Nr. 5 | September 2022 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP

inklusive BUSINESS NEWS

Engineered Systems

Aus der Praxis

11g- und 12c-Migrationen nach 19c Multitenant

Im Interview

Interview mit Frank Weise und Jonas Matthes, make IT

Business News

Das NetSuite-Phänomen


DOAG 2022
Konferenz + Ausstellung
In Nürnberg
20.-23. SEPT.
Die Oracle-
ANWENDERKONFERENZ
anwenderkonferenz.doag.org
Eventpartner:
AOUG
SOUG
iJUG Verbund

Jessica Steger
Vorstand Infrastruktur und Middleware, Leiterin Infrastruktur & Middleware Community

Liebe Mitglieder, liebe Leserinnen und Leser,

die aktuelle Ausgabe des Red Stack Magazin hat Oracle Engineered Systems als Schwerpunkt, die seit Jahren eine optimierte Umgebung für Oracle-Datenbanken bieten.

Der zentrale Mehrwert dieser Systeme ist, dass man eine Komplettlösung für die Datenbanken inklusive Datenbank-Servern und Storage erhält, die über Administrationstools einfach gepatcht und monitort werden können und die vom gleichen Hersteller wie die Datenbank-Software kommen. Somit können Funktionsweisen der Datenbanken mithilfe der Engineered Systems über den ganzen Stack bis herunter auf die Hardware-Ebene optimiert werden.

Im Interview mit der make IT GmbH werden deren langjährigen Erfahrungen mit der Oracle Database Appliance (ODA) dargestellt. Günther Stürner erzählt in seinem Kommentar etwas zur Geschichte der Exadata, Manfred Drozd berichtet von der Performance der neusten Exadata-Generation X9M und Clemens Bleile teilt Wissen aus einem Migrationsprojekt von einer Solaris-Umgebung auf Exadata Cloud@Customer.

Jedes Engineered System hat seine Alleinstellungsmerkmale, die bei einer neuen Architektur-Lösung für Oracle-Datenbanken berücksichtigt werden sollten. Die vier Artikel geben einen guten Einblick und zeigen deren Vorteile für den Einsatz mit Oracle-Datenbanken in den Bereichen Administration, Performance und Sicherheit auf.

Für die Oracle-Datenbank-Interessierten gibt es weitere Artikel über den Einsatz von Datenbank-Triggern, die Datenbank-Migration nach 19c Multitenant sowie neue Features von APEX und der Datenbank-Version 21c.

Im Business-News-Bereich des Magazins wird auf das NetSuite-Phänomen und auf das komplexe Thema der Compliance im internationalen Umfeld eingegangen.

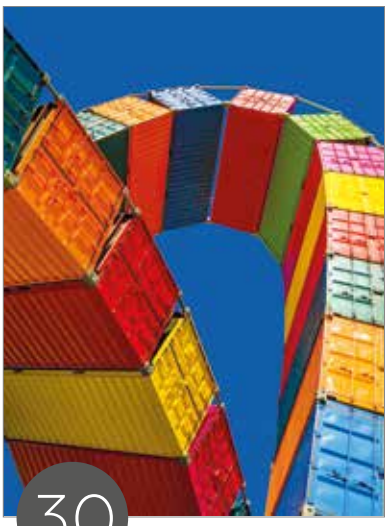
Wir freuen uns alle, dass in diesem Jahr die DOAG Konferenz und Ausstellung wieder in Präsenz stattfinden wird. Dieses Jahr gibt es einen Thementag am 20.09. und die beiden Konferenztage am 21. und 22.09. mit vielen interessanten Vorträgen und dem Engineered-Systems-Experten-Panel am 21.09. um 17:00 Uhr, bei dem Sie dem Oracle Product Management Ihre Fragen stellen können.

Bleiben Sie gesund.

Jessica Steger

Ausgabe Nr. 5/2022
auf Abruf!

Exadata Cloud@ Customer (ExaCC)



Container Only – Multitenant in Oracle 21c

Einleitung

Engineered Systems

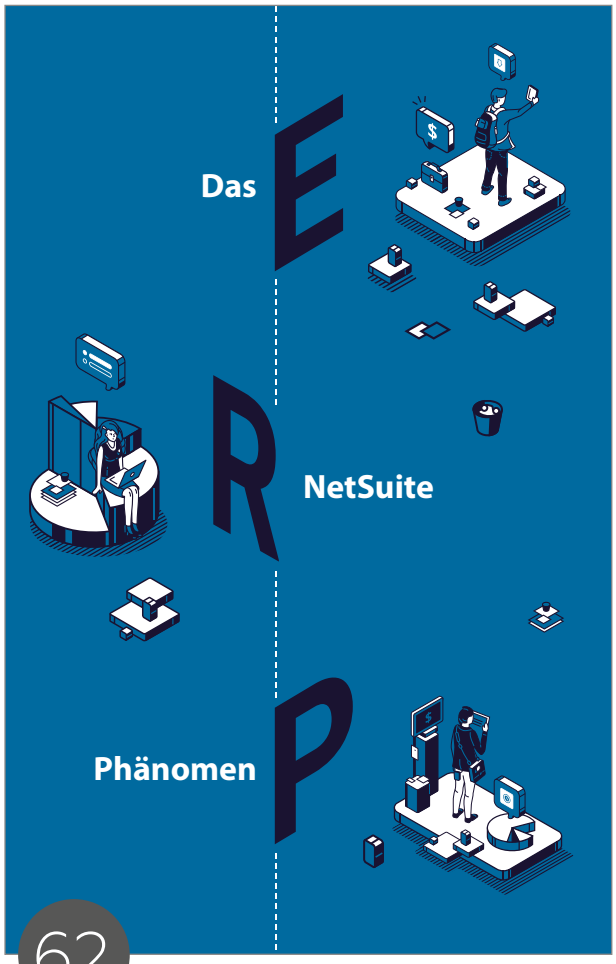
PL/SQL

Datenbank

APEX

BUSINESS NEWS

Das NetSuite-Phänomen



Leitartikel | Das NetSuite-Phänomen

Viele Wege führen nach Rom –
11g- und 12c-Migrationen nach 19c Multitenant

APEX 22.1 –
Hidden Gems

ACT GLOBAL, BE LOCAL?
Deutsche Compliance im
internationalen Umfeld

Intern

News

TIMELINE

8. Juli 2022

Das Programm der European NetSuite User Days, die dieses Jahr vom 19. bis 20. September erstmals auch im Nürnberger NCC Ost stattfinden, wird veröffentlicht. Die Besucher erwarten viele interessante Kundenberichte mit anschaulichen Beispielen aus der Praxis, Informationen aus erster Hand direkt vom NetSuite-Produktmanagement und viele Möglichkeiten zum Erfahrungsaustausch. Erfahrene Experten zeigen den Teilnehmern, wie Sie das Beste aus Ihrem NetSuite-System herausholen können. NetSuite-Manager werden Fragen direkt beantworten.

12. Juli 2022

Zum Neustart der Regio Dresden gibt es in Dresden eine lokkere Fachdiskussion zur Oracle-Datenbank-Anwendung.

12. August.2022

In der WebSession mit Stefan Seck erfahren die Teilnehmer alles rund um das Thema „Backup und Data Guard mit odacli".

29. August 2022

Die Themen des Regionaltreffens der Regio Hamburg sind „Cross-Plattform-Migration: von Windows auf Linux mit wenig bis gar keiner Downtime" inklusive einer Live-Demo. Es referiert Benjamin Kurschies von Lufthansa Industry Solutions AS. Im Anschluss an den Vortrag können die Teilnehmer beim Networking ihre persönliche Erfahrungen austauschen.

6. bis 7. September 2022

Im Berliner Expertenseminar „Flows for APEX – prozessorientierte Apps erstellen" zeigen Niels de Bruijn und Moritz Klein, wie sich mit „Flows for APEX" verschiedene Geschäftsprozesse modellieren lassen und welche Aspekte man dabei beachten sollte.

9. September 2022

In der 120-minütigen WebSession mit Amin Farvardin erfahren die Teilnehmer alles rund um das Thema „Oracle Database 21c New Features für den DBA". Erörtert wird insbesondere auch, wie verlässlich diese sind.

JavaLand

21. – 23. MÄRZ 2023

im Phantasialand bei Köln

www.javaland.eu

Präsentiert von: DOAG Heise Medien

Community Partner: iJUG Verbund

„Die Performance und die Zusammenlegung von Storage-, Betriebssystem- und Datenbank-Administration haben uns überzeugt."

Die Oracle Database Appliance stellt in vielen mittelständischen Unternehmen, aber auch in Konzernen, seit 2011 eine wichtige Plattform für den Betrieb von Oracle-Datenbanken dar. Marco Friebe, Themenverantwortlicher Oracle Database Appliance bei der DOAG Infrastruktur und Middleware Community, sprach mit Frank Weise und Jonas Matthes, beide DBA bei dem Mittelständler make IT GmbH.

Bitte stellen Sie sich kurz unseren Lesern vor. Wer sind Sie und was ist Ihre Rolle bei der make IT?

Mein Name ist Frank Weise, 46 Jahre alt und begeisterter Hobbyfotograf. Ich bin seit 1992 im IT-Umfeld tätig, zunächst als Serveradministrator und in der Anwendungsbetreuung. Als ich später bei eigenen Net-Anwendungsentwicklungen immer wieder auf Datenbankproblematiken gestoßen bin, habe ich mein Serverwissen genutzt und mich dann wieder mehr den Datenbankservern und den Datenbankinfrastrukturen gewidmet. Dies seit 2012 speziell mit Oracle-Datenbankservern. Diese Server betreibt die make IT GmbH hauptsächlich für das Energieunternehmen „eins energie in Sachsen GmbH & Co. KG" und den Netzbetreiber „inetz GmbH" sowie für öffentliche Verkehrsunternehmen in der Region Chemnitz.

Mein Kollege, Jonas Matthes, 55 Jahre alt, ist ebenso seit 1992 im IT-Umfeld unterwegs, anfangs als Server- und Netzwerkadministrator, später in den Thematiken Oracle-Datenbanken, Oracle-Anwendungsentwicklung, Geoinformationssysteme und Datenschutz aktiv.

Welche Bedeutung nimmt die Oracle-Datenbank und damit deren Betriebsumgebung in Ihrem Unternehmen ein?

Oracle nimmt im Unternehmen derzeit Platz 2 unter den Datenbanksystemen ein. In größerem Umfang werden nur die SAP-Datenbanken, die „leider" unter DB2 auf dem IBM-System i laufen, eingesetzt. Mit Oracle werden bei uns fast alle neben SAP stehenden Geschäftsprozesse abgebildet. So zum Beispiel die Kraftwerksverwaltung, der Stromhandel, die Netz-Betriebsmittelverwaltung, die Netz-Störungsverwaltung, CRM-Systeme und noch vieles mehr.

Welche Anforderungen und Kriterien haben zur Entscheidung des Einsatzes von ODA geführt?

Bis 2014 waren die vorhandenen Oracle-Server meist dedizierte Hardware-Server mit Windows-Server-Betriebssystemen. Davon gab es natürlich mehrere mit verschiedensten Oracle-DB-Versionen. Auch die Probleme waren in dieser Umgebung vielfältig. Die größten Probleme lagen in der Performance der Datenbanken, dicht gefolgt von Speicherknappheit und ewig langen Backup- und Restorezeiten. Als dann der Intraday-Stromhandel immer wichtiger wurde, zeigte sich, dass die vorhandene Infrastruktur dieser Herausforderung nicht mehr gewachsen war. Ein Versuch, die langsamen Datenträger durch schnellere Storagesysteme zu ersetzen, brachte zwar Besserung, war aber aufgrund der Aufgabentrennung Storageadmin, Serveradmin und Datenbankadmin nicht immer flüssig in Einklang zu bringen. Einer der DOAG-Konferenz-Besuche zeigte uns dann die Lösung der Performanceprobleme, die EXADATA. Jedoch ergab sich nach genauerer Recherche, die wir gemeinsam mit unserem langjährigen Oracle-Partner aus Dresden durchführten, dass die Lösung bereits über das ODA-System erreicht werden konnte. Dank der

guten Zusammenarbeit von Oracle, unserem Partner und der make IT konnten wir im Jahr 2014 ein sehr erfolgreiches Projekt „Proof of ODA Concept" durchführen, damals noch auf der ODA X3-2. Die Verarbeitungszeiten der nächtlichen Jobs waren um zwei Drittel schneller im Vergleich zur bisher benötigen Zeit.

Durch den Einsatz der ODA und die damit verbundene Konsolidierung vieler Datenbanken auf einer spezialisierten Appliance-Hardware konnten wir die Anzahl unserer Datenbank-Server und damit natürlich auch den Verwaltungsaufwand reduzieren.

Die Performance und die Zusammenlegung von Storage-, Betriebssystem- und Datenbank-Administration haben uns überzeugt.

Welche Vorteile konnten Sie mit dem ODA-Betriebsmodell erzielen und welche Nachteile haben Sie sich erkauft?

Die Vorteile liegen hier klar auf der Hand. Trotz der Vielfalt der bei uns eingesetzten ODA-Systeme ist die Administration immer gleich. Ein Skript, das eine Datenbank auf der X7s anlegt (oder updatet), funktioniert identisch auf der X8-2HA. Ebenso ist das bei den Appliance Updates. Ein Updatepaket funktioniert auf allen Systemen gleich (also meistens ;-). Das ist bei ständig steigender Datenbankanzahl und steigenden Sicherheitsanforderungen ein sehr großer Vorteil.

Als Nachteil sehen wir im Moment nur, dass DB- und OS-Patches für die ODA etwas später erscheinen als für die anderen Plattformen. Das ist speziell bei plötzlich durch die Medien getriebenen kritischen Sicherheitslücken ungünstig. (Anmerkung der Redaktion: Diese Anforderung wird seit ODA Version 18.5 bzw. 18.7 durch „Out-of-Cycle Database Patches" bzw. „Out-of-Cycle OS Kernel Patching" abgedeckt.)

Können Sie bitte kurz Ihre ODA- und Datenbank-Landschaft beschreiben?

Wir betreiben mittlerweile neun ODAs in verschiedensten Ausführungen von X5 bis X8, von „S" bis „HA", sowohl als Enterprise- als auch als Standard-Edition. Diese ODAs befinden sich in verschiedenen Netzwerkzonen mit unterschiedlichem Schutzbedarf und verschiedenen Verfügbarkeitsanforderungen. Im Produktivbetrieb sind ca. 50 Datenbanken im Bereich Energiewirtschaft und öffentlicher Personen-Nahverkehr im Einsatz.

Wie ist Ihr (DBA)-Team organisatorisch aufgestellt und wie „meistert" es den Komplettbetrieb der Engineered Systems bestehend aus Hardware, Netzwerk, Betriebssystem, KVM, Gridinfrastruktur, ASM und Datenbank?

Wir sind ein Zweier-DBA-Team und teilen uns die Aufgaben nahezu zu gleichen Teilen. Ich, Frank Weise, mit den Hauptaufgaben der Hardware-, Netzwerk- und Betriebssystem-Betreuung und mein Kollege Jonas Matthes mit der Betreuung der Datenbanken, Gridinfrastruktur und ASM. Konfigurationsänderungen werden von uns als Skript dokumentiert, so ist der jeweils andere im Vertretungsfall immer in der Lage, auch die Aufgaben des Kollegen zu übernehmen. Diese Vorgehensweise hat sich zusammen mit dem ODA-Einsatz sehr gut bewährt. KVM im Zusammenhang mit ODA ist momentan noch kein Thema bei uns.

Was wünschen Sie sich für die Zukunft von Oracle und des ODA-Entwicklungs-Teams?

Der größte Wunsch ist wohl, dass das ODA-Konzept bei Oracle weiter Bestand hat und weiter (ggf. auch schneller) mit Sicherheits- und Feature-Updates gefüttert wird. Damit der Mittelstand auch weiterhin eine Alternative zur Cloud und zur wesentlich teureren EXADATA hat. Auch in der Weiterentwicklung der Mischlösung ODA/Cloud sehen wir Potenzial.

Als Softwareentwickler fänden wir den Ausbau der REST-APIs mit ASM-Optionen sehr schön.

Was ist Ihr Resümee? Werden Sie weiterhin auf Engineered Systems setzen und Ihren ODA-Bestand erneuern beziehungsweise diesen für weitere Projekte ausbauen?

Mit der Umstellung auf ODAs haben wir die Oracle-Datenbank-Umgebung deutlich performanter, sicherer und flexibler gemacht. Es war genau die richtige Entscheidung zum genau richtigen Zeitpunkt. Natürlich wird es hinsichtlich „Cloud" in Zukunft immer schwerer, ein On-Premises-System wie die ODA zu rechtfertigen. Aber dennoch werden wir auch in Zukunft Oracle-Datenbankprojekte mit ODAs umsetzen. Auch bei der kontinuierlichen Erneuerung der Landschaft werden diese zum Einsatz kommen.

FIRMENPROFIL MAKE IT GMBH:

Die make IT GmbH stellt als Tochter der eins-Gruppe (ca. 1.200 Mitarbeiter), einem Energie-Dienstleister in Südwestsachsen, IT-Dienstleistungen für die gesamte Firmengruppe bereit. Die ca. 80 Mitarbeiter der make IT decken das gesamte Spektrum an benötigten Basis-IT-Dienstleistungen ab und hosten fachspezifische Anwendungen des eins-Konzerns sowie des Mitgesellschafters CVAG.

Neben SAP-Dienstleistungen und der Bereitstellung der Infrastrukturen werden durch die make IT auch Rechenzentrumsleistungen realisiert. Ein wesentliches Angebot des Rechenzentrums bildet die Verfügbarkeit von Oracle-Datenbank-Kapazität auf Oracle Database Appliances (ODA).

BERLINER EXPERTENSEMINARE

Die Berliner Expertenseminare sind Expertenschulungen und Weiterbildungen der DOAG, die mit einer Hands-On-Mentalität über zwei Tage geballtes Fachwissen mit praxisnahen Übungen vermitteln. Profis geben in kleiner Runde ihr großes Know-how weiter und sorgen für einen optimalen Wissenstransfer, der unmittelbar danach angewendet und in die täglichen Aufgaben und Herausforderungen fließen kann. Für ein exquisites Buffet ist während des gesamten Seminars ebenfalls gesorgt.

11.10. - 12.10.2022

Einführung in Docker als Basis für Entwicklung und Produktion

Berliner Expertenseminar mit Kai Donato und Philipp Hartenfeller

In diesem Seminar werden die Grundlagen für den geübten Umgang mit Docker geschaffen, dem populärsten Werkzeug zur Containerisierung – ein Standard, um Software isoliert und über verschiedene Infrastrukturen hinweg ohne Komptabilitätsprobleme auszuführen.

https://shop.doag.org/event/id.78.einfuhrung-in-docker-als-basis-fur-entwicklung-und-produktion/

25.10. - 26.10.2022

Professionelle PL/SQL Datenbankprogrammierung

Berliner Expertenseminar mit Jürgen Sieben

Das Seminar beleuchtet die Arbeitsweise der Datenbank aus Sicht eines Entwicklers und vermittelt die Kenntnisse, die zur Erstellung robuster, skalierbarer Anwendungen erforderlich sind.

https://shop.doag.org/event/id.82.professionelle-pl-sql-datenbankprogrammierung/

08.11. - 09.11.2022

Good APEX Practices for Enterprise Projects – incl. APEX 22.1

Berliner Expertenseminar mit Moritz Klein, Markus Dötsch und Carsten Czarski

Die Experten thematisieren die Punkte, die in einem größeren APEX-Projekt mit mehreren Applikationen und/oder Entwicklern wichtig sind. Als Bonus gibt es „APEX 22.1 und mehr: Neues aus dem Entwickler-Labor".

https://shop.doag.org/event/id.79.good-apex-practices-for-enterprise-projects-incl-apex-22-1/

22.11. - 23.11.2022

Oracle BI/Analytics Publisher als Reporting-Lösung für Forms- und APEX-Anwendungen

Berliner Expertenseminar mit Jürgen Menge

Die Schwerpunkte des Seminars sind die Infrastruktur sowie ausgewählte Aspekte der Entwicklung und des Betriebs von Publisher-Berichten inklusive der Integration der Berichte in eigene Anwendungen (Oracle Forms, APEX, etc.).

https://shop.doag.org/event/id.75.oracle-bi-analytics-publisher-als-reporting-losung-fur-forms-und-apex-anwendungen/

29.11. - 30.11.2022

Oracle PL/SQL Performance Tuning

Berliner Expertenseminar mit Jürgen Sieben

In diesem Seminar werden Programmierstrategien erarbeitet, die die Skalierbarkeit der Datenbank erhalten und bekannte Performanzprobleme umgehen. Hinzu kommt ein Set von Best Practices, die als Grundlage für eine performante Datenbankprogrammierung eingesetzt werden können

https://shop.doag.org/event/id.81.oracle-pl-sql-performance-tuning/

01.03. - 02.03.2023

Praxisworkshop Oracle Database Appliance

Berliner Expertenseminar mit Florian Barth

Im Seminar werden die grundlegende Architektur und die umfassenden Plug-and-Play-Features der ODA, als auch mögliche Stolpersteine vorgestellt. Hinzu kommen viele Tipps und Tricks für jede Phase des ODA Life Cycles.

https://shop.doag.org/event/id.84.praxisworkshop-oracle-database-appliance/

AUS DER FERNE BETRACHTET:

PLAN B, EIN GLÜCKSFALL

„Erstens kommt es anders und zweitens als man denkt". Diese Erkenntnis musste auch das Oracle-Projektteam akzeptieren, das ab 2004 an dem streng geheimen Projekt SAGE (Storage Appliance for Grid Environments) arbeitete.

Die geniale Idee einer Datenbank-Komponente, die auf Storage-Systemen unterschiedlicher Hersteller läuft und die Datenbankperformance in ungeahnte Höhen treiben sollte, funktionierte zwar perfekt, aber die damaligen Storage-Platzhirsche zeigten keinerlei Interesse an „Fremdsoftware" auf ihren Systemen.

Das war ein herber Rückschlag. Alles war bereit, die erste Version machte das, was sie sollte, die Performancesteigerung war real und unglaublich, und doch war die Einführung dieser Wundersoftware in weite Ferne gerückt. Vielleicht war es aber nur ein Kommunikationsproblem, dass keiner der Protagonisten aus der Storage-Fraktion anbeißen wollte. Alles war so furchtbar geheim, viele Dinge noch ungeklärt, von Fragen zu Regresspflicht und Support bis hin zum Preis. Es war irgendwie verständlich, dass keiner die Katze im Sack kaufen wollte.

Damit war der Plan A Geschichte. Ein guter Plan B musste her, wollte man dieses Projekt noch retten.

Ein Ausweg aus dem Dilemma schien eine integrale Software-Hardware-Lösung. Der Einstieg von Oracle ins Hardware-Geschäft wurde durch die Hilfe des Steigbügelhalters HP ermöglicht. Die Exadata-Datenbank-Maschine war geboren und damit die mit Abstand leistungsstärkste Umgebung für den Betrieb von Oracle-Datenbanken. Das galt 2008 und gilt auch heute noch.

Für Manfred Drozd von der Firma Peakmarks AG *(https://www.peakmarks.com/de)*, einer Schweizer Firma, die sich seit vielen Jahren mit der Vermessung von Oracle-Lastprofilen auf unterschiedlichen HW-Plattformen beschäftigt, ist die Exadata das Maß aller Dinge, wenn es um den Oracle-Datenbankbetrieb geht: „Exadata war in allen Tests deutlich leistungsfähiger", so die Aussage zu seinen kürzlich durchgeführten Messungen.

„Kunden sind immer noch beeindruckt, dass Exadata-Systeme ein oder zwei Wochen nach der Anlieferung für den produktiven Betrieb zur Verfügung stehen und vom ersten Tag an die volle Leistung bringen", schwärmt Manfred Drozd.

Wenn **Oracle-Datenbank, dann Exadata** – müsste der abgewandelte Brillen-Slogan heißen, denn es gibt keine bessere Kombination.

Exadata kann man in der Zwischenzeit in unterschiedlichsten Formen genießen. Die Form, mit der ich persönlich fast täglich arbeite, ist die kostengünstigste überhaupt. Mein Free-Cloud Account in Frankfurt treibt meine APEX-Bemühungen sowie einige APPs, die daraus entstanden sind, und **apex.oracle.com**, das zweites Standbein meines Apex-Hobbys, läuft ebenfalls auf der besten aller Oracle-Datenbankplattformen. Einfach einmal ein paar Exadata-spezifische Funktionen ausprobieren. Klappt prima. Exadata für 0.- €. Ich habe schon schlechtere Deals gemacht.

Exadata ist in der Oracle Cloud überall zu finden. Von den kostenlosen (free cloud) Services, die Entwickler oder Interessenten anlocken sollen, bis hin zu den großen, unternehmenswichtigen Systemen. Oracle-Datenbanken in der Oracle Cloud ist (fast) immer gleichbedeutend mit Betrieb durch Exadata!

Neben der klassischen Variante – ein Kunde kauft Exadata-HW, stellt sie ins eigene Rechenzentrum und betreibt die gesam-

te Infrastruktur selbst – gibt es seit einiger Zeit eine smarte Art, mit Exadata den Datenbankbetrieb zu optimieren, zu konsolidieren und zu beschleunigen, ohne dass die Daten den Firmencampus verlassen müssen.

Exadata Cloud@Customer bedeutet übersetzt so viel wie: die Oracle (Datenbank) Cloud im Rechenzentrum des Kunden auf Basis von Exadata als Cloud-Service mit all den Automatisierungsfunktionen wie Provisionierung, automatische Backups, Data-Guard auf Knopfdruck, up-scale, down-scale etc., und alles wird von Oracle betrieben und gemäß Cloud-Metrik abgerechnet.

Das neueste Angebot mit dem etwas sperrigen Namen „Dedicated Region Cloud@Customer" geht noch einen Schritt weiter. Auch hier ist die Exadata die Basis für alle Datenbanken und auch hier steht die Exadata, oder wenn nötig auch mehrere, im RZ des Kunden. Im Unterschied zu der oben beschriebenen Variante kommen jedoch noch weitere Server, Netzwerke etc. hinzu. Es wird eine vollständige Oracle-Cloud-Umgebung beim Kunden aufgebaut. Alles, was zum Beispiel im Frankfurter Oracle Cloud RZ verbaut und software-mäßig installiert ist – Hardware, Software, Netzwerke, etc. –, findet sich auch im Kunden-RZ. Eine vollständige Oracle-Cloud-Umgebung mit allen Services im eigenen Haus und für den eigenen Bedarf. Und auch hier gilt, es wird keine Hardware oder Software gekauft, sondern es wird via Cloud-Metrik abgerechnet.

Die Exadata als Datenbank-Plattform wird intensiv von Oracle selbst genutzt. Das ist die beste Voraussetzung für stetige und sinnvolle Weiterentwicklung, denn Fehlentwicklungen treffen Oracle am härtesten.

So gesehen war das Scheitern des ursprünglichen Plans ein Glücksfall für Oracle und für viele Oracle-Kunden.

Günther Stürner
E-Mail: *guenther.stuerner@dbms-publishing.de*

DESCRIPTION :

DDC : DIRECT DIGITAL CONTROLLER
DB : DISTRIBUTION BOARD (LIGHTING PANEL)
S/S : START/STOP
ST : STATUS

Erfahrungsbericht Exadata X9M

Manfred Drozd, peakmarks

Anhand von drei Kundenprojekten möchten wir unsere Erfahrungen mit der Exadata Generation X9M beschreiben. Dieser Erfahrungsbericht wurde auf der DOAG-Datenbank-Konferenz 2022 präsentiert.

Intel Xeon Scalable Processor der 3. Generation

Die Exadata X9M-2 verwendet die neueste Intel-Xeon-Prozessorgeneration. Dabei wurde ein Wechsel von PCI Gen 3 auf PCI Gen 4 vorgenommen. Das bedeutet eine Verdopplung bei der internen Datenübertragung und wirkt sich bei den Datenbank-Servern beim Hauptspeicherzugriff und bei den Storage-Servern auf die I/O-Leistung äußerst positiv aus. Kunden älterer Exadata-Systeme stehen nun vor der Migration auf neue Systeme und hätten gerne Angaben für eine zuverlässige Kapazitätsplanung der Datenbank-Server.

Remote Direct Memory Access over Converged Ethernet (RoCE)

RoCE wurde bereits mit der Exadata X8M eingeführt. Bei Tests Anfang 2020 haben wir erhebliche Stabilitätsprobleme (Operating System panics, ORA-600) festgestellt, die erst im Laufe des Sommers 2020 gefixt wurden. Wir haben darum unseren 24-stündigen Stresstest auf der Exadata X9M wiederholt.

Persistant Memory

Auch die Persistant-Memory-Technologie wurde bereits mit der Exadata-X8M-Generation eingeführt. Sie wird in den Storage-Servern eingesetzt und dient sowohl als zusätzlicher Cache für Daten als auch als ein persistenter Puffer für RDO-Schreiboperationen. Corona-bedingt hatten wir aber erst jetzt Gelegenheit, diese Technologie im Detail zu untersuchen und die Versprechen des Oracle-Marketings zu überprüfen.

Exadata X9M Cloud@ Customer

Auch Oracle-Datenbanken werden in Zukunft vermehrt in der Cloud betrieben. Mit dem Cloud@Customer-Konzept ermöglicht Oracle als einziger Cloud-Anbieter, die Hardware im eigenen Data Center aufzustellen, aber die Infrastruktur von Oracle-Cloud-Mitarbeitern betreiben zu lassen. Nun stellt sich die Frage nach der Leistungsfähigkeit dieser Lösung im Vergleich zu Cloud-Angeboten der Mitbewerber.

Vorgehensweise

Bei Kundenanfragen zur Stabilität und Effizienz neuer Hardware-Technologien von Oracle-Plattformen, bei Preis-Leistungs-Vergleichen verschiedener Plattform-Anbieter (sei es On-Premises oder in der Cloud) und bei Fragen zur Kapazitätsplanung verwenden wir die peakmarks® Benchmark Software. Die peakmarks® Benchmark Software ist mit über 30 Workloads in 8 Themengruppen die umfassendste Benchmark-Software für Oracle-Plattformen, liefert eine Reihe von verständlichen Performance-Kennzahlen für repräsentative Datenbank-Operationen und ermöglicht so fundierte und faktenbasierte Entscheidungen *(siehe Abbildung 1)*.

Zur Leistungsfähigkeit der X9M-Datenbank-Server

Die Leistungsfähigkeit von Server-Systemen im Oracle-Datenbankbetrieb wird mit Workloads gemessen, die verschiedene SQL-Abfragen auf Daten im Oracle Buffer Cache ausführen. Es gibt also keine I/O-Operationen. *Abbildung 2* zeigt das Leistungsverhalten eines einzelnen Datenbank-Servers beim Workload SRV-QUERY1. Dieser Workload greift über einen Unique Index auf einen Datensatz zu und wird millionenfach in jeder Applikation ausgeführt, wie zum Beispiel Selektiere Kunde, Produkt, Bestellung etc.

Bei diesem Workload hat sich der Durchsatz von X5 auf X9 verdreifacht und die Antwortzeit der Queries nahezu halbiert. Die Durchsatzsteigerung ist einerseits auf die Leistungssteigerung pro Core und andererseits auf den Zuwachs an Cores zurückzuführen.

Zur Leistungsfähigkeit der X9M-Storage-Server

Die Leistungsfähigkeit der Storage-Server wird auf verschiedene Art und Weise gemessen:

- Durchsatz beim sequential read in [MBps]
- Durchsatz beim random read in [IOPS] und I/O-Servicezeit in [µs]

Sowohl die neue Prozessorgeneration mit PCI Gen4, also auch RoCE, als auch die PMEM-Technologie sollten sich hier po-

sitiv auf die Performance auswirken. Die Leistungsangaben für sequential read, die Oracle in den Datenblättern angibt, werden auch im Benchmark erreicht. Dies gilt sowohl für konventionelle Leseoperationen von den Storage-Servern als auch bei Verwendung der Smart-Scan-Technologie. Die Leistungsangaben für random read sind allerdings sehr optimistisch und können nur unter bestimmten Bedingungen erreicht werden. *Abbildung 3* zeigt das Random-I/O-Leistungsverhalten von Exadata Quarter Racks verschiedener Generationen bei 100% read, einer Datenbankgröße von 8 TByte und einer Oracle-Datenbank-Blockgröße von 8 KByte.

Auch hier hat es zwischen den einzelnen Exadata-Generationen eine deutliche Leistungssteigerung gegeben. Dabei ist der Durchsatz um den Faktor 5 gestiegen und die I/O-Servicezeit um den Faktor 10 gesunken. Da Oracle in seinen Exadata-High-Capacity-Storage-Servern ein Tiering über 3 Ebenen (PMEM, Flash Cache, Hard Disk Drives) verwendet, können allerdings sowohl der Durchsatz als auch die I/O-Servicezeiten stark schwanken. Wir haben die oben genannten Zahlen erst nach mehreren Wiederholungen (PMEM cache warmup) erreicht. Oracle gibt im Datenblatt einen maximalen Durchsatz von 5,6 Millionen IOPS bei 19 µs I/O-Servicezeit an. Diese Zahlen werden aber nur bei 100% Read-Operationen unter der Voraussetzung erreicht, dass sich alle Daten im PMEM Cache (im Quarterrack 4.5 TByte) befinden. Dieses Szenario ist nicht besonders realistisch.

