Nr. 6 | Dezember 2019 | ISSN 2366-7915 | www.doag.org

# Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP


## Aus der Praxis

Die eigene Cloud – Wann eine On-Premises-Lösung Sinn macht

## Im Interview

Hakan Aydas, Vodafone Group Services GmbH

## APEX

APEX und Oracle Autonomous Database


APEX connect
by DOAG
5. - 7. Mai 2020
in Brühl
DOAG
apex.doag.org

*Niels de Bruijn*
*Leiter der Development Community*

# Liebe Mitglieder, liebe Leserinnen und Leser,

der Erfolg von Oracle Application Express (APEX) ist längst nicht mehr zu stoppen. Allein in Deutschland arbeiten Hunderte Unternehmen mit APEX als Low-Code-Plattform. Ob Excel-Ablösung im Handel, Abrechnungssystem bei einem Telekommunikationsunternehmen oder Fondsdatenverwaltung bei einer Fondsgesellschaft: APEX ist für individuelle, datenzentrierte Webanwendungen eine Allzweck-Waffe im Unternehmen geworden.

Wir sollten allerdings nicht vergessen, dass die Oracle-Datenbank der Motor von APEX ist. Wobei APEX als kostenlose Option der Datenbank daherkommt, muss die Datenbank selbst lizenziert werden. Teuer muss dies allerdings nicht sein: für den Betrieb mit APEX On-Premises reicht in der Regel die kleine SE2 Edition der Datenbank aus. Im Cloud-Umfeld ist dagegen die „Free Tier" auf cloud.oracle.com sehr zu begrüßen. Endlich bekommt ein jeder die Möglichkeit, APEX-Anwendungen mit effektiv etwa 12 GB an Datenhaltung kostenlos produktiv in beispielsweise Frankfurt durch Oracle betreiben zu lassen. Das Ganze natürlich zwei Mal, damit man auch direkt eine Entwicklungsumgebung innerhalb von wenigen Minuten zur Verfügung hat.

Nicht nur APEX in der Cloud findet seinen Platz in dieser Ausgabe, auch Themen wie beispielsweise APEX 19.2, Exadata Cloud at Customer, Dev Tools, Cognitive Services, Workflow, ABTC oder PWA sind vertreten. Mein Dank gilt daher insbesondere allen Autoren, die ihre Freizeit investiert haben, um dieses Wissen weiterzugeben.

Auch 2020 geht es wieder mit Vollgas weiter, da wir sehr viele Aktivitäten für die Community geplant haben. Sicherlich ein Highlight ist hier die APEX connect, wo alle APEX-, PL/SQL- und JavaScript-Begeisterten sich vom 5. bis 7. Mai im Phantasialand treffen. Also, spätestens bis dann!
2020 kann kommen!

Viel Spaß beim Lesen wünscht Euch


*Mit den Azure Cognitive Services bietet Microsoft ein interessantes Angebot im Bereich Machine Learning as a Service an.*


*Progressive Web Apps (PWAs) sorgen dafür, dass der Übergang zwischen dem Web und klassischen Apps immer mehr verschwindet.*


*Code-Reviews stellen eine wichtige Maßnahme der Qualitätssicherung dar.*

## Einleitung


3 Editorial

5 Timeline

7 „Ein ganz großer Vorteil, den uns APEX bietet, ist die enorme Flexibilität der Applikation."
Interview mit Hakan Aydas


## APEX


9 APEX und die Azure Cognitive Sevices APEX-Anwendungen mit Machine Learning as a Service erweitern
Fabian Neureiter

17 APEX und Oracle Autonomous Database
Ulrike Schwinn

23 Progressive Web Apps mit APEX
Till Albert

28 Know your Browser Dev Tools!
Daniel Hochleitner

36 APEX 19.2: Was ist neu?
Carsten Czarski

59 Ein APEX Workflow-Tool für Citizen Developer
Michael Weinberger


## Cloud


41 Die eigene Cloud – Wann eine On-Premises-Lösung Sinn macht
Thomas Nau, Janne Schulz


## Datenbank


14 OOW 2019 Newsflash: Exadata in der Cloud
Frank Schneede

66 Lieblingskind oder Totgeburt?
Holger Bär


## PL/SQL


47 Aber das hat gestern noch funktioniert! Testing mit utPLSQL
Samuel Nitsche

53 Kampf den Jugendsünden – Ein Plädoyer für Code-Reviews
Tobias Wirtz


## Aktuelles


72 DOAG Botschafter für Technologie 2019: Johannes Ahrends
Martin Meyer


## Intern


73 Termine

73 neue Mitglieder

74 Impressum

74 Inserenten

# Timeline

## 08. Oktober 2019

*Der Call for Papers für die DOAG 2020 Datenbank am 25. und 26. Mai hat begonnen. Referenten können bis zum 15. Januar 2020 Vortragsthemen einreichen. Die Veranstaltung findet wie in den Vorjahren im Van der Valk Airporthotel in Düsseldorf statt. Neben der Datenbank stehen auch die Engineered Systems – vor allem Oracle Exadata und Oracle Database Appliance (ODA) – auf der Agenda.*

## 15. Oktober 2019

*Das Berliner Expertenseminar zum Thema Oracle Cloud Infrastructure mit Martin Berger findet statt. Die Angebotspalette in der Oracle Cloud wächst stetig. Mit den Angeboten Compute und Database Service sind zwei Produkte dabei, welche sich hervorragend für einen Datenbankbetrieb in der Oracle Cloud eignen. In diesem 2-tägigen Workshop lernen die 14 Teilnehmer die Cloud Infrastructure-Produktpalette kennen, setzen selber Server und Datenbankinstanzen auf, transferieren Daten lokal in die Cloud und wissen diese zu verwalten. Daneben werden auch Themen wie Sicherheit, Migration und die Integration in bestehende On-Premises-Umgebungen behandelt.*

## 15. Oktober 2019

*Der DOAG Security Day in Hamburg bietet praxisnahe Vorträge zu verschiedenen Aspekten der Datenbank-Security. Die 33 Besucher erfahren jede Menge Neues über Entwickler und DBAs im Security-Dilemma, Oracle Data Safe, Transparent Data Encryption für Non-CDB und die Multitenant Architektur, Privilege Capturing und DBSAT Version 3. Auf der Agenda steht auch die Frage, wie es zwei Jahre nach Inkrafttreten der Europäischen Datenschutz-Grundverordnung (DSGVO) mit diesem Thema weitergeht.*

## 22. Oktober 2019

*Der DOAG MultiCloud Day in Frankfurt am Main steht an. Die Veranstaltung bietet den rund 30 Teilnehmern einen Überblick über konkrete Lösungsansätze und den aktuellen Stand der MultiCloud-Technik im Zusammenhang mit Oracle-Datenbanken. Zu diesem Zweck versammeln sich mehrere Cloud-Anbieter, Integratoren und Co-Location-Betreiber in den Räumen des DE-CIX. Vorgestellt werden die MultiCloud-Ansätze von Oracle OCI, Microsoft Azure und Google Cloud sowie tatsächliche Multi-Cloud-Integrationsprojekte. Im Anschluss können die Besucher im Rahmen des Abendevents das Interxion Rechenzentrum in Frankfurt besichtigen und dabei einen Blick hinter die Kulissen werfen.*

## 23. Oktober 2019

*Am DOAG Monitoring Day erwartet die 20 Besucher ebenfalls in Frankfurt am Main ein spannendes Vortragsprogramm rund um das Thema Oracle Monitoring. Vorträge zu Monitoring der Oracle-Datenbank mit Oracle-Boardmitteln, Oracle Monitoring mit Elasticsearch, Zentrales Monitoring und Analytics mit Splunk sorgen für einen spannenden, informativen Tag.*

## 08. November 2019

*Das DOAG Datenbank Webinar "Patching und Upgrade aktueller Datenbankversionen" findet statt. Thilo Künkel von Opitz Consulting zeigt bekannte Fallstricke und Lizenzierungsfallen, gibt Tipps und Tricks beim Patching und Upgrade aktueller Datenbankversionen und zum Umgang mit dem Oracle-Support.*

## 19. bis 22. November 2019

*Das Event des Jahres findet erneut in Nürnberg statt. Die DOAG 2019 Konferenz + Ausstellung bietet ein dreitägiges umfassendes Vortragsprogramm mit internationalen Top-Rednern. Am Schulungstag kann das neue Wissen direkt in die Praxis umgesetzt werden. An allen drei Konferenztagen bietet Oracle Workshops zum Thema "Be ready for Oracle Support: My Oracle Support Accreditation Workshop" an. Das Ziel ist es, die Zusammenarbeit mit dem Oracle Support zu verbessern. Die Teilnehmer erhalten Best Practices und Empfehlungen von Oracle-Experten in My Oracle Support, um so die Stabilität Ihrer Systeme durch die Nutzung der proaktiven Werkzeuge in My Oracle Support zu erhöhen. Außerdem geht es um die Optimierung der Problemidentifizierung und -behebung mittels My Oracle Support.*

## 19. November 2019

*In der Eröffnungsrede zur DOAG Konferenz + Ausstellung spricht der DOAG-Vorstandsvorsitzende Stefan Kinnen über Veränderungen und Herausforderungen, die digitale Trends auf Gesellschaft, Wirtschaft und auch auf die DOAG haben.*

*Jürgen Kunz (Oracle) macht in seiner "Welcome Keynote Oracle" auf die derzeitigen technologischen Herausforderungen in den Unternehmen aufmerksam und hebt die Künstliche Intelligenz (KI) und ihre disruptiven Auswirkungen hervor. Darüber hinaus thematisiert er die aktuelle Cloud-Strategie von Oracle. Gerhard Schlabschi, Oracle EMEA Competitive Intelligence, erläutert, welche Gründe für die "Oracle Cloud Services" sprechen und veranschaulicht deren bisherige Akzeptanz.*

## 19. November 2019

*In der vom ehemaligen DOAG-Vorstandsvorsitzenden und Ehrenmitglied Dr. Dietmar Neugebauer moderierten Podiumsdiskussion mit dem DOAG Legal Council auf der DOAG 2019 Konferenz + Ausstellung berichten die Rechtsanwälte Dr. Jana Jentzsch, Dr. Jan Bohnstedt, Dr. David Bomhard , Dr. Ivo Rungg, Dr. Michael Isler und Michael Paege, Leiter des DOAG Competence Center Lizenzierung über die Auswirkungen des „US Cloud Act", über die Streichung des RAC aus der SE2, präsentieren das Ergebnis des Gutachtens zum Matching Support Level und die Änderungen in den Bedingungen für das Oracle-Lizenzaudit.*

## 19. November 2019

*Im Rahmen der DOAG-Konferenz laden die Austrian Oracle User Group (AOUG) und die Swiss Oracle User Group (SOUG) Teilnehmer aus Österreich und der Schweiz zum AOUG und SOUG Community-Abend ein.*

## 20. November 2019

*Lars Thomsen gibt in seiner Keynote „520 Wochen Zukunft" einen Ein- und Ausblick auf Veränderungen, Trends und Tipping Points in Gesellschaft, Wirtschaft und Technologie bis 2029.*

## 21. November 2019

*In seiner Keynote mit dem Titel "Geheimwaffen der Kommunikation: Sanfte Strategien mit durchschlagender Wirkung" macht Leo Martin den Besuchern die Erfolgsfaktoren wirksamer Kommunikation sichtbar.*

## 22. November 2019

*Die DOAG 2019 Konferenz + Ausstellung, größte Oracle-Konferenz in Europa, geht mit dem Schulungstag zu Ende. Fried Saacke, Vorstandsmitglied und Leiter der DOAG-Geschäftsstelle, zieht ein positives Fazit: „Auch in diesem Jahr ist es uns gelungen die Teilnehmerzahlen weitgehend stabil zu halten. Besonders freuen mich daher die vielen positiven Rückmeldungen, die mich erreicht haben. Dies zeigt, dass es uns wieder gelungen ist, die richtigen Themen anzusprechen. Ich danke dafür allen, die an dem Gelingen der Veranstaltung tatkräftig mitgewirkt haben, den Leitungskräften aus dem Verein, den zuverlässigen Partnern und dem engagierten Team aus der DOAG-Geschäftsstelle. Mein Dank gilt auch allen Referenten und Ausstellern. Ich freue mich schon jetzt auf die gemeinsame Gestaltung der DOAG 2020."*

## 28. November 2019

*Das Organisationsteam der DOAG-Geschäftsstelle hält in Berlin eine Feedback-Runde zur DOAG 2019 Konferenz + Ausstellung. Alle eingegangenen Kommentare und Vorschläge werden bewertet und daraus bereits die Weichen für eine erfolgreiche Jahreskonferenz 2020 gestellt.*

*Dr. Dietmar Neugebauer*
*Ehemaliger DOAG-Vorstandsvorsitzender*

## Aus der Ferne betrachtet: Oracle Application Express, einfach ...

eine geniale Möglichkeit, um Abschied zu nehmen von Excel-Dateien, die in Unternehmen per Mails verschickt und auf den verschiedensten Laufwerken in den unterschiedlichsten Versionen den Mitarbeitern zur Verfügung gestellt werden. Mit Oracle Application Express gibt es nur eine Quelle der Datenhaltung mit allen Vorzügen einer Datenbank; auf die dort bereitgestellten Anwendungen kann über den Browser vom PC oder vom Smartphone aus einfach zugegriffen werden.

Hinzu kommt, dass hier gerade mit dem neuesten Release eine sehr einfache Entwicklungsumgebung zur Verfügung gestellt wird. In dieser modernen Entwicklungsumgebung ist man in der Lage, mit „Drag & Drop"-Anwendungen schnell zu entwickeln. Oracle APEX steht kostenfrei zur Verfügung und kann aus dem Oracle Store heruntergeladen werden. Gerade Fachabteilungen profitieren gerne von dieser Lösung.

Wieso ausgerechnet Fachabteilungen? Warum wird gerade in großen Unternehmen Oracle APEX nicht als strategische Plattform eingesetzt?

Hier zählt sicherlich zu den Gründen, dass trotz des kostenlosen Einsatzes von APEX die Kosten für die Nutzung der Oracle-Datenbank nicht zu vernachlässigen sind. Weiterhin kommt hinzu, dass aufgrund der im Vergleich zu anderen Entwicklungsumgebungen und Datenbanken geringen Präsenz von Oracle-Produkten im Lehr- und Ausbildungsbereich hier ein Mangel an Fachkräften besteht. Einfachere Anwendungen sind mit APEX zwar schnell entwickelt, aber gerade komplexere Anwendungen in größeren Unternehmen verlangen doch ein tieferes Wissen in der Software-Entwicklung.

Diesem schon länger angesprochenen Defizit will Oracle mit der Ankündigung und der bereits vorhandenen Verfügbarkeit der „Free Oracle Autonomous Database" begegnen. Wie auch Joel Kallmann, der Verantwortliche für die APEX-Software-Entwicklung, auf Twitter und in seinem Blog vorstellt, bekommt jeder innerhalb von fünf Minuten kostenlos eine Oracle-Datenbank mit Oracle Application Express in der Cloud installiert, vollständig eingerichtet und so kann man sofort mit der Entwicklung von APEX-Anwendungen beginnen. Dazu hat man noch die Möglichkeit, mit SQL Developer oder direkt mit SQL*PLus und SQLcl darauf zuzugreifen. Dies sollte gerade für den IT-Ausbildungsbereich ein attraktiver Einstieg sein. Ist nur zu hoffen, dass diese Nachricht auch an den Ausbildungsstätten ankommt – das ist sicherlich eine Aufgabe für Oracle, aber auch für die DOAG und ihre Mitglieder, dort als Multiplikatoren aktiv zu werden. Die DOAG-Botschafterin Ulrike Schwinn hat im vorliegenden Red Stack Magazin in ihrem Artikel die Grundlagen dafür beschrieben.

Bleibt zum Schluss nur noch ein Kritikpunkt – die Oracle-proprietäre Entwicklungsumgebung und Datenbank. Gerade heute, wo die Open-Source-Produkte auch in großen Unternehmen Einzug gehalten haben und auch den Schwerpunkt in der Ausbildung einnehmen. Wie würden wir es nennen, wenn Oracle sich hier öffnen würde? Doch wohl

.... einfach Open Application Express!


# „Ein ganz großer Vorteil, den uns APEX bietet, ist die enorme Flexibilität der Applikation."

Niels de Bruijn, DOAG Vorstand Development, und Martin Meyer, Redaktionsleiter des Red Stack Magazin, sprachen mit Hakan Aydas, Manager of Performance & Efficiency bei der Vodafone Group Services GmbH, über den Einsatz von Oracle Application Express (APEX).

*Was sind Ihre Aufgaben bei Vodafone?*
Mein Kernaufgabe bei der Vodafone Group ist, im internationalen Datacenter-Betrieb alle relevanten Infrastruktur-Kern-Komponenten (z. B. Storage, Compute, IP Network, Datenbanken und Big Data) über multiple Core Tools zu verbinden und die daraus gewonnenen Informationen in einem großen Data Lake vorzuhalten. Entsprechend den an uns herangetragenen Anforderungen werden aus diesem Data Lake mit Zuhilfenahme unserer APEX-Lösung Applikationen, Dashboards oder auch andere Lösungen entwickelt und sowohl unserem Senior-Management als auch unseren internen Kunden zur Verfügung gestellt.

*Wie sind Sie mit Oracle Application Express in Kontakt gekommen?*
In Kontakt mit APEX kam ich das erste Mal im Jahr 2014, während wir intern nach einer Lösung gesucht haben, unser Reporting auf eine einheitliche, zentrale, standardisierte Struktur umzustellen. Bis in das Jahr 2014 hinein wurden unsere Infrastruktur, KPIs, Dashboards und Reports auf multiplen, nicht zentralen Lösungen angeboten, ohne Standards und jegliche Datenstruktur dahinter.

*Welche Lösungen auf der Basis von APEX haben Sie im Einsatz?*
Die aktuellen Lösungen, mit denen wir tagtäglich arbeiten und die wir anbieten, starten bei simplen KPI Dashboards und gehen bis zu komplexen Infrastruktur-Provisionierungs-Applikationen wie z. B. für unsere Storage-, Datenbank- und Datacenter-Rack-Space-Umgebungen. Zudem unterstützen wir mit unseren APEX-Lösungen größere hausinterne Initiativen wie unser Infrastruktur-Lifecycle-Management. APEX hilft uns bei der „Cloudifizierung" unserer IT-Umgebung, bei der Gegenüberstellung der Private/Public Cloud und unserer Legacy-Umgebung.

*Was macht aus Ihrer Sicht den Erfolg von APEX in Ihrer Abteilung aus?*
Ein ganz großer Vorteil, den uns APEX bietet, ist die enorme Flexibilität der Applikation. Damit sind wir in der Lage, extrem schnell auf Anforderungen unseres Senior-Managements oder unserer internen Kunden zu reagieren und entsprechend schnell Lösungen anzubieten.

*Wo sehen Sie noch Verbesserungspotenzial für APEX?*
Ich wünsche mir für die Zukunft einen stärkeren Fokus auf UI/UX. Leider sehen viele APEX-Anwendungen immer noch wenig modern aus.

*Können Sie dies konkretisieren? Auch bei APEX 19.1 finden Sie die UI/UX von APEX-Anwendungen nicht ausreichend? Betrifft dies die Navigation oder die unzureichende Benutzung des Bildschirms?*
Wir haben mit 19.1 noch so gut wie keine Erfahrungen. Daher kann ich lediglich auf unsere aktuelle Version 5.1.4. und die vorhergegangenen Versionen referenzieren. Es betrifft weniger die Navigation, sondern unter anderem die Skalierung der Anwendungen am Bildschirm. Einen weiteren Punkt sehe ich darin, dass wir immer noch auf Plug-ins zugreifen müssen, um diverse Charts oder Frames darstellen zu können. Für Entwickler, die wissen, wo sie die richtigen Plug-ins finden, ist dies kein großes Thema. Entwickler, die hingegen nicht so tief in den Essentials drin sind, mühen sich mit der Entwicklung von Workarounds ab. Als Nächstes sehe ich in der 5.1.4.-Version immer noch das Problem der Farbspektren. Die Auswahl an implementierten Standard-Farben beschneidet uns bei komplexeren Darstellungen bezüglich der Erkennbarkeit.

*Welche Fähigkeiten bringt aus Ihrer Sicht ein guter APEX-Entwickler mit?*
Ein guter APEX-Experte sollte nach meiner Erwartungshaltung definitiv Agilität und Flexibilität mitbringen, um schnell auf Anforderungen reagieren zu können. Technische Skills wie SQL, HTML, Oracle-Datenmodell, SQL-Tuning gehören zu den Main Skills, die unabdingbar für einen Entwickler sind. Bedingt dadurch, dass APEX-Entwickler meist in kleineren Teams arbeiten, sollten sie auch Fähigkeiten mitbringen, um als Berater auftreten zu können. Sie sollten das Managen von Projekten durch agile Arbeitsmethoden beherrschen und dabei auch über sehr gute kommunikative Fähigkeiten verfügen, um mit Stakeholdern und dem Senior-Management zu kommunizieren.

*Welches Vorgehensmodell wird bei der APEX-Entwicklung verwendet?*
Zunächst einmal muss eine Analyse der gestellten Anforderungen vorgenommen werden. Danach ist zu prüfen, ob die benötigten Daten bereits Bestandteil unseres Data Lake sind. Ist dies nicht der Fall, ist ein Datenmodell zu entwickeln. Anschießend müssen die Quelldaten evaluiert und ein automatisierter Datenfluss mittels Standard-APIs oder Ladeskripten eingerichtet werden. Die ersten Entwürfe der Dashboards werden im Folgenden mithilfe von APEX auf der Test-/Dev-Umgebung entwickelt und danach mit dem Auftraggeber abgestimmt. Nach einer abgeschlossenen Entwicklung wird mit Continuous Delivery die Lösung auf die produktive Umgebung geladen.

*Ist die Cloud ein Thema in Ihrem Bereich?*
Bei der Vodafone genießt das Thema Cloud sehr große Aufmerksamkeit. Das Angebot für unsere Ländergesellschaften umfasst aktuell sowohl Private- und Public- Cloud-Lösungen als auch Hybrid-Cloud-Ansätze. Unsere APEX-Lösung spielt auch hier eine zentrale Rolle für die Erfassung und Darstellung unserer Cloud-KPIs und unseres „Cloudifizierungs"-Fortschritts.

*Oracle tut sich nach wie vor schwer, wenn es um die Lizenzierung von Oracle-Software in einer virtuellen Umgebung geht. Wie bewerten Sie dieses Thema?*
Nach meiner Ansicht werden wir in Zukunft um diese Art von Lizenzmodellen gar nicht herumkommen. Die voranschreitende weltweite „Cloudifizierung" großer IT-Landschaften wird immer weiter vorangetrieben, sodass sich Oracle frühzeitig mit dem Lizenzthema beschäftigen sollte.

## Zur Person: **Hakan Aydas**

Hakan Aydas startete seine Karriere 1999 bei der Mannesmann Mobilfunk im Bereich IT-Infrastruktur Monitoring.

Im Rahmen einer europaweiten Datacenter-Konsolidierung, wechselte er zur EITO (European IT Operations) der Vodafone Group und wirkte dort maßgeblich am Aufbau des Datacenter übergreifenden IT-Infrastruktur Capacity Managements mit.

Im Jahr 2013 übernahm er als Manager die Leitung der neu gegründeten Abteilung „Performance & Efficiency" bei der Vodafone Group. Hier stellte er sich der Herausforderung, multiple Datenquellen und die Daten aus den operativen Kern-Applikationen in einen zentralen Data Lake zusammenzuführen.

Während dieses Projektes wurde er auf eine kleine vorhandene APEX-Lösung aufmerksam und entschied im Rahmen eines Piloten, die Visualisierung diverser Dashboards und Applikationen mittels APEX zu realisieren.

Diese Lösung wurde nach sechsmonatiger Entwicklungsarbeit dem Senior Management im Rahmen eines Management Workshops vorgestellt und erhielt umgehend den Zuspruch des Head of Operations.

Hakan Aydas bekam die benötigten Ressourcen bereitgestellt, um die vorhandene neue APEX- Plattform weiterzuentwickeln und zusätzliche Funktionalitäten zu implementieren.

Das zwölfköpfige multinationale virtuelle Entwicklerteam arbeitet parallel an der Realisierung internationaler Projekte, um diese innerhalb der APEX-Lösungen zu verwirklichen.

# APEX und die Azure Cognitive Sevices

# APEX-Anwendungen mit Machine Learning as a Service erweitern

Fabian Neureiter, MT AG

In der breiten Öffentlichkeit, aber auch in Kreisen professioneller Technik-Interessierter, gibt es aktuell wenig, was so heiß diskutiert wird wie Machine Learning und künstliche Intelligenz. Mal geht man von einer existenziellen Bedrohung für die Menschheit à la Terminator aus, ein anderes Mal von einer Art eierlegenden Wollmilchsau, die alle irdischen Probleme zu lösen vermag. Beide Schlussfolgerungen basieren auf den enormen Fähigkeiten der KI-Techniken. Der Einsatz dieser Techniken ist heutzutage so einfach wie nie. Wo früher noch auf Teams von Data Scientists zurückgegriffen werden musste, da das entsprechende Wissen so speziell war und die verfügbaren Technologien so viel Einarbeitungszeit benötigten, sind wir heute im Zeitalter des Cloud Machine Learning, dem Machine Learning as a Service (MLaaS), angekommen.

Beim MLaaS werden verschiedene Schritte des Machine Learning, wie das Vorverarbeiten der auszuwertenden Daten, das Training der Modelle und Ähnliches, vor dem Nutzer verborgen. Der User speist seine Daten lediglich in die Cloud ein und erhält das entsprechende, in der Cloud berechnete Resultat zurück. Dies ermöglicht eine einfache Erweiterung von Anwendungen um maschinelles Lernen per REST-Schnittstelle.

Jeder der großen Cloud-Anbieter stellt mittlerweile KI-Dienste zur Verfügung: Zum Beispiel bietet Google die Google ML Cloud an, Amazon die KI Services, IBM das Watson Machine Learning und Microsoft unter anderem die Cognitive Services.

Spätestens seit Oracle und Microsoft im Juli 2019 eine Kooperation angekündigt haben *[1]*, ist es sinnvoll, sich als APEX-Entwickler das Cloud-Angebot von Microsoft genauer anzuschauen. Insbesondere im Bereich künstliche Intelligenz ist die Azure-Plattform breit aufgestellt.

So wird maschinelles Lernen als Drag & Drop-Oberfläche (**Azure Machine Learning Studio**) angeboten, was die Erstellung von einfachen ML-Lösungen selbst mit nur rudimentären Data-Science- und Programmierkenntnissen ermöglicht. Dabei können verschiedene Bausteine so miteinander kombiniert werden, dass ein Workflow zur Datenauswertung entsteht.

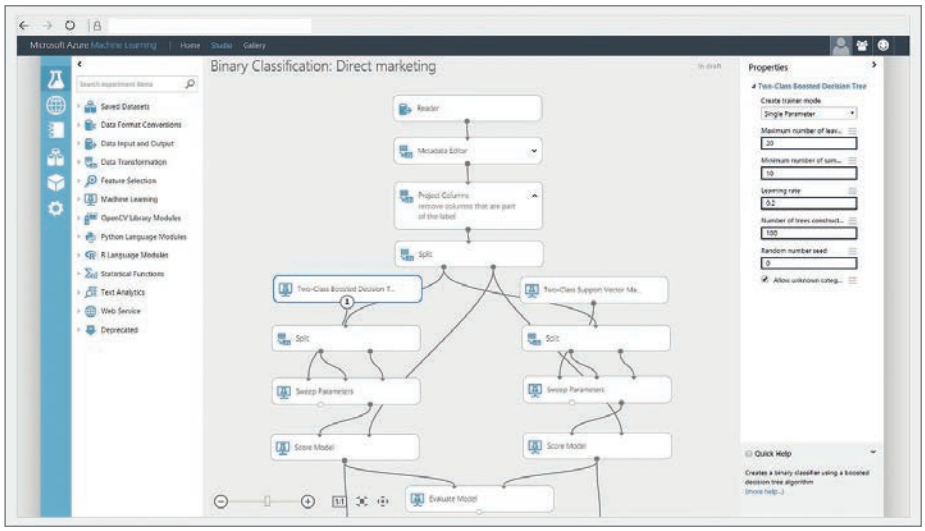



*Abbildung 1: Die Oberfläche des Azure Machine Learning Studio (© https://azure.microsoft.com/de-de/services/machine-learning-studio)*

Diese Vereinfachung hat jedoch zur Folge, dass das flexible Manipulieren von Modell-Parametern sowie die Anpassungsmöglichkeit des Programmcodes, auf dem die einzelnen Bausteine basieren, nicht mehr möglich ist.

Der eigene Bereich **Azure Machine Learning** geht einen anderen Weg. Dort, wo im Machine Learning Studio nur die grafische Oberfläche zur Verfügung steht, wird hier zwar ebenfalls eine (reduzierte) UI geboten. Es ist jedoch ebenso möglich, über ein SDK Modelle komplett im Code zu erstellen, vorzugsweise in Python. Dabei werden die gängigsten Frameworks, etwa Pytorch und Tensorflow, bereits unterstützt.

Eine dritte Möglichkeit, Microsofts KI-Dienste zu nutzen, bieten die **Azure Cognitive Services**. Hierbei handelt es sich um vortrainierte Modelle, beispielsweise zur Erkennung natürlicher Sprache oder von Gesichtern, die per REST-API abgefragt werden können und deren Fähigkeiten somit relativ leicht in bestehende Anwendungen integrierbar sind. Dadurch werden sie für einen APEX-Entwickler durchaus interessant, da APEX vor allem durch die Web Source Modules eine Interaktion mit derlei Services ermöglicht. Es sind außerdem keine Kenntnisse im Bereich Machine Learning nötig, um diese Dienste erfolgreich einsetzen zu können. Was die Cognitive Services sind und wie sie mit APEX interagieren können, soll dieser Artikel anhand eines Beispiels demonstrieren: einer **Chat-Funktionalität**, die Nachrichten von verschiedensprachigen Teilnehmern übersetzen kann.

## Azure Cognitive Services

Zunächst aber ein kurzer Überblick über die Cognitive Services:

Microsoft gliedert diese in die fünf Aufgabengruppen **Bildanalyse, Spracheingabe, Sprache, Entscheidungshilfe** und **Suchen**.

## Bildanalyse

Die API-Schnittstellen der Bildanalyse bieten im Allgemeinen die Möglichkeit, aus Bildern Informationen zu gewinnen. Dabei kann es sich um Bilder mit Textinhalt, Bilder von Freihandschrift oder Bilder von Personen, insbesondere Gesichtern, handeln. Auch Bewegtbilder, also Videos, können ausgewertet werden. So greift der Video Indexer auf eine Kombination aus verschiedenen Bildanalysemethoden zurück. Er ist in der Lage, Gesichter zu erkennen und zuzuordnen, kann aber auch die dargestellte Szene auswerten und Bildelemente extrahieren. Beispiele dafür wären etwa das Geschlecht der zu sehenden Person und die Farbe gezeigter Gegenstände.

## Spracheingabe

Die Dienste der Spracheingabe ermöglichen zum einen die Transkription von Sprache und die Echtzeit-Übersetzung, zum anderen das Erkennen von Personen anhand von Sprachdaten. Dies kann zum Beispiel zur Verifikation über Sprache genutzt werden. Die automatische Verschriftlichung von Gesagtem wird vom Video Indexer nicht nur eingesetzt, um Transkripte eines Videos zu erstellen. Es können auch, zum Beispiel bei einer Rede, besprochene Themen erkannt und zum Erstellen entsprechender Tags verwendet werden.

Soll ein Text im Hinblick auf darin enthaltene Absichten oder auf einen bestimmten Kontext hin ausgewertet werden, bieten sich die Schnittstellen der Gruppe Sprache an. Das Textanalyse-API sowie das Language-Understanding-API eignen sich dafür ideal. Es kann aber auch Text übersetzt und beim Verständnis von Texten (geplant für die nahe Zu-

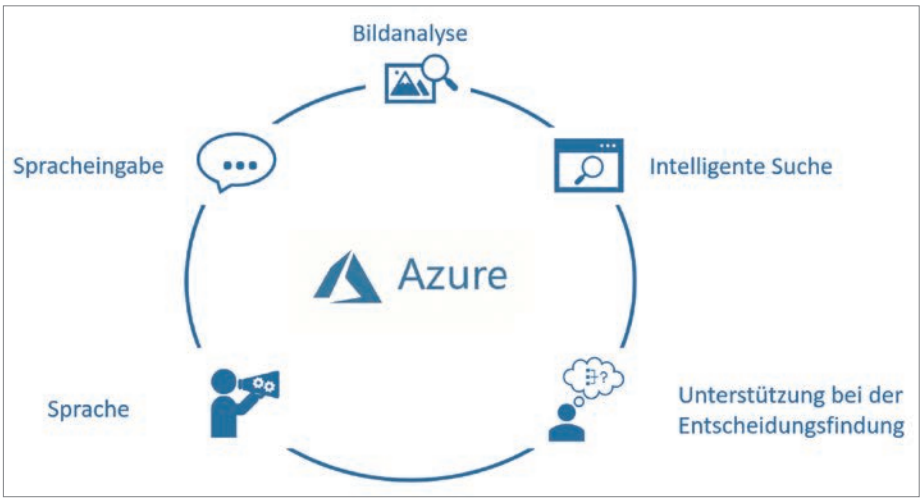



*Abbildung 2: Übersicht Cognitive Services (Azure Logo © Microsoft)*

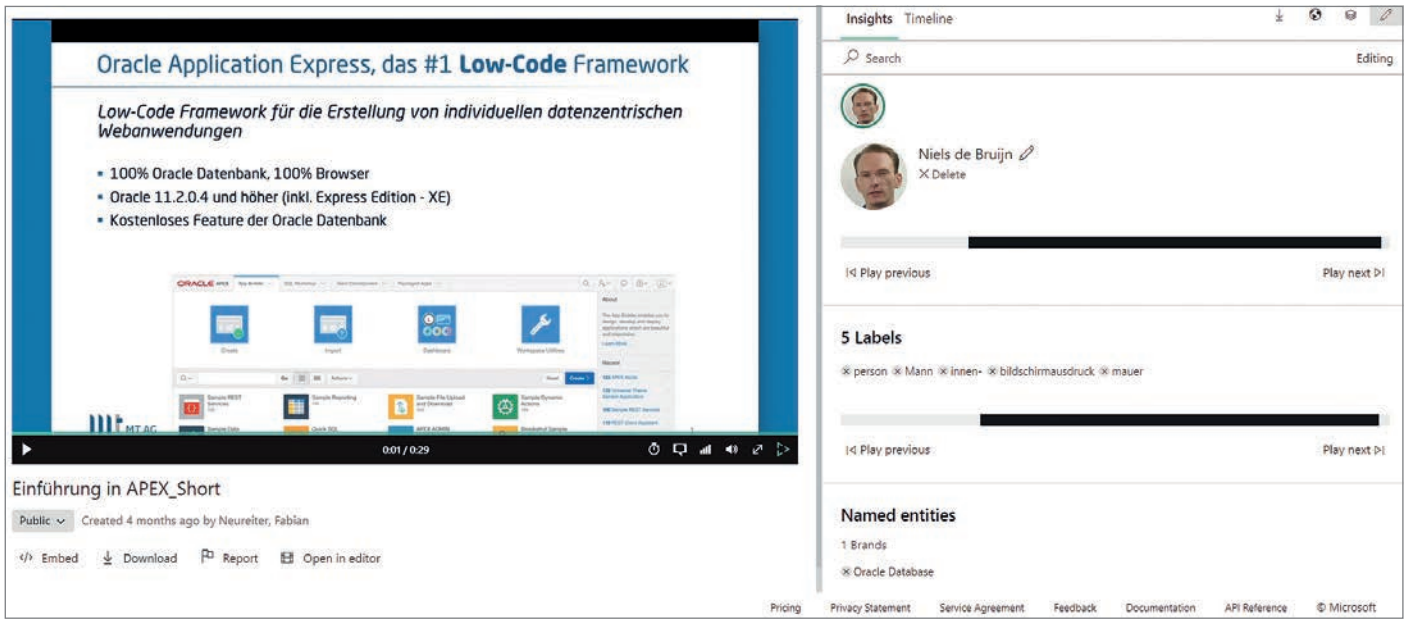



*Abbildung 3: Die UI des Video Indexer (Quelle: Fabian Neureiter)*

kunft) geholfen werden. Des Weiteren wird dem Use Case der selbstständigen Q&A-Erstellung eine eigene Schnittstelle, der Q&A Maker, zur Verfügung gestellt. Damit können aus Kundenanfragen die Intentionen und Problemfälle des Kunden automatisch erkannt und zugeordnet werden.

Ein weiteres Anwendungsgebiet maschinellen Lernens ist das Auswerten von Daten, um fundiertere Entscheidungen treffen zu können. Die Azure Cognitive Services bieten auch dafür APIs an. So können Anomalien in (Live-) Datensätzen erfasst werden. In Social-Media-Anwendungen ist es möglich, anstößigen Inhalt zu erkennen und herauszufiltern und im Anschluss den jeweiligen User zu verwarnen oder zu sperren.

Schließlich besteht noch die Möglichkeit, die Fähigkeiten von Microsofts hauseigener Suchmaschine Bing zu nutzen und diese per API in die eigene Anwendung zu integrieren. Es kann beispielsweise eine Schnittstelle angesprochen werden, die eine webbasierte Bilder- und Videosuche erlaubt, eine Suche nach benutzerdefinierten Kriterien wird jedoch auch angeboten.

Wie die Integration der genannten Dienste funktionieren kann, wird nun an einem Beispiel gezeigt.

```
[
   {
      "Text": "Hi, DOAG! It's very nice to meet you all. I sincerely
      hope you like Machine Learning!"
   }
]
```

*Listing 1: Beispiel für den Request-Body. Der Subscription Key wird im Header mitgesendet. Die gewünschte Sprache wird als Query-Parameter angehängt.*

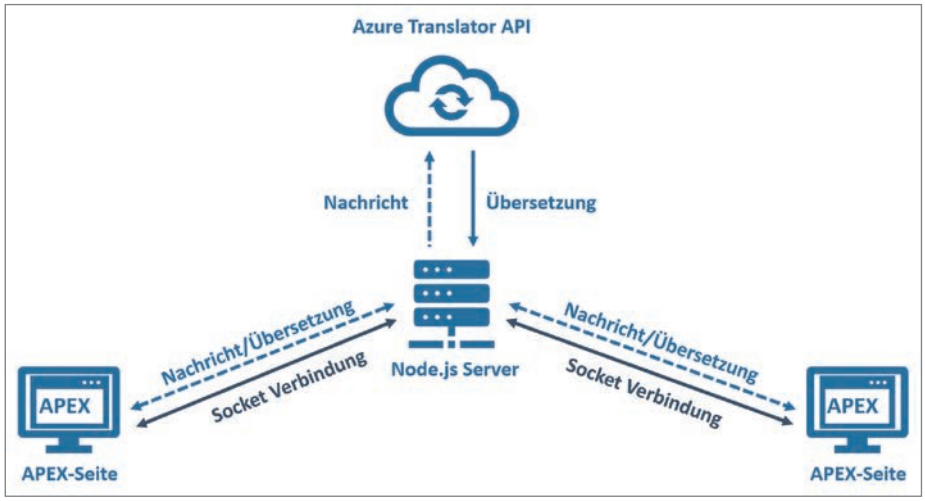



*Abbildung 4: Veranschaulichung des Aufbaus der Beispielanwendung (Quelle: Fabian Neureiter)*

## Integration des Translate-API in eine APEX-Anwendung

Zunächst soll es um eine Chatfunktionalität in APEX gehen. Da es durchaus vorkommen kann, dass in einem konzernweiten Chat verschiedensprachige Personen aufeinandertreffen, sollte es bei der Chatanwendung die Möglichkeit geben, die Nachrichten in eine vorher definierte „Muttersprache" zu übersetzen.

Zum Einsatz kommt dafür das Azure-Translate-API. Es wurde im Azure Portal als Ressource erstellt und damit auch ein Subscription Key generiert. Dieser Authentifizierungsschlüssel muss, wenn man keine Authentifizierung über Tokens verwenden möchte, bei jeder Anfrage im Header mitge-

```
[
   {
      "detectedLanguage": {
         "language": "en",
         "score": 1.0
      },
      "translations": [
         {
            "text": "Ciao, DOAG! È molto bello conoscervi tutti. Spero sinceramente che ti piaccia l'appren-
dimento automatico!",
            "to": "it"
         }
      ]
   }
]
```

*Listing 2: Der Response-Body des Translator-API. Zu sehen ist auch, dass die Sprache des Textes aus dem POST-Request automatisch korrekt erkannt wurde.*

sendet werden. Um einen Text übersetzen zu lassen, reicht es aus, einen POST-Request an das API zu senden *(siehe Listing 1)*.

Mehr ist für die Interaktion mit dem API nicht nötig. Darüber hinaus kann man noch Verschiedenes in Bezug auf die Art der Authentifizierung und das Übersetzen von mehreren Texten gleichzeitig einstellen, dafür soll aber auf die umfangreiche Dokumentation *[2]* verwiesen werden.

Für die Beispielanwendung wird außerdem noch ein minimaler Node.js-Server mit dem Framework Socket.io verwendet, Letzteres bringt verschiedene Funktionen zur Verwaltung von Socket-Verbindungen mit.

Socket-Verbindungen bieten eine persistente Verbindung von Client und Server, sodass unter anderem jederzeit Daten mit geringer Latenz hin- und hergeschickt werden können *[3]*. Dies eignet sich hervorragend für einen Chat. Jeder Client verbindet sich also mit dem Node.js-Server, dieser wiederum leitet die entsprechenden Nachrichten an die Sockets, die verbundenen Clients, weiter. Doch das ist nicht alles. Verbindet sich ein Client das erste Mal mit dem Server, liefert er eine Nachricht mit einem Nutzernamen und seiner „Muttersprache" mit. Soll an diesen Client nun eine Nachricht eines anderen Chatteilnehmers gesendet werden, der nicht dieselbe „Muttersprache" angegeben hat, wird die Nachricht an das Textübersetzungs-API der Cognitive Services gesendet.

Erhält der Node.js-Server vom API eine Antwort im JSON-Format *(siehe Listing 2)*, leitet er diese an den Zielclient weiter.

Ein Nutzer erhält somit alle Nachrichten in der von ihm angegebenen Sprache, unabhängig davon, in welcher Sprache sie ursprünglich verfasst wurden *(siehe Abbildungen 5 und 6)*.

Die Anbindung in APEX ist nun denkbar einfach. Abgesehen von der Erstellung eines User-Interface kann Socket.io clientseitig per JavaScript File URL-Feld im Page Designer entweder als externe Web-URL oder als lokales Workspace-File eingebunden werden. Da Socket.io eventbasiert arbeitet, müssen noch Events *(siehe Listing 3)* definiert werden, die die Nachrichten absenden und einkommende Nachrichten registrieren. In der Beispiel-

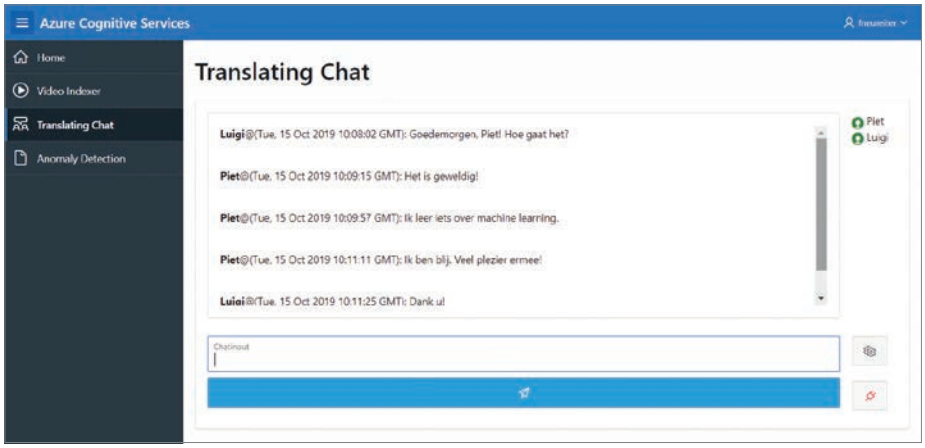



*Abbildung 5: Die Anwendung aus Sicht des fiktiven Nutzers „Piet" mit der Muttersprache Niederländisch (Quelle: Fabian Neureiter)*

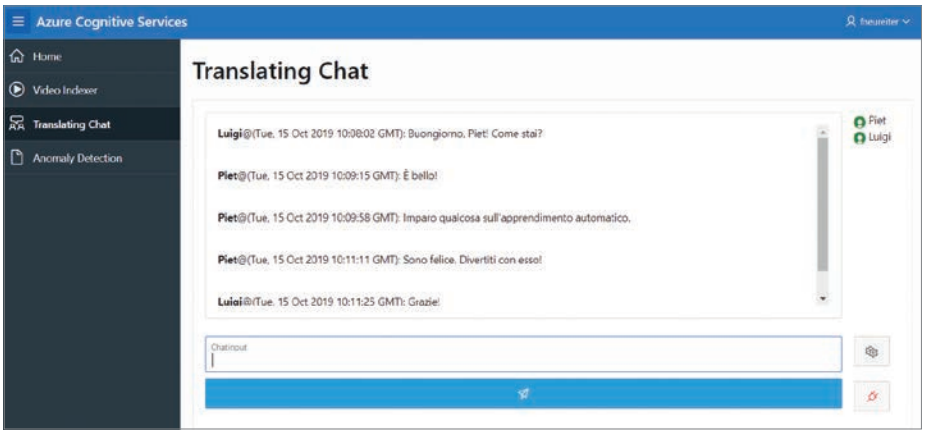



*Abbildung 6: Die Anwendung aus Sicht des fiktiven Nutzers „Luigi" mit der Muttersprache Italienisch (Quelle: Fabian Neureiter).*

anwendung wurden die Events vollständig in JavaScript erstellt und im JavaScript-Feld der Hauptseite im Page Designer hinzugefügt.

