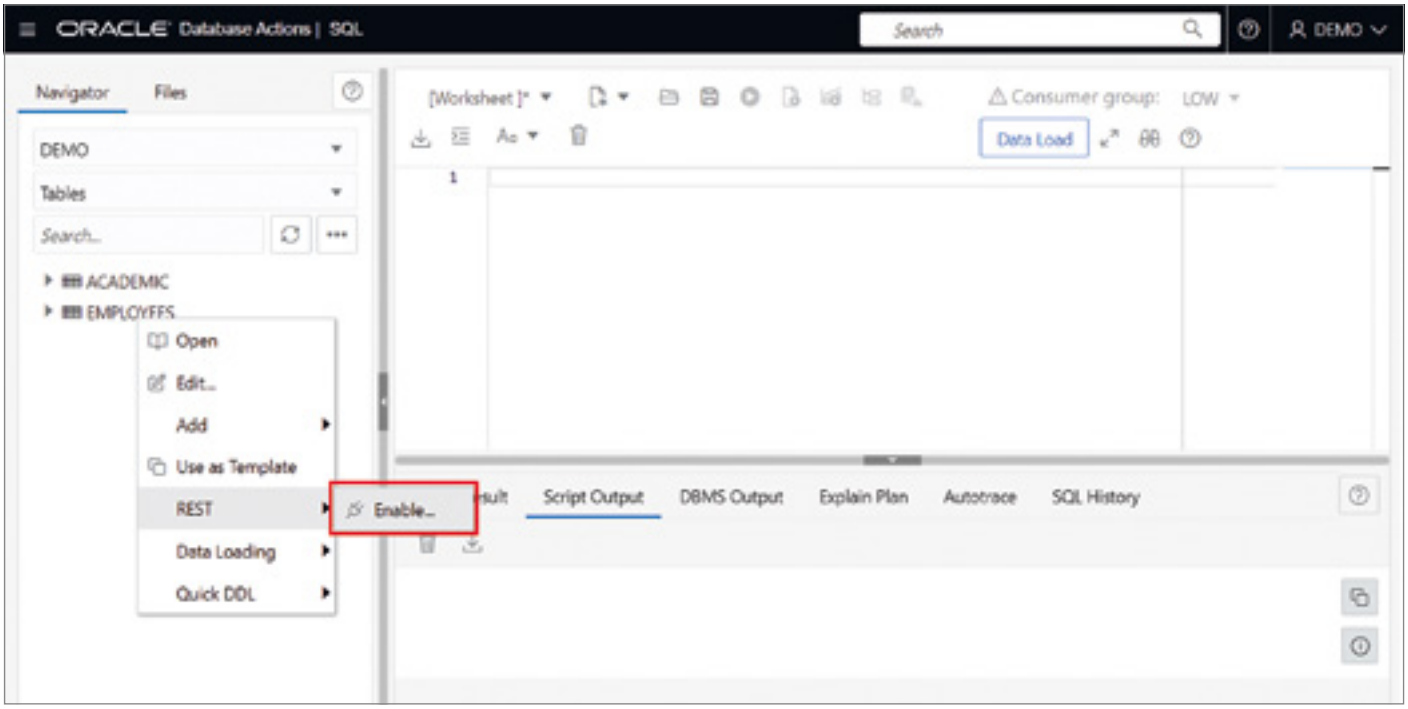


Abbildung 6: Auto-REST aktivieren (Quelle: Timo Herwix).

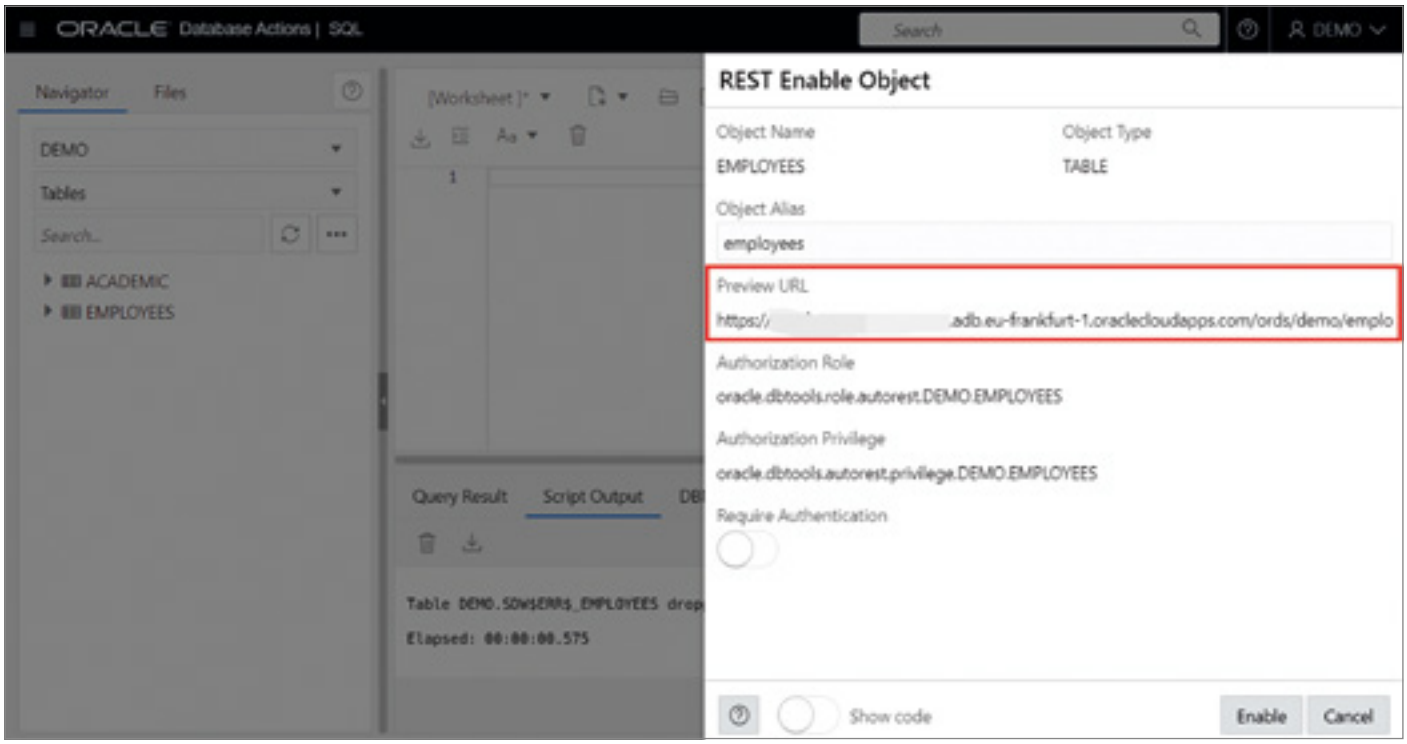


Abbildung 7: Auto-REST aktivieren (Quelle: Timo Herwix)