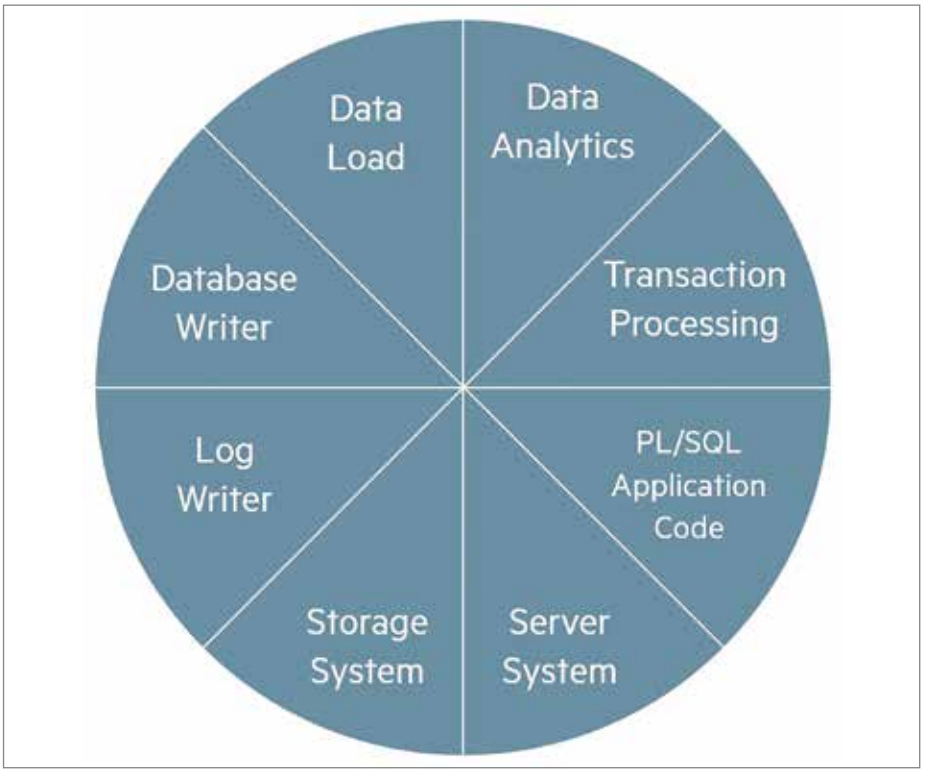


Abbildung 1: Vollständiger 360-Grad-Performance-Überblick (Quelle: peakmarks)

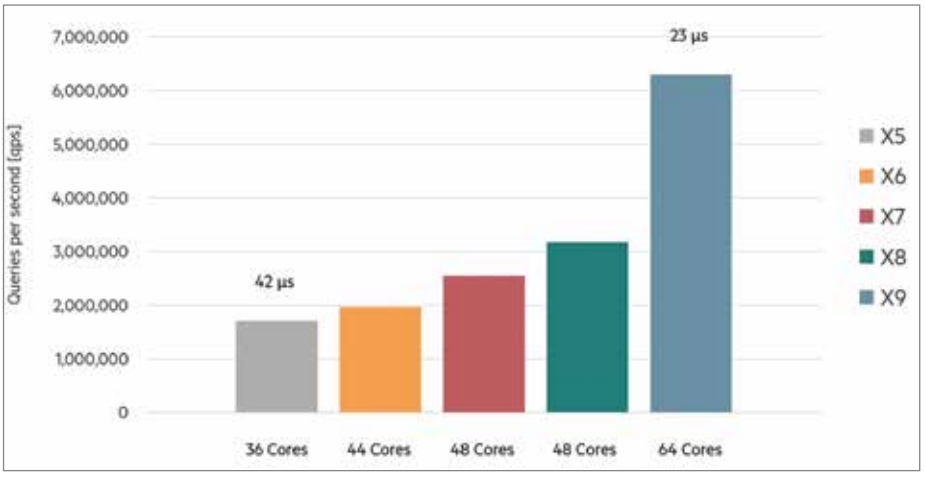


Abbildung 2: Maximaler Durchsatz beim Workload SRV-QUERY1 (Quelle: peakmarks)

Zur Leistungsfähigkeit der REDO-Logprotokollierung

Die peakmarks® Benchmark Software verfügt über eigene Workloads zum Vermessen der REDO Log Writer Performance bei verschiedenen Transaktionsgrößen. Der Workload LGWR-LAT1 ändert pro Transaktion einen Datensatz. Die REDO-Datenmenge pro Transaktion beträgt ca. 2 KByte.

Abbildung 4 zeigt das Performance-Verhalten einer Exadata X9M-2 Quarter Rack beim Workload LGWR-LAT1 in verschiedenen Lastsituationen vom einzelnen Prozess bis zur Systemsättigung.

Die kürzeste Latenzzeit für den „log-file sync"-Event liegt bei 47 Mikrosekun-

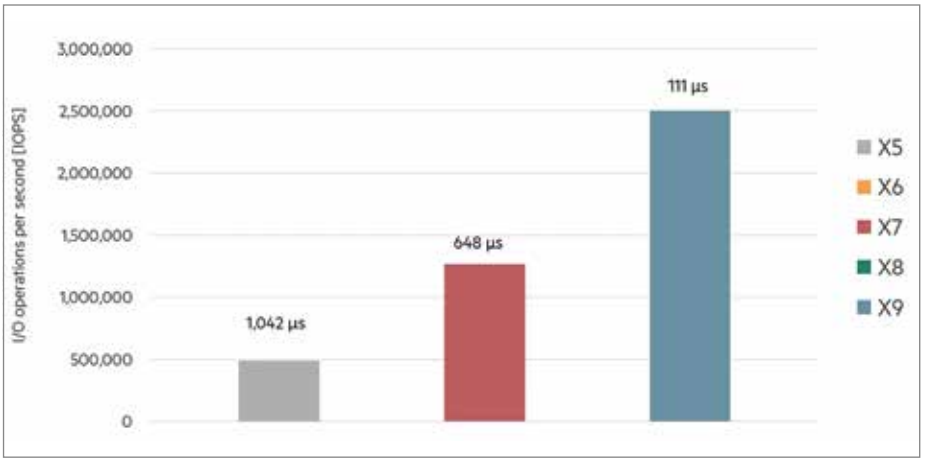


Abbildung 3: Maximaler Durchsatz beim Workload STO-RANDOM (Quelle: peakmarks)

Run	Test	Workload	Nodes	Jobs	CPU busy [%]	CPU user [%]	CPU sys [%]	CPU idle [%]	CPU iow [%]	Commit throughput [tps]	Commit latency [ms]	REDO blocks [rbps]	REDO writes [IOPS]	REDO syn writes [IOPS]	REDO data [MBps]	REDO data [kBpt]	LogFile sync [ms]	FlCache write [%]	Elapsed time [s]
4	192	LGWR-LAT1	2	2	2	1	0	98	0	17,085	0.117	74,132	17,134	17,085	32	1.92	0.047	24.78	302
	194	LGWR-LAT1	2	16	7	6	1	93	0	118,158	0.135	549,506	49,541	118,160	226	1.96	0.058	29.76	302
	196	LGWR-LAT1	2	32	11	10	1	89	0	211,878	0.151	1,000,856	43,353	211,881	406	1.96	0.071	21.71	302
	198	LGWR-LAT1	2	48	16	14	1	84	0	283,116	0.169	1,317,684	36,121	283,119	541	1.96	0.085	22.98	302
	200	LGWR-LAT1	2	64	20	18	1	80	0	332,211	0.192	1,513,514	30,699	332,215	633	1.95	0.102	23.99	302
	201	LGWR-LAT1	2	72	22	20	1	78	0	350,942	0.204	1,582,717	28,571	350,947	668	1.95	0.111	24.28	302
	202	LGWR-LAT1	2	80	24	21	1	76	0	367,848	0.217	1,644,659	26,908	367,853	699	1.95	0.120	24.88	302

Abbildung 4: Workload LGWR-LAT1 in verschiedenen Lastsituationen (Quelle: peakmarks)

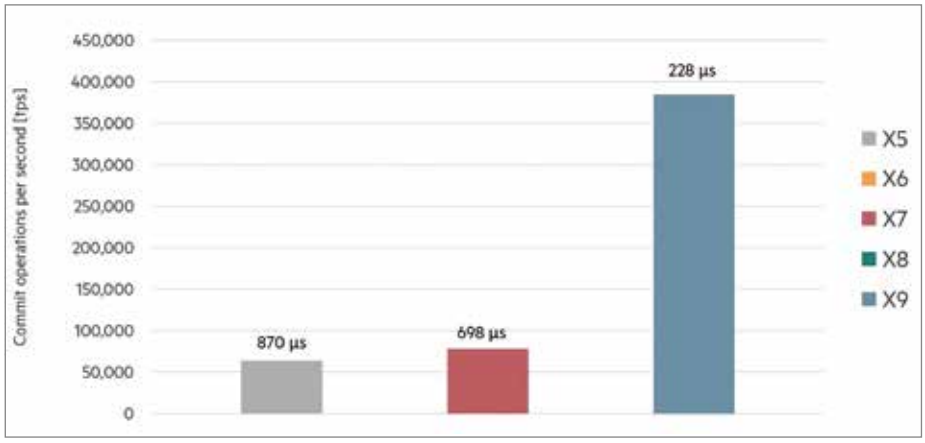


Abbildung 5: Maximaler Durchsatz beim Workload LGWR-LAT1 (Quelle: peakmarks)

den. Selbst bei einem Transaktionsvolumen von über 360.000 Transaktionen pro Sekunde steigt dieser Wert nur auf 120 Mikrosekunden an. Uns ist keine Oracle-Plattform bekannt – weder On-Premises noch in der Cloud –, die derart gute Performancezahlen für den Log Writer liefert.

Abbildung 5 zeigt die Leistungssteigerungen verschiedener Exadata-Generationen (Quarter Rack mit High-Capacity-Storage-Servern) für diesen Workload. Auch hier wird noch einmal deutlich, welche Leistungssteigerung der Einsatz der PMEM-Technologie ermöglicht. Das Transaktionsvolumen hat sich von X5 nach X9 um den Faktor 5-6 verbessert, die Latenzzeit der Log-Writer-Prozesse ist drastisch gesunken.

Zur Leistungsfähigkeit der Cloud@Customer-Lösung

Selbst für die bekannten Cloud Services von Amazon, Google und Microsoft findet man keine vergleichbaren und nachvollziehbaren Performance-Kennzahlen. Wir haben daher im Kundenauftrag die Cloud Services für Oracle-Plattformen von verschiedenen Anbietern vermessen.

Bei den Datenbank-Servern bieten die Mitbewerber von Oracle eine höhere Flexibilität bei der Wahl der Prozessoren und der Hauptspeicherkapazität an. Dort können Intel-Xeon-Prozessoren mit geringer Anzahl von Cores (dafür hohe Pro-Core-Leistung) und Prozessoren mit hoher Anzahl von Cores (gute Scale-up-Fähigkeit), mit 2 oder auch 4 Sockets mit entsprechend hoher Hauptspeicherkapazität gewählt werden. Grundsätzlich ist die Leistungsfähigkeit bei den Datenbank-Servern mit 2 Sockets aber unter allen Wettbewerbern ähnlich. Bei den Exadata-Datenbank-Servern wünschen wir uns in Zukunft auch bei den 2-Socket-Servern mehr Hauptspeicherkapazität.

Anders sieht das Bild bei den Storage-Servern aus. Die Cloud-Service-Anbieter setzen für Oracle-Plattformen durchweg All-Flash-Storage-Systeme mit einer konstanten Performance (Durchsatz und I/O-Servicezeit) ein. Storage-Systeme werden in der Regel über IP angebunden, verwenden aber keine RoCE-Technologie. Der I/O-Durchsatz pro Datenbank-Server ist niedriger, die I/O-Servicezeit deutlich höher (zwischen 200 – 500 Mikrosekunden) als bei Exadata-Systemen.

Oracle-Exadata-Systeme verwenden bei den High-Capacity-Storage-Servern ein Storage Tiering, wobei Durchsatz und I/O-Servicezeiten stark schwanken können. Die komplett mit Flash bestückten Extreme-Flash-Storage-Server werden aus Kostengründen kaum eingesetzt. In „eingeschwungenem" Zustand (hohe Cache-Trefferraten bei PMEM und Flash Cache) ist aber die I/O-Leistungsfähigkeit der Exadata-Cloud-Lösung allen anderen Mitbewerbern deutlich überlegen. Einzig die Effizienz bei der Speicherplatznutzung (ASM-Disk-Gruppen in High-Redundancy-Konfiguration) trübt ein wenig das Bild.

Zusammenfassung

Die Exadata-Generation X9 zeichnet sich durch hohe Zuverlässigkeit und Stabilität aus. Die Performance-Steigerungen gegenüber älteren Generationen sind enorm und wirken sich auf Applikationen aller Art positiv aus. Während die Cloud-Angebote der Mitbewerber nur durchschnittliche Leistungsanforderungen erfüllen können, ist die Exadata Cloud@Customer auch für Anwendungen mit höchsten Leistungsanforderungen geeignet.

Aus Platzgründen können wir nur einige wenige Benchmark-Ergebnisse vorstellen. Für mehr Informationen wenden Sie sich bitte an den Autor.

Manfred Drozd
manfred.drozd@peakmarks.com

Exadata Cloud@ Customer (ExaCC)

Clemens Bleile, dbi services

Was ist Exadata Cloud@Customer (ExaCC), in welchen Fällen bietet es eine Alternative zu anderen Lösungen wie Exadata (On-Premises) beziehungsweise Exadata Cloud Service, wie migriert man von der existierenden On-Premises-Welt auf die ExaCC und welche Hürden sind zu nehmen? Diese Fragen will der vorliegende Artikel auf Grundlage von Projekterfahrungen beantworten.

Was ist Exadata Cloud@ Customer (ExaCC)?

Oracle kennt drei Deployment-Modelle für die Exadata-Plattform *(siehe Abbildung 1)*:

- Exadata Database Machine (On-Premises; gekaufte Maschinen; vollständig in der Verantwortung des Kunden)
- Exadata Cloud@Customer (Cloud at Customer; gemietete Maschinen im eigenen Rechenzentrum mit Nutzungs-Abonnement (Subscription))
- Exadata Cloud Service (Public Cloud; gemietete Maschinen in der Public Cloud mit Nutzungs-Abonnement (Subscription))

Der Unterschied zwischen ExaCC und Exadata Cloud Service besteht heute hauptsächlich darin, wo die Exadata-Maschinen betrieben werden. Ziel einer ExaCC ist es neben dem Cloud-typischen Management, die Datenhoheit zu behalten (Daten verlassen das eigene Intranet nicht) und aufgrund der physischen Nähe zu On-Prem-Servern kleine Latenzzeiten zwischen Applikations- und DB-Server zu ermöglichen. Auf der ExaCC (wie auch auf dem Exadata Cloud Service) wird virtualisiert, das heißt, man kann mehrere virtuelle (Real Application) Cluster auf dem physischen Cluster betreiben. Man hat also nicht die Entscheidungsfreiheit (Bare Metal oder virtualisiert) wie auf einer gekauften Exadata Database Machine. Oracle als Betreiber der ExaCC ist zuständig für die Hardware und Software (bis zum und inklusive dem Hypervisor -> Dom0). Der Kunde ist zuständig für die virtuellen Maschinen (DomU), das heißt das Pflegen der Software in den virtuellen Maschinen (Betriebssystem, Grid Infrastructure, DB, *siehe auch Abbildung 2)*.

Das Erstellen von virtuellen Maschinen oder von Datenbanken erfolgt über das Cloud-Interface (Login im Browser zur Public-Cloud, *siehe Abbildung 3)*.

Hauptgründe für den Einsatz von ExaCC

Im Vergleich zum Public-Cloud-Service bietet die ExaCC hauptsächlich zwei Unterscheidungsmerkmale:

- Datenhoheit (zum Beispiel durch gesetzliche Vorgaben dürfen bestimmte Daten das Land nicht verlassen)
- niedrige Latenzzeiten

Speziell der Punkt mit den Latenzzeiten kann das Entscheidungskriterium für die Nutzung von ExaCC sein. Viele Applikationen sind abhängig von geringen Latenzzeiten, um performante Abfrageresultate zu erzielen. Eine Latenz zur Public Cloud von 3,5 ms im Vergleich zu einer lokalen Latenz von 0,25 ms (Faktor 14) kann einen

massiven Performance-Unterschied ausmachen. Die niedrigen Latenzzeiten sind für folgende Anwendungsfälle wichtig:

- Viele Daten werden zwischen Applikation und DB transportiert.
- Die Applikation führt viele Fetch-Aufrufe aus.
- SQL-Statements werden oft (zum Beispiel mehrere Millionen Aufrufe) ausgeführt.

Es ist nicht ungewöhnlich, dass Verarbeitungen, die On-Prem zwei Minuten dauerten, nach der Migration in die Public Cloud aufgrund der höheren Latenzzeit plötzlich etwa inakzeptable 20 Minuten benötigen. Meist handelt es sich um suboptimal entwickelte Anwendungen, aber in der Realität ist das Umschreiben der Applikation (aus Performance-Gründen) bei der Migration in die Cloud keine Option.

Der Autor hat einen Blog darüber geschrieben, wie man die Latenz zwischen Applikations- und DB-Server einfach messen kann *[1]*. Solche Messungen sollten vor jeder Migration des DB-Servers (oder des Applikations-Servers) in ein entferntes RZ durchgeführt werden.

Kundenprojekt: Stand und Vorgehensweise

Im Kundenprojekt wurde von einer DB-Umgebung auf Sun Solaris auf eine ExaCC migriert. Hauptgründe waren

- Erreichen von Kapazitätsgrenzen
- wiederkehrende Performance-Probleme
- reduzierte Verlässlichkeit (Reliability)
- steigende Hardware-Wartungskosten
- steigende Software-(Support-)Kosten
- keine Flexibilität mit der fixen Anzahl CPUs
- die Anforderung, zusätzliche Lizenzen von Oracle zu erwerben

Als Alternativen zu ExaCC wurden Linux Bare Metal, VMware und Oracle Database Appliance erwogen. Die Entscheidungskriterien waren

- HW-/SW-Unterstützung (Support)
- Patch-Aufwand
- Kosten
- Flexibilität bei der Oracle-Lizenzierung
- (Hoch-)Verfügbarkeit

Abbildung 1: Exadata-Deployment-Modelle (Quelle: © Oracle)

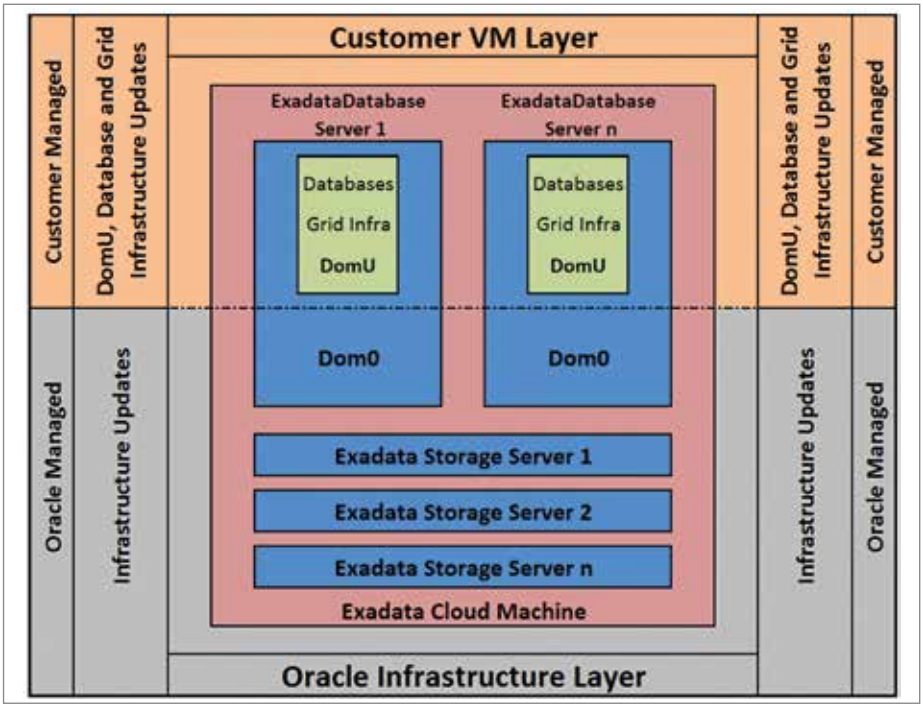


Abbildung 2: Verantwortlichkeiten (Quelle: © Oracle)

- Wiederverwendung von Storage
- Automatisierung
- vereinfachtes Management über Cloud-Interface beziehungsweise API
- Sicherheit (Software aktuell halten, Daten verschlüsseln)
- Flexibilität eines Cloud-Modells

Nachdem die Entscheidung zugunsten von ExaCC gefallen war, wurde eine Kapazitätsplanung durchgeführt. Für ExaCC gibt es vier Optionen (zum damaligen Zeitpunkt basierend auf der Exadata X8M, *siehe Abbildung 4)*.

Der Kunde entschied sich für zwei Quarter-Rack-Systeme (ausgelegt als Primär- und Standby-DB-Systeme). Somit ergab sich eine maximal nutzbare Kapazität von 4 * 50 OCPUs (1 OCPU = 1 Core). Die berechnete Minimal-Kapazität lag bei 68 OCPUs (40 im Rechenzentrum 1 und 28 in Rechenzentrum 2). Auf jedem physischen Cluster wurden sechs virtuelle Cluster erstellt *(siehe Abbildung 5)*.

Was war zu beachten?

Folgende Punkte waren zu beachten:

1. Von Capex zu Opex
   Statt der üblichen Investitionen alle 4-5 Jahre in neue Hardware (Captial Expenses = Capex) verändert sich das Finanz-Modell zu monatlichen Zahlungen für die gemietete Hardware und die genutzten Ressourcen (Operational Expenses = Opex).

2. Ausbildung des DBA-Teams
   Das DBA-Team musste in folgenden Bereichen ausgebildet werden:

- Exadata Features
  - Flash Cache
  - Smart Scans
  - Storage Indexes
  - HCC
- RAC / Grid Infrastructure
  - Clusterware
  - ASM
- Multitenant
  - Betrieb
  - Erweiterte Administration
- Snapshots
  - PDB Snapshot Copy
  - Snapshot-Hierarchie
- Zusätzliche Features
  - Advanced Security
  - Compression
  - In-Memory

3. Migration von Solaris (Big Endian) auf Linux (Little Endian)

Die Migration von Big Endian auf Little Endian reduzierte die Migrationsmethoden auf

- Data Pump
- logische Replikation (Golden Gate)
- Transportable Tablespaces
- Full Transportable Exp/Imp
- Zero Downtime Migration (ZDM-Werkzeug) *[2]*

Tatsächlich stellte sich heraus, dass die letzten drei Migrations-Methoden ausfielen, da ein neues Tablespace-Konzept und die Migration auf den AL32UTF8-Zeichensatz Transportable Tablespaces nicht zuließen. Das sehr zu empfehlende ZDM-Werkzeug konnte nicht genutzt werden, weil zum Zeitpunkt der Migrations-Tests keine ZDM-Version zur Verfügung stand, die Solaris als Betriebssystem der Quell-Datenbanken unterstützte.

4. Die Exadata-Maschinen gehören Oracle und müssen nach 4-5 Jahren zurückgegeben werden

Nach 4-5 Jahren Nutzung müssen die Maschinen an Oracle zurückgegeben werden. Entscheidet man sich auch zukünftig, ExaCC zu nutzen, so wird aktuelle ExaCC-Hardware zur Verfügung gestellt. Für eine Übergangsphase von 2-3 Monaten kann die dann alte ExaCC für die Migration auf die neue ExaCC genutzt werden. Es ist die Aufgabe des Kunden, diese Migration durchzuführen. Diese Migration ist recht einfach, da man sich (wahrscheinlich) auf gleichem Hardware-Typ befindet und keine Konvertierungen mehr nötig sind (Endian, Zeichensatz, Tablespace-Konzept, Multitenant).

Vorgehensweise Migration

In einer Vor-Migrations-Studie wurde die Zeit genutzt, bevor die ExaCC-Maschinen geliefert wurden:

- Inventur aller Datenbanken
- Ressourcen-Verbrauch aller Datenbanken berechnen
- Planung der benötigten Exadata-Ressourcen
- Mitarbeiter ausbilden
- Änderungen planen
  - Zeichensatz
  - neues Tablespace-Konzept
  - Planen der neu eingesetzten Features

Nach der Vor-Studie wurde die Migration getestet und durchgeführt. Das heißt, pro DB wurde ein eigener kleiner Migrationsplan erstellt.

Zur Kapazitätsberechung der Exadata wurde die AWR-Historie genutzt. Zudem wurde jede DB mit Attributen gelistet:

- Kritikalität
- Typ (Dev/Test/QA/Prod)

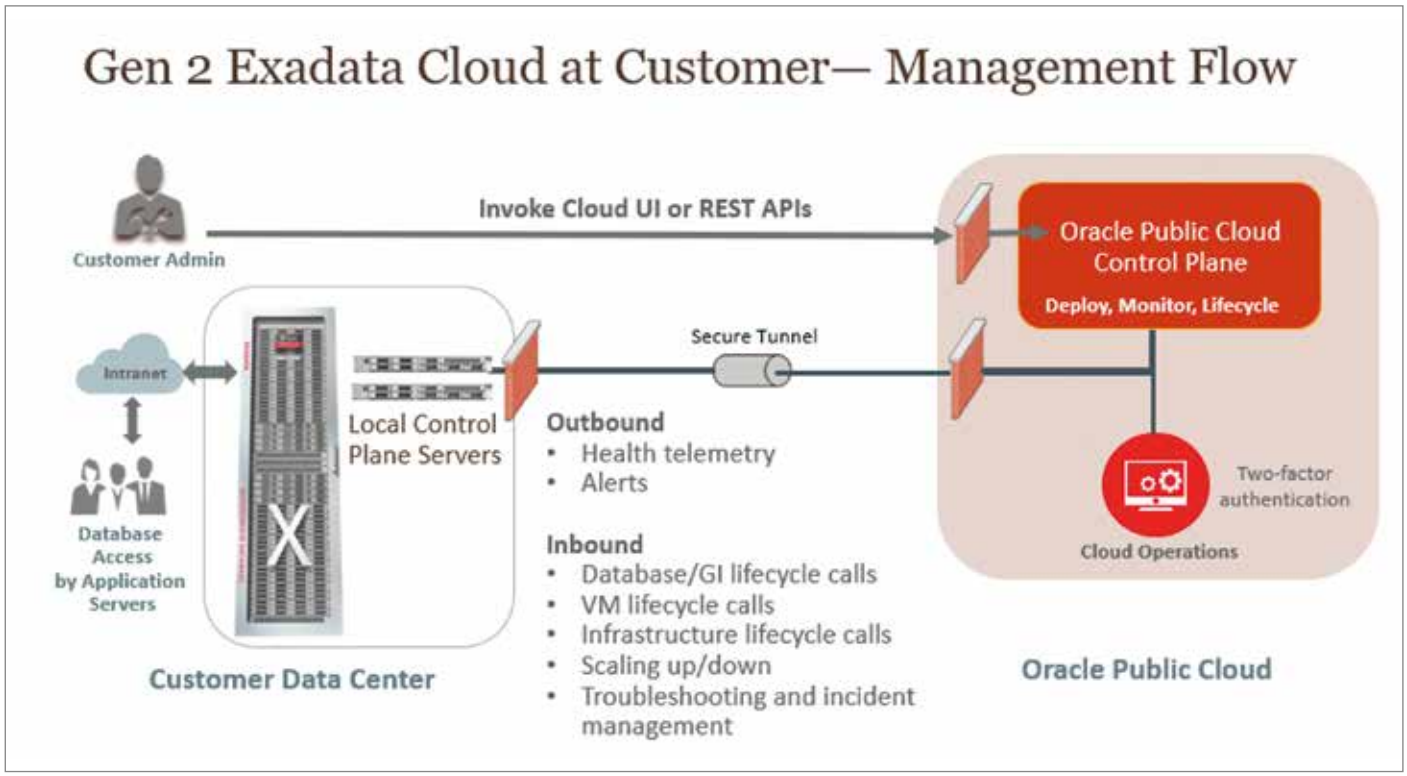


Abbildung 3: Management der ExaCC (Quelle: © Oracle)

	Base*	Quarter Rack	Half Rack	Full Rack
DB servers	2	2	4	8
Max OCPU	48	100	200	400
Total Memory	656 GB	2,780 GB	5,560 GB	11,120 GB
Storage servers	3	3	6	12
Usable disk storage	74.8 TB	149.7 TB	299.4 TB	598.7 TB

Abbildung 4: ExaCC-Spezifikationen X8M (Quelle: © APACOUC)

- Data Guard (J/N)
- SGA-/PGA-Größe
- spezielle DB-Parameter
- Character Set
- maximale Temp-Größe
- Migrations-Methode
- älteste Client-Version

Ein wichtiger Punkt war auch die Kapazitätsplanung für die nächsten Jahre:

- geplante Versionen
- Wachstum
- VM-Sizing

Die verfügbarkeitskritischen Datenbanken wurden mithilfe logischer Replikation (Golden Gate) migriert, das heißt Aufsetzen der logischen Replikation und zum Migrations-Zeitpunkt ein „Switchover" mit sehr kurzer Unterbrechungszeit. Alle anderen Datenbanken wurden mittels Data Pump migriert.

Backup & Recovery

Traditionell wurden selbstgeschriebene Skripte für das Backup & Recovery verwendet. Als Alternativen boten sich an:

- Automatisches Backup & Recovery von Oracle über das Cloud-Interface
- Backup & Recovery mithilfe des von Oracle zur Verfügung gestellten Werkzeugs bkup_api
- DB-Management-Kit (DMK) Backup & Recovery Werkzeug von dbi services

Aufgrund der größeren Flexibilität entschied man sich, das dbi-services-Werkzeug DMK Backup & Recovery einzusetzen. Hierzu bedurfte es aber der Lösung zweier Probleme:

- keine Oracle-Crontab auf ExaCC
- SSH auf die Maschinen nur über öffentliche IP möglich (kein SSH über VIPs)

Die Probleme wurden gelöst, indem alle Backup-Jobs von einer zentralen Maschine aus gestartet werden (vom Enterprise Manager Server). Eine vorherige SQL*Plus-Verbindung zur DB gibt den Host-Namen zurück, über den dann per SSH das Backup-Skript gestartet wird.

Einspielen von Patches

Oracle patcht alle Komponenten bis zu und einschließlich dem Hypervisor (Dom0):

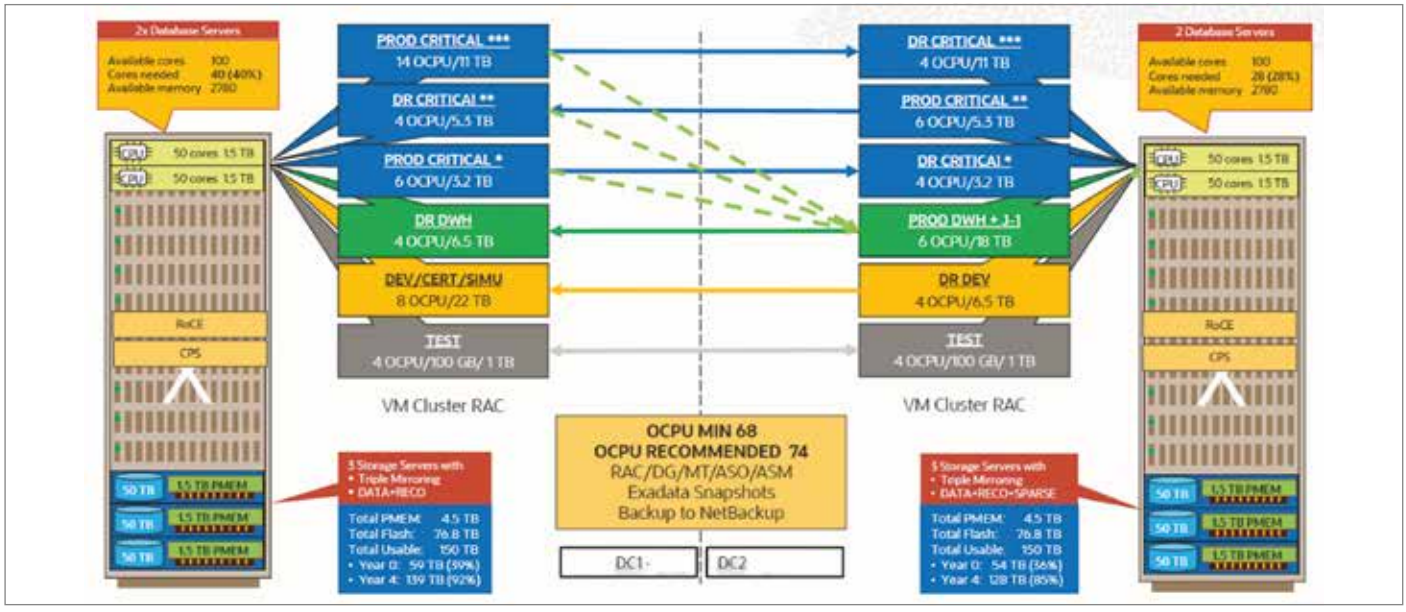


Abbildung 5: Ziel-Architektur (Quelle: © dbi services)

- Netzwerk-Switches
- Power Distribution Units
- ILOM
- Exadata Storage Servers
- Dom0

Hierzu wird ein Wartungsfenster pro Quartal benötigt. Man ist frei, den Zeitpunkt des Wartungsfensters im Quartal zu definieren, kann ihn aber nicht ins nächste Quartal verschieben.

Der Kunde ist verantwortlich für das Patchen

- des Betriebssystems der VMs (DomU)
- der Grid Infrastructure
- der Datenbanken

Man kann obige Komponenten über

- die Cloud-Service-Konsole
- ein API
- manuelles Einspielen

patchen. Das manuelle Einspielen von Patches sollte nur in Ausnahmefällen erwogen werden.