Eine Definition der Events über Dynamic Actions wäre aber ebenfalls möglich und in einer Produktiv-Anwendung auch wünschenswert.

Somit erhält man eine Anwendung in APEX, die eine grundlegende Chatfunktionalität bietet und KI-Dienste aus der Cloud nutzt.

Verlässt man sich nun aber darauf, dass das Translate-API immer korrekt und im korrekten Kontext übersetzt, werden sich die potenziellen Nutzer das ein oder andere Mal reichlich wundern. So werden in der Regel längere Texte ungenauer übersetzt als einzelne, kurze Sätze. Für eine Chat-Testanwendung ist dies jedoch noch vollkommen akzeptabel. Die hinter dem API stehenden Machine-Learning-Modelle werden außerdem laufend weiter trainiert, weswegen davon auszugehen ist, dass sich die Übersetzungsfehler mit fortschreitender Zeit immer weiter minimieren werden.

```
//Client verbindet sich mit Server
var socket = io.connect('Server-URL');
//Client teilt über die Socket-Verbindung Muttersprache und Nutzernamen mit
socket.emit('identify', {
      language: language.value,
      handle: handle.value
});
//Client registriert Chat-Event, zeigt erhaltene Nachricht an
socket.on('chat', function(data){
   let date = new Date();
   outputString = '<p><b>' + data.handle + '</b>@('+ date.toUTCString()
      + '): '
      + data.message
      + '</p><br>';
   display_region.innerHTML += outputString;
});
// Event, um Nachrichten zu versenden
socket.emit('chat', {
      handle: handle.value,
      message: input.value
});
```

*Listing 3: Die wichtigsten Events der Anwendung. Dies ist nur ein Beispiel, um einen ungefähren Eindruck zu vermitteln. Die tatsächliche Implementierung sieht anders aus.*

## Fazit

Mit den Azure Cognitive Services bietet Microsoft ein interessantes Angebot im Bereich Machine Learning as a Service an. Für die Verwendung der Dienste sprechen zunächst die einfache Verwendung und die stetige Weiterentwicklung seitens Microsoft. Dabei profitiert man als Nutzer auch von der Konkurrenz der Anbieter untereinander, insbesondere Google, Microsoft und Amazon. Jeder Plattformbetreiber versucht, die Nutzung seiner Dienste möglichst einfach bedien- und anpassbar zu gestalten. Gibt es außerdem neue oder verbesserte Techniken, beispielsweise eine neue Herangehensweise an Gesichtserkennung, hat es bis jetzt nicht lange gedauert, bis diese dem Nutzer zur Verfügung gestellt wurden. Dies ist in einem sich so schnell entwickelnden Themengebiet, wie dem des maschinellen Lernens, aber auch nötig. Sich mit der Azure Cloud zu beschäftigten, könnte jedoch besonders im Hinblick auf die geplante Cloud-Kooperation von Microsoft und Oracle interessant sein. Es braucht also keine Vorhersage anhand von Machine Learning, um ahnen zu können, dass uns eine spannende Zukunft bevorsteht.

## Quellen


[1] Microsoft News Center (2019, Juni 5). Microsoft and Oracle to interconnect Microsoft Azure and Oracle Cloud. Abgerufen 13. Oktober 2019, von *https://news.microsoft.com/2019/06/05/microsoft-and-oracle-to-interconnect-microsoft-azure-and-oracle-cloud/*

[2] Dokumentation Translator-API. Abgerufen 13. Oktober 2019, von *https://docs.microsoft.com/de-de/azure/cognitive-services/translator*

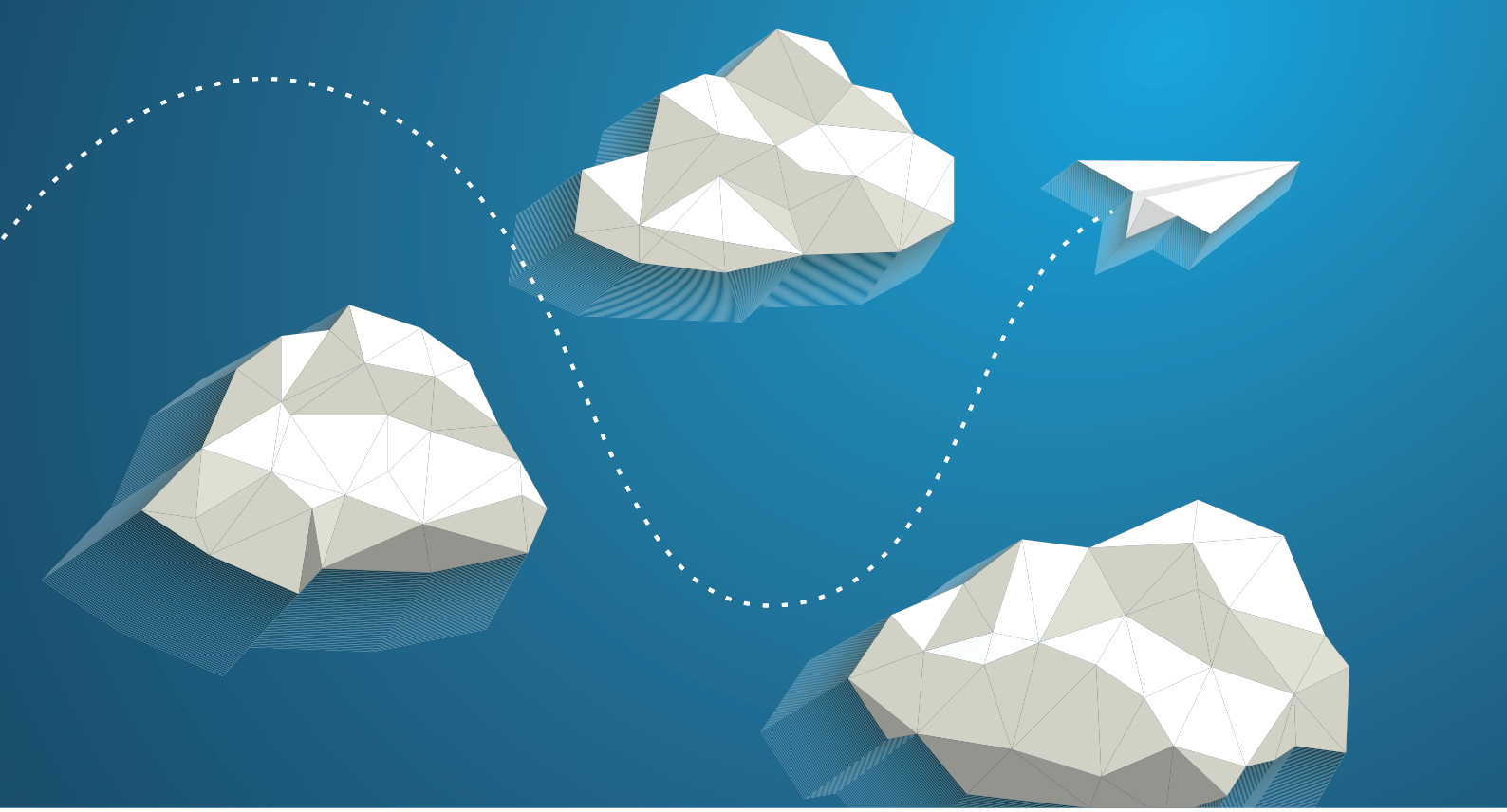
[3] Malte Ubl und Eiji Kitamura (2010, Oktober 20). Einführung zu WebSockets: Sockets im Web. Abgerufen 13. Oktober 2019, von *https://www.html5rocks.com/de/tutorials/websockets/basics/*


## Über den Autor


Fabian Neureiter ist Student der Medieninformatik an der Westfälischen Hochschule und seit Ende 2018 Werkstudent bei der MT AG im Bereich Anwendungsentwicklung mit Oracle Application Express (APEX). Er ist sehr an Machine Learning, JavaScript und funktionaler Programmierung interessiert. Zurzeit beschäftigt er sich mit JavaScript-Testing im APEX-Kontext.



Fabian Neureiter
Fabian.Neureiter@mt-ag.com



# OOW 2019 Newsflash: Exadata in der Cloud

Frank Schneede, Oracle Deutschland

Die Exadata Database Machine ist als On-Prem-Lösung eine seit vielen Jahren bekannte und etablierte Lösung für den hochperformanten Betrieb von Oracle-Datenbanken. Seit einiger Zeit steht die Exadata-Technologie auch für die Nutzung in der Oracle Public Cloud beziehungsweise als Exadata Cloud at Customer zur Verfügung. Auf der diesjährigen Oracle Open World wurde die Exadata Cloud at Customer Generation 2 vorgestellt, die einige sehr interessante Änderungen mit sich bringt und in diesem kurzen Newsflash vorgestellt werden soll.

Die Exadata Cloud at Customer ist eine Lösung, in der eine virtualisierte Exadata Database Machine zwar physikalisch im Rechenzentrum des Kunden steht, aber als Cloud-Lösung betrieben wird. Das für die Agilität und Betriebsprozesse notwendige Toolset wurde bislang auf einer separaten Cloud Control Plane betrieben, die der Kunde ebenfalls beschaffen und in einem separaten Rack betreiben musste. Die bisherige Lösung war komplex im Setup und relativ kostenintensiv, weswegen viele Kunden davon abgeschreckt worden sind. Die Exadata Cloud at Customer Gen 2 hat neben der Aktualisierung der Hardware insbesondere in diesem Bereich eine wesentliche Weiterentwicklung erfahren.

## Exadata Cloud at Customer Gen 2 (ExaCC) – X8-2 Hardware

Die **Exadata Cloud at Customer Gen 2** basiert auf der aktuellen Exadata-X8-2-Generation und steht wie bisher als Base, Quarter, Half oder Full Rack zur Verfügung.

Die Datenbankserver sind 2-Sockelsysteme, enthalten zwei aktuelle **Intel® Cascade Lake 26 Core CPUs** und verfügen über **720GB** nutzbaren Hauptspeicher pro Datenbankserver für User VM – das ist doppelt so viel wie die Standardausstattung in der On-Prem Exadata. In den Storage-Servern werden zwei **Intel® Cascade Lake 24 Core CPUs** verwendet, damit verfügen diese Server im Vergleich zur On-Prem Exadata sogar über **50% mehr verfügbare Leistung** für SQL Offloading. Die Kapazität der Festplatten und der Flash-Module entspricht der On-Prem-Maschine.

| Metric | Base* | Quarter | Half | Full |
|---|---|---|---|---|
| **Number of Database Servers** | 2 | 2 | 4 | 8 |
| **Max Number of OCPUs** | 48 | 100 | 200 | 400 |
| **Total Memory (GB)** | 720 GB | 1,440 GB | 2,880 GB | 5,760 GB |
| **Number of Storage Servers** | 3 | 3 | 6 | 12 |
| **Total Usable Disk Capacity** | 74.8 TB | 149.7 TB | 299.4 TB | 598.7 TB |
| **Max DB Size - local backup** | 29.9 TB | 59.9 TB | 119.8 TB | 239.5 TB |
| **Max DB Size - no local backup** | 59.9 TB | 119.8 TB | 239.5 TB | 479.0 TB |

*Tabelle 1: Übersicht Exadata Cloud at Customer Gen 2 Hardware Shapes*

In der ExaCC sind zudem je Datenbankserver 4 lokale Festplatten mit je 1,2TB Rohdatenkapazität verbaut, die für VMs, OS, Oracle-Home-Verzeichnisse und Logs verwendet werden. Zudem verfügt die Maschine über ein vorkonfiguriertes Cluster-File-System für große Files und Filesharing über DB-Server hinweg. Die Systeme sind also im Vergleich zur Vorgängergeneration um einiges leistungsfähiger geworden, wie *Tabelle 1* anschaulich zeigt:

## Cloud Control Plane in der Oracle Cloud Infrastructure (OCI)

Eine sehr umfassende Änderung in der neuen Architektur der ExaCC Gen2 stellt die Verlagerung des Control Plane in die Oracle Public Cloud dar. War bislang für den Betrieb der Control Plane ein eigenes Rack mit umfangreicher Hardware und entsprechenden Kosten für die notwendigen Services erforderlich, so wird die Rechenleistung für das Tooling nun in der Public Cloud bereitgestellt. Lediglich für die Kommunikation über einen gesicherten VPN-Tunnel werden zwei Server benötigt, die in die beiden obersten Einschübe des ExaCC-Racks integriert werden. Der Kunde wählt dann als Region für „seine" Control Plane üblicherweise die nächste OCI-Region, das wäre für deutsche Kunden Frankfurt.

Mit dieser Architektur wird erreicht, dass die gesamte Konfiguration der Control Plane erheblich einfacher und vor allem kostengünstiger ist! Es muss zur Konfiguration nur der Zugriff auf die Control Plane in der OCI definiert werden. Darüber hinaus wird so erreicht, dass das Tooling von **Exadata Cloud at Customer** und **Exadata Cloud Service** vereinheitlicht wird. *Abbildung 1* zeigt die Cloud-Anbindung und Informationsflüsse.

## Vereinfachte Anbindung an das Kundennetzwerk

Durch den Wegfall der komplexen Konfiguration der Cloud Control Plane beim Kunden hat sich die Einbindung der **Exadata Cloud at Customer Gen 2** ins Kundennetzwerk erheblich vereinfacht. Sie wird nun auf ähnliche Weise wie eine On-Prem Exadata behandelt. Dem Kunden stehen wahlweise 10- oder 25-Gb/s-Fiber- oder 10-Gb/s-Kupfer- Anbindungen zur Verfügung, er kann dazu seine eigenen Standard Switches verwenden, die optional im Exadata Rack platziert werden können. Der Kunde hat volle Kontrolle über die Netzwerkkonfiguration; die Konfiguration von Client- und Backup-Netzwerk wird durch flexible VLAN-Konfiguration ermöglicht.

Anschließend muss nur noch die Verbindung der lokalen **Control Plane Server** (CPS) zur **Public Cloud Control Plane** hergestellt werden. Diese OpenVPN- Verbindung der CPS zur Oracle Public Cloud ist essenziell und wird daher durch Oracle bestmöglich abgesichert.

Der Zugriff auf die CPS ist über Role Based Access Control beschränkt auf ausgewählte Named User. Verbindungen von Oracle Cloud Operations auf die CPS erfolgen von Bastion-Servern aus, die in jeder OCI-Region verfügbar sind und jeden Zugriff protokollieren. Die Bastion-Server werden über ein spezielles Team administriert, die wiederum keinen Zugriff auf die Control Plane Server haben. Über das **Oracle Corporate Securi-**

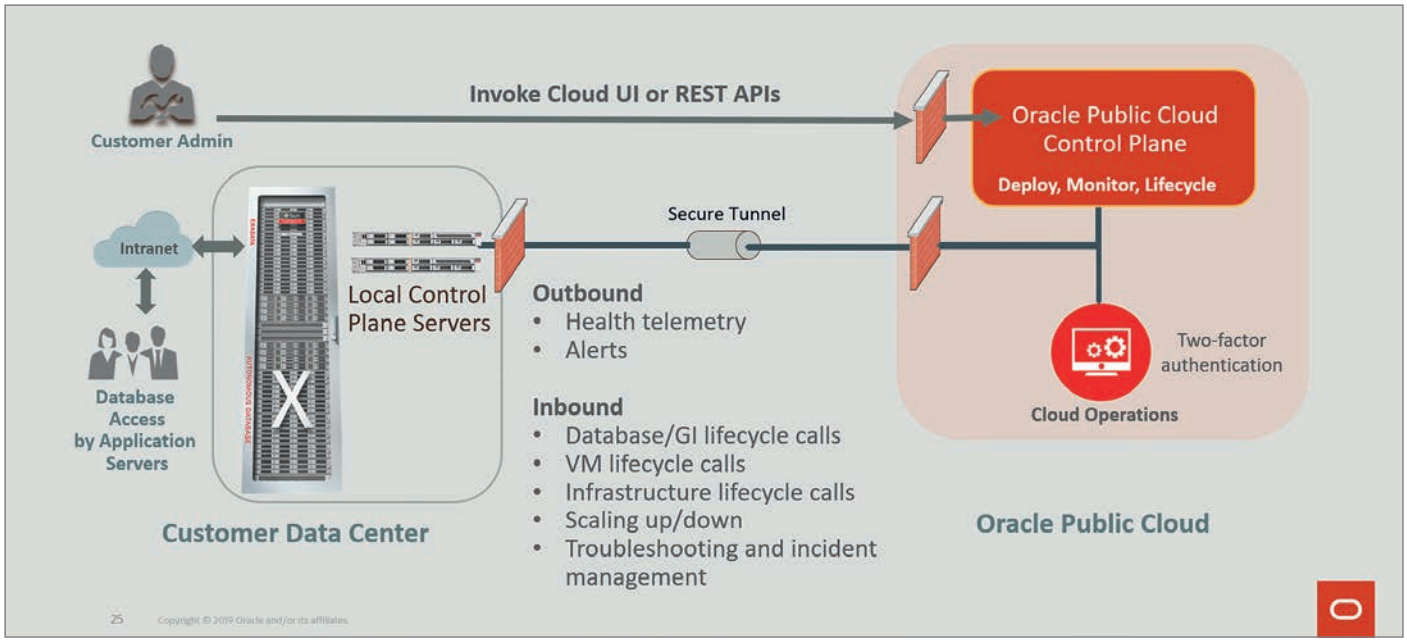



*Abbildung 1: Exadata Cloud at Customer Gen 2 Management Flow (Quelle: © Oracle)*

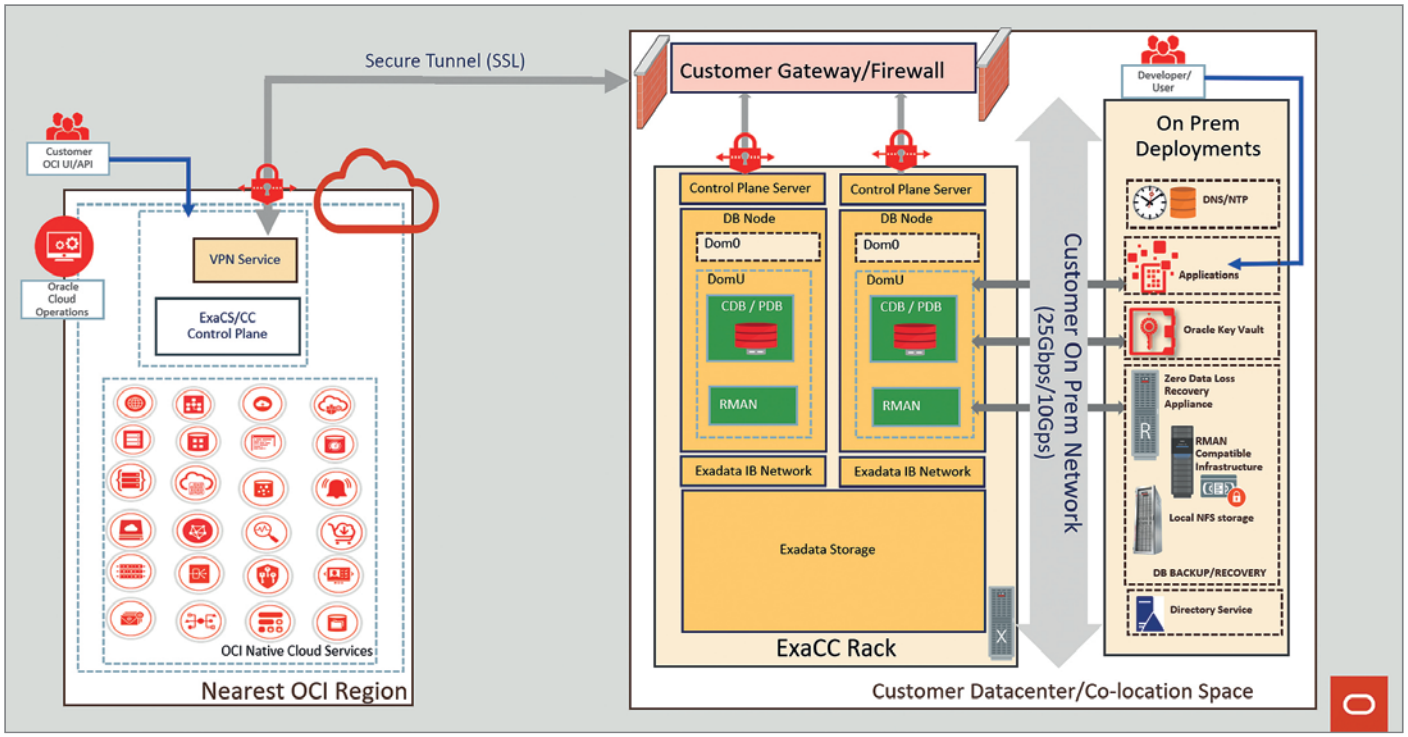



*Abbildung 2: Exadata Cloud at Customer Gen 2 Deployment Architecture (Quelle: © Oracle)*

**ty Solution Assurance Programm (CSSAP)** wird die Entwicklung der Sicherheitsfeatures der **Exadata Cloud at Customer** kontrolliert, bewertet und zertifiziert.

Zum Abschluss wird das ExaCC-System dem Cloud-Account des Kunden hinzugefügt. *Abbildung 2* illustriert das vollständige Deployment.

## Unterstützung der aktuellen Oracle Database 19c

Während in der ExaCC Gen 1 eine Installation der Oracle Database 19c nur händisch möglich war, wird das nun vollständig über Public Cloud Control Plane REST APIs und User Interface unterstützt. Damit können alle aktuell unterstützten Datenbankversionen von 11.2.0.4 bis 19c auf der ExaCC Gen 2 betrieben werden. Auch Backup & Recovery zu einem lokalen NFS Filer wird durch das Web UI bereitgestellt.

Andere Funktionen wie RAC Node Subsetting, DB/GI/DomU OS Patching, Shared Oracle Homes werden noch über das CLI der User VM bereitgestellt, eine Realisierung über Cloud REST APIs ist geplant. An der Umsetzung weiterer Funktionen für die ExaCC Gen 2 wie dem Multi-VM-Betrieb und der automatischen Bereitstellung einer Data-Guard-Umgebung wird zurzeit noch gearbeitet.

## Vorbereitung für den Einsatz von Autonomous Database at Customer

Das große Thema in der OCI ist aktuell Autonomous Database, hier stehen momentan zwei Services in der OCI bereit, **Autonomous Data Warehouse (ADW)** und **Autonomous Transaction Processing (ATP)**. Technologische Kernkomponenten dieser Services sind Exadata und die Oracle Cloud Infrastructure. Aus diesem Grunde können diese Services jedoch technisch nicht auf On-Prem-Systemen ohne Cloudanbindung umgesetzt werden! Die **Exadata Cloud at Customer** beinhaltet genau diese Kernkomponenten und ist daher entsprechend vorbereitet, um künftig **Autonomous Database at Customer** Services bereitzustellen.

## Fazit

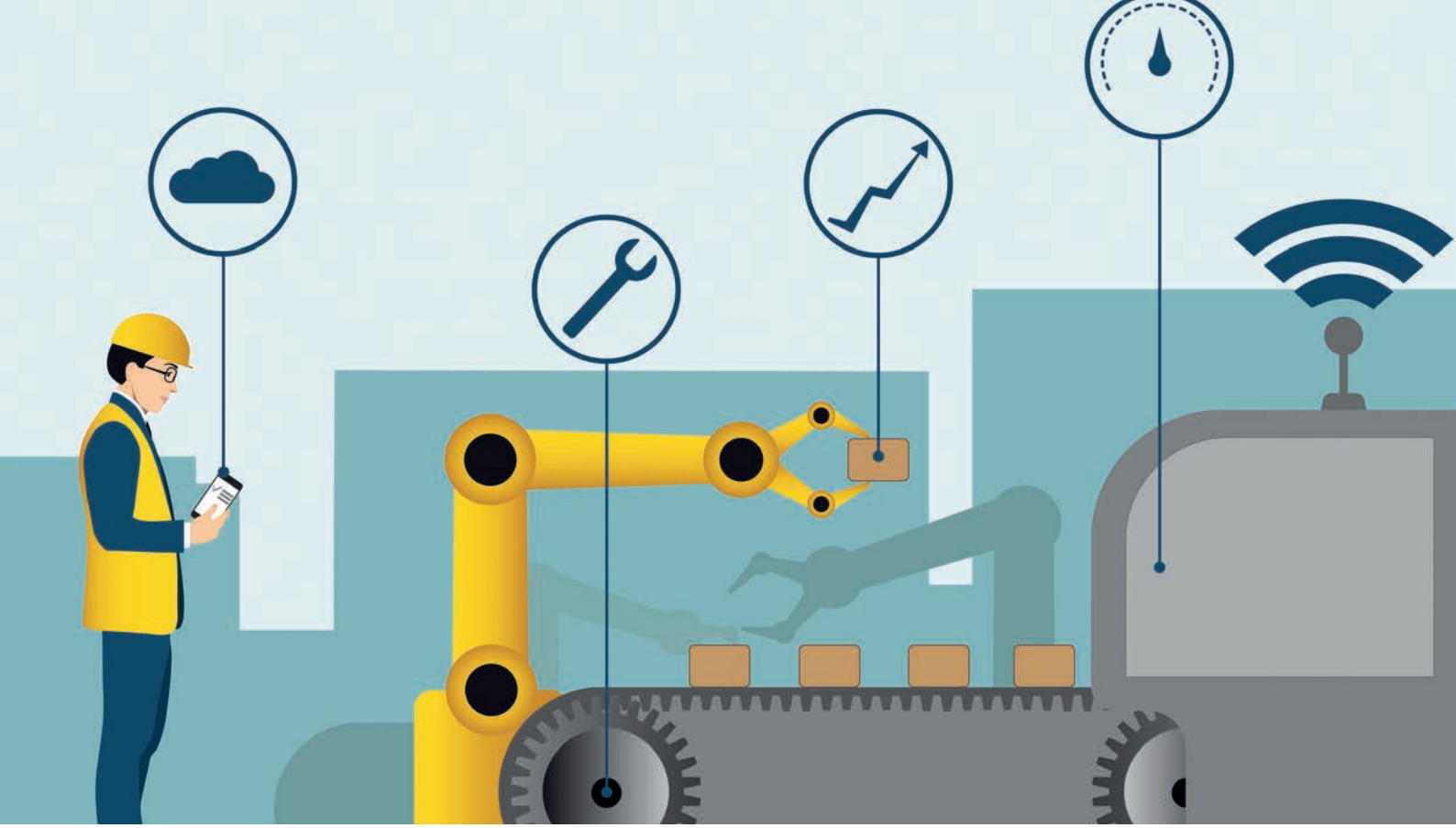
Die **Exadata Cloud at Customer Gen 2** eliminiert die wesentlichen Punkte, die Kunden bislang davon abgehalten haben, auf Exadata Cloud at Customer zu setzen: deutlich kostengünstiger in der Beschaffung, konsistenter im Betrieb und wesentlich einfacher sowie schneller in der Einrichtung, die der eines Exadata-On-Prem-Systems künftig kaum nachstehen dürfte. Damit ist der Weg von **Exadata Database Machine** On-Prem über **Exadata Cloud at Customer** zum **Exadata Cloud Service** einen großen Schritt einfacher und durchgängiger geworden.

Für Kunden, die derzeit noch eine ExaCC Gen 1 einsetzen, wird an einem Upgradepfad gearbeitet, der laut OOW Announcement im kommenden Jahr kostenfrei zur Verfügung stehen dürfte.

## Weiterführende Informationen

- *https://www.oracle.com/cloud/cloud-at-customer.html*
- *https://www.oracle.com/a/ocom/docs/engineered-systems/exadata/gen2-exacc-ds.pdf*
- *https://docs.cloud.oracle.com/iaas/Content/Database/Concepts/eccoverview.htm*

Frank Schneede
Frank.Schneede@oracle.com



# APEX und Oracle Autonomous Database

Ulrike Schwinn, Oracle Deutschland

In Oracle Autonomous Database sind nicht nur SQL Developer Web und die Oracle-Data-Rest-Schnittstellen integriert, sondern seit Ende Juni dieses Jahres nun auch Oracle Application Express. Und damit nicht genug: Larry Ellison kündigte im September während der Oracle Open World neue Always Free Services an, die eine zeitlich unbegrenzte Nutzung von Autonomous-Datenbanken mit bestimmten Ressourcen-Limitierungen erlaubt. Somit lässt sich APEX auf Dauer im Rahmen der Limitierungen in einer Cloud-Datenbank kostenfrei betreiben.

Oracle APEX gibt es jetzt auch auf (serverless) Autonomous Database. Was bedeutet dies für die Entwicklung und Bereitstellung von Oracle APEX? „Oracle APEX on Autonomous Database" ist eine installierte, vorkonfigurierte, vollständig verwaltete und gesicherte Umgebung und stellt bis auf wenige Ausnahmen die gleichen Funktionen wie auf On-Premises-Installationen zur Verfügung. Die Konfiguration, das Patchen, das Monitoren und das Upgrade aller Oracle-Application-Express-Komponenten werden dabei vollständig von Oracle verwaltet, sodass man sich vollständig auf die Entwicklung konzentrieren kann. Serverless Autonomous Database bietet je nach Workload-Art zwei unterschiedliche Instanztypen an: ADW (Autonomous Data Warehouse) für Analysen und Data-Warehouse-Anwendungen und ATP (Autonomous Transaction Processing) für Transaction Processing beziehungsweise Mixed Workloads. Hat man eine (serverless) Autonomous Database – egal welchen Workload-Typs – provisioniert, kann man sofort mit der Entwicklung von APEX-Anwendungen beginnen. Das Provisionieren und das Setup von APEX dauert dabei nur einige Minuten.

## Oracle Autonomous Database

Autonomous Database besteht aus dem neuesten Oracle-Datenbank-Release sowie einer Cloud-Infrastruktur-Automatisierung und verwendet Oracle Exadata Database Machine. Autonomous Database steht in allen Rechenzentren und damit natürlich auch in Frankfurt zur Verfügung. Autonomous Service bedeutet übrigens auch, dass man keinerlei Zugriff auf die darunterliegende Infrastruktur beziehungsweise Plattform haben muss oder haben kann, da sich Oracle um das Management rund um den Service kümmert. Cloning, 1 TB Minimalallokation von Storage, Auto Scaling, automatisches Backup und Patchen, komprimierte und verschlüsselte Daten sowie Wallet-basierte Connections sind nur einige der Features, die in Autonomous Database zur Verfügung stehen.

## Always Free Services: kostenlose Autonomous Database auf Dauer

Autonomous Database ist auch Bestandteil der neuen Always Free Services, den Larry Ellison auf der OOW2019 angekündigt hat. Always Free (zu deutsch: immer kostenlos) besteht aus dem folgenden Leistungsangebot, das **zeitlich unbegrenzt** genutzt werden kann:

- Zwei Oracle Autonomous Databases mit Tools wie Oracle Application Express (APEX) und Oracle SQL Developer.
- Zwei Oracle Cloud Infrastructure Compute VMs inklusive Block-, Objekt- und Archivspeicher, Load Balancer und Datenverkehr, Überwachung und Benachrichtigungen. (Die Ressourcen-Restriktionen findet sich auf der Free-Tier-Webseite *(siehe: https://www.oracle.com/cloud/free/)*

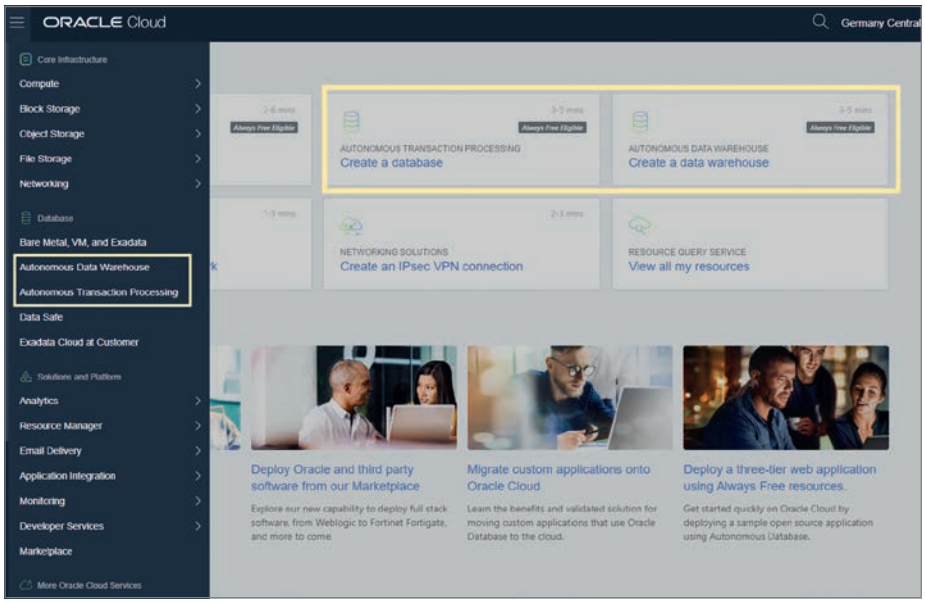



*Abbildung 1: Autonomous Database Services im Cloud-Menü und auf der Homepage (Quelle: Ulrike Schwinn)*

Das Angebot erlaubt die kostenfreie Erstellung und Nutzung von zwei Autonomous Databases auf unbegrenzte Zeit. Die Konfiguration einer Always-Free-Oracle-Autonomous-Database-Instanz ist dabei im Unterschied zur bezahlten Variante beschränkt auf 1 OCPU, 20 GB Speicher und 8 GB Arbeitsspeicher. Die meisten Funktionen der autonomen Datenbank sind darin verfügbar, außer Scale Up/Down, Auto Scaling, manuelle Backups und Restore. Der Produktsupport ist in diesem Angebot natürlich nicht eingeschlossen. Ein Upgrade zu einer bezahlten Version ist natürlich immer möglich.

Um sich anzumelden, kann man einfach die Free-Tier-Webseite unter *https://www.oracle.com/cloud/free/* verwenden und

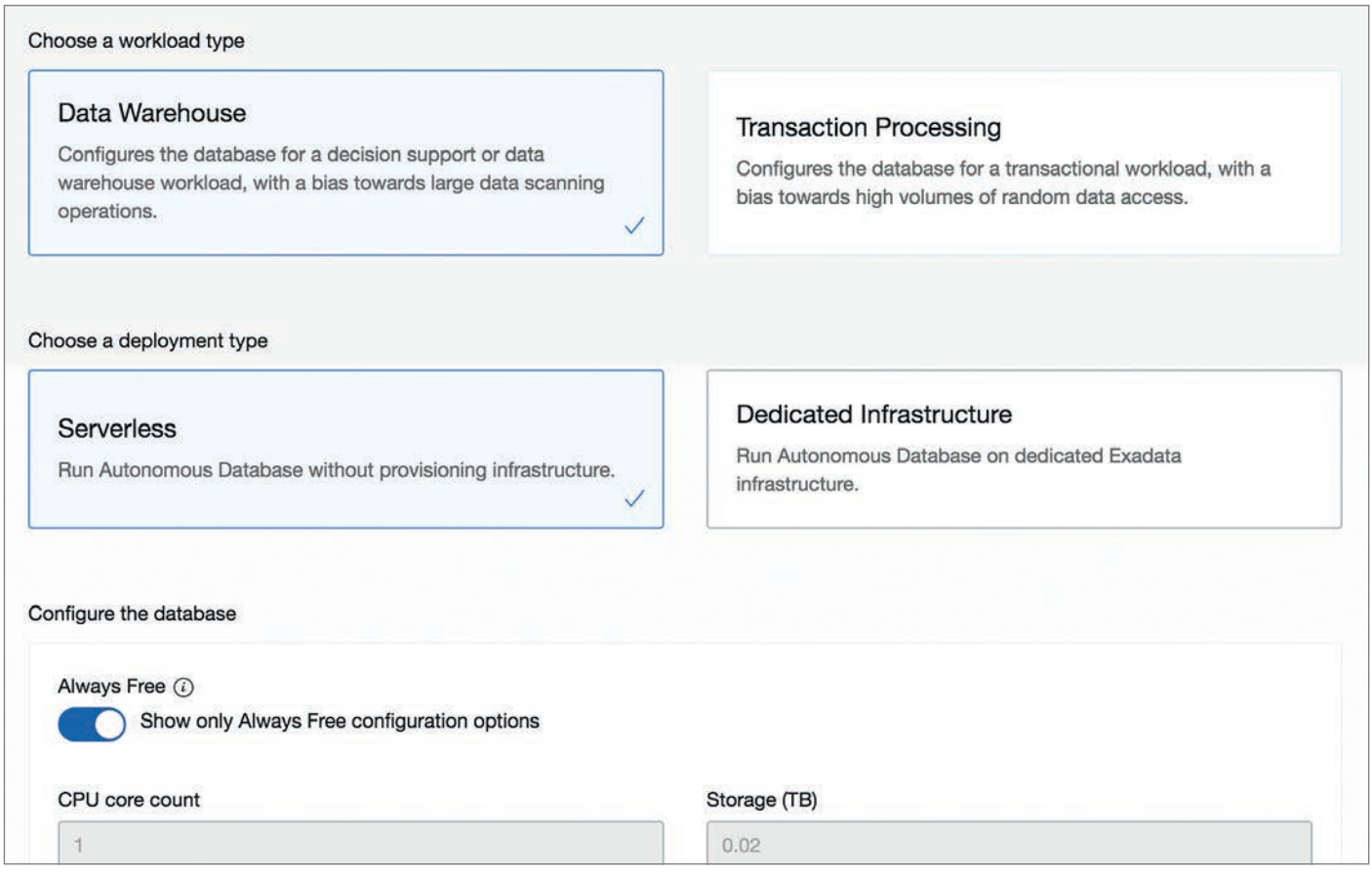



*Abbildung 2: Aktivierung des Always-Free-Schalters (Quelle: Ulrike Schwinn)*

auf den Button zur Registrierung „Start for free“ klicken. In der Regel müssen neue Benutzer bei der Anmeldung zu Identifikationszwecken eine Kreditkarte angeben. Anwender, die Oracle zum Beispiel durch Oracle-Academy- oder Oracle-Openworld-Registrierung bekannt sind, können sich auch ohne Kreditkarte anmelden. Während der Registrierung im zweiten Schritt wird man aufgefordert, eine „Home Region“ auszuwählen. Dies ist wichtig, da Always Free Services nur in der Heimatregion laufen und dies nach der Erstellung des Kontos nicht mehr geändert werden kann. Nachdem die Registrierung durchgeführt ist, erhält man eine E-Mail mit den Zugangsinformationen.

Wie funktioniert nun das Provisionieren einer Autonomous Database als Always Free Service?

Zum Provisionieren einer Autonomous Database, als ATP- oder ADW-Instanz, kann das Cloud-Menü oder die Startseite der Oracle Cloud Console mit Quick Actions verwendet werden. Dann ist nur noch die Eingabe des Datenbanknamens und das Passwort des Users Admin im „Create Autonomous Database“-Template erforderlich. Dabei sollte man nicht vergessen, den Schalter (Toggle) „Always Free “ zu aktivieren.

Nach ein paar Minuten steht die Always Free Autonomous Database zur Verfügung und ist mit der Überschrift „Always Free“ gelistet.

Wenn man versucht, Funktionen der autonomen Datenbank zu verwenden, die in Always Free Database nicht verfügbar sind, erhält man eine Meldung mit der Option, ein Upgrade auf die bezahlte Variante durchzuführen.

Nun kann es auch schon losgehen. Alles Wichtige für den Einstieg mit APEX findet sich im Handbuch „Using Oracle Autonomous Transaction Processing“ in Kapitel 7 „Creating Applications with Oracle Application Express in Autonomous Database“. Die wenigen Schritte sind auch im folgenden Abschnitt beschrieben.

Noch ein Hinweis zur Nutzung: Nach der Bereitstellung kann Always Free Autonomous Database so lange wie gewünscht kostenlos weiterverwendet werden und auch jederzeit beendet werden. Wenn der Always Free Autonomous Database Service allerdings für einen Zeitraum von sieben aufeinanderfolgenden Tagen keine Aktivität aufweist, wird die Datenbank automatisch angehalten. In diesem Fall kann man die Datenbank neu starten und weiterverwenden. Wenn Always Free Autonomous Database hingegen in drei aufeinanderfolgenden Monaten in einem gestoppten Zustand bleibt, steht die Ressource nicht mehr zur Verfügung.

## Oracle Application Express

Da die APEX-Instanz einer Autonomous-Datenbank über keine vordefinierten Workspaces für Oracle Application Express verfügt, muss zuerst ein neuer Workspace erstellt werden. Dazu navigiert man im Da-

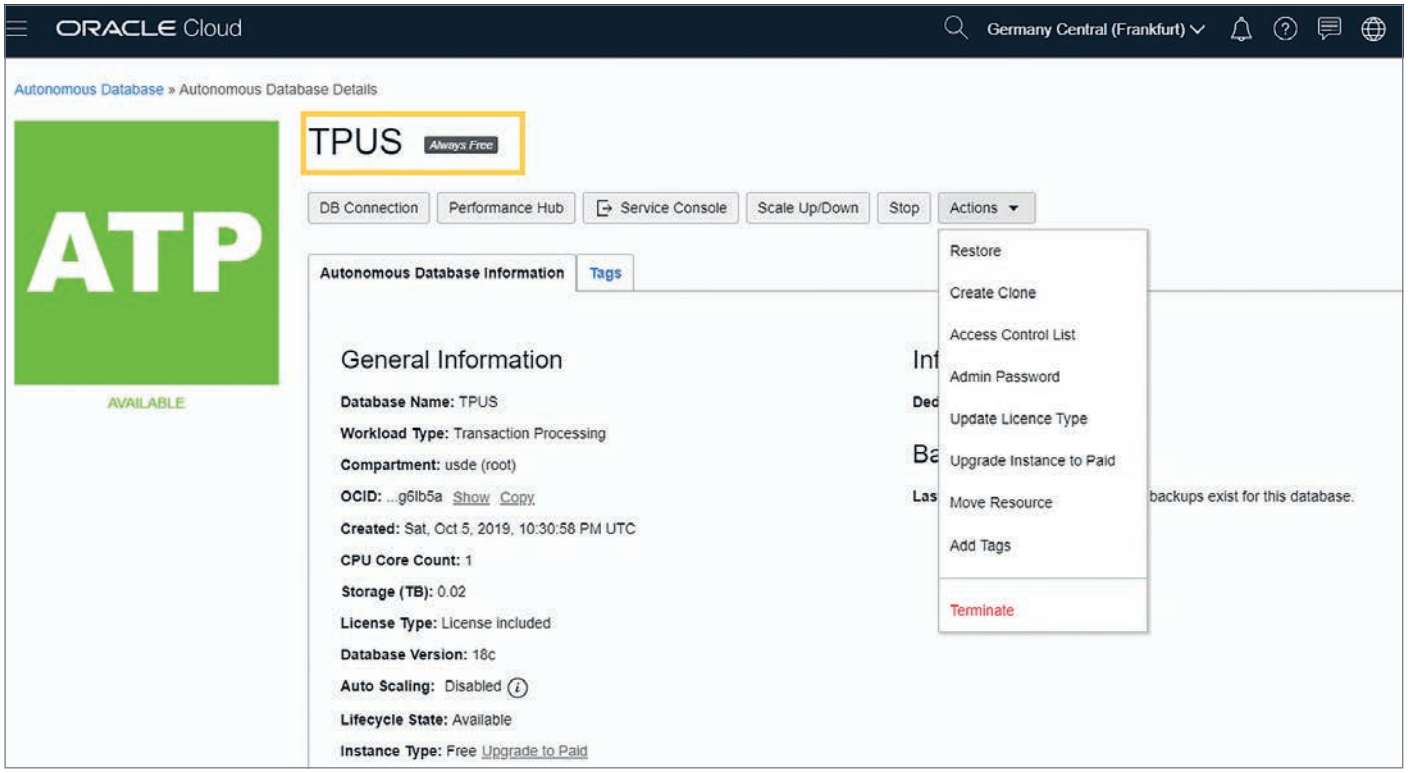



*Abbildung 3: Always-Free-Instanz (Quelle: Ulrike Schwinn)*

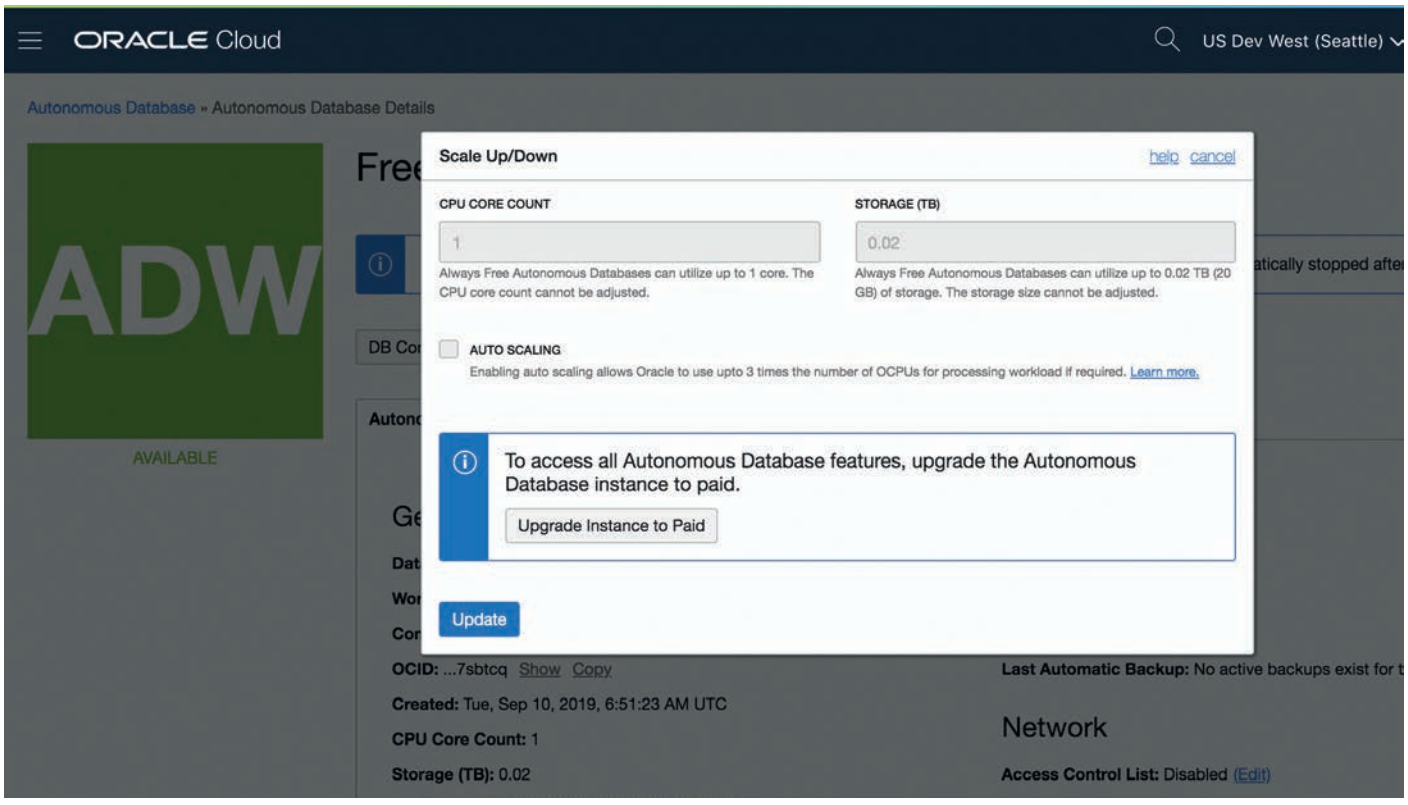



*Abbildung 4: Hinweis auf das Upgrade (Quelle: Ulrike Schwinn)*

tenbank-Cloud-Menü über die Service-Konsole zum Menüpunkt Development.