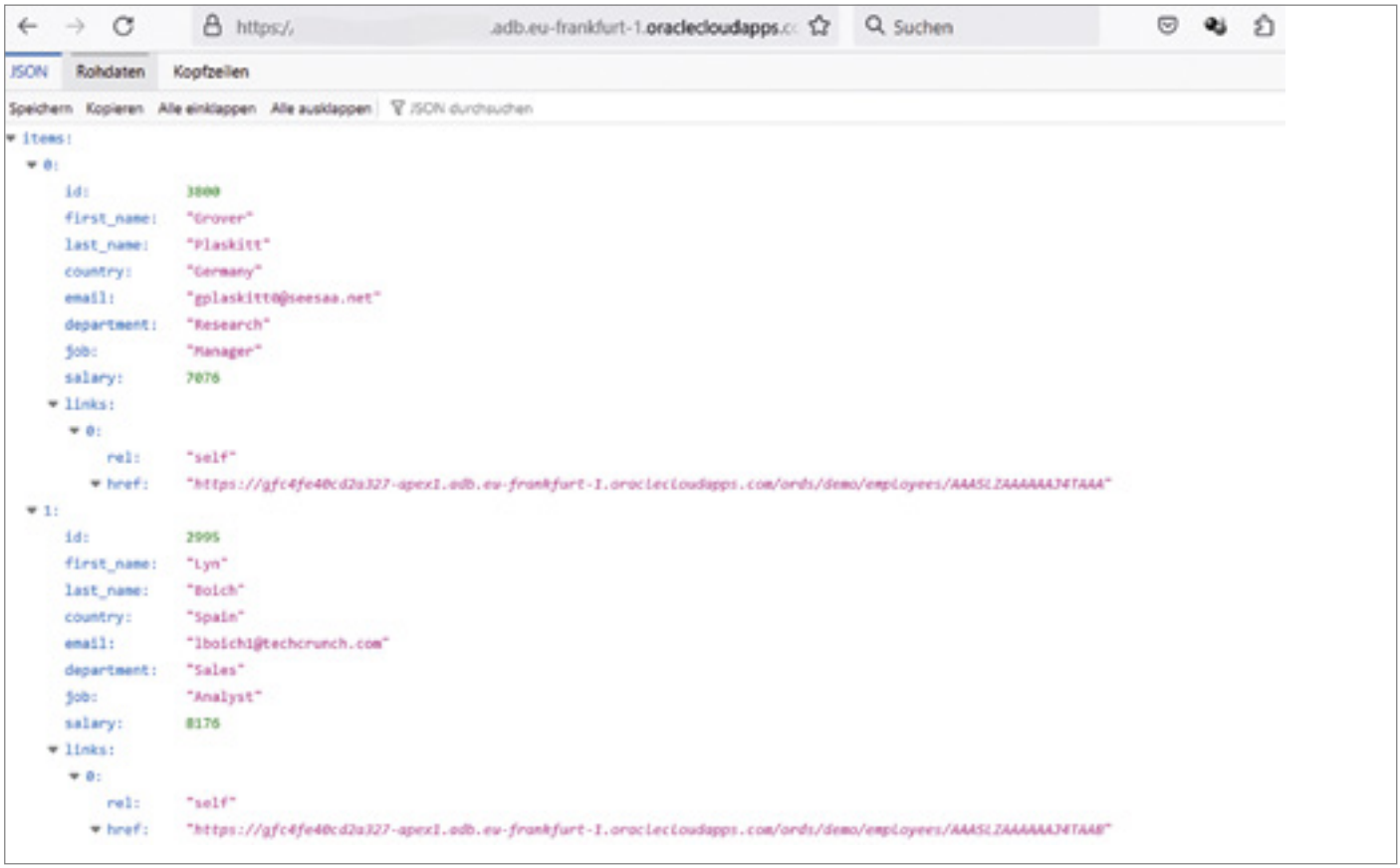


Abbildung 8: Aufruf REST-Service (Quelle: Timo Herwix)