Zusammenfassung

ExaCC ist eine gute Lösung, wenn man

- Daten aus Sicherheits- oder Latenz-Gründen lokal halten muss
- konsolidieren will
- Performance-Probleme hat und von Exadata-Features profitiert
- mehr Oracle-Features nutzen will (mit dem „Lizenz enthalten"-Modell)
- von Capex auf Opex wechseln will
- für die Zukunft gewappnet sein will
  - Datenwachstum
  - Digitalisierung

Es gibt einige Dinge zu beachten, wenn man erwägt, auf ExaCC zu migrieren:

- Man muss die Migration gut planen.
- Mitarbeiter müssen ausgebildet werden.
- Das Zero-Downtime-Migration-Werkzeug wird empfohlen.
- Vorhandene/lizenzierte Features sollten genutzt beziehungsweise berücksichtigt werden.
- Man sollte in Container-Datenbanken konsolidieren.
- Man sollte auf den Zeichensatz AL32UTF8 konvertieren.
- Nach 4-5 Jahren muss wieder migriert werden.

Quellen

[1] Script to calculate the network latency between the application and the Oracle DB-server:
https://blog.dbi-services.com/script-to-calculate-the-network-latency-between-the-application-and-the-oracle-db-server

[2] Zero-Downtime-Migration-Werkzeug:
https://www.oracle.com/database/technologies/rac/zdm.html
https://www.oracle.com/a/tech/docs/oracle-zdm-step-by-step-guide.pdf

Über den Autor

Clemens Bleile hat mehr als 30 Jahre IT-Erfahrung, 13 davon im Oracle Support und 17 Jahre im Oracle Consulting. Er hat sich auf Oracle-Datenbank-Performance-Tuning (SQL-Tuning, DB-Tuning) und die Entwicklung von Oracle-DB-IT-Architekturen (hochverfügbar, wartungsarm, kosteneffiziente Speicherung von Daten) spezialisiert und ist ein Experte im Bereich Problem-Analyse und -Lösung.

Clemens Bleile
clemens.bleile@dbi-services.com

Oracle Datenbanken Monthly News

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

DOAG Online

Das sechsköpfige Redaktionsteam von Oracle Deutschland hat wieder Neuigkeiten rund um die Datenbank zusammengetragen, die dieses Mal von Wolfgang Thiem in einem ca. 20 minütigen Video präsentiert werden.

Wie immer geht es dabei nicht nur um Neuigkeiten wie Ankündigungen, sondern auch um aktuelle Events (mit Terminen), Releasestände & Patches und interessante Informationen aus dem Web.

Aktuelle Informationen wie die Ankündigung der Verfügbarkeit von Oracle Database Service für Microsoft Azure (Oracle Database Service for Azure), Eröffnung der ersten Sovereign Cloud Regionen in der EU, aber auch das neue Data Guard Feature in 21c oder Real Time Insight in Exadata sind nur einige Themen der neuen Ausgabe.

In der aktuellen Ausgabe wird wieder ein zusätzliches Quick Link Posting zur Verfügung gestellt, um einen schnellen Zugriff auf den zugehörigen Content zu gewährleisten.
https://www.doag.org/de/home/news/oracle-datenbanken-monthly-news-12/

To Trigger or not to Trigger

Jürgen Sieben, ConDeS

Dieser Artikel basiert auf einer Fundstelle in einem Fachbuch, die mich bei der Lektüre zu Widerspruch gereizt hat. Mir geht es natürlich nicht darum, mit dem Finger auf andere Autoren zu zeigen, sondern es dient als Ankerpunkt, um ein Thema zu erläutern, von dem ich hoffe, dass es für viele Leser interessant sein könnte. Die Fundstelle stellt eine Behauptung über Trigger auf und führt zur Frage: Wann und wie soll man Trigger denn nun einsetzen?

Die Fundstelle

In einem (ansonsten überaus lesenswerten) Buch über Oracle Performance Tuning stolperte ich über folgende Behauptung:

Aus Performancegründen sollten Sie natürlich so wenig wie möglich in Triggern gestalten, weil sich dadurch ja alle Transaktionen auf der Tabelle komplexer und auch zeitaufwendiger gestalten. Der zusätzliche Zeitaufwand kann nicht bemerkbar sein oder auch nicht (sic!); was einfach davon abhängt, wie viel Sie in den Trigger packen. Alle Aktionen innerhalb des Triggers erweitern ja die bereits bestehende Transaktion. Allerdings muss man auch sagen, dass Trigger ein erprobtes und vielgenutztes Datenbank-Feature sind, was Sie nicht von deren Einsatz abhalten sollte.

Hmm – der Absatz ist natürlich ein wenig verunglückt, nicht nur wegen des logischen Fehlers in der Verneinung, sondern auch im letzten Satz, bei dem ich mich frage, warum mich die Tatsache, dass der Trigger „erprobt und vielgenutzt" sei, von deren Einsatz abhalten sollte. Aber darum geht es mir nicht. Mich hat zum Widerspruch gereizt, dass behauptet wird, die Performance-Einbußen eines Triggers hingen (ausschließlich) von der Menge Code ab, die im Trigger ausgeführt wird. Das legt nahe, dass ein Trigger desto schneller ist, je weniger Code in ihm ausgeführt wird.

Nun mag das zwar auch eine Rolle spielen (wenn etwas gemacht wird, dauert das eben immer länger, als wenn man es lässt), doch drängt sich mir die Frage auf, ob Sie in Ihren Trigger Logik „aus Spaß" einfügen, die Sie also auch weglassen könnten. Falls dies so wäre, schlösse ich mich natürlich der Empfehlung an, diesen überflüssigen Code wegzulassen. Doch was ist, wenn Sie die Logik nun einmal ausführen müssen? Verbieten sich Trigger dann wegen der Codemenge? Grund genug, einmal zu betrachten, wie das mit den Triggern eigentlich genau ist und welche Optionen sich uns bieten.

Das Problem

Wie so häufig, ist der Trigger nicht „grundsätzlich schlecht". Das zentrale Problem, das Trigger auslösen, sind die Umgebungswechsel, wie ich das in meinem Artikel

„Umgebungswechsel vermeiden“ *(https://www.doag.org/formes/pubfiles/7276237/04-2015-DOAG-SOUG-News-Jurgen-Sieben-Umgebungswechsel-vermeiden.pdf)* schon geschildert hatte. Diese Umgebungswechsel resultieren daraus, dass die Logik des Triggers in PL/SQL, die Auslösung des Triggers jedoch in SQL implementiert ist, sodass beim Auslösen eines Triggers aus SQL ein Umgebungswechsel zu PL/SQL resultiert. Dies passiert zwar immer und bei jedem Triggertyp, doch ist eine Triggerart für die Performanz von besonderer Bedeutung: der Zeilentrigger, da dieser für jede durch die DML-Anweisung in SQL betroffene Zeile einmal auslöst. Da eine DML-Anweisung sehr viele Zeilen betreffen kann, potenziert sich hier also das grundsätzlich immer vorhandene Problem und macht sich dadurch entsprechend bemerkbar. Ist Ihr Problem jedoch, dass sehr viele Benutzer sehr viele einzelne DML-Anweisungen ausführen, ist auch ein Anweisungstrigger bereits ein Problem, einfach weil die Umgebungswechsel nun auch auf Anweisungsebene eine signifikante Anzahl erreichen.

Datenbanktrigger, die bei sonstigen Ereignissen auslösen, wie zum Beispiel dem An- oder Abmelden eines Benutzers an der Datenbank, können im Regelfall für die folgende Betrachtung ignoriert werden, weil die triggernden Ereignisse zu selten auftreten, als dass die Zeit für die Umgebungswechsel bedeutsam würde.

Wie relevant ist dieses Problem?

Um ein Gefühl für die Größe dieses Problems zu erhalten, müssen wir zunächst einmal relevant viel Arbeit in der Datenbank verrichten. Sind die Zeilenmengen oder die Anzahl der einzelnen DML-Anweisungen gering, sind es auch die negativen Auswirkungen der Trigger und die gesamte Antwortzeit dürfte in einem tolerablen Rahmen liegen.

Um die Auswirkungen zu beobachten, erstellen wir uns einen Beispielbenutzer DOAG, dem ich der Einfachheit halber die DBA-Rolle zugewiesen habe. Dadurch hat er Zugriff auf recht viele Datenbankobjekte, mit deren Metadaten ich im Folgenden arbeiten möchte. Ich verwende folgende Beispieltabelle in *Listing 1*.

Das Einfügen der Daten inklusive Pflege des Unique-Index für den Primärschlüssel ging recht flott. Wenn Sie diese Daten aktualisieren, geht auch das recht zügig *(siehe Listing 2)*.

Lassen Sie uns alternativ einmal einen Blick darauf werfen, wie die Laufzeiten wären, wenn viele Benutzer jeweils eine Zeile in die Tabelle einfügen würden. Wir machen uns die Dinge etwas leichter und simulieren keine parallele Bearbeitung durch viele Benutzer, sondern fügen seriell viele Zeilen in jeweils einer `insert`-Anweisung ein, und zwar durch diesen Code *(siehe Listing 3)*.

Ich musste, um ein realistisches Bild zu zeichnen, den Parameter `plsql_optimize_level` auf `0` zurückstellen, damit der Compiler keine interne Optimierung und damit eine Mengenverarbeitung durchführen kann. Ich habe die Prozedur zudem als Invokers-Rights-Prozedur ausgeführt, damit die Anzahl der Zeilen in `TEST_TABLE` zwischen beiden `insert`-Varianten identisch ist (ansonsten würden über Rollenrechte zugewiesene Datenbankobjekte nicht eingefügt). Die Laufzeit bei diesem Ansatz ist ohne Trigger deutlich langsamer als bei der Mengenverarbeitung, aber das war zu erwarten und ist nicht wirklich zu ändern *(siehe Listing 4)*.

Nun jedoch fügen wir einen Zeilentrigger hinzu, der die gleiche Aufgabe wie unsere `update`-Anweisung von vorhin durchführt *(siehe Listing 5)*.

Der analoge Aufruf über eine Prozedur `update_test_table`, die das gleiche Update erzeugt, zeigt, dass auch hier ein zweiter Umgebungswechsel durch den Trigger erforderlich ist *(siehe Listing 6)*.

Die Datenbank macht jedoch immer noch einen guten Job, denn hier werden immerhin 92249 Umgebungswechsel in knapp zwei Sekunden durchgeführt, also in etwa 0,2 tausendstel Sekunden pro Umgebungswechsel. Die PL/SQL-Prozedur hat ja durch die eigene zeilenweise Arbeitsweise bereits einen Umgebungswechsel hin zu SQL gemacht, der durch den Trigger nun um eine „Rücküberweisung“ nach PL/SQL erweitert wird. Das macht sich außerordentlich deutlich bemerkbar.

Noch schlimmer wird es, wenn wir im Trigger noch einen weiteren Umgebungswechsel einfügen, zum Beispiel durch die Verwendung der Funktion `user`, die, wie Sie wissen, als `select user from dual` implementiert ist und daher auf die SQL-Seite muss *(siehe Listing 7)*.

Leider ist es im Trigger nicht möglich, den Aufruf der Funktion `user` irgendwie zu vermeiden, weil ja der gesamte Code pro betroffener Zeile einmal ausgeführt wird. Der zusätzliche Aufwand ist auch bei unserer Prozedur deutlich zu spüren *(siehe Listing 8)*.

Die Anweisung zeigt nun bereits eine signifikant längere Laufzeit, sie benötigt

```
SQL> create table test_table(
  2    owner varchar2(30),
  3    object_id number,
  4    object_name varchar2(30),
  5    object_type varchar2(30),
  6    constraint pk_test_table primary key(object_id)
  7  );

Tabelle wurde erstellt.

SQL> set timing on;
SQL> select count(*)
  2    from all_objects;

  COUNT(*)
----------
     92248
Abgelaufen: 00:00:00.32

SQL> insert into test_table(owner, object_id, object_name, object_type)
  2  select owner, object_id, object_name, object_type
  3    from all_objects;

92249 Zeilen erstellt.
Abgelaufen: 00:00:00.86
```

Listing 1: Beispieltabelle

bei der Mengenverarbeitung gut zehnmal länger für die im Prinzip gleiche Arbeit (wobei bei diesen geringen Zeiten die Streuung der Messungen relativ groß sein kann). Ohne Trigger hätte die `initcap`-Funktion ebenso wie der Änderungsbenutzer aus `user` direkt in die `update`-Anweisung eingefügt werden können, ohne die Einbuße bei der Performanz. Halten wir jedoch auch fest: Langsam ist sehr relativ. Wir haben im Beispiel über 90.000 Zeilen in knapp 4 Sekunden aktualisiert, pro Zeile also in etwa 0,4 tausendstel Sekunden. Das ist sehr viel Zeit, wenn Sie 90.000.000 Zeilen aktualisieren müssen (bei linearer Zunahme der Ausführungszeit entspricht das immerhin beinahe einer Dreiviertelstunde), aber wenig, wenn Sie nur 500 Zeilen zu aktualisieren haben. Doch auch diese Zeit kann als zu lang empfunden werden, wenn Sie zeilenweise arbeiten (müssen), wie unsere Prozedur eindrucksvoll zeigt.

Natürlich wird die Sache noch einmal aufwendiger, wenn Sie einen zweiten Trigger, entweder auf den gleichen oder einen weiteren Event, anfügen oder wenn die Trigger mehr PL/SQL-Arbeit zu leisten haben, aber das Prinzip ist, glaube ich, nun klar.

Lösungsansätze

Die Lösungsansätze sollten zunächst einmal im Blick haben, warum Trigger überhaupt eingefügt wurden. Der große Vorteil eines Triggers besteht in der Zentralisierung von Logik. Ist diese einmal im Trigger definiert, halten sich alle DML-Anweisungen der Anwendung, aber auch Ad-hoc-Anweisungen, an die dort definierten Regeln. Die einzige Möglichkeit, diese Regeln zu umgehen, besteht darin, einen Trigger zu deaktivieren oder ganz zu löschen. Eine Lösung des Problems sollte diesen Vorteil nach Möglichkeit erhalten.

Andererseits kann man sicherlich auch eine leichte Verschlechterung dieses Komforts akzeptieren, wenn dadurch ein anderer Vorteil größer würde. Und damit meine ich nicht nur die Performanz, sondern die Übersichtlichkeit und Wartbarkeit des Codes. Trigger haben den Nachteil, die Geschäftslogik auf viele unterschiedliche „Programminseln“ zu verteilen. Nehmen wir an, Logik beziehe sich auf Daten, die in einer einfachen 1:n-Rela-

```
SQL> update test_table
  2     set object_name = initcap(object_name);

92249 Zeilen aktualisiert.
Abgelaufen: 00:00:00.35
```

Listing 2: Aktualisieren der Daten

```
SQL> alter session set plsql_optimize_level = 0;

Session wurde geändert.

SQL> create or replace procedure fill_test_table
  2    authid current_user
  3  as
  4    cursor object_cur is
  5      select owner, object_id, object_name, object_type
  6         from all_objects;
  7  begin
  8    for obj in object_cur loop
  9      insert into test_table(owner, object_id, object_name, object_
type)
 10      values (obj.owner, obj.object_id,
 10              obj.object_name, obj.object_type);
 12    end loop;
 13  end fill_test_table;
 14  /

Prozedur wurde erstellt.
```

Listing 3: Hinzufügen seriell vieler Zeilen in jeweils einer insert-Anweisung

```
SQL> call fill_test_table();

Aufruf wurde abgeschlossen.
Abgelaufen: 00:00:04.46

SQL> commit;

Transaktion mit COMMIT abgeschlossen.
```

Listing 4: Laufzeit bei Ansatz ohne Trigger

```
SQL> create trigger trg_test_table_briu
  2  before insert or update on test_table
  3  for each row
  4  begin
  5    :new.object_name := initcap(:new.object_name);
  6  end;
  7  /

Trigger wurde erstellt.

SQL> update test_table
  2     set object_name = object_name;

92249 Zeilen aktualisiert.

Abgelaufen: 00:00:01.78
```

Listing 5: Hinzufügen eines Zeilentriggers, der die gleiche Aufgabe wie unsere `update`-Anweisung von vorhin durchführt

tion in zwei Tabellen gespeichert werden. Wo legen Sie nun die Logik für einen Datensatz an? Teilen wir die Logik auf mehrere Trigger auf, verschärfen wir nicht nur die Performanz-Problematik, sondern zudem wird die Wartung unseres Codes aufwendiger und das Verständnis erschwert. Wenn eine Lösung hierfür einen Ausweg böte, wäre auch das ein Gewinn.

Vermeidung von Triggern durch SQL

Der erste Lösungsansatz besteht darin, die Aufgaben der Trigger durch SQL-Funktionalität zu ersetzen. Hier ist vor allem die neue *Identity-Column* zu erwähnen, die es ermöglicht, einen Trigger, der einen Primärschlüssel erzeugt, durch eine Autowertspalte zu ersetzen. Eine weitere Möglichkeit besteht in der Nutzung der erweiterten `default`-Klausel einer Spalte; beide Möglichkeiten beziehen sich allerdings auf Datenbankversion 12c und aufwärts. Mit dieser Version ist es möglich, einige Trigger einzusparen und ersatzlos durch SQL zu ersetzen. Lässt sich ein Trigger auf diese Weise einsparen, sollten Sie das in jedem Fall auch tun.

Compound Trigger

Wenn Trigger schon erforderlich sind, könnte die Anzahl dieser Trigger durch einen Compound-Trigger verringert werden. Diese Trigger, die ab Version 11g zur Verfügung stehen, fassen alle auslösenden Events in einem einzigen Trigger zusammen. Dieser Ansatz sorgt zunächst einmal dafür, dass die gesamte Logik einer Tabelle an einer Stelle versammelt ist, löst aber nicht das Problem von 1:n-Relationen. Er sieht in der Grundform so aus (wobei die Stubs nicht geschrieben werden müssen, *siehe Listing 9)*.

Dieser Trigger läuft zunächst einmal genauso lange wie der vorher existierende Trigger, den ich für die nächsten Versuche vorab gelöscht habe. Dieser Trigger kann also die Umgebungswechsel nicht verhindern. Deutlich wird dies, wenn der Zweig `after row` durch weitere (ebenfalls nicht sehr intelligente) Logik erweitert wird *(siehe Listing 10)*.

Wir verdoppeln den Aufwand, denn nun muss auch nach jeder Zeile ein Umgebungswechsel nach PL/SQL (und von dort zurück nach SQL) durchgeführt werden *(siehe Listing 11)*.

Doch hat der Compound Trigger einen großen Vorteil: Durch seine Struktur können wir in ihm auch globale Variablen, Cursor etc. definieren. Nutzen wir diese Möglichkeit und speichern den aktuellen Benutzer in einer globalen Variablen `g_user` *(siehe Listing 12)*.

Nun ersetzen wir die Aufrufe der Funktion `user` durch eine Referenz auf `g_user` und sehen, dass die Datenbank die Hälfte der Roundtrips einspart *(siehe Listing 13)*.

Mit dieser Variante sind wir, trotz zweier SQL-Events, schneller als mit einem normalen Trigger bei nur einem Ereignis. Wenn schon Trigger, dann also zumindest Compound-Trigger, zumal sich mit diesem Trigger auch elegant die lästigen Mutating-Table-Probleme lösen lassen.

Vermeidung von Triggern durch Table-API

Die „große" Lösung besteht in der Vermeidung von Triggern durch ein Table-API. Voraussetzung hierfür ist, dass nicht unkontrolliert DML-Anweisungen gegen die Tabelle ausgeführt werden können. Leider können Sie dies nicht verhindern, solange Sie sich als der Eigentümer dieser Tabellen an der Datenbank anmelden. Melden Sie sich als ein anderer Benutzer an, müssen Sie vom Eigentümer das Recht erhalten haben, eine Tabelle zu

```
SQL> call update_test_table();

Aufruf wurde abgeschlossen.
Abgelaufen: 00:00:07.12
```

Listing 6: Analoger Aufruf über eine Prozedur `update_test_table`

```
SQL> alter table test_table add (change_user varchar2(30));

Tabelle wurde geändert.

SQL> create or replace trigger trg_test_table_briu
  2  before insert or update on test_table
  3  for each row
  4  begin
  5    :new.object_name := initcap(:new.object_name);
  6    :new.change_user := user;
  7  end;
  8  /

Trigger wurde erstellt.

SQL> update test_table
  2     set object_name = object_name;

92249 Zeilen aktualisiert.

Abgelaufen: 00:00:03.90
```

Listing 7: Trigger mit Einfügung eines weiteren Umgebungswechsels, zum Beispiel durch die Verwendung der Funktion `user`

```
SQL> call update_test_table();

Aufruf wurde abgeschlossen.
Abgelaufen: 00:00:09.75
```

Listing 8: Zusätzlicher zeitlicher Aufwand

benutzen. Und hier setzt die Lösung an, denn genau das wird der Eigentümer der Tabelle nicht tun. Stattdessen richtet er eine Prozedur ein, auf die Sie ein Ausführungsrecht erhalten. Diese Prozedur wiederum schreibt die Daten in die Tabelle.

Da wir bereits eine solche Prozedur haben, können wir sie auch benutzen, um den Ablauf eines Table-API zu simulieren. Unsere Update-Prozedur macht dabei folgende Dinge: Sie wandelt die Daten vor der Verarbeitung in das gewünschte Format und vermeidet häufige Umgebungswechsel durch Implementierung einer Bulk-Strategie. Der erste der beiden Punkte ist nicht weiter spannend, das hatten wir in einer `update`-Anweisung ja auch bereits realisiert, der Punkt ist die Bulk-Verarbeitung. Dabei werden mehrere Zeilen im PL/SQL-Code aufbereitet und im Bulk an die SQL-Seite zur Abarbeitung weitergegeben. Wie dies grundsätzlich geschieht, hatte ich bereits im letzten Artikel geschildert. Für unser Beispiel ist es sogar noch einfacher, weil die for `record in cursor`-Schleife unserer Prozedur „von Haus aus" diese Bulk-Implementierung durchführt, sobald wir den Compiler dies tun lassen. Um zu zeigen, wohin uns unser Table-API führt, lösche ich also den Trigger, realisiere die Triggerfunktionalität in der Prozedur und kompiliere diese mit dem Setting `PLSQL_OPTIMIZE_LEVEL` auf mindestens 2 *(siehe Listing 14)*.

Nun ist unsere Performanz besser als beim zeilenweisen Aufruf ohne Trigger, zudem sind die Funktionen des Triggers implementiert. Im direkten Vergleich zu „reinem" SQL ohne Trigger ist diese API-Lösung dann aber doch nicht ganz überzeugend *(siehe Listing 15)*.

Aber eben immer noch deutlich besser als alle anderen Implementierungen. Zudem bietet die Implementierung der Prozedur noch Raum für Verbesserungen und die reine SQL-Variante ist eben auch nur dann möglich, wenn die Daten bereits in der Datenbank vorhanden sind. Die Idee des Table-API ist es hingegen, eine Kollektion von Werten aus Anwendungen entgegenzunehmen und möglichst effizient in die Datenbank zu schreiben.

Zusammenfassung

Wie üblich gibt es bei der Datenbank kein einfaches Wahr oder Falsch. Trigger kön-

```
SQL> create or replace trigger trg_test_table_comp
  2    for delete or insert or update on test_table
  3    compound trigger
  4
  5    before statement -- STUB
  6    is
  7    begin
  8      null;
  9    end before statement;
 10
 11    before each row
 12    is
 13    begin
 14      :new.object_name := initcap(:new.object_name);
 15      :new.change_user := user;
 16    end before each row;
 17
 18    after each row -- STUB
 19    is
 20    begin
 21      null;
 22    end after each row;
 23
 24    after statement -- STUB
 25    is
 26    begin
 27      null;
 28    end after statement;
 29  end;
 30  /

Trigger wurde erstellt.

SQL> update test_table
  2     set object_name = object_name;

92249 Zeilen aktualisiert.
Abgelaufen: 00:00:03.95
```

Listing 9: Grundform mit Compound-Trigger-Ansatz

```
...
 18    after each row
 19    is
 20    begin
 21      if :new.change_user != user then
 22        raise_application_error(-20000, 'Falscher Benutzer');
 23      end if;
 24    end after each row;
...
```

Listing 10: Trigger bei Erweiterung des Zweigs `after row` durch weitere Logik

```
SQL> update test_table
  2     set object_name = object_name;

92249 Zeilen aktualisiert.
Abgelaufen: 00:00:04.61
```

Listing 11: Verdoppeln des Aufwands, wodurch auch nach jeder Zeile ein Umgebungswechsel nach PL/SQL (und von dort zurück nach SQL) durchgeführt werden muss

```
SQL> create or replace trigger trg_test_table_comp
  2    for delete or insert or update on test_table
  3    compound trigger
  4
  5    g_user varchar2(30) := user;
...
```

Listing 12: Speichern des aktuellen Benutzers in einer globalen Variablen `g_user`

```
SQL> update test_table
  2     set object_name = object_name;

92249 Zeilen aktualisiert.
Abgelaufen: 00:00:03.10
```

Listing 13: Ersetzen der Aufrufe der Funktion `user` *durch eine Referenz auf* `g_user`

```
SQL> drop trigger trg_test_table_comp;

Trigger wurde gelöscht.

SQL> alter session set plsql_optimize_level = 3;

Session wurde geändert.

SQL> create or replace procedure update_test_table
  2  as
  3    cursor object_cur is
  4      select owner, object_id, object_name, object_type
  5         from all_objects;
  6    g_user varchar2(30) := user;
  7  begin
  8    for obj in object_cur loop
  9      update test_table
 10        set object_name = initcap(obj.object_name),
 11            change_user = g_user
 12      where object_id = obj.object_id;
 13    end loop;
 14  end update_test_table;
 15  /

Prozedur wurde erstellt.

SQL> call update_test_table();

Aufruf wurde abgeschlossen.
Abgelaufen: 00:00:01.07
```

Listing 14: Löschen des Triggers, Realisieren der Triggerfunktionalität in der Prozedur und Kompilieren dieser mit dem Setting `PLSQL_OPTIMIZE_LEVEL` *auf mindestens 2*

```
SQL> update test_table
  2     set object_name = initcap(object_name),
  3         change_user = user;

92251 Zeilen aktualisiert.
Abgelaufen: 00:00:00.31
```

Listing 15: Vergleich mit „reinem" SQL

nen, im richtigen Kontext angewendet, sehr sinnvolle Datenbankobjekte sein. Beispiele wären Stammdatentabellen und andere, weniger intensiv genutzte Tabellen. Trigger sind aber mit einer performanten Anwendung nicht in Einklang zu bringen, wenn entweder sehr viele Einzelanweisungen oder Mengenoperationen auf Tabellen bearbeitet werden sollen. Dann eignen sich Table-APIs besser. Table-APIs setzen allerdings eine architektonische Entscheidung voraus, dass nicht jeder Anwendungsbenutzer unmittelbar auf die Tabellen schreiben darf. Das mag durchsetzbar sein oder nicht, aber damit steht und fällt die Strategie. Sobald jedoch die Aufgabe des Triggers auch durch eine SQL-Funktionalität wahrgenommen werden kann, sollte dies in jedem Fall gemacht werden, auch durch entsprechende Refaktorisierung des Codes.

Da dies eine PL/SQL-Kolumne ist, verbietet sich der Hinweis, dass für eine mengenorientierte Verarbeitung PL/SQL nicht unbedingt erforderlich ist, sondern auch in Java oder anderen Sprachen implementiert werden kann. Stichwort wäre hier die `batch-update`-Methode des Datenbanktreibers.

Gemeinsames Ziel aller Programmierung gegen die Datenbank ist die Implementierung mengenorientierter Ansätze und die Vermeidung zeilenbasierter Programmierung. Die jedoch wird durch Trigger, gerade durch Zeilentrigger, unausweichlich in das Projekt gebracht. Die Menge Code des Triggers hat nur mittelbar mit der gesamten Ausführungszeit zu tun, zumal die dort implementierte ja ohnehin irgendwo gerechnet werden muss. Es ist der Aufwand für den Umgebungswechsel, der hier den limitierenden Faktor darstellt, daher sollte hierauf auch in besonderem Maße geachtet werden.

Jürgen Sieben
j.sieben@condes.de

Container Only – Multitenant in Oracle 21c

Markus Flechtner, Ordix

Mit der Version 21c der Oracle-Datenbank schneidet Oracle alte Zöpfe ab. Acht Jahre nach der Einführung der Container-Datenbank-Architektur mit Oracle 12c heißt es, sich von der klassischen Datenbank-Architektur zu verabschieden. Grund genug, sich einmal einige der neuen Funktionen für die Multitenant-Architektur, mit denen uns Oracle den Wechsel schmackhaft machen möchte, anzuschauen.

Oracle Database 21c ist ein Innovation Release

Im Gegensatz zum aktuell meistverwendeten Release Oracle 19c ist Oracle 21c ein Innovation Release. Ein Innovation Release sollte dazu dienen, sich mit den neuen Features vertraut zu machen. Es sollte eher nicht für Produktionssysteme verwendet werden. Das zeigt sich auch im Support für Oracle 21c: Der „Premier Support" endet am 30. April 2024 und es gibt keinen „Extended Support" (Quelle: Support-Note 161818.1). Damit endet der Support für Oracle 21c etwa zum gleichen Zeitpunkt wie für den Premier Support für Oracle 19c; mit dem Unterschied, dass es für das „Long Term Support Release" 19c anschließend Extended Support gibt. Als nächstes Long Term Support Release wird im kommenden Jahr „Oracle 23c" erwartet. Die Einschränkungen des Innovation Release zeigen sich auch bei den Plattformen, für die Oracle 21c verfügbar ist: Linux, Windows und HP-UX. Für andere Plattformen wie Solaris oder AIX soll Oracle 23c das nächste Release werden.

Multitenant ist die einzige Architektur

Mit Oracle 21c verabschiedet sich Oracle von der klassischen Datenbank-Architektur (Non-CDB), die wir seit Oracle 6 kennen. Seit Oracle 12c gab es die beiden Architekturen „Non-CDB" und „CDB" (Multitenant) parallel und mit Oracle 21c gibt es nur noch die Multitenant-Architektur. Für diejenigen, die noch nicht mit Oracle 19c oder früher zur Container-Datenbank-Architektur gewechselt sind, steht dieser Schritt somit mit Oracle 21c (oder 23c) an.

Upgrade und Migration zu Multitenant

Mit dem Abschied von der Non-CDB-Architektur macht Oracle den Wechsel zu den Container-Datenbanken auch einfacher. Da ist zuallererst das Tool „AutoUpgrade" zu nennen. AutoUpgrade kann automatisch eine Non-CDB aktualisieren und als PDB in eine vorhandene CDB einhängen. Upgrade- und Dictionary-Konvertierungsskripte inklusive. Oder ein Upgrade einer PDB durch Unplug/Plug in eine höhere Version. AutoUpgrade wird außerhalb des Release-Zyklus der Datenbank permanent weiterentwickelt und steht über die Support-Note 2485457.1 zum Download zur Verfügung. Über AutoUpgrade ist viel geschrieben worden; die beste Informationsquelle findet sich sicher auf dem Blog unter *mikedietrich-de.com [1]*.

Aber auch außerhalb von AutoUpgrade hat Oracle das Thema Migration und Upgrade weiterentwickelt: Wenn man eine ältere PDB in eine 21c-CDB einhängt, dann erkennt Oracle das und macht automatisch das Upgrade. Und wenn man

```
2022-06-19T12:51:03.882993+02:00
DEMONCDB(6):Starting Upgrade on PDB Open
[...]
2022-06-19T12:51:08.084897+02:00
DEMONCDB(6):alter pluggable database application APP$CDB$CATALOG begin
upgrade
  '19.0.0.0.0' to '21.0.0.0.0.partial' on error capture
DEMONCDB(6):Completed: alter pluggable database application APP$CDB$-
CATALOG begin upgrade
  '19.0.0.0.0' to '21.0.0.0.0.partial' on error capture
[...]
DEMONCDB(6):SERVER ACTION=NONCDB_TO_PDB id=: Converted non-CDB to PDB
in release 21.3.0.0.0 Container=DEMONCDB Id=6
DEMONCDB(6):alter pluggable database application app$cdb$pdbonly$ncd-
btopdb end upgrade
DEMONCDB(6):Completed: alter pluggable database application app$cdb$pd-
bonly$ncdbtopdb end upgrade
2022-06-19T13:43:28.073067+02:00
DEMONCDB(6):Finished Conversion from non-CDB on PDB Open
```

Listing 1: Auszug aus dem alert.log beim Einhängen einer 19c-Non-CDB in eine 21c-CDB

eine Non-CDB einhängt, dann geschehen die Dictionary-Konvertierung und das Upgrade beim ersten Öffnen der Datenbank automatisch. Das dauert dann natürlich etwas länger und im alert.log findet man folgende Einträge *(siehe Listing 1)*.

Nach dem ersten Öffnen der PDB muss man nur noch die ungültigen Objekte mit dem Skript „utlrp.sql" re-kompilieren, damit die Datenbank-Komponenten in DBA_REGISTRY vom Status „UPGRADED" nach „VALID" wechseln. Dieses Feature, „Replay Upgrade" genannt, wird durch zwei Eigenschaften der Datenbank gesteuert *(siehe Listing 2)*.

Data Guard: PDB Recovery Isolation

In vielen Fällen wird eine Produktionsdatenbank mit einem Data Guard abgesichert. Leider waren bislang Data Guard und Multitenant nicht die besten Freunde. Das hat sich zwar im Laufe der Versionen von Oracle 12c bis 19c schon gebessert, aber es blieb ein Problem beim Kopieren einer PDB im laufenden Betrieb („hot-cloning einer read-write PDB"): Der MRP-Prozess auf der Standby-Seite brach mit der Fehlermeldung „ORA-1153 an incompatible media recovery is active" ab. Der Grund: Beim Data Guard läuft ein permanentes Recovery und beim Kopieren einer PDB macht Oracle am Ende ein Recovery auf der kopierten PDB, um die Transaktionen, die während des Kopierens angefallen sind, nachzufahren. Zwei Recoveries zur gleichen Zeit, das ging nicht. In Oracle 21c gibt es dazu mit „PDB Side Recovery" beziehungsweise „PDB recovery isolation" ein neues Feature, das dieses Problem löst. Mit Active Data Guard läuft dabei alles automatisch:

- Die PDB wird auf der Standby-Seite als „disabled" angelegt. Damit wird sie vom normalen Recovery ausgeschlossen.
- Das Recovery der PDB erfolgt in einer separaten Session („PDB side recovery").
- Die PDB wird „enabled" und dann vom normalen MRP-Prozess recovered.