Dann klickt man auf den Button „Oracle APEX" und gelangt von dort aus in den Bereich „Administration Services". Nach erfolgreicher Eingabe des Passworts des Users ADMIN befindet man sich im „Instance Administration"-Bereich. Hier lassen sich Workspaces und Applikationen verwalten und überwachen. Beim ersten Anmelden wird zuerst ein Fenster angezeigt, um einen weiteren Workspace anzulegen.

Nun ist alles bereit zum Arbeiten mit APEX. Man navigiert erneut über die Service Console auf die Application Express Login Site und loggt sich jetzt mit dem neuen Workspace und Username/Passwort ein. Am besten „bookmarkt" man sich

gleich die entsprechende URL. In meinem Fall sieht die URL ungefähr so aus:

https://xxxxxx-tpus.adb.eu-frankfurt-1.oraclecloudapps.com/ords/apex

Nun kann man mit der Entwicklung von APEX-Applikationen beginnen, bestehende Applikationen oder eine der zahlreichen Sample Apps importieren. Dabei lassen sich mit dem „Create Application"-Wizard APEX-Applikationen entweder auf schon bestehenden Daten oder nach dem Neuladen von Daten erstellen. Zum Datenladen in APEX eignet sich beispielsweise auch der „Data Workshop", der über „SQL Workshop" und „Utilities" zu finden ist. Von hier aus lassen sich dann ganz einfach Daten in unterschiedlichen Formaten, wie zum Beispiel CSV, XLSX, XML und JSON, laden oder entladen.

## Das Zusammenspiel

Wie sieht nun das Zusammenspiel mit (serverless) Autonomous Database und APEX aus? Oracle APEX verwendet eine 3-Tier-Architektur, bei der Anfragen vom Browser über einen Webserver an die Datenbank gesendet werden. Wie man an der URL erkennen kann, wird Oracle REST Data Services (ORDS) verwendet, um die Anfragen an die Datenbank zur Bearbeitung zu senden. Innerhalb der Datenbank wird der Request dann von Oracle APEX verarbeitet. Nach Abschluss der Verarbeitung wird das Ergebnis über ORDS an den Browser zurückgegeben.

Da wir es mit einer Managed Plattform zu tun haben, das bedeutet Konfiguration, Patching, Monitoring und Upgrades von APEX-Komponenten wird von Oracle durchgeführt, erübrigt sich eigentlich die Frage nach der APEX-Version oder auch nach dem Monitoring von SQL-Perfor-

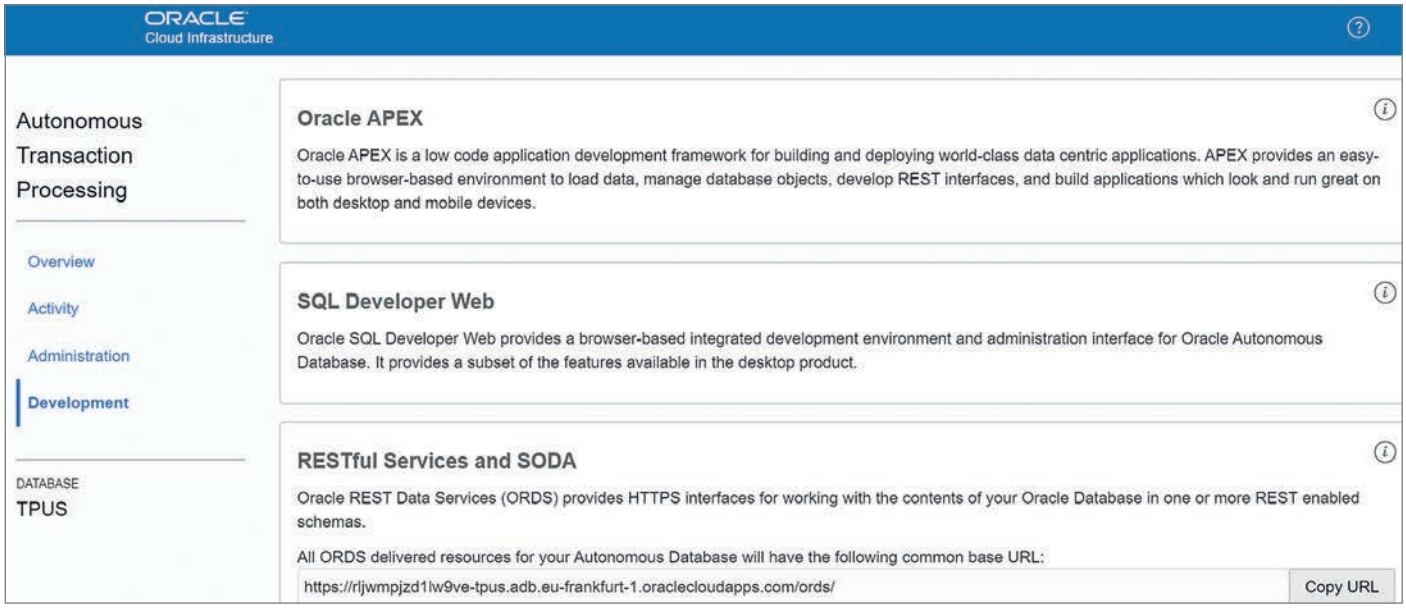



*Abbildung 5: APEX in der Service Console (Quelle: Ulrike Schwinn)*

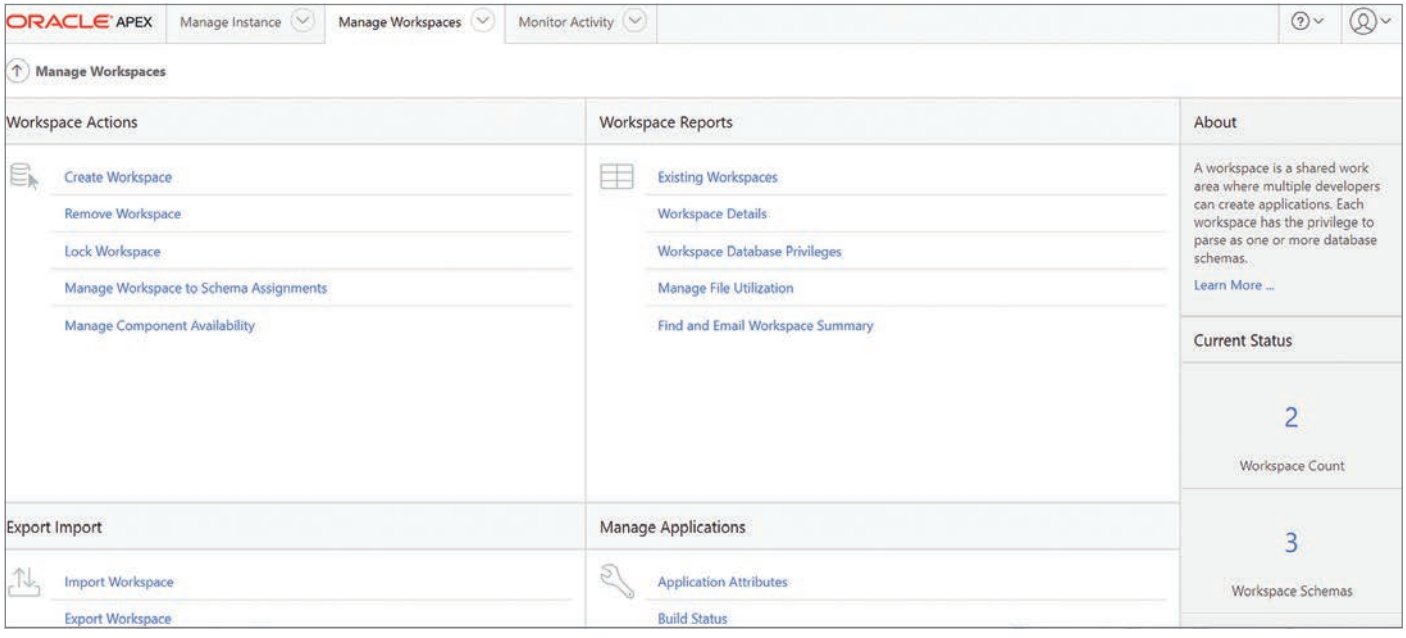



*Abbildung 6: Überblick über Administration Services (Quelle: Ulrike Schwinn)*

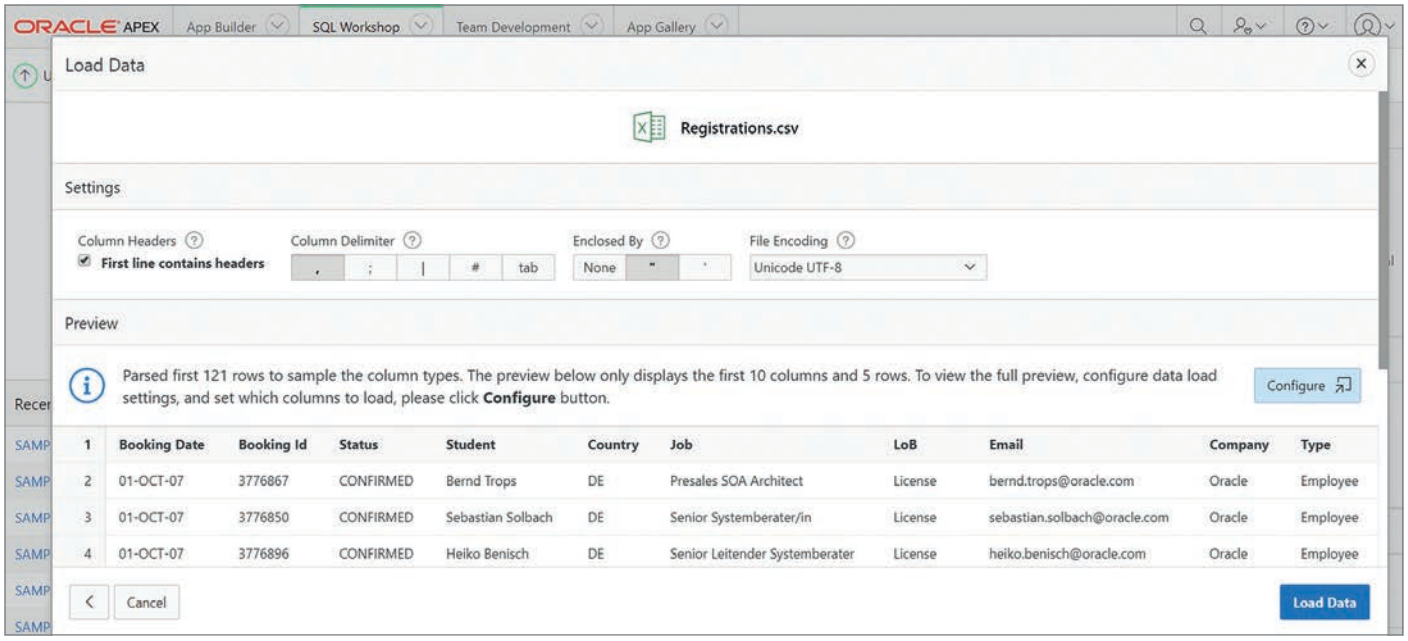



*Abbildung 7: Laden einer CSV-Datei über „Data Workshop" (apexauto7.png) (Quelle: Ulrike Schwinn)*

mance. Trotzdem lassen sich auch die Aktivitäten wie SQL-Statements über den Activity Monitor in der Autonomous Database Service Console monitoren. Möchte man wissen, welche APEX-Version installiert ist, kann man sich wie in allen Tools mit dem Help Button behelfen. APEX steht, wie zu erwarten war, in der aktuellen Version 19.x zur Verfügung.

Wie schon beschrieben, lassen sich auch bestehende Applikationen importieren. Beachten sollte man allerdings, dass ein paar Einschränkungen in der Autonomous Database und bei der Verwendung von APEX existieren. Einschränkungen im APEX-Umfeld gibt es beispielsweise bezüglich Administration Service, Printing Service oder im Authentication-Bereich. Nachzulesen ist dies etwa im Handbuch „Using Oracle Autonomous Transaction Processing" in Kapitel 7 „Creating Applications with Oracle Application Express in Autonomous Database".

Autonomous Database konfiguriert und optimiert die Datenbank automatisch. Für die Konfiguration der Datenbank müssen und können somit keine Verwaltungsoperationen und Einstellungen durchgeführt werden. SQL-Befehle, die für die Datenbankverwaltung verwendet werden, wie zum Beispiel CREATE TABLESPACE, sind daher nicht verfügbar. Ebenso sind andere administrative Schnittstellen wie der Zugriff auf das Betriebssystem und Dienstprogramme wie RMAN nicht verfügbar. Einen Überblick über die Einschränkungen in Autonomous Database finden sich in den Handbüchern beispielsweise in „Using Oracle Autonomous Transaction Processing" im Appendix „Autonomous Transaction Processing for Experienced Oracle Database User". Autonomous Database unterstützt wie die On-Premises-Installationen XML, JSON, Geodaten und linguistische Abfragen. Falls es in diesen Bereichen Limitierungen gibt, sind diese im Abschnitt zu „Restrictions for Database Features" zu finden.

## Fazit

APEX mit Always Free Autonomous Database Cloud Service ist eine perfekte Kombination, um Oracle-Datenbank-Applikationen mit APEX in einer Cloud- Datenbank zu verwenden. Nach 5 bis 10 Minuten besitzt man eine lauffähige Autonomous Database, hat sich einen Workspace zugelegt und kann mit der Entwicklung starten. Auch für Anfänger ist die Hürde damit gering, die ersten Schritte mit einer Oracle-Datenbank durchzuführen und die ersten APEX-Applikationen erfolgreich zu programmieren. Am besten gleich ausprobieren!

## Links und weitere Informationen

[1] Oracle Always free Tier Homepage: *https://www.oracle.com/cloud/free/*
[2] Using Oracle Autonomous Transaction Processing: Kapitel 7, Appendix
[3] Using Oracle Autonomous Data Warehouse: Kapitel 7, Appendix
[4] Oracle Cloud Infrastructure Documentation: Overview of the always free autonomous Database
[5] *https://blogs.oracle.com/coretec*

## Über die Autorin

Ulrike Schwinn studierte Mathematik und ist bei der Firma Oracle in München im Bereich Systemberatung tätig. In ihrer Funktion berät und schult sie Kunden in Fragen zu neuesten Oracle-Datenbank und zu Cloud-Technologien, wie zum Beispiel Oracle Autonomous Database. Sie wirkte an mehreren Oracle-Buchprojekten mit, veröffentlicht Blog-Postings im Internet und ist als Referentin auf IT-Kongressen vertreten.

## Profile und Blogs

*https://www.linkedin.com/in/ulrikeschwinn/*
*https://twitter.com/uschwinn*
*https://www.xing.com/profile/Ulrike_Schwinn*
*https://blogs.oracle.com/coretec*

Ulrike Schwinn
Ulrike.Schwinn@oracle.com

# Progressive Web Apps mit APEX

Till Albert, MT

Progressive Web Apps (PWAs) sind voll im Trend und sorgen dafür, dass der Übergang zwischen dem Web und klassischen Apps immer mehr verschwindet. Im folgenden Artikel werden die Stärken und Schwächen einer PWA zusammen mit der Nutzung im Oracle-Umfeld, insbesondere mit APEX, näher unter die Lupe genommen.

## Was zeichnet eine Progressive Web APP aus?

Progressive Web Apps sind in erster Linie erst einmal eins: ganz normale Webseiten. Denn Progressive Web Apps arbeiten nach dem Prinzip des „Progressive Enhancement", nach dem eine Webseite auf jedem Endgerät jederzeit funktionsfähig ausgeführt werden kann, auch wenn es nur eine sehr grundlegende Webseite mit eingeschränkten JavaScript-/CSS-Features ist. Jederzeit bedeutet in diesen Fall wirklich jederzeit, selbst wenn das Gerät über keine Verbindung zum Internet verfügt. Das setzt natürlich voraus, dass die Webseite bereits einmal vorher abgerufen wurde. Dazu später mehr.

Jedes Endgerät weist natürlich seine ganz eigenen Eigenschaften auf, das können Smartphones und andere smarte Geräte sein oder aber auch der PC/Mac. Eine PWA soll auf all diesen Geräten nicht bloß funktionieren, sondern wie eine native App integriert werden und sich auch so verhalten. Wie gut das funktioniert, hängt primär von den Herstellern ab, die die für die PWAs notwendigen APIs implementieren müssen. Google ist bei diesem Thema Vorreiter und die neusten PWA-Features sind in Chrome und der Android-Plattform am schnellsten integriert. Auch Mozilla und Microsoft sind ganz vorne mit dabei. Nur Apple zögert. So wurden zwingend notwendige APIs erst im letzten Jahr von Apple in den Safari-Browser integriert. Damit laufen PWAs zwar grundlegend im Ökosystem von Apple, jedoch werden die neusten Features nur nach und nach implementiert.

Die Zurückhaltung von Apple dürfte einen Grund haben. So ist Apple bekannt für seine restriktive Auswahl von Apps für den hauseigenen App-Store. Einer der zentralen Punkte von Progressive Web Apps ist jedoch, dass deren Distribution und Installation ohne einen App-Store möglich ist. Mit steigender Popularität und Verbreitung von PWAs würde Apple also die Möglichkeiten zur Einschränkung von Apps im eigenen Ökosystem nach und nach verlieren.

Nachfolgend werden weitere Eigenschaften genannt, die eine PWA ausmachen.

- Responsive: Die Oberfläche der Anwendung passt sich dem Endgerät an
- HTTPS: Zwingend notwendig für die Nutzung von Service Worker

```
{ "name": "APEX Demo PWA",
  "short_name": "APEX PWA",
  "start_url": "https://myhost/apex/f?p=100:1",
  "description": "Beispiel einer APEX PWA",
  "theme_color": "#0071d4",
  "background_color": "#0071d4",
  "display": "standalone",
  "orientation": "portrait-primary",
  "Scope": "/",
 "icons": [
    {
      "src": "images/icons/icon-72×72.png",
      "sizes": "72×72",
      "type": "image/png"
    },
    ...
    {
      "src": "images/icons/icon-512×512.png",
      "sizes": "512×512",
      "type": "image/png"
    }
  ]
}
```

*Abbildung 1: Beispiel eines Web-App-Manifests (Quelle: Till Albert)*

- Immer die neuste Version: Da es sich um Webseiten handelt, sind keine Updates durch einen App Store notwendig
- Push-Benachrichtigungen: Die App kann den Nutzer zur Interaktion animieren

## Wo werden PWAs bereits eingesetzt und warum?

Interessant sind PWAs vor allem für den E-Commerce-Bereich, denn die Conversion-Rate, die Anzahl der Website-Besucher, die mit der Seite interagieren oder zum Kunden werden, lässt sich mithilfe von PWAs erhöhen. Das liegt vor allem an der einfachen Handhabung von PWAs. Der Nutzer muss nicht erst zwingend eine App installieren, sondern kann diese mit wenigen Klicks aus dem Browser direkt installieren.

Neben der einfachen Handhabung zur Installation tragen noch weitere Features zur Steigerung der Benutzerfreundlichkeit bei. So kann sich der Einsatz einer PWA positiv auf die Ladezeit der Webseite auswirken. Das wird zum einen durch die Service-Worker-Technologie ermöglicht, die das Caching von Dateien und damit auch eine Offline-Fähigkeit ermöglicht. Zum anderen durch den typischen Aufbau einer PWA, bei dem versucht wird, das Grundgerüst der Anwendung immer zu cachen, damit dieses beim Laden der Webseite sofort verfügbar ist. Mehr dazu später.

Die Webseite pwa.bar bietet eine interessante Übersicht über PWAs großer Unternehmen. Laut dieser konnte zum Beispiel AliExpress seine Conversion Rate mithilfe einer PWA mehr als verdoppeln.

Ebenfalls interessant, um mit dem Nutzer in Verbindung zu bleiben, selbst wenn dieser die PWA nicht installiert hat, sind Push-Benachrichtigungen. Mit diesen können Nutzer, sobald sie diese abonniert haben, Benachrichtigungen auch dann erhalten, wenn sie gerade nicht auf der Website selbst surfen.

## Aufbau einer Progressive Web App

Wie bereits anfangs erwähnt, handelt es sich im Prinzip um eine ganz normale Webseite. Das eigentliche Herzstück einer PWA ist aber das Service Worker API, das einige wichtige Funktionen bereitstellt, die eine Progressive Web App erst zu dem machen, was sie ist. Dadurch werden wichtige Funktionen wie Offline-Fähigkeit, Caching und Push-Benachrichtigungen erst ermöglicht.

Doch was genau muss man sich unter einem Service Worker vorstellen? Am einfachsten lässt es sich wohl mit einer Art Proxy vergleichen, der zwischen dem Webbrowser und dem Server interagiert. Alle Anfragen, die vom Client ausgehen, werden durch den Service Worker verarbeitet. Das hat den großen Vorteil, dass der Service Worker so sicherstellen kann, dass der Client immer eine Antwort erhält. Falls eine Verbindung zum Internet besteht, wird die Anfrage normal weitergeleitet; falls keine Verbindung besteht, so wird in einer kleinen Datenbank im Browser nachgeschaut, ob für den Request bereits ein Eintrag vorhanden ist. Ist das der Fall, wird dieser dann zurückgegeben und der Client kann die Antwort normal verarbeiten. Je nach Konfiguration wird bei jeder Anfrage und vorhandener Internetverbindung die Ressource in der Datenbank aktualisiert. Ist eine Ressource noch nicht in der Datenbank vorhanden und wird dennoch ohne Internetverbindung abgefragt, so kann eine Seite mit einer nutzerfreundlichen Fehlermeldung angezeigt werden. Zwingend notwendig für den Einsatz von Service Worker ist die Verwendung von HTTPS. Aus Gründen der Sicherheit funktionieren Service Worker ohne HTTPS nicht, da sie mächtige Funktionen übernehmen und Antworten des Servers durch eigene Inhalte ersetzen können.

Ein weiterer zentraler Bestandteil einer Progressive Web App ist das Web-App-Manifest. Dabei handelt es sich um ein Dokument im JSON-Format, das verschiedene Meta-Daten enthält und das Verhalten sowie verschiedene Eigenschaften einer PWA beschreibt. Das sind zum Beispiel der Name der App und eine Beschreibung, Icons in verschiedenen Auflösungen, die Standard-Ausrichtung der App oder Theme-Farben. Der Hauptzweck des Manifests ist die Installierbarkeit der PWA auf den verschiedenen Endgeräten. Erst durch das Manifest erkennt der Browser, dass es sich um eine Web App handelt.

## Implementierung in APEX

Wie anfangs erwähnt, zeichnet sich eine Progressive Web App durch einige Eigenschaften aus. Dabei kommt uns zugute, dass APEX einige dieser Eigenschaften sowieso im Standard schon unterstützt. So sind alle APEX- Anwendungen, die mit dem Universal Theme entwickelt wurden, von Hause aus bereits responsive und funktionieren in allen gängigen Browsern. Auch ein App-ähnliches Interface kann damit schnell implementiert werden. HTTP ist mit APEX ebenfalls kein Problem und sollte sowieso Standard sein.

Damit haben wir mit APEX also schon ein paar Punkte abgedeckt, ohne einen größeren Implementierungs-Aufwand. Einen besonderen Mehrwert zur klassischen APEX-Anwendung haben wir allerdings noch nicht. Das ändert sich jedoch, sobald das Web-App-Manifest in die Anwendung integriert wird *(siehe Abbildung 1)*. Eine einfache und schnelle Möglichkeit, um das Manifest generieren zu können, bietet die Webseite pwabuilder.com, die auch direkt dabei hilft, passende Icons für die verschiedenen Formate der Geräte zu erstellen. Diese Icons werden dann später, wenn die App installiert wird, als App-Icon verwendet.

Besonders wichtig ist dabei das Attribut „start_url“, das den Pfad zum Einstiegspunkt der App enthält, wenn die App gestartet wird.

Das generierte Manifest muss nun noch in die APEX-Anwendung integriert werden. Dazu genügt es jedoch nicht, dieses in die APEX-Anwendung hochzuladen, wie sonst bei JavaScript-Dateien üblich. Denn das Manifest muss von der Root-Ebene der Anwendung aus erreichbar sein. Wenn also eine APEX-Anwendung die URL https://host.de/ords/f?p=102:30 hat, dann muss das Manifest unter *https://host.de/manifest.json* erreichbar sein. Dies kann jedoch nur erreicht werden, wenn die Datei im entsprechenden Verzeichnis des Web-Servers abgelegt wird. Ein Zugriff auf den Web-Server ist also zwingend notwendig, womit viele Cloud-Dienste nicht dafür geeignet sind, da dieser Zugang oftmals fehlt.

Ist die Datei schließlich auf dem Web-Server abgelegt, muss sie noch in der APEX-Anwendung referenziert werden. Damit das Manifest korrekt vom Browser erkannt werden kann, muss es im Head-Bereich vor dem Body des HTML-Dokuments geladen werden. Das wird erreicht, indem in den User Interfaces (Shared Components) der APEX-Anwendung die Datei entsprechend verlinkt wird *(siehe Abbildung 2)*.

Sobald dies erledigt ist, kann die APEX-Anwendung nun ganz einfach mit einem Klick, ohne Nutzung eines App Stores, als App auf den meisten Endgeräten installiert werden. Das funktioniert, je nach Browser, leicht unterschiedlich, im Chrome- Browser findet man den Eintrag „Progressive Web App installieren…“ im Menü oben rechts. Das gilt übrigens nicht nur für mobile Endgeräte wie Smartphones. Auch auf einem PC oder Mac lassen sich die PWAs über den Browser als Applikation installieren. Sie können, wie eine normal installierte Anwendung, vom Desktop oder Startmenü aus aufgerufen werden und arbeiten in einem eigenständigen Fenster. Damit lässt sich mit relativ wenig Aufwand eine oft gewünschte Anforderung der Anwender umsetzen.

## Offline-Fähigkeit mit APEX?

Eines der spannendsten Features einer Progressive Web App ist wohl die Möglichkeit, diese auch ohne eine Verbindung zum Internet nutzen zu können. Wie bereits vorher erwähnt, basiert die Offline-Fähigkeit auf einem Service Worker in Verbindung mit einer kleinen Datenbank im Browser. Eine der Grundlagen, damit dieses Konzept erfolgreich funktioniert, ist eine strikte Trennung von Inhalten, die immer wiederverwendet werden, also das Grundgerüst der Anwendung, und den eigentlichen Daten, die bei jeder Anfrage abweichen können. Dieses Prinzip bezeichnet Google als Application Shell. Dieses Grundgerüst wird vom Service Worker immer im Cache beziehungsweise in der Datenbank im Browser gespeichert und kann beim Laden der Seite sofort ohne große Verzögerung angezeigt werden, eben auch offline. Dynamische Daten werden dann mit weiteren Anfragen entweder aus dem Internet oder

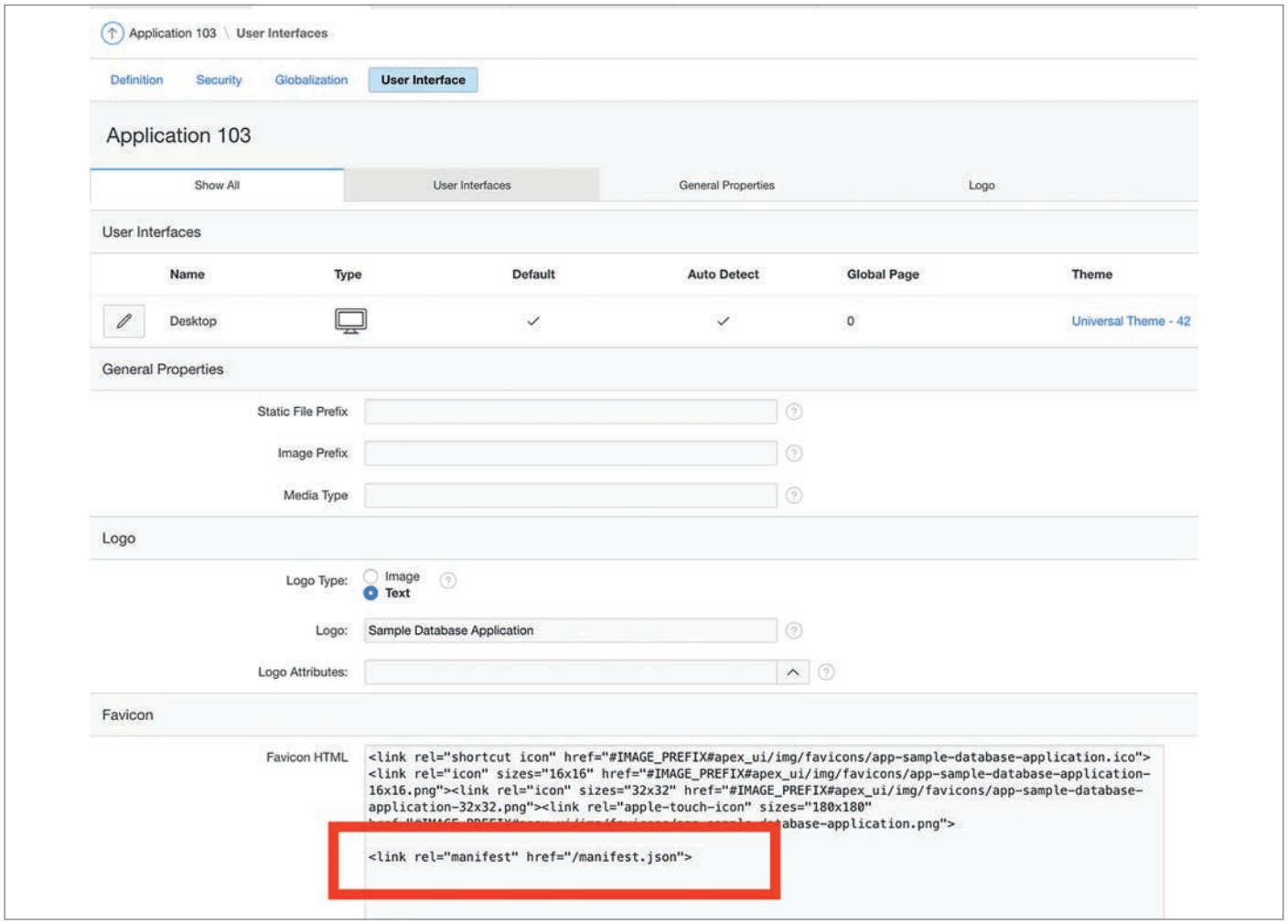



*Abbildung 2: Einbindung des Web-App-Manifests in die APEX-Anwendung (Quelle: Till Albert)*

ebenfalls aus der Datenbank im Browser nachgeladen.

Möchte man ein solches Konzept nun mit APEX umsetzen, so wird man schnell feststellen, dass dies nur eingeschränkt möglich ist. Grund dafür ist, dass APEX eben seine Inhalte nicht getrennt nach User Interface / Application Shell und Daten vom Server ausliefert, sondern diese vermischt. Das liegt vor allem auch daran, dass das meiste HTML auf der Seite des Servers generiert wird. Installiert man nun trotzdem einen Service Worker, der die APEX-Anwendung cacht, so wird eben die gesamte Seite im aktuellen Zustand in den Cache geschrieben, und nicht nur das Grundgerüst. Auch müsste sichergestellt werden, dass Komponenten, wie das Interactive Grid oder Interactive Report, keine Anfragen an den Server stellen, nachdem die Seite aus dem Cache geladen wurde beziehungsweise wenn die Anwendung keine Verbindung zum Internet hat. Das passiert zum Beispiel beim Filtern oder Aggregieren von Daten im Interactive Report. Je nach Anwendungsfall kann das für Anwendungen, auf die größtenteils lesend zugegriffen wird, ein Kompromiss sein. Eine offline-fähige CRUD-Anwendung wird man auf diese Weise aber nicht umsetzen können. Zumindest nicht mit den Standard-Mitteln von APEX. Eine Herangehensweise, mit zusätzlichem Aufwand, wäre eine Formular-Seite ohne Standard APEX Fetch und Save-Prozesse. Um das Formular mit Daten zu füllen, müsste nach dem Laden der Seite eine Anfrage an die Oracle- Datenbank via REST gemacht werden, wobei die REST-Anfrage mit purem JavaScript implementiert wird. Das Speichern der Daten müsste ebenfalls auf diese Weise geschehen. Mit dieser Herangehensweise wären bei der Anfrage der Daten diese von der APEX-Oberfläche getrennt, und könnten mithilfe eines Service Worker in eine Datenbank im Browser geschrieben werden.

Wie auch beim Web-App-Manifest ist auch bei der Erstellung des Service Worker die Seite pwabuilder.com eine große Hilfe. Damit lässt sich, je nach Anwendungsfall, der Service Worker nach Wunsch konfigurieren und dieser kann dann anschließend in die APEX-Anwendung eingebunden werden. Wichtig ist dabei wieder, dass die JavaScript-Datei des Service Worker auf dem Web Server liegt, wie auch das Manifest.

## Kurzer Exkurs: PWA mit Oracle JET

Wie man die Offline-Fähigkeit von Web-Anwendungen im Oracle-Umfeld einfacher umsetzen kann, lässt sich an Oracle JET zeigen. Das JavaScript-Framework basiert auf einer anderen Architektur. Im Gegensatz zu APEX erfolgt eine strikte Trennung von User Interface und den eigentlichen Daten. Somit lässt sich die Application Shell mithilfe von Service Worker im Browser cachen; die dynamischen Inhalte werden getrennt davon ange-

fragt und gespeichert. Von Oracle gibt es für das Arbeiten mit Offline-Daten in Oracle JET sogar ein eigenes Toolkit, das die Arbeit erleichtert *[1]*. Das sogenannte „Offline Persistence Toolkit" ist Open Source und kann in jedem JavaScript-Projekt, auch ohne Oracle Jet, genutzt werden.

## Fazit und Ausblick

Die Möglichkeiten der Technologie rund um Progressive Web Apps können eine normale APEX-Anwendung um interessante neue Features erweitern. Auch wenn die Offline-Fähigkeit derzeit nur eingeschränkt mit APEX realisierbar ist, so kann das Web-App-Manifest relativ schnell in eine APEX-Anwendung integriert werden und bringt damit bereits einen deutlichen Mehrwert für den Endanwender mit. Dieser Mehrwert wird in naher Zukunft noch vergrößert werden, wenn mehr und mehr native Features als APIs zur Verfügung gestellt werden. So sind für die nächsten Monate unter anderem ein API für den Zugriff auf die Kontakte auf dem Endgerät geplant sowie ein API für den Zugriff auf bestimmte Ordner des lokalen Dateisystems. Und das ist erst die Spitze des Eisbergs, denn die Liste an geplanten Schnittstellen für die Zukunft ist lang und macht bereits jetzt Freude auf mehr *[2]*.

## Quellen


[1] *https://github.com/oracle/offline-persistence-toolkit*

[2] *https://developers.google.com/web/updates/capabilities*


## Über den Autor

Till Albert ist als Berater bei der MT AG im Fachbereich APEX tätig und arbeitet vor allem mit den Technologien Oracle APEX und JavaScript. Zu diesen Themen gibt er auch regelmäßig sein Wissen auf Konferenzen weiter.


Till Albert
till.albert@mt-ag.com

# Know your Browser Dev Tools!

Daniel Hochleitner, IT Consulting & Development

Was sehr oft bei der Betrachtung von Oracle APEX untergeht, ist neben den datenbankseitigen Features, dass Oracle APEX auch ein Web-Entwicklungs-Framework darstellt. Also liegt es doch nahe, dass man sich neben den DB- Entwicklungstools, etwa dem Oracle SQL Developer, auch mit den Web-Entwicklungstools vertraut macht, um eben das volle Potenzial zu nutzen, den clientseitigen Code wie JavaScript oder CSS zu verbessern oder Fehler schneller zu finden. Going Full-Stack!

Seit dem Entwicklungsstart von Oracle APEX (1999) ist ein Teil des Frameworks mit Webtechnologie verwirklicht worden, es gibt also neben dem datenbankseitigen SQL- und PL/SQL-Code sowie Routinen auch einen großen Teil an clientseitigem Code wie JavaScript oder CSS. Erst kürzlich auf der APEX-Alpe-Adria-Konferenz 2019 in Kroatien zeigte der „Vater von APEX" Mike Hichwa eine interessante Folie, die die Code-Verteilung von APEX anhand der verwendeten Programmiersprachen veranschaulicht. Hier konnte man sehr gut erkennen, dass ein Großteil des APEX- Quellcodes eben nicht wie erwartet aus der Datenbank kommt, sondern tatsächlich aus JavaScript und CSS besteht (ca. 70%). Wenn man nun die beliebtesten Programmiersprachen – laut Stack-Overflow-Umfrage – damit vergleicht, stellt man schnell fest, dass APEX hier genau im Trend liegt *(siehe Abbildung 1)*.

Meine Beobachtungen der letzten Jahre zeigen jedoch etwas anderes, speziell

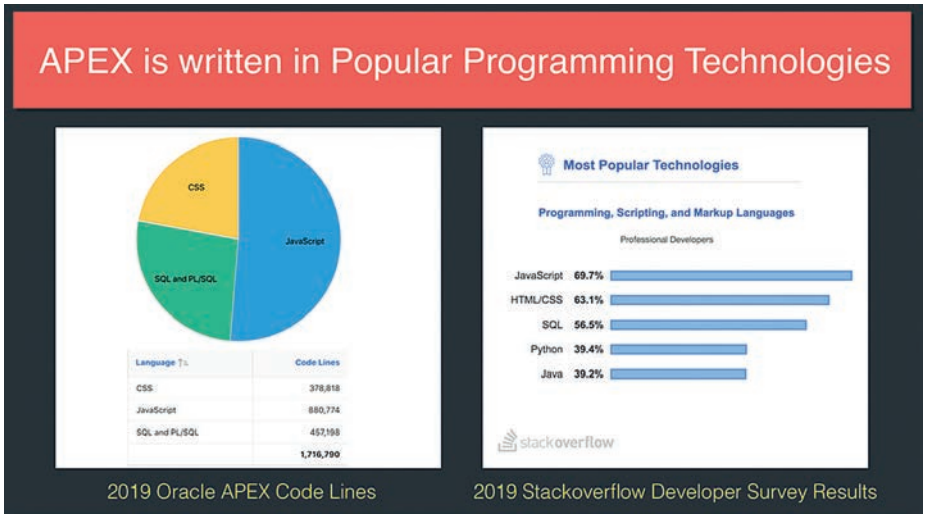



*Abbildung 1: Code-Verteilung Oracle APEX / Stack-Overflow-Umfrage (Quelle: Daniel Hochleitner)*

wenn es um das Know-how der Entwickler geht. Hier kommen, wie wahrscheinlich von vielen erwartet, die meisten Oracle-APEX-Entwickler aus der „Datenbank-Ecke“ und sind somit mehr mit der Oracle-Datenbank und dem kompletten Oracle Stack vertraut, anstatt im gleichen Maße mit JavaScript oder CSS zu hantieren. Genau deswegen liegt es doch nahe, dieses Know-how zu erweitern und uns ebenfalls mit den clientseitigen Werkzeugen vertraut zu machen, um eben Dinge zu verwenden, die uns aus der Datenbank-Entwicklung bekannt vorkommen und die wir bei der täglichen Arbeit schätzen.

Das wäre zum einen, den jeweiligen Code näher zu betrachten, aber auch Dinge wie Code-Autovervollständigung, Logging-Meldungen ausgeben, Code Debugging mit Breakpoints und vieles mehr.

Genau hier helfen uns die in jeden modernen Browser integrierten Developer Tools!

In diesem Artikel werden speziell die Entwicklungswerkzeuge von Google Chrome verwendet, was allerdings nicht bedeutet, dass die Tools etwa von Firefox besser oder schlechter sind. Jeder Browser hat hier und da seine Schwächen und Stärken; ich persönlich bevorzuge Google Chrome bei der täglichen Arbeit beziehungsweise Browser, die auf dem Open-Source-Browser-Kern „Chromium“ aufbauen. Die Entwicklungstools sind in diesem Fall nämlich über die verschiedenen Hersteller hinweg gleich. Jeder sollte jedoch am Ende die Werkzeuge verwenden, die ihm am besten liegen!

## Was leisten nun Browser Developer Tools?

Wie bereits erwähnt, haben die verschiedenen Browser-Hersteller jeweils leicht andere Ansätze, was die integrierten Entwicklertools betrifft, jedoch teilen sie sich alle gewisse Basis-Funktionalitäten, die vieles gemeinsam haben, zum Beispiel:

- HTML und den DOM Tree betrachten bzw. manipulieren
- JavaScript-Code und Funktionen ausführen
- CSS Styling anwenden
- clientseitiges Debugging und Logging
- Netzwerkverkehr und Performance analysieren
- Browser-Speicher verwalten
- verschiedene Bildschirmgrößen und Auflösungen testen

| | Windows/Linux | macOS |
|---|---|---|
| Open Dev Tools | F12 or Control+Shift+I | Command+Option+I |
| Open Inspector | Control+Shift+C | Command+Option+C |
| Open Console | Control+Shift+J | Command+Option+J |
| View Source | Control+U | Command+Option+U |

*Abbildung 2: Browser Shortcuts Google Chrome (Quelle: Daniel Hochleitner)*

| | Windows/Linux | macOS |
|---|---|---|
| Open Dev Tools | F12 or Control+Shift+I | Command+Option+I |
| Open Inspector | Control+Shift+C | Command+Option+C |
| Open Console | Control+Shift+K | Command+Option+K |
| View Source | Right Click > View Page Source | Right Click > View Page Source |

*Abbildung 3: Browser Shortcuts Firefox (Quelle: Daniel Hochleitner)*

In *Abbildung 2 und 3* wird gezeigt, welche Shortcuts für Chrome und Firefox verwendet werden können, um zum Beispiel die Entwicklertools oder direkt die JavaScript-Konsole zu öffnen. Wer die Shortcuts nicht verwenden möchte, kann auch jederzeit über das jeweilige Browser-Menü die verschiedenen Funktionen finden.

In *Abbildung 4* werden noch alle Reiter beziehungsweise Tabs oder „Panels“ der Developer Tools aufgeführt und kurz deren Hauptfunktionen erläutert.