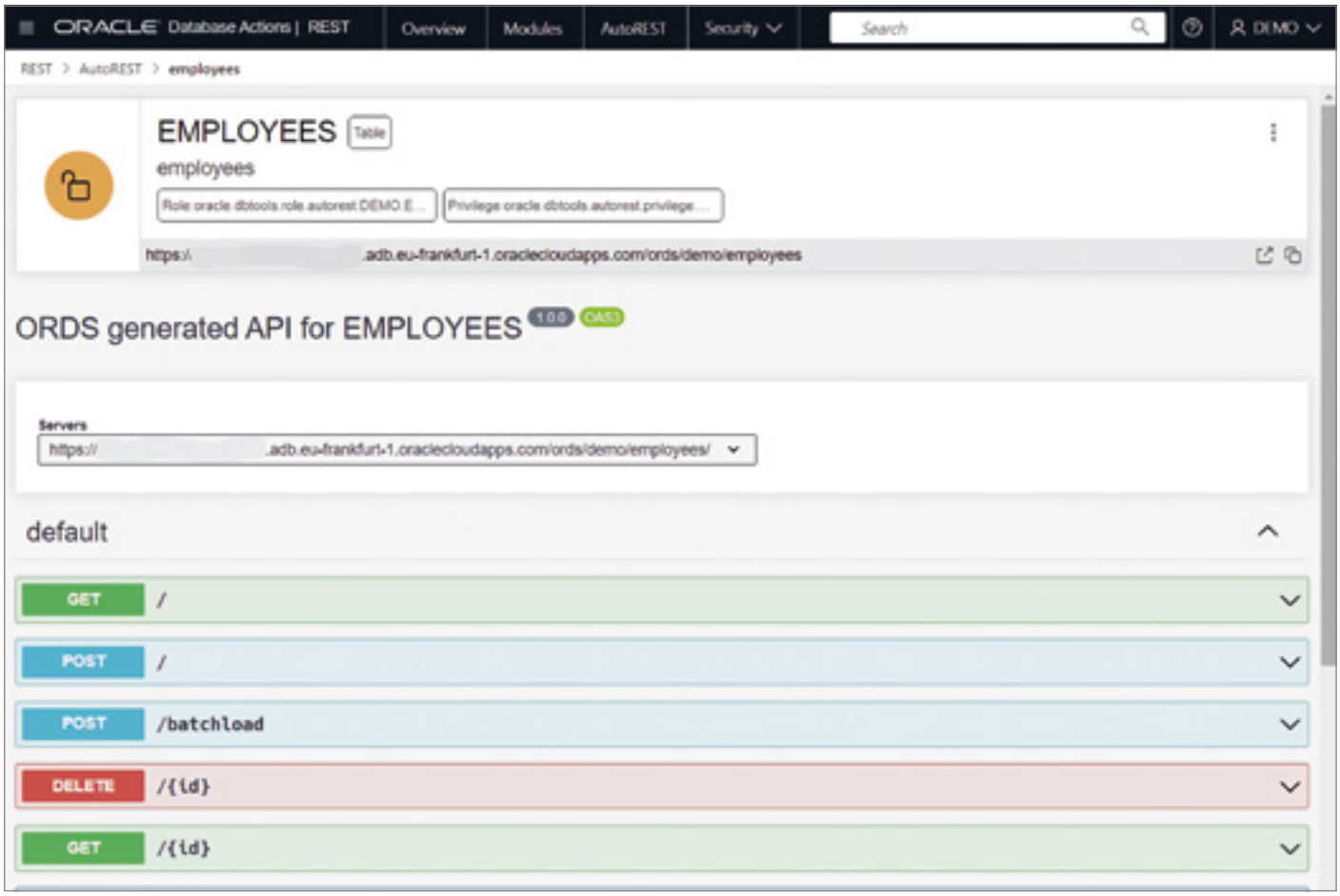


Abbildung 9: OpenAPI Support (Quelle: Timo Herwix)

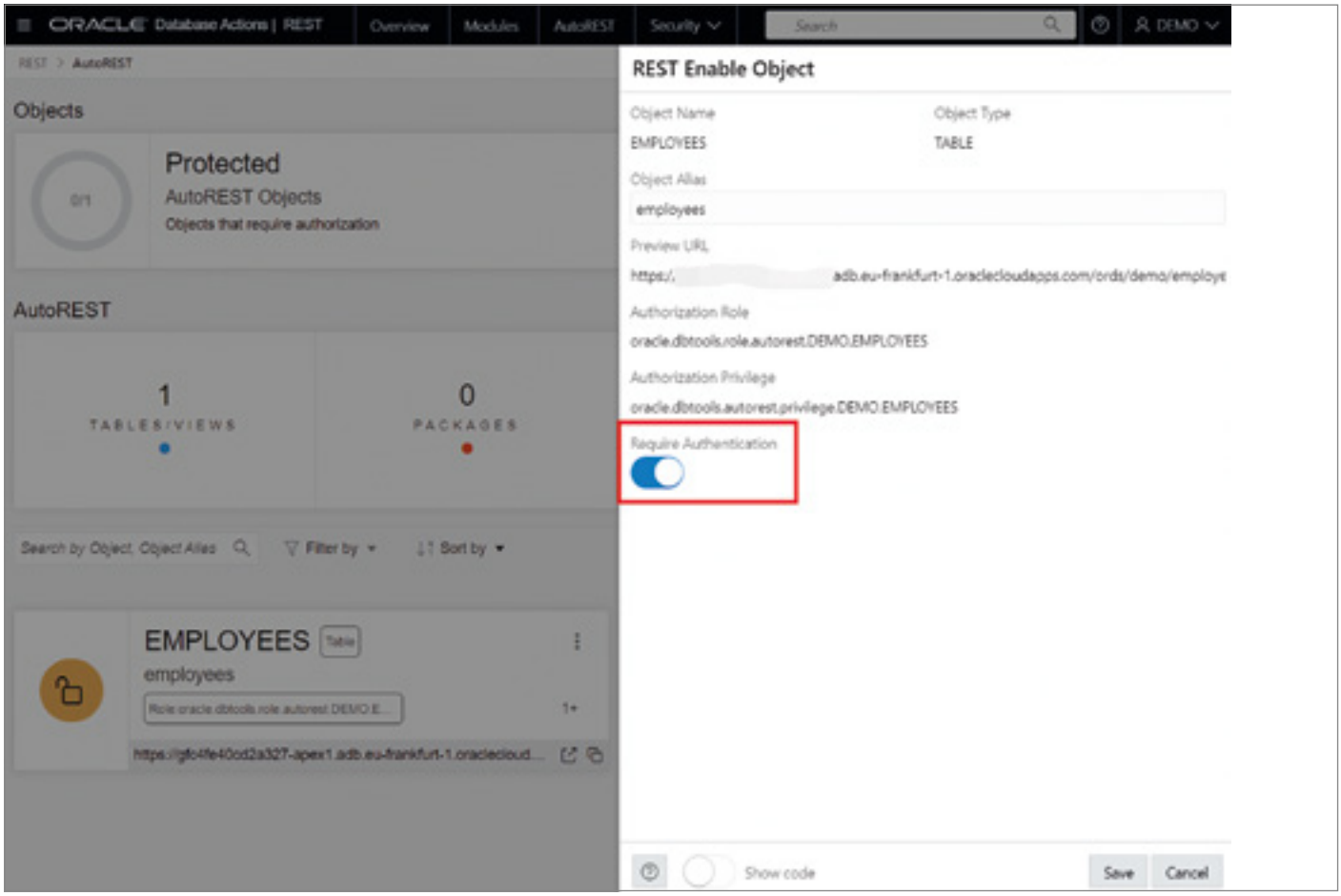


Abbildung 10: Aktivierung OAuth2-Client (Quelle: Timo Herwix)