Im alert.log sieht dieser Vorgang wie folgt aus *(siehe Listing 3)*.

```
alter database property set UPGRADE_PDB_ON_OPEN='true'
Completed: alter database property set UPGRADE_PDB_ON_OPEN='true'
alter database property set CONVERT_NONCDB_ON_OPEN='true'
Completed: alter database property set CONVERT_NONCDB_ON_OPEN='true'
```

Listing 2: Datenbank-Eigenschaften für „Replay Upgrade"

```
2022-06-19T14:14:54.280230+02:00
Recovery created pluggable database DGTEST
DGTEST(4):Tablespace-SYSTEM during PDB create offlined since source is in
r/w mode or this is a refresh clone
DGTEST(4):File #17 added to control file as '/u01/oradata/DGCDB_S2/
DGTEST/system01.dbf'.
[...]
2022-06-19T14:15:48.129967+02:00
DGTEST(4):.... (PID:5215): Side Recovery Complete [krds.c:1584]
2022-06-19T14:16:08.462159+02:00
all data files of pdbid 4 are brought online.
 Started logmerger process
```

Listing 3: Recovery einer neuen PDB auf der Standby-Seite

```
oracle@statler:~/ [MUPPETS1] crsctl status resource -t -w "TYPE = ora.
pdb.type"
--------------------------------------------------------------Name
Target  State        Server                   State details
--------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------
ora.muppets.muppetpdb.pdb
      2        ONLINE  ONLINE       statler                  STABLE
      3        ONLINE  ONLINE       waldorf                  STABLE
--------------------------------------------------------------
```

Listing 4: PDBs als Cluster-Ressource

```
 SQL> CREATE MANDATORY PROFILE C##ALL_USER_PROFILE
  2  LIMIT PASSWORD_VERIFY_FUNCTION
  3  ora12c_stig_verify_function
  4  PASSWORD_GRACE_TIME 7
  5  CONTAINER=ALL;
Profile created.

SQL> ALTER SYSTEM SET mandatory_user_profile=
  2  'C##ALL_USER_PROFILE';
System altered.
```

Listing 5: Mandatory-Profile

```
DEDICATED_THROUGH_BROKER_LISTENER=ON
```

Listing 6: Notwendige Änderungen in der listener.ora

```
SQL> ALTER SYSTEM SET use_dedicated_broker=TRUE;
SQL> ALTER SYSTEM SET dbnest_enable=cdb_resource_pdb_all SCOPE=SPFILE;
```

Listing 7: Notwendige Parameter-Änderungen für die Datenbank-Instanz

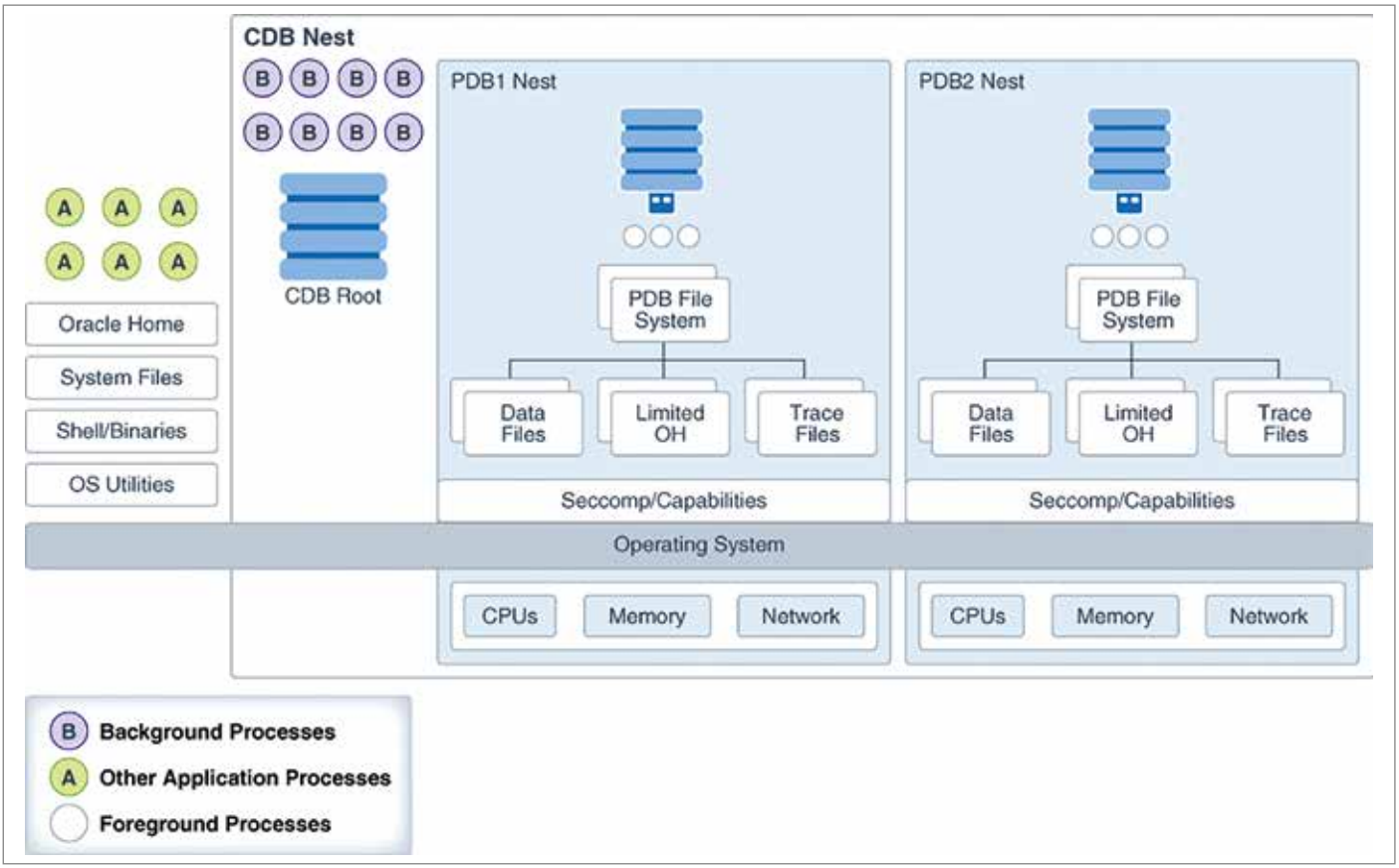


Abbildung 1: Architektur von DB Nest (Quelle: Oracle Database 21c Security Guide)

Ohne Active Data Guard ist der Prozess etwas aufwendiger:

- Die Dateien der PDB werden auf die Standby-Seite kopiert.
- Auf der Standby-Seite wird die PDB mit „DISABLED AUTOMATIC RECOVERY" markiert.

Das heißt, die PDB ist noch nicht in das automatische Recovery aufgenommen, aber – wichtiger – der MRP-Prozess und damit das Recovery der vorhandenen PDBs läuft weiter. Damit die neue PDB in das Data-Guard-Recovery aufgenommen wird, sind nur zwei weitere Schritte erforderlich:

- Recovery der PDB „from Service" (d. h. der Primary-Datenbank)
- Aktivieren des Recovery für die PDB

PDBs in der Grid Infrastructure

Ein häufiger Kritikpunkt an Container-Datenbanken im Real Application Cluster war bislang, dass die PDBs nicht als Ressource im Cluster sichtbar waren. Ein „crsctl status resource" zeigte alles an – nur nicht die PDBs. Dieses Problem ist behoben, es gibt einen neuen Ressourcen-Typ „ora.pdb.type" *(siehe Listing 4)*.

Das Werkzeug „srvctl" wurde entsprechend erweitert, sodass jetzt auch Befehle wie „srvctl start pdb", „srvctl stop pdb" etc. möglich sind. Neue PDBs werden auch automatisch als Cluster-Ressourcen angelegt. Allerdings können nur 21c-PDBs als Cluster-Ressourcen verwaltet werden.

Verbesserte Passwort-Sicherheit mit „Mandatory Profiles"

In den Datenbanken kann jetzt ein Profil hinterlegt werden, mit dem Passwort-Kriterien für alle PDBs beziehungsweise Benutzer definiert werden können. Aktuell können allerdings nur die Funktion zur Passwort-Prüfung und die „Grace Time" festgelegt werden *(siehe Listing 5)*.

Wird der statische Parameter MANDATORY_PROFILE in CDB$ROOT gesetzt, dann gilt dieses Profil für alle PDBs. Auf PDB-Ebene kann ein weiteres Profil gesetzt werden, das dann für alle Benutzer in dieser PDB gilt. Wenn es sowohl in CDB$ROOT als auch in der PDB ein entsprechendes Profil gibt, dann hat das Profil mit den schärferen Einschränkungen Priorität (Beispiel: höhere Mindestlänge des Passwortes).

Pluggable-Datenbanken im Nest

„DB Nest" ist ein neues Feature, mit dem die „Containerisierung" bei den Datenbanken angekommen ist. Bei den bekannten Container-Technologien, wie etwa Docker, werden Funktionen des Betriebssystems, wie zum Beispiel cgroups, namespaces oder secure computing, zur Isolierung der einzelnen Container und zur Beschränkung des Zugriffs auf Ressourcen des Betriebssystems genutzt. Mit „DB Nest" ist dies nun auch für Datenbanken auf Linux x86-64 möglich.

Ein „Nest" ist eine „Laufzeit-Umgebung" für Datenbanken *(siehe Abbildung 1)*.

Im „Parent Nest" der CDB liegen dabei die Nester der PDBs PDB1 und PDB2. Über secure computing wird der Zugriff auf OS-Ressourcen eingeschränkt.

```
2022-06-19T16:44:55.468015+02:00
DB Nest is enabled, SGA file deletion disabled.
Starting ORACLE instance (normal) (OS id: 25266)
2022-06-19T16:44:55.497281+02:00
Instance running inside DB Nest (ORA_DEMO21CA_DEMO21CA)
2022-06-19T16:44:55.501905+02:00
[...]
DEMOPDB(5):Creating (DEMOPDB) Nest for PDB(5)
2022-06-19T16:46:05.386947+02:00
[...]
DEMOPDB(5):DB Nest (PDB00005, 1102414643) open successful
2022-06-19T16:46:05.776135+02:00
```

Listing 8: Alert.log beim Start mit DB Nest

```
oracle@abel:~/ [DEMO21CA] dbnest list
--------------------------------------------------------------------------------
 Id : Nest             : Parent           : : Tag              :    State
--------------------------------------------------------------------------------
  1 : ORA_DEMO21CA_DEMO21CA :                  :  ORA_DEMO21CA_DEMO21CA :     OPEN
      Net State        :
      Namespace State  : (pid=0,cnid=4026531836,pnid=4026531836,no namespace,type=0x0)
      Resources        : (cpu=0)
      Property enabled : resources
      Seccomp status   : (level=none)
      FS Isolation     : (disabled)
--------------------------------------------------------------------------------
 [...]
--------------------------------------------------------------------------------
  4 : PDB00005         : ORA_DEMO21CA_DEMO21CA :  DEMOPDB (uid=1102414643) :     OPEN
      Net State        :
      Namespace State  : (pid=26885,cnid=4026532186,pnid=4026531836,type=0x7)
      Resources        : (cpu=0)
      Property enabled : namespaces,resources
      Seccomp status   : (level=strict1)
      FS Isolation     : (default-config)
--------------------------------------------------------------------------------
Number of active nest namespaces = 4
```

Listing 9: Anzeige mit „dbnest list“

Für die Konfiguration von DB Nest gibt es zwei Parameter: DB_NEST_ENABLE und DBNEST_PDB_FS_CONF. Zusätzlich muss ein dedizierter Broker definiert werden *(siehe Listing 6)*.

Nach den in *Listing 6 und 7* beschriebenen Änderungen müssen Listener und Datenbank-Instanz durchgestartet werden. Dabei ist zu beachten, dass OS-Authentisierung mit DB Nest nicht funktioniert. Sprich: Es ist eine Passwort-Datei erforderlich und die Anmeldung erfolgt mittels „sqlplus sys/syspassword as sysdba“.

In der alert.log wird auch protokolliert, dass die Datenbank mit DB Nest gestartet wurde *(siehe Listing 8)*.

Mit dem neuen Befehl „dbnest“ kann man sich die vorhandenen „Nester“ anzeigen lassen *(siehe Listing 9)*.

Der Befehl dbnest ist leider nicht dokumentiert, sodass ausgehend von der Hilfefunktion mit „dbnest -h“ viel Forschungsarbeit erforderlich ist, um erste Einblicke in dieses Tool zu bekommen *(siehe Listing 10)*.

Leider werden viele der Optionen aktuell (Release-Stand RU 21.6) noch mit Fehlermeldungen quittiert.

Der statische Parameter DBNEST_PDB_FS_CONF dient dazu, anzugeben, auf welche Verzeichnisse aus der CDB heraus nicht zugegriffen werden kann („Blacklist“, siehe *Listing 8 und 9*) beziehungsweise welche verfügbar sein sollen *(siehe Listing 11)*.

Insgesamt ist die Dokumentation von Oracle zu DB Nest sehr knapp gehalten. Gerade einmal elf Seiten umfasst die Beschreibung im Security Guide (*[3]*). Auch in der Oracle-Blog-Welt wird das Thema eher stiefmütterlich behandelt. Am ausführlichsten sind dabei noch die Beiträge unter *[4]*.

Zusammengefasst

Nach acht Jahren mit Multitenant ist die Abschaffung der klassischen Non-CDB-Architektur ein lang erwarteter Schritt von Oracle. Mit den Verbesserungen im Cluster (PDB als Cluster-Ressource) und bei Data Guard („PDB Side Recovery“) hat Oracle einige Kritikpunkte an den vorherigen Releases beseitigt. Zudem werden neben „Auto Upgrade“ durch „Replay Upgrade“ der Wechsel in die Container-Welt

```
oracle@abel:~/ [DEMO21CA] dbnest -h
 Usage : dbnest <command> [options]

 List of options and commands.

 [...]
  update <nest> [options]                  Do required modifications to
                                          nest.
    --cpu <cpu count>                     Number of cpus required for nest
    --cpuids <list>                       Comma delimited cpuid range/list
                                          e.g: 0,1-2 1-3,5 2-3 1-3,5-7 2,3
    --cpu_excl <1|0>                      Alloc the CPUs exclusively or not
                                          1 : alloc exclusive
                                          0 : alloc shared (default)
    --cpu_from_end <1|0>                  Start CPU allocation from the end
                                          1 : alloc from end
                                          0 : alloc from cpu0 (default)
    --cpushares <cpushares>               CPU shares for this nest
    --max_mem <maximum memory>            Max memory (in MB) for the nest
    --max_swap <maximum memory>           Max swap (in MB) for the nest
    --cfs_period <period>                 CFS Period for the nest
    --cfs_quota <quota>                   CFS Quota for the nest
    --cpu_rt_min <micro_seconds>          Minimum cpu.rt_runtime_us value
    --cpu_rt_req <micro_seconds>          Required cpu.rt_runtime_us value
[...]
```

Listing 10: Optionen von dbnest (Auszug)

und das Versionsupgrade vereinfacht. „DB Nest" ist ein technisch sehr interessantes Feature. Es ist leider nur sehr spärlich dokumentiert und erfordert viel eigene Recherche, wenn man sich damit vertraut machen möchte. Produktionsreif ist es meiner Einschätzung nach aktuell noch nicht.

```
SQL> host cat /u00/app/oracle/admin/DEMOCDB/nest_blacklist.txt
DBNEST_NO_FS_ROOT_MODE
/bin
/usr/bin

SQL> ALTER SYSTEM SET DBNEST_PDB_FS_CONF=
 2    '/u00/app/oracle/admin/DEMOCDB/nest_blacklist.txt'
 3    SCOPE=SPFILE;
System altered.
```

Listing 11: Beispiel für die „Blacklist-Variante"

Quellen und Referenzen

[1] Zahlreiche Blog-Artikel zu AutoUpgrade gibt es auf mikedietrichde.com
[2] MOS-Note 2642908.1 „Standby CDB continuity – PDB Side recovery"
[3] „Oracle Database 21c Security Guide" – Chapter 15 – Informationen zu DB Nest
[4] *https://mahmoudhatem.wordpress.com*

Über den Autor

Markus Flechtner ist seit mehr als 25 Jahren im Oracle-Umfeld tätig. Nach einer Tätigkeit als Entwickler wechselte er in den DBA-Bereich. Als Berater spezialisierte er sich auf den Real Applicaton Cluster sowie Upgrade- und Migrationsprojekte. Zudem gibt er zahlreiche Schulungen rund um die Oracle-Datenbank. Er ist Co-Autor des Buches „Der Oracle DBA" und häufiger Sprecher auf nationalen und internationalen Konferenzen. Seit 2020 ist er Oracle ACE. Bei der DOAG ist er gewählter Delegierter für die jährliche Delegiertenversammlung und Themenverantwortlicher für Open-Source-Datenbanken in der Datenbank Community.

Markus Flechtner
mfl@ordix.de

Viele Wege führen nach Rom – 11g- und 12c-Migrationen nach 19c Multitenant

Raphael Salguero Aragón, ORDIX

In diesem Artikel werden verschiedene Migrationswege von den älteren Oracle Releases 11g und 12c in Richtung 19c (aktuelles Longterm-Release) aufgezeigt. Unabhängig von der Ausgangsarchitektur wird ein Wechsel in die Multitenant-Architektur vorgenommen. Dabei sollen die Vor- und Nachteile einzelner Methoden in Abhängigkeit von bestimmten Rahmenbedingungen beleuchtet werden. Wichtig: Die einzelnen Methoden werden technisch nicht bis ins kleinste Detail erklärt, dafür ist der Umfang nicht ausreichend. Fragen können daher gerne direkt an den Autor adressiert werden.

Zunächst eine wichtige Einschränkung: Es gibt zahlreiche Mittel und Wege, wie wir eine Datenbank aus 11g/12c als PDB mit 19c zur Verfügung stellen können. Der vorliegende Artikel stellt einen Erfahrungsbericht aus einem Projektumfeld dar, in dem es vor allem darum ging, möglichst viele Datenbanken in kürzester Zeit zu migrieren und dabei auch die Hardware beziehungsweise Plattform zu wechseln. Neben einer tatsächlichen Cloud-Lösung kamen auch verschiedene interne Umgebungen zum Einsatz. Dadurch fallen aber grundsätzlich einige Migrationsmethoden durch das Raster.

Für die nachfolgenden Beschreibungen sind daher folgende Rahmenbedingungen wichtig:

- Hardwarewechsel
- 19c Multitenant als Zielarchitektur
- Möglichst überschaubarer Aufwand
- Kein Endian-Wechsel
- Einfache Reproduzierbarkeit

Wie schaut Rom aus?

Wie bereits erwähnt ist mit der Oracle-Version 19c und der Multitenant-Architektur die Zielumgebung bereits vorgegeben. Um den notwendigen Aufwand weiter reduzieren zu können, wird die Ziel-CDB bereits mit dem Zeichensatz AL32UTF8 zur Verfügung gestellt.

Zusätzlich war durch das Projektumfeld vorgegeben, dass sich die Lokalität der Zielarchitektur zwischen einer tatsächlichen Cloud-Lösung und einer On-Premises-Umgebung unterscheiden kann. Daher sollten die verwendeten Migrationsmethoden beide Wege abdecken können.

Die Wege nach Rom

Da sowohl die Rahmenbedingungen als auch die Zielarchitektur nun bekannt sind, geht es nachfolgend um die möglichen sinnvollen Migrationsmethoden.

Dabei steht bei nahezu jeder Ausgangssituation die Variante *Data Pump* zur Auswahl.

Die Stärken des Data Pump – oder vielmehr der beiden Tools *expdp* und *impdp* –liegen ganz klar in der Flexibilität. Ob es sich bei der Quell-Datenbank um eine 11g, 12c, PDB oder NonCDB handelt, spielt für die tatsächliche Migration zunächst eine untergeordnete Rolle. Viel wichtiger sind hierbei einzelne Einschränkungen, wie beispielsweise Abhängigkeiten von Objekttypen. Dazu jedoch später mehr.

Neben dem *Data Pump* standen noch die Methoden des *Unplug/Plug-In* und des *Clone* zur Verfügung. Beide Varianten haben einen immensen Vorteil gegenüber Data Pump, wenn es um den allgemeinen Aufwand geht. Dafür ist hier aber eine Quell-Datenbank der Version 12.1 beziehungsweise 12.2 notwendig.

Die Variante *Unplug/Plug-In* funktioniert logischerweise nur mit bereits vorhandenen PDBs, weshalb diese Methode

also neben der Quellversion auch abhängig von der Quellarchitektur ist.

Der *Clone* hingegen funktioniert sowohl mit einer PDB als auch mit einer NonCDB. Dafür hat sich in der Praxis allerdings gezeigt, dass entgegen der offiziellen Dokumentation der Clone einer Instanz mit 12.1 nicht fehlerfrei funktioniert *(siehe auch MOS 2090019.1)*.

Geht es um die Auswahl des passenden Weges, hat sich die in *Abbildung 1* dargestellte Entscheidungshilfe bewährt.

Bei einer Quellversion <= 12.1 wird grundsätzlich der *Data Pump* ausgewählt. Bei 12.2 unabhängig von der Architektur der *Clone* und in Spezialfällen auch die Variante *Unplug/Plug-In*, wenn es sich bereits um eine PDB handelt, die migriert werden muss.

Abbildung 1 zeigt zusätzlich notwendige Vor- und Nacharbeiten, die bei der Auswahl der entsprechenden Migrationsmethode zu berücksichtigen sind. Darauf wird bei der Betrachtung der einzelnen Migrationsmethode genauer eingegangen.

Bestandsaufnahme/ Prechecks

Bevor mit der eigentlichen Migration gestartet werden kann, sollte im Vorfeld eine Reihe von *Prechecks* durchgeführt werden. Zu diesen Prechecks gehören neben allgemeinen Informationen (Version, Brutto- bzw. Netto-Größe) auch spezielle Informationen, wie zum Beispiel:

- Auflistung der vorhandenen/verwendeten Komponenten
- Character Set
- Datentypen
- Partitionen
- Time Zone
- Datenbanklinks
- Directories

Wozu sind diese Informationen wichtig? Während die Größe der Datenbank selbstverständlich einen Indikator für die Migrationsdauer liefern kann, engt das Dasein von Large Objects (LOB) oder RAW-Datentypen in Kombination mit einer 11g-Datenbankversion die Migrationsmethode weiter ein beziehungsweise reduziert die Flexibilität.

Auch Objekte wie Database Links und Directories spielen eine Rolle, da die Voraussetzungen auf dem Zielsystem hierfür auch gegeben sein müssen (Firewall bezüglich DB Link und Dateisystem bezüglich der Directories).

Diese Punkte sollten im Vorfeld betrachtet und gegebenenfalls mit dem Kunden diskutiert werden, bevor der finale Migrationsweg festgelegt wird.

Der steinige Weg von 11g/12.1

Für die Migration einer 11g-Datenbank nach 19c Multitenant kommt unserer Entscheidungshilfe nach nur *Data Pump* infrage, was exemplarisch demonstriert wird.

Die Erstellung der passenden PDB inklusive des passenden Zeichensatzes und der notwendigen Komponenten für eine Migration via *Data Pump* wird in diesem Artikel als gegeben angesehen.

Zunächst werden daher die zu migrierenden Schemata ausgewählt und anschließend auf dem Zielsystem bereitgestellt. Dafür müssen die benötigten Tablespaces erstellt werden. Nun kann mithilfe eines DDL-Skriptes *(https://tinyurl.com/userddl)* der Benutzer 1:1 in einer leeren Ziel-PDB erstellt werden.

Selbstverständlich kann der Benutzer auch während des eigentlichen Imports erstellt werden. Bei Migrationsprojekten ist jedoch oftmals eine frühzeitige Bereitstellung der Benutzer sinnvoll, damit die Anwendung ihre Verbindungen testen kann. So können frühzeitig Firewall- oder Clientprobleme identifiziert werden.

Im Nachgang sollten nun notwendige Datenbankobjekte (Synonyme, DB-Links, Verzeichnisse etc.) analog zum Quellsystem angelegt werden, bevor die eigentliche Datenmigration stattfindet.

Data Pump liefert nun grundsätzlich 2 Möglichkeiten, die Daten in das Zielsystem zu importieren:

- Import via Datenbanklink
- Export -> Datentransfer (scp, cp, ...) -> Import

Für den Import via Datenbanklink (1) muss lediglich ein Benutzer mit ausreichenden Berechtigungen auf dem Quellsystem erstellt werden. Für diesen Benutzer wird auf dem Zielsystem nun ein passender Datenbanklink angelegt, der über einen hinterlegten Eintrag in der tnsnames.ora auf das Quellsystem zeigt:

```
CREATE DATABASE LINK DB_LINK CONNECT
TO remote_dp_user IDENTIFIED BY "remo-
te_dp_user123!remote" USING '11G_XE';
```

Hiermit sind nun die wichtigsten Vorbereitungen für den Import via Datenbanklink getroffen.

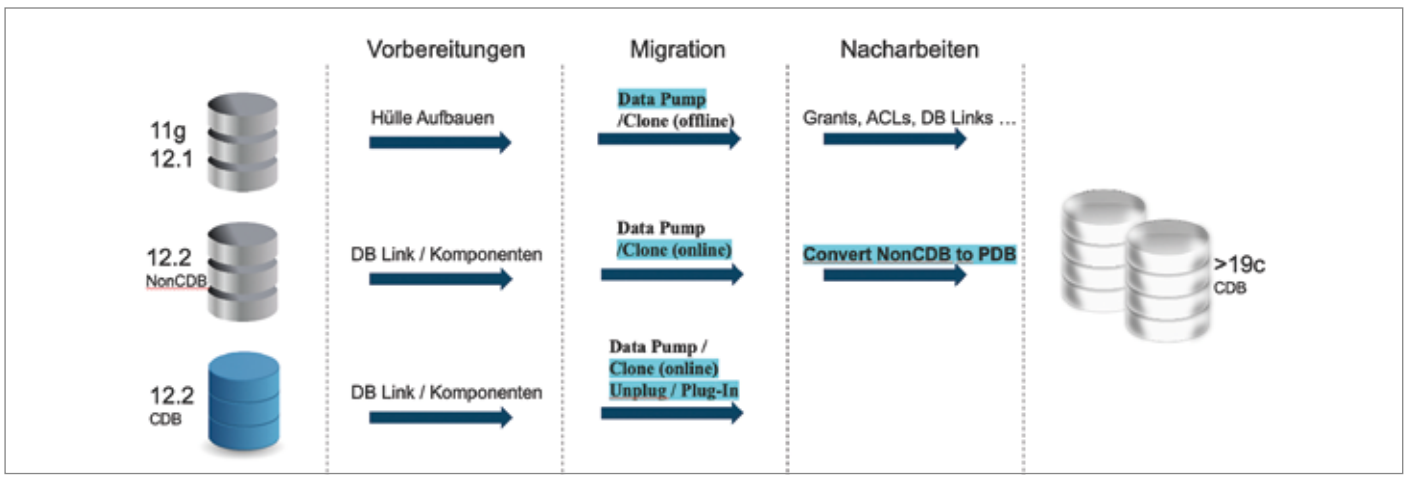


Abbildung 1: Entscheidungshilfe für verschiedene Migrationsmethoden (© Raphael Salguero Aragón, ORDIX AG)

```
OWNER_LOB   ANZAHL
----------- ----------
PM          5
IX          2
```

Listing 1: Anzahl von LOB-Datensätzen pro Schema

```
ORA-31693: Table data object "PM"."PRINT_MEDIA" failed to load/unload
and is being skipped due to error:
ORA-00942: table or view does not exist
ORA-02063: preceding line from DB_LINK
```

Listing 2: ORA-31693 bei Import eines LOB-Datensatzes

Für die zweite Variante ist zwar kein spezieller Benutzer/Datenbanklink notwendig, dafür müssen aber sowohl auf dem Quell- als auch auf dem Zielsystem die Directories für die Export-Dumps zur Verfügung gestellt werden. An dieser Stelle spielt die Größe der zu migrierenden Datenbank eine entscheidende Rolle.

Während bei der ersten Variante die Daten direkt über das Netzwerk migriert werden, werden sie bei der Alternativvariante zuerst lokal exportiert, dann auf das Zielsystem transferiert und von dort aus importiert.

Grundsätzlich führen beide Wege zum erklärten Ziel. Unter Umständen kann es jedoch dazu kommen, dass Datensätze vom Typ LOB über den Datenbanklink nicht migriert werden können. Leider konnte hier keine allgemeingütige Gesetzmäßigkeit erkannt werden, sodass das Motto „Try&Error" ausgepackt werden musste. Folgendes Beispiel basiert auf den beiden Schemata PM und IX mit fünf und zwei LOB-Datensätzen *(siehe Listing 1)*.

Während alle IX-Datensätze problemlos migriert werden, wirft der Import beim PM-Schema die Fehlermeldung ORA-31693 *(siehe Listing 2)*.

Ein Vergleich auf dem Zielsystem zeigt schnell, dass die LOB-Datensätze fehlen *(siehe Listing 3)*.

Hier hilft nur der „klassische" *Data-Pump-Weg* ohne Nutzung des DB-Links *(siehe Listing 4)*.

Die Migration von 11g nach 19c in die Multitenant-Umgebung funktioniert zusammenfassend mit *Data Pump* sehr gut, wobei der Aufwand stark von der Komplexität, Größe und den verwendeten Datentypen der Quelle abhängt. In einem Migrationsprojekt sind solche Migrationskandidaten daher immer mit erheblichem Mehraufwand verbunden.

Der kieselige Weg von 12.2 Non-CDB

Für eine Migration von einer 12.2 Non-CDB-Datenbank eignet sich neben dem *Data Pump* vor allem die Variante des *Clone*.

Während Logik und Strukturen der Datenblöcke für den Data Pump eine große Rolle spielen, arbeitet der Clone eher vergleichbar mit einem scp-Kommando und kopiert die Datafiles über das Netzwerk auf das Zielsystem.

Dadurch gewinnt der Clone einen enormen Geschwindigkeitsvorteil gegenüber dem Data Pump und reduziert die Komplexität, da keine Rücksicht auf verschiedene Datentypen genommen werden muss.

Im Vorfeld der Migration muss ein Datenbanklink vom Zielsystem auf das Quellsystem erstellt werden. Hierzu ist es ratsam, auf der Quelldatenbank zunächst einen Benutzer mit den notwendigen Rechten zu erstellen.

Daraufhin kann auf dem Zielsystem der Link angelegt werden.

Der eigentliche Clone wird über das Kommando CREATE PLUGGABLE DATABASE von der Zielseite aus gestartet. Notwendig ist hier neben der Angabe des Datenbanklinks auch gegebenenfalls der Parameter FILE_NAME_CONVERT, falls das Zielsystem ohne Oracle Managed Files konfiguriert ist. Die Angabe der Parallelität ist optional. Wichtig ist jedoch, dass die Quelldatenbank mit dem Alias NON$CDB angesprochen wird *(siehe gelbe Markierung in Listing 7)*.

Die Non-CDB-Datenbank ist nach diesem Kommando erfolgreich auf dem Zielsystem verfügbar und muss nun im nächsten Schritt von einer Non-CDB in eine PDB konvertiert werden. Dazu muss sichergestellt werden, dass sich die PDB im UPGRADE-Modus befindet, nachfolgend werden die Skripte *noncdb_to_pdb.sql* sowie *utlrp.sql* ausgeführt *(siehe Listing 8)*.

Da es sich noch immer um eine Datenbank der Version 12.2 im Mantel einer Version 19 handelt, muss abschließend ein Upgrade der PDB durchgeführt werden. Auch hierzu wird die PDB im UPGRADE-Modus belassen und via dbupgrade oder autoupgrade das Upgrade gestartet.

Danach wird ein letztes Mal via *utlrp.sql* sichergestellt, dass keine invaliden Komponenten zurückbleiben, und optional der OPEN_MODE der PDB für zukünftige Neustarts gespeichert.

Der Weg mit Rückenwind von 12.2 CDB

Handelt es sich bei der Quelldatenbank bereits um eine PDB der Version 12.2, bietet sich ebenfalls der Clone an. Hierbei wird fast analog der zuvor beschriebene Weg verfolgt. Lediglich drei Ausnahmen sind zu nennen:

- Für den Datenbanklink kann hier auch ein Common-Benutzer in der CDB erstellt werden, der die notwendigen Berechtigungen für alle Container zugewiesen bekommt *(siehe Listing 11)*.
- In der Syntax CREATE PLUGGABLE DATABASE wird nun direkt der Service der PDB angesprochen *(siehe Listing 12)*.
- Die Konvertierung via *noncdb_to_pdb.sql* entfällt.