Bevor wir nun mit dem ersten wirklichen Thema „Source-Code-Verwaltung“ starten, sehen wir in *Abbildung 5* einen stark vereinfachten Aufbau einer HTML-Seite und den daraus resultierenden DOM Tree. Der Browser verarbeitet und analysiert also das hierarchisch aufgebaute HTML der Seite und erstellt daraus den DOM in einer Baumstruktur mit Knoten und Parent-Child-Beziehungen. Auf den DOM Tree kann dann zum Beispiel mit JavaScript zugegriffen werden.

## Source-Code-Verwaltung

Dieses Thema ist in drei Bereiche unterteilt:

- **Elements Panel:** HTML-Struktur ansehen und bearbeiten sowie CSS Styling und Regeln festlegen
- **Console Panel:** JavaScript-Konsole, mit deren Hilfe JavaScript-Code und Funktionen ausgeführt werden können
- **Sources Panel:** Übersicht aller geladenen Webseiten-Ressourcen wie JavaScript- oder CSS-Dateien, mit integrierten Code Editor und Debugger

| | Description |
|---|---|
| Elements | Inspect & Edit elements and styles (CSS) in the DOM tree |
| Console | JavaScript Console to code and execute commands |
| Sources | Overview of all loaded resources of the page like documents, JS & CSS files including an code editor and debugger |
| Network | View, record and analyze any network traffic from the browser to the server side |
| Performance | Messure the performance of your page and gather runtime statistics |
| Memory | Gather memory statistics of your page |
| Application | View information about storage (Cookies, LocalStorage, IndexedDB), cache and your application |
| Security | View information about SSL and HTTPS connections |
| Audits | Perform audits like performance, PWA, best practices, accessibility, SEO or throttled network traffic |

*Abbildung 4: Übersicht Developer-Tools-Reiter (Quelle: Daniel Hochleitner)*

## Elements Panel

Starten wir also mit dem „Elements Panel". *Abbildung 6* zeigt, dass dieser Reiter in zwei Bereiche aufgeteilt ist; auf der linken Seite können wir die HTML-Struktur der Seite einsehen und bearbeiten und auf der rechten Seite werden uns die CSS- Regeln der verschiedenen HTML-Elemente, die das Aussehen der Webseite beeinflussen, angezeigt. Wird ein einzelnes HTML-Element auf der linken Seite markiert, sehen wir rechts die diesem Element zugehörigen CSS-Regeln beziehungsweise die Regeln, die übergeordnete Elemente dem ausgewählten Element vererben.

Auf beiden Panel-Seiten können auch Änderungen durchgeführt werden.

Eine Anmerkung wäre hier, dass die Änderungen, die hier getätigt werden, nicht persistent sind und nur zur Laufzeit der Seite ihre Gültigkeit behalten. Lädt die Seite neu, zum Beispiel mit F5 (Windows) oder CMD-R (macOS) beziehungsweise über den Reload Button des jeweiligen Browsers, verschwinden die getätigten Änderungen wieder. Später im Bereich „Sources Panel", zeige ich eine Möglichkeit, wie diese Änderungen trotzdem persistent abgespeichert werden können, sodass die Informationen beim Seiten-Wechsel oder Neuladen nicht verloren gehen.

Wenn nun Änderungen gemacht werden, können wir das im gesamten HTML der geladenen Seite tun, es können also im „Elements Panel" bereits vorhandene Elemente erweitert werden und zum Beispiel Attribute oder Klassen hinzugefügt werden. Es können jedoch auch komplette HTML-Strukturen von Hand eingefügt oder via Kontextmenü kopiert, aber auch dupliziert werden. Im Kontextmenu, das übrigens über einen Rechtsklick aufgerufen wird, gibt es allerdings auch praktische Dinge wie die Kopierfunktion des Pfades oder des JavaScript-Selektors des gewählten Elements. Somit kann später Zeit gespart beziehungsweise auch Schreibfehler verhindert werden, falls aus JavaScript heraus auf das Element zugegriffen werden soll.

Da uns der komplette HTML Source Code der Seite zur Verfügung steht und

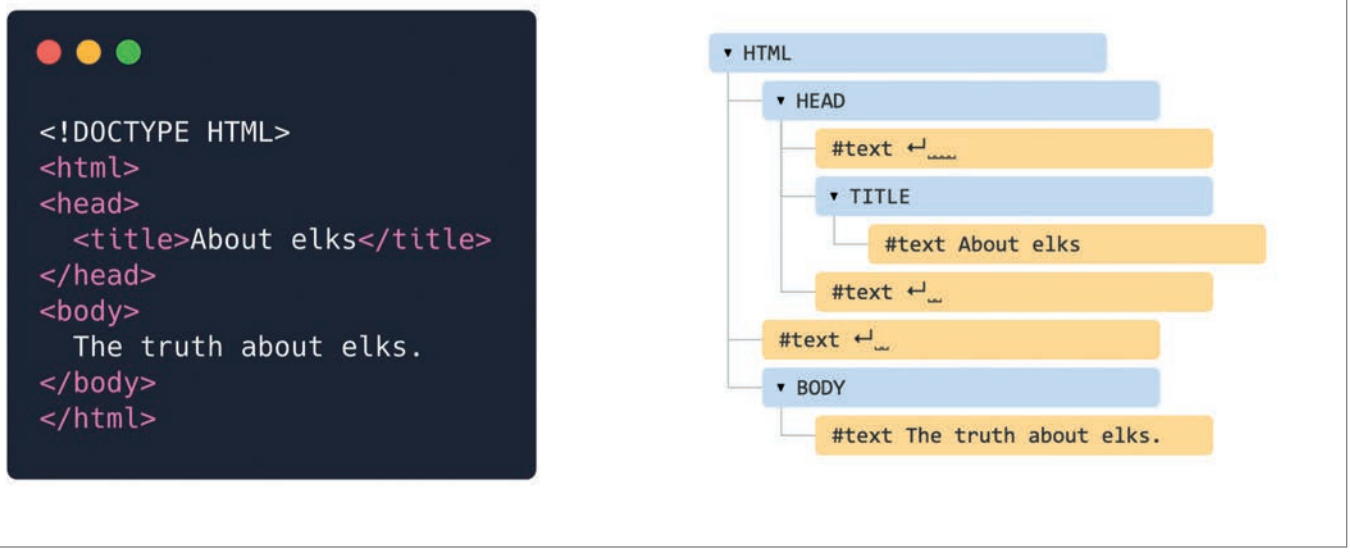



*Abbildung 5: Beispiel HTML und DOM Tree (Quelle: Daniel Hochleitner)*

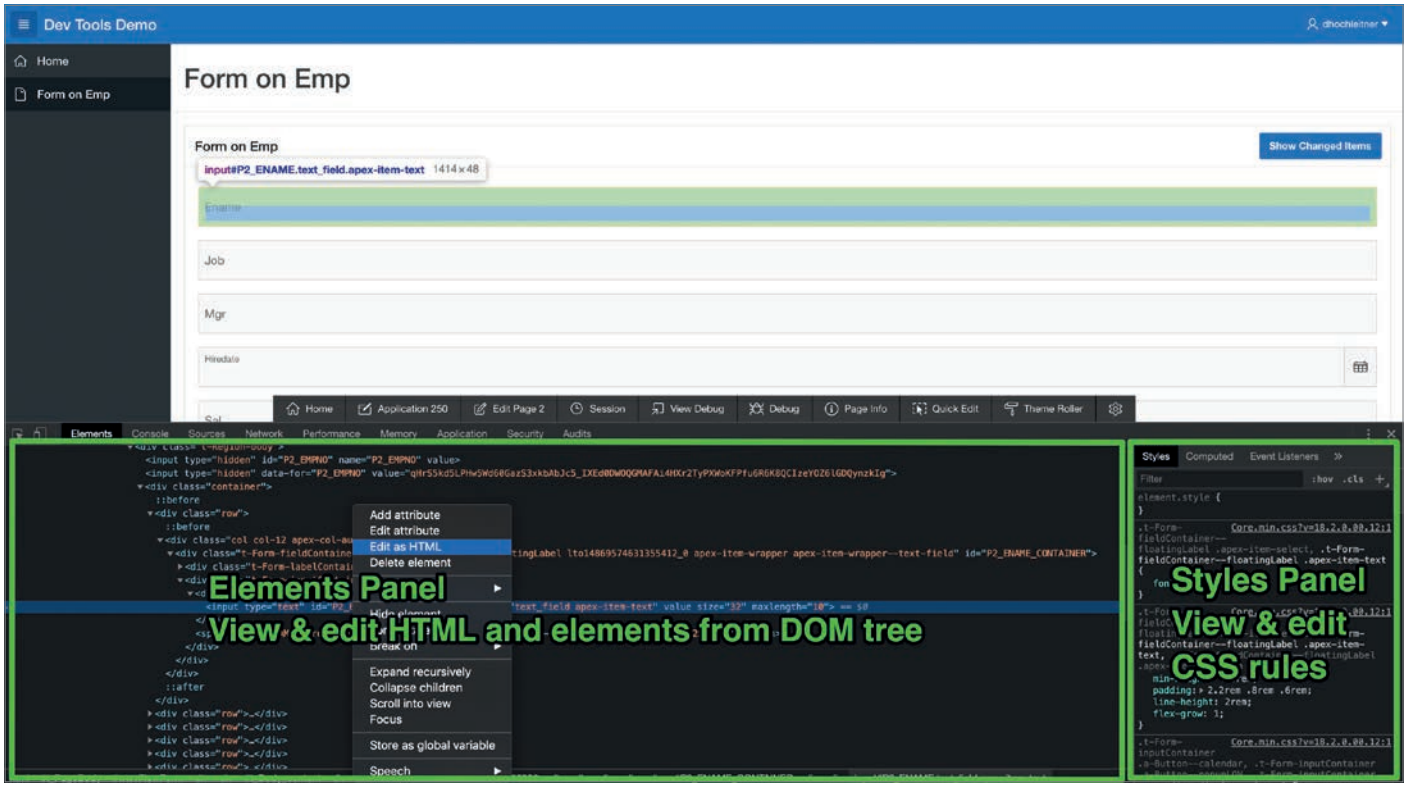



*Abbildung 6: Elements Panel (Quelle: Daniel Hochleitner)*

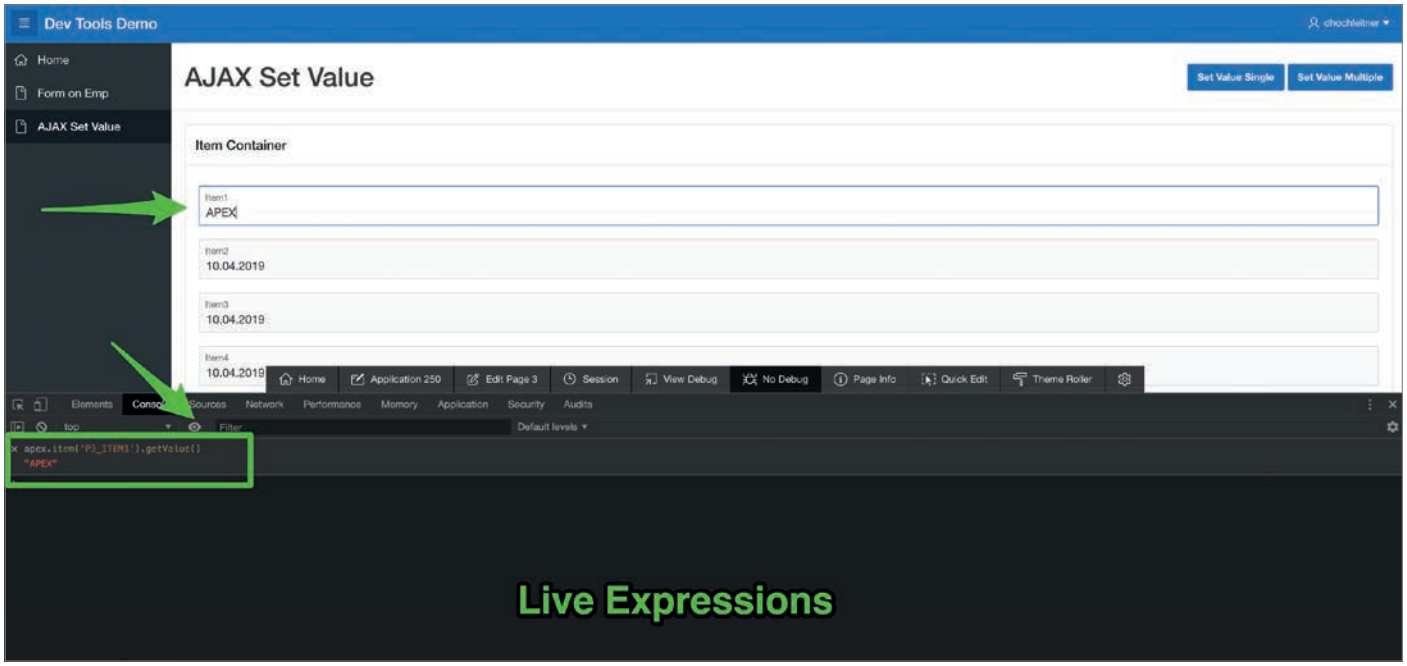



*Abbildung 7: Console Panel – Live Expressions (Quelle: Daniel Hochleitner)*

wir auch wirklich vieles daran ändern können, ist es schwierig, jede Einzelheit bis ins letzte Detail auszuführen. Ich würde jedem raten, sich mit den Möglichkeiten vertraut zu machen und live zu beobachten, welche visuellen Änderungen sich bei HTML- Änderungen ergeben. Das Gleiche gilt natürlich ebenso für CSS-Änderungen im rechten Bereich des Panels. Falls man zum Beispiel eine „background-color" eines Elements verändert, wird man das auch immer sofort visuell sehen können, in diesem Fall wird sich somit die Hintergrundfarbe ändern.

Wenn die Styling-Arbeiten im CSS abgeschlossen sind, können diese dann auf der jeweiligen APEX-Seite hinterlegt werden (Page à CSS à Inline). Falls die Änderungen jedoch für die komplette Anwendung gelten sollen, öffnen wir den seit APEX 5 verfügbaren Theme Roller, der in der APEX-eigenen Developer Toolbar zu finden ist. Dort gibt es eine Text-Area, in der benutzerdefiniertes CSS hinterlegt werden kann. Danach sind die Änderungen auf jeder Seite der APEX-Anwendung

verfügbar und aktiv. Beide Methoden machen unsere Arbeit also auch zugleich persistent.

## Console Panel

In diesem Panel können wir nun endlich auch JavaScript ausführen. Was bedeutet, dass wir uns aus der kompletten Syntax und allen Methoden, die uns „Vanilla JavaScript" (Standard JavaScript ohne zusätzliche Libraries) zur Verfügung stellt, bedienen können. Im Falle von APEX stehen aber weitaus mehr Libraries und APIs zur freien Verwendung bereit. Die bekannteste von ihnen wird wohl JQuery sein, eine Library, die uns Methoden zur DOM beziehungsweise HTML-Manipulierung an die Hand gibt. Auch die APEX JS Library selbst hat mittlerweile einen sehr großen Umfang (siehe Links zur API Doku im Quellenverzeichnis). Hier kann zum Beispiel direkt auf Items mit apex.item zugegriffen oder Success- und Error-Meldungen mit apex.message ausgegeben werden. Um den browsereigenen Speicher (Cookies, LocalStorage, SessionStorage) zu verwenden, steht apex.storage zur Auswahl. Es gibt noch unzählige weitere Funktionalitäten, die ich an dieser Stelle aber unerwähnt lasse.

Neben dem reinen JavaScript-Code, der hier geschrieben werden kann, können aber auch Komfort-Features wie Code-Autovervollständigung verwendet werden. Darüber hinaus können ebenso Function Returns oder Variablen-Werte zur Anzeige gebracht werden. Die JavaScript-Konsole kann neben der Möglichkeit, darin zu programmieren, auch für Logging oder Debugging-Ausgaben verwendet werden. Die Zeiten, in denen ein JavaScript-Alert-Dialog benutzt wird, um uns gewisse APEX Item oder Variablen-Werte anzuzeigen, sollten nun wirklich endgültig vorbei sein!

Mit console.log oder den APEX-eigenen Funktionen des Namespace apex.debug kann man nun Konsolen-Ausgaben erzeugen, im Falle von apex.debug werden diese nur generiert, falls der Debug-Mode von APEX aktiviert ist. Das hat den netten Vorteil, dass solche Aufrufe durchaus auch in produktivem Code benutzt werden können, da nicht ständig geloggt wird, sondern nur zu Debugging-Zwecken, wenn wir es wirklich benötigen. Viele externe APEX-Plug-ins benutzen diese Library durchaus häufig. Hier lohnt also immer auch ein Blick in die Konsolen-Ausgabe. Aber auch APEX selbst benutzt diese Methoden, um zum Beispiel Informationen zu Dynamic Actions auszugeben. Beide Libraries haben auch Funktionen, um den verschiedenen Debug-Leveln zu entsprechen, zum Beispiel Info / Warning / Error.

Ein weiteres tolles Feature sind die sogenannten „Live Expressions" der Konsole. Dort können wir einen JavaScript-Aufruf hinterlegen, der dann live, also stetig, ausgeführt wird. Ich benutze das gern, um mir Werte von Items auszugeben, ohne dabei auf Events hören zu müssen, oder um die ständige Neueingabe von console.log zu verringern. In *Abbildung 7* sieht man die Live Expression in Aktion, in der mit apex.item (‚P1_ITEM').getValue() immer wieder der aktuelle Wert des Items P1_ITEM zur Anzeige gebracht wird.

## Sources Panel

Das „Sources Panel" ist zuzusagen die In-Browser IDE, also eine Entwicklungsumgebung. Wie wir es von einer IDE gewohnt sind, werden uns im linken Bereich alle geladenen Dateien der Webseite, also JavaScript- und CSS- Dateien, als Baumstruktur angezeigt. Wählen wir nun eine Datei aus, wird der komplette Source

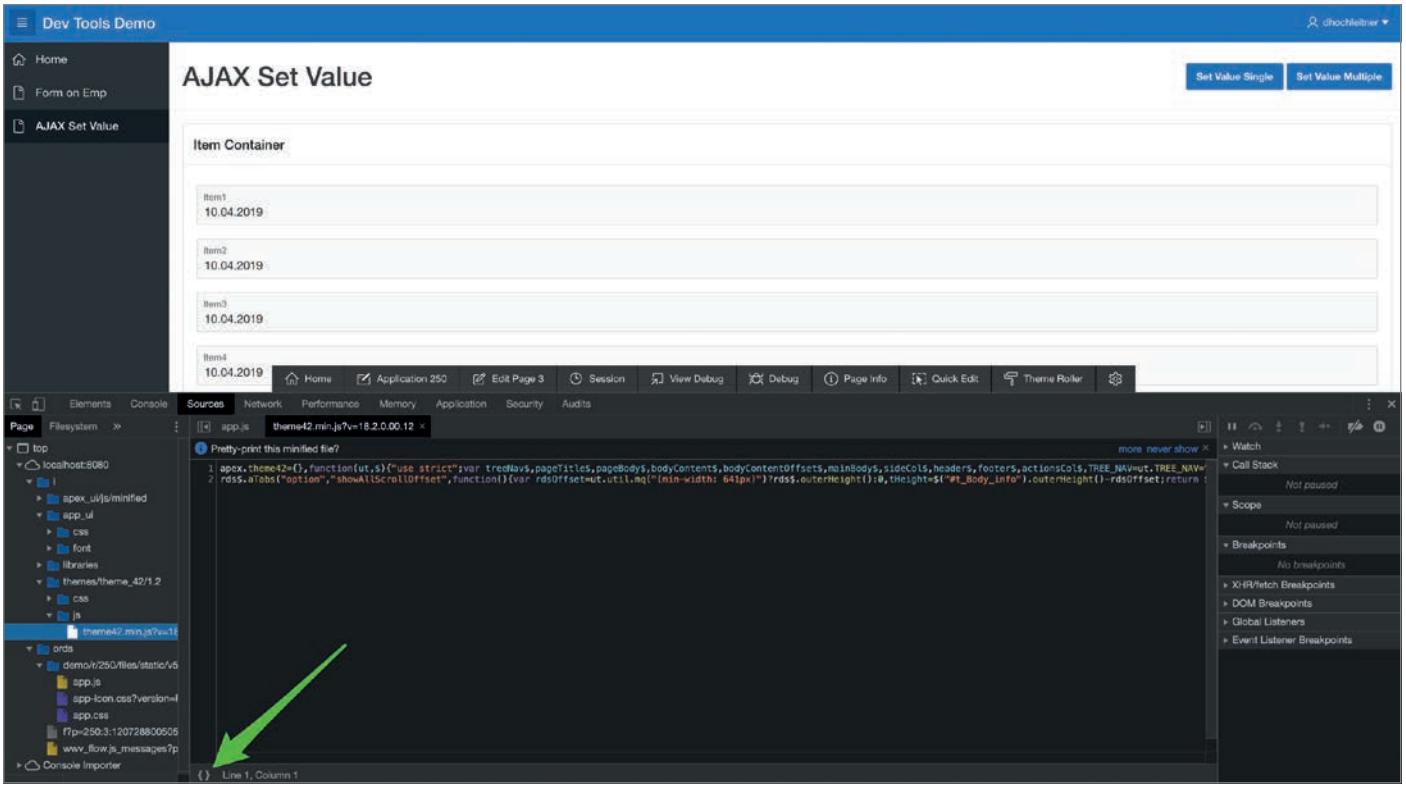



*Abbildung 8: Sources Panel – Pretty Print minified Code (Quelle: Daniel Hochleitner)*

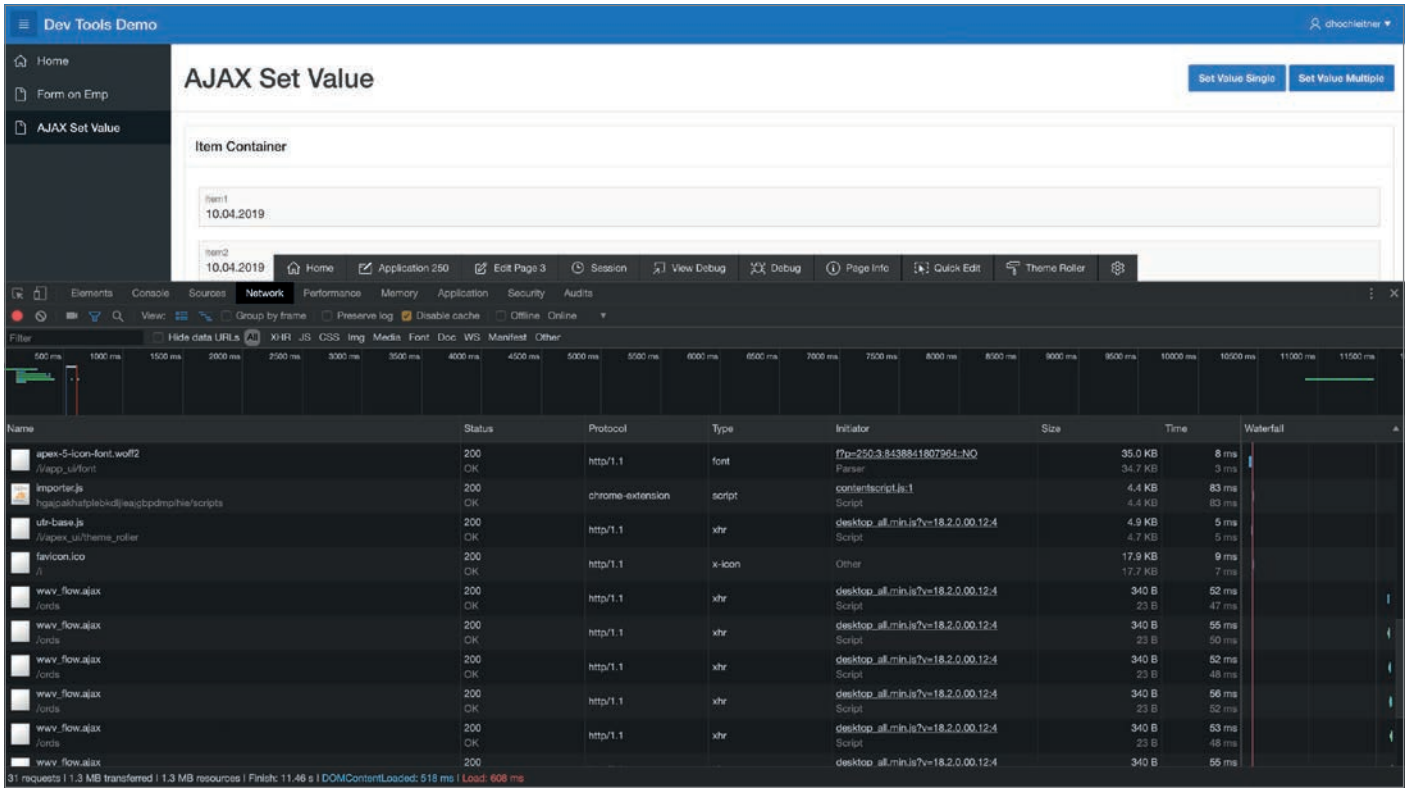



*Abbildung 9: Network Panel – Übersicht (Quelle: Daniel Hochleitner)*

Code im rechten Bereich präsentiert. Handelt es sich um eine „minified" Datei, also beispielsweise myfile.min.js, können wir mithilfe der Developer-Tools diese auch wieder schön und für Menschen lesbar darstellen *(siehe Abbildung 8)*.

Minification wird im Web-Umfeld benutzt, um Dateien zu verkleinern (Entfernung von Kommentaren und Whitespace, kürzere Variablen-Namen), um Netzwerk-Ressourcen zu sparen und die Übertragung vom Server zum Client zu beschleunigen, aber auch um die Verarbeitung im Browser zu beschleunigen, indem der Source Code optimiert wird (Entfernung von unnötigem Code und Routinen, Verwendung von Kurzschreibweisen und vieles mehr).

Im Source Code selbst können nun ebenfalls Änderungen getätigt, Debugging und Logging-Einträge hinzufügt oder Breakpoints gesetzt werden. Ganz rechts gibt es dann noch weitere Debugging-Werkzeuge in Form von Buttons, um zum Beispiel die Script-Ausführung zu pausieren oder eben das schrittbasierte Debugging zu erlauben (Step into next Function, Step out of current Function, Step over Function etc.). Das sind wir von Tools wie dem SQL Developer gewohnt, und somit kann nun Ähnliches auch mit clientseitigem Code getan werden.

Aber auch hier sind unsere Änderungen nicht persistent. Wie bereits weiter oben angekündigt, steht uns in diesem Panel ein kleiner Helfer zur Seite. Die Funktion nennt sich „Local Overrides" und ist im linken Bereich über dem Dateien-Baum zu finden (Tab-Reiter). Dort kann nun eine Art Synchronisierung zwischen Dateien des Servers und lokalen Dateien eingerichtet werden. Als Erstes wählen wir einen leeren lokalen Ordner unseres Client-Rechners aus. Sobald wir nun Änderungen an den Dateien des Servers durchführen, wird eine Kopie dieser Datei (mit relativem Pfad) in diesem Ordner gespeichert. Diese Datei beinhaltet dann all unsere Änderungen und wird fortan beim Laden der Seite benutzt. Sind wir nun mit dem Ergebnis zufrieden und haben die Entwicklung abgeschlossen, kann die lokale Datei zurück auf den Server geladen beziehungsweise direkt in der APEX-Applikation unter Shared Components à Static Application Files ablegt werden.

## Network Panel

Da wir das Kapitel der Source-Code-Verwaltung abgeschlossen haben, können wir mit dem „Network Panel" fortfahren. Hier kann der komplette Netzwerkverkehr unserer Webseite betrachtet und genau analysiert werden. *Abbildung 9* zeigt beispielhaft den Netzwerk-Traffic einer APEX-Seite, also konkret das Hin- und Hersenden von Daten zwischen Serverseite und Browser.

Tabellarisch wird hier jede einzelne Ressource und Interaktion angezeigt und wir sehen Details wie:

- Ressourcen-Namen (z.B. myfile.js oder myfile.css)
- HTTP-Methode (z.B. GET oder POST)
- HTTP-Status-Code
- HTTP-Protokoll
- Größe des Payload
- Zeit-Informationen und Wasserfall-Diagramm
- und vieles mehr.

Anhand dieser Informationen kann schon vieles daraus abgeleitet werden: Werden Dateien komprimiert übertragen? Wie lange dauern einzelne Übertragungen? Funktioniert das Browser Caching oder gibt es Fehler beim Laden von Ressourcen? Warum lädt eine APEX-Seite nach dem initialen Laden des Hauptdokuments noch Daten von der Server-Seite nach (z.B. wwv_flow.ajax Einträge)? Somit

können gewisse Rückschlüsse gezogen werden, wenn es um Performance-Aspekte von Anwendungen geht.

Wird ein Eintrag aus der Liste ausgewählt, sehen wir dessen Details *(siehe Abbildung 10)*. Dort bekommen wir weitere und noch detaillierte Angaben zu diesem einen Request: Welche Informationen wurden zum Server gesendet? Was gibt der Server als Response zurück und welche HTTP-Header wurden dabei verwendet?

Falls der Server mit HTML, CSS, JavaScript beziehungsweise JSON antwortet, bekommen wir das oft mit Syntax Highlighting oder formatiertem Code präsentiert.

Via Kontextmenü (Rechtsklick auf einen Eintrag) stehen weitere Möglichkeiten zur Verfügung. Ich verwende hier gern das Kopieren des Request. Entweder kann die URL kopiert werden oder bereits fertige Aufrufe für fetch (JavaScript) oder cURL (Bash, Terminal, Powershell).

## Application Panel

Im „Application Panel" stehen hauptsächlich Funktionen zur Verfügung, um den lokalen Browser-Speicher zu verwalten, aber auch Dinge wie applikationsweite Informationen, die vor allem für PWAs (Progressive Web Apps) von Bedeutung sind (manifest.json oder Service Workers).

Im APEX-Umfeld wird man sich allerdings hauptsächlich mit dem lokalen Speicher beschäftigen. Hier bekommen wir Informationen zu Cookies, LocalStorage und SessionStorage. Die beiden Letzteren sind sehr ähnlich aufgebaut (Key-Value Store), einziger Unterschied ist, dass SessionStorage eine begrenzte Gültigkeit aufweist – sobald die jeweilige Applikation oder das Browser-Fenster geschlossen wird, wird dieser Speicher automatisch gelöscht.

Neben dem reinen Betrachten des Speichers können die Einträge bearbeitet, neue Einträge hinzugefügt oder eben der Speicher geleert werden.

Den lokalen Speicher können wir beispielsweise nutzen, um gewisse Daten der Serverseite lokal zu cachen, um den Server zu entlasten oder um sich Einstellungen, die später in JavaScript benötigt werden, zu merken. Auch APEX selbst nutzt es an einigen Stellen, ein Beispiel wäre die Collapsible Region. Hier wird der Zustand des Ein- oder Ausklappens gespeichert, um beim nächsten Laden der Seite die gespeicherte Einstellung wiederzuverwenden.

## Audits Panel

Wie der Name schon verrät, können in diesem Bereich gewisse Audits oder auch Tests unserer Anwendung oder Webseite ausgeführt werden. Die Testfälle teilen sich in folgende Bereiche auf:

- Performance
- Mobile / Desktop
- Progressive Web App
- Best Practices
- Accessibility
- SEO

Hier kommt es am Ende stark auf den jeweiligen Einsatzzweck unserer Anwendung an und darauf, welche Testfälle für uns am besten geeignet sind. Hat man eine reine In-House-Anwendung, wird man sich den Test für SEO (Search Engine Optimization) wohl sparen können. Ich persönlich benutze aus den verfügbaren Kategorien meistens Performance und Accessibility, weil das meiner Meinung nach sehr wichtige Themen sind. Eine langsame Anwendung macht niemandem Spaß. Auch sollte sichergestellt sein, dass die Seite von jedem benutzbar ist, etwa durch den Einsatz von Sprachausgabe und Bildschirmlupe.

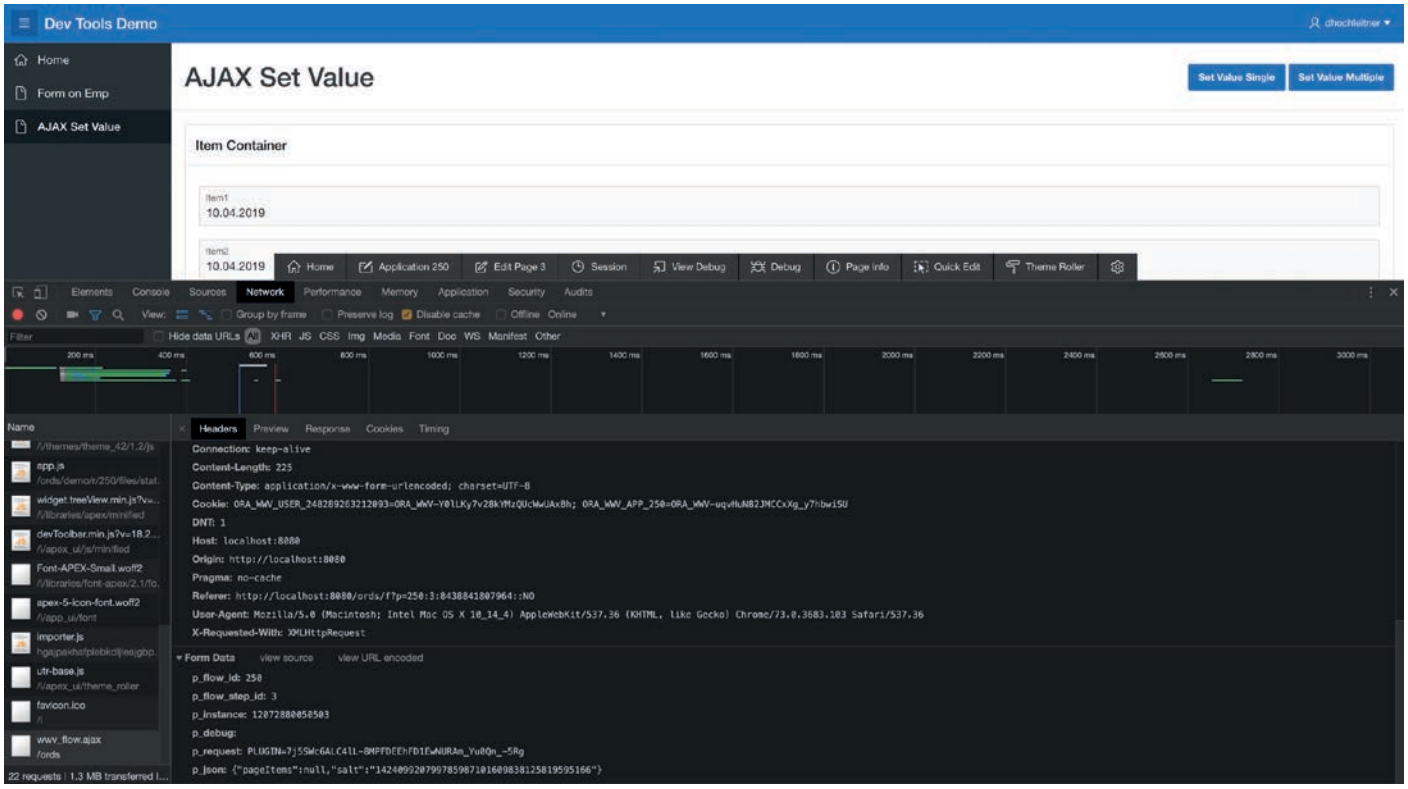



*Abbildung 10: Network Panel – Detail-Ansicht (Quelle: Daniel Hochleitner)*

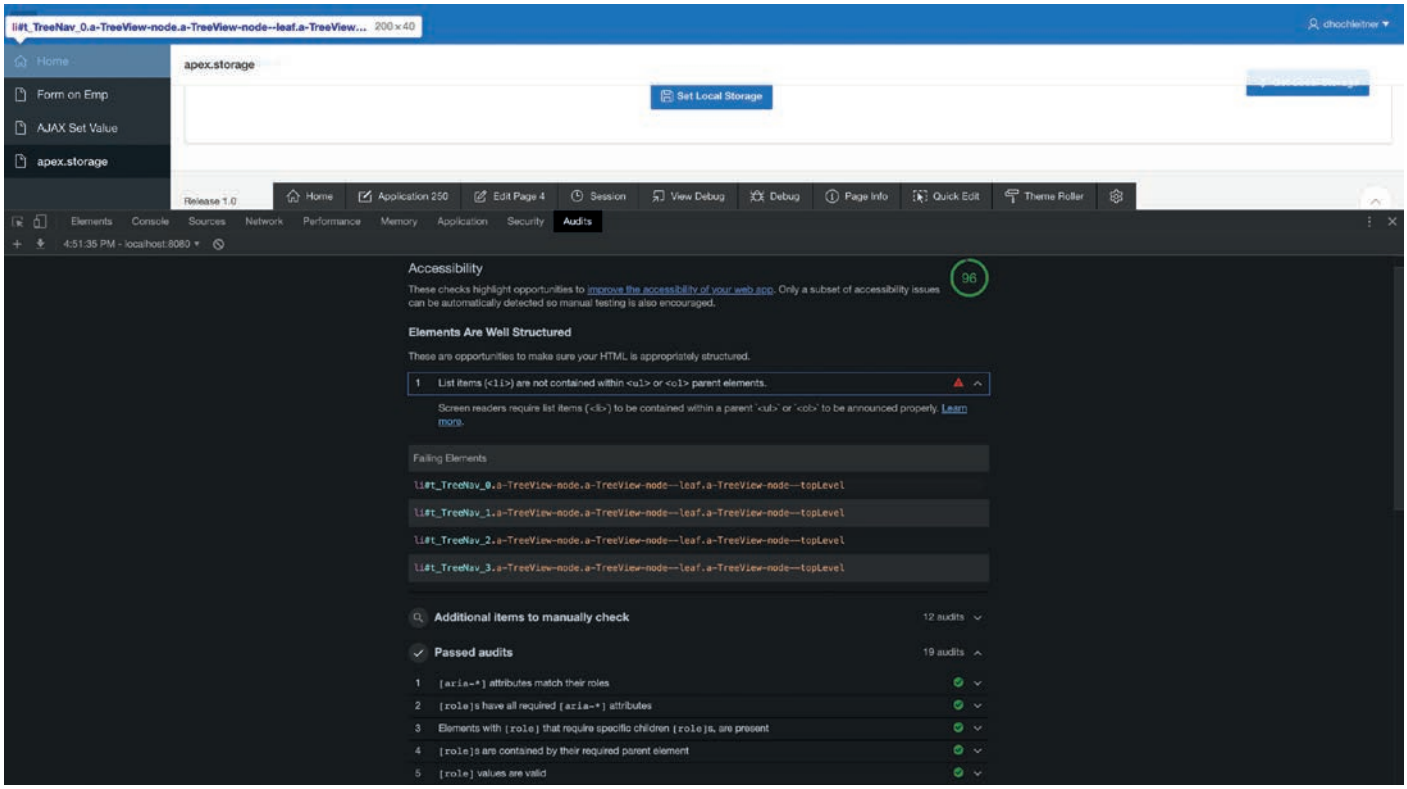



*Abbildung 11: Audits Panel – Summery Page Accessibility (Quelle: Daniel Hochleitner)*

Wenn wir einen Test starten, dauert es meistens ein paar Augenblicke, bis dieser abgeschlossen ist, danach wird uns eine Summery Page mit erfolgreichen beziehungsweise fehlerhaften Testfällen gezeigt *(siehe Abbildung 11)*.

Im Bereich Accessibility macht APEX standardmäßig schon vieles richtig, man muss sich jedoch auch an den Standard halten, also zum Beispiel Breadcrumbs benutzen anstatt eine Standard-Region.

Für das Testen von mobilen Ansichten verwende ich statt des „Audits Panel" die Device Toolbar; den Aktionsbutton dazu findet man auf Höhe der Developer-Toolbar- Reiter (linke Seite). Dort angekommen, können wir die Ansicht der Webseite verändern und aus vordefinierten Geräteklassen wählen, etwa iPhone, iPad, Google Pixel oder Ähnlichen, neben der nativen Auflösung des gewählten Gerätes wird auch der richtige User Agent dem Server bekannt gemacht. Es gibt dort noch weitere Funktionen, um beispielsweise Screenshots für verschiedene Bildschirmgrößen beziehungsweise Auflösungen anzufertigen.

An dieser Stelle möchte ich das Thema gern abschließen; die Developer Tools haben noch weitaus mehr Funktionen, wie den Performance-Reiter, in dem man das komplette JavaScript Callstack untersuchen kann, aber das würde diesen Rahmen sprengen. Ich rate jedem, der etwas im Web- oder APEX-Umfeld macht, sich die Browser Dev Tools genauer anzusehen, Übung macht schließlich den Meister! Ich habe versucht, die Themen zu behandeln, die mir persönlich am wichtigsten sind und die mir am besten bei der clientseitigen Entwicklung helfen. Ich hoffe, ich konnte dem einen oder anderen die Angst vor der Browser-Entwicklung nehmen und den Einstieg in diese Welt erleichtern. Happy Coding! :)

## Quellen


[1] Oracle APEX 19.1 JavaScript API Docs: *https://docs.oracle.com/en/database/oracle/application-express/19.1/aexjs/index.html*

[2] JQuery API Docs: *https://api.jquery.com*

[3] Minify Resources: *https://developers.google.com/speed/docs/insights/MinifyResources*

[4] DOM Tree: *https://javascript.info/dom-nodes*

[5] Chrome Dev Tools Docs: *https://developers.google.com/web/tools/chrome-devtools*

[6] Firefox Dev Tools Docs: *https://developer.mozilla.org/en-US/docs/Tools*

[7] Chrome Dev Tools News: *https://developers.google.com/web/updates/2019/*


## Über den Autor

Daniel Hochleitner ist freiberuflicher Software-Entwickler mit Fokus auf Oracle APEX und der Oracle-Datenbank. Er bringt sich gern in Open-Source-Projekten ein und teilt sein Wissen auch auf Konferenzen oder lokalen User-Group-Meetings.


Daniel Hochleitner
dhochleitner@posteo.de


# APEX 19.2: Was ist neu?

Carsten Czarski, Oracle

Auch 2019 hält das APEX-Entwicklerteam sein Versprechen ein, zwei Releases pro Jahr freizugeben. Nachdem im März die Version 19.1 erschienen war (das Red Stack Magazin berichtete), liegt nun Release 19.2 mit zahlreichen neuen Funktionen vor. Wie immer sind nicht nur komplett neue Funktionen (wie „Faceted Search" oder das neue „Team Development") dabei, es wird auch die Funktionalität früherer Releases abgerundet und vervollständigt.

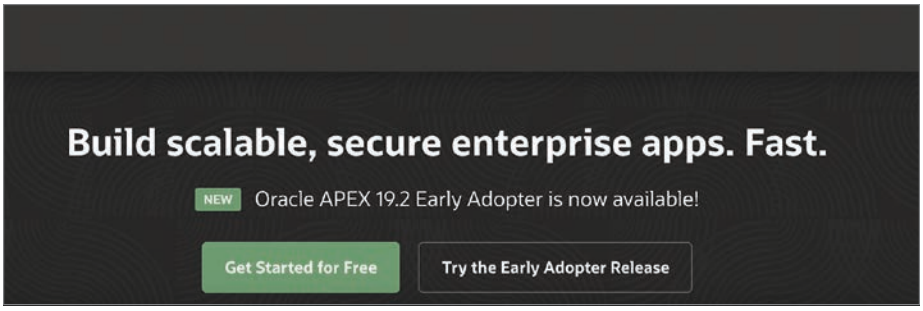



*Abbildung 1: APEX 19.2 Ankündigung (Quelle: Oracle)*

## Faceted Search

Das wohl auffälligste neue Feature in APEX 19.2 ist die Facettensuche (Faceted Search), die vielen von Verkaufsseiten im Internet bekannt sein dürfte: Die sogenannten Facets sind oft auf der linken Seite zu sehen und erlauben das Setzen von Filtern wie Marke, Preisrahmen oder Farbe. Oft wird zu jeder Ausprägung auch angezeigt, wie oft diese im Datenbestand vorkommt. Nimmt man Änderungen vor, wird die Ergebnisliste, und auch die Information zur Anzahl der Vorkommen im Datenbestand, sofort aktualisiert. *Abbildung 2* zeigt eine APEX-Anwendung mit Facettensuche.