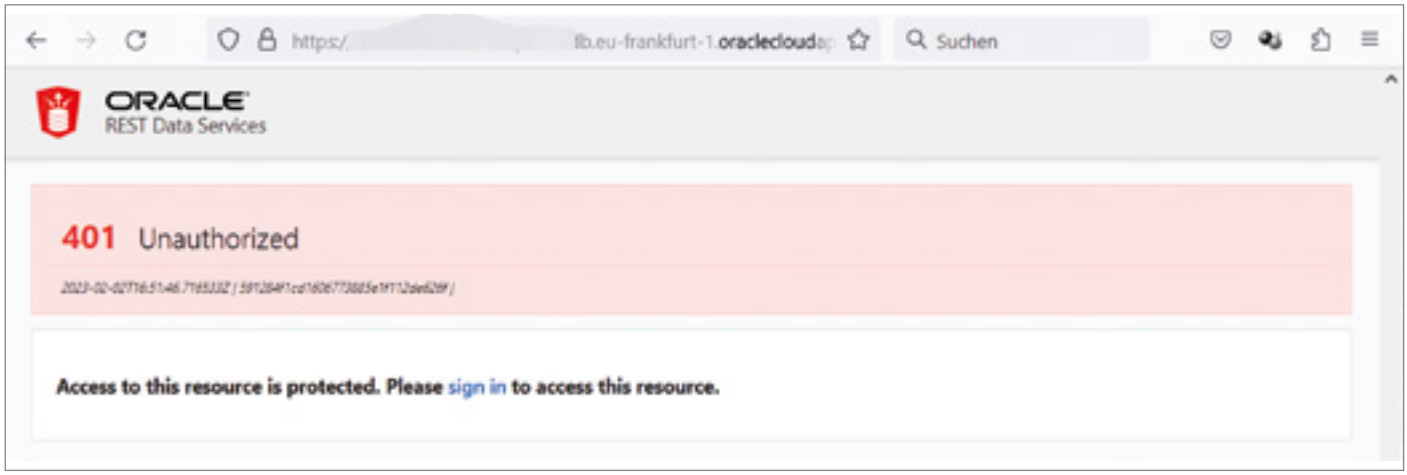


Abbildung 11: Aufruf abgesicherter REST-Service (Quelle: Timo Herwix)

```
curl --location \
"https://my_oci_endpoint.adb.eu-frankfurt-1.oraclecloudapps.com/ords/demo/employees/" \
--header "Authorization: Bearer <VALUE>"
```

Listing 1: cURL-Befehl zum Testen des REST-Service

eine autonome Datenbank aufgesetzt wurde, muss man sich auch um keinerlei Infrastruktur kümmern.

Quellen

[1] *https://www.jmjcloud.com/blog/convince-your-boss-to-try-apex-and-ords*

[2] *https://www.thatjeffsmith.com/archive/2017/03/running-oracle-rest-data-services-ords-without-oracle-application-express-apex/*

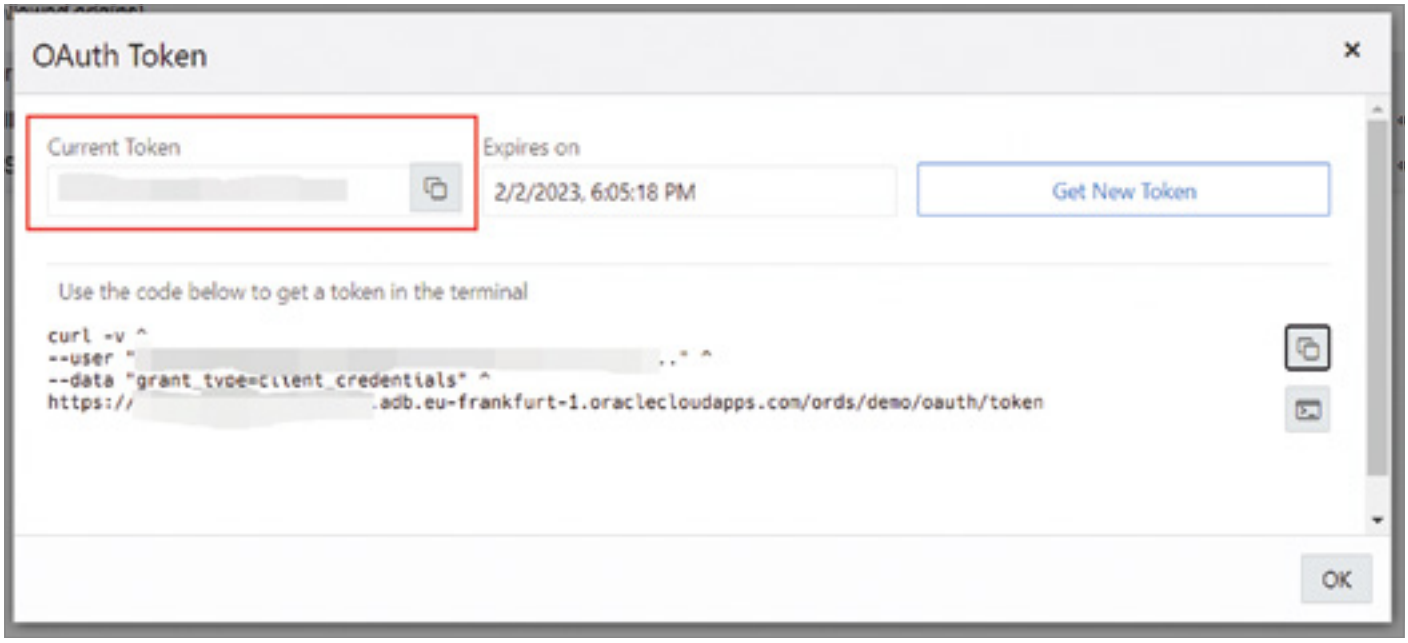


Abbildung 12: Bearer Token anzeigen (Quelle: Timo Herwix)

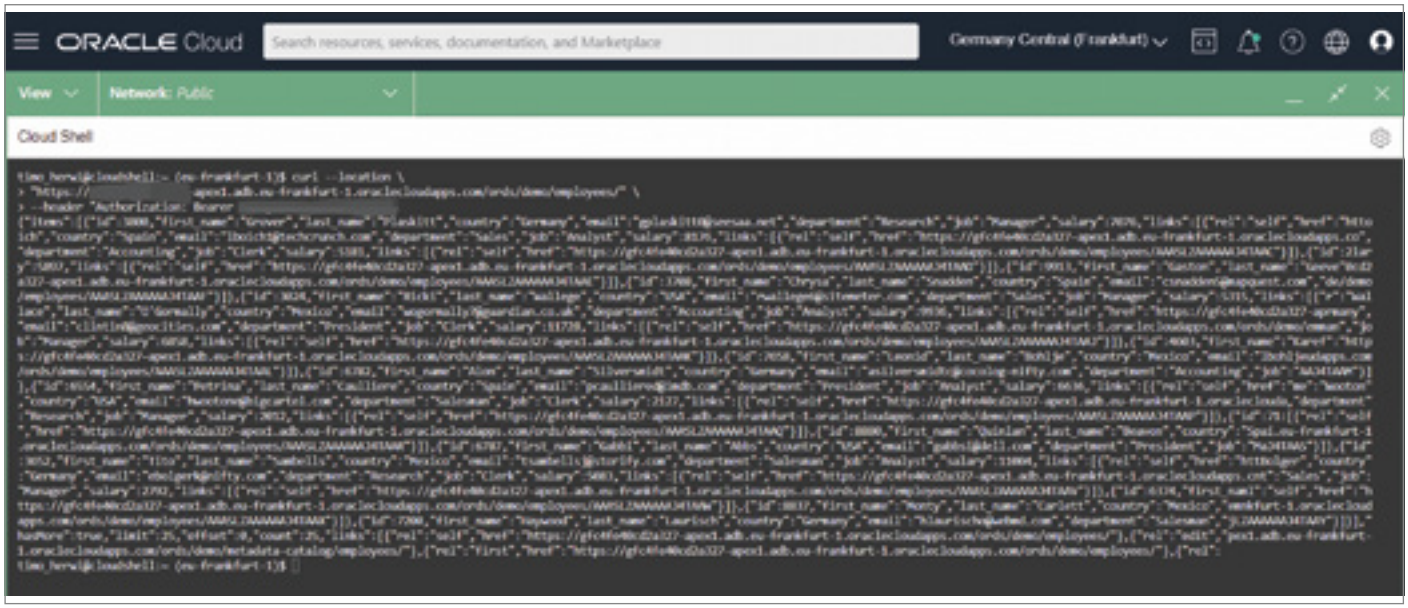


Abbildung 13: cURL Test mittels Cloud Shell (Quelle: Timo Herwix)