Grundsätzlich empfehle ich jedem, den Clone als Migrationsmethode zu testen. Ist die Quelle entsprechend konfiguriert *(siehe nachfolgenden Abschnitt „Hot oder Cold Clone")*, lässt sich die Migration ohne große Auswirkungen auf das Quellsystem testen und wir sparen uns Fehlerquellen, die vor allem beim Einsatz „seltener" Features im Data Pump auftreten können.

```
select count(*) from pm.print_media;

  COUNT(*)
----------
    0
```

Listing 3: Anzahl der LOB-Datensätze im PM-Schema nach Import via DB-Link

```
select count(*) from pm.print_media;

  COUNT(*)
----------
    4
```

Listing 4: Anzahl der LOB-Datensätze im PM-Schema nach Import aus Dump

```
CREATE USER remote_clone_user IDENTIFIED BY "XYZ";
GRANT CREATE SESSION, CREATE PLUGGABLE DATABASE TO remote_clone_user;
```

Listing 5: DDL des Migrationsbenutzers

```
CREATE DATABASE LINK clone_link CONNECT TO remote_clone_user IDENTI-
FIED BY "XYZ " USING '122_NONCDB';
```

Listing 6: DDL des Datenbanklinks für die Migration

```
CREATE PLUGGABLE DATABASE ORCLPDB2 FROM NON$CDB@clone_link
FILE_NAME_CONVERT = ('/opt/oracle/oradata/ORCLCDB/ORCLPDB1','/opt/
oracle/oradata/ORCLCDB/ORCLPDB2')
PARALLEL 1;
```

Listing 7: Hot-Clone-Befehl für eine NON-CDB

```
alter pluggable database ORCLPDB2 open upgrade;
alter session set container = ORCLPDB2;
@?/rdbms/admin/noncdb_to_pdb.sql
@?/rdbms/admin/utlrp.sql
```

Listing 8: Konvertierung einer NON-CDB zur PDB nach Hot Clone

```
$ORACLE_HOME/bin/dbupgrade -c "ORCLPDB2"
```

Listing 9: Upgrade der 12.2 PDB nach 19c

Hot Clone oder Cold Clone

Der Unterschied zwischen einem Hot Clone und einem Cold Clone liegt in der Verfügbarkeit der Quelldatenbank. Grundsätzlich empfehle ich aus Migrationssicht in jedem Fall, die Quelldatenbank in den Read-Only-Modus zu setzen, damit während der Migration nicht versehentlich Datensätze verändert werden und der Migrationszeitpunkt genau definiert ist (Cold Clone).

Besitzt die Quelldatenbank ein ausreichend großes UNDO-Tablespace, kann die Quelle auch unverändert im Read-Write-Modus verbleiben (Hot Clone).

Ist die Quelle bereits eine PDB, ist das LOCAL UNDO MANAGEMENT eine Voraussetzung für den Hot Clone. Jede zu migrierende PDB benötigt ihr eigenes UNDO Tablespace für die fortlaufenden Transaktionen.

Unplug – eine weitere Alternative für 12.2 PDB

Es mag Konstellationen geben, in denen eine Netzwerkverbindung zwischen der Quelle und dem Ziel nicht gegeben ist und somit auch die Variante des Clone hinfällig wird.

In dieser Situation bietet es sich an, die Quell-PDB als Archiv-PDB „auszupluggen", auf das Ziel zu transportieren und sie dort „einzupluggen". Dies ist auch eine Option, wenn der Status quo zum Zeitpunkt der Migration gesichert werden muss.

Ein gravierender Nachteil bei dieser Variante ist allerdings der notwendige Speicherplatz.

In der Praxis hat es sich bewährt, die Quell-PDB vor dem „unplug" zu kopieren, um die Ausgangssituation in jedem Fall wiederherstellen zu können. Dadurch wird der Festplattenbedarf schon in diesem Schritt verdoppelt. Für die Kopie bringen wir die PDB optimalerweise in den Read-Only-Modus und nutzen wieder das Kommando CREATE PLUGGABLE DATABASE *(siehe Listing 13)*.

Im nächsten Schritt schließen wir die Kopie der PDB und pluggen sie als Archive-PDB aus *(siehe Listing 14)*.

Diese Archive-PDB muss nun auf das Zielsystem kopiert und dort „eingepluggt" werden *(siehe Listing 15)*.

Abhängig von der OMF-Konfiguration muss auch in diesem Schritt noch der Parameter FILE_NAME_CONVERT angegeben werden. Eine Parallelisierung ist hier ebenfalls möglich.

Nachfolgend sind ab hier die Schritte wieder identisch zum Clone (Upgrade, utlrp, ...), dauern aber in der Summe ein Vielfaches länger, da sämtliche Daten mehrfach bewegt werden. Diese Methode ist daher nur eine Alternative.

Data Pump vs. Clone

Im Vergleich beider Varianten hat der Clone in puncto Geschwindigkeit und Auf-

```
@?/rdbms/admin/utlrp.sql
alter pluggable database ORCLPDB2 save state;
```

Listing 10: Fehlende Nacharbeiten nach Hot Clone, Konvertierung und Upgrade

```
CREATE DATABASE LINK clone_link CONNECT TO c##remote_clone_user IDENTI-
FIED BY "XYZ " USING '122_PDB';
```

Listing 11: Datenbanklink für den Hot Clone einer 12.2 PDB

```
CREATE PLUGGABLE DATABASE ORCLPDB2 FROM ORCLPDB2@clone_link
FILE_NAME_CONVERT = ('/opt/oracle/oradata/ORCLCDB/ORCLPDB1','/opt/
oracle/oradata/ORCLCDB/ORCLPDB2')
PARALLEL 1;
```

Listing 12: Hot Clone einer 12.2 PDB

```
alter pluggable database ORCLPDB1 close immediate;
alter pluggable database ORCLPDB1 open read only;
create pluggable database CP_OF_ORCLPDB1 from ORCLPDB1
       file_name_convert=(<AKTUELLE_PFAD>,<NEUER_PFAD>)
       parallel 4;
```

Listing 13: Erstellen einer PDB-Kopie für den Unplug

```
alter pluggable database ORCLPDB1 close immediate;
alter pluggable database ORCLPDB1 unplug into '<PFAD>/CP_OF_ORCLPDB1.
pdb';
```

Listing 14: Unplug einer PDB

```
create pluggable database ORCLPDB2 as clone using '<PFAD>/CP_OF_OR-
CLPDB1.pdb';
```

Listing 15: Plug-In einer PDB

wand die Nase deutlich vorn. Trotzdem ist er aufgrund der Versionseinschränkungen noch kein gängiger Weg für viele Migrationsprojekte.

Data Pump hingegen ist als Werkzeug den meisten DBAs bereits bekannt und nahezu unabhängig von der Quellversion verfügbar. Der Vor- und Nachbereitungsaufwand ist allerdings meistens deutlich größer, liefert dafür jedoch eine zusätzliche Flexibilität (z.B. Reorganisation, Schema-Mapping, ...).

Am Ende hängt die Entscheidung vor allem auch von zwei Faktoren ab:

- Womit fühlt sich der DBA sicher und wohl?
- Welche zeitlichen Vorgaben gibt es (Tests, Downtimes etc.)?

Spielen Zeit und Vorkenntnisse gegen den DBA, so sollte – soweit möglich – der Clone verwendet werden. Themen wie Reorganisation können und sollten grundsätzlich in einem separaten Schritt vollzogen werden, wenn der Fokus erst mal auf der Migration an sich liegt.

An der Stelle kann ich nur empfehlen, die Trennung auch deutlich zu kommunizieren und zu fordern. Ein Migrationsprojekt mit einer Vielzahl von Quelldatenbanken inklusive einer parallelen Reorganisierung erhöht nur unnötig die Komplexität und gleichzeitig auch das Risiko, dass eine abgesprochene Downtime nicht eingehalten werden kann.

Denn am Ende muss auch ein DBA beachten, dass die Datenbanken zwar oftmals den Kern einer Anwendung darstellen, aber trotzdem nur eines von vielen Rädern einer Anwendung abbilden. Da eine Datenbankmigration/ein Datenbankupgrade jedoch Auswirkungen auf die gesamte Anwendung haben, muss auch jede Perspektive berücksichtigt werden. Und das bedeutet am Ende eigentlich nur eins – zusätzlichen Aufwand.

Da mit dem Wechsel nach 19c zukünftig zu jeder Zeit ein Hot Clone möglich sein wird (zumindest theoretisch), setze ich große Hoffnungen darauf, dass zukünftige Migrationsprojekte durch diesen Weg deutlich vereinfacht werden und am Ende des Tages niemand auf den langen Export einer LOB-Spalte warten muss.

Abgrenzung

Ich habe in diesem Artikel bewusst nicht jeden technischen Schritt beschrieben. Dafür bestehen zu viele Abhängigkeiten von den jeweiligen Quellversionen und Systemen. Außerdem hätte ich den Rahmen des Artikels niemals einhalten können.

Daher aber umso mehr der Hinweis, dass Sie sich bei Fragen zu diesem Thema jederzeit bei mir melden können.

Über den Autor

Raphael Salguero Aragón hat sein duales Studium bei der ORDIX AG absolviert und arbeitet seit 2017 als Datenbankadministrator in diversen Projekten. Neben Oracle gehört auch die Open-Source-Datenbank MySQL zu seinen Kerngebieten. Seit 2018 teilt Raphael regelmäßig Erfahrungen aus der Praxis in Form von Präsentationen auf diversen Veranstaltungen.

Raphael Salguero Aragón
ras@ordix.de

Oracle Database 21c und Database In-Memory

Markus Kißling, Oracle Global Services Germany

Oracle Database In-Memory kommt als Turbo bei analytischen Workloads und Mixed Workload Environments auch in großen Umgebungen zum Einsatz. Dieser Artikel beschreibt die New Features in Oracle Database 21c. Da Oracle Database In-Memory auch in der kostenlosen Oracle Database Express Edition (XE) enthalten ist, lassen sich die hier vorgestellten New Features auch unter XE ausführen. Zudem steht Database In-Memory mit dem neuen Base Level Feature allen Anwendern der Oracle Database Enterprise Edition kostenlos zur Verfügung.

Oracle-Exadata-Anwender profitieren auch in 21c vom In-Memory-Format auf dem Exadata Flash Storage, dem sogenannten CellMemory.

Sehr gute Kandidaten für Oracle Database In-Memory sind das klassische Reporting im Data Warehouse und das in BI-Umgebungen. Oracle stellt als einziger Hersteller am Markt das duale Format bereit *(siehe Abbildung 1)*. Dieses ermöglicht den direkten Zugriff auf die operativen Daten durch die gleichzeitige Nutzung des Row Store für den OLTP-Betrieb (im Zeilenformat) und des In-Memory Column Store für analytische Abfragen (im Spaltenformat). Deshalb steht der Ausführung von analytischen Abfragen in Echtzeit nichts im Wege. Dies ermöglicht auch gewachsenen Anwendungen völlig neue Ansätze bei der zukünftigen Anwendungsentwicklung. Oracle Database In-Memory als Option der Oracle Database Enterprise Edition hat dadurch ein Alleinstellungsmerkmal am Markt, da die Daten unmittelbar nach der transaktionellen Verarbeitung im In-Memory Column Store verfügbar sind, ohne diese zuerst in separate Reporting-Datenbanken übertragen zu müssen. Die Sichtweise auf die Daten ist immer transaktional konsistent. Oracle Database In-Memory stellt für den Oracle Optimizer eine weitere Zugriffsart dar und kann, vereinfacht gesagt, mit der

Einführung einer neuen „Index-Art“ verglichen werden.

Die Aktivierung von Database In-Memory ist denkbar einfach und wird über nur wenige Parameter und Objekt-Attribute konfiguriert. Der Speicherbedarf wird über den INMEMORY_SIZE-Parameter gesetzt. Die Datenbank-Instanz muss danach neu durchgestartet werden. Anschließend werden die gewünschten Objekte (Tabellen, Partitionen, Sub-Partitionen und Materialized Views) mit dem ALTER TABLE-Kommando INMEMORY gesetzt.

Oracle Database 21x Express Edition (XE)

Als Schnell-Einstieg dient der Blog-Artikel unter *[1]*. Anhand des Artikels kann eine komplette Umgebung zum Testen von Database In-Memory mit der kostenlosen Oracle Database 21c XE innerhalb weniger Minuten aufgebaut werden. Die Beispieldaten und eine Abfrage *(siehe Listing 1)* sind dort enthalten und können ganz einfach in der eigenen Umgebung eingespielt werden. Der Vorteil des In-Memory Column Store im Vergleich zum Row Store ist selbst bei kleinen Datenmengen deutlich erkennbar.

Oracle Database In-Memory-Innovationen

Oracle Database In-Memory wurde im Juli 2014 mit dem ersten Patchset der Datenbank-Version 12.1 (12.1.0.2) eingeführt. Über die Jahre sind zahlreiche neue Funktionen in diese Option der Oracle Enterprise Edition eingeflossen. Neben beschleunigten klassischen Abfragen auf strukturierte Daten werden auch Abfragen auf unstrukturierte Daten unterstützt. Beide Abfrage-Arten können auch miteinander kombiniert werden (Stichwort: Converged Database).

Abbildung 2 veranschaulicht die Entwicklungs-Iterationen von Database In-Memory mit den unterschiedlichen Entwicklungs-Schwerpunkten.

Die Basisfunktionen des In-Memory Column Store und der neuen Zugriffsalgorithmen für den Oracle Optimizer für die In-Memory-Verarbeitung stellen die sogenannten Foundation Features dar *(siehe Abbildung 2)*. Diese wurden

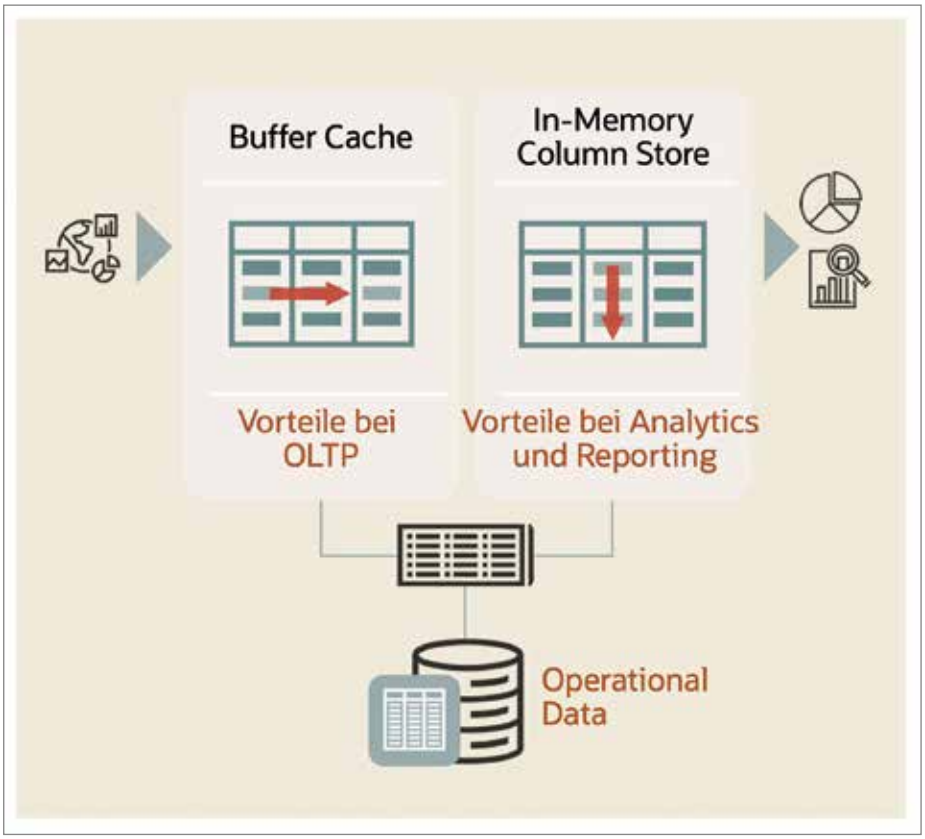


Abbildung 1: Das duale Format (Quelle: Oracle)

mit Oracle 12.1.0.2 eingeführt und werden ständig weiter optimiert. So fließen Erfahrungen aus Kundenprojekten anhand deren Workloads direkt in die neuen Datenbank-Releases beziehungsweise neuen Release Updates ein und werden – wenn möglich – auch auf die Vor-Versionen der Datenbank zurück portiert.

Double Buffering stellt ein solches Beispiel dar. Hier wird beim Aktualisieren der In-Memory Column Units (IMCU) nach DML-Operationen die Original IMCU weiterhin für In-Memory-Abfragen verwendet, bis die neue IMCU vollständig aufgebaut ist.

Im Folgenden werden die New Features in Oracle Database 21c näher betrachtet.

Das „Self Management“ des In-Memory Column Store und „Covergence“ (im Sinne eines Zusammenführens mehrerer Technologien und Paradigmen) bildet in diesem Release den Schwerpunkt.

Automatic In-Memory (AIM)

Mit 21c und Automatic In-Memory bewegt sich die Option immer mehr in Richtung „Self Managing“ und „Autonomous“.

```
Select sum(lo_extendedprice * lo_discount) revenue
from
  LINEORDER l,
  DATE_DIM d
where
  l.lo_orderdate = d.d_datekey
  and l.lo_discount between 2 and 3
  and l.lo_quantity < 24
  and d.d_date='December 24, 1996';

In-Memory Column Store - Elapsed: 0.009 seconds
Row Store (Buffer Cache) - Elapsed: 0.868 seconds
```

Listing 1: Analytische Abfrage, gegen In-Memory Column Store und Row Store (Buffer Cache) ausgeführt – Beispiel aus Blog-Artikel unter [1].

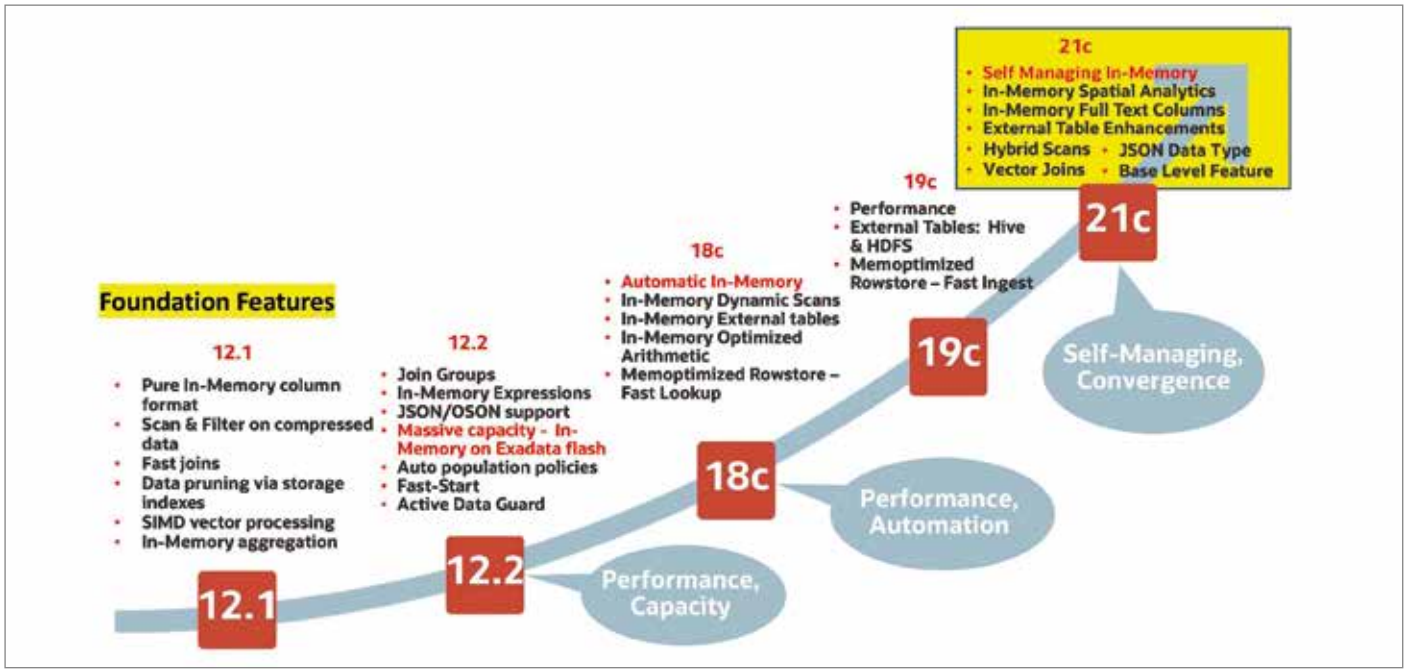


Abbildung 2: Oracle Database In-Memory-Innovationen (Quelle: Markus Kißling)

Mit dem neu eingeführten AIM Level „HIGH" werden alle Objekte automatisch für die In-Memory-Verarbeitung und das Laden in den In-Memory Column Store in Betracht gezogen, sobald analytische Abfragen auf diese erfolgen. Dies betrifft die Applikationstabellen und nicht die Tabellen des Data Dictionary. Standardmäßig ist AIM ausgeschaltet und wird durch das Setzen des Parameters IN_MEMORY_AUTOMATIC_LEVEL auf „HIGH" aktiviert.

Damit werden dann die am häufigsten verwendeten Objekte automatisch in den In-Memory Column Store geladen. Bei Bedarf, wenn kein Speicherplatz mehr vorhanden ist, werden die weniger beziehungsweise nicht benutzten Objekte und deren Spalten zuerst mit einer höheren Kompression versehen und bei weiterer Nicht-Verwendung dann aus dem In-Memory Column Store entfernt.

Das Ziel ist, immer die aktiven Objekte („Hot Data") in den In-Memory Column Store zu laden und dort zu halten. Dadurch profitieren die kritischen Geschäftsabfragen „out of the box" und automatisch von der In-Memory-Verarbeitung. Der Anwender beziehungsweise der DBA muss sich nicht mehr selbst um das Beladen der Objekte kümmern. Die Funktionsweise von AIM ist in *Abbildung 3* skizziert.

Wie bereits in früheren Releases können Objekte weiterhin vom Anwender manuell auf INMEMORY gesetzt oder auch komplett von der In-Memory-Verarbeitung ausgeschlossen werden.

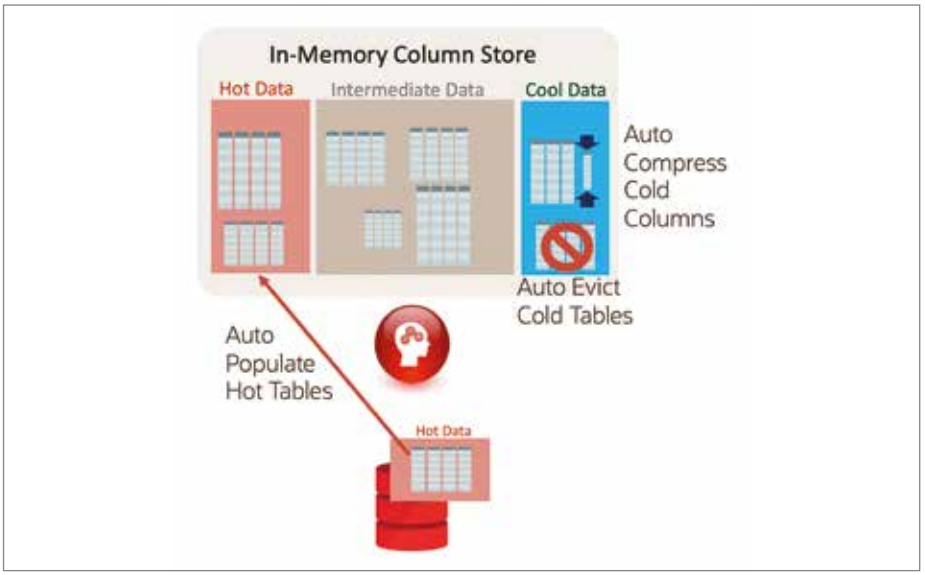


Abbildung 3: Die Funktionsweise von Automatic In-Memory (AIM) (Quelle: Markus Kißling)

Folgende AIM-Parameter sind in Oracle Database 21c verfügbar.

```
IN_MEMORY_AUTOMATIC_LEVEL = OFF
(Default) / LOW / MEDIUM / HIGH
```

In Oracle Database In-Memory sind folgende In-Memory Compression Levels verfügbar, neu in 21c wurde „AUTO" eingeführt.

- NONE (keine Komprimierung)
- MEMCOMPRESS FOR DML
- MEMCOMPRESS FOR QUERY LOW (Default)
- MEMCOMPRESS FOR QUERY HIGH
- MEMCOMPRESS FOR CAPACITY LOW
- MEMCOMPRESS FOR CAPACITY HIGH
- MEMCOMPRESS AUTO (neu in 21c bei Nutzung von AIM)

Dass AIM in 21c zum Einsatz kommt, kann man daran erkennen, dass ein Objekt automatisch auf den Compression Level „AUTO" gesetzt wurde *(siehe Abbildung 4)*. Dies bedeutet neben dem automatischen Laden in den IM Column Store auch, dass die Objekte und gegebenenfalls deren Columns basierend auf deren Nutzung und dem Workload mit

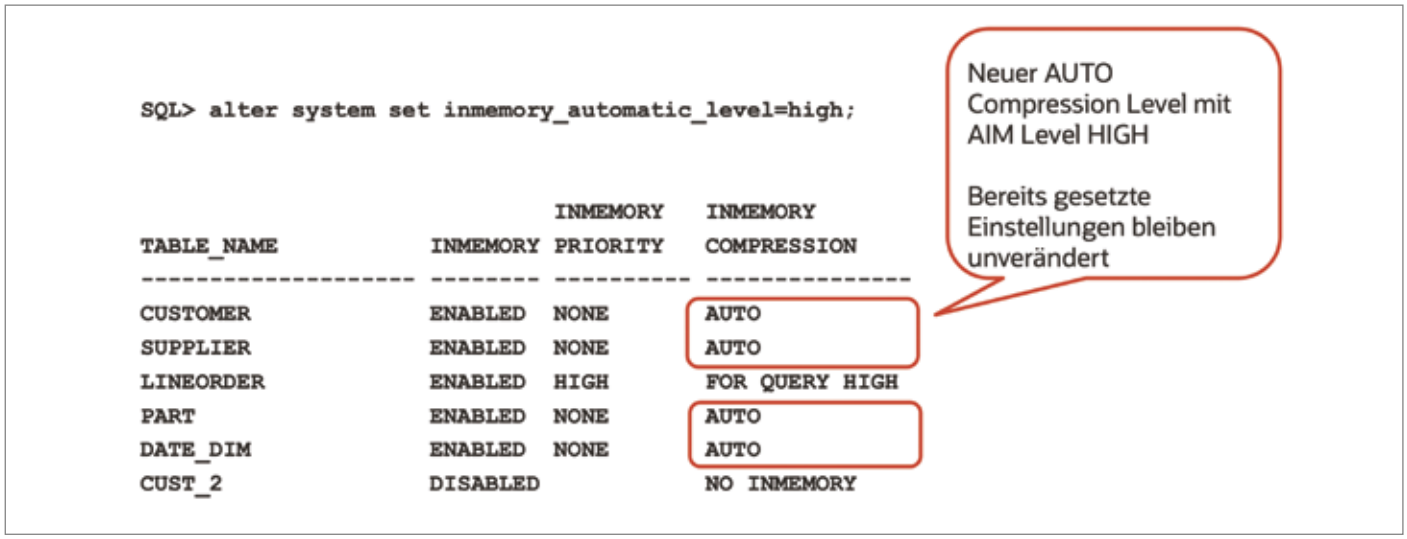


Abbildung 4: Automatic In-Memory (AIM) aktiviert (Quelle: Markus Kißling)

der passenden Komprimierung versehen werden – und dies zur Laufzeit.

Zudem gibt es die Data Dictionary Views DBA_INMEMORY_AIMTASKS und DBA_INMEMORY_AIMTASKDETAILS, über die sich die AIM-Aktivitäten monitoren lassen.

In-Memory External Tables

Die In-Memory-Unterstützung von Partitioned External Tables und Hybrid Partitioned Tables in 21c erfüllt den Wunsch zahlreicher Anwender.

Dadurch kann vom Anwender bestimmt werden, welche Partitionen in den In-Memory Column Store geladen werden sollen und welche nicht. Bei den Hybrid Partitioned Tables kann die Verwendung von internen Partitionen und externen Quellen (z.B. CSV-Dateien), die als externe Partitionen eingebunden sind, miteinander kombiniert werden.

Der Hauptvorteil: Die Daten müssen vor deren Verwendung nicht zuerst in der Oracle-Datenbank aus den externen Quellen materialisiert werden, sondern sie werden direkt in den IM Column Store im Spaltenformat geladen und können von dort verwendet werden. Durch das Einbinden der nur für In-Memory relevanten Partitionen lässt sich der In-Memory-Speicher sehr effizient nutzen.

Damit die externen Datenquellen verwendet werden können, muss der folgende Session-Parameter gesetzt werden:

alter session set QUERY_REWRITE_INTEGRITY = stale_tolerated;

Das folgende Beispiel zeigt die Verwendung von Hybrid Partitioned Tables *(siehe Abbildungen 5 bis 7)*.

In-Memory Spatial

Entwickler profitieren bei der Nutzung von In-Memory Spatial dadurch, dass

```
ALTER TABLE city_points ADD (shape SDO_GEOMETRY);
```

Listing 2: Spatial Geometry Column anfügen

```
CREATE TABLE sh_sales_hybrid_part ( column_list )
EXTERNAL PARTITION ATTRIBUTES (
       TYPE ORACLE_LOADER
       DEFAULT DIRECTORY
       ACCESS PARAMETERS ( ) )
     REJECT LIMIT UNLIMITED
partition by range ( time_id )
( partition part_1998 values less than ( to_date( '01-JAN-1999', 'DD-MON-YYYY') ) EXTERNAL LOCATION ('sh_sales.dat'),
  partition part_1999 values less than ( to_date( '01-JAN-2000', 'DD-MON-YYYY') ) EXTERNAL LOCATION ('sh_sales.dat'),
  partition part_2000 values less than ( to_date( '01-JAN-2001', 'DD-MON-YYYY') ) EXTERNAL LOCATION ('sh_sales.dat'),
  partition part_2001 values less than ( to_date( '01-JAN-2002', 'DD-MON-YYYY') ),
  partition part_2002 values less than ( to_date( '01-JAN-2003', 'DD-MON-YYYY') ),
  partition part_2003 values less than ( to_date( '01-JAN-2004', 'DD-MON-YYYY') ) )
```

Abbildung 5: Hybrid Partitioned Table erzeugen (Quelle: Markus Kißling)

```
alter session set QUERY_REWRITE_INTEGRITY = stale_tolerated;

select to_char(time_id,'YYYY'), count(*) from sh_sales_hybrid_part group by
to_char(time_id,'YYYY');

TO_C            COUNT(*)
---- --------------------
1998                873945
1999               1049292
2000               1125576
2001                873945
2002               1049292
2003               1125576
```

Abbildung 6: Abfrage auf Hybrid Partitioned Table (Quelle: Markus Kißling)

```
-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |       |       |   263 (100)|          |       |       |
|   1 |  HASH GROUP BY                     |                     |  1403K|    12M|   263   (4)| 00:00:01 |       |       |
|   2 |   PARTITION RANGE ALL              |                     |  1403K|    12M|   262   (3)| 00:00:01 |     1 |     6 |
|   3 |    TABLE ACCESS HYBRID PART INMEMORY FULL| SH_SALES_HYBRID_PART |  1403K|    12M|   262   (3)| 00:00:01 |     1 |     6 |
|   4 |     TABLE ACCESS INMEMORY FULL     | SH_SALES_HYBRID_PART |       |       |            |          |     1 |     6 |
-----------------------------------------------------------------------------------------------------------------------
```

Abbildung 7: Hybrid Partitioned Table – Ausführungsplan bei In-Memory-Zugriff (Quelle: Markus Kißling)

Spatial-SDO-Filter-Operationen (SDO_FILTER) auf Spatial-Tabellen direkt im In-Memory Column Store erfolgen. Zu beachten ist, dass die Verwendung der In-Memory Virtual Columns aktiviert sein muss. Des Weiteren muss das In-Memory Expression Framework aktiviert sein. Dies erfolgt über den Initialisierungsparameter INMEMORY_EXPRESSIONS_USAGE, der auf ENABLE zu setzen ist.

Die neuen Möglichkeiten lassen sich anhand des folgenden kurzen Beispiels leicht nachvollziehen.

Zuerst wird eine Spatial Geometry Column an eine Tabelle angefügt *(siehe Listing 2)*.

Die neue Column in der Tabelle wird dann mit den geometrischen Objekten basierend auf den Locations-Koordinaten aktualisiert.

Zudem müssen die Spatial-Metadaten über die View user_sdo_geom_metadata erweitert werden. Die Details dazu sind im Database In-Memory Guide unter *[2]* zu finden.

Listing 3 enthält den Befehl, über den die Tabelle als Kandidat für den In-Memory Column Store gekennzeichnet wird. Über die Schlüsselworte INMEMORY SPATIAL (shape) wird die Column „shape“, die die geometrische Spatial Column darstellt, in den In-Memory Column Store geladen. Hier kommen auch In-Memory Expressions zum Einsatz, in denen die Spatial-typischen Funktionen für alle enthaltenen Rows in der Tabelle vorberechnet in den In-Memory Virtual Columns abgespeichert werden.

Anschließend kann dann über die geometrischen Punkte mit Unterstützung von In-Memory Spatial selektiert werden *(siehe Listing 4)*.

Ein Vorteil in 21c ist, dass kein Spatial-Index mehr separat erstellt werden muss, der an einer anderen Stelle in der Datenbank abgespeichert wird. Zudem kommt die volle Scan-Performance des In-Memory Column Store zum Einsatz.