Der einfachste Weg, eine Seite mit Faceted Search zu erzeugen, ist der *Create Page Wizard:* Wählt man **Create Report Page** aus, so erscheint **Face-**

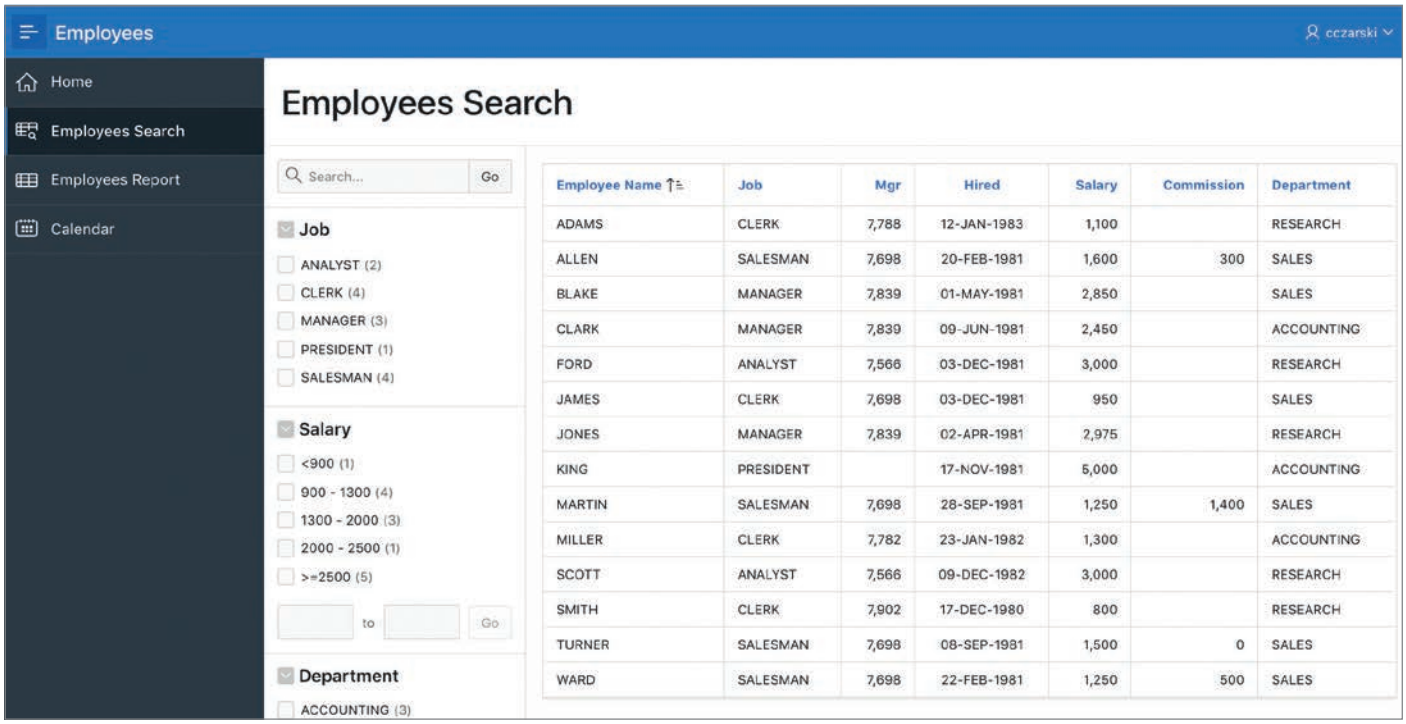



*Abbildung 2: APEX Faceted Search in Aktion (Quelle: Carsten Czarski)*


*Abbildung 3: „Faceted Search" als neue Option für eine Berichtsseite (Quelle: Carsten Czarski)*

**ted Search** als neue Option *(siehe Abbildung 3)*.

Sobald man eine Tabelle als Datenquelle ausgewählt hat, zeigt APEX Informationen zu deren Spalten und auch zu den Daten an (APEX analysiert die Tabellen im Hintergrund und speichert Informationen zu deren Daten in einem Dictionary Cache ab). Dies geht weit über einfache Datenbank-Statistiken hinaus – anhand der Informationen kann APEX nicht nur vorschlagen, für welche Spalten ein Facet gebildet werden sollte – es kann darüber hinaus auch Fremdschlüssel zurückverfolgen, nötige Wertelisten bestimmen oder Bereichsgrenzen für Spalten mit stetigen Werten berechnen *(siehe Abbildung 4)*.

Im Page Designer *(siehe Abbildung 5)* können Facets gelöscht, neue hinzugefügt oder Details zu den Facets verändert werden. APEX bietet verschiedene Optionen zu den einzelnen Facets an. Beispiele sind:

- Sollen „Counts" (Anzahl der Zeilen mit der jeweiligen Ausprägung) ermittelt und dargestellt werden oder nicht?
- Sollen aktivierte Elemente an den Anfang der Liste gestellt werden?
- Sollen Ausprägungen ohne Ergebniszeilen angezeigt oder versteckt werden?

Im Page Designer ist die Anatomie der Faceted-Search-Seite gut zu erkennen. Eine Region vom Typ Classic Report zeigt die Ergebnisse an und eine vom Typ Faceted Search hält die Facets. Für ein künftiges Release ist geplant, auch andere Regionstypen für die Ergebnisanzeige zu unterstützen (zum Beispiel JET Charts).

Page Designer erlaubt das komplett manuelle Erstellen einer Faceted-Search-Seite, indem zunächst ein Classic Report und dann eine Region vom Typ Faceted

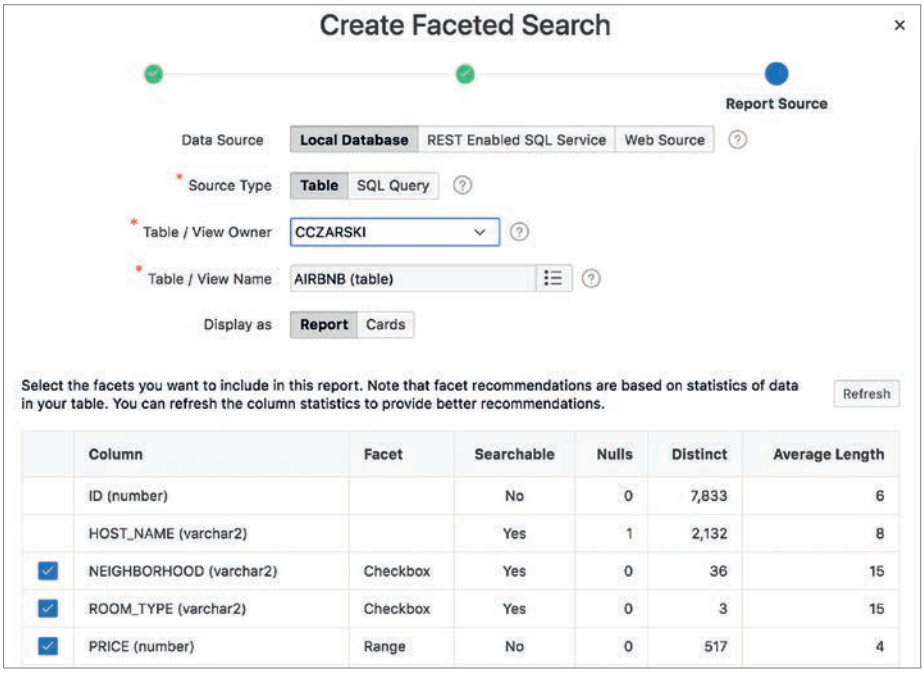



*Abbildung 4: APEX Create Faceted Search Page wizard (Quelle: Carsten Czarski)*

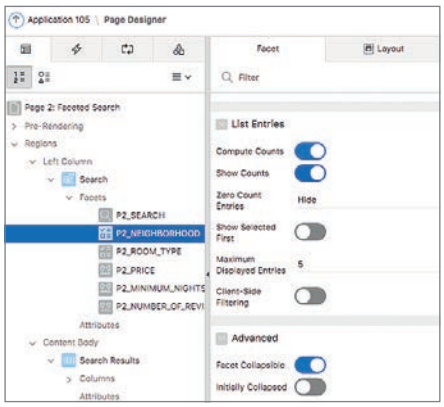



*Abbildung 5: Faceted Search im Page Designer (Quelle: Carsten Czarski)*

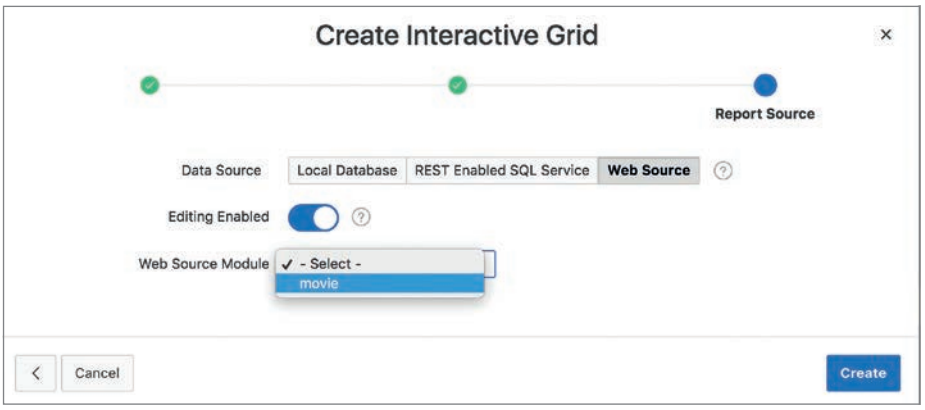



*Abbildung 6: Interactive Grid unterstützen externe Datenquellen (Quelle: Carsten Czarski)*

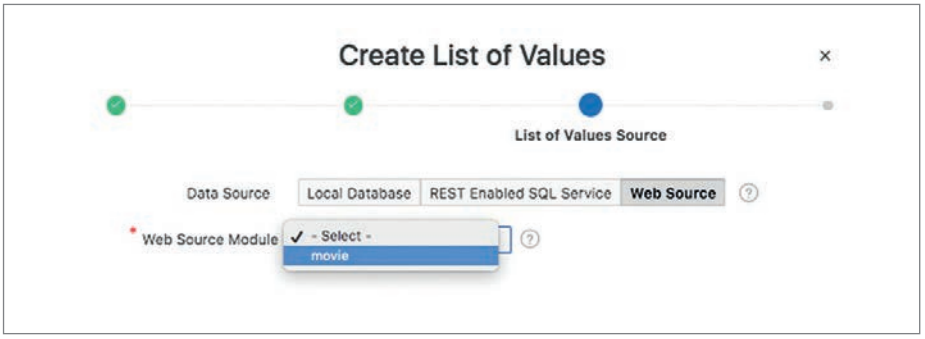



*Abbildung 7: Wertelisten (LOV) können auch externe Datenquellen ansprechen (Quelle: Carsten Czarski)*

Search mit den einzelnen Facets erzeugt wird. So kann Faceted Search nicht nur mit Tabellen, sondern auch mit SQL-Abfragen, REST-Enabled SQL oder REST-Services (Web Source Modules) genutzt werden.

## Externe Datenquellen für das Interactive Grid

Mit APEX 18.1 wurden externe Datenquellen, wie REST-Services oder REST-Enabled SQL, und deren Unterstützung durch Read-Only-Regionstypen (Reports, Charts und andere) eingeführt. APEX 19.1 erweitert diese Unterstützung mit der neuen Form Region für Formularseiten; APEX 19.2 macht die Unterstützung mit dem Interactive Grid komplett.

Als „Nebeneffekt" wird Interactive Grid auch bei Nutzung lokaler Tabellen oder SQL-Abfragen deutlich flexibler: Nun kann eine PL/SQL Function returning SQL Query als Datenquelle angegeben werden, was bislang nicht möglich war. So kann der Developer ein Interactive Grid mit einer dynamisch erzeugten SQL-Abfrage als Datenquelle erstellen *(siehe Abbildung 6)*.

## Neuerungen für Lists of Values (LOV)

Der Bereich der Wertelisten (LOV) wurde in APEX 19.2 komplett renoviert. Für gemeinsam verwendete (Shared Components) LOV können nun auch externe Datenquellen (Web Source Modules, REST-Enabled SQL) verwendet werden *(siehe Abbildung 7)*.

Eine solche, externe Daten nutzende Werteliste kann genauso benutzt werden wie eine, die eine lokale SQL-Abfrage verwendet. *Abbildung 8* zeigt die Nutzung für ein Radiogroup-Item auf einer APEX-Seite.

Doch nicht nur die Unterstützung externer Datenquellen ist neu – APEX 19.2 bringt deutlich mehr Struktur in die Definition der Datenquelle einer LOV: So muss eine SQL-Abfrage nicht mehr zwingend zuerst die Display- und dann die Return-Spalte selektieren – vielmehr kann eine beliebige Abfrage verwendet werden. Im Column Mapping werden die Abfragespalten dann der konkreten Verwendung zugeordnet *(siehe Abbildung 9)*.

Neben Display und Return werden auch Spalten zur Gruppierung und zur Anzeige eines Icons unterstützt. Letztere muss Font APEX-Syntax zurückliefern (beispielsweise „**fa fa-apex**"). Die Additional Display Columns sind vor allem für die neue Popup LOV von Bedeutung *(siehe Abbildung 10)*.

Das neue Popup-LOV-Formularelement stellt eine ganz wesentliche Verbesserung gegenüber älteren APEX-Versionen dar. Beim Erstellen eines solchen im Page Designer ergeben sich zunächst kaum Unterschiede, wohl aber beim Ergebnis *(siehe Abbildung 11)*.


*Abbildung 8: Nutzung der REST-Werteliste auf einer APEX-Seite (Quelle: Carsten Czarski)*

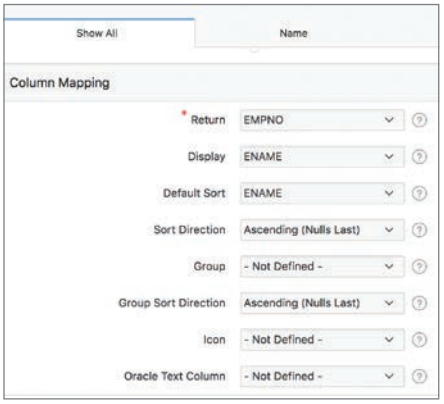



*Abbildung 9: Zuordnung von Abfragespalten in der LOV-Definition (Quelle: Carsten Czarski)*

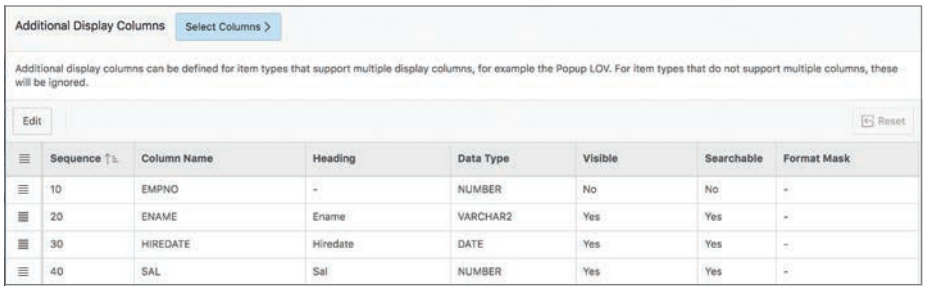



*Abbildung 10: Zusätzliche Display-Spalten festlegen (Quelle: Carsten Czarski)*

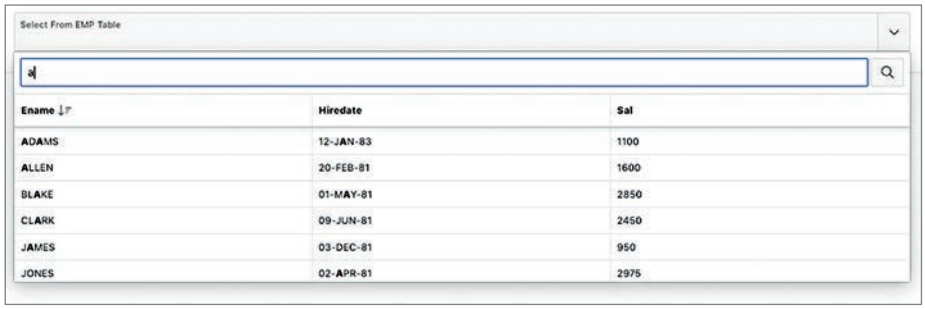



*Abbildung 11: Das neue Popup-LOV-Element in APEX 19.2 (Quelle: Carsten Czarski)*

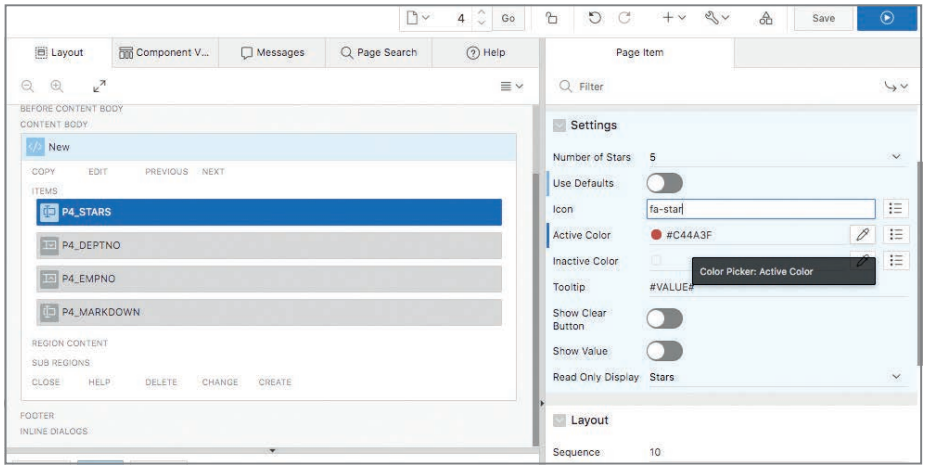



*Abbildung 12: Der neue Star Rating Item Type im Page Designer (Quelle: Carsten Czarski)*

Mithilfe zusätzlicher Display-Spalten kann die neue Popup LOV wesentlich mehr Information zum Anwender transportieren. Die eingebaute Suche erfolgt standardmäßig case-insensitiv über alle Display-Spalten, unterstützt aber noch andere Optionen:

- *Search as you type* führt die Suche sofort aus, ohne dass der Anwender erst [Enter] drücken muss.
- Ist in der LOV-Definition eine *Oracle-TEXT*-Indexspalte definiert, so wird dieser Index für die Suche verwendet. Damit können Oracle TEXT Features wie Fuzzy- oder linguistische Suche auch für die Suche in einer APEX Popup LOV genutzt werden.
- Für Wertelisten, die ein Web Source Module als Datenquelle nutzen, kann der Suchbegriff direkt zum REST-Service gesendet werden, sodass die Suche beim REST-Service selbst stattfindet.

## Neue Page Item Types

APEX 19.2 führt zwei neue Typen für Seitenelemente ein. Das Star Rating Item ist nahezu selbsterklärend: Der Entwickler legt das darzustellende Symbol (normalerweise ein Stern) und die maximale Anzahl der Icons fest – der Anwender wählt einfach die Anzahl der Symbole aus und vergibt so eine Bewertung von 1 bis N *(siehe Abbildung 12)*.

Der neue Markdown Editor *(siehe Abbildung 13)* ist besonders interessant, wenn der Anwender die Möglichkeit haben soll, einfache Formatierungen zu verwenden – ohne dabei mit der Formatierungsfülle des Rich Text Editor überfordert zu werden. Das populäre Markdown-Format unterstützt eine übersichtliche Auswahl von Formatoptionen wie Fett, Kursiv, Durchgestrichen, Zitat oder Code-Darstellung. Natürlich steht der Rich Text Editor auch weiterhin zur Verfügung.

Der Vorteil der Markdown-Syntax *(siehe Listing 1)* ist, dass man sie im Zweifelsfall auch ohne „Renderer" lesen und verstehen kann: Würde man die Tabellenspalte mit SQL*Plus selektieren, so hätte man zwar keine Formatierungen, der Text ist aber dennoch recht einfach lesbar – und man kann sogar erkennen, was hervorgehoben sein soll.

Markdown wird auch auf populären Plattformen wie GitHub oder Discourse verwendet; mit dem Markdown Editor erstellte Texte können also auch auf diesen Plattformen verwendet werden.

## Neues „Team Development"

Der Bereich Team Development im APEX Application Builder wurde komplett neu geschrieben, stark vereinfacht und wesentlich benutzerfreundlicher gestaltet *(siehe Abbildung 14)*. Als Vorbild dienen die Issues-Berei-

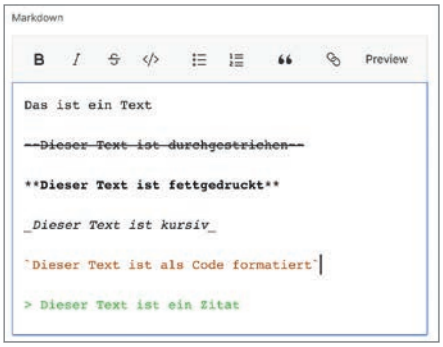



*Abbildung 13: Bearbeiten eines Textes mit dem Markdown Editor (Quelle: Carsten Czarski)*

```
Das ist ein Text.
~~Dieser Text ist durchgestrichen~~
**Dieser Text ist Fettgedruckt**
_Dieser Text ist kursiv_
'Dieser Text ist als Code formatiert'
> Dieser Text ist ein Zitat
```

*Listing 1: Markdown-Syntax*

che auf Code-Repositories wie GitHub und anderen. Alle Vorgänge, ob Bug, Features oder To-dos, werden als Issue eingetragen – „Issue" steht also weniger für ein Problem, sondern eher allgemein für „Vorgang".

Ein Vorgang wird Personen zugeordnet und mit Labels versehen, sodass man einen schnellen Überblick über offene Vorgänge mit deren Status bekommt. Für Kommentare zu einem Issue wird der bereits besprochene Markdown Editor verwendet *(siehe Abbildung 15)*.

Die Zielgruppe ist vor allem das kleinere Entwicklerteam, das kein eigenes Issue Tracking verwendet – hier kann das fertige und sofort nutzbare APEX Team Development großen Mehrwert bieten.

## Weitere neue Features

Neben den erwähnten größeren neuen Features bringt APEX, wie immer, auch viele Neuerungen im Detail mit. So unterstützt SQL Workshop Data Loading nun auch das Laden in bereits vorhandene Tabellen – mit entsprechendem Column Mapping. Das zugrunde liegende PL/SQL Package APEX_DATA_PARSER unterstützt nun auch das Laden von bis zu 20 CLOB-Spalten.

Wie mit jedem APEX-Release wurden die enthaltenen JavaScript-Bibliotheken aktualisiert. APEX 19.2 verwendet unter anderem Oracle JET 7.2.0, jQuery 3.4.1 und jQueryUI 1.12.1. Auch verwendete Open-Source-Bibliotheken, wie CKEditor, CodeMirror, FullCalendar und andere, wurden aktualisiert.

Ist APEX auf Datenbankversion 18 oder höher installiert, so werden die REST-Schnittstellen der Oracle Cloud Infrastructure (OCI), mitsamt der zugehörigen Authentifizierung, nativ unterstützt. Zum Zugriff auf den Object Store oder andere Cloud-Funktionen können Web Source Modules verwendet werden – alternativ kann man auch mit dem PL/SQL-Paket APEX_WEB_SERVICE und eigenem PL/SQL-Code arbeiten.

## Zusammenfassung

APEX 19.2 setzt den 2018 eingeführten Release-Zyklus mit zwei Releases pro Jahr konsequent fort. Wie immer gibt es „größere" neue Funktionen wie Faceted Search, die neuen Wertelisten (LOV), REST-Unterstützung für Interactive Grid oder das neue Team Development.

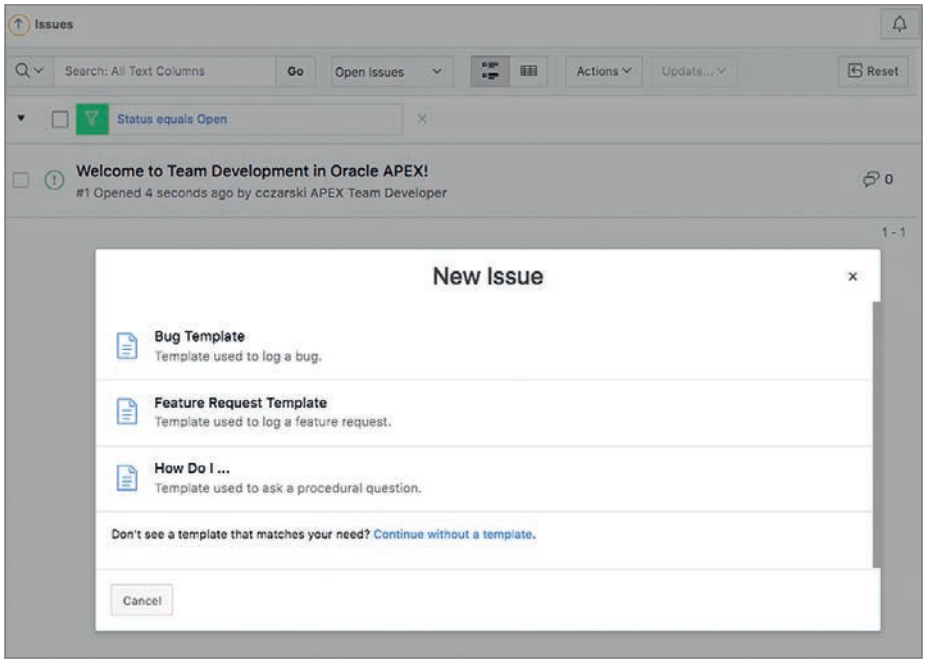



*Abbildung 14: Neuen Vorgang in APEX Team Development erfassen (Quelle: Carsten Czarski)*

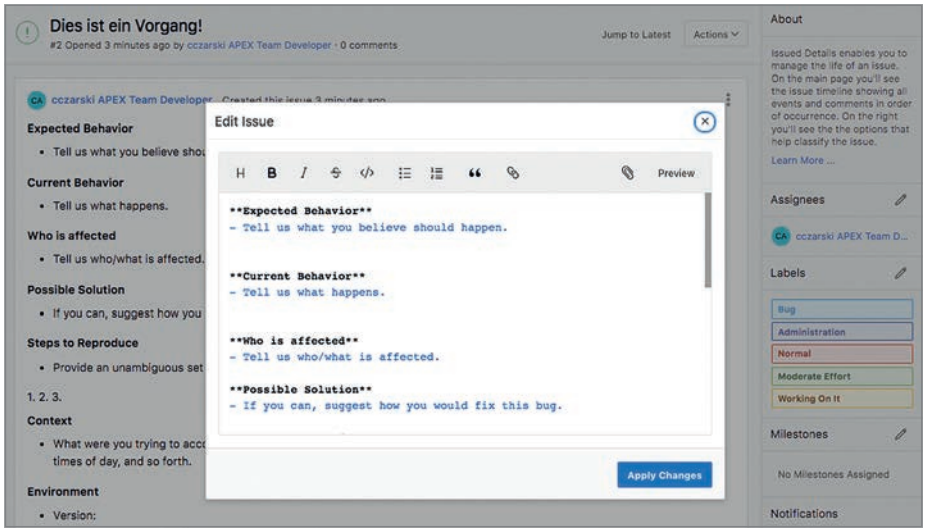



*Abbildung 15: Einen Vorgang bearbeiten mit APEX Team Development (Quelle: Carsten Czarski)*

Daneben wurden eine Reihe kleinerer Features wie die neuen Formularelemente, Unterstützung für die REST-Services der Oracle-Cloud-Infrastruktur oder zusätzliche API-Funktionen in APEX_DATA_PARSER APEX_REGION oder APEX_JSON eingeführt. Es lohnt sich, das Release auf apex.oracle.com genauer anzusehen beziehungsweise es für eine eigene Installation herunterzuladen.

## Weitere Informationen

- Informationen zu Application Express und Demo-Umgebung
  *http://apex.oracle.com*
- Oracle Application Express Blog
  *http://blogs.oracle.com/apex*

Carsten Czarski
carsten.czarski@oracle.com

# Die eigene Cloud – Wann eine On-Premises-Lösung Sinn macht

Thomas Nau, Universität Ulm, Kommunikations- und Informationszentrum (kiz)
Janne Schulz, Universität Mannheim

Die stabile, durchgehende, sichere und zeitgemäße Versorgung mit IT-Services ist für alle Hochschulen zwingend. Mit zunehmender Diversifizierung und Komplexität der IT-Versorgung wird es für viele Hochschulen jedoch unmöglich, alle notwendigen Dienste und Ressourcen in der gebotenen hohen Qualität selbst bereitstellen zu können. In vielen Fällen wird nur noch eine gemeinsame, abgestimmte Versorgung den Bedarf decken können. Zur Versorgung der Hochschulen mit hochverfügbaren IT-Services werden auch innerhalb der Hochschullandschaft bereits seit einigen Jahren in mehreren Projekten und in verschiedenen Ausrichtungen in den Einrichtungen im Land Virtualisierungstechniken eingesetzt. Eine für den nächsten Evolutionsschritt der IT-Infrastruktur an den Universitäten und Hochschulen des Landes Baden-Württemberg dennoch zu klärenden Kernfrage war, über welches Betriebsmodell und in welchem Umfang Cloud-Ressourcen für diese Einrichtungen bereitgestellt werden sollten. Dieser Artikel stellt die in diesem Projekt gefällte Entscheidung und die Gründe für die Schaffung einer eigenen Cloud-Infrastruktur vor. Er zeigt deren Komplexität auf und fasst die Entstehungsgeschichte sowie die gemachten Erfahrungen in der Hoffnung zusammen, anderen als Anregung für eigene interne Diskussionen zu dienen.

Auch wenn die Diskussionen insbesondere technischer Cloud-Themen in der Presse mittlerweile zurückgegangen sind, belegen aktuelle Zahlen, dass das Geschäft mit „der Cloud" nach wie vor ungebrochen wächst. Dabei denken die meisten Leserinnen und Leser, abgesehen von den im IT-Umfeld tätigen Personen, vermutlich in erster Linie an die Speicherung von Daten „im Internet", also in einer Storage-Cloud. Prominentester Vertreter für diese Kategorie sind die Angebote der Firma Dropbox.

Aber auch der ganzheitliche Ansatz, also die Nutzung von Cloud-Ressourcen über die reine Speicherung von Daten hinaus, beispielsweise in Form von virtuellen Maschinen, hat in den letzten Jahren für kleine und große Unternehmen stark an Bedeutung zugenommen. So beziehen immer mehr von ihnen den überwiegenden Teil ihrer IT – sowohl Server- als auch Speicherkapazitäten – in virtueller Form von großen Anbietern wie Amazon, Microsoft oder Google. Auch zahllose regionale und überregionale Anbieter offerieren entsprechende Dienstleistungen. Darüber hinaus erfreuen sich derartige Angebote auch im privaten Umfeld, aufgrund

der oft erfreulich günstigen Konditionen, zunehmender Beliebtheit.

Die Vorteile solcher Hosting-Angebote liegen auf der Hand, wobei die Argumente der „Flexibilität der Verfügbarkeit" beziehungsweise die „Skalierbarkeit" klar dominieren. Neue Speicherpools oder Server werden auf Knopfdruck und in kürzester Zeit bereit-gestellt. Gleichzeitig entfällt fast immer die Notwendigkeit, sich um die zugrunde liegende Hardwareinfrastruktur sowie deren Vernetzung und sichere Anbindung an das Internet zu kümmern. Bei sinkendem Bedarf werden Server stillgelegt und so Kosten gespart. Für größere Umgebungen stehen Lösungen bereit, die den kompletten Zyklus vom Deployment bis zur Deprovisionierung autonom steuern.

Trotz dieses verlockenden Angebotes sollten sich Verantwortliche in Rechenzentren die Frage stellen, ob der Betrieb einer eigenen Cloud-Infrastruktur, besonders bei spezifischen oder stark heterogenen Anforderungen und Anwendungsszenarien, möglicherweise eine bessere und effizientere Lösung als die Nutzung der oben genannten Angebote darstellt.

## Argumente für den Eigenbetrieb

In Deutschland wird in diesem Zusammenhang oft und gerne primär die Frage des Datenschutzes auf, nach unserer Ansicht, teilweise fragwürdigem Niveau geführt. Die Vertraulichkeit der eigenen Daten, egal welcher Ausprägung, lässt sich heute in reinen Speicher-Clouds mit Ende-zu-Ende-Verschlüsselung sehr gut gewährleisten, jedoch stellt sich die Situation bei der Verarbeitung solcher Daten auf fremd-gehosteten Systemen als technisch deutlich schwieriger dar. Daneben können nach der Erfahrung der Autoren in komplexen heterogenen Umgebungen, wie etwa an Forschungs- und Lehreinrichtungen, die Vorteile einer eigenen Infrastruktur, auch ohne besondere Berücksichtigung der Datenschutzproblematik, oft klar überwiegen. Allein der Einsatz spezieller lizenzierter Software kann einen Einsatz „in der externen Cloud" verhindern, da viele kleinere ISVs („Independent Software Vendor") ihre Lizenzbestimmungen nicht für einen solchen Einsatz überarbeitet haben. Darüber hinaus ist die Frage, inwieweit auf externe Angebote gesetzt wird, oft untrennbar damit verbunden, in welcher Tiefe lokales IT-Know-how strategische oder finanzielle Vorteile für das Unternehmen oder die Bildungseinrichtung mit sich bringt.

### Kooperationen als Motivation

Kooperationen im IT-Bereich der Hochschulen und Universitäten in Baden-Württemberg haben eine lange Tradition. Die Anfänge, bis heute stets stark gefördert und unterstützt durch das Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg (MWK *[1]*), reichen hierbei mehrere Jahrzehnte zurück. Standen zu Beginn die gemeinsame Beschaffung, insbesondere von Software, sowie der enge Erfahrungsaustausch im Vordergrund, so erlaubte der technische Fortschritt im letzten Jahrzehnt die Weiterentwicklung zu einem verteilten Betriebsmodell. Getragen und ermöglicht wurde diese Entwicklung besonders durch die folgenden Aktivitäten.

### Exzellente Vernetzung

Der kontinuierliche Ausbau des Landeshochschulnetzes BelWü *[2]*, das die Baden-Württembergischen Universitäten und Hochschulen mit Bandbreiten von bis zu 100 Gbit/s untereinander verbindet, schuf eine der zentralen Voraussetzungen, um Hardware und betriebliches Know-how, jeweils auf spezifische Anforderungen bezogen, auf wenige Standorte zu konsolidieren und gleichzeitig allen anderen beispielsweise als Service zur Verfügung zu stellen. Basierend auf einer optischen Plattform lassen sich darüber hinaus auch dedizierte Verbindungen hoher Bandbreite und geringer Latenz parallel zum Produktionsnetz schalten.

### Verteiltes Identitätsmanagement bwIDM

Die zweite Voraussetzung für die Entwicklung eines verteilten Betriebsmodells ist die Verfügbarkeit eines gemeinsamen, aber gleichzeitig verteilten Identitätsmanagements. Das auf Basis von Shibboleth *[3]* etablierte föderierte Identitätsmanagement bwIDM *[4]* belässt die Verantwortung für Autorisierung und Authentifizierung bei den jeweiligen Heimateinrichtungen. Gleichzeitig ermöglicht es aber den weit über 100.000 Nutzern des Landes die einfache Nutzung kooperativ erbrachter Dienste. Zu diesen zählen neben der bwCloud unter anderem die HPC-Cluster oder Literaturdatenbanken. Darüber hinaus ist bwIDM auch an die deutschlandweite Authentifizierungs- und Autorisierungs-Infrastruktur des DFN (DFN-AAI *[5]*) angebunden. Unter diesen Voraussetzungen erscheint die weitreichende Nutzung von Cloud-Diensten als logischer nächster Schritt der Evolution.

## Das „bwCloud"-Projekt

Der Landesdienst bwCloud entstand aus zwei nacheinander vom MWK geförderten Landesprojekten. Die primäre Aufgabe des 2015 gestarteten ersten Landesprojekts war es, die Möglichkeiten für eine standortübergreifende Server-Virtualisierung auszuloten und, wenn möglich, den Grundstein für einen stabilen und nachhaltigen Betrieb auf Basis einer prototypischen Infrastructure-as-a-Service-Implementierung (IaaS) zu legen. Durchgeführt wurde das Projekt von den Rechen- bzw. Informationszentren des KIT sowie den Universitäten Freiburg, Mannheim und Ulm. Diese haben auch den derzeitigen Produktivbetrieb übernommen.

Für deren Auswahl mussten im Projekt frühzeitig folgende grundlegende Fragen beantwortet werden:

1. Welche Vorteile lassen sich aus einer Cloud-Infrastruktur für die Bereiche Forschung und Lehre ziehen?
2. In welchem Umfang ist eine Cloud-Infrastruktur in der Lage, „herkömmliche" Virtualisierungsumgebungen in IT-Zentren von Universitäten zu ergänzen oder gar zu ersetzen? Welche Vor-, aber auch Nachteile und Aufwände sind hier zu erwarten?
3. Wie hoch ist das Potenzial für die Konsolidierung von physischen Klein- und Kleinstinstallationen in Instituten und Arbeitsgruppen in virtuelle Umgebungen unter Berücksichtigung von Self-Service-Mechanismen?
4. Welche technischen, aber auch personellen Voraussetzungen und Aufwände sind für einen sicheren Betrieb zu erfüllen?

Es wurde schnell deutlich, dass eine Beantwortung dieser Fragen, wenn überhaupt, erst später im Regelbetrieb möglich sein und damit ein gewisses Risiko einhergehen würde. Um dieses Risiko zu minimieren, wurde ein Prototyp aufgebaut, der es sowohl den Nutzern als auch den Betreibern ermöglichen würde, realistischere Antworten hinsichtlich der genannten Fragen zu finden.

## Die (Vor-) Auswahl

Wie auch für größere Unternehmen ist die Frage nach der Zahl der Standorte oft nicht nur eine finanzielle, sondern auch eine politische und strategische Frage. Auch widersprechen sich die potenziellen „besten" Antworten. Politisch etwa mag eine hohe Anzahl von Standorten gewünscht sein, jedoch wird sich dies auf die Verfügbarkeit und Kosten von qualifiziertem Personal ebenso niederschlagen wie auf das Risiko fragmentierter Betriebsteams. Letzteres lässt sich mit dem Ansatz nur eines Betriebsstandortes optimieren, jedoch auf Kosten der Redundanz und Betriebssicherheit.

### Mehrere Betriebsstandorte

Bereits nach anfänglicher Diskussion war schnell klar, dass sich nachhaltige Synergien nur dann erreichen lassen, wenn ein einheitliches Betriebsmodell entwickelt, zugrunde gelegt und von einer landesweit aufgestellten Betriebsgruppe umgesetzt wird. Der Standort der Hardware ist hierbei, insbesondere durch die hohe Leistungsfähigkeit des Landeshochschulnetzes BelWü, letztlich irrelevant, lediglich der Aufbau mehrerer Standorte zur Sicherstellung notwendiger Redundanzen waren zu beachten. Das Betriebsmodell ist jedoch so aufgebaut, dass im Laufe der Zeit bestehende Betriebsstandorte ausscheiden oder, mit gutem Grund, neue eingegliedert werden können.

### Open-Source-basierte Cloud-Lösung

Die bewusste Entscheidung, auf eine Open-Source-basierte Cloud-Lösung und nicht auf kommerzielle Virtualisierungslösungen zu setzen, diente vor allem der Beantwortung der oben genannten zweiten Frage. Dies geschah vor allem vor dem Hintergrund dort anfallender signifikanter Lizenzkosten. Damit standen als technische Grundlage, sowohl hinsichtlich der möglichen Linux-Plattform als auch bezüglich der eigentlichen „Cloud-Lösung", etliche Möglichkeiten und Produkte wie beispielsweise OpenNebula, CloudStack und OpenStack *[6]* zur Auswahl, die evaluiert und untersucht wurden. Dabei war OpenStack in Bezug auf Popularität, auch im Sinne von Hype, sicherlich ganz vorne einzuordnen. Die Entscheidung, auf OpenStack zu setzen, wurde aufgrund des bereits vorhandenen lokalen Know-hows und wegen seiner sehr aktiven Nutzer- und Entwicklergemeinde getroffen. Zu beachten ist, dass es sich bei OpenStack weniger um ein „fertiges Produkt" als eher um ein Framework *[7]* verschiedener Dienste handelt, die gemeinsam miteinander über ein Netzwerk und ein standardisiertes Kommunikationsprotokoll (API) kommunizieren. Diese Architektur geht zum einen mit erhöhtem Wartungs- und Konfigurationsaufwand einher, bietet aber zum anderen letztlich die notwendige Flexibilität, die bei der Entwicklung einer Cloud-Lösung mit mehreren Betriebsstandorten zwingend ist. Für die Versorgung der grundlegenden OpenStack-Komponenten Cinder (block-storage), Glance (image-storage) und Swift (object-storage) mit schnellem Plattenplatz

fiel die Wahl auf Ceph [8], das ebenso wie OpenStack auf CentOS [9] betrieben werden kann.

## Evaluation, erste Schritte und Ergebnisse

### Der erste Prototyp

Im Rahmen der Projektförderung wurde an den vier Betriebsstandorten x86_64-basierte Hardware zu Evaluationszwecken beschafft. Zuletzt umfasste das Testsystem, akkumuliert über die vier Betriebsstandorte, rund 400 CPU-Cores, 4 TByte an RAM, 180 TByte an Plattenspeicher sowie etliche 10- bzw. 40-Gbit-Netzwerklinks. In Bezug auf die Hardwarekomponenten (Speicher-Systeme, Netzwerkkomponenten und Hauptspeicher) wurde das Testsystem an den vier Standorten unterschiedlich ausgelegt. Dies erlaubte es, die Einflüsse der unterschiedlichen Technologien, beziehungsweise deren Ausprägung, auf Performance und Stabilität zu erfassen. Vor allem sollten die Auswirkungen des Verhältnisses von Hauptspeicher zur Anzahl von CPU-Kernen auf die virtuellen Maschinen getestet werden. Dieses Verhältnis kann ab einer gewissen Größe zu einem bestimmenden Faktor für die Systemauswahl werden.

### OpenStack-Dienste und Vernetzung

Die Kapselung der OpenStack-Dienste in eigenständige Komponenten und deren Kommunikation mit einem definierten REST-API [10] auf Basis von HTTPS schafft eine ideale Umgebung für die Automatisierung mithilfe von Ansible [11]. Eine leistungsstarke Vernetzung vorausgesetzt, ist damit auch die einfache Interaktion der OpenStack-Umgebungen an den vier Standorten möglich. Die enge Vernetzung stellt aber auch die Achillesferse dar: Die gesamte Umgebung ist vollständig von der Netzwerkinfrastruktur abhängig. Dies zieht eine deutlich komplexere Ursachenforschung im Falle von Fehlern nach sich. Auch führte es zu einem weiteren zentralen Ergebnis. Die Idee einer Multi-Site-Single-Region, also das Aufspannen einer Region über zwei oder mehr Betriebsstandorte, wurde verworfen. Zwar wäre bei diesem Betriebsmodell eine Live-Migration von virtuellen Maschinen über die Grenzen der Betriebsstandorte hinaus möglich, jedoch zeigte sich diese Betriebsart als zu anfällig für selbst kleinste Störungen.

Als Ergebnis dieser Entscheidung werden die vier OpenStack-Regionen daher nach dem Ansatz der Multi-Site-Multi-Region betrieben. Das bedeutet, sie bilden logisch jeweils eine eigene Umgebung mit einer individuellen Konfiguration hinsichtlich der Netzanbindung. Dazu gehört auch der eigenständige Betrieb einiger OpenStack-Basisdienste wie Nova (Compute), Swift (Blockspeicher) und Neutron (Netzwerk). Gemeinsam genutzt werden dagegen das Dashboard Horizon sowie die Authentifizierungskomponente Keystone und das Image-Repository Glance. Die Vorteile dieses gewählten Betriebsmodells Multi-Site-Multi-Region sind:

- ✓ Die Einbindung lokaler Infrastruktur an den Betreiberstandorten, etwa Active-Directory beziehungsweise LDAP, aber auch Fileserver, ist einfacher möglich, da in aller Regel Firewalls diese Dienste nach außen schützen.
- ✓ Verwendung von IP-Adressen aus dem lokalen Adressbereich der Betreiber. Damit ist der Zugriff auf entsprechend beschränkte lizenzierte Software aus einer VM möglich.
- ✓ Die Hardware darf zwischen den Regionen deutliche Unterschiede aufweisen.

Als Nachteile sind zu nennen:

- ✗ Eine Live-Migration von virtuellen Maschinen über die Grenzen einer Region hinweg ist nicht möglich.
- ✗ Die Regionen müssen hinsichtlich der Softwareversionen durch das Betriebsteam koordiniert und zeitnah synchronisiert gehalten werden.
- ✗ Quotas oder deren Anteile müssen bei Nutzung mehrerer Regionen manuell transferiert werden.
- ✗ Die gemeinsam benutzten kritischen Komponenten, Keystone und Horizon beziehungsweise deren Netzanbindung, bilden Single Points of Failure und müssen entsprechend redundant ausgelegt werden.

### Speicher-Pools und der Hyperconverged-Ansatz

Abwägungen bezüglich Flexibilität, Kosten, Open Source-Verfügbarkeit usw. führten schnell zu der Entscheidung für den Einsatz von Ceph im Bereich der Speicherorganisation und -anbindung. Allerdings steht den mit ausschlaggebenden Vorteilen einer kostenfreien Software, die auch auf Commodity-Hardware gut skalieren kann, eine nicht triviale Konfiguration entgegen. Deren Entwicklung war, bedingt durch die sehr steile Lernkurve, äußerst zeitintensiv. Oft musste mangels passender Dokumentation sogar auf Try-and-Error-Methoden zurückgegriffen werden. Bleibt zu erwähnen, dass es auch für Ceph sowohl kommerzielle als auch kostenfreie Alternativen gibt, die in anderen Umgebungen gegebenenfalls schneller zum Ziel führen. Ähnlich wie für OpenStack waren in der Evaluationsphase auch wegweisende Entscheidungen für die Auslegung der Speicher-Pools zu treffen. Eine wichtige Grundsatzentscheidung betrifft die Fusion von Compute- und Storage- Knoten. Für kleinere Installationen, so auch bei unserem Prototyp, bietet sich eine Hyperconverged-Lösung an. Dabei ist jeder Server mit genügend Plattenspeicher, SSDs, RAM und CPUs ausgerüstet, um sowohl die OpenStack-Compute- als auch die Storage-Aufgaben von Ceph parallel übernehmen zu können. So lässt sich mit geringem Einsatz von Hardware ein System aufbauen, in dem einzelne Szenarien erprobt werden können.