Über den Autor

Timo Herwix ist seit 2019 als Senior Consultant bei der MT in Ratingen tätig. In seinem Tagesgeschäft arbeitet er mit Oracle-Datenbanken und der Entwicklung von Oracle-APEX-Anwendungen. Vor seinem Einstieg bei der MT arbeitete Timo Herwix als Data-Warehouse-Spezialist, wo er Erfahrungen als Datenbankadministrator, im Performance Tuning und bei der SQL-Entwicklung sammelte. Neben Datenbankmodellierung, PL/SQL-Entwicklung und dem Entwickeln von APEX-Anwendungen interessiert er sich generell für Webentwicklung und deren Architektur. 2022 hielt Timo Herwix auf der „DOAG-Konferenz + Ausstellung" einen Vortrag zum Einstieg in Oracle REST Data Services für autonome Datenbanken.

Timo Herwix
Timo.herwix@mt-itsolutions.com

Oracle PL/Scope – [unbekanntes] Schweizer Taschenmesser zur PL/SQL-Code-Analyse oder „Einfach mal machen“

Peter van Garsel, eWorkIng

Was ist PL/Scope und was leistet PL/Scope?

Der Oracle PL/SQL Compiler stellt kostenfrei und mit wenig Aufwand [vielfältige] Hilfen bereit, die PL/SQL-Developer entsprechend ihren Aufgaben (Implementierung, Dokumentation, Refactoring ...) nutzen können.

Zur Einführung von PL/Scope werden exemplarisch folgende Fragestellungen behandelt:

- Wie ist eine Variable definiert?
- Wo und wie wird eine Variable benutzt?
- Wo wird ein SQL-Statement implementiert und wie wird es dargestellt?
- Wird ein SQL-Statement mehrfach in [unterschiedlichen] Blöcken benutzt?

Prolog

PL/Scope stellt semantisch aufbereitet Quellcode-Informationen in der View user_identifiers und für SQL-Statements in der View user_statements bereit.

Die bereitgestellten Metadaten gehen deutlich über die Informationen hinaus, die mit SQL- Statements auf die View user_source column text abgefragt werden können.

Generell ist bei der Nutzung von PL/Scope ein **Step-by-Step**-Ansatz zu wählen, da Lösungswege für bestimmte Fragestellungen komplex ausfallen können.

PL/Scope wird in der Praxis eher selten genutzt, weil es wenig bekannt ist beziehungsweise wenig promotet wird. Nachfolgende Quellen liefern weitere Beispiellösungen, Ergänzungen:

- Oracle Live SQL >> Code Library >> Suchwörter; PL/Scope | user_identifiers
- eBooks (PL/SQL) >>Jürgen Sieben bzw. Steven Feuerstein
- GitHub >> PhilippSalvisberg; SQL-Developer extension beziehungsweise customized Views 4 PL/Scope

Bekannte Quellen für eine Code-Analyse sind:

- Oracle Data Dictionary Views
  - user_procedures
  - user_arguments
  - user_dependencies
  - user_source
  - user_errors, Oracle Compilier Warnings
  - ...
- Oracle Supplied Packages
  - Profiler, dbms_profiler, 8i
  - Hierarchical Profiler, dbms_hprof, 11g R1
  - Code Coverage, dbms_plsql_code_coverage, 12.2

Check: Kann PL/Scope genutzt werden? *(siehe Listing 1)*
Wie wird PL/Scope „aktiviert“? *(siehe Listing 2)*

Hinweis

PLSCOPE_SETTINGS und PLSQL_WARNINGS können unabhängig auf DB-Ebene, Session-Ebene und [auch] in Tool-Konfiguration eingestellt werden. Deshalb bitte Tool-Konfiguration entsprechend überprüfen.

PL/Scope liefert generell keine Metadaten in nachfolgenden Fällen:

- Anonyme Blöcke
- Dynamic SQL, NDS (native dynamic SQL, dbms_sql ...)
- Invalid Code bzw. user_objects.status = INVALID
- SYS-Objects
- Wrapped Code
- plscope_settings sind IDENTIFIERS:NONE für relevantes Objekt

Start mit Song „Bottles of beer“ *(siehe Listing 3 und 4)*.
First RUN SQL – user_identifiers and user_source *(siehe Listing 5 und Abbildung 1)*.
Die Procedure BOB zeigt beispielhaft, welche Informationen nun via View user_identifiers und View user_source bereitgestellt beziehungsweise welche Informationen **nicht** bereitgestellt werden.

Die Oracle-Dokumentation (Database PL/SQL Language Reference, 19c >> PL/SQL Language Fundamentals >> Identifiers) definiert Identifiers als „ Identifiers name PL/SQL elements“.

Über die Spalten usage_id beziehungsweise usage_context_id werden Hierarchien zwischen identifiers abgebildet *(siehe Oracle DOC About „Identifiers Usage Context“)*.

Die View user_identifiers liefert also Metainformationen über „Named PL/SQL Elements“ und deren Nutzung. Nicht mehr, aber auch nicht weniger.

Die View user_identifiers kann also nicht Metadaten über **alle** Lines der View user_source bereitstellen. In dem gewählten Beispiel können deshalb keine Kommentare, FOR-Anweisung, IF ausgewertet werden.

Check: Wo ist Variable L_PARTY_START deklariert? *(siehe Listing 6)*

Check: Wo wird Variable L_PARTY_START wie genutzt? *(siehe Listing 7)*

Check: Namenskonvention für IN-Parameter? *(siehe Listing 8)*

Show Case: Welche Informationen stellt user_statements wie bereit? *(siehe Listing 9)*

Das SQL-Statement wird offensichtlich von der View user_statements reformatiert und mit den Bind-Variablen in der View dargestellt.