In-Memory Full Text Columns

Für die Nutzung von In-Memory Text Columns müssen ebenfalls die In-Memory Virtual Columns aktiviert sein (INMEMORY_VIRTUAL_COLUMNS = ENABLE). Mit 21c können dann einzelne Columns oder Listen von Columns, die Text-Daten enthalten, mit der INMEMORY TEXT Clause versehen werden. Dies kann mit dem

```
ALTER TABLE city_points INMEMORY PRIORITY high INMEMORY SPATIAL
(shape);

EXEC DBMS_INMEMORY.POPULATE('chicago','city_points');
```

Listing 3: Tabelle „city_points“ als Kandidat für In-Memory markieren und die Column „shape“ als Spatial Geometry Column für die In-Memory-Verarbeitung versehen

```
SELECT city_name FROM city_points c where sdo_filter(c.shape,
   sdo_geometry(2001,8307,sdo_point_type(-122.453613,37.661791,null),nu
ll,null)) = 'TRUE';
```

Listing 4: Selektion der geometrischen Punkte

```
select count(*) from chicago_text where district = '009' and CONTAINS(description, 'BATTERY', 1) > 0;

-----------------------------------------------------------------------------------------
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |              |       |       | 11236 (100)|          |
|   1 |  SORT AGGREGATE             |              |     1 | 18399 |            |          |
|*  2 |   TABLE ACCESS INMEMORY FULL| CHICAGO_TEXT |  26937 |   472M| 11236   (75)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   2 - inmemory(("DISTRICT"='009' AND SYS_CTX_CONTAINS2("DESCRIPTION" /*+ LOB_BY_VALUE */  USING '"CHICAGO"."DESC_SEARCH_POLICY"' ,
              'BATTERY' , SYS_CTX_MKIVIDX("DESCRIPTION" /*+ LOB_BY_VALUE */  RETURNING RAW(32767)))>0))
       filter(("DISTRICT"='009' AND SYS_CTX_CONTAINS2("DESCRIPTION" /*+ LOB_BY_VALUE */  USING '"CHICAGO"."DESC_SEARCH_POLICY"' ,
              'BATTERY' , SYS_CTX_MKIVIDX("DESCRIPTION" /*+ LOB_BY_VALUE */  RETURNING RAW(32767)))>0))
```

Abbildung 8: Selektion einer In-Memory Text-Column und der In-Memory-Ausführungsplan mit den Filter-Prädikaten (Quelle: Markus Kißling)

CREATE TABLE- oder ALTER TABLE-Befehl erfolgen.

Zu beachten ist, dass für die Verwendung der In-Memory Text Columns die MAX_STRING_SIZE auf EXTENDED gesetzt werden muss. Damit wird das Byte-Limit für VARCHAR2, NVARCHAR2 und RAW Columns auf 32 KB erhöht.

Zusätzlich ist eine Custom Indexing Policy über den Aufruf von CTX_DDL.CREATE_POLICY zu erzeugen.

Beim Laden der Tabelle in den In-Memory Column Store wird dann der Text-Index in die In-Memory Virtual Columns geladen und die Text-Daten können im Anschluss wie gewohnt abgefragt werden. Somit entfällt die Notwendigkeit, den Text-Index manuell zu erzeugen.

Abbildung 8 veranschaulicht den Zugriff auf eine In-Memory Text-Column und den dazugehörigen Ausführungsplan. Die Filter-Prädikate veranschaulichen die Nutzung der für den Text-Index relevanten Details, die in In-Memory Virtual Columns gespeichert sind.

In-Memory JSON Data Type

Mit 21c wurde der JSON-Datentyp eingeführt, der auch mit Oracle Database In-Memory eingesetzt werden kann. In diesen neuen Datentyp werden JSON-Daten bis zu 8 KB inline gespeichert, größere out-of-line. Oracle verwendet hier das optimierte und komprimierte Binärformat namens OSON.

Sind die Objekte für In-Memory aktiviert, werden bei der Inline-Variante zusätzliche In-Memory Expression Units (IMEUs) innerhalb des In-Memory Column Store aufgebaut. Diese IMEUs enthalten zusätzliche optimierte In-Memory JSON-Index-Strukturen und ermöglichen eine zusätzliche In-Memory Scan-Performance durch die Verwendung des SIMD Vector Processing. Wie bei In-Memory Spatial muss der Initialisierungsparameter INMEMORY_EXPRESSION_USAGE auf ENABLE gesetzt werden.

Im folgenden Beispiel *(siehe Abbildung 9)* wird das JSON-Dokument als JSON-Datentyp abgespeichert und unter der Verwendung von JSON_EXIST selektiert.

Dem Ausführungsplan *(siehe Abbildung 10)* ist der In-Memory-Zugriff zu entnehmen. Die Prädikat-Informationen verdeutlichen, dass das OSON-Format genutzt wird und über die Prädikat-Informationen, die in In-Memory Virtual Columns im In-Memory Column Store gespeichert sind, zugegriffen wird.

Im Database In-Memory Guide *[2]* sind die weiteren Möglichkeiten mit OSON Out-of-Line-Daten und den anderen möglichen Datentypen, die ihm Rahmen von

```
SQL> desc J_PURCHASEORDER
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL VARCHAR2(32)
 DATE_LOADED                                        TIMESTAMP(6) WITH TIME ZONE
 PO_DOCUMENT                                        JSON


SELECT COUNT(1)
FROM
  j_purchaseorder
WHERE
  json_exists(po_document,'$.ShippingInstructions?(@.Address.zipCode == 99236)')
```

Abbildung 9: Selektion einer JSON-Datentyp Column (Quelle: Markus Kißling)

In-Memory verwendet werden können, ausführlich beschrieben.

In-Memory Zugriffsoptimierung mit In-Memory Vector Joins und In-Memory Hybrid Scans

Ziel bei der Entwicklung der In-Memory Vector Joins war es, Hash Joins beim Scannen der Tabellen weiter zu beschleunigen. Durch die Verwendung der SIMD Vector Instructions moderner Intel- und SPARC-Prozessoren können mehrere Rows gleichzeitig in deren CPU-Vector-Register geladen und verarbeitet werden. Die In-Memory-Verarbeitung umfasst das Scannen, das Filtern der Join-Prädikate und die Berechnung der Werte der zu aggregierenden Spalten. Durch die Nutzung der SIMD Vector Instructions und CPU-Vector-Register können mit den In-Memory Vector Joins Milliarden von Rows pro Sekunde verarbeitet werden.

In 21c ist dieses Feature über den Initialisierungsparameter INMEMORY_DEEP_VECTORIZATION bereits aktiviert. Ausschalten lässt sich das Feature durch Setzen des Parameters auf „FALSE“.

Die Motivation bei der Entwicklung der In-Memory Hybrid Scans war es, den Speicherplatz des In-Memory Column Store noch effizienter zu nutzen und Columns beziehungsweise Listen von Columns, die selten in Abfragen genutzt werden, vom Beladen in den In-Memory Column Store auszuschließen.

Ein kleines Beispiel *(siehe Abbildung 11)* soll die Verwendung von In-Memory Hybrid Scans darstellen. Über den „alter table sales no inmemory (invoice)“-Befehl wird die Column „invoice“ vom Beladen in den In-Memory Column Store ausgeschlossen.

In Releases vor 21c konnten einzelnen Columns beziehungsweise Listen von Columns mit NO INMEMORY auch schon ausgeschlossen werden.

Allerdings hat dann eine Selektion auf eine solche ausgeschlossene Column beziehungsweise Liste von Columns dazu geführt, dass ein In-Memory-Zugriff auf eine solche Tabelle nicht möglich war. Der Zugriff erfolgte in diesem Fall immer komplett gegen den Row Store (Buffer Cache). In diesem Beispiel hätte das Selektieren der nicht in den In-Memory Column Store geladenen Column zur Folge, dass die komplette Abfrage gegen den Buffer Cache ausgeführt würde.

Mit 21c ist dies nun möglich. Basierend auf der Column „Price“, die das Filter-Prädikat darstellt („Price > 1000“), und dem Zugriff auf den entsprechenden Wert der Column „Invoice“ erfolgt dann im Anschluss der Abfrage der Daten noch eine Art Lookup der entsprechenden Werte aus der Column „Invoice“. Sie werden aus dem Row Store (Buffer Cache) abgefragt. Es ist noch anzumerken, dass die Columns, die in den Joins verwendet werden und nach denen gefiltert wird, zwingend in den In-Memory Column Store geladen sein müssen, um In-Memory Hybrid Scans verwenden zu können.

Zu erkennen ist die Verwendung der In-Memory Hybrid Scans im Ausführungsplan innerhalb der Operation TABLE ACCESS INMEMORY FULL (HYBRID).

Beim Vergleich der Performance eines reinen In-Memory-Scans im In-Memory Column Store mit dem Scan gegen den Buffer-Cache-Scan bewegen wir uns zeitlich in der Mitte. Wir profitieren hier immer noch von der enormen Geschwindigkeit des In-Memory-Scans, auch wenn im Anschluss Columns aus dem Buffer Cache per Lookup über die Row-ID zu laden sind.

In-Memory Base Level Feature

Das Base Level Feature wurde mit Oracle Database 21c eingeführt und zurück nach Oracle Database 19c portiert. Es ist ab dem Release Update 19.8 enthalten.

Anwender der Oracle Enterprise Edition können den In-Memory Column Store bis zu einer Größe von 16 GB nutzen, ohne Database In-Memory lizenzieren zu müssen *(siehe Abbildung 12)*.

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                 |       |       |     1 (100)|          |
|   1 |  SORT AGGREGATE             |                 |     1 |  4102 |            |          |
|*  2 |   TABLE ACCESS INMEMORY FULL| J_PURCHASEORDER |     6 | 24612 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   2 - inmemory(JSON_EXISTS2("PO_DOCUMENT" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.ShippingInstructions?(@.Address.zipCode == 99236)' FALSE ON ERROR)=1)
       filter(JSON_EXISTS2("PO_DOCUMENT" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.ShippingInstructions?(@.Address.zipCode == 99236)' FALSE ON ERROR)=1)
```

Abbildung 10: In-Memory-Ausführungsplan, Zugriff auf JSON-Datentyp Column (Quelle: Markus Kißling)

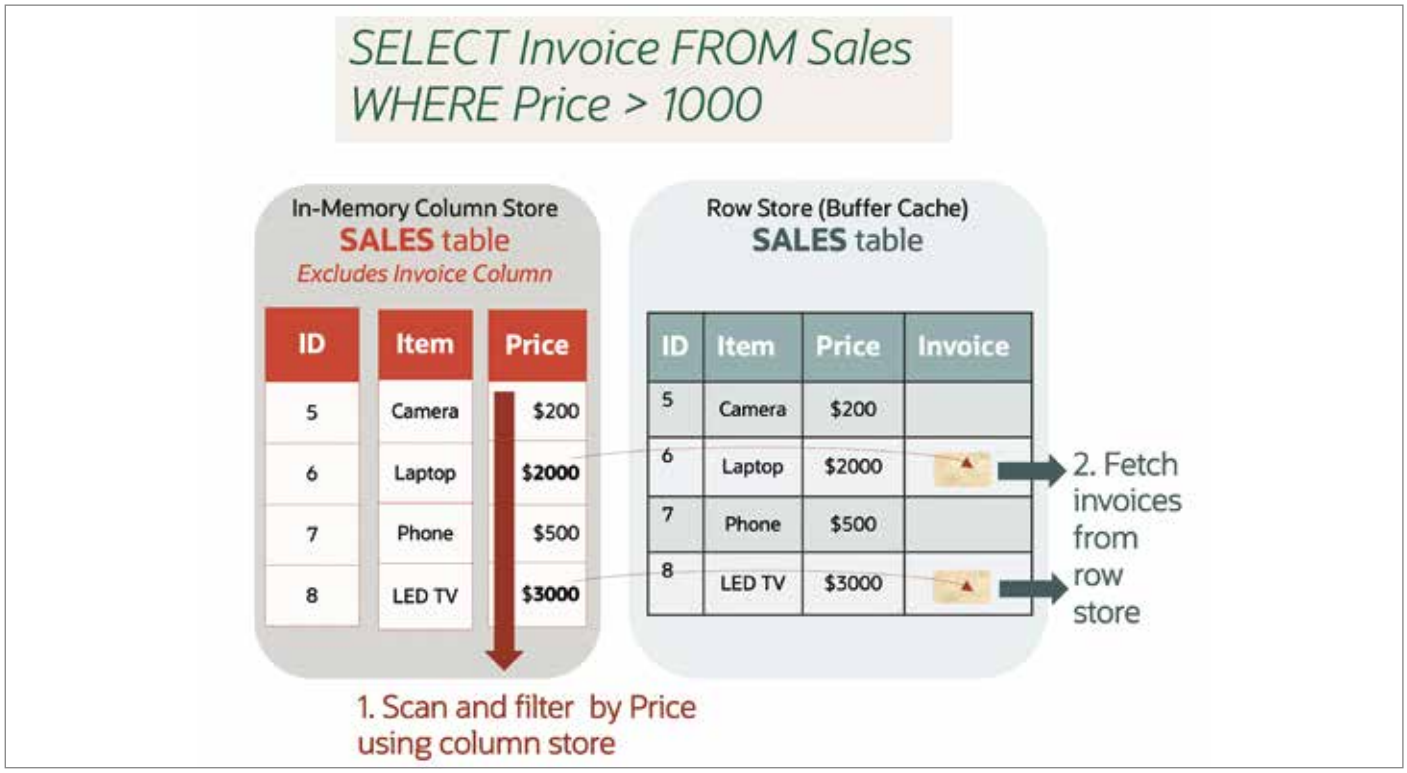


Abbildung 11: In-Memory Hybrid Scans ermöglichen die Verwendung von Columns aus dem Row Store (Quelle: Markus Kißling)

Um das Feature zu aktivieren, muss der Initialisierungsparameter INMEMORY_FORCE auf BASE_LEVEL gesetzt werden. Der INMEMORY_SIZE-Parameter kann dann auf einen Wert bis maximal 16 GB pro Instanz gesetzt werden.

Bei RAC-Umgebungen stehen jeweils 16 GB pro Instanz zur Verfügung. Wird ein Wert über 16 GB angegeben, wird eine Fehlermeldung beim Hochfahren der Datenbank ausgegeben.

Zu berücksichtigen sind die Einschränkungen, dass keine Columns vom Beladen in den In-Memory Column Store ausgeschlossen werden können und dass nur der Compression Level QUERY LOW verwendet werden kann. AIM steht hier nicht zur Verfügung und das CellMemory Feature auf Exadata ist deaktiviert.

Exadata Flash Cache (CellMemory)

Exadata-Anwendern steht das In-Memory-Format auch auf dem Exadata Flash Storage zur Verfügung. Dadurch können Terabytes an Daten für die In-Memory-Verarbeitung genutzt werden. CellMemory steht schon seit Oracle Database 12.2 *(siehe Abbildung 2)* zur Verfügung und nicht erst mit 21c und soll hier der Vollständigkeit halber erwähnt werden. Die Basis für CellMemory stellt die Exadata Storage Software dar, die mit der Datenbank-Software Hand-in-Hand zusammenarbeitet.

Die Prozesse auf den Exadata-Storage-Servern wandeln die Datenbank-Blöcke, die unkomprimiert, OLTP-komprimiert oder HCC-komprimiert auf Platte vorliegen können, zur Laufzeit in das In-Memory-Format um, sobald das System analytische Abfragen identifiziert. CellMemory baut auf der Exadata-Smart-Scan-Technology auf und so werden nur die wenigen von den Storage-Servern gefilterten Blöcke an die Datenbank (Compute Nodes) zurückgegeben – unter Nutzung des In-Memory-Formates, das für analytische Abfragen optimiert ist.

Der In-Memory Column Store, der sich im DRAM der Datenbank (Compute Node) befindet, und der Flash-Speicher auf den Exadata-Storage-Servern können auch gleichzeitig genutzt werden *(siehe Abbildung 13)*.

Anwender können aber auch ausschließlich das In-Memory-Format im Flash der Exadata-Storage-Server (CellMemory) verwenden.

Dazu müssen folgende Initialisierungsparameter gesetzt werden:

INMEMORY_FORCE = CELLMEMORY_LEVEL
INMEMORY_SIZE = 0

CellMemory ist auf allen Exadata-Plattformen verfügbar:

- On-Premises
- Exadata Cloud Service
- Exadata Cloud at Customer

Auf Exadata lassen sich die In-Memory Column Stores redundant mit In-Memory Duplication auf den unterschiedlichen RAC-Knoten aufbauen. Die In-Memory-Umgebung kann damit hochverfügbar betrieben werden. Auf einer Exadata-Active-Data-Guard-Umgebung lassen sich speziell für das Reporting In-Memory Column Stores auf dem Standby-System aufbauen. Dadurch lässt sich der OLTP-Betrieb auf Primärknoten ohne Auswirkungen durch die analytischen Abfragen betreiben, da diese isoliert auf der Standby-Umgebung ausgeführt werden.

Fazit

Mit der Implementierung von Automatic In-Memory (AIM) wurde einer der größten Kundenwünsche nach Automatisierung berücksichtigt.

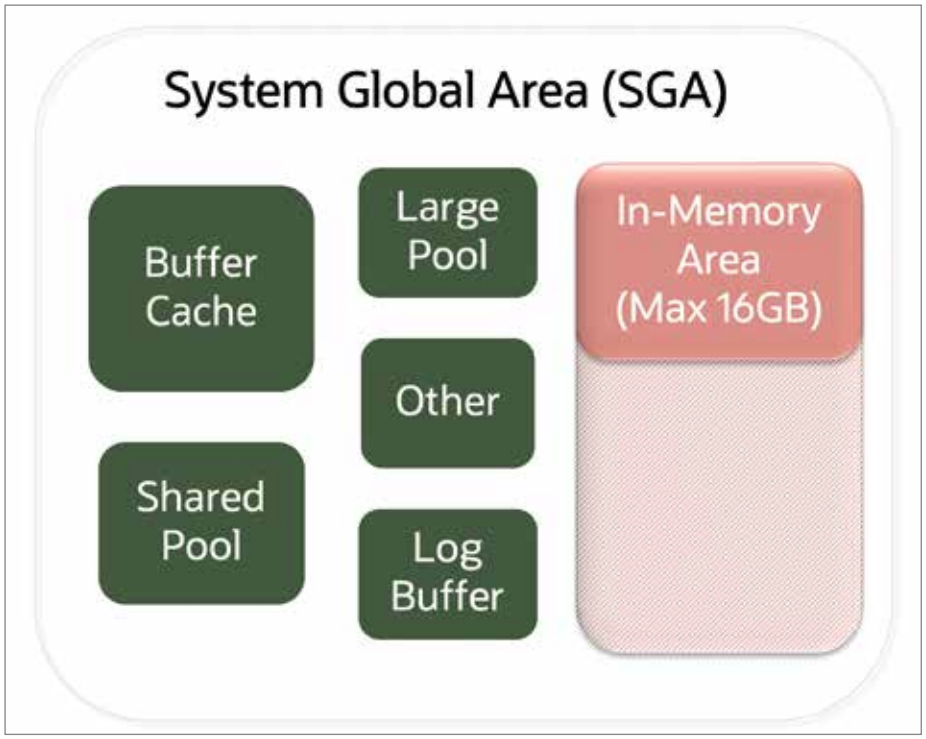


Abbildung 12: Base Level Feature (Quelle: Markus Kißling)

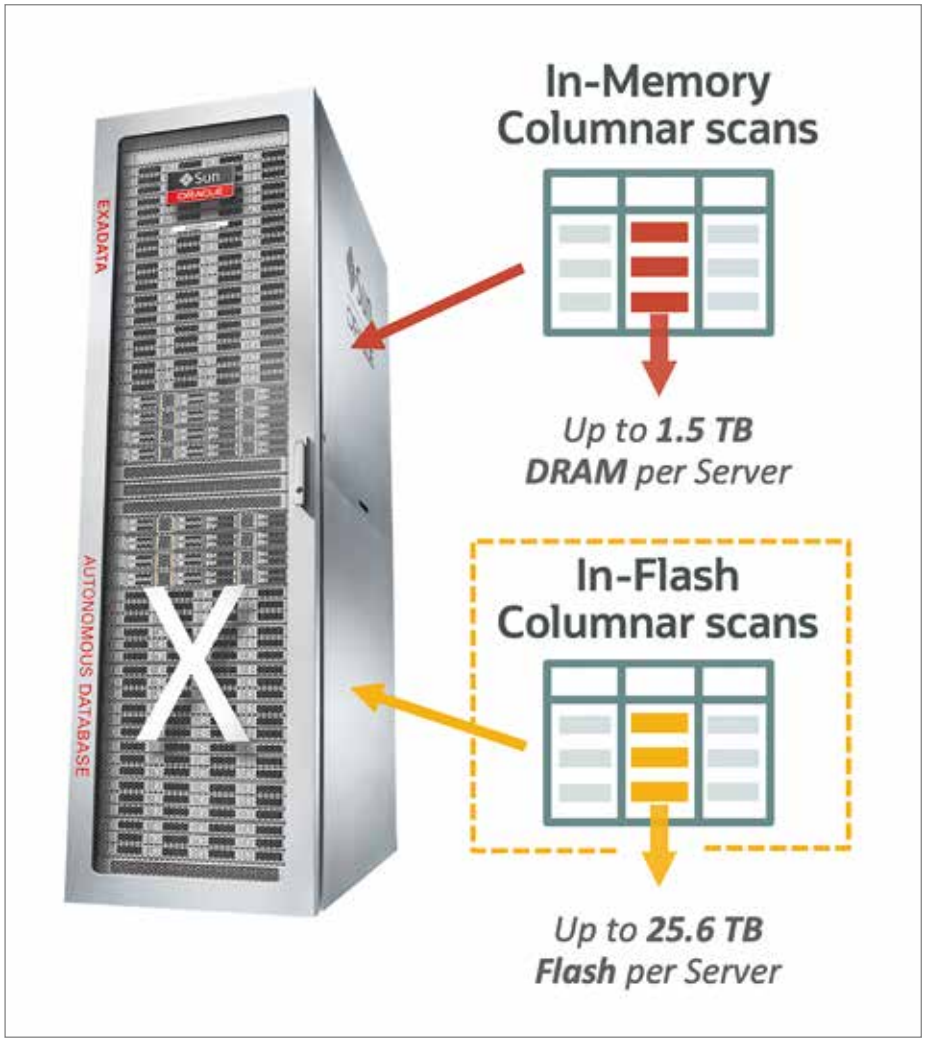


Abbildung 13: In-Memory Columnar Scans im DRAM und In-Flash Columnar Scans im CellMemory des Exadata Flash Storage (Quelle: Markus Kißling)

Im Rahmen des Converged-Database-Paradigmas finden die heterogenen Datentypen und Zugriffsarten in Oracle Database In-Memory Einzug. Externe und hybride Speicherungsmöglichkeiten sind im Zusammenhang mit In-Memory ergänzt worden.

Wie in jedem neuen Release gibt es zum Thema Performance neue Erweiterungen.

Exadata spielt ebenso eine zentrale Rolle bei der Implementierung der neuen Features.

Weiterführende Informationen zu Oracle Database In-Memory wie White Papers, Quick Start Guides für POCs und Links zu den Videos des Database In-Memory Youtube-Channels sind unter *[3]* zu finden.

Quellen

[1] *https://blogs.oracle.com/coretec/post/oracle-database-in-memory---der-schnelle-einstieg*
[2] *https://docs.oracle.com/en/database/oracle/oracle-database/21/inmem/index.html*
[3] *https://blogs.oracle.com/in-memory/post/dbim-resources*

Über den Autor

Markus Kißling ist als Product Manager bei Oracle für Oracle Database In-Memory zuständig. Zuvor hat er als Oracle Consultant und Oracle Sales Consultant Oracle-Lösungen rund um die Themen Oracle Database und Development & Migration Tools betreut.

Markus Kißling
markus.kissling@oracle.com

#bepartofIT

Freue dich auf spannende, vielseitige Aufgaben und unterstütze unser Dresdner Team in spannenden Kundenprojekten!

Werde Datenbank-Spezialist (m/w/d) bei Robotron!

Unsere Kunden setzen seit vielen Jahren auf die Expertise von Robotron im Bereich der Datenbankadministration und -entwicklung, vorrangig basierend auf Oracle-Technologien, Microsoft-SQL oder PostgreSQL. Werde auch du Teil dieser Erfolgsgeschichte! In unserem Projekt-Team entwirfst, implementierst und installierst du moderne Datenbankinfrastrukturen auf Basis neuester Technologien, On-Premises oder in Cloud-Umgebungen. Du bist professioneller Ansprechpartner für unsere Kunden zu Themen wie Datensicherung, Hochverfügbarkeit, Optimierung und IT-Security. Dabei wirst du von unserem motivierten Team tatkräftig unterstützt. Denn Teamgeist wird bei uns besonders groß geschrieben: zusammen diskutieren, konzipieren und implementieren wir nachhaltige Lösungen, auch beim Abteilungsgrillen.

Haben wir dein Interesse geweckt? Alle Informationen findest du unter www.robotron.de/datenbankspezialist

DAS BIETEN WIR:

- **individuelle Einarbeitung** für einen optimalen Einstieg
- **regelmäßige Weiterbildung** im eigenen Schulungszentrum
- **attraktives Gehalt**
- jährliche **Sonderzahlungen** wie Urlaubs- und Weihnachtsgeld
- **JobRad**
- flexible **Arbeits- und Teilzeitmodelle**
- **Flexibilität** für mobiles Arbeiten
- **Betreuungsangebot für Kinder** bis 3 Jahre in unserer Kindervilla
- gemeinsame **Sport- und Freizeitaktivitäten** im firmeneigenen Sportverein
- regelmäßige **Firmenevents**

robotron®

APEX 22.1 – Hidden Gems

Ronny Weiß, Oracle Global Services Germany

Hidden Gems ist sicherlich ein Klassiker, aber selbst absolute APEX-Profis ertappen sich dabei, dass gewisse Features entweder unbekannt oder in Vergessenheit geraten sind. Deshalb lohnt es sich immer wieder, dieses Thema in einem Vortrag oder Artikel vorzustellen. Vor einigen Wochen auf der APEX Connect 2022 wurde dazu von mir ein Vortrag gehalten. Geplant war, dass der Vortrag in einem der kleineren Räume stattfinden sollte. Da das Interesse jedoch so groß war, musste in den großen Saal gewechselt werden. Dies war Anlass genug, die Highlights des Vortrags noch einmal in einem Artikel zusammenzufassen. Der Aufbau des Artikels wird strukturell dem Vortrag ähneln, sodass es eine klare Trennung zwischen Application Builder, JavaScript und PL/SQL gibt und die einzelnen Features der Reihe nach behandelt werden

Application Builder – App Utilities – Embedded Code

Das erste interessante Feature findet man in den Utilities einer APEX-App. Der Embedded Code Viewer ermöglicht es, den Code einer App effizient zu reviewen *(siehe Abbildung 1)*. Man kann sehr schnell nach den Sprachen (PL/SQL, SQL und JavaScript) oder dem Bereich filtern sowie die einzelnen Codebestandteile herunterladen.

Application Builder – App Utilities – Font APEX

Ebenfalls im Utilities-Bereich eurer APEX-App findet ihr den Font APEX Icon Finder *(siehe Abbildung 2)*. Damit könnt ihr sehr schnell ein passendes Icon finden. Klickt man auf ein Icon, kann man dieses in einem modalen Dialog konfigurieren und den Quelltext dafür direkt kopieren.

Application Builder – App Utilities – APEX Logs

Ein weiteres Feature in APEX, das man über die App Utilities => APEX Views erreichen kann, sind die integrierten APEX Logs und Debug Messages *(siehe Abbildung 4)*. Diese können dabei helfen, die Nutzung von Apps, Rest-Sources und anderen Features zu analysieren und Fehler zu finden. Den Report kann man über Actions => Filter mit einem regulären Ausdruck *(siehe Abbildung 3)* filtern und erhält so eine Übersicht über alle verfügbaren Views.

Interessant ist vor allem die Debug Messages View. Sie zeigt nicht nur APEX-interne Debug- oder Fehler-Logs an, sondern man kann über das APEX_DEBUG PL/SQL-API auch eigene PL/SQL-Strukturen mit Logs versehen *(siehe Abbildung 5)*.

Application Builder – Deferred Page Rendering

Neu in APEX 22.1 ist das Deferred Page Rendering. Es aktiviert das verzögerte Laden eurer APEX-Seite, so wie man es von APEX 20.2 und älter kannte. Vor dem 21.1-Release wurde der Seiteninhalt versteckt, bis der gesamte JavaScript-Code der Seite geladen war. Mit 21.1 wurde der Seiteninhalt direkt angezeigt, was zum Teil zu einem unschönen Flackern einiger Seitenelemente führte, das Laden der Seite aber optisch schneller machte. Mit diesem Feature kann der Developer für jede Seite das gewünschte Ladeverhalten einstellen *(siehe Abbildung 6)*.

Application Builder – Component Settings

Einige native APEX-Komponenten und installierte Plug-ins besitzen die Möglichkeit, bestimmte Einstellungen für die gesamte App einheitlich zu setzen. Diese Component Settings findet man in den Shared Components einer App. Ein gutes Beispiel dafür ist der Date Picker. Diesen kann man entweder pro Item oder allgemein über die Component Settings konfigurieren *(siehe Abbildung 7)*.

Application Builder – Text Messages as Substitution

Viele kennen sicherlich das Text-Message-Feature in APEX. Dieses ermöglicht es, Messages dynamisch zum Beispiel in einem SQL-Statement zu übersetzen. Dabei wird in PL/SQL oder SQL das APEX_LANG.MESSAGE API genutzt. In JavaScript kann man die Funktionalität über apex.lang.getMessage aufrufen. Hier gilt es zu beachten, dass für die jeweilige Message „Used in JavaScript“ aktiviert sein muss. Man kann jedoch über apex.lang.loadMessages Messages nachladen, die nur sehr selten benötigt werden und bei denen „Used in JavaScript“ deaktiviert ist.

Es gibt aber auch die Möglichkeit, das Message-API über Substitutionen zu benutzen. Dabei kann man über &APP_TEXT$MY_MESSAGE., etwa beim Rendern einer Static HTML Region, die Textmessage in die Region integrieren *(siehe Abbildung 8)*. Mithilfe von Attributen wie !STRIPHTML steuert man, ob zum Beispiel HTML entfernt werden soll *(siehe Abbildung 9)*.

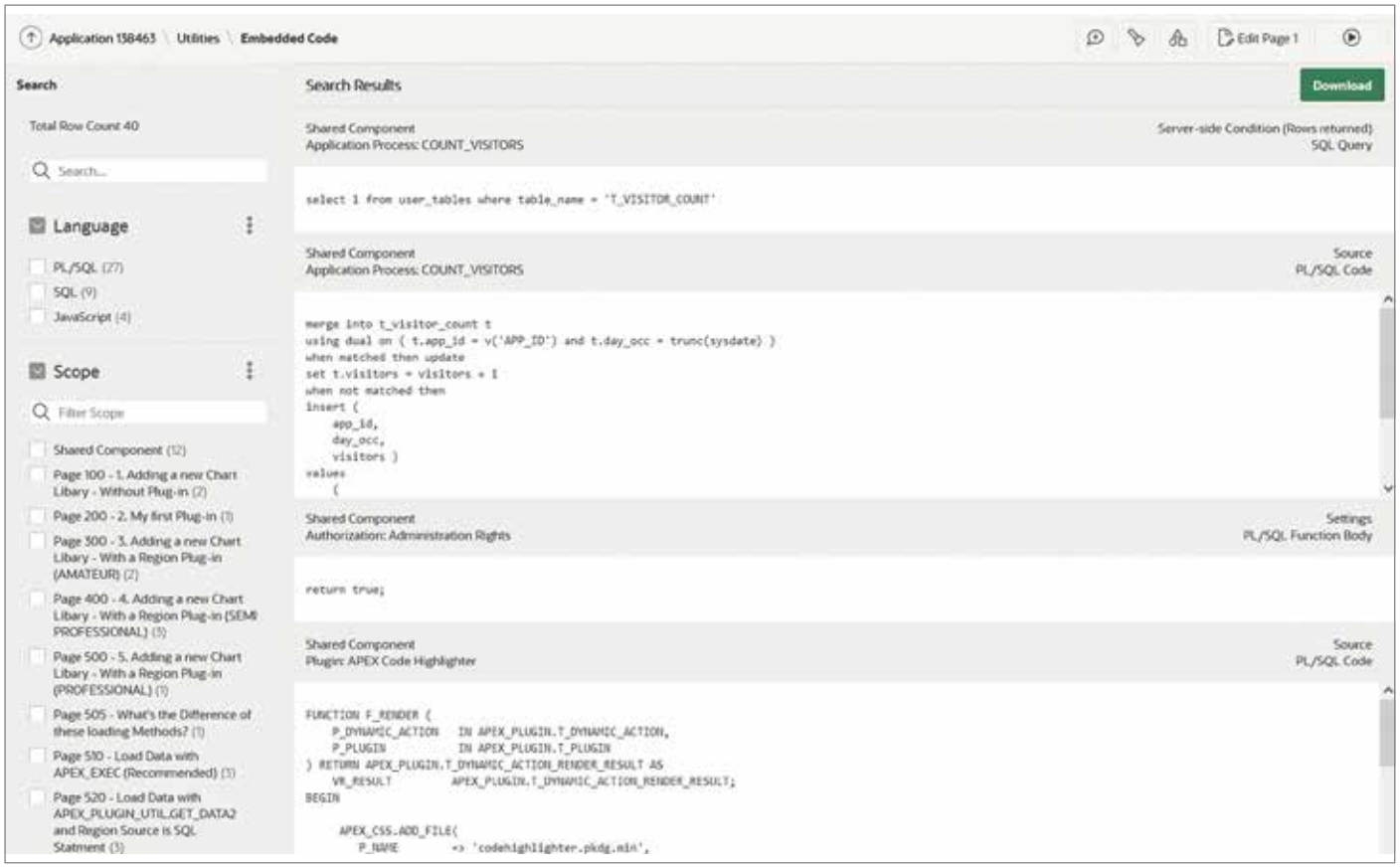


Abbildung 1: Embedded Code Viewer in APEX 22.1 (Quelle: Ronny Weiß)

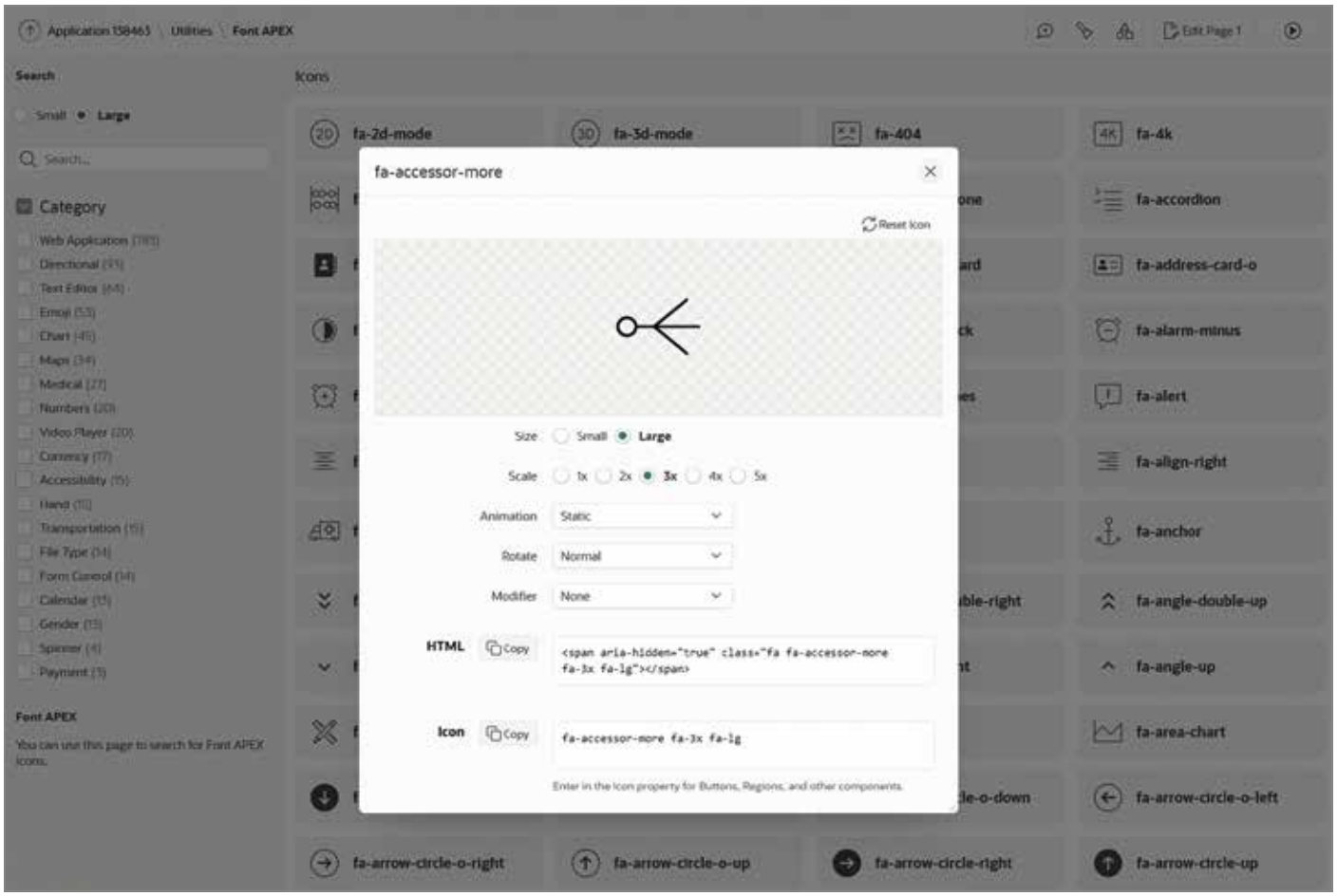


Abbildung 2: Font APEX in APEX 22.1 (Quelle: Ronny Weiß)

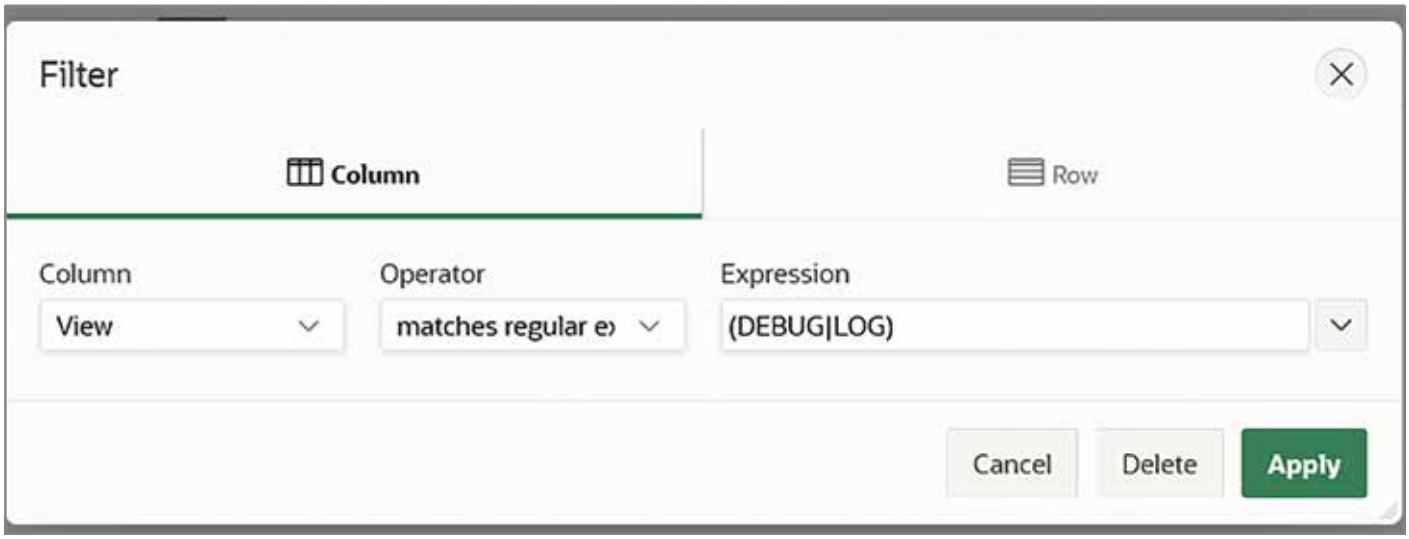


Abbildung 3: IR Column Filter in APEX 22.1 (Quelle: Ronny Weiß)

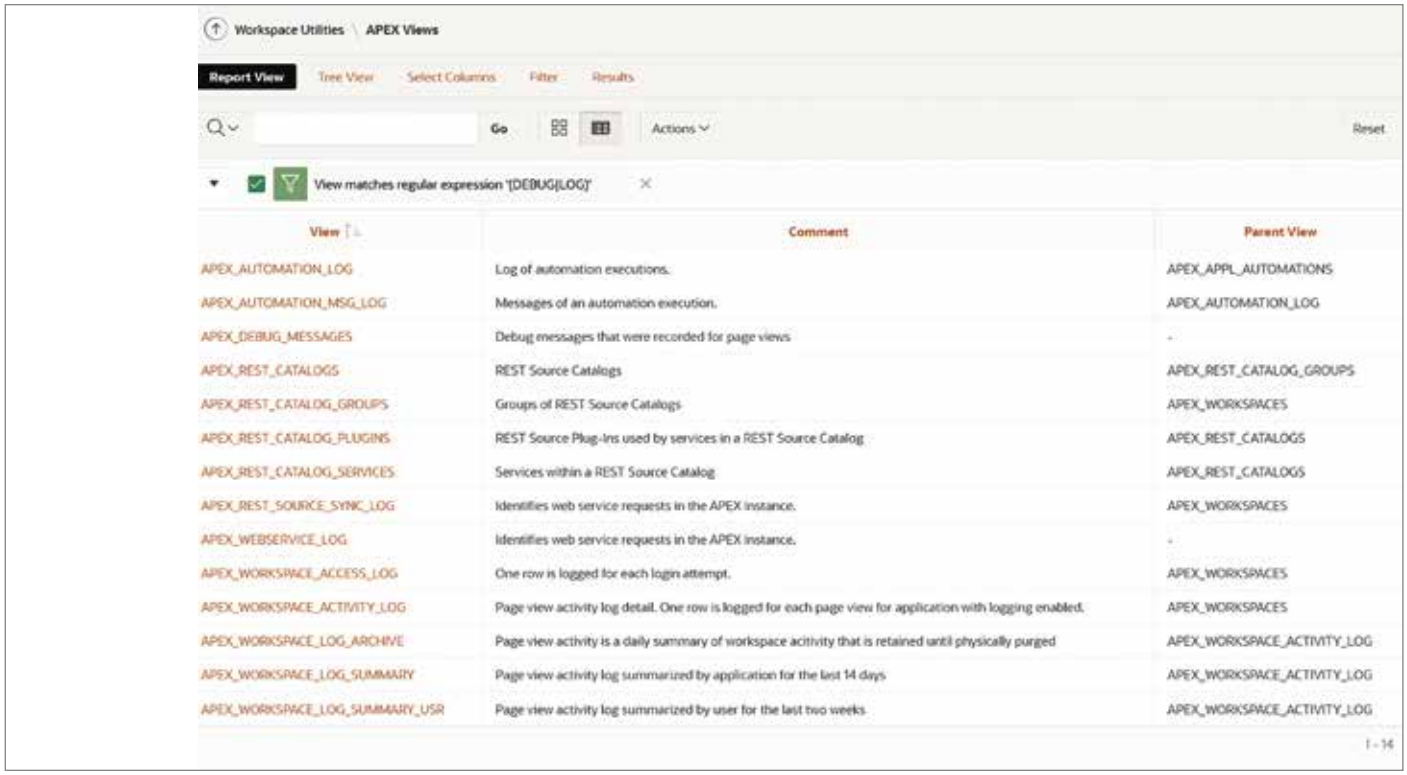


Abbildung 4: APEX Logs and Debug Messages View in APEX 22.1 (Quelle: Ronny Weiß)