### Betriebliche Aspekte

Waren die während der Evaluierung zu treffenden technischen Entscheidungen durchaus komplex, so stellt sich die echte Herausforderung dennoch an anderer Stelle, dem täglichen Betrieb. Auf der einen Seite sprechen die zwei Mal pro Jahr erscheinenden OpenStack-Releases zwar für eine Verbesserung und Weiterentwicklung der Software, auf der anderen Seite stellen die häufigen Release-Wechsel für einen Produktivbetrieb eine (zu) hohe Herausforderung dar. Trotz eines gut etablierten und agilen DevOps-Teams.

Konnten während der Testphase Updates häufig eingespielt werden, auch um automatisierte Abläufe zu testen, so sehen wir im Produktivbetrieb aufgrund des erforderlichen Testaufwandes davon ab, „immer aktuell“ zu sein. Ausnahmen bilden selbstverständlich Releases, die Sicherheitsprobleme adressieren oder einen nutzbaren Mehrwert an Funktionalität bieten.

### Nutzermanagement

Wie bereits erwähnt, stellt das föderierte Identitätsmanagement bwIDM das Fun-

dament für die Authentifizierung, Verifizierung und Autorisierung auf Self-Service-Basis bereit. Es versetzt die jeweiligen Heimateinrichtungen in die Lage, individuell Personen für die Nutzung des Cloud-Dienstes freizuschalten oder zu sperren. Im Rahmen der Registrierung werden dann weitere Parameter, etwa Quotas, in der OpenStack-eigenen Datenbank ergänzt. Ist dies aus Nutzersicht meist ausreichend, so stellt das OpenStack fehlende ausgereifte Rollen- und Rechtemanagement, wie man es etwa aus dem Microsoft-Umfeld kennt, aus betrieblicher Sicht eine ernsthafte Einschränkung dar. Derzeit unterstützt OpenStack nur die Rollen Administrator und Nutzer. Für das langfristig ebenfalls angedachte Einsatzziel Rechenzentrumsbetrieb (Operations) stellt dies eine hohe Hürde dar, da in diesem Umfeld feingranulare, hierarchische Rechte- und Rollenkonzepte seit Langem Standard sind.

## Nutzung des Prototyps und erste Ergebnisse

Am Ende des Testzeitraumes Mitte 2018 waren gleichzeitig rund 1.000 virtuelle Maschinen aktiv, die von mehreren Hundert registrierten Nutzern aus mehr als 20 Hochschulen und Universitäten des Landes betrieben wurden *(siehe Abbildung 1)*. Wir gehen davon aus, dass in dieser Phase bereits einige Produktionsumgebungen in den Prototyp ausgelagert wurden.

In der für das Gesamtsystem kritischen Netzwerk- und Speicher-Umgebung ließen sich mit dem heterogenen Ansatz deutliche Unterschiede in der Performance nachweisen. So erlaubt beispielsweise das „Hardware Offloading" von VXLAN *[12]* -Operationen an die Netzwerkkarte ebenso deutlich höhere Bandbreiten wie die Nutzung der „Embedded Switches", die einige NICs (Network Interface Card) mitbringen. Ähnliches konnte auch bei den unterschiedlichen Techniken zum Caching in Ceph beobachtet werden.

Darüber hinaus lieferte die Evaluierungsphase jedoch auch unerwartete Fakten. Die Auswertung der für die Planung des Produktivsystems wichtigen Kennzahl, der Verteilung der virtuellen Maschinen hinsichtlich Speicher und CPU, war eine Überraschung: Über 80% der VMs forderten lediglich maximal 4 GByte Hauptspeicher und maximal 2 virtuelle CPUs (vCPU *[13]*) an. Knapp 30% kamen sogar mit nur einer vCPU und 1 GByte beziehungsweise 512 MByte an Hauptspeicher aus. Obwohl unsere Schätzungen deutlich über diesen Werten lagen, konnten diese aufgrund der großen Datenbasis dennoch als signifikant angesehen werden. Für die langfristige Planung hat sich der generische Ansatz, die Trennung von Speicher- und Computer-Hardware, als zielführender herausgestellt. Damit ergeben sich getrennte Investitionspfade, die auch eine gezielte und unabhängige Erweiterung einzelner Funktionsgruppen erlauben. Auch gestattet es diese Struktur, die Hardware für den jeweiligen Einsatzzweck zu optimieren. Erkauft werden diese Vorteile durch eine größere Netzwerkkomplexität und höhere Anfangsinvestitionen.

## Übergang in den Produktivbetrieb

Der 2018 unter dem Akronym bwCloud SCOPE gestartete Produktivbetrieb setzt die in der Evaluierungsphase gewonnenen Kenntnisse und entwickelten Best-Practices konsequent auf neuer Hardware mit einem einheitlichen Betriebsmodell um. Mit mehr als 1.800 CPU-Kernen und 30 Terabyte RAM, über 2,7 Petabyte an Speicherkapazität und einer 40/100-Gbit/s-Anbindung an das Landeshochschulnetz ist diese Evolutionsstufe in der Lage, mehrere Tausend virtuelle Maschinen in einer Viel-

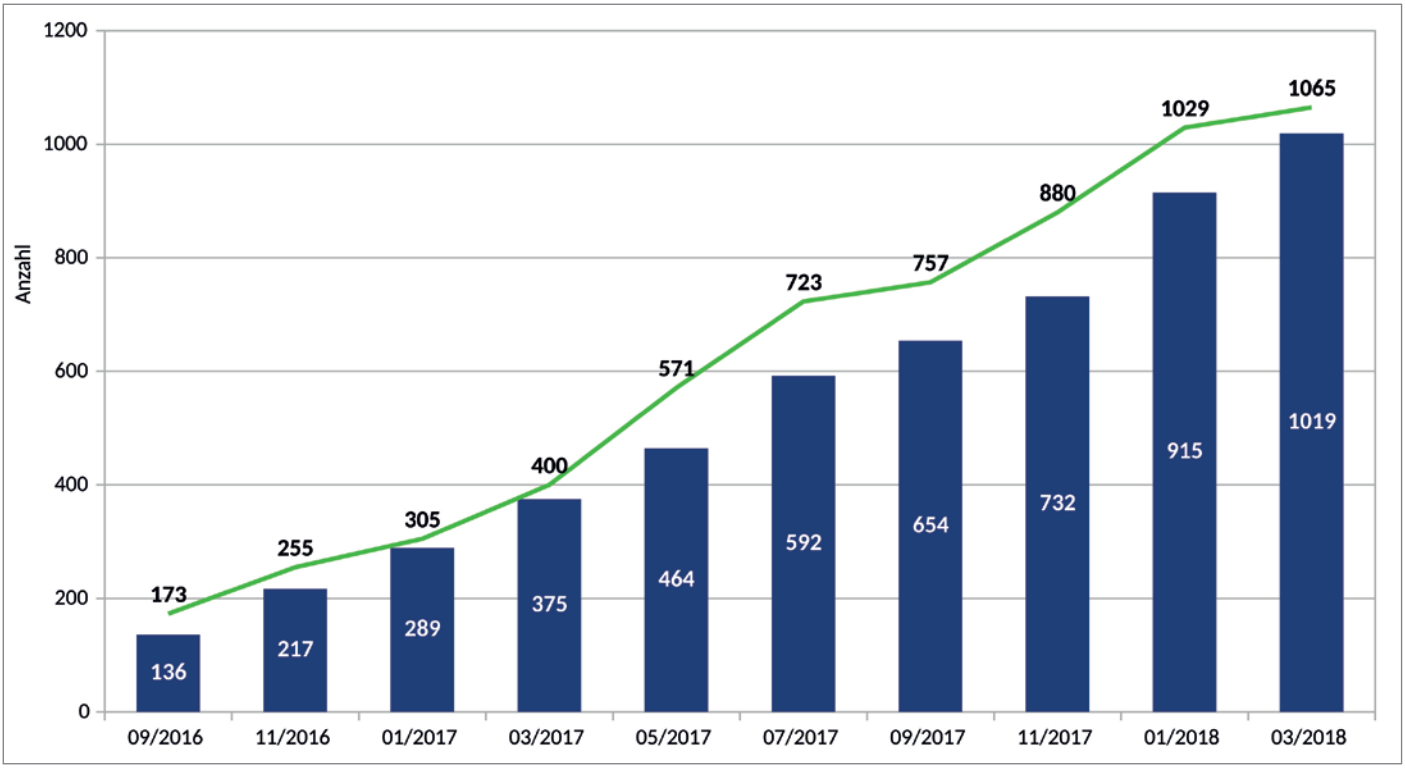



*Abbildung 1: Die grüne Kurve gibt die Zahl der VMs wieder, während die blauen Balken die Zahl der registrierten Nutzer aus den aktiven 25 Hochschulen und Universitäten darstellen. Stand der Zahlen: März 2018, (Quelle: bwCloud)*

zahl von Ausprägungen zur Verfügung zu stellen. Das Akronym SCOPE spiegelt dabei die in der vorherigen Testphase identifizierten Nutzungsszenarien und ihre unterschiedlichen Anforderungen wider: **Sc**ience, **Op**erations and **E**ducation.

Die in den letzten Jahren gewonnene Expertise lässt uns für alle drei Nutzergruppen wesentliche monetäre, personelle und organisatorische Verbesserungen erwarten. An erster Stelle steht natürlich die Versorgung der Forscherinnen und Forscher (Science) mit Ressourcen für intensive Berechnungen. Dazu können virtuelle Maschinen mit sehr viel Ressourcenumfang und in großer Anzahl erzeugt und betrieben werden. Sollten spezifische Anforderungen an die Hardware gestellt werden, beispielsweise die Anbindung spezieller Grafikkarten, so kann dies mit der bwCloud-Betriebsgruppe gemeinsam organisiert werden. Die Konsolidierung von „Individualrechenzentren" erlaubt eine effizientere Nutzung der Hardware-Investitionen und insbesondere auch effizientere Energieversorgungs- und Kühlkonzepte. Zu guter Letzt steigt auch insgesamt die Planungssicherheit durch die Verfügbarkeit entsprechender Metriken.

Gänzlich neue Möglichkeiten entstehen im Bereich der Lehre (Education). So lassen sich beispielsweise zur Unterstützung von Vorlesungen in kürzester Zeit virtuelle IT-Landschaften ausrollen, die dann von Studenten begleitend nicht nur genutzt, sondern aktiv angepasst werden können. Neben der Entwicklung neuer Lehr- und Lernszenarien bietet die bwCloud eine Grundversorgung der Studierenden mit jeweils einer virtuellen Maschine an. Diese Grundversorgung wird durch das MWK ermöglicht und soll ihnen bei der erfolgreichen Bewältigung des Studiums helfen. Die virtuellen Maschinen können als Software-Repositorium oder als Umgebung zur Entwicklung eigener Projekte dienen.

Schwieriger zu adressieren ist der Bereich der Rechenzentren (Operations), wenngleich dieser seit Langem das Potenzial der Virtualisierung nutzt. Primäre Motivation in diesem Bereich ist aber bislang weniger das Bestreben, eine echte DevOps-Kultur zu etablieren, als vielmehr die Kostenersparnis für Hardware und Energie beziehungsweise Kühlung. Die bwCloud punktet hier vor allem mit der Reduktion der oft erheblichen Lizenzkosten vorhandener Virtualisierungsprodukte und kann je nach Anforderung und Einsatzzweck eine günstige und effiziente Ergänzung zu den heute etablierten Virtualisierungsumgebungen darstellen.

## Fazit

Zusammenfassend kann nach unserer Erfahrung gesagt werden, dass besonders in Umgebungen mit komplexen Anforderungen der Betrieb einer eigenen Cloud-Infrastruktur durchaus seine Berechtigung hat. Bestätigt wird dies sowohl durch die sehr hohe Akzeptanz seitens der Nutzer als auch durch die Bereitstellung der notwendigen Investitionssummen an den Betriebsstandorten sowie die weitere Förderung von Seiten des MWK. Die freie Architektur für Cloud-Computing OpenStack und die verteilte Storage-Lösung Ceph sind eine stabile Basis für einen produktiven Betrieb, erfordern aber erhebliche „Startinvestitionen", da beide eher einen Baukasten als eine fertige Lösung darstellen. Daher ist die Verfügbarkeit von geschultem Personal mit dem notwendigen Hintergrundwissen eine zwingende Voraussetzung für den Erfolg dieses oder vergleichbarer Projekte. Dies gilt umso mehr, da auch zukünftige Upgrades sicherlich eine Herausforderung darstellen werden und fehlende betriebsrelevante Funktionen weiterhin selbst implementiert werden müssen.

Bewährt hat sich der Start mit einer einfachen Umgebung, in der Fehler gemacht werden können, um daraus zu lernen. Nur so ist das Team später in der Lage, die richtigen Fragen zu stellen oder auftretende Probleme zeitnah zu lösen. Der schrittweise Weg von einem Prototyp mit dennoch vielfältigen Nutzungsszenarien hin zu einer großen Umgebung mit vielen Nutzern ist bei der Realisierung komplexer verteilter Systeme die einzig sinnvolle Variante. Nur so lassen sich sowohl eigene Betriebskenntnisse aufbauen als auch der Service kontinuierlich an die Anforderungen der Nutzer anpassen. Für die kommenden Jahre sehen wir die bwCloud als wesentlichen Bestandteil, um sehr schnell auf sich ständig ändernde und neue Anforderungen im SCOPE-Umfeld reagieren zu können.

## Über die Autoren

Thomas Nau ist seit mehr als 30 Jahren im IT-Umfeld tätig und leitet derzeit die „Abteilung Infrastruktur" des Kommunikations- und Informationszentrums (kiz) an der Universität Ulm. Das kiz, dessen stellvertretender Leiter der Autor ebenfalls ist, trägt unter anderem die Gesamtverantwortung für die universitäre IT-Infrastruktur.

Janne Schulz ist derzeit der Projektleiter des Landesprojektes „bwCloud SCOPE" und hat die bwCloud von Anfang an begleitet und geprägt. Zuvor war er als Assistent der Geschäftsleitung am Rechenzentrum der Uni Freiburg maßgeblich unter anderem an der Ausarbeitung des Förderantrages für das HPC-Forschungsgroßgerät „bwForCluster Nemo" beteiligt.

## Quellen


[1] *https://mwk.baden-wuerttemberg.de*
[2] *https://www.belwue.de*
[3] *https://de.wikipedia.org/wiki/Shibboleth_(Internet)*
[4] *https://www.bwidm.de/*
[5] *https://www.aai.dfn.de/der-dienst/*
[6] *https://www.openstack.org/*
[7] Kritiker bezeichnen OpenStack gelegentlich auch als „Python-Skript mit Ambitionen".
[8] *https://ceph.com/*
[9] *https://www.centos.org/*
[10] *https://de.wikipedia.org/wiki/Representational_State_Transfer*
[11] *https://www.ansible.com/*
[12] *https://de.wikipedia.org/wiki/Virtual_Extensible_LAN*
[13] Sechzehn vCPU entsprechen einem physikalischen CPU-Kern



Dipl. Inf. Janne Chr. Schulz
janne.schulz@rz.uni-mannheim.de



Dipl. Phys. Thomas Nau
thomas.nau@uni-ulm.de


# Aber das hat gestern noch funktioniert! Testing mit utPLSQL

Samuel Nitsche, Smart Enterprise Solutions

Automatisierte Selbst-Tests sind inzwischen aus der Applikations- und Webentwicklung nicht mehr wegzudenken, dennoch werden sie in der Datenbankwelt nur zögerlich eingesetzt. Dieser Artikel beschreibt, wie einfach Sie mit utPLSQL Version 3 *(siehe unter https://utplsql.org)* ins Entwickeln von Selbst-Tests einsteigen können und welche Vorteile dies für Ihr Projekt und Ihre Software hat.

### „Aber das hat gestern noch funktioniert!"

Dieser Ausspruch dürfte den meisten Entwickler*innen und wahrscheinlich auch einigen Benutzer*innen bekannt vorkommen. Was ist nur über Nacht passiert, dass mein Interface plötzlich nicht mehr funktioniert und dies natürlich direkt vom Kunden gemeldet wird?

Nach kurzem Nachdenken erinnern wir uns daran, dass wir die View `v_star-wars_characters` gestern kurz vor Feierabend – auf Kundenwunsch hin – noch um eine neue Spalte erweitert haben. Nichts Besonderes, aber ganz dringend: lediglich ein weiteres Feld, das Informationen darüber enthält, in welchen Filmen die jeweilige Person vorkommt.

| ID | NAME | EPISODES (neu) |
|---|---|---|
| 1 | Darth Vader | 3,4,5,6 |
| 2 | Luke Skywalker | 4,5,6,7,8,9 |
| 3 | Rey | 7,8,9 |

Ein einfaches: `create or replace view`. Anschließend haben wir natürlich per `select` getestet, ob das neue Feld auch korrekt geliefert wird. Doch wenn nun versucht wird, über die Applikation den Namen einer Person zu ändern, meldet die Datenbank den folgenden Fehler *(siehe Listing 1)*.

Vielleicht haben Sie es schon erraten, vielleicht sind Sie selbst schon über diese Eigenschaft der Oracle-Datenbank gestolpert: Beim Ersetzen einer View geht der zugehörige Instead-Of-Trigger verloren. Eine kleine, simple Änderung mit großen und ärgerlichen Auswirkungen.

### Wie hätte man diesen Fehler verhindern können?

„Besser testen" ist hier natürlich die naheliegende Antwort und relativ schnell werden wir bei Checklisten und standardisierten Testprotokollen landen, mit denen sicher-

gestellt werden kann, dass eben wirklich alles Entscheidende geprüft wird. Nach jeder Änderung. An dieser Stelle werden automatisierte Selbst-Tests interessant, denn sie sind genau das: eine Reihe von exakt definierten, standardisierten „Checks" einer bestimmten Funktionalität – zum Beispiel, ob ein Update auf eine View möglich ist.

Einen solchen Test können wir mit ganz normalen PL/SQL-Bordmitteln umsetzen *(siehe Listing 2)*. Diesen Block könnten wir jetzt als Skript speichern und nach jedem Update ausführen. Allerdings wird dies mit der Zeit relativ unübersichtlich, weshalb wir stattdessen das freie Open Source Framework utPLSQL benutzen, das uns einiges an Arbeit abnimmt.

## utPLSQL installieren

Um utPLSQL nutzen zu können, muss das Framework zunächst in der Datenbank installiert werden.

Dazu laden Sie am besten die aktuelle Version von GitHub herunter *(siehe unter https://github.com/utPLSQL/utPLSQL/releases/latest)*. Im Ordner „`source`" finden sich vorbereitete Skripte, mit denen sich utPLSQL problemlos installieren lässt:

- „`install_headless.sql`", um eine Standardinstallation ins Schema „ut3" mit Public Synonymen durchzuführen (erfordert SYS-user)
- „install.sql", um utPLSQL in ein anderes Schema zu installieren. In diesem Fall muss zusätzlich die Nutzung der utPLSQL-Methoden erlaubt werden:
  - Mittels „`create_synonyms_and_grants_for_public.sql`„ für alle Benutzer*innen
  - Mittels "`create_user_grants.sql`" und "`create_user_synonyms.sql`" für eine bestimmte Benutzer*in

Eine detaillierte Installationsanleitung finden Sie unter *http://utplsql.org/utPLSQL/latest/userguide/install.html* oder im Verzeichnis *docs/userguide/install.html* der heruntergeladenen ZIP-Archivs.

## Ein erster Test mit utPLSQL

Während die meisten PL/SQL-Entwickler eher die prozedurale Programmierung gewohnt sind und stark auf Packages setzen, benutzt utPLSQL die objektorientierten Funktionen der Oracle-Datenbank und bietet damit eine „fluent" API an. Das mag im ersten Moment fremd erscheinen, doch Sie werden sich schnell daran gewöhnen und diesen Ansatz eventuell schätzen lernen.

utPLSQL bietet uns nach der Installation einige sehr hilfreiche Public-Methoden an, allen voran die sogenannten „Expectations" *(siehe Abbildung 1)*.

Das zweite Kernstück des Frameworks sind sogenannte „Annotations", mithilfe derer wir ganz normale PL/SQL Packages in Test-Suites umwandeln können *(siehe Listing 3)*.

`%suite` ist dabei die einzige Annotation, die absolut erforderlich ist. Sie signalisiert utPLSQL, dass es sich bei dem Package um eine Test-Suite handelt. Mit %test markieren wir die nachfolgende Prozedur als Test, den wir nun wie gewohnt im Package-Body implementieren können *(siehe Listing 4)*.

Genau wie im reinen PL/SQL-Beispiel führen wir ein Update auf die View aus. Anschließend selektieren wir das aktualisierte Feld und vergleichen das Ergebnis mit dem erwarteten Wert mittels utPLSQL-Expectation.

Da wir in einem Testszenario meist nicht genau wissen, ob und welche Daten in einer Tabelle existieren, stellen wir zunächst sicher, dass wir auf jeden Fall Daten haben, die wir aktualisieren können. Hier machen wir uns die Tatsache zunutze, dass der Primärschlüssel der meisten Tabellen ein INTEGER-Wert ist, die zugehörige Sequence oder Identity aber in den meisten Fällen bei 1 beginnt. Wir können in diesem Fall negative IDs für unsere Testdaten benutzen, ohne dass diese mit eventuell existierenden Einträgen kollidieren.

Nun können wir den Test mit der ut.run-Methode durchführen *(siehe Listing 5)*.

Wie erwartet, schlägt auch dieser Test fehl und utPLSQL gibt uns gleich den kompletten Stacktrace.

```
„ORA-01733: Virtuelle Spalte hier nicht zulässig"
```

*Listing 1 - Fehlermeldung*

```
declare
   l_name varchar2(2000);
begin
   update v_starwars_characters set name = 'Anakin Skywalker' where id
= 1;
   select name into l_name
       from v_starwars_characters where id = 1;

   if ( l_name <> 'Anakin Skywalker') then
       raise_application_error(-20000, 'Update did not work!');
   end if;
end;
```

*Listing 2: Einfacher Test mit PL/SQL-Bordmitteln*

```
ut.expect(actualValue)
    .to_equal(expectedValue)
    .to_be_greater_than(value)
    .to_be_between(min, max)
    .to_be_like('%partialString%')
    .not_to_be_null()
    .not_to_be_less_than(value)
    …
```

*Abbildung 1: utPLSQL Expectations (Quelle © Samuel Nitsche). Für PL/SQL ist die objektorientierte, „fluent"-Syntax etwas ungewohnt.*

```
create or replace package ut_v_starwars_characters as
  -- %suite(View: V_STARWARS_CHARACTERS)

  -- %test(Update character-name via view)
  procedure update_name;
end;
```

*Listing 3: utPLSQL Test-Suite Header mit Annotations %suite und %test – diese werden vom Framework geparst und interpretiert, allerdings nur in Package-Headern*

```
create or replace package body ut_v_starwars_characters as
  procedure update_name
  as
      l_actual_name v_starwars_characters.name%type;
  begin
      -- Arrange: Setup test-data
      insert into star_wars_characters (id, name) values (-1, 'Test-Char');

      -- Act: Do the actual update
      update v_starwars_characters set name = 'Darth utPLSQL' where id = -1;

      -- Assert: Check the output
      select name into l_actual_name from v_starwars_characters where id = -1;
      ut.expect(l_actual_name).to_equal('Darth utPLSQL');
  end;
end;
```

*Listing 4: Implementierung des ersten utPLSQL-Tests*

```
set serveroutput on
call ut.run('ut_v_starwars_characters');
View: V_STARWARS_CHARACTERS
  Update character-name via view [,002 sec] (FAILED - 1)

Failures:

  1) update_name
      ORA-01732: Datenmanipulationsoperation auf dieser View nicht zulässig
      ORA-06512: in "SITHDB.UT_V_STARWARS_CHARACTERS", Zeile 10
      ORA-06512: in "SITHDB.UT_V_STARWARS_CHARACTERS", Zeile 10
      ORA-06512: in Zeile 6
Finished in ,002346 seconds
1 tests, 0 failed, 1 errored, 0 disabled, 0 warning(s)
```

*Listing 5: Ergebnis des Unit-Tests*

```
create or replace trigger save_v_starwars_characters
  instead of update on v_starwars_characters
  for each row
  begin
      null;
  end;
```

*Listing 6: Instead-Of-Trigger ohne Funktionalität*

Interessant ist auch, dass wir in der abschließenden Zusammenfassung keinen „failed"-, sondern einen „errored"-Test angezeigt bekommen. Dieser Test hat eine ORA-Exception verursacht, die das Framework für uns abgefangen und dokumentiert hat. Hätten wir noch weitere Tests, würden diese trotzdem durchgeführt werden.

Nun können wir uns um das eigentliche Problem kümmern und den verloren gegangenen Instead-Of-Trigger wieder einspielen (zu Demonstrationszwecken zunächst ohne jede Funktionalität) *(siehe Listing 6)*.

Führen wir nun unsere Test-Suite abermals durch, erhalten wir ein etwas anderes Ergebnis *(siehe Listing 7)*.

Der Test schlägt noch immer fehl, aber dieses Mal ist es tatsächlich unsere Expectation, die den Fehler verursacht, und kein ORA-Fehler. Die Fehlerausgabe sagt uns auch sehr genau, was falsch lief und in welcher Zeile unseres Tests das Problem aufgetreten ist.

Nun implementieren wir den Trigger vollständig und führen anschließend wiederum unsere Test-Suite durch:

Unsere View funktioniert wieder wie erwartet – und das nächste Mal, wenn der Trigger verloren geht, werden wir es merken.

## Vorteile automatisierter Tests

Da Sie diesen Artikel lesen, ist die Wahrscheinlichkeit relativ hoch, dass Sie nicht erst vom Nutzen automatisierter Selbst-Tests überzeugt werden müssen. Dennoch möchte ich kurz einige Vorteile auflisten, die automatisierte Tests in verschiedener Hinsicht bieten.

- Automatisierte Tests sind „change detectors", die uns bei Änderung bestehender Funktionalität warnen (unabhängig davon, ob diese Änderung gewollt ist oder – wie in unserem Beispiel – ungewollt)
- Sie sind transportabel und beliebig oft (nahezu kostenlos) auf unterschiedlichen Systemen wiederholbar
- Falls sie implementiert wurden, um Bugfixes zu bestätigen, schließen sie einmal aufgetretene Fehler aus
- Sie können als Bestätigung dienen, dass vereinbarte Anforderungen an die Software erfüllt sind

Diese Vorteile gelten für alle automatisierten Tests gleichermaßen. Wenn wir wie in diesem Fall von Tests ausgehen, die von den Entwickler*innen selbst im Zuge des Entwicklungsprozesses geschrieben werden, kommen noch einige weitere Vorteile dazu:

- Die Erstellung von Tests kann dabei helfen, den Fokus vom **Wie** auf das **Was** zu verlagern und eine Funktionalität aus unterschiedlichen Perspektiven zu betrachten
- Gut und verständlich geschriebene Tests können als Code-Beispiel und

Anders als bei sogenannten „fail-fast"-Frameworks wertet utPLSQL alle Expectations eines Tests aus und bricht nicht sofort nach dem ersten Fehlschlag den Test ab. Alle Fehlschläge werden gesammelt in das Testergebnis aufgenommen und untereinander dargestellt.

Dokumentation dafür dienen, wie eine Funktionalität zu verwenden ist
- Selbst-Tests regen dazu an, „einfachere" Programmierkonstrukte zu verwenden, da sich diese leichter testen lassen – was wiederum positive Auswirkungen auf die Wartbarkeit des Codes hat

Der aus meiner Sicht jedoch wichtigste Vorteil, den eine solide, automatisierte Testbasis bietet, ist, dass sie die Voraussetzung für die Entwickler*innen schafft, stetig und selbstbewusst den eigenen Quellcode zu verbessern, also ständiges Refactoring zu betreiben.

```
call ut.run();
View: V_STARWARS_CHARACTERS
   Update character-name via view [,409 sec] (FAILED - 1)

Failures:

   1) update_name
      Actual: 'Test-Char' (varchar2) was expected to equal: 'Darth ut-
PLSQL' (varchar2)
      at "SITHDB.UT_V_STARWARS_CHARACTERS.UPDATE_NAME", line 14 ut.ex-
pect(l_actual_name).to_equal('Darth utPLSQL');


Finished in ,41099 seconds
1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)
```

*Listing 7: Ergebnis des Unit-Tests mit „failed" statt „errored"*

```
create or replace trigger save_v_starwars_characters
   instead of update on v_starwars_characters
   for each row
   begin
       update star_wars_characters
           set name = :new.name
           where id = :new.id;
   end;
/

call ut.run();
View: V_STARWARS_CHARACTERS
   Update character-name via view [,004 sec]

Finished in ,005951 seconds
1 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)
```

*Listing 8: Korrektur des Triggers und erfolgreicher Test*

```
create or replace package ut_v_starwars_characters as
   -- %suite(View: V_STARWARS_CHARACTERS)

   -- %beforeall
   procedure setup_test_data;

   -- %test(Update character-name via view)
   procedure update_name;

   -- %test(View returns correct list of episodes)
   procedure return_list_of_episodes;
end;
```

*Listing 9: Package-Header mit neuem Test und weiteren Annotations*

## Ein weiterer Test und mehr Annotations

Eine Funktionalität unserer View haben wir abgesichert, doch gerade die neue Spalte enthält ein Stück Logik, bei dem es sich durchaus lohnt, das Verhalten durch einen automatisierten Test abzusichern (insbesondere dann, wenn wir davon ausgehen, dass wir die Funktionalität in Zukunft vielleicht noch mehrfach ändern oder erweitern werden).

Wir definieren also einen weiteren Test und nutzen gleich noch eine weitere utPLSQL-Annotation:

Die %beforeall-Annotation sorgt dafür, dass diese Prozedur einmalig pro Test-Suite ausgeführt wird, und zwar vor allen Tests.

Der Nutzen wird klarer, wenn wir uns die Implementierung ansehen:

`setup_test_data` stellt nun die Situation her, die wir für die Ausführung beider Tests benötigen: Ein Eintrag in der Tabelle `star_wars_characters` sowie mehrere Einträge in der Tabelle `appearance_in_episode`. Diese beiden Tabellen bilden die Grundlage der View, die wir testen möchten.

Die Tests selbst sind nun sehr einfach zu lesen und zu verstehen – zur Verbesserung der Lesbarkeit haben wir zusätzlich noch die Hilfsfunktion `get_view_row()` hinzugefügt, die einfach die komplette View-Zeile zurückgibt. Für den Check verwenden wir wiederum die `to_equal`-Expectation.

Vielleicht fragen Sie sich schon die ganze Zeit, was mit den ganzen Testdaten passiert, die wir anlegen?

utPLSQL arbeitet im Standard-Modus mit Savepoints und Rollbacks. Vor jeder Suite sowie vor jedem Test wird jeweils ein Savepoint gesetzt, zu dem nach Abschluss des Tests beziehungsweise der Suite zurückgerollt wird. Das bedeutet, dass die Testdaten, die wir anlegen, sowie alle anderen Änderungen an Daten am Ende des Tests wieder verschwunden sind.

Das bedeutet natürlich auch, dass wir in diesem Modus nur Funktionen testen können, die keine Transaktionskontrolle wie

`commit` und `rollback` oder DDL enthalten. Auch diese Szenarien können natürlich mit utPLSQL-Tests abgesichert werden. In diesem Fall fügt man unterhalb von `%suite` die Annotation `%rollback(manual)` ein, muss sich dann aber selbst darum kümmern, dass Änderungen aufgeräumt werden (beispielsweise mittels `%afterall`).

Für alle Situationen, in denen keine Transaktionskontrolle notwendig ist, bietet der Rollback-Mechanismus von utPLSQL hingegen eine große Erleichterung.

## Absichern von Sonderfällen

Was wir bisher getestet und abgesichert haben sind die offensichtlichen Dinge, die wir erwarten. Was aber passiert, wenn wir einen Star-Wars-Charakter haben, der in keinem der Filme vorkommt, beispielsweise die beliebte Ahsoka Tano aus der „The Clone Wars"-Fernsehserie?

Es ist wichtig, über die offensichtlichen Use-Cases hinaus zu denken, über das hinaus zu denken, was wir von den Benutzer*innen erwarten oder möchten. Die Wahrscheinlichkeit ist sehr hoch, dass Benutzer*innen unsere Applikation auf eine Art und Weise nutzen, die wir normalerweise nicht erwarten würden.

Lassen Sie uns also einen weiteren Test für diesen Fall schreiben *(siehe Listing 11)*.

Wenn dieser Test erfolgreich durchgeführt wird, haben wir damit zwei Dinge auf einmal bewiesen:

- Auch wenn ein Charakter nicht in den Filmen vorkommt, liefert die View eine Zeile zurück (ansonsten würde eine `NO_DATA_FOUND`-Exception ausgelöst)
- Der Wert in der Spalte `EPISODES` ist in diesem Fall `NULL`.

utPLSQL bietet eine ganze Reihe von Annotations, die es ermöglichen, setup und cleanup vom eigentlichen Test zu entkoppeln:

- %beforeall
- %beforeeach
- %beforetest
- %aftertest
- %aftereach
- %afterall

```
create or replace package body ut_v_starwars_characters as

   function get_view_row
       return v_starwars_characters%rowtype
   as
       l_result v_starwars_characters%rowtype;
   begin
       select * into l_result
           from v_starwars_characters
           where id = -1;
       return l_result;
   end;

   procedure setup_test_data
   as
   begin
       insert into star_wars_characters (id, name) values (-1, 'Test-
Char');
       insert into appearance_in_episode (character_fk, episode_no)
           values ( -1, 3 );
       insert into appearance_in_episode (character_fk, episode_no)
           values ( -1, 5 );
   end;

   procedure update_name
   as
   begin
       update v_starwars_characters set name = 'Darth utPLSQL' where id
= -1;

       ut.expect(get_view_row().name)
           .to_equal('Darth utPLSQL');
   end;

   procedure return_list_of_episodes
   as
   begin
       ut.expect(get_view_row().episodes)
           .to_equal('3,5');
   end;
end;
```

*Listing 10: Implementierung der erweiterten Test-Suite*

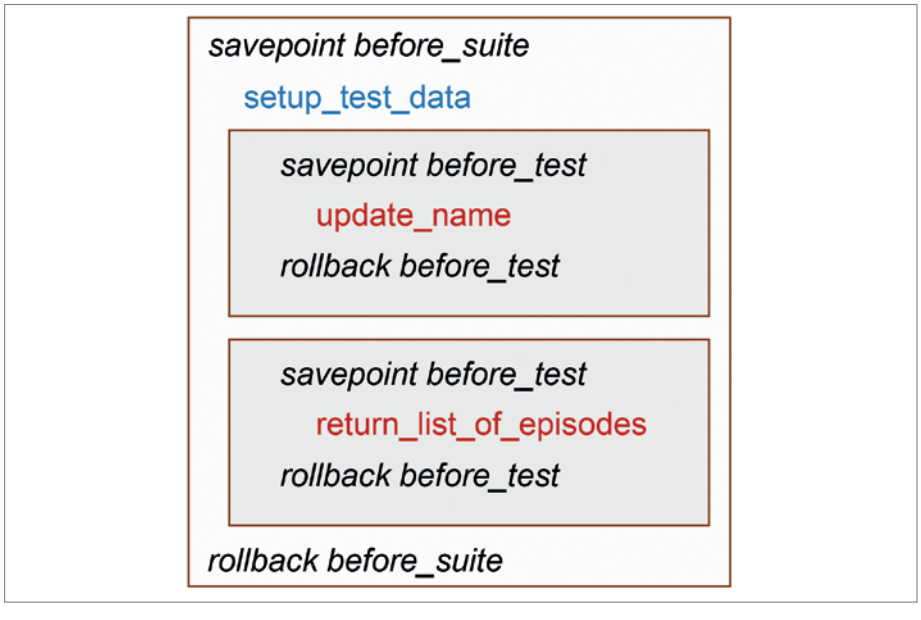



*Abbildung 2: Reihenfolge und Savepoints der Test-Suite (Quelle © Samuel Nitsche)*

```
-- Package-Header
…
-- %test(View returns row but empty list of episodes when character has
no appearance)
procedure return_empty_list_of_episodes;
…
-- Package-Body
…
procedure return_empty_list_of_episodes
as
begin
   delete from appearance_in_episode where character_fk = -1;

   ut.expect(get_view_row().episodes)
       .to_be_null();
end;
…
```

*Listing 11: Test, um das Verhalten der View bei Nicht-Erwähnung in den Filmen abzusichern*

Wir könnten nun auch noch sicherstellen, dass die View-Zeile immer noch den erwarteten Namen zurückgibt und nicht etwa `NULL`. Hier zeigt sich aber auch ein Dilemma beim Schreiben automatisierter Selbst-Tests.

## Wie viele Tests sind genug?

Wie weit sollten meine Tests gehen? Sollte ich jede noch so kleine Eventualität mit einem Test absichern, in unserem Fall zum Beispiel überprüfen, dass Insert nicht erlaubt ist oder dass der Name nicht auf einen Wert geändert werden kann, der bereits existiert?

Diese Frage werden nur Sie beantworten können, denn nur Sie kennen die Umstände und Risiken Ihres Projekts und Ihrer Datenbankapplikation. Hilfreich bei der Entscheidung können die folgenden Fragen sein:

- Wie gravierend sind die Folgen, wenn sich eine bestimmte Funktionalität nicht verhält wie erwartet?
- Wie wahrscheinlich ist es, dass ein bestimmter Fall auftritt (z.B., dass ein Tabellen-Constraint versehentlich gelöscht wird und damit Mehrfach-Namen möglich werden)?
- Wie wahrscheinlich ist es, dass sich der Code dieser Funktionalität in Zukunft ändern wird und wie häufig wird dies der Fall sein?
- Wie schwierig beziehungsweise aufwendig ist es, die Funktionalität automatisiert zu testen?

Software-Entwicklung ist oftmals das Ausbalancieren von Kompromissen und genauso verhält es sich auch mit der Entwicklung automatisierter Tests. Ob und in welchem Umfang diese sinnvoll sind, hängt sehr stark von Ihren Zielen, Ihrem Entwicklungsprozess, Ihren Unternehmens- und Projektumständen ab.

Schon ein paar wenige Tests, die wichtiges bestehendes Verhalten auf einem relativ hohen Level absichern, können ungemein hilfreich sein. Wenn Sie hingegen eine Software entwickeln, die über Jahre gewartet, erweitert und verbessert werden soll, wird eine detailliertere Basis von Unit-Tests, die das ständige Refactoring des Codes ermöglichen, einen echten Mehrwert bringen und Ihre Entwicklungsgeschwindigkeit und -qualität drastisch erhöhen.

Mein Tipp: Fangen Sie klein an und probieren Sie aus. Jedes Mal, wenn ein Fehler auftritt oder gemeldet wird, müssen Sie ohnehin analysieren und versuchen, den Fehler nachzustellen. Dies können Sie in der Regel auch so tun, dass Sie die entsprechenden Voraussetzungen in Form eines Test-Setups festlegen. Wenn der Fehler dann gefunden und behoben ist, haben Sie gleich einen Test, der sicherstellt, dass dieser Fehler in Zukunft nicht wieder auftritt.

Aller Anfang ist schwer, lassen Sie sich nicht davon entmutigen. Experimentieren Sie und beobachten Sie, was Ihnen hilft. Es sind oft nicht die großen, gewaltigen Schritte, die nachhaltige Änderung bringen, sondern die kleinen, stetigen, die ein Teil des Alltags werden.

## Weitere Informationen und Tools

Viele Informationen rund um utPLSQL finden Sie im „Resources"-Bereich von utplsql.org *(siehe unter https://utplsql.org/resources)*.

Mittlerweile gibt es auch eine umfangreiche Sammlung von Tools rund um utPLSQL:

- SQLDeveloper Plug-in:
  *https://www.salvis.com/blog/2019/07/06/running-utplsql-tests-in-sql-developer/*
  Command Line Interface:
  *https://github.com/utPLSQL/utPLSQL-cli*
- Maven-Plug-in:
  *https://github.com/utPLSQL/utPLSQL-maven-plugin*
- Demo-Projekt inklusive CI/CD-Integration in Travis:
  *https://github.com/utPLSQL/utPLSQL-demo-project*

Alle Codebeispiele inklusive Setup finden Sie unter:
*http://bit.ly/sithdb-aber-das-hat-gestern-noch-funktioniert*

Autoren-Blog:
*https://cleandatabase.wordpress.com*

## Über den Autor

Samuel Nitsche ist ein stets neugieriger Software-Entwickler und Oracle ACE mit beinahe 20 Jahren Entwicklungserfahrung. Nebenberuflich schreibt er regelmäßig zu Datenbank-Entwicklungsthemen, präsentiert auf Meetups und Konferenzen (gerne auch in Sith-Robe) und gehört dem Kern-Entwicklungsteam hinter utPLSQL an.

Samuel Nitsche
derpesse@gmail.com

# Kampf den Jugendsünden – Ein Plädoyer für Code-Reviews

Tobias Wirtz, Papstar GmbH

Es gibt diverse Gründe, warum SQLs und PL/SQL-Code spätestens nach einigen Jahren einem Code-Review unterzogen und in diesem Zuge nach festen Vorgaben modernisiert werden sollten – nicht erst dann, wenn Code nicht mehr lauffähig ist, weil verwendete Funktionalitäten den Status „Desupported" erreicht haben. Nebenbei verringert dies die Frustration der Entwickler, sich mit zum Teil Jahrzehnte altem und daher ungeliebtem Code auseinandersetzen zu müssen. In diesem Artikel berichtet der Autor von seinen Erfahrungen, die er seit 2001 bei der Papstar GmbH, einem europaweit agierenden Vertriebsunternehmen im Bereich Einwegartikel, gesammelt hat. Hier wird seit rund 20 Jahren ein ERP-System selbst entwickelt und eingesetzt, bei dem große Teile der Business-Logik in eine Oracle-Datenbank ausgelagert sind. Nachdem zunächst eine Oracle-Datenbank in der Version 8i zum Einsatz kam, wurde regelmäßig auf das neueste Release migriert, zuletzt im April 2019 auf Oracle 18 – jeweils ohne Reviews des Codes vorzunehmen. Die hier aufgeführten Code-Beispiele sind diesem System entnommen, jedoch gekürzt und teilweise vereinfacht.

Was versteht man eigentlich unter einem Code-Review? Das aus dem Englischen entnommene Wort review stammt ursprünglich von dem lateinischen Wort revidere, „wieder hinsehen", ab. Man sieht sich den Code also noch einmal an. Und genau das ist, stark vereinfacht ausgedrückt, das, was bei einem Code-Review passiert: eine manuelle Prüfung des Codes, also des Arbeitsergebnisses der Softwareentwicklung. In der heutigen Zeit sollten Reviews eigentlich unverzichtbar sein, besonders im Hinblick darauf, dass sie eine Maßnahme des Qualitätsmanagements darstellen können. „Der Einsatz von Reviews führt zu einer deutlichen Reduktion von Fehlern", findet man bei Wikipedia unter dem Lemma „Review" (Softwaretest) *[1]*. „Fehler, die im Review auffallen, können häufig bedeutend kostengünstiger behoben werden, als wenn diese erst während der Testdurchführung gefunden werden", steht nur kurz darauf.

Ein Review gilt als statische Testmethode und gehört in die Kategorie der analytischen Qualitätssicherungsmaßnahmen. Ohne den Entwicklern hier zu

nahe treten zu wollen, aber auch für diese gilt: Vertrauen ist gut, Kontrolle ist besser. Nur mit (mindestens) einem zusätzlichen, wachsamen Auge kann eine hohe Qualität des Codes dauerhaft sichergestellt werden. Der Übergang von der reinen Fehlersuche bis zur Optimierung des Codes nach funktionalen oder auch ästhetischen Gesichtspunkten ist bei einem Review meist fließend.

Die Bandbreite eines Code-Reviews kann dabei äußerst unterschiedlich sein: von einem einfachen „Drübergucken und Abnicken" des Codes durch einen zweiten Entwickler über eine standardisierte Checkliste bis zu einem sehr formalen und äußerst umfangreichen Prozess mit einem Gutachter und einem Moderator inklusive Protokollierung in einem Formblatt. Ein Schreibtischtest oder Code-Walkthrough, also das Nachvollziehen der Programm-Abläufe „im Kopf" des Entwicklers, ist dabei eine der Mindestanforderungen zur Prüfung des Codes. Eine weitere Option ist das Debugging. Hier bieten eigentlich alle am Markt verbreiteten Tools entsprechende Optionen, wie das Setzen von Breakpoints und das Auslesen von Variablen zur Laufzeit.

Ein weiterer Punkt, der bei einem Code-Review überprüft werden sollte, ist die Einhaltung von Entwicklungsstandards. Feste Regeln, was die Verwendung von Kommentaren, die Benennung und Deklaration von Variablen oder gar die Einrückung von bedingten oder verschachtelten Anweisungen angeht, sollte es in jedem Entwicklerteam geben.

Darüber hinaus kann ein Code-Review auch der Optimierung des Codes dienen: Hierzu zählt die Verwendung neuer Features, die bei der ursprünglichen Entwicklung des Codes entweder noch nicht existent oder dem Entwickler nicht bekannt waren. Der klassische Gegenpart dazu ist die Ablösung alter Funktionalitäten. Allerspätestens dann, wenn eine Funktionalität mit dem kommenden Release den Status Desupported erreicht, sollte hier Handlungsbedarf gegeben sein. Sonst geht man die Gefahr ein, beim (früher oder später notwendigen) Release-Wechsel Schiffbruch zu erleiden. Aber auch eine Optimierung des Codes in Hinblick auf die Erhöhung der Performance sollte im Zuge eines Reviews immer im Hinterkopf sein.