Diese „Anpassungen“ sind bei allen Suchen auf diese Spalte text immer entsprechend zu berücksichtigen.

Show Case: Ist ein SQL-Statement mehrfach implementiert? *(siehe Listing 10)*

```
show parameter plscope_settings
-- Result
-- NAME               | TYPE    | VALUE
-- ------------------ + ------- +-------------------------------
-- plscope_settings   + string  | IDENTIFIERS:ALL, STATEMENTS:ALL

SELECT name, value
  FROM v$parameter
  WHERE name ='plscope_settings'
-- Result
-- NAME              | VALUE
--  ----------------|-------------------------------
-- plscope_settings |IDENTIFIERS:ALL, STATEMENTS:ALL
```

Listing 1: Kann PL/Scope genutzt werden?

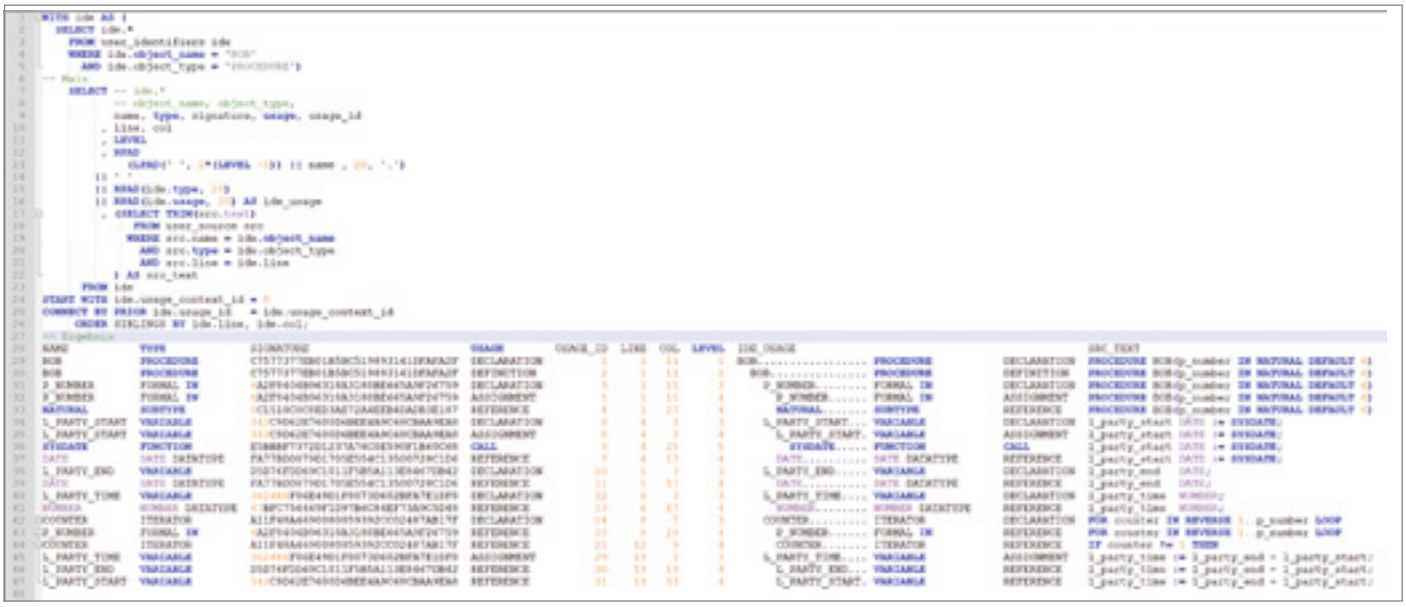


Abbildung 1: PL/Scope-Metadaten im Nutzungskontext

```
ALTER SESSION SET plscope_settings = 'IDENTIFIERS:ALL, STATEMENTS:ALL';
-- Result
-- Session altered.
```

Listing 2: Wie wird PL/Scope aktiviert?

```
CREATE OR REPLACE PROCEDURE BOB(p_number IN NATURAL DEFAULT 6)
-- Source; //www.99-bottles-of-beer.net
IS
  l_party_start DATE := SYSDATE;
  l_party_end   DATE;
  l_party_time  NUMBER;
BEGIN
  FOR counter IN REVERSE 1..p_number LOOP
      dbms_output.put_line(counter || ' bottles of beer on the wall,');
      dbms_output.put_line(counter || ' bottles of beer.');
      dbms_output.put_line('Take one down, pass it around,');
     IF counter != 1 THEN
        dbms_output.put_line(counter - 1 || ' bottles of beer on the wall.');
       ELSE
        dbms_output.put_line('No more beers');
     END IF;
  END LOOP;
  SELECT SYSDATE INTO l_party_end FROM DUAL;
  l_party_time := l_party_end - l_party_start;
  dbms_output.put_line('Party duration ' || l_party_time);
END BOB;
```

Listing 3: Anlage Procedure BOB

```
SELECT status
  FROM user_objects
 WHERE object_name = 'BOB';
-- Requried; VALID

SELECT plscope_settings
  FROM user_plsql_object_settings
 WHERE name = 'BOB';
 -- Requried; IDENTIFIERS:ALL, STATEMENTS:ALL
```

Listing 4: Procedure BOB ready 4 PL/Scope

```
WITH ide AS (
  SELECT ide.*
    FROM user_identifiers ide
    WHERE ide.object_name = 'BOB'
      AND ide.object_type = 'PROCEDURE')
-- Main
    SELECT name, type, usage, usage_id
         , line, col
         , LEVEL
         , RPAD
             (LPAD(' ', 2*(LEVEL -1)) || name , 20)
        || ' '
        || RPAD(ide.type, 20)
        || RPAD(ide.usage, 20) AS ide_usage
         , (SELECT TRIM(src.text)
              FROM user_source src
             WHERE src.name = ide.object_name
               AND src.type = ide.object_type
               AND src.line = ide.line
           ) AS src_text
      FROM ide
START WITH ide.usage_context_id = 0
CONNECT BY PRIOR ide.usage_id   = ide.usage_context_id
     ORDER SIBLINGS BY ide.line, ide.col;
;
```

Listing 5: PL/Scope-Metadaten im Nutzungskontext

```
  SELECT dcl.object_type, dcl.name, dcl.line, dcl.type, dcl.usage
       , (SELECT src.text FROM user_source src
           WHERE type = dcl.object_type and name = dcl.object_name
             AND line = dcl.line) AS src_text
    FROM user_identifiers dcl
   WHERE dcl.object_name = 'BOB'
     AND dcl.name        = 'L_PARTY_START'
     AND dcl.usage       = 'DECLARATION'
     AND dcl.type        = 'VARIABLE'
ORDER BY line;

Ergebnis
OBJECT_TYPE  NAME           LINE  TYPE      USAGE        SRC_TEXT
PROCEDURE    L_PARTY_START  4     VARIABLE  DECLARATION  l_party_start DATE := SYSDATE;
```