```
1  create or replace NONEDITIONABLE function calc_my_results (
2      p_in_start number
3  ) return varchar2 as
4  begin
5      apex_debug.enter('calc_my_results', 'p_in_start', p_in_start);
6      if p_in_start <= 0 then
7          apex_debug.info('calc_my_results - p_in_start is zero or negative');
8          return p_in_start * 100 * 100;
9      else
10         apex_debug.info('calc_my_results - p_in_start is greater then zero');
11         return p_in_start;
12     end if;
13
14 exception
15     when others then
16         apex_debug.error('Critical error %s', sqlerrm);
17 end;
```

Abbildung 5: Sample Function with APEX_DEBUG (Quelle: Ronny Weiß)

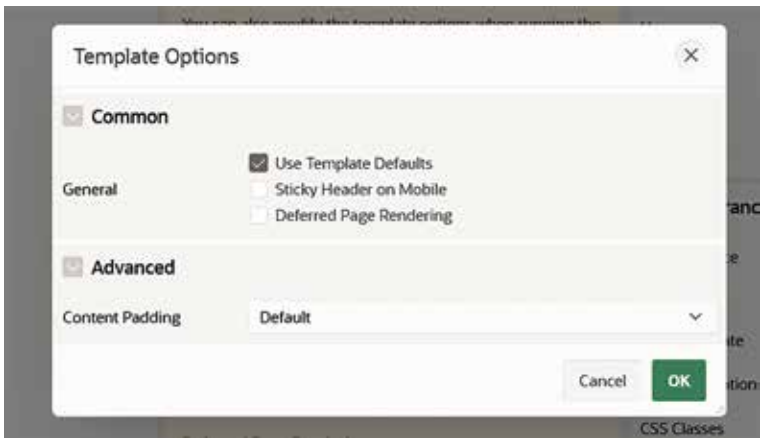


Abbildung 6: Page Template Options in APEX 22.1 (Quelle: Ronny Weiß)

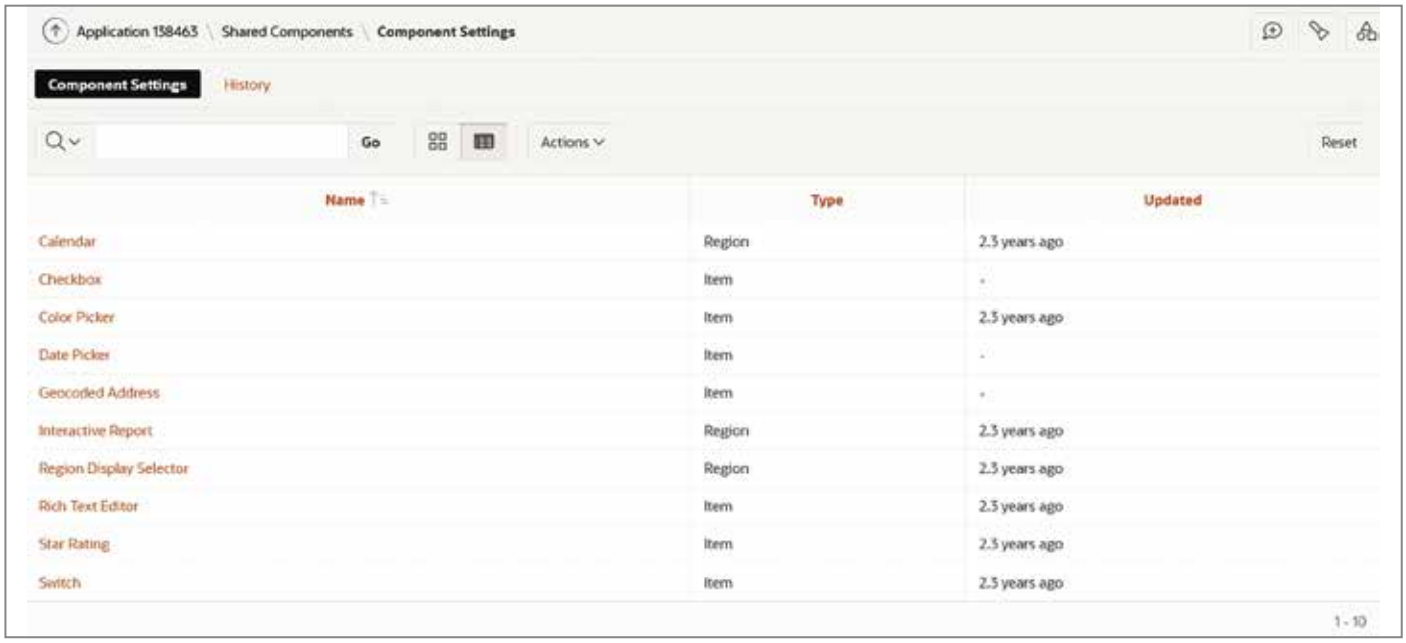


Abbildung 7: Component Settings in APEX 22.1 (Quelle: Ronny Weiß)

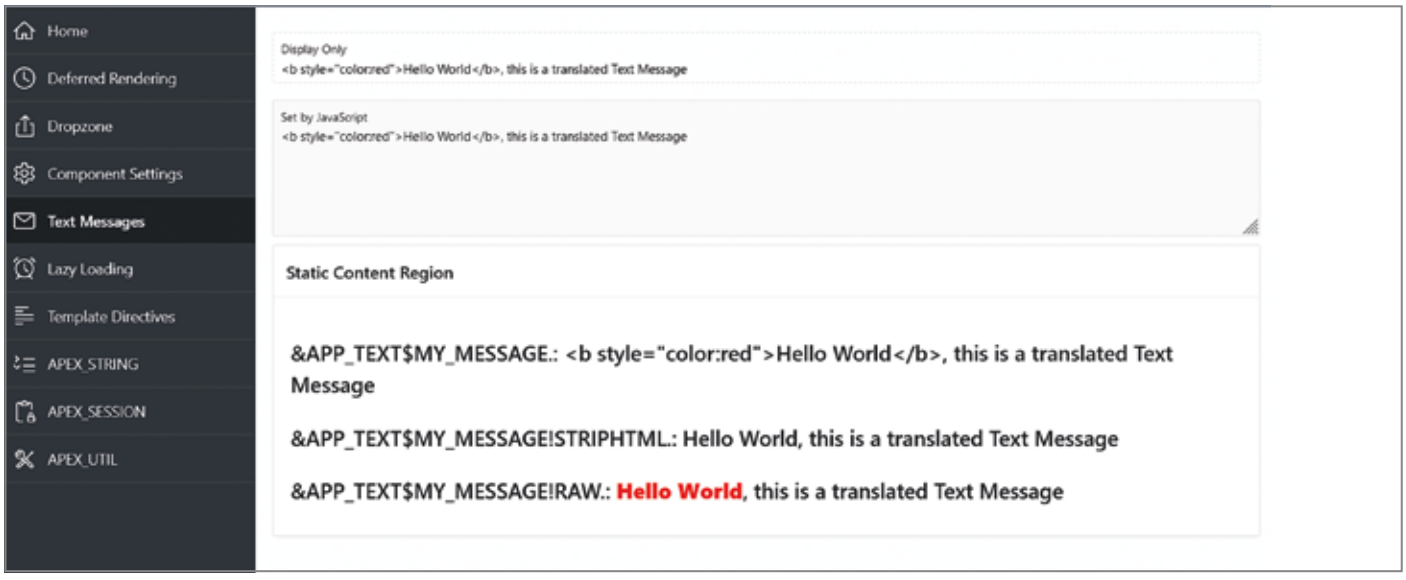


Abbildung 8: APEX Message Substitution in Static Region (Quelle: Ronny Weiß)

Abbildung 9: APEX-Message-Substitution-Quelltext in Static Region (Quelle: Ronny Weiß)

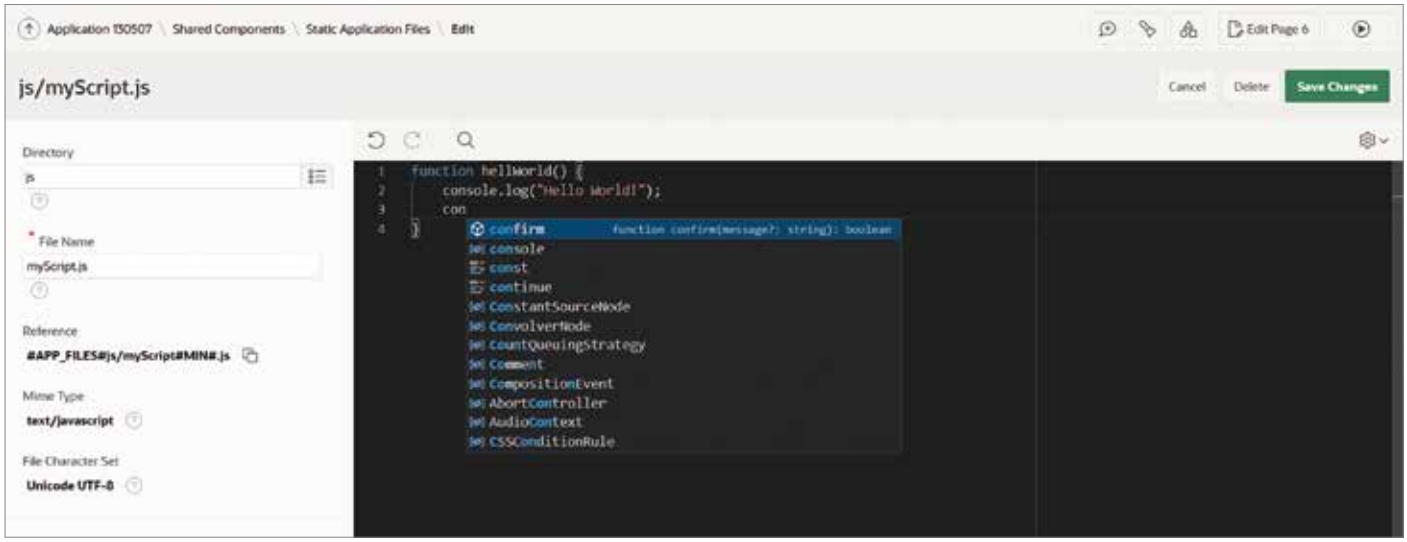


Abbildung 10: APEX Static File Editor mit Autovervollständigung (Quelle: Ronny Weiß)

Application Builder – Static Code Editor

In den letzten APEX-Releases wurde an immer mehr Stellen der Monaco Code Editor eingebaut. Bei den Static Application Files kann man damit sogar neue JavaScript- oder CSS-Dateien erstellen. Diese werden beim Speichern automatisch minimiert Der Code Editor unterstützt Multi-Cursor, ist Context Aware und bietet auch eine intelligente Autovervollständigung *(siehe Abbildung 10)*. Das Bearbeiten einer Script- oder CSS-Datei kann dadurch direkt in APEX erfolgen und steigert so weiter die Entwicklungsgeschwindigkeit innerhalb von APEX.

Application Builder – Spotlight Search

Möchte man schnell zu einem bestimmten Kontext etwas finden oder in seiner APEX-App suchen, ist Spotlight Search ein sehr praktisches Werkzeug *(siehe Abbildung 11)*.

Aufrufen kann man es über das Suchfeld in der rechten oberen Ecke der Seite oder per Shortcut über CTRL + Quote (CTRL + Ä bei einem deutschen Tastaturlayout). Praktisch ist vor allem, dass man per Link direkt zum gesuchten Inhalt springen kann. Dadurch kann man Spotlight Search verwenden, um effizient durch APEX zu navigieren.

Application Builder – Lazy Loading

Während der letzten Releases wurde die Unterstützung von Lazy Loading konse-

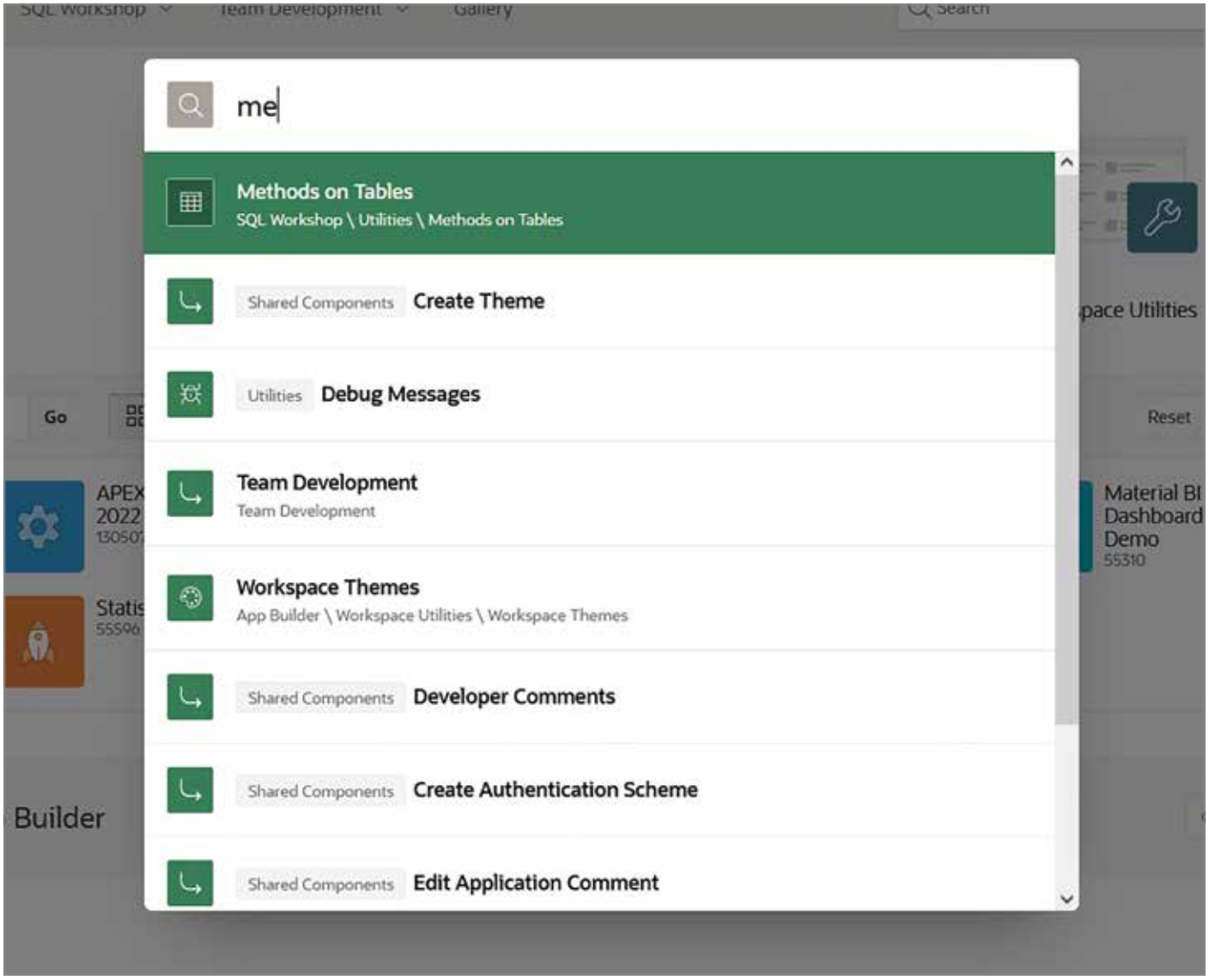


Abbildung 11: Spotlight Search in APEX 22.1 (Quelle: Ronny Weiß)

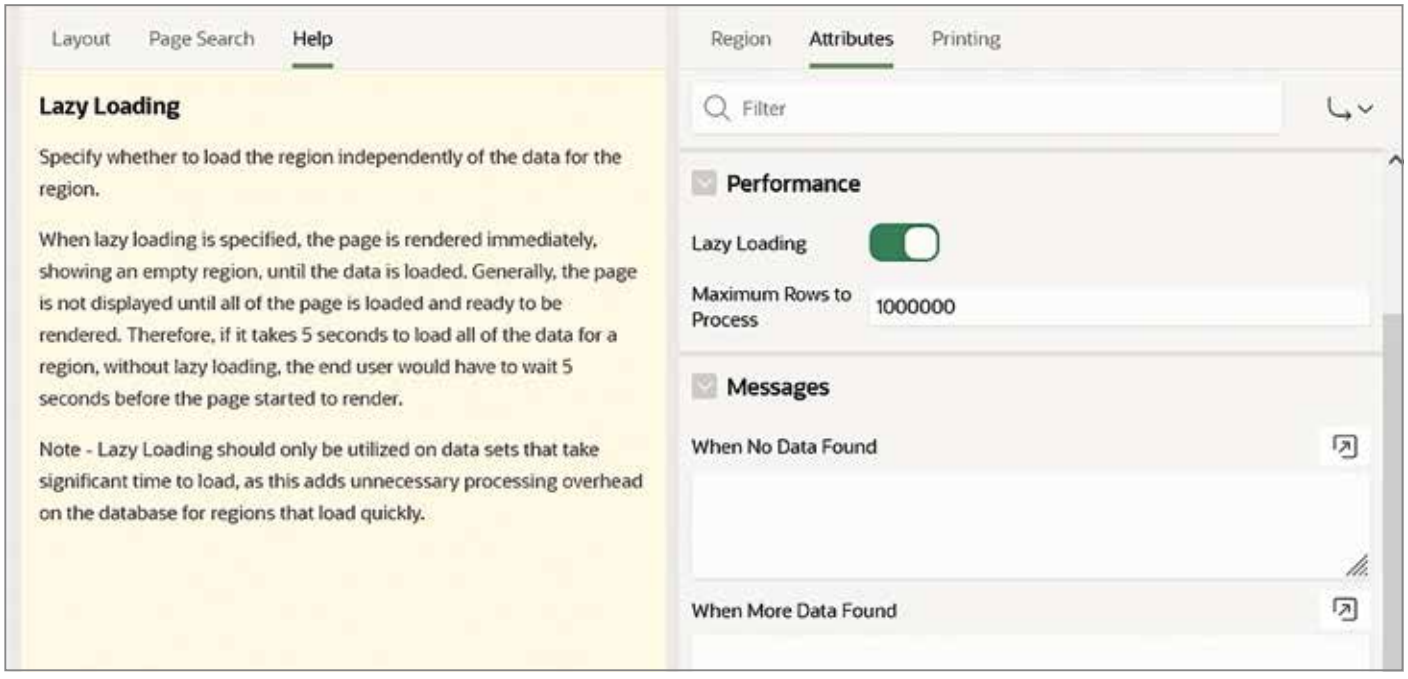


Abbildung 12: Lazy Loading in APEX 22.1 (Quelle: Ronny Weiß)

quent ausgebaut. In APEX 22.1 unterstützen folgende Komponenten dieses Feature:

- Interactive Grid
- Interactive Report
- Classic Report
- Tree Region
- Charts Region

Das Lazy Loading bietet für die Usability einer APEX-App sehr viele Vorteile. Das Hauptgerüst und schnellladende Bestandteile einer Seite werden dem Nutzer sehr schnell angezeigt und ein gegebenenfalls länger ladender Report wird dann asynchron nach dem Rendern geladen *(siehe Abbildung 12)*. Möchte der Endnutzer etwa nur einen Knopf für das Erstellen einer neuen Zeile drücken und sofort weiter navigieren, dann muss er nicht warten, bis der Report geladen ist.

JavaScript – apex.date

In APEX 21.2 wurde apex.date als neues JavaScript-API hinzugefügt *(siehe Abbildung 13)*. Neben dem apex.locale-API, das unter anderem Nummern mit Datenbankformaten parsen (apex.locacle.toNumber) und formatieren (apex.locale.formatNumber) kann, bietet apex.date die Möglichkeit, Operationen rund um das Thema Datumsobjekte durchzuführen.

Es bietet neben dem Parsen (apex.date.parse) und Formatieren (apex.date.format) von Daten auch viele andere praktische Funktionen, wie das Vergleichen von zwei Daten (apex.date.isSame) oder das Finden des ersten Tages eines Monats (apex.date.firstOfMonth).

JavaScript – Template Directives

Die Template Directives können ebenfalls ein sehr nützliches Feature sein *(siehe Abbildung 14)*. Diese Funktionalität dient dazu, in ein Template, beispielsweise in HTML, Daten einzufügen.

Folgende Optionen sind damit möglich:

- Daten-Substitution
- Direktiven
  - IF-Conditions
  - CASE-Conditions

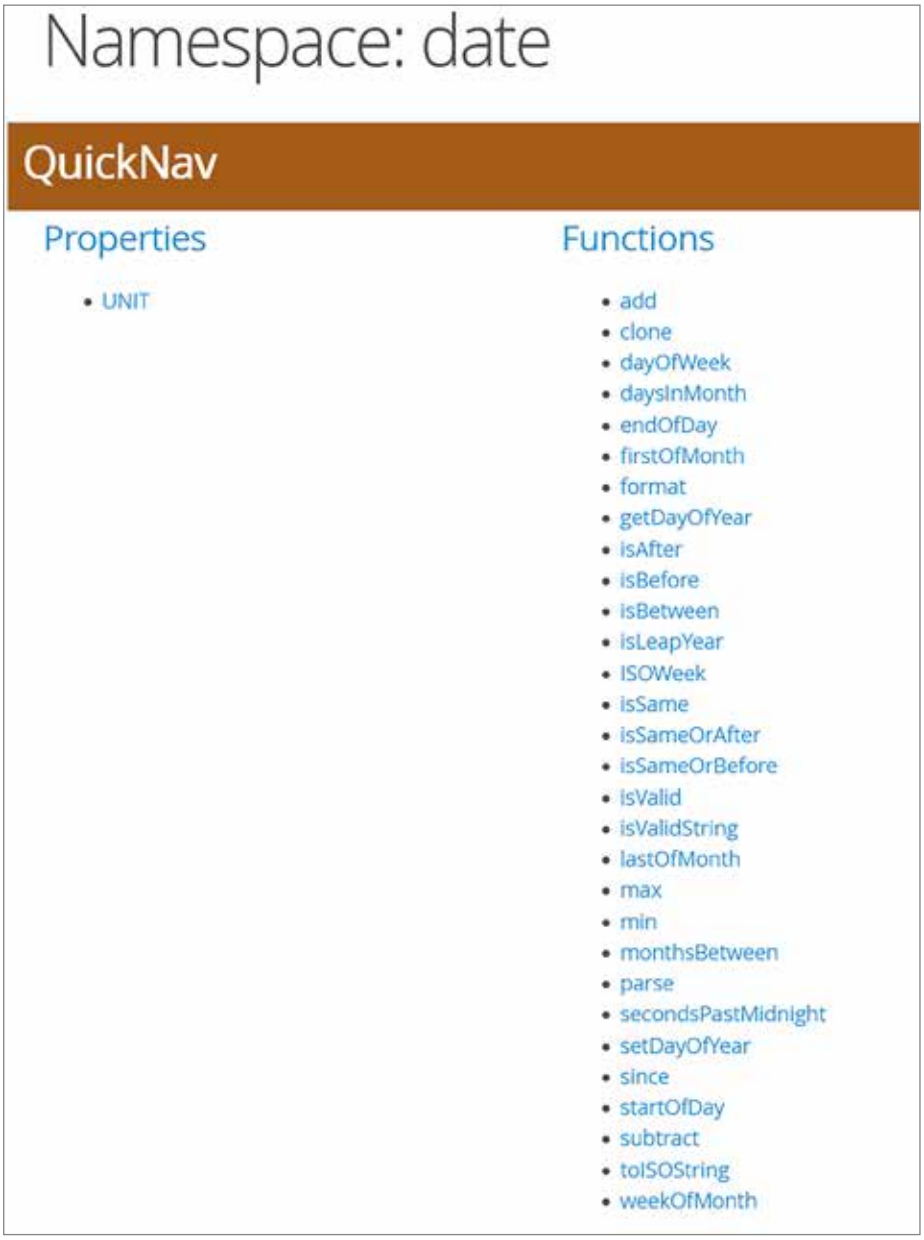


Abbildung 13: apex.date-API-Übersicht in APEX 22.1 (Quelle: Ronny Weiß)

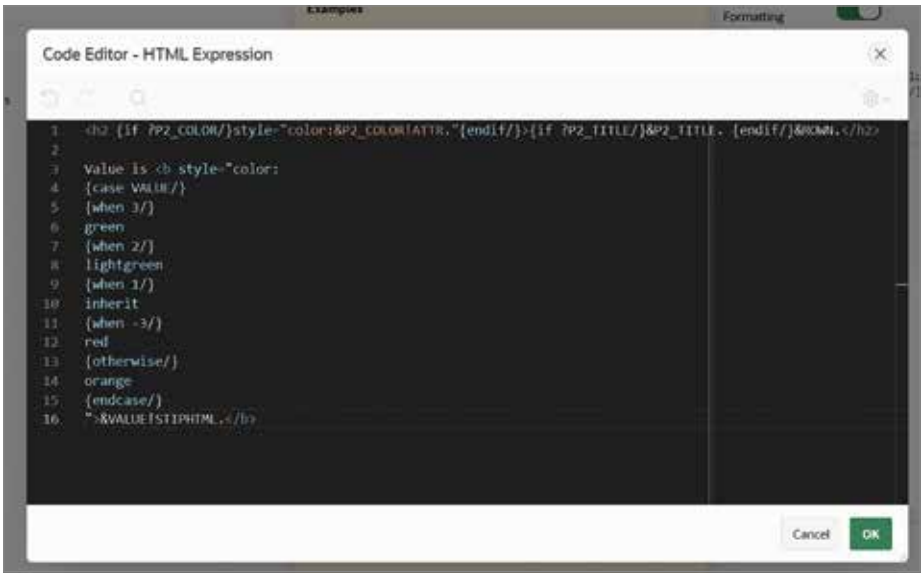


Abbildung 14: Template Directives in APEX 22.1 (Quelle: Ronny Weiß)

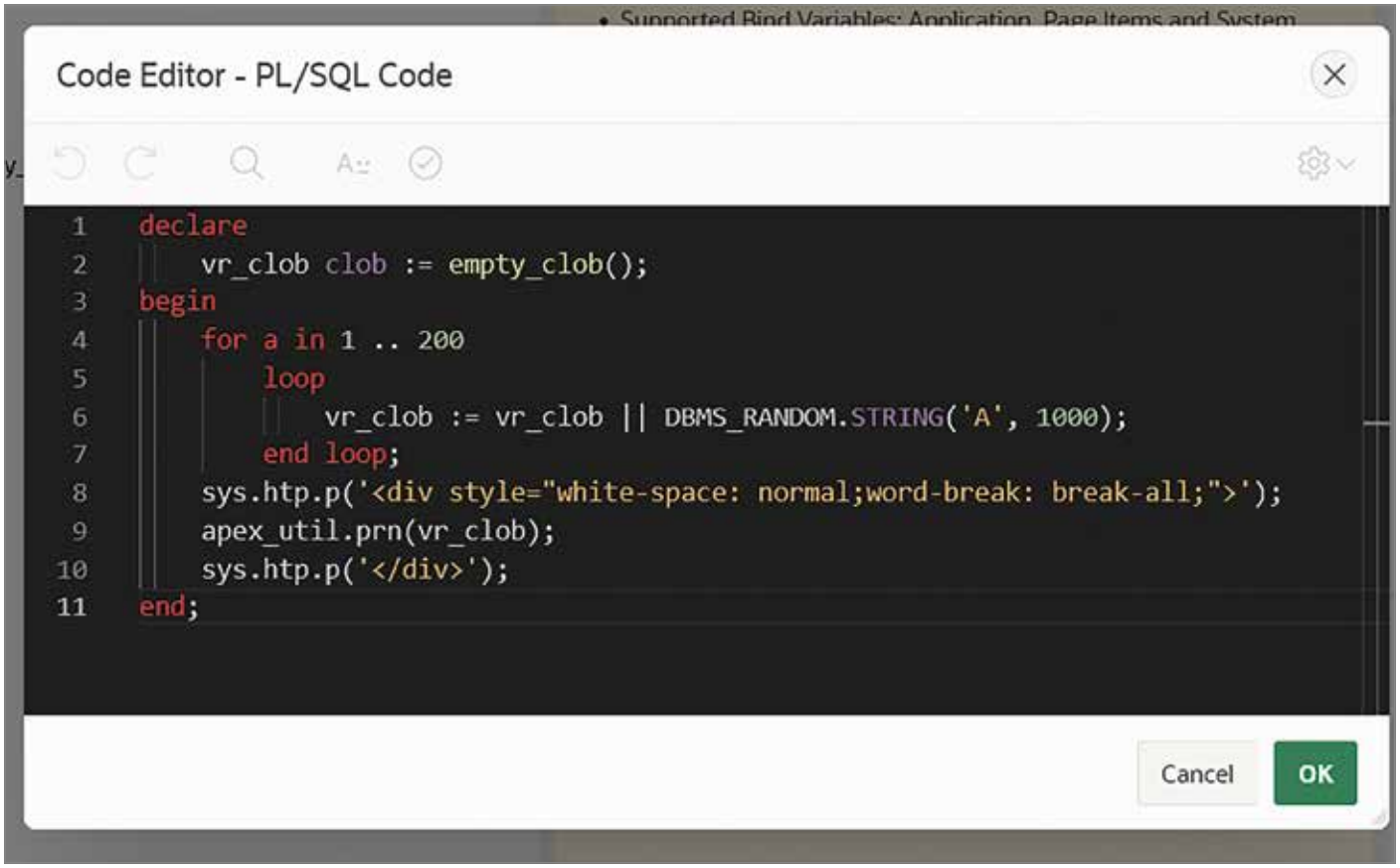


Abbildung 15: apex_util.prn (Quelle: Ronny Weiß)

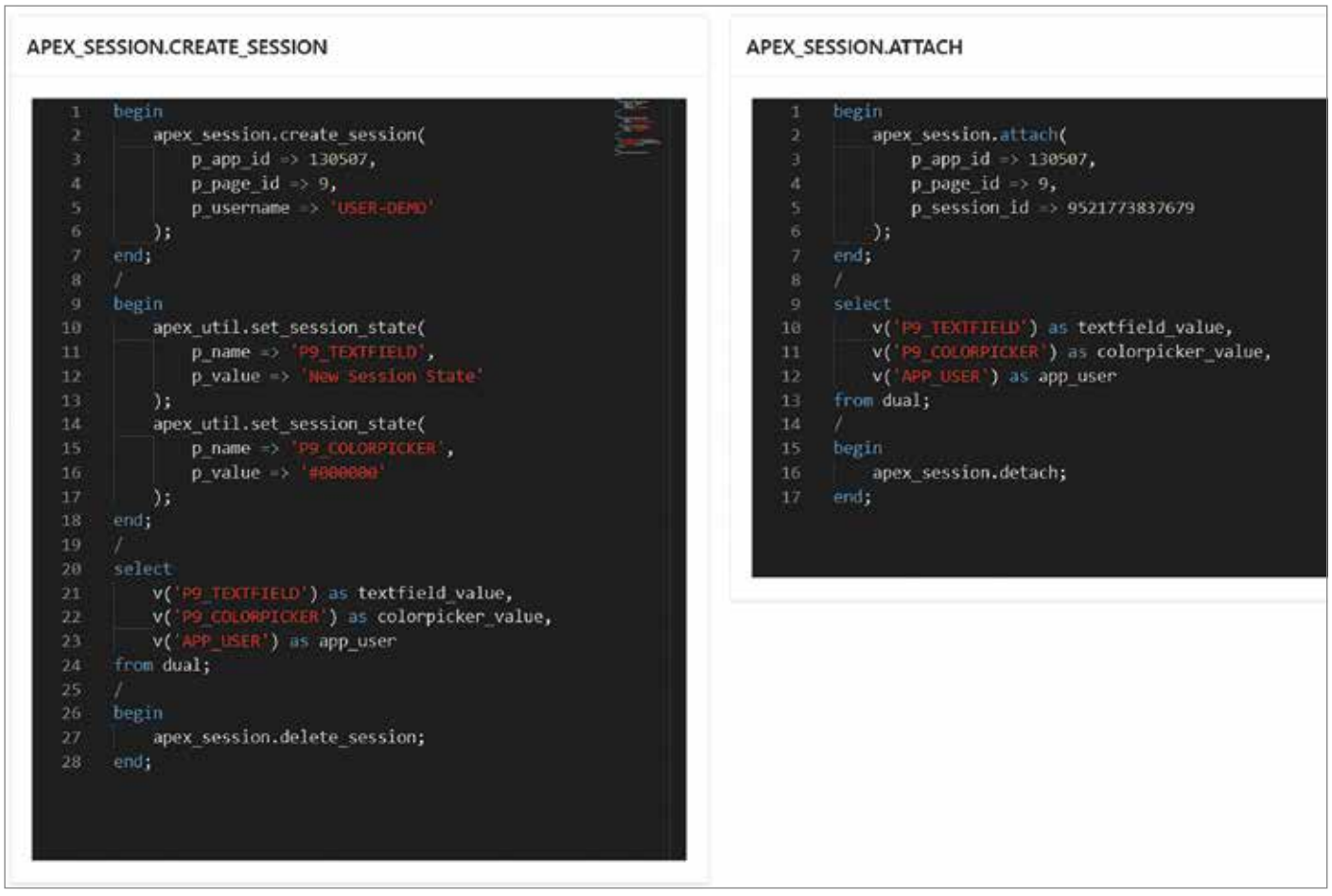


Abbildung 16: apex_session.create_session/attach-Beispiel (Quelle: Ronny Weiß)

  - LOOPs
- Kommentare

In APEX 22.1 werden Template Directives in folgenden Features unterstützt:

- Cards Region (Client)
- Interactive Grid (Client)
- Map Region (Client)
  - Info Window
  - Tooltip
- E-Mail-Templates (Server)
- JavaScript-/Plug-in-Entwicklung – apex.util.applyTemplates

PL/SQL – apex_util.prn

Möchte man HTML auf einer Seite dynamisch ausgeben, konnte man bisher nur sys.htp benutzen und war dadurch auf die Maximallänge von VARCHAR2 beschränkt. Wollte man längere Texte (CLOBs) auf die Seite ausgeben, so musste man sich bisher immer eine Hilfsfunktion bauen. Diese Arbeit kann man sich nun sparen und einfach apex_util.prn verwenden *(siehe Abbildung 15)*.

PL/SQL – apex_session.create/attach

In den APEX PL/SQL-APIs findet man auch das apex_session-Package. Es beinhaltet zwei sehr praktische Funktionen. Braucht man außerhalb von APEX zum Entwickeln oder Debuggen eine APEX-Session, so kann man sich diese sehr einfach über apex_session.create erstellen. Mit apex_session.attach kann man sich aber auch auf eine bestehende Session hängen *(siehe Abbildung 16)*. Es gilt jedoch zu beachten, dass bei einem attach nur der aktuelle Zustand genutzt werden kann. Ändert sich der Session State, muss man sich erneut auf die Session hängen.

PL/SQL – apex_string/string_util

Ein weiteres sehr interessantes PL/SQL-API ist apex_string bzw. apex_string_util. Dieses API kennt man sicherlich durch apex_string.split, das häufig dazu genutzt wird, in SQL Sources in Regionen zu filtern, bei denen man Multi-Value-Items benutzt.

Code - GREP