Neben dem Wie ist natürlich auch das Wann eines Code-Reviews von fundamentaler Bedeutung. So sollte neuer Code unmittelbar vor dem Produktivschalten immer einem Review unterzogen werden. Gerade bei unerfahrenen oder neuen Entwicklern, die mit den Entwicklungsstandards im Unternehmen noch nicht gut vertraut sind, sollte das Review dabei immer gemeinsam vorgenommen werden. Dadurch, dass der Entwickler selbst teilnimmt, kann dieser die notwendigen Anpassungen an „seinem" Code deutlich besser verinnerlichen.

Auch im direkten Zusammenhang mit einem Release-Wechsel sind Code-Reviews sinnvoll: Nicht immer ist ein neues Release auch zu 100 Prozent abwärtskompatibel. Möglicherweise problematischer Code, der durch das Studieren von Release Notes identifiziert wird, sollte vor der Umstellung systematisch bereinigt werden. Aber auch nach einem Release-Wechsel sollte ein Review durchgeführt werden. Neue Funktionalitäten bringen in der Regel einen Mehrwert, der auch schnellstmöglich ausgenutzt werden sollte. Das Review gilt in diesem Zusammenhang dann eher als regelmäßige Wartungsarbeit statt als außerplanmäßige Überprüfung. Eine Verschiebung der Einbindung neuer Features, beispielsweise auf die nächste funktionale Anpassung des Codes, führt oftmals dazu, dass diese über lange Zeit gar nicht genutzt werden.

Sollten sich die Entwicklungsstandards ändern, ist natürlich auch eine Anpassung des Codes notwendig. Nur so kann man verhindern, dass in der Praxis Code mit unterschiedlichen Entwicklungsstandards vorliegt. Gerade bei größeren Anpassungen der Standards sollten diese dann auch zeitnah bei sämtlichem Code zur Anwendung kommen. Zu guter Letzt: Code sollte auch nach festen Zeitintervallen einem Review unterzogen werden. Diese befinden sich jedoch in einem langfristigen Rahmen von mehreren Jahren – es ergibt kaum Sinn, alle 12 Monate den gleichen Code immer wieder zu überprüfen.

Hiermit soll der Tatsache Rechnung getragen werden, dass sich neben der Software und den Entwicklungsstandards auch das Know-how der Entwickler weiterentwickelt. Zumindest sollte es bei jedem guten Entwickler der Fall sein, im Laufe seines Berufslebens immer weiter dazuzulernen. Dies macht dann auch bei bestehendem und funktionierendem Code sowie unveränderten Entwicklungsstandards ein Review nach mehreren Jahren unabdingbar.

Sollte auf diese Reviews verzichtet werden, so veraltet der Code mit der Zeit. Wurde dies früher noch oftmals mit Sprüchen wie „never change a running system" schöngeredet, zeigt sich in der Praxis, dass Entwickler schnell eine Abneigung gegen alten und ungewarteten Code entwickeln. Ist dann eine funktionale Anpassung dieses Codes notwendig, entsteht hier auch schnell Frustration. Die allgemeine Motivation des Entwicklers sinkt.

Wie den meisten aufmerksamen Lesern aufgefallen sein dürfte, hat der Autor diesen Artikel bis zu diesem Punkt keineswegs auf die Oracle-Datenbank und SQL beziehungsweise PL/SQL bezogen. Vielmehr sind es allgemeine Aussagen zum Thema „Code-Reviews", die sich auch fast unverändert auf Source-Code in jeder anderen Sprache übertragen lassen. In den folgenden Abschnitten wird jedoch implizit auf SQL und PL/SQL Bezug genommen.

## PL/Scope

Auch wenn ein Code-Review eine manuelle Prüfung des Codes ist, gibt es technische Hilfsmittel, die eine Analyse des Codes durchaus vereinfachen. Im PL/SQL-Bereich ist eines dieser Tools PL/Scope. Der Vorteil an PL/Scope ist, dass es seit der Version 11.1.0.7, also dem Jahr 2008, standardmäßig mit der Oracle-Datenbank ausgeliefert wird und weder zusätzlich lizenziert noch installiert werden muss.

Daher erscheint es durchaus verwunderlich, dass in einer (selbstverständlich nicht repräsentativen) Umfrage während eines Vortrags des Autors bei der APEX connect im Mai dieses Jahres weniger als 20 Prozent der Zuhörer angaben, schon einmal von PL/Scope gehört zu haben.

Wer den SQL Developer verwendet, hat möglicherweise ohne sein Wissen schon einmal PL/Scope benutzt: Nichts anderes verbirgt sich nämlich hinter dem „PL/SQL-Compiler", der unter Voreinstellungen > Datenbank zu finden ist.

Standardmäßig ist PL/Scope deaktiviert, kann aber mit einem einfachen ALTER SESSION aktiviert werden *(siehe Listing*

*1)*. Im SQL Developer ist dies auch über oben genannten Menüpunkt möglich.

PL/Scope legt beim Kompilieren Metadaten für alle Bezeichner in PL/SQL-Objekten und SQL-Statements an. Diese sind im Anschluss für den Entwickler über Views abrufbar. Diese Views können dann verwendet werden, um beispielsweise die Einhaltung von Namenskonventionen für Variablen zu prüfen.

Hierzu legte der Autor eine Tabelle PLSQL_REGELN an, in der zu jeder Variante von Variablen (zum Beispiel CONSTANT oder VARIABLE) und Datentyp (zum Beispiel DATE oder VARCHAR2) ein regulärer Ausdruck (RegEx) hinterlegt ist, der als Vorgabe zur Benennung dient *(siehe Tabelle 1)*. Kurzer Hinweis: PL/Scope ist nicht Case-sensitiv, Groß-/Kleinschreibung kann daher nicht geprüft werden.

Anschließend wird eine Stored Function mit einer nicht den oben genannten Namenskonventionen entsprechenden Konstanten angelegt und kompiliert *(siehe Listing 2)*. Mithilfe einer SELECT-Anweisung *(siehe Listing 3)* wird dann überprüft, ob die dadurch in der PL/Scope-View user_identifiers generierten Deklarationen den Regeln aus PLSQL_REGELN ent-

```
ALTER SESSION
SET PLSCOPE_SETTINGS = 'IDENTIFIERS:ALL,STATEMENTS:ALL'
```

*Listing 1: Aktivierung von PL/Scope mit ALTER SESSION*

| TYP | DATENTYP | EXPRESSION |
|---|---|---|
| CONSTANT | DATE | \ACDT |
| FORMAL IN | DATE | \AP_ |
| VARIABLE | DATE | \ADT |
| CONSTANT | NUMBER | \ACN |
| FORMAL IN | NUMBER | \AP_ |
| VARIABLE | NUMBER | \AN |
| CONSTANT | VARCHAR2 | \ACV |
| FORMAL IN | VARCHAR2 | \AP_ |
| VARIABLE | VARCHAR2 | \AV |

*Tabelle 1: Beispielhafte RegEx-Definition für verschiedene Datentypen (Quelle: Tobias Wirtz)*

```
CREATE OR REPLACE FUNCTION FNC_WRONG_CONSTANT
RETURN VARCHAR2
IS
   -- Obwohl es sich um eine Konstante handelt, fehlt das Präfix
   -- "c" vor dem Namen
   vRufnummerPAPSTARKall CONSTANT VARCHAR2(10 CHAR) := '+49 2441 83';
BEGIN
   …
```

*Listing 2: Funktionsdefinition mit fehlerhaftem Namen einer Konstanten*

```
WITH Regeln as (SELECT * FROM PS_PLSQL_REGELN)
SELECT u.Zeile,
      u.Name,
      u.Typ,
      u.Datentyp,
      Regeln.Expression
  FROM (SELECT Name,
         Type Typ,
         Nachfolger Datentyp,
         Line Zeile
      FROM (SELECT ui.Name,
                   CASE WHEN LEAD(ui.Usage) OVER(ORDER BY USAGE_ID) = 'REFERENCE'
                       THEN LEAD(ui.Name) OVER(ORDER BY USAGE_ID)
                       ELSE NULL
                   END Nachfolger,
                   ui.Type,
                   ui.Usage,
                   ui.Usage_ID,
                   ui.Line,
                   ui.Col
              FROM user_identifiers ui
              WHERE ui.Object_Name = 'FNC_WRONG_CONSTANT')
          WHERE Type IN (SELECT DISTINCT Typ FROM Regeln)
              AND Usage = 'DECLARATION'
      ) u
  LEFT OUTER JOIN Regeln ON Regeln.Typ = u.Typ
                AND (NVL(Regeln.Datentyp, 'NIX') = NVL(u.Datentyp, 'NIX')
                    OR u.Datentyp IS NULL AND REGEXP_LIKE(u.Name, Regeln.Expression))
  WHERE NOT REGEXP_LIKE(u.Name, Regeln.Expression) OR Regeln.Expression IS NULL;
```

*Listing 3: SQL zur Überprüfung der Einhaltung von Namenskonventionen*

| ZEILE | NAME | TYP | DATENTYP | EXPRESSION |
|---|---|---|---|---|
| 6 | VRUFNUMMERPAPSTARKALL | CONSTANT | VARCHAR2 | \ACV |

*Tabelle 2: Ergebnis des SQL aus Listing 3 (Quelle: Tobias Wirtz)*

```
CREATE TABLE BEWEGUNGSDATEN
(ID NUMBER, Datum_Neuanlage DATE, Datum_Aenderung DATE);

CREATE TRIGGER TG_BEWEGUNGSDATEN
BEFORE INSERT OR UPDATE OR DELETE ON BEWEGUNGSDATEN
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
DECLARE
      dt_Datum DATE;

BEGIN

      SELECT SYSDATE INTO dt_Datum FROM DUAL;

      IF INSERTING THEN
          :NEW.Datum_Neuanlage := dt_Datum;
      END IF;

      IF UPDATING THEN
          :NEW.Datum_Aenderung := dt_Datum;
      END IF;
END;
```

*Listing 4: Definition einer simplen Tabelle mit Kontextwechsel im Trigger*

```
INSERT INTO BEWEGUNGSDATEN (ID)
SELECT Level
   FROM DUAL
CONNECT BY Level <= 500000;
-- 19 Sekunden MIT Kontextwechsel, 3 Sekunden OHNE
DELETE FROM BEWEGUNGSDATEN;
-- 24 Sekunden MIT Kontextwechsel, 5 Sekunden OHNE
```

*Listing 5: Vorher-Nachher-Vergleich der Ausführungszeiten beim Insert von 500.000 generierten Datensätzen*

```
INSERT INTO KONDITIONEN_DETAIL_KOPF
(
      Mandant,
      KundenNr,
      Laufzeit_Von,
      Benutzer_Neuanlage
)
VALUES
(
      rec.Mandant,
      rec.KundenNr,
      ADD_MONTHS(rec.Laufzeit_Von, 12),
      P_Benutzer
)
RETURNING ID
      INTO nID;
```

*Listing 6: Duplizieren eines Datensatzes mit gleichzeitigem Auslesen der neuen ID mit der RETURNING CLAUSE*

sprechen. Das Ergebnis umfasst eine Zeile mit dem Hinweis auf die Codezeile, den Namen, den Typ und Datentyp der Deklaration sowie dem RegEx. Ausgegeben werden natürlich nur die Deklarationen, bei denen es Differenzen zwischen dem RegEx und der tatsächlichen Bezeichnung gibt *(siehe Tabelle 2)*.

Dies ist nur ein relativ simples Beispiel dafür, wie die Metadaten ausgewertet und verwendet werden können. Die Möglichkeiten an dieser Stelle sind nahezu unbegrenzt. PL/Scope bietet zudem noch viele weitere Optionen, auf die hier nicht im Detail eingegangen werden kann. Hier möchte der Autor gern auf den sehr guten und umfangreichen Artikel von Sabine Heimsath im Red Stack Magazin 04/2017 verweisen *[2]*.

## Was sollte im Review auffallen?

### Code-Beispiel 1: Kontextwechsel

Kontextwechsel gehören zu den größten Problemen, denen man in der PL/SQL-Programmierung regelmäßig begegnet. Gerade bei großen Datenmengen geht die Performance stark in die Knie, wenn häufig zwischen SQL und PL/SQL gewechselt werden muss. Ein SELECT … FROM DUAL sollte daher in PL/SQL nach Möglichkeit vermieden werden. Insbesondere bei Triggern auf Tabellen mit vielen Bewegungen macht sich dies stark bemerkbar *(siehe Listing 4)*. Aber nicht nur große INSERTs oder UPDATEs dauern in diesem Beispiel unnötig lange: Die Kontextwechsel werden zudem auch vollkommen unnötigerweise bei DELETE-Operationen jedes Mal durchgeführt. Mit einem Verzicht auf den SELECT … FROM DUAL und einer direkten Zuweisung verringern sich die Ausführungszeiten auf einen Bruchteil *(siehe Listing 5)*.

### Code-Beispiel 2: RETURNING CLAUSE

Die Nutzung der RETURNING CLAUSE ersetzt bei einem INSERT oder UPDATE ein anschließendes und gleichzeitig fehler-

```
UPDATE DESADV_AUFTRAG_KOPF
   SET Liefer_Datum = rec_AUFTRAG.Liefer_Datum,
       Desadv_Datum = SYSDATE
WHERE AuftragsNr  = rec_AUFTRAG.AuftragsNr
   AND Mandant    = 17;

IF SQL%ROWCOUNT = 0 THEN

   INSERT INTO DESADV_AUFTRAG_KOPF
   (
       Mandant,
       AuftragsNr,
       Liefer_Datum,
       Desadv_Datum
   )
   VALUES
   (
       17,
       rec_AUFTRAG.AuftragsNr,
       rec_AUFTRAG.Liefer_Datum,
       SYSDATE
   );

END IF;
```

*Listing 7: Getrenntes UPDATE und INSERT*

```
MERGE INTO DESADV_AUFTRAG_KOPF z
USING (SELECT rec_AUFTRAG.AuftragsNr,
          rec_AUFTRAG.Liefer_Datum,
          17 AS Mandant,
          SYSDATE AS Desadv_Datum
       FROM DUAL) q
   ON (q.Mandant     = z.Mandant
   AND q.AuftragsNr  = z.AuftragsNr)
WHEN MATCHED THEN
UPDATE
   SET Liefer_Datum = q.Liefer_Datum,
       Desadv_Datum = q.Desadv_Datum
WHEN NOT MATCHED THEN
INSERT
(  Mandant,
   AuftragsNr,
   Liefer_Datum,
   Desadv_Datum
)
VALUES
(  q.Mandant,
   q.AuftragsNr,
   q.Liefer_Datum,
   q.Desadv_Datum
);
```

*Listing 8: Komfortabler und weniger fehleranfällig: Das MERGE-Statement*

```
DECLARE
   v_Langbezeichnung     VARCHAR2(200 CHAR);
...
BEGIN
   SELECT Langbezeichnung
       INTO v_Langbezeichnung
       FROM ARTIKEL
   WHERE Mandant     = P_Mandant
       AND ArtikelNr  = P_ArtikelNr;
...
```

*Listing 9: Statisch deklarierte Variable – riskant bei (nachträglichen) Änderungen der Spaltengröße*

anfälliges Nachlesen. Die Fehleranfälligkeit resultiert daraus, dass Anpassungen am DML-Statement auch immer beim anschließenden Nachlesen vorgenommen werden müssen – der Code muss also dupliziert werden. Neben „einfachen" Informationen wie zum Beispiel dem Inhalt einer IDENTITY-Spalte können auch Aggregat-Funktionen in der RETURNING CLAUSE verwendet werden *(siehe Listing 6)*. Zudem kann bei größeren Änderungen nicht nur eine Variable gefüllt, sondern auch BULK COLLECT verwendet werden.

### Code-Beispiel 3: MERGE statt getrenntem INSERT/UPDATE

Mit der Version 9i hat Oracle im Jahr 2001 den MERGE-Befehl eingeführt. Mit diesem lässt sich in einem Statement beispielsweise ein UPDATE ausführen, sollte ein Datensatz existieren. Ansonsten wird ein INSERT auf die Tabelle ausgeführt. Bevor es den MERGE-Befehl gab, musste entweder zuerst geprüft werden, ob ein Datensatz vorhanden ist. Alternativ konnte mithilfe von SQL%ROWCOUNT darauf reagiert werden, dass beim UPDATE-Statement kein Datensatz betroffen war *(siehe Listing 7)*. Da es sich auch hier um zwei verschiedene Statements handelt, ist ebenfalls eine Fehleranfälligkeit gegeben: So wird der im ursprünglichen Code an zwei Stellen angegebene Mandant 17 im MERGE nur noch einmal aufgeführt *(siehe Listing 8)*.

### Code-Beispiel 4: Kein INSERT INTO von Tabellenspalten in Variablen mit statischer Deklaration

Datenmodelle sind nicht in Stein gemeißelt. Oftmals erfordern neue oder geänderte Anforderungen eine Anpassung des Datenmodells, nicht selten müssen dabei bestehende Spalten verändert (meist vergrößert) werden. Gerade in komplexen und verteilten Systemen ist dabei nicht immer auf den ersten Blick ersichtlich, wo überall mit Daten aus einer Tabellenspalte gearbeitet wird. Bei der Verwendung von INSERT INTO mit statisch deklarierten Variablen besteht daher immer die Gefahr, eine Deklaration zu übersehen und dann später den altbekannten Fehler ORA-06502 zu erhalten *(siehe Listing 9)*.

Stattdessen sollte eine dynamische Variablendeklaration mit %TYPE verwendet werden *(siehe Listing 10)*.

```
DECLARE
   v_Langbezeichnung         ARTIKEL.Langebezeichnung%TYPE;
...
BEGIN
   SELECT Langbezeichnung
       INTO v_Langbezeichnung
       FROM ARTIKEL
   WHERE Mandant      = P_Mandant
       AND ArtikelNr  = P_ArtikelNr;
...
```

*Listing 10: Stattdessen: dynamische Variablendeklaration mit %TYPE*

```
SELECT NVL(NVL(mo.Stundenlohn, st.StundenLohn),
      NVL((SELECT Value
          FROM PARAMETER
          WHERE Section      = 'Abrechnung'
          AND Parameter      = 'Stundensatz'
          AND Benutzer       = 'Alle'),
          SF_MINDESTLOHN(st.Mandant))) Stundenlohn
   FROM ...
```

*Listing 11: Mehrfach verschachtelte NVL-Funktionen zur Ermittlung des Stundenlohns*

```
SELECT COALESCE(mo.Stundenlohn,
      st.StundenLohn,
      (SELECT Value
      FROM PARAMETER
      WHERE Section  = ‚Abrechnung‘
      AND Parameter  = ‚Stundensatz‘
      AND Benutzer   = ‚Alle‘),
      SF_MINDESTLOHN(st.Mandant)
      ) Stundenlohn
   FROM ...
```

*Listing 12: COALESCE – übersichtlicher und performanter*

### Code-Beispiel 5: COALESCE statt verschachtelter NVL-Anweisungen

Die NVL-Funktion ist praktisch – sie ist kurz und hat eine simple Syntax, was sie bei Entwicklern sehr beliebt macht. Sie hat allerdings entscheidende Nachteile: Da sie nur zwei Parameter hat, muss mit Verschachtelungen gearbeitet werden, wenn ein Anwendungsfall mit mehr Werten vorliegt. Außerdem werden immer alle Optionen tatsächlich ausgeführt, unabhängig davon, ob dies auch notwendig ist. Gerade bei der Verwendung von PL/SQL-Funktionen oder Subselects kann dies deutliche Auswirkungen auf die Performance haben *(siehe Listing 11)*.

Stattdessen sollte hier die COALESCE-Funktion verwendet werden. Diese ist nicht nur deutlich übersichtlicher, sondern nutzt auch die Short-Circuit-Evaluation. Sie bricht also ab, sobald ein Wert gefunden wurde, der nicht NULL ist *(siehe Listing 12)*.

## Fazit

Code-Reviews stellen eine wichtige Maßnahme der Qualitätssicherung dar. Besonders bei älterem Code sollten diese als zwingend notwendige Wartungsmaßnahme verstanden werden. So wird zum einen sichergestellt, dass der Code den aktuellen Entwicklungsstandards entspricht. Zum anderen werden regelmäßig neue, oftmals komfortablere Funktionalitäten eingebaut. Dies verringert die Entwicklung von Abneigungen der Entwickler gegen das Auseinandersetzen mit altem, rückständigem Code. Die angeführten Code-Fragmente sind einzelne Beispiele aus der Praxis, bei denen dem Autor Anpassungen im Zuge eines Reviews sinnvoll erschienen. Es handelte sich ausdrücklich um Optimierungen des Codes – nicht um Fehler, die behoben wurden.

## Quellen und weiterführende Informationen


[1] Review (Softwaretest) *https://de.wikipedia.org/wiki/Review_(Softwaretest)*

[2] Sabine Heimsath: Schöner Coden – PL/SQL analysieren mit PL/Scope in: DOAG Red Stack Magazin September 2017


## Über den Autor

Im Zuge seiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung bei der Papstar GmbH kam Tobias Wirtz 2001 erstmals mit Oracle-Datenbanken in Kontakt. Nach seiner Ausbildung blieb er im Unternehmen und entwickelte hier mit den Schwerpunkten SQL und PL/SQL das firmeneigene ERP-System weiter. Seit 2014 arbeitet er auch mit APEX und fühlt sich in der zugehörigen DOAG-Community zuhause. Er hielt zudem mehrere Vorträge auf der APEX connect. Daneben ist er seit dem Jahr 2015 auch der Ausbilder für IT-Berufe bei der Papstar GmbH und nebenberuflich als freier Motorsport-Journalist tätig.


Tobias Wirtz
tobias.wirtz@papstar.de

# Ein APEX Workflow-Tool für Citizen Developer

Michael Weinberger, Verbund

Oracle APEX ist eine Plattform, mit der auch Citizen Developer ohne tiefergehende Programmierkenntnisse in sehr kurzer Zeit sichere und stabile Webapplikationen ohne Codierungsaufwand erstellen können. Während sich sehr gute und sichere Benutzeroberflächen mit Leichtigkeit anlegen lassen, ist es weniger leicht, eine Geschäftslogik einzubinden. Eine solche muss explizit programmiert werden. Das mag in einzelnen Applikationen noch gehen – bei der Zusammenarbeit unterschiedlicher Applikationen (die vielleicht auch noch in anderen Sprachen wie Java oder .NET geschrieben sind) wird es komplex.

Für die Umsetzung solcher Anforderungen bieten sich Workflow-Tools an. Deren typische Installation (Tomcat/Java auf Server), Konfiguration, Ansteuerung über Webservice und grafische Darstellung (JavaScript) stellen aber ihrerseits wieder für Citizen Developer zumindest eine Herausforderung dar. Besonders in einer geschlossenen Umgebung wie beispielsweise apex.oracle.com

In dieser Miniserie möchte ich einen einfacheren Weg zeigen. Bereits 2005 wurde von Yeb Havinga eine Workflow-Engine in PL/SQL veröffentlicht *[2]*. Diese wurde als reine Background Engine umgesetzt und hat das Manko, keine Benutzeroberfläche zu bieten. Dank ihrer Implementierung in Oracle PL/SQL lässt sie sich jedoch sehr gut mit einer APEX-Oberfläche ergänzen. Für eine praktische Nutzung werden zwei Applikationen benötigt: eine Administrationsumgebung, mit der Workflows angelegt, verwaltet und überwacht werden können, und eine weitere, mit der alle in irgendeinem Workflow eingebundenen Personen ihre persönliche Aufgabenlisten zur Abarbeitung bekommen.

Ein Workflow kann für verschiedenste Arten von Prozessen eingerichtet werden. Einer der häufigeren Prozesstypen, mit dem Citizen Developer konfrontiert werden, ist meiner Erfahrung nach der Genehmigungs-/Benachrichtigungs-Workflow. Bei diesem Typus wird meist ein Dokument durch mehrere Prüfinstanzen und an verschiedene Stellen geschickt, um am Ende in diversen proprietären Applikationen abgelegt zu werden.

Auf diesen Typ von Workflow möchte ich mich in dieser Serie konzentrieren. Dabei verzichte ich darauf, ein durchgehendes Beispiel zu geben – in der Originaldokumentation *[2]* sind ausreichend Beispiele vorhanden.

Da dieses Thema zu umfangreich für einen einzigen Artikel ist – allein die Beschreibung der Workflow-Engine sprengt das Format –, wird es als Miniserie veröffentlicht. In Teil 1 gehe ich auf die PLFlow Engine ein und zeige, wie sie installiert und eingebunden werden kann.

In Teil 2 konzentriere ich mich auf die grafische Prozesssteuerung mittels APEX-Plug-in und erkläre, wie Benutzerinteraktionen mit APEX ermöglicht werden. Die Weitergabe von Daten oder Steuerbefehlen an andere Applikationen schließt Teil 2 ab.

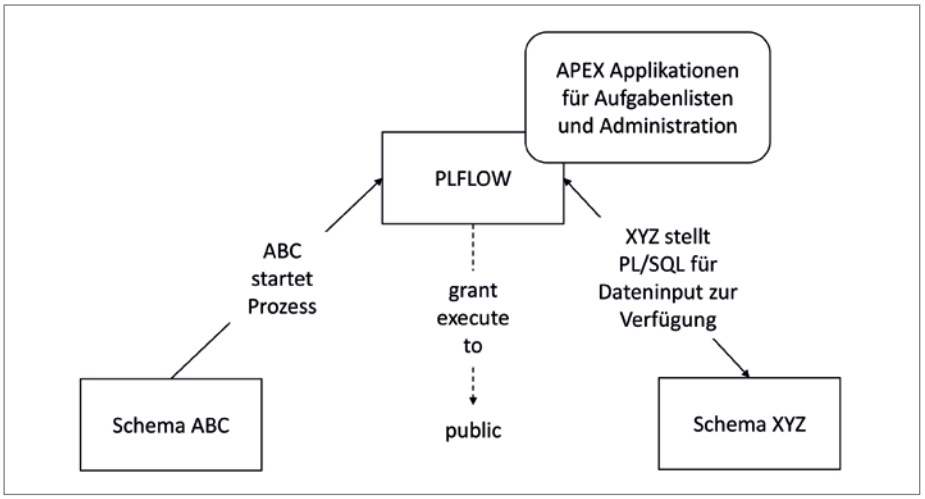



*Abbildung 1: Architektur (Quelle: Michael Weinberger)*

## Die Architektur

PLFlow wird nur 1x pro Oracle-Instanz benötigt. Die Installation der Engine sollte daher in einem eigenen Schema namens PLFLOW erfolgen. Dabei dürfen weder das Package PL_FLOW noch die Tabellen veröffentlicht werden. Stattdessen muss für das Starten von Workflows eine eigene Prozedur angelegt und veröffentlicht werden. Genauso muss ein anderes Schema (z.B. Schema_XYZ), in dem der Workflow irgendetwas anstoßen oder irgendwelche Daten ändern soll, der PLFlow Engine eine entsprechende Prozedur zur Verfügung stellen.

Für die aus zwei Applikationen bestehende Verwaltungsoberfläche empfiehlt sich ein eigener Workspace auf dem Schema PLFLOW.

Der Job Scheduler wird nicht unbedingt benötigt. Die PLFlow Engine ist eventbasiert – der Prozess läuft in einem durch, bis er irgendeine Eingabe oder Handlung benötigt. Danach pausiert er und wird erst wieder gestartet, wenn die nächste Eingabe getätigt wurde. Der Job Scheduler ist aber durchaus nützlich – wenn es um das Starten von Betriebssystemaktivitäten geht oder ein Workflow zeitbasiert gestartet werden soll.

## Workflow versus Prozess

Beide Begriffe sind weitgehend synonym. In diesem Artikel wird „Prozess" für die Definition eines Arbeitsablaufes und „Workflow" für einen in Abarbeitung befindlichen Prozess benutzt.

## Die Installation...

Nach dem Herunterladen von PLFlow sollten nicht die mitgelieferten Installationsskripts benutzt werden – da diese das Zielschema neu anlegen würden. Es ist einfacher, sich auf das PLFLOW-Schema

```
alter package PL_FLOW compile
PLSQL_Code_Type = native
PLSQL_Optimize_Level = 2
  reuse settings
```

*Listing 1: Native compile PL_FLOW*

| Tabelle | Sequenz |
|---|---|
| WF_PROCESSES | PRCE_SEQ |
| WF_PARTICIPANTS | PATI_SEQ |
| WF_ACTIVITIES | ACTI_SEQ |
| WF_ATTRIBUTES | ATRI_SEQ |

*Tabelle 1*

```
alter table WF_ATTRIBUTES add lov varchar2(4000);
```

*Listing 2*

```
update WF_PARTICIPANTS set id=0 where id=1;
```

*Listing 3*

anzumelden und die folgenden Skripts manuell auszuführen:

1. create_int_table_type.sql
2. who.sql
3. plflowddl.tab
4. plflowddl.ind
5. plflowddl.con
6. plflowddl.sqs
7. plflowrepstuff.sql
   An dieser Stelle kommt es zu Fehlermeldungen, weil DBMS_REPUTIL inzwischen deprecated ist. Wichtig: Die Funktion „make_parallel“ aus diesem Skript muss in jedem Fall installiert werden.
8. plflow.pks
   Hier werden vier Parameter benötigt – für die Kommunikation per E-Mail (smtp host, domain, default sender address und mail_error_to adress), die als Konstanten im Package eingetragen werden.
9. plflow.pkb
10. plflowsystemdata.sql
11. create_procdef_check_triggers.sql

Wichtig ist auch, die Programme ohne Debugging zu kompilieren – das kostet einiges an Performance – und native *(siehe Listing 1)*.

## ...und Modifikation für APEX

Um PLFLOW für APEX einfacher nutzbar zu machen, müssen ein paar before insert Trigger/Sequenzen für eine automatische Primärschlüsselerstellung (nur, falls diese null sind) hinzugefügt werden *(siehe Tabelle 1)*.

Außerdem wollen wir den Endanwender bei seinen Antworten unterstützen (und unverständliche Antworten verhindern) *(siehe Listing 2)*.

Dann sollte die Workflow-Engine immer ID 0 haben *(siehe Listing 3)*:

Für die Dokumente des Genehmigungs-/Benachrichtigungs-Workflows benötigen wir eine zusätzliche Tabelle (mit Trigger/Sequence) im PLFLOW-Schema *(siehe Listing 4)*:

In dieser Tabelle *(siehe Tabelle 2)* wird beim Start des Prozesses ein Dokument abgelegt. Das kann dann von dem jeweiligen Bearbeiter eingesehen werden.

Bei der Implementation der beiden APEX-Applikationen werden noch weitere neue Objekte beziehungsweise Anpassungen an bestehenden Objekten notwendig. Sie werden jedoch erst an entsprechender Stelle vorgestellt.

```
CREATE TABLE PLFLOW.WF_PAYLOADS
( ID              NUMBER(10) NOT NULL,
  PRIN_ID         NUMBER(10),
  NAME            VARCHAR2(255 BYTE),
  BESCHREIBUNG    VARCHAR2(2000 BYTE),
  MIME            VARCHAR2(75 BYTE),
  DATUM           DATE,
  DATEN           BLOB,
  BEGLEITTEXT     CLOB
);
```

*Listing 4: Erste Ergänzungen zu den Installationsskripts*

| | |
|---|---|
| PRIN_ID | Instanz des laufenden Prozesses |
| ID | Primärschlüssel |
| NAME | ...des Dokumententyps, z.B. Spesenkopien |
| BEGLEITTEXT | Ein großes Freitextfeld, in dem jeder Bearbeiter Kommentare zu der Bearbeitung einträgt. |

*Tabelle 2*

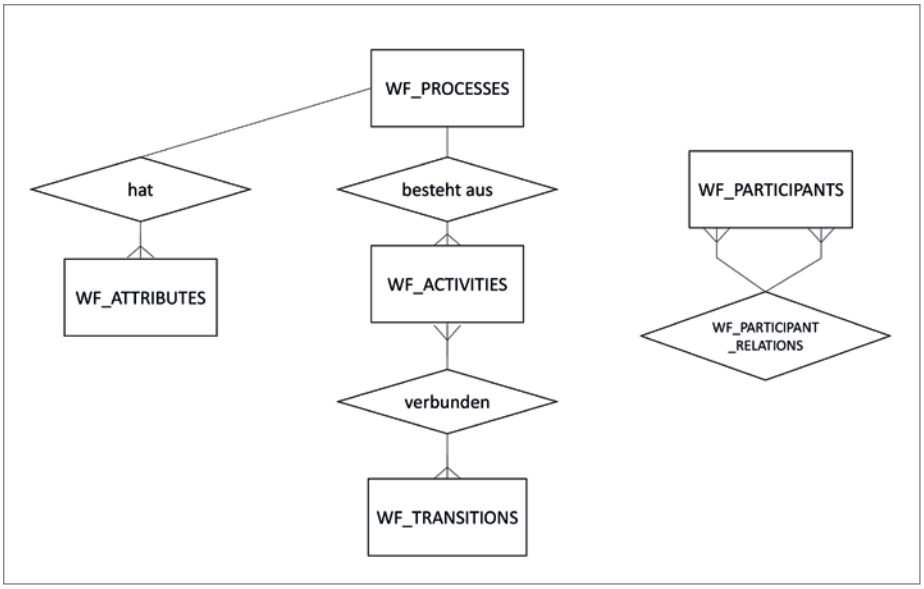



*Abbildung 2: Datenmodell (Quelle: siehe [1])*

```
INSERT INTO PLFLOW.WF_PROCESSES (NAME,
                                 DESCRIPTION,
                                 CREATION_DATE,
                                 VERSION,
                                 AUTHOR)
      VALUES ('SPESENABRECHNUNG EINREICHEN',
          'Prozess für Einreichung, Genehmigung und Abrechnung einer
Spesenabrechnung',
          SYSDATE,
          '1.0',
          'WEINBERM');
```

*Listing 5: Prozess mit SQL anlegen*

## Das Datenmodell von PLFlow

Die Installationsskripts legen alle Objekte an. Diese bauen das in *Abbildung 2* darge-

stellte Datenmodell auf:

Jeder Prozess wird in der WF_PROCESSES-Tabelle erfasst. Er besteht aus verschiedenen Aktivitäten beziehungsweise Prozessschritten (WF_ACTIVITIES), die untereinander verbunden sind (WF_TRANSITIONS). Dabei sind Verzweigungen möglich. Den Prozess begleitende Informationen (wie z.B. Ja/Nein-Entscheidungen) werden in WF_ATTRIBUTES definiert.

Die Liste involvierter Personen oder Rollen (WF_PARTICIPANTS) ist global – sie steht allen Prozessen gleichermaßen zur Verfügung. Vertretungen werden in WF_PARTICIPANT_RELATIONS gespeichert.

*Abbildung 2* ist so nicht ganz vollständig. Wird ein Workflow auf einem dieser Prozesse gestartet, erfolgt eine Kopie der Informationen in entsprechende *_INSTANCES-Tabellen. Erst auf diesen Schattentabellen wird dann eigentlich gearbeitet. Nach der Beendigung des Workflows erfolgt eine Speicherung in den *_INSTANCES_ARCH-Tabellen. Weiter gibt es APPLICATIONS und PARAMETERS – diese erlauben Programmaufrufe – sowie DEADLINES für einen Time-out-Mechanismus.

Der Zugriff auf die Definitionen eines Prozesses (Anlegen, Ändern) erfolgt über SQL. Für die Steuerung eines Workflows (Starten, Stoppen) darf dagegen ausschließlich das Package PL_FLOW verwendet werden. Trotzdem darf Letzteres nicht allgemein zur Verfügung stehen, denn durch die Kombination von Starten und Stoppen könnte jeder nicht nur seine Workflows starten, sondern auch beliebige andere (!) stoppen.

Deshalb ist noch eine Start-Workflow-Prozedur notwendig, die in PLFLOW erstellt wird und öffentlich (eventuell sogar als REST Service) zur Verfügung gestellt wird.

```
Insert into PLFLOW.WF_PARTICIPANTS(NAME, DESCRIPTION, PARTICIPANT_TYPE)
  Values ('MITAREBEITER', 'Der Antragsteller', 'ROLE');
```

*Listing 6: Rollen oder Mitarbeiter anlegen*

```
INSERT INTO wf_participant_relations
( pati_id_arg1, pati_id_arg2, relation_type )
```

*Listing 7: Berechtigungen vergeben. arg1 ist die ID des Mitarbeiters, arg2 die der Rolle. Als relation_type wird 'GRANT' angegeben*

| | |
|---|---|
| PRCE_ID | Fremdschlüssel auf die Prozesstabelle WF_PROCESSES |
| ID | ID des Prozessschrittes. Bildet zusammen mit PRCE_ID den Primärschlüssel der Tabelle. Ist außerdem die Standardsortierung. Wird aber durch Trigger automatisch und eindeutig vergeben. |
| NAME | |
| DESCRIPTION | |
| JOIN | Vorhergehende Schritte. Mögliche Werte: XOR, AND, NULL. Wenn es mehr als einen vorhergehenden Schritt gibt, bedeutet AND, dass alle Schritte erledigt sein müssen. XOR bedeutet, dass nur ein Schritt erledigt sein muss. Gibt es genau einen vorhergehenden Schritt, hat dort NULL zu stehen. |
| SPLIT | Betrifft die nachfolgenden Prozessschritte. Mögliche Werte: XOR, AND, NULL. Wenn es mehr als einen nachfolgenden Schritt gibt, bedeutet AND, dass alle Schritte ausgeführt werden. XOR bedeutet, dass nur einer der Schritte gestartet wird. Gibt es genau einen nachfolgenden Schritt, dann steht in der Spalte NULL |
| START_MODE und FINISH_MODE | Wahlweise AUTOMATIC oder MANUAL. Für einen Genehmigungsprozess ist MANUAL die passende Einstellung. AUTOMATIC wird für automatisch laufende Unterprogramme verwendet. |
| IMPLEMENTATION | Wahlweise: NO, SUBFLOW, TOOL. Manuelle Prozessschritte haben immer NO als IMPLEMENTATION.<br>Mittels TOOL wird die Steuerung an (PL/SQL-) Programme übergeben |
| PATI_QUERY | In PATI_QUERY wird festgelegt, wer (Rolle oder Person) diesen Prozessschritt durchführt. Das kann wahlweise eine statische ID oder eine Query, die eine statische ID zurückgibt, sein. Datenlieferant für beides ist die Tabelle WF_PARTICIPANTS |
| ASSIGN_TO | Beinhaltet eine SQL-Abfrage, falls der Durchführende dynamisch ermittelt werden muss – in der Regel dieselbe Person, die bereits einen bestimmten anderen Schritt durchgeführt hat |

*Tabelle 3*

## Die Verwaltungsoberfläche

PLFLOW bietet von sich aus keine Verwaltungsoberfläche oder Visualisierung. Da sich die Engine jedoch ausschließlich in der Oracle-Datenbank befindet, bietet sich eine mit APEX selbstgebaute Verwaltungsoberfläche an. In PLFLOW erfolgen alle Definitionsarbeiten an Prozessen durch direkte Einträge in die Tabellen.

Ein erster Prozess könnte beispielsweise folgendermaßen angelegt werden *(siehe Listing 5)*.

In APEX bieten sich Interactive Grids oder Interactive Reports mit Formularseiten als Verwaltungsoberflächen an. Im Artikel werde ich auch weiter SQL-Statements zu Beispielzwecken verwenden. Die Umsetzung in APEX überlasse ich dem Leser.

PLFLOW scheint nicht mehrmandantenfähig ausgelegt zu sein – nachdem aber WF_PROCESSES die zentrale Tabelle ist, kann dort ein Berechtigungssystem (das heißt Tabellen mit Rollen/Rechten pro Workflow) beschreiben, wer auf welchem Prozess Administrationsrechte hat. Alternativ könnte die Spalte AUTHOR eine ähnliche Rolle spielen.

Das bringt uns zu den Personen, die in einem Prozess mitwirken. Das Konzept von PLFLOW sieht Rollen (PARTICIPANT_

```
select pati_id
from wf_performers
where state = 'CURRENT'
and acin_id =
(select id
from wf_activity_instances
where acti_prce_id=10
and acti_id=10
and prin_id = :prin_id_in)
```

*Listing 8: Mögliches SQL-Statement für ASSIGN_TO Wert*

| | |
|---|---|
| ACTI_PRCE_ID_FROM | Ausgangsprozess (ID) |
| ACTI_ID_FROM | Von-Prozessschritt (WF_ACTIVITIES.ID) |
| ACTI_PRCE_ID_TO | Zielprozess (ID). Üblicherweise derselbe wir der Ausgangsprozess. Er kann aber (z.B. beim Start eines Subprozesses) abweichen. |
| ACTI_ID_TO | Nächster Schritt (WF_ACTIVITIES.ID) im laufenden Prozess |
| NAME | |
| DESCRIPTION | |
| CONDITION | Wahlweise NULL, OTHERWISE oder eine Formel oder eine Deadline.<br>Gibt an, unter welchen Bedingungen der Prozess zum nächsten Prozessschritt weitergeht.<br>NULL bedeutet, dass dieser nächste Prozessschritt immer gestartet wird.<br>OTHERWISE entspricht dem ELSE in einem IF..THEN.<br>EXCEPTION |
| CONDITION_TYPE | Muss „CONDITION“ enthalten, wenn die Spalte CONDITION eine Formel oder Deadline enthält.<br>EXCEPTION bedeutet, dass in CONDITION der Name einer Deadline steht |

*Tabelle 4*

| | |
|---|---|
| PRCE_ID | ID des zugehörigen Prozesses |
| DATA_TYPE | BOOLEAN, INTEGER oder CHARACTER |
| LENGTH | |
| NAME | Der Attributname, wie er in einer CONDITION abgefragt wird |
| DESCRIPTION | |
| KEEP | Y oder N (default) |
| LOV | Für APEX lesbare Liste der auswählbaren Attributwerte, durch Doppelpunkt getrennt. Beispielsweise Y:N |

*Tabelle 5*

```
SELECT rowid, -- incoming transitions
ACTI_PRCE_ID_FROM, ACTI_ID_FROM, ACTI_PRCE_ID_TO,
   ACTI_ID_TO, NAME, DESCRIPTION,
   CONDITION, CONDITION_TYPE
FROM PLFLOW.WF_TRANSITIONS
where (ACTI_PRCE_ID_TO = :P201_PRCE_ID)
   and (ACTI_ID_TO = :P201_ID)
UNION
SELECT rowid, -- outgoing transitions
ACTI_PRCE_ID_FROM, ACTI_ID_FROM, ACTI_PRCE_ID_TO,
   ACTI_ID_TO, NAME, DESCRIPTION,
   CONDITION, CONDITION_TYPE
FROM PLFLOW.WF_TRANSITIONS
where (ACTI_PRCE_ID_FROM = :P201_PRCE_ID)
   and (ACTI_ID_FROM = :P201_ID)
```

*Listing 9: SQL für Detail Grid Transitions, abhängig von einer Activity.*

WF_DEADLINES

| | |
|---|---|
| ACTI_ID | Activity, für die eine Deadline gesetzt wird |
| EXECUTION | SYNCHR, ASYNCHR |
| CONDITION | Tage als Zahl. Bildet die Deadline, ab der die Engine handelt |
| EXCEPTION_NAME | Ein Name, der als CONDITION in der WF_TRANSITIONS eingetragen wird. CONDITION_TYPE muss auf EXCEPTION stehen |

*Tabelle 6*

TYPE = ROLE) und Menschen (PARTICIPANT_TYPE = HUMAN) vor *(siehe Listing 6)*.

Eine Rollenzuweisung sieht folgendermaßen aus *(siehe Listing 7)*.

Hier ist zu beachten, dass PLFlow keine Zuordnung von Participants zu Prozessen hat! Es sind alle Mitarbeiter und alle Rollen für alle Prozesse sichtbar!

Das bedeutet zweierlei:

1. Die Rollenbezeichnungen sollten derart global definiert werden, dass sie für alle Prozesse gelten können.
2. Es sollten von Anfang an alle Mitarbeiter übernommen und später automatisch aktualisiert werden. Als ID für Mitarbeiter könnte dann die Personalnummer herangezogen werden. Quelle für die Mitarbeiter wäre beispielsweise ein internes Telefonverzeichnis oder das Active Directory oder...

Als nächster Schritt werden die einzelnen Prozessschritte beziehungsweise Aktivitäten angelegt. Auch das geschieht über SQL. Für die einzelnen Schritte des Prozesses (WF_ACTIVITIES) ist einiges zu beachten *(siehe Tabelle 3)*:

Es folgt ein mögliches SQL-Statement für ASSIGN_TO Wert. In diesem SQL müssen die ID für den Prozess (acti_prce_id) und den Prozessschritt (acti_id) hartcodiert eingetragen werden *(siehe Listing 8)*.

Als vorletzter Schritt folgen die Übergänge zwischen den einzelnen Aktivitäten. Diese werden in der Tabelle WF_TRANSITIONS eingetragen. Eine Zeile beschreibt immer den Übergang zwischen zwei Prozessschritten. Dabei ist es möglich, einen Prozessschritt von einem fremden Prozess aus zu starten (Subprozess).