Listing 6: Wo ist Variable deklariert?

```
SELECT dcl.object_type, dcl.name, usg.line, usg.type, usg.usage
       , (SELECT src.text FROM user_source src
           WHERE type = usg.object_type and name = usg.object_name
             AND line = usg.line) AS src_text
    FROM user_identifiers dcl, user_identifiers usg
   WHERE dcl.object_name = 'BOB'
     AND dcl.name        = 'L_PARTY_START'
     AND dcl.usage       = 'DECLARATION'
     AND dcl.type        = 'VARIABLE'
     AND usg.signature   = dcl.signature
     AND usg.usage       <> 'DECLARATION'
ORDER BY line;

OBJECT_TYPE  NAME           LINE  TYPE      USAGE       SRC_TEXT
PROCEDURE    L_PARTY_START     4  VARIABLE  ASSIGNMENT  l_party_start DATE := SYSDATE;
PROCEDURE    L_PARTY_START    19  VARIABLE  REFERENCE   l_party_time := l_party_end - l_party_start;
```

Listing 7: Wo wird Variable wie genutzt?

```
SELECT prog.name, parm.line, parm.name parameter, parm.type
     , 'FORMAL IN - Def nicht korrekt' AS con_error
  FROM user_identifiers parm
     , user_identifiers prog
 WHERE     parm.object_name      = 'BOB'
       AND parm.object_type      = 'PROCEDURE'
       AND prog.object_name      = parm.object_name
       AND prog.object_type      = parm.object_type
       AND parm.usage_context_id = prog.usage_id
       AND parm.type             = 'FORMAL IN'
       AND parm.usage            = 'DECLARATION'
       AND UPPER (parm.name) NOT LIKE '%\_IN' ESCAPE '\';

NAME  LINE  PARAMETER  TYPE      CON_ERROR
BOB      1  P_NUMBER   FORMAL IN  FORMAL IN - Def nicht korrekt
```

Listing 8: Namenskonvention [Example IN- Parameter]

```
-- Procedure anlegen
CREATE OR REPLACE PROCEDURE p1
(
  p_empno  IN emp.empno%TYPE
 ,p_change IN NUMBER
) IS
BEGIN
  UPDATE emp e
     SET e.sal = e.sal * p_change
   WHERE e.empno = p_empno;
END p1;

-- Wie wird SQL - Statement abgebildet
SELECT type, sql_id, has_in_binds, text
  FROM user_statements
 WHERE object_name = 'P1';

-- Ergebnis
TYPE   | SQL_ID        | HAS_IN_BINDS | TEXT
------ + ------------- + ------------ + ------------------
UPDATE | 82thy9h77fc1q | YES          | UPDATE EMP E SET E.SAL = E.SAL * :B2 WHERE E.EMPNO = :B1

-- Reformatierung via View user_statements Column text
UPDATE EMP E SET E.SAL = E.SAL * :B2 WHERE E.EMPNO = :B1
```

Listing 9: Welche Informationen stellt user_statements wie bereit?

```
CREATE OR REPLACE PROCEDURE p2(p_empno_app  IN emp.empno%TYPE, p_change_app IN NUMBER)
IS
BEGIN
  UPDATE emp e SET e.sal = e.sal * p_change_app WHERE e.empno = p_empno_app;
END p2;

  SELECT object_name, line, sql_id, text
    FROM user_statements
   WHERE object_name IN ('P1', 'P2')
ORDER BY sql_id;


OBJECT_NAME  TYPE    LINE  SQL_ID         HAS_IN_BINDS  TEXT
P1           UPDATE     7  82thy9h77fc1q  YES           UPDATE EMP E SET E.SAL = E.SAL * :B2 WHERE
E.EMPNO = :B1
P2           UPDATE     4  82thy9h77fc1q  YES           UPDATE EMP E SET E.SAL = E.SAL * :B2 WHERE
E.EMPNO = :B1

OBJECT_NAME LINE SQL_ID         TEXT
P2             4 82thy9h77fc1q  UPDATE EMP E SET E.SAL = E.SAL * :B2 WHERE E.EMPNO = :B1
P1             7 82thy9h77fc1q  UPDATE EMP E SET E.SAL = E.SAL * :B2 WHERE E.EMPNO = :B1
```

Listing 10: Ist ein SQL-Statement mehrfach implementiert?

Quelle: https://rideafrica.org/meet-the-team

Via View user_statements können identische SQL-Statements erkannt werden (sql_id ist für die Procedures P1 und P2 **identisch**).
Hint: Unterschiede in der Formatierung innerhalb der Procedures P1 beziehungsweise P2 sind hierbei nicht relevant.

Fazit

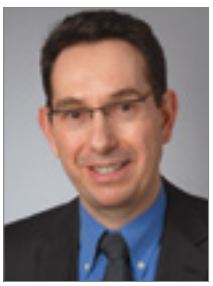
PL/Scope ist eine gute Erweiterung des Oracle Data Dictionary (ab Rel 12.1) für Aufgabenstellungen rund um Code-Analysen. Unterschiedlichste Fragestellungen aus den Bereichen Dokumentation, Refactoring etc. können semantisch korrekt einfach abgefragt werden.

Quellen

1. View all_identifiers: *https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/ALL_IDENTIFIERS.html*
2. View all_statements: *https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/ALL_STATEMENTS.html*
3. Using PL/Scope: *https://docs.oracle.com/en/database/oracle/oracle-database/19/adfns/plscope.html*
4. Database PL/SQL Lang Ref: *https://docs.oracle.com/en/database/oracle/oracle-database/21/lnpls/plsql-optimization-and-tuning.html*
5. Oracle SQL Developer, extension: *https://github.com/PhilippSalvisberg/plscope-utils*
6. Tim Hall: *https://oracle-base.com/articles/12c/plscope-12cr2*
7. Morgans Lib: *http://morganslibrary.org/reference/plsql/plscope.html*
8. Sabine Heimsath Schöner Coden – PL/SQL analysieren mit PL/Scope: *https://databine.databee.org/download/sabine-heimsath-schoener-coden-plsql-analysieren-mit-plscope_2.pdf*

Über den Autor

Peter van Garsel ist seit 1998 als externer Berater im Bereich Backend-Software-Development tätig. Unterschiedlichste Projektkontexte hat er als Oracle Database10g Administrator Certified Professional (OCP, 2004), Status: retired, beziehungsweise als Oracle Advanced PL/SQL Developer Certified Professional (OCP, 2014) unterstützt.