Ein Übergang zu einer Aktivität kann einer Bedingung unterworfen sein. Welche Activity im laufenden Prozess als nächste nachfolgt, kann damit situationsabhängig sein. In der Programmierung entspricht das einem IF..THEN..ELSE.

Die wichtigsten Spalten sind in *Tabelle 4* aufgeführt.

Die Formeln in einer CONDITION folgen einer eigenen Syntax und müssen folgendermaßen eingetragen sein: a.name='HAPPY' und i.value = ‚N'

Immer identisch sind dabei a.name=“ und i.value

Ausgetauscht werden nur die Frage beziehungsweise der Attributname (HAP-

PY), die Antwort beziehungsweise der Attributwert (N) und der Operator zwischen i.value und ‚N' (=). Als Operator kommen alle Vergleichsoperatoren infrage.

Da sich die Formel in der Spalte CONDITION etwas komplexer gestalten kann, empfiehlt es sich, in APEX die Formel aus drei Feldern (Attributsname, Attributswert und Operator) zusammensetzbar zu implementieren. Auch sollte der Inhalt der Spalte CONDITION_TYPE automatisch gesetzt werden.

```
PL_FLOW.CreateProcessInstance(
prce_id_in => <ID vom Prozess, der gestartet werden soll>
prin_id_in => l_prin_id –Rückgabewert: neue Instanz ID des Prozesses
   );
```

*Listing 10: Neuen Prozess anfordern*

```
INSERT INTO PLFLOW.WF_PAYLOADS (
   ID, PRCE_ID, NAME,
   BESCHREIBUNG, DATEN, MIME,
   DATUM, PRIN_ID)
VALUES ( null /* ID */,
 <ID vom Prozess, der gestartet werden soll> /* PRCE_ID */,
 <sprechender Name> /* NAME */,
 <optionale Beschreibung>/* BESCHREIBUNG */,
 <der BLOB> /* DATEN */,
 /* MIME */,
 /* DATUM */,
l_prin_id /* PRIN_ID */);
```

*Listing 11: Optional: Dokument hochladen*

```
PL_FLOW.AssignProcessInstanceAttribute(
          prin_id_in  => l_prin_id,
          name_in     => <Name des Attributes>,
          value_in    => <Wert des Attributes> );
```

*Listing 12: Optional: Startwerte zuweisen*

```
PL_FLOW.StartProcess(
      prin_id_in => l_prin_id,
      pati_id_in => < ID des Participants, der den Prozess startet>
   );
```

*Listing 13: Prozess starten*

```
SELECT id into v_first_activity_id
   FROM wf_activities
   WHERE PRCE_ID = <ID vom Prozess, der gestartet werden soll>
   and start_activity_prce_id = 1;

SELECT id
   INTO l_acin_id
   FROM wf_activity_instances
   WHERE prin_id = l_prin_id
   AND acti_id = v_first_activity_id;

PL_FLOW.ChangeActivityInstanceState(
      acin_id_in => l_acin_id,
      state_in   => 'RUNNING',
      pati_id_in => l_pati_id_manager
   );
```

*Listing 14: Ersten Prozessschritt beginnen*

Das führt uns zu der letzten *Tabelle 5*, WF_ATTRIBUTES.

Hier werden alle Informationen, Fragen und Antworten, die im Prozessablauf entstehen, gesammelt. Es gibt ein besonderes Attribut mit dem Namen PAYLOAD und dem Data_type PAYLOAD. Dieses ist als Hinweis für die APEX-Applikationen gedacht, damit diese wissen, dass in der WF_PAYLOADS-Tabelle wichtige Informationen (z.B. die Spesenabrechnung) stehen.

Die Zusammenhänge zwischen den Tabellen WF_ACTIVITIES und WF_TRANSITIONS lassen sich in APEX am einfachsten über Master-Detail abbilden. Detailtabelle ist dabei die WF_TRANSITIONS *(siehe Listing 9)*.

## Deadlines

Sind eine schöne und wichtige Erweiterung. Jede Aktivität kann mit einer Deadline versehen werden. Ist die entsprechende Zeit abgelaufen, wird ein Event getriggert, der eine entsprechende Transition auslöst *(siehe Tabelle 6)*.

## Starten eines Prozesses aus einer Applikation heraus

Eingangs habe ich erwähnt, dass ein großer Vorteil von PLFLOW in der einfachen Ansteuerung aus eigenentwickelten APEX-Applikationen besteht.

Da die Ansteuerung ausschließlich über das PL_FLOW-Package durchgeführt wird, genügt ein einfacher APEX-Prozess, der folgende PL/SQL-Schritte durchführt:

1. Einen neuen Prozess anfordern *(siehe Listing 10)*
2. Das zu bearbeitende Dokument hochladen *(siehe Listing 11)*
3. Weitere Attribute initialisieren *(siehe Listing 12)*

Schritt 3 ist nur notwendig, wenn zusätzliche, spezielle Startparameter existieren, die nicht aus dem INITIAL_VALUE der Ta-

```
PLFLOW.StarteProzess( );
```

*Listing 16: Prozessstart aus einem Programm heraus*

```
PL_FLOW.ChangeActivityInstanceState(
      acin_id_in  => l_acin_id,
      state_in    => 'COMPLETED',
          pati_id_in => l_pati_id_manager
          < ID des Participants, der den Prozess startet>
      );
```

*Listing 15: Ersten Prozessschritt beenden*

belle WF_ATTRIBUTES abgeleitet werden können.

4. Prozess starten *(siehe Listing 13)*
5. Erste Aktivität im Prozessfluss suchen und starten *(siehe Listing 14)*
6. Erste Aktivität abschließen *(siehe Listing 15)*

Diese 6 Schritte sollen nicht direkt in APEX ausgeführt werden. Sie müssen aus Sicherheitsgründen selbst in einer eigenen Startprozedur oder Funktion zusammengefasst werden. Diese Prozedur (und nur die) wird dann veröffentlicht.

Damit ergibt sich für den APEX-Entwickler, der einen Prozess starten will, ein vereinfachter Aufruf *(siehe Listing 16)*.

Als Übergabeparameter sind dazu notwendig:

- ID des Prozesses, der gestartet werden soll
- Das Dokument als BLOB, das Gegenstand des Genehmigungs-Workflows ist
- ID der Person, die den Workflow startet

Rückgabeparameter wäre dann die

- ID der gestarteten Prozessinstanz

In APEX wird dieser Aufruf einfach als „PL/SQL-Prozess“ ausgeführt.

Diese Prozedur kann natürlich auch für externe Applikationen als Webservice (über APEX/ORDS) zur Verfügung gestellt werden.

Jeder Prozess ist eindeutig, in WF_PROCESSES gespeichert und nur 1x vorhanden. Es kann jedoch von jedem Prozess beliebig viele Instanzen (Workflows) in WF_PROCESS_INSTANCES geben, die gerade laufen (RUNNING).

Nach Beendigung eines Workflows wird die Instanz in WF_PROCESS_INSTANCES_ARCH archiviert. Dort findet sich auch der Status (COMPLETED, TERMINATED, ABORTED), wie der Workflow abgeschlossen wurde.

## Prozesssteuerung

Über die Prozedur PL_FLOW.ChangeProcessInstanceState(prin_id_in, state_in) lassen sich Prozesse unterbrechen und können weiterlaufen. Erlaubte Zustände für state_in sind:

- RUNNING
- SUSPENDED
- TERMINATED
- ABORTED

Diese Prozedur sollte nur über die Administrationsoberfläche nutzbar sein, da es in PL_FLOW keine Prozessbesitzer gibt.

## Ende des Workflows:

Ist ein Workflow beendet, erfolgt keine automatische Benachrichtigung. Wenn eine solche benötigt wird, um beispielsweise ein Ergebnis zurückzumelden, ist das als expliziter Workflow-Schritt vorzusehen.

## Fazit Teil 1

Mit dem Wissen aus Teil 1 ist es bereits möglich, eine administrative APEX-Oberfläche zu bauen, mit der eigene automatische Prozesse definiert werden können. Workflows können mit einem Einzeiler aus Drittapplikationen gestartet und über die Administrationsoberfläche gestoppt werden.

Die manuelle Interaktion der Teilnehmer in einem Workflow wird in Teil 2 detailliert erklärt. Außerdem wird ein APEX-Plug-in vorgestellt, mit dem sich der Prozessfluss grafisch darstellen und bearbeiten lässt.

## Quellen


[1] (2015): ERD, *https://erdplus.com/*
[2] Yeb Havinga (2005) *https://sourceforge.net/projects/plflow/*


## Über den Autor

Michael Weinberger ist Teamleiter bei VERBUND. Er arbeitet seit 2005 mit APEX/HTMLDB und hat 2009 begonnen, eine konzernweite Low-Code-Applikationsplattform (self-service, automatische Authentifizierung, SQL-Schnittstellen zu Benutzerverzeichnissen und ein zentrales Plug-in/Code Repository) für Fachbereichsentwickler mittels APEX aufzubauen. Seitdem bildet er neue Entwickler in den Fachbereichen aus und unterstützt diese beim Aufbau ihrer eigenen Applikationslandschaften.

Sein aktuelles Projekt ist die erste kommerzielle SaaS-Plattform des Konzerns.


Mag. Michael Weinberger
Michael.Weinberger@verbund.com

# Lieblingskind oder Totgeburt?

Holger Bär, Atos

In diesem Artikel wird der seit Oracle 12c eingeführte Automatic Big Table Cache näher betrachtet. Die generelle Funktionsweise, aber vor allem die Grenzen der Technologie, sollen aufgezeigt werden, um die Basis für einen erfolgreichen Einsatz des Cache zu liefern. Die Grundlage des Artikels lieferte neben der intensiven Auseinandersetzung mit der Technologie, die hinter dem neuen Cache steckt, ein Kundenprojekt, in dem der Automatic Big Table Cache seit über 2 Jahren eingesetzt wird.

Eine eher weniger beachtete Neuerung von Oracle 12c ist der Automatic Big Table Cache. Obwohl offiziell Teil der In-MemoryFunktionen, findet man überraschend wenig Blogs oder Artikel zu der Funktion. Die vorhandenen Informationen (eine kurze Erwähnung innerhalb von In-Memory White Papers, ein nicht über die prinzipielle Produktdemo hinausgehender Post von Ulrike Schwinn sowie ein vom Autor leider verpasster Vortrag auf der DOAG Konferenz 2015 von Markus Geis) und eine Reihe von „Me Too" Blog-Posts ohne erkennbaren Praxisbezug lassen den interessierten Administrator doch etwas ratlos zurück. Grund genug, sich einmal etwas genauer mit dem Thema auseinanderzusetzen und der Frage nachzugehen: Wird das das neue Lieblingskind oder ist es eher eine Totgeburt?

Um diese Frage zu beantworten, muss man die Funktionsweise des Automatic Big Table Cache im Vergleich zu dem klassischen Buffer Cache genauer betrachten.

Der Buffer Cache ist (für jede konfigurierte Blockgröße getrennt) in sogenannte Working Data Sets unterteilt, in denen die Buffer (genauer, die Buffer Header) in Form von doppelt verlinkten Listen über einen Least-Recently-Used-Algorithmus verwaltet werden. Dabei wandern – vereinfacht dargestellt – wenig benutzte Blöcke zum „kalten" Ende des Cache, häufig benutzte Blöcke zum „heißen" Ende. Für eine genauere Beschreibung des LRU-Algorithmus, der Anpassung des Touch Count und weiterer Details sei dem interessierten Leser das Buch „Oracle Core" von Jonathan Lewis empfohlen.

## Funktionsweise des ABTC

Automatic Big Table Cache wird zu den In-Memory-Funktionen von Oracle gerechnet, unterliegt aber im Gegensatz zu diesen nicht einer gesonderten Lizenzierung und steht in allen Editionen zur Verfügung. Im Unterschied zum blockorientierten LRU-Algorithmus des Buffer Cache werden ganze Segmente für die Entscheidung betrachtet, was im Cache gehalten werden soll. Dadurch wird die Verwaltung des Cache zumindest theoretisch stark vereinfacht. Ähnlich dem Touch-Count des LRU-Algorithmus wird einfach für ein ganzes Segment gezählt, wie oft darauf mit einem Full Scan zugegriffen wird. Den Zähler nennt Oracle folgerichtig auch Temperatur; je häufiger auf ein Segment zugegriffen wird, desto höher ist also seine Temperatur. Die Erhöhung erfolgt dabei immer in 1000er-Schritten.

Die Größe des ABTC wird über den Parameter db_big_table_cache_percent_target gesetzt und in Prozent des gesamten Buffer Cache angegeben. Der Parameter ist dynamisch, kann also auch ohne Neustart der Instanz angepasst werden und darf Werte zwischen 0 und 90 annehmen. Damit der ABTC zum Einsatz kommt, muss außerdem PARALLEL_DEGREE_POLICY auf AUTO oder ADAPTIVE stehen.

Um die Arbeit des Cache zu überwachen, gibt es schließlich noch 2 SYS-Views

- [G]V$BT_SCAN_CACHE
- [G]V$BT_SCAN_OBJ_TEMPS

Weiter sollte man noch wissen, dass der ABTC in einer Single-Instanz für serielle und parallele Abfragen zur Anwendung kommt, im RAC lediglich bei parallelen Abfragen. Und auch wenn der Name anderes suggeriert, so können neben Tabellen beziehungsweise deren (Sub-)Partitionen auch Indexsegmente im Cache gehalten werden – falls denn auf den Index mit einem Full Scan zugegriffen wird. Im Folgenden ist immer ein Full Scan gemeint, wenn von Laden oder Lesen die Rede ist.

Aber wozu das Ganze? Wie der Name schon sagt, geht es um große Segmente. Als groß wird in diesem Zusammenhang alles betrachtet, was größer ist als der Wert für den _small_table_threshold. Dieser Parameter hat als Default 2% des DB_CACHE_BUFFER und ist in Blöcken angegeben. Segmente, die größer als _small_table_threshold sind, werden in der Regel unter Umgehung des Buffer Cache per Direct Read direkt in die PGA geladen. Da diese Blöcke dann wieder verworfen werden, führt dies zu ständigem IO, falls ein oder mehrere „große" Segmente wiederholt oder in unterschiedlichen Sessions abgefragt werden. Genau hier setzt der ABTC an, um häufig genutzte Segmente dann doch wieder im Speicher zu halten und sie allen Anwendern zur Verfügung stellen zu können. Somit steht der Automatic Big Table Cache zwischen dem FULL DATABASE CACHING und dem Smart Flash Cache und dient wie diese dazu, wo möglich IO zu reduzieren oder zu optimieren.

Auch Segmente, die größer sind als der konfigurierte Big Table Cache (also deren Segment über MEM_BUFFER_ALLOC aus der View [g]V$BT_SCAN_CACHE liegt), werden für den Cache in Betracht gezogen, jedoch nur teilweise geladen.

## ABTC in Aktion

Auch wenn das grundlegende Setup in diversen Artikeln hinreichend beschrieben

ist, soll in einem minimalen Beispiel die Methode gezeigt werden, mit der die Analyse der Arbeitsweise des ABTC durchgeführt wurde. Stellen wir also zunächst einmal sicher, dass wir ABTC konfiguriert haben:

Als Nächstes benötigen wir eine Tabelle, um einen Workload zu simulieren; dabei müssen wir sicherstellen, dass die Tabelle genug Blöcke enthält, um für den ABTC in Betracht zu kommen. In unserem Fall, einem kleinen Testsystem mit _small_table_threshold=2598 (2% des Buffer Cache), legen wir eine Tabelle mit 1.000.000 Zeilen an. In einem Tablespace mit ASSM und einer Blockgröße von 8k erhalten wir eine Tabelle mit 29696 Blöcken *(siehe Listing 2)*. Die Größe ist bewusst so gewählt, um nicht in dem Bereich von 2-10% des Buffer Cache zu landen, für den zusätzliche Algorithmen zur Anwendung kommen, bei der Entscheidung, ob gecacht wird oder nicht (siehe *[1]*).

Da auf der Tabelle kein Index angelegt wurde, wird jede Abfrage in einem FULL TABLE SCAN durchgeführt werden müssen. Damit ist die Tabelle ein guter Kandidat für den ABTC:

Dies lässt sich in der View V$BT_SCAN_OBJ_TEMPS leicht beobachten, da neben der Temperatur auch noch die Policy angegeben wird – als MEM_ONLY wird dabei die vollständige Speicherung des Segments im Cache bezeichnet, bei MEM_PART gelingt dies nur teilweise und DISK bedeutet, dass das betroffene Segment zwar ein Kandidat für ABTC wäre, aufgrund seiner Temperatur jedoch aktuell nicht im Cache gehalten wird *(siehe Listing 3)*.

Nun stellt sich natürlich die Frage, wie eine einmal in den Cache geladene Tabelle von dort jemals wieder verschwindet.

Dafür hat Oracle 2 Mechanismen vorgesehen

- Verdrängung
- Abkühlung

Bei der Verdrängung wird einfach ähnlich zum LRU-Mechanismus ermittelt, ob die Temperatur eines Objekts ein anderes, das bereits im Cache liegt, verdrängt. Dazu muss die Temperatur des Objekts höher sein als die des zu verdrängenden Objekts und es darf kein anderer freier Speicher zur Verfügung stehen.

Doch was passiert mit einer Tabelle, die für eine gewisse Zeit populär ist, dann aber nicht mehr abgefragt wird? Hier hat

```
ALTER SYSTEM SET db_big_table_cache_percent_target=70;
--ALTER SYSTEM SET COMPATIBLE='12.1.0.2';
ALTER SYSTEM SET PARALLEL_DEGREE_POLICY=AUTO;
```

*Listing 1: Aktivieren des Automatic Big Table Cache*

```
create table medium (id , vcpad ) as
with rs as ( select null
from dual connect by level <= 1000)
select    rownum rn ,
lpad (rownum,200,'_')
from rs,rs;


Select blocks
from user_segments
where segment_name='MEDIUM';

   BLOCKS
----------
   29696

select min(id), max(id), count(vcpad)
from medium;

   MIN(ID)     MAX(ID)      COUNT(VCPAD)
----------   ----------  ------------
      1       1000000     1000000
```

*Listing 2: Erstellen einer Testtabelle*

```
select    DATAOBJ#, SIZE_IN_BLKS, TEMPERATURE,
      POLICY, CACHED_IN_MEM
from v$bt_scan_obj_temps;

   DATAOBJ#    SIZE_IN_BLKS   TEMPERATURE    POLICY      CACHED_IN_MEM
----------   ------------   -----------   ----------  -------------
   107188      29631          1000          MEM_ONLY       29631
```

*Listing 3: Inhalt von v$bt_scan_obj_temps*

Oracle einen weiteren Mechanismus geschaffen, durch den die Temperatur reduziert wird. Leider ist dieser Mechanismus undokumentiert, scheint jedoch eine zeitliche Komponente im Sinne von „wann wurde zuletzt auf die Tabelle zugegriffen“ zu enthalten.

## Cachenutzung monitoren

Ganz im Sinne von „alles automatisch“ hat Oracle auch keine Methode dokumentiert, die Nutzung des Cache zu monitoren beziehungsweise die Effektivität in irgendeiner Form festzustellen. Da die Datenmenge sehr überschaubar ist, hat es sich bewährt, in regelmäßigen Abständen einen Snapshot von v$bt_scan_obj_temps zu erstellen. Damit bekommt man wenigstens eine gewisse Vorstellung von den in der Praxis tatsächlich im Cache liegenden Segmenten und kann sich auch ein gewisses Bild davon machen, ob die Dimensionierung des Cache ausreichend ist.

In *Listing 4* ist eine einfache Methode dargestellt, mit deren Hilfe Snapshots regelmäßig erstellt werden können.

Auf den ersten Blick mag es irritierend erscheinen, auch Inhalte aus DBA_OBJECTS zu sichern, das hat jedoch zwei einfache Gründe: Betrachtet man die Historie über einen längeren Zeitraum, dann hat man schnell Probleme, ein Segment noch zu identifizieren – sei es, weil die Tabelle beziehungsweise der Index inzwischen gelöscht wurde, oder aber weil sich die DATA_OBJECT_ID ge-

ändert hat. Dies ist der Fall bei Operationen wie zum Beispiel TRUNCATE TABLE, Index rebuild oder auch ein ALTER TABLE … MOVE.

Mithilfe der historisierten Daten ist es dann ein Leichtes, Fragen zu beantworten wie zum Beispiel „Welche Tabellen sind wann im Cache zu finden?“. In *Abbildung 1* ist exemplarisch der Temperaturverlauf für eine willkürlich gewählte, aber regelmäßig populäre Tabelle dargestellt. Es ist gut zu erkennen, wie die Tabelle nach einer gewissen Zeit der Popularität wieder abkühlt. Im Diagramm nicht dargestellt ist die jeweils angewandte Policy, es ist also nicht erkennbar, ob die Tabelle zum jeweiligen Zeitpunkt überhaupt im ABTC liegt.

## Data Object ID und daraus resultierende Probleme

Bei der Analyse der historisierten Daten eines Produktivsystems fiel auf, dass die oben erwähnte Zuordnung der Cacheeinträge zu einer DATA_OBJECT_ID noch weitere Probleme birgt. Im Gegensatz zum „normalen“ Buffer Cache, in dem beispielsweise nach einem Truncate der Status der Blöcke auf „free“ gesetzt wird, bleibt die DATA_OBJECT_ID zunächst im Big-Table Cache erhalten. Erst wenn aufgrund der Temperatur die DATA_OBJECT_ID durch eine andere verdrängt wird, wird der belegte Speicher tatsächlich auch recycelt. Somit ist nach jeder Aktion die DATA_OBJECT_ID verändert und ein Teil des Cache nicht nutzbar.

## Automatic? Nein Danke!

Der offensichtlichste Nachteil des ABTC ist die sehr einfach gehaltene Verwaltung des Cache – erst wenn die Temperatur eines Segments durch häufige Zugriffe höher wird als die niedrigste Temperatur eines bereits im Cache abgelegten Segments, kann das neue Segment Platz im Cache belegen. Wird ein kleineres Segment durch ein größeres verdrängt, so kann nur der dadurch frei gewordene Platz genutzt werden – die Policy wird dann als MEM_PART angezeigt werden, Zugriffe auf das Segment werden also vorerst weiterhin IO erzeugen. Dabei suggeriert die Policy, dass nur ein Teil der Blöcke von der Platte nachgeladen werden müssen, Experimente mit Autotrace in SQL*Plus legen jedoch nahe, dass zumindest in manchen Fällen alle Blöcke von Platte geladen werden.

Bei einem Workload, der häufig zwischen mehreren Kandidaten hin- und herwechselt, kann daher in unglücklichen Konstellationen ein Segment gerade erst dann im Cache landen, wenn es anschließend nicht oder nur noch selten wieder benutzt wird. Zumindest nicht, bevor es nicht durch ein anderes Segment wieder

```
create table bt_cache_history
 (
  tstamp         date,
  source         varchar2(10 byte),
  owner          varchar2(128 byte),
  data_object_id    number,
  object_type    varchar2(25 byte),
  object_name    varchar2(128 byte),
  size_in_blks   number,
  temperature    number,
  policy         varchar2(10 byte),
  cached_in_mem  number
 ) ;


create table dbo_history
 (
  insertts       date default sysdate,
  owner          varchar2(128 byte),
  object_name    varchar2(128 byte),
  object_type    varchar2(23 byte),
  data_object_id    number,
  created        date,
  last_ddl_time  date
 ) ;

create or replace package btmon
is
  procedure snap;
end;
/
create or replace package body btmon
as
  procedure dbo_snap
  is
  begin
      insert into dbo_history
      select
          sysdate,
          owner,
          object_name || case when subobject_name is not null then
'.'||subobject_name end object_name,
          object_type,
          data_object_id,
          created,
          last_ddl_time
      from   dba_objects
      where data_object_id in ( select distinct data_object_id
                                from bt_cache_history);
      commit;
  end dbo_snap;

  procedure btcache_snap
  is
  begin
      insert into bt_cache_history
      select
          sysdate as tstamp,
          'objects' as source,
```

```
          owner,
          dataobj#,
          object_type,
          object_name || case when subobject_name is not null then
'.'||subobject_name end object_name,
          size_in_blks,
          temperature,
          policy,
          cached_in_mem
      from dba_objects do right join v$bt_scan_obj_temps bso on (data_
object_id = dataobj#)
      union
      select
          sysdate as tstamp,
          'recyclebin' as source,
          owner,
          dataobj#,
          type,
          object_name,
          size_in_blks,
          temperature,
          policy,
          cached_in_mem
      from dba_recyclebin rb, v$bt_scan_obj_temps bso
      where purge_object=dataobj#
      order by temperature desc
      ;
   end btcache_snap;
   procedure snap
   is
   begin
      btcache_snap;
      dbo_snap;
   end snap;
end;
/

begin
   dbms_scheduler.create_job(job_name=>'snap_btcache',
          enabled => true,
          job_type => 'STORED_PROCEDURE',
          job_action => 'btmon.snap',
          repeat_interval =>'FREQ=MINUTELY;INTERVAL=5');
end;
/
```

*Listing 4: Einfaches Monitoring der Cachenutzung*

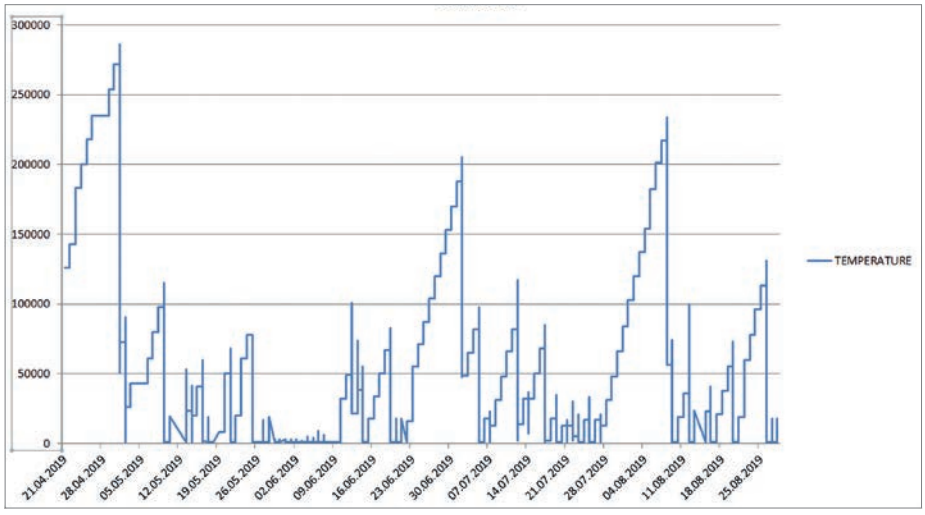



*Abbildung 1: Temperaturverlauf am Beispiel einer Tabelle (Quelle: Holger Bär)*

verdrängt wurde. In diesem Fall müsste der Cache entsprechend vergrößert werden, um alle Kandidaten halten zu können.

Leider gibt es auch in der aktuellen Version 19c keine dokumentierte Möglichkeit, gezielt Segmente aus dem Cache zu entfernen oder den Cache komplett zu leeren. Die naheliegende „ALTER SYSTEM FLUSH SCAN_CACHE"-Syntax hierzu fehlt. Ebenso fehlt eine Option, Segmente von vorneherein komplett für den Big Table Cache auszuschließen, sieht man von der Größenbeschränkung durch das Sizing des Cache ab.

Ein „ALTER SYSTEM FLUSH BUFFER_CACHE" gibt zwar die Buffer Header in v$bh wieder frei (Status wechselt auf „free"), da aber die Einträge in v$bt_scan_obj_temps nicht gelöscht werden, wird der Cache effektiv nicht nutzbar.

## V$BT_SCAN* Views lügen

Nachdem die Probleme mit der DATA_OBJECT_ID deutlich wurden, wurde die Belastbarkeit der Angaben in V$BT_SCAN_OBJ_TEMPS genauer untersucht, indem gezielt die durch die angegebene POLICY zu erwartende Reduzierung an physikalischem IO mit den tatsächlich gefundenen Werten in AUTOTRACE verglichen wurden. Dabei zeigt sich leider ein sehr ernüchterndes Bild: Die Angabe CACHED_IN_MEM ist allenfalls bei klassischen HEAP-Tabellen plausibel und nur für die POLICY MEM_ONLY ein brauchbarer Indikator dafür, dass Daten aus dem Cache geladen werden.

Fehlerhafte Angaben erhält man zum Beispiel bei Single Instance und dem Versuch, einen Index zu cachen: In der aktuellen 19c wird zwar behauptet, dass der Index gecacht sei, tatsächlich wird jedoch laut SQL*Plus Autotrace permanent von der Platte gelesen. Ohnehin gelingt es nur, einen Index in den Big Table Cache zu bekommen, wenn auch mit Parallel Query darauf zugegriffen wird, obwohl bei Single Instanz ja ein nicht paralleles Query genügen sollte.

Ein ebenfalls sehr verwirrendes Bild erhält man im Fall von Partitionierung – hier bringt Oracle das Kunststück fertig, gelegentlich als Größe der Partition die Summe der Blöcke aller im Big Table Cache auftauchenden Partitionen derselben Tabelle anzugeben *(siehe unten)*.

Auch die Angabe CACHED_IN_MEM muss mit Vorsicht verwendet werden – wenn durch Verdrängung ein Objekt laut

```
select
 set_id,
 Pool_id,
 blk_size,
 cnum_set,
 cnum_dwb,
 anum_dwb,
 round(cnum_dwb/cnum_set,2) alloc
 from x$kcbwds
 where cnum_dwb >0;
```

*Listing 5: Tatsächliche Reservierung von Buffern über die verschiedenen Blockgrößen*

Policy von DISK gelesen wird, ist ein Wert für CACHED_IN_MEM>0 keine Seltenheit.

## Multiple Blockgrößen

Wie Jonathan Lewis oft sinngemäß schreibt: Die beste Methode, um die Grenzen einer Funktionalität (nicht nur in Oracle) zu finden, ist, die Funktion mit anderen, eher selten genutzten Optionen zu vermischen. Daher war es naheliegend, dem ABTC nachzuspüren, was beim Einsatz mehrerer Blockgrößen in der gleichen Datenbank geschieht. Um es kurz zu machen – der Cache funktioniert dann nicht mehr zuverlässig. Oracle hat entweder schlicht übersehen oder bewusst ignoriert, dass es mehr als eine Blockgröße in einer Datenbank geben kann. Die X$-Tabellen, auf denen die V$-Views aufbauen, haben keinen Bezug zur Blockgröße des Tablespace, in dem ein Segment liegt. Das führt zu der kuriosen Situation, dass der Big Table Cache real für jede konfigurierte Blockgröße als Anteil des jeweiligen Cache (Parameter db_2k_cache_size.. db_32k_cache_size) eingerichtet wird, für die Platzierung laut v$bt_scan_obj_temps aber so getan wird, als ob alles im gleichen Pool liegen würde. Als Folge kann eine Tabelle in einem 16k-Tablespace verhindern, dass eine andere aus einem 8k-Tablespace in den Cache geladen wird, obwohl für beide Platz wäre!

Dass die dynamische Anpassung von ABTC auch nur für die Default-Blocksize durchgeführt wird und alle anderen einen Neustart benötigen, verwundert an der Stelle nicht weiter. Dem interessierten Leser sei hier die Tabelle X$KCBWDS für eigene Studien nahe-gelegt, dabei ist cnum_set die Anzahl der Buffer im Working Data Set, cnum_dwb sind die für den ABTC reservierten Buffer. Ein Startpunkt für eigene Studien mag das Statement in *Listing 5* sein.

Zusammenfassend kann man sagen, dass der Automatic Big Table Cache über die gleichen grundlegenden Strukturen verwaltet wird wie der normale Buffer Cache und dadurch natürlich ebenfalls blockorientiert organisiert ist; zusätzlich wurde mit der Segmentsicht eine Abstraktionsschicht eingezogen. Allerdings fehlen dieser Abstraktionsschicht wesentliche Informationen wie die Blockgröße, dadurch ist die Verwendung mehrerer Blockgrößen in Verbindung mit dem ABTC nicht nur nicht sinnvoll, sondern es muss stark davon abgeraten werden.

```
drop table mtab purge;
create table mtab (pcol number, padding varchar2(50))
partition by list (pcol) (
   partition p1 values (1),
   partition p2 values (2)
);

insert into mtab
with rowsource as (select null from dual connect by level <=1000)
select mod(rownum,2)+1 , lpad(rownum,45,'*') from rowsource,rowsource;
commit;
execute dbms_stats.gather_table_stats(user,'mtab',granularity=>'ALL')
select table_name, partition_name, blocks, num_rows from user_tab_partitions;
select min(pcol),max(pcol)from mtab;
@scan_obj

select count(*) from mtab where pcol=2;
@scan_obj
prompt now adding partition
PAUSE
alter table mtab add partition p3 values (3);
insert into mtab select 3,padding from mtab where pcol=1;

commit;
execute dbms_stats.gather_table_stats(user,'mtab',granularity=>'ALL')
select table_name, partition_name, blocks,num_rows from user_tab_parti-
tions;
select min(pcol),max(pcol)from mtab;
@scan_obj
```

*Listing 6: Demo-Skript Partitionierung*

## Partitionierung

Da der ABTC nicht auf Tabellen beschränkt ist, soll auch ein kurzer Blick auf den Einsatz bei Partitionierung geworfen werden. Auch hier kann der ABTC verwendet werden, wie im folgenden Beispiel demonstriert werden soll. Wir erstellen eine Tabelle mit 2 Partitionen und jeweils 500.000 Zeilen pro Partition, was in einer Partitionsgröße von ca. 4030 Blöcken resultiert. Nach dem initialen Befüllen der Partitionen werden die Statistiken aktualisiert, eine dritte Partition angehängt und wieder Statistiken ermittelt.

Am Ende sehen wir, dass Oracle laut v$bt_scan_obj_temps das Kunststück fer-

```
TABLE_NAME                     PARTITION_NAME     BLOCKS   NUM_ROWS
------------------------------ -------------- ---------- ----------
MTAB                           P1                   4030     500000
MTAB                           P2                   4030     500000
MTAB                           P3                   4030     500000

 MIN(PCOL)  MAX(PCOL)
---------- ----------
         1          3

  DATAOBJ# OBJECT_TYPE             OBJECT_NAME             SIZE_IN_BLKS TEMPERATURE POLICY     CACHED_IN_MEM
---------- ----------------------- ----------------------- ------------ ----------- --------- -------------
    108244 TABLE PARTITION         MTAB.P1                        12090        9000 MEM_ONLY           8060
    108245 TABLE PARTITION         MTAB.P2                        12090        8000 MEM_ONLY           4030
    108246 TABLE PARTITION         MTAB.P3                        12090        4000 MEM_ONLY           4030
```

*Abbildung 2: Gekürzte Ausgabe der Demo zur Partitionierung (Quelle: Holger Bär)*

```
--scan_obj.sql
set autotrace off
set lines 300
col owner for a12
col object_name for a34
col tbs for a14
select
      owner,
      (select name from v$tablespace ts where ts.ts#=bso.ts#)tbs,
      dataobj#,
      object_type,
      object_name || case when subobject_name is not null then '.'||su-
bobject_name end object_name,
      Size_in_blks,
      Temperature,
      Policy,
      Cached_in_mem from dba_objects do right join v$bt_scan_obj_temps
bso on (data_object_id = dataobj#)
order by temperature desc
/
```

*Listing 7: Hilfsskript zur Anzeige der gecachten Objekte*

tigbringt, für jede Partition in der Spalte SIZE_IN_BLKS die Summe aller Partitionen anzugeben, wohingegen CACHED_IN_MEM die Summe von 2 Partitionen(!) ist. Greift man nun gezielt auf eine einzelne Partition zu, so werden die Werte zwar korrigiert, dies erschwert jedoch das Überwachen des Cache ungemein, da man leider den Angaben der View für partitionierte Tabellen nicht glauben kann. Dass im gezeigten Beispiel der Auslöser für die fehlerhafte Angaben das Ermitteln der Statistiken ist, beruhigt in dem Zusammenhang nur wenig.

In der Ausgabe in *Abbildung 2* sehen wir, dass SIZE_IN_BLKS drei Mal so groß ist, wie die jeweilige Partition Blöcke hat. Das Skript scan_obj.sql ist in *Listing 7* wiedergegeben.

Operationen wie Partition Exchange oder Partition Move führen zu einer neuen Data_object_id und damit ebenfalls zu den oben genannten Problemen. Allerdings kann man bei Partition Exchange von dem Verhalten eventuell sogar profitieren: Lädt man zunächst die Daten in eine Tabelle und bearbeitet diese dann in mehreren Schritten, so wird – vorausgesetzt die Tabelle ist mittlerweile im ABTC geladen – nach dem Partition Exchange die Partition direkt im Cache vorgehalten, da ja tatsächlich nur das Data Dictionary verändert wurde, die Data_object_id jedoch erhalten bleibt.

## BUGs

Neben den bereits dargestellten Problemen gab und gibt es auch im Oracle Support einige Bugs dokumentiert.

- Bug 21514877:
  - BT_CACHE_TARGET Is Re-adjusted To 0 When A DataPump Job is Started (Doc ID 2060627.1)
  - Fixed mit Patch 2154877 / Upgrade 12.2 / PSU JUL2017 (1707)
- Bug 19800418:
  - Ora-00600[kcblrs_2] wenn dynamic_sampling aktiv ist
  - Fixed mit Patch 19800418 / Upgrade 12.2
- Doc ID 2471008.1
  - Wrong Result Sometimes Occur When Set DB_BIG_TABLE_CACHE_PERCENT_TARGET Parameter
  - Angeblich gefixt mit 12.2 bzw. Patch 20214104

Gerade der letzte Bug kann selbst mit Version 19c immer noch nachgestellt werden – allerdings erst, wenn der ABTC bereits in Benutzung war. Ein ähnliches Phänomen lässt sich auch beobachten, wenn man das in *Listing 6* aufgeführte Skript mehrfach hintereinander ablaufen lässt. Früher oder später kann man beobachten, dass anstelle von jeweils 50000 Zeilen pro Partition eine geringere Anzahl von Zeilen in der letzten Partition generiert werden.

## Zusammenfassung

Der ABTC passt in den aktuellen Trend von Oracle, alles „Autonomous“ haben zu wollen. Leider scheinen bei der Implementierung aber grundlegende Versäumnisse entstanden zu sein, anders sind die aufgeführten Probleme und Bugs nicht zu erklären. Da bei fehlerhaften Daten der Spaß aufhört, muss gerade beim eigentlichen Anwendungszweck – dem Bearbeiten großer Datenmengen im Batchbetrieb – von der Verwendung dringend abgeraten werden, vor allem wenn auch noch Partitionierung ins Spiel kommt.

Anwendungen, die eher im OLTP-Bereich anzusiedeln sind und mit hohen Direct Reads zu kämpfen haben, können hingegen profitieren. Hier ist nur ein passendes Monitoring des Cache zu empfehlen, um eine Unterdimensionierung zu verhindern.

## Quellen


[1] Roger Macnicol (2017): *https://blogs.oracle.com/smartscan-deep-dive/when-bloggers-get-it-wrong-part-1*, *https://blogs.oracle.com/smartscan-deep-dive/when-bloggers-get-it-wrong-part-2*

[2] Jonathan Lewis (2011): Oracle Core: Essential Internals for DBAs and Developers, Apress


## Über den Autor

Holger Bär ist als Senior Database Architect bei der science + computing AG angestellt, einer 100%igen Tochter der Atos Deutschland. Vor etwa 20 Jahren hatte er erste Kontakte mit der Oracle-Datenbank, die ihn seither nicht mehr losgelassen hat.


Holger Bär
holger.baer@atos.net

# DOAG Botschafter für Technologie 2019: Johannes Ahrends

Martin Meyer, Redaktionsleitung

Für sein großes Engagement innerhalb der DOAG wurde Johannes Ahrends, Themenverantwortlicher für Datenbankadministration und Standard Edition, auf der DOAG 2019 Konferenz + Ausstellung zum Botschafter für Technologie gekürt. Diese Auszeichnung, die ihm Christian Trieb, DOAG-Vorstand Datenbank und Leiter der Datenbank Community und des Competence Center Support, überreichte, hat er sich verdient.

Johannes Ahrends ist bei der DOAG Themenverantwortlicher für die Oracle Datenbank Administration sowie die Standard Edition.

Regelmäßig hält er Vorträge auf DOAG-Veranstaltungen und bietet Workshops und Schulungen zu unterschiedlichen Themen an.

Johannes Ahrends beschäftigt sich schon seit 1992 mit Oracle-Datenbanken. Zuvor widmete er sich intensiv dem Unix-Betriebssystem, so dass diese Kombination heute sein absoluter Favorit ist.

Mit Vorliebe berät er Unternehmen dabei, wie sie bei der Nutzung der Oracle Database Standard Edition Kosten sparen können.

Nach Stationen bei Oracle, Herrmann & Lenz Services und Quest Software gründete er im Jahre 2011 sein eigenes Unternehmen, die CarajanDB GmbH. Im selben Jahr wurde er von Oracle für sein Engagement mit dem ACE Award ausgezeichnet.

Seine Expertise umfasst die Administration und den Betrieb der Oracle Standard Edition Edition, komplexe Lösungen mit Real Application Cluster, Dataguard und GoldenGate für die Oracle-Hochverfügbarkeit, Migrationen und Upgrades, Backup und Recovery sowie Performance-Optimierung.

Als Autor mehrerer Bücher, unter anderem „Oracle 11g Release 2 für den DBA“, hat er in den vergangenen Jahren sein Wissen zu Papier gebracht. Mittlerweile hat er selbst auf das „papierlose Büro“ umgestellt und bloggt stattdessen regelmäßig.

Momentan beschäftigt er sich sehr intensiv mit der Multitenant Database Option sowie Cloning-Technologien.

Johannes Ahrends

# Wir begrüßen unsere neuen Mitglieder

## Persönliche Mitglieder

- Ingo Kallenbach
- Clemens Dudek
- Josua König
- Wen-Jane Yang
- Reza Shafiei
- Rebecca Schlecht
- Thierry Gascard
- Emmanuel Kwamena Asabir
- Daniel Dymala
- Ulrich Seifert

## Firmenmitglieder DOAG

- Degussa Bank AG, Daniel Ruppert

# Termine


*16.01.2020*
**Regionaltreffen NRW**
Martin Schmitter, Torsten Rosenwald
Köln

*23.01.2020*
**DOAG 2020 NOON2NOON**
München


*06.02.2020*
**Regionaltreffen Rhein-Neckar**
Frank Stöcker
Mannheim

*28.02.2020*
**Regionaltreffen Bremen**
Ralf Kölling
Bremen

*12. - 13.02.2020*
**Berliner Expertenseminar zum Thema Gestaltung moderner User Interfaces mit APEX 19.2**
Moritz Klein, Carsten Czarski
Berlin

*20.02.2020*
**DOAG Forms Day 2020**
Jan-Peter Timmermann
Frankfurt


*04.03.2020*
**Regionaltreffen Berlin/Brandenburg**
Michel Keemers
Berlin

*12.03.2020*
**Regionaltreffen NRW**
Martin Schmitter, Torsten Rosenwald
Köln

*17.- 19.03.2020*
**JavaLand 2020**
in Brühl bei Köln

*19.03.2020*
**CloudLand Preevent**
in Brühl bei Köln

*31.03.2020*
**Berliner Expertenseminar zum Thema Oracle Performance Diagnostics and Tuning**
Moritz Klein, Carsten Czarski
Berlin

## Impressum

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

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

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

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

### Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
ViSdP: Mylène Diacquenod
Redaktionsleitung: Martin Meyer
Kontakt: redaktion@doag.org
Weitere Redakteure (in alphabetischer Reihenfolge): Niels de Bruijn, Carsten Czarski, Lisa Damerow, Mylène Diacquenod, Dietmar Neugebauer Fried Saacke, Frank Schneede

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

### Fotonachweis:
Titel: © Galina Peshkova | *https://de.123rf.com*
S. 7: © Udo Schotten | *https://de.123rf.com*
S. 9: © maxicam | *https://de.123rf.com*
S. 14: © Jozef Mičic | *https://de.123rf.com*
*S. 17: © Евгений Герасимов* | *htt ps://de.123rf.com*
S. 23: © Jess Sanz | *htt ps://de.123rf.com*
S. 28: © aurielaki | *https://de.123rf.com*
S. 36: © Samantha Craddock | *https://de.123rf.com*
S. 41: © Dzianis Rakhuba | *https://de.fotolia.com*
S. 47: © liravega | *https://de.123rf.com*
S. 59: © buchachon | *https://de.123rf.com*
S. 72: © DOAG | *https://www.doag.org*
S. 73: Hintergrund: © Dzianis Kuryanovich | *https://de.123rf.com*

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

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



## Inserentenverzeichnis

B4BMEDIA.NET AG *www.e3zine.com* **U 3**
dbi services sa *www.dbi-services.com* **S. 27**
DOAG e.V. *www.doag.org* **U 2**
MuniQsoft Consulting GmbH *www.muniqsoft-consulting.de* **S. 19**
MuniQsoft Training GmbH *www.muniqsoft-training.de* **S. 3**
Robotron Datenbank-Software GmbH *www.robotron.de* **S. 43**
Trivadis AG *www.trivadis.com* **U 4**

Das E-3 Magazin

Information und Bildungsarbeit von und für die SAP-Community

# Überfordert?

## Wir bieten Information und Bildungsarbeit von und für die SAP-Community





e-3.de | e3zine.com

# WIR LEBEN DIGITALISIERUNG.


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

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

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

trivadis