Peter van Garsel
p.garsel@eworking.de

www.cloudland.org

CloudLand

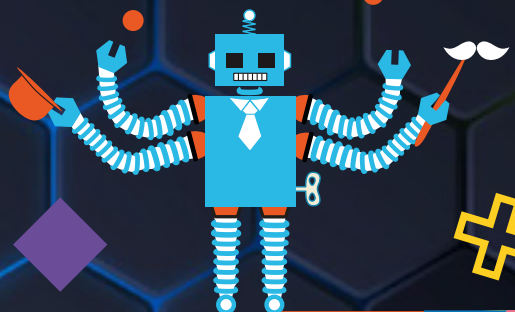
DAS FESTIVAL DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY 2023

• Microservices & DDD • CI/CD & Automatisierung
• Container & Cloud-Technologien • DevOps & Methodik

20. - 23. JUNI
im Phantasialand in Brühl

TAG 3

GAMING NIGHT



#CloudLand2023

Eventpartner: Heise Medien

BEST OF DOAG ONLINE

Eine Auswahl der besten DOAG News April/Mai 2023

Das war die APEX connect 2023

Die neunte Ausgabe der Fachkonferenz für Oracles Application Express glänzte durch ein ausverkauftes Haus, strahlenden Sonnenschein und außerordentlich gute Stimmung.

Wie war das gleich mit dem „Traditional Auditing“ und „Unified Auditing“ bei Oracle 19c?

Auf unserer Noon2Noon-Security-Veranstaltung in Hannover Anfang März 2023 gab es einige Fragezeichen in den Köpfen zum Auditing der Datenbank.

DOAG.tv mit Beda Hammerschmidt – Über JSON und die Roadmoap

Der in den USA beheimatete Senior Director (Development) Oracle Database schildert seinen Weg zu JSON, bei dem auch IBM eine nicht ganz unwichtige Rolle gespielt hat.

"Rekrutierung in Zeiten des 'War for Talents' ist eine Herausforderung" – Interview mit Stefan Latuski

Der Keynote Speaker der CloudLand 2023 spricht über seinen Arbeitgeber, die Zukunft von KI und wie es zur gemeinsamen Keynote mit seiner Frau kam.

DOAG Datenbank Kolumne: Oracle Database 23c Free

DOAG Datenbank Kolumne: Oracle Database 23c Free

Delegiertenversammlung 2023: DOAG stellt sich modern und breit auf

Am vergangenen Samstag hat die Delegiertenversammlung der DOAG wichtige Weichen gestellt. Im Rahmen der Öffnung zu neuen Themen wurde eine weitere Community gegründet und der Vorstand neu gewählt.

Wir begrüßen unsere neuen Mitglieder

Natürliche Mitglieder:

- Florian Polster
- Dr. Harald Gerhards
- Mirko Freudenberger
- Daniel Maier
- Dr. Jiayue He
- Werner Ewald
- Roman Mandjarov
- Raphael Salguero
- Abhishek Arora
- Johannes Ubrig
- Michael Schweiker
- Grzegorz Janowski
- Dr. Dirk Noetzold
- Tobias Schnittger
- Bartos Helwing
- Dr. Mark Noetzold

Korporative Mitglieder:

- JSteadforce GmbH, Repräsentantin: Inge Kleinert
- Controlware GmbH, Repräsentant: Jörg Bechtel

Termine

Juni 06

09.06.2023
ansible-oracle: Deployment von Oracle Datenbankserver mittels Ansible
DB WebSession mit Thorsten Bruhns, Senior Solution Architect, Opitz Consulting Deutschland
Online

13.06.2023
Oracle Exadata Insights: AIOps for Optimizing Resources
DOAG IMC WebSession with Murtaza Husain and John Beresniewicz, Oracle
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

Juli 07

05. - 06.07.2023
Expertenseminar: Oracle APEX und Oracle Rest
Berliner Expertenseminar mit Marco Patzwahl
Berlin

August 08

30. - 31.08.2023
Expertenseminar: Oracle Cloud Infrastructure – von der Konsole zur Automation – Kickstart!
Berliner Expertenseminar mit Martin Berger und Stefan Oehrli
Berlin

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): Martin Berger, Marcel Boermann-Pfeifer, Axel Bröker, Randolf Eberle-Geist, Jonas Gassenmeyer, Timo Herwix, Dr. Thomas Karle, Jan Karremans, Christian Linck, Marcos López, Martin Meyer, Sinan Petrus Toma, Frank Prechtel, Matthias Sauer, Dr. Simon Schafheitle, André Sept, Dani Schnider, Jürgen Sieben, Günther Stürner, Peter van Garsel, Prof. Antoinette Weibel, Armin Wildenberg, Jérôme Witt.

Titel, Gestaltung und Satz:
Diana Tkach
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

Fotonachweis:
Titel: © ankitasiddiqui | *www.pixabay.com*
S. 8: © WorldInMyEyes | *www.pixabay.com*
S. 14: © Skitterphoto | *www.pixabay.com*
S. 22: © Shotkitimages | *www.pixabay.com*
S. 26: © akitada31 | *www.pixabay.com*
S. 28: © dropolto | *www.pixabay.com*
S. 32: © vsalgado23 | *www.unsplash.com*
S. 38: © PublicDomainArchive | *www.pixabay.com*
S. 44: © jankosmowski | *www.pixabay.com*
Titel S. 52: © artvizual | *www.pixabay.com*
S. 56: © rawpixel.com | *www.freepik.com*
S. 57: © fullvector | *www.freepik.com*
S. 62: © Sketchepedia | *www.freepik.com*
S. 66: © freepik | *www.freepik.com*
S. 70: © schuetz-mediendesign | *www.pixabay.com*
S. 76: @ wal_172619 | *www.pixabay.com*
S. 89: © falco | *www.pixabay.com*
S. 97: © *https://rideafrica.org/meet-the-team*
S. 102: © *https://rideafrica.org/meet-the-team*
S. 105: © 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

#CloudLand2023
www.cloudland.org

CloudLand

DAS FESTIVAL DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY 2023

• Microservices & DDD • CI/CD & Automatisierung
• Container & Cloud-Technologien • DevOps & Methodik

20. - 23. JUNI
im Phantasialand in Brühl

TAG 4

CUSTOMER STORIES ROUNDTABLE

MAIN SPONSORS

MAYFLOWER

Media Markt SATURN Technology

Summer Night powered by InterFace AG the face of informatics

QA|WARE SOFTWARE ENGINEERING

REWE DIGITAL

Eventpartner Heise Medien