```
SELECT
    COLUMN_VALUE
FROM
    TABLE (
        APEX_STRING.GREP (
                        P_TABLE => APEX_STRING.SPLIT(
                        P_PATTERN => '^[0-9]*',
                        P_MODIFIER => 'i',
                        P_SUBEXPRESSION => null )
        )
```

Abbildung 17: apex_string.grep-Beispiel (Quelle: Ronny Weiß)

Dieses API bietet jedoch eine Reihe weiterer interessanter Nutzungsmöglichkeiten:

- apex_string_util.find_email_addresses: Findet E-Mailadressen innerhalb eines Strings
- apex_string.find_links: Findet Links innerhalb eines Textes
- apex_string.find_tags: Findet #-Tags innerhalb eines Textes
- apex_string.grep: Findet mithilfe von regulären Ausdrücken bestimmte Strings innerhalb eines Textes *(siehe Abbildung 17)*
- apex_string_util.to_display_filesize: Zeigt Größen von Dateien in einem leserlichen Format an

Fazit

APEX bietet so viele großartige Features, um effizient einfache, aber auch sehr professionelle APEX-Apps zu bauen! Selbst ich als Senior-APEX-Entwickler bin stets auf der Suche nach neuen Tools und Tipps, die die Arbeit mit APEX noch weiter optimieren und Zeit zu sparen helfen. Leider ist es unmöglich, alle in diesem Artikel zu nennen, aber es wird sicherlich wieder einen Vortrag oder Artikel geben, in dem man die neusten Hidden Gems in APEX vorstellen kann.

Ein Tipp noch für APEX-Beginner. Wer APEX noch nicht ausprobiert hat, richtet sich am besten einen kostenlosen Zugang auf *https://apex.oracle.com* oder *https://www.oracle.com/cloud/free* ein. Danach kann es mit der ersten APEX-App losgehen und ihr könnt die genannten Features direkt ausprobieren.

Über den Autor

Ich arbeite als Softwareentwickler seit 2013 mit Oracle APEX und entwickle seit 2017 freie Open-Source-Plug-ins für die Oracle APEX Community. Seit September 2021 arbeite ich im Oracle APEX Development Team.

Ronny Weiß
ronny.weiss@oracle.com

BUSINESS

Das ERP Phänomen

NetSuite

NEWS
05/2022

Das NetSuite-Phänomen: Warum NetSuite zum „Lieblings-ERP“ der Wachstumsunternehmen geworden ist

Dr. Frank Schönthaler, PROMATIS Gruppe, Ettlingen (TechnologieRegion Karlsruhe)

Mit dem vollständigsten Portfolio von Applications Cloud Services hat sich Oracle weltweit in eine marktführende Position katapultiert. Selbst im deutschsprachigen Raum konnte Oracle mit namhaften Neukunden für Furore sorgen. Ein interessantes Phänomen zeigt sich im Segment der Wachstumsunternehmen. Dort gewinnt Oracle mit NetSuite ERP die meisten Neukunden und erzielt eine besonders hohe Kundenzufriedenheit. Warum ist das so? Was zeichnet NetSuite ERP gerade in diesem Marktsegment aus? Und was müssen NetSuite-Kunden beachten, um mit ihrem neuen ERP-System erfolgreich zu sein? Auf all diese Fragen gibt der vorliegende Beitrag Antworten.

Einführung

Was die Appelle von Politikern und Wirtschaftsführern über viele Jahre nicht vermochten, haben die sich gegenseitig verstärkenden Krisen unserer Zeit spielend geschafft: Aus dem Willen zur Digitalisierung ist ein Megatrend geworden, der digitalen Technologien Zugang zu allen erdenklichen Bereichen der Gesellschaft und Wirtschaft verschafft. In den Unternehmen zeigt die fortschreitende Digitalisierung immer auch Auswirkungen auf die Geschäftsprozesse: Während die Prozesse der ersten Generation an den Unternehmensgrenzen Halt machten, mussten die Prozesse der zweiten Generation auch Business-to-Business-Schnittstellen und Self-Service-Funktionalitäten für Kunden, strategische Partner und Mitarbeiter anbieten. Nunmehr stehen die Unternehmen aller Branchen vor der Herausforderung, in ihren Prozessen komplette „Journeys“ für Kunden, Partner und Mitarbeiter zu offerieren – und das in bester Benutzerfreundlichkeit zu jeder Zeit an jedem Ort der Welt mit garantierter Sicherheit und Performance. Mögen die Unternehmen in ihren Prozessen noch genügend Elastizität eingebaut haben, um diesen Herausforderungen gerecht werden zu können, stoßen sie doch mit ihren herkömmlichen On-Premises-Unternehmenssoftware-Systemen an ihre Grenzen. Schnelles und konsequentes Handeln ist gefordert, um diese Grenzen zu überwinden und in der digitalen Ökonomie auf der Gewinnerseite zu stehen. Viele Unternehmen begegnen dieser Herausforderung mit der Migration ihrer erfolgskritischen Geschäftsprozesse in die Cloud, was in der Folge zu einer stark steigenden Nachfrage nach Cloud-basierter Unternehmenssoftware führt.

Im Markt für Unternehmenssoftware ist aber nicht nur eine steigende Nachfrage zu verzeichnen, sondern es zeigen sich auch signifikante Verschiebungen im Ranking der Wettbewerber. Folgt man den Analysten von Gartner (vgl. *[1]*) hat sich Oracle mit seiner Financials-Cloud-Lösung für mittlere, große und globale Unternehmen deutlich vor Wettbewerbern wie Workday, SAP, Microsoft und Infor im Markt platziert. In der Gartner-Cloud-ERP-Studie für produktzentrierte Unternehmen (vgl. *[2]*) ist der Abstand zu Microsoft und Infor noch gravierender, während es SAP nicht einmal in den Leader-Quadranten geschafft hat. So ist es nicht verwunderlich, dass sich Oracle auch im deutschsprachigen Raum aus der Nische herausgearbeitet hat und mittlerweile durch erfolgreiche Oracle-Cloud-Applications-Implementierungen in namhaften Unternehmen aufhorchen lässt.

Die Gartner-Studien zeigen aber auch, dass Oracle tatsächlich mit zwei Siegpferden im Wettbewerb ist: Ebenfalls im Leader-Quadranten (vgl. *[1]*) positioniert steht Oracle NetSuite mit mehr als 22.000 Kunden bezüglich der Umsetzungsfähigkeit knapp hinter Workday, aber vor S/4HANA Cloud (SAP) auf Position 3. Mit 22.000 Kunden ist NetSuite das #1-ERP-System weltweit. Und gerade im deutschsprachigen Markt verzeichnet NetSuite hohe Wachstumsraten. Zudem lässt sich das Phänomen beobachten, dass sich NetSuite vor allem bei Wachstumsunternehmen großer Beliebtheit erfreut. Warum ist das gerade in diesen Unternehmen der Fall? Das ist die zentrale Frage, die im Mittelpunkt dieses Beitrags steht.

Zunächst werden im Vergleich zur Oracle Applications Cloud Software die architektonischen Besonderheiten von NetSuite herausgearbeitet. Das agile Implementierungsvorgehen bei NetSuite steht im Mittelpunkt des Folgekapitels. Darauf aufbauend wird aufgezeigt, welcher Bedarf an Mitwirkungsleistungen sich aus diesem Vorgehen gerade für Wachstumsunternehmen ergibt. Eine kurze Zusammenfassung und ein Blick in die Zukunft schließen den Beitrag.

Skalierbare Lösungsarchitektur

In Wachstumsunternehmen ist Skalierbarkeit eines der Schlüsselworte: Skalierbarkeit des Geschäftsmodells, der Kunden- und Lieferantenbasis, des Humankapitals, der technischen Kapazitäten und Ressourcen und natürlich auch der Geschäftsprozesse und Unternehmenssoftware-Systeme. Skalierbarkeit bedeutet aber nicht nur die Fähig-

Abbildung 1: Oracle hat zwei Siegpferde im Cloud-Applications-Wettbewerb (Bildquelle: Sebastian Graf, Karlsbad [© 2022])

keit, schnell zu wachsen, sondern auch bei Bedarf noch schneller wieder zu schrumpfen. Die Forderungen an die Skalierbarkeit sind klar: schnell, stetig und kostengünstig und das bei zumindest gleichbleibend guter Produkt- und Servicequalität. Skalierbarkeit ist in Wachstumsunternehmen einer der wichtigsten Erfolgsfaktoren und zweifellos einer der entscheidenden Indikatoren in der Beurteilung des Unternehmenswerts.

Die Überlegungen zur Skalierbarkeit machen deutlich, dass in puncto Unternehmenssoftware für Wachstumsunternehmen kein Weg an Cloud-basierten Systemen vorbeiführt. Selbst moderne integrierte On-Premises-Systeme sind viel zu monolithisch und zu starr, um mit der Agilität Schritt zu halten, die in der DNA des Wachstumsunternehmens kodiert ist. Im Gegensatz dazu bietet die Nutzung von Software Services aus der Cloud (Software as a Service, SaaS) sehr viel mehr Flexibilität für die Evolution des Wachstumsunternehmens. Dabei muss das Wachstumsunternehmen achtgeben, dass es nicht in eine technologische Sackgasse gerät, in der die Evolution nur durch einen großen „Gemischtwarenladen" isolierter oder nur schwach integrierter Services unterschiedlichster Hersteller erkauft werden kann.

Vor diesem Hintergrund empfiehlt es sich, den Blick auf Enterprise-SaaS-Lösungen zu richten, die bereits im Grundausbau eine breite Funktionalität mitbringen. In Verbindung mit leistungsfähigen Technologien zur Integration von Services – kundenspezifischen oder von Fremdherstellern – bieten sie zusätzliche Funktionstiefe und Branchenspezifika. *Abbildung 2* zeigt am Beispiel der Oracle Cloud Applications die Architektur einer Enterprise-SaaS-Lösung.

Mit den Oracle-Fusion-Modulen im Kern der Oracle Cloud Applications steht eine auf Funktions-, Prozess- und Datenebene voll integrierte Unternehmenssoftware-Lösung zur Verfügung. Sie kann bedarfsgerecht lizenziert und implementiert werden. Über ein reichhaltiges Angebot von standardbasierten File- und Webservice-Schnittstellen kann die Lösung flexibel mit anderen Systemen kommunizieren oder auch durch Zusatzmodule iterativ erweitert werden. Allerdings wird eine derart ausgebaute Lösung rasch unübersichtlich und komplex, was sich spätestens in der Betriebsphase negativ auf Qualität und Kosten auswirkt. Als Abhilfe empfiehlt sich der Einsatz einer standardbasierten Integrationsplattform, die idealerweise auch als Cloud Service „deployed" wird. Sie vermindert die Komplexität des Gesamtsystems, erlaubt das Management und Monitoring der Integrationsprozesse und Schnittstellen und vereinfacht die evolutionäre Weiterentwicklung des Systems. Im Fall von Oracle Fusion wird zumeist der Oracle Integration Cloud Service (OICS) eingesetzt, der eine Vielzahl vorgefertigter Konnektoren und Integrationen mitbringt. Zum Funktionsumfang des OICS gehören eine Low-Code-Entwicklungsumgebung (Visual Builder) und flexibel einsetzbare Prozessautomatisierungs- und Case Management Engines auf Basis von BPEL (SOA Cloud) und BPMN (Process Cloud). Vorgefertigte Integrationskomponenten sind auch der Schlüssel in die Oracle-Welt mit zahlreichen funktionalen Erweiterungsmodulen.

Zusammen mit dem kompletten Portfolio der Oracle-Plattform- und Infrastrukturservices sowie den standardbasierten Technologien zur Integration mit allen am Markt verfügbaren Software Services bietet die vorgestellte Architektur praktisch keine Grenzen beim Aufbau einer zukunftssicheren Unternehmenssoftware-Lösung. Doch nicht selten fühlen sich Wachstumsunternehmen – gerade in den frühen Jahren ihrer Entwicklung – von der funktionalen Breite, aber auch Tiefe einer solchen Enterprise-

Lösung geradezu erschlagen: Brauchen wir das wirklich alles? Sind denn unsere Stammdatenstrukturen wirklich so reichhaltig und komplex? Wer soll denn die Vielzahl dieser Transaktionsdaten erfassen? Und obgleich wir die funktionale Breite brauchen, brauchen wir doch (noch) nicht die volle funktionale Tiefe? Ist nicht generell die Komplexität viel zu hoch? Und dauert die Einführung nicht viel zu lange? Wann sehen wir denn überhaupt die ersten Ergebnisse? So lauten oft die Fragen. Geht das denn nicht auch kleiner, einfacher, leichtgewichtiger, kostengünstiger – ohne dass wir uns den Weg in die Zukunft verbauen? Und gibt es denn nicht Wege, unsere eigenen IT-Experten im Implementierungsprozess mit einzusetzen, um so Kosten und Zeit zu sparen und im nachfolgenden Betrieb und in der Weiterentwicklung unabhängiger von einem externen Implementierungspartner zu sein?

Auf alle diese Fragen bietet Oracle NetSuite Antworten. Und den Schlüssel hierzu liefert die Lösungsarchitektur, die in *Abbildung 3* dargestellt ist.

Analog zur Enterprise-Lösung aus *Abbildung 2* weist NetSuite einen funktionalen Kern auf, der dann mit Zusatzmodulen individuell erweitert werden kann *(siehe Abbildung 3)*. Der Kern kann entweder in Form des von NetSuite vorkonfektionierten SuiteSuccess-Moduls oder als NetSuite-Mid-Market-Modul lizenziert werden. Im Unterschied zu Oracle Fusion aus *Abbildung 2* sind die Komponenten des Kernmoduls bei NetSuite im Lizenzumfang enthalten, während sie bei Fusion einzeln und oft auf Basis unterschiedlicher Metriken lizenziert werden müssen. Gerade für Unternehmen, die die gesamte funktionale Breite benötigen – dies ist bei Wachstumsunternehmen regelmäßig der Fall – ergeben sich so oft signifikante Kostenvorteile für NetSuite. Im Umkehrschluss lässt sich erwarten, dass für punktuell eingesetzte Cloud-Lösungen diese Kostenvorteile hinfällig sein können. Beispielsweise könnte eine voll ausgebaute Oracle-Fusion-Procurement-Cloud-Lösung gegenüber einer NetSuite-Lösung gleicher Funktionalität durchaus Kostenvorteile aufweisen.

Beim Vergleich zwischen SuiteSuccess und dem Mid-Market-Modul fällt die Wahl sehr oft auf SuiteSuccess, selbst wenn nicht das SuiteSuccess-Einführungsverfahren gewählt wird, sondern ein alternatives. Der Grund liegt schlichtweg am Mehr an mitgelieferter Prozess- und Auswertungsfunktionalität und an der Dokumentation in SuiteSuccess.

In der Praxis ist die Funktionalität des Kernmoduls für viele Unternehmen – insbesondere für Start-ups und kleinere Wachstumsunternehmen – bereits ausreichend. In solchen Fällen kann eine ERP-Einführung dann auch mal nur ein paar Tage oder wenige Wochen dauern. Die NetSuite-Idee ist jedoch, dass dieser funktional leichtgewichtige Kern durch zusätzliche Module iterativ ausgebaut werden kann. Reicht also beispielsweise die im Kern vorhandene Finanzfunktionalität nicht aus, kann Advanced Financials angeflanscht werden – oder Advanced Order Management, Advanced Purchasing usf. Auch Branchenspezifika werden über solche Zusatzmodule abgedeckt, wie etwa Abomodelle mit NetSuite Subscription Management oder NetSuite Software Management für Softwarehersteller.

Obgleich das Angebot an NetSuite-Modulen stetig wächst, ist der vom Hersteller im Standard gebotene Funktionsumfang nicht mit dem der Oracle Cloud Applications vergleichbar. In Fällen, in denen der Standardfunktionsumfang nicht ausreicht, sieht der NetSuite-Lösungsansatz die Verwendung von SuiteApps vor *(vgl. https://www.suiteapp.com/)*, die NetSuite auf einem eigens dafür eingerichteten elektronischen Marktplatz anbietet. SuiteApps können als Microservices interpretiert werden, die dann für spezifische funktionale (zum Beispiel Service-Außendienst) oder techni-

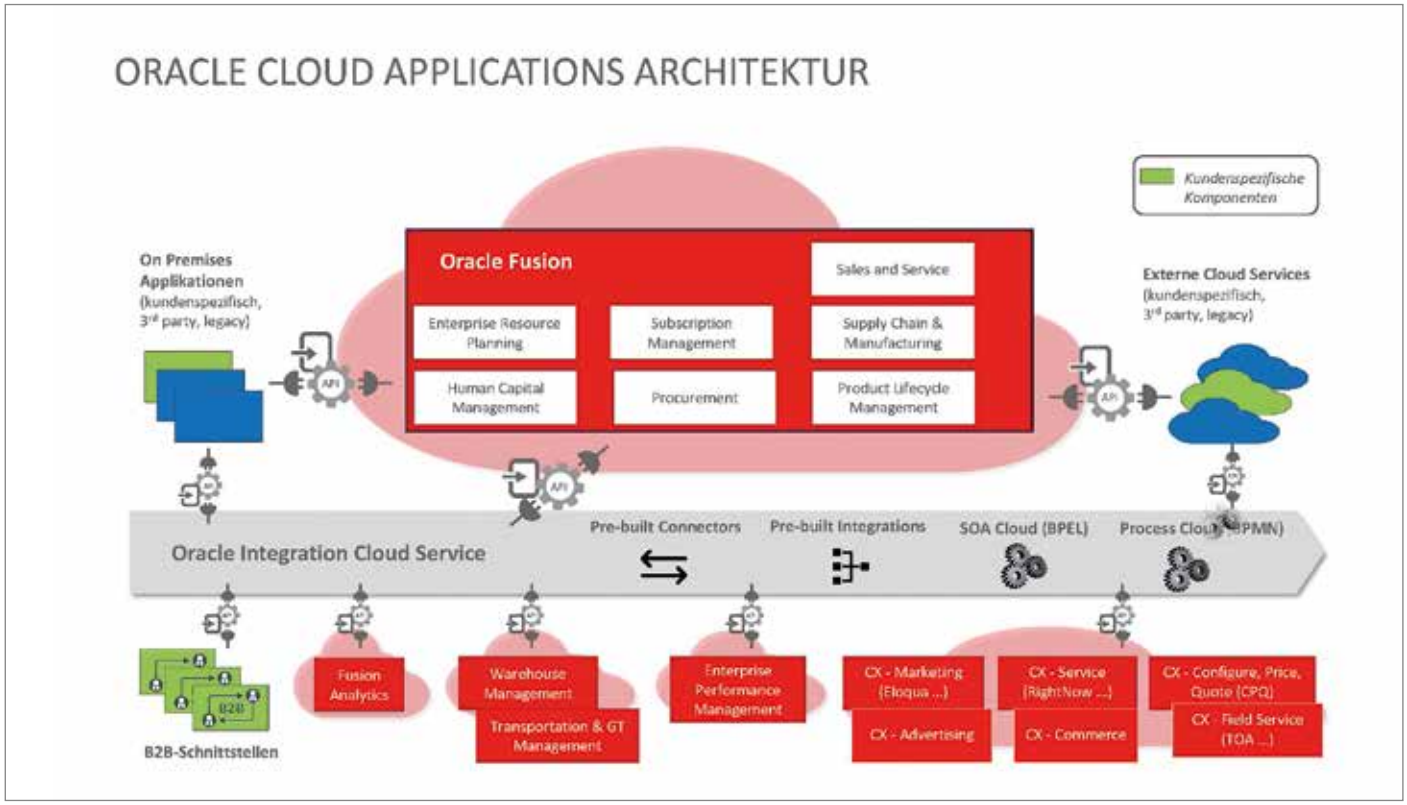


Abbildung 2: Architektur einer Enterprise-SaaS-Lösung am Beispiel der Oracle Cloud Applications (Quelle: PROMATIS)

sche Anforderungen (zum Beispiel mehrsprachiges Scannen) in verschiedenen Branchen (zum Beispiel ERP für See- und Yachthäfen) oder geografischen Regionen (zum Beispiel deutsche Bankenschnittstelle) angeboten werden.

Mit der „Built for NetSuite" (BFN)-Initiative bietet NetSuite im SuiteCloud Developer Network (SDN) Partnern Ausbildungs- und Beratungsleistungen sowie BFN-Zertifizierungen an, um für die im Marktplatz von Partnern angebotenen SuiteApps die vom Hersteller NetSuite gewohnten Sicherheits- und Qualitätsstandards gewährleisten zu können. SuiteApps werden in drei Kategorien eingeteilt:

- Native
  Die SuiteApp ist vollumfänglich durch NetSuite BFN-verifiziert und residiert komplett auf der SuiteCloud-Plattform.
- Integrated
  Der größte Teil der SuiteApp befindet sich außerhalb der SuiteCloud-Plattform. Es handelt sich also um eine externe Komponente, die über eine Integration – möglicherweise mit einem Standardkonnektor – angebunden werden muss. NetSuite verifiziert hierbei nur die Integrationskomponenten, nicht aber die Lösung selbst.
- Hybrid
  Hierbei handelt es sich um eine SuiteApp mit einem Mix aus nativen und integrierten Komponenten, die auch nach den jeweils entsprechenden Verfahren BFN-verifiziert sind.

Analog zu den SuiteApps des NetSuite-Marktplatzes können kundenspezifische SuiteApps erstellt und integriert werden. Hierzu wird SuiteCloud IDE *(https://www.netsuite.com/portal/platform/developer/suitecloud-ide.shtml)*, die Entwicklungsumgebung von NetSuite, benutzt. In dieser Umgebung kann direkt in der NetSuite-Infrastruktur zu 100% kompatible SuiteApp-Funktionalität entwickelt werden, wobei hauptsächlich JavaScript- und HTML-Editoren zum Einsatz kommen.

Neben der gefälligen Usability und der leichtgewichtigen NetSuite-Architektur überzeugen in Wachstumsunternehmen vor allem diese Möglichkeiten zur kundenspezifischen Erweiterung und Weiterentwicklung der NetSuite-Lösung. Dies gilt insbesondere in Technologieunternehmen, die eigene IT-Ressourcen zur Verfügung haben. Dass die professionelle Erweiterung eines Unternehmenssoftware-Systems nicht nur technisches Know-how benötigt, sondern vor allem auch betriebswirtschaftliches und Prozesswissen, wird spätestens im Betrieb der neu entwickelten Kundenmodule offensichtlich. Dazu später aber mehr.

Agile Implementierung

Wachstumsunternehmen zeichnen sich zumeist durch ihre Agilität aus, die ihnen eine schnelle und konsequente Ausrichtung an den Bedürfnissen des Markts ermöglicht. Geschäftsmodelle, Geschäftsprozesse und Organisationsstrukturen, aber auch die Unternehmenssoftware-Systeme unterliegen einem kontinuierlichen Verbesserungsprozess, der auf einem konsequenten Monitoring der Unternehmensperformance gründet. Die Agilität der Unternehmen zeigt sich zudem in den agilen Verfahren, die in der Umsetzung von strategischen Programmen und Projekten angewendet werden. Ein Blick zurück zur Darstellung der NetSuite-Lösungsarchitektur in *Abbildung 3* macht deutlich, wie gut sich NetSuite mit agilen Vorgehensweisen in der Implementierung verträgt – zweifellos einer der Gründe, weshalb NetSuite gerade in Wachstumsunternehmen so beliebt ist. In der agilen Implementierung von NetSuite fungiert das Kernmodul SuiteSuccess regelmäßig als MVP (minimum viable product), das als funktionsfähiges Pilotsystem in den produktiven Betrieb übernommen wird. In den folgenden Sprints werden dann die Er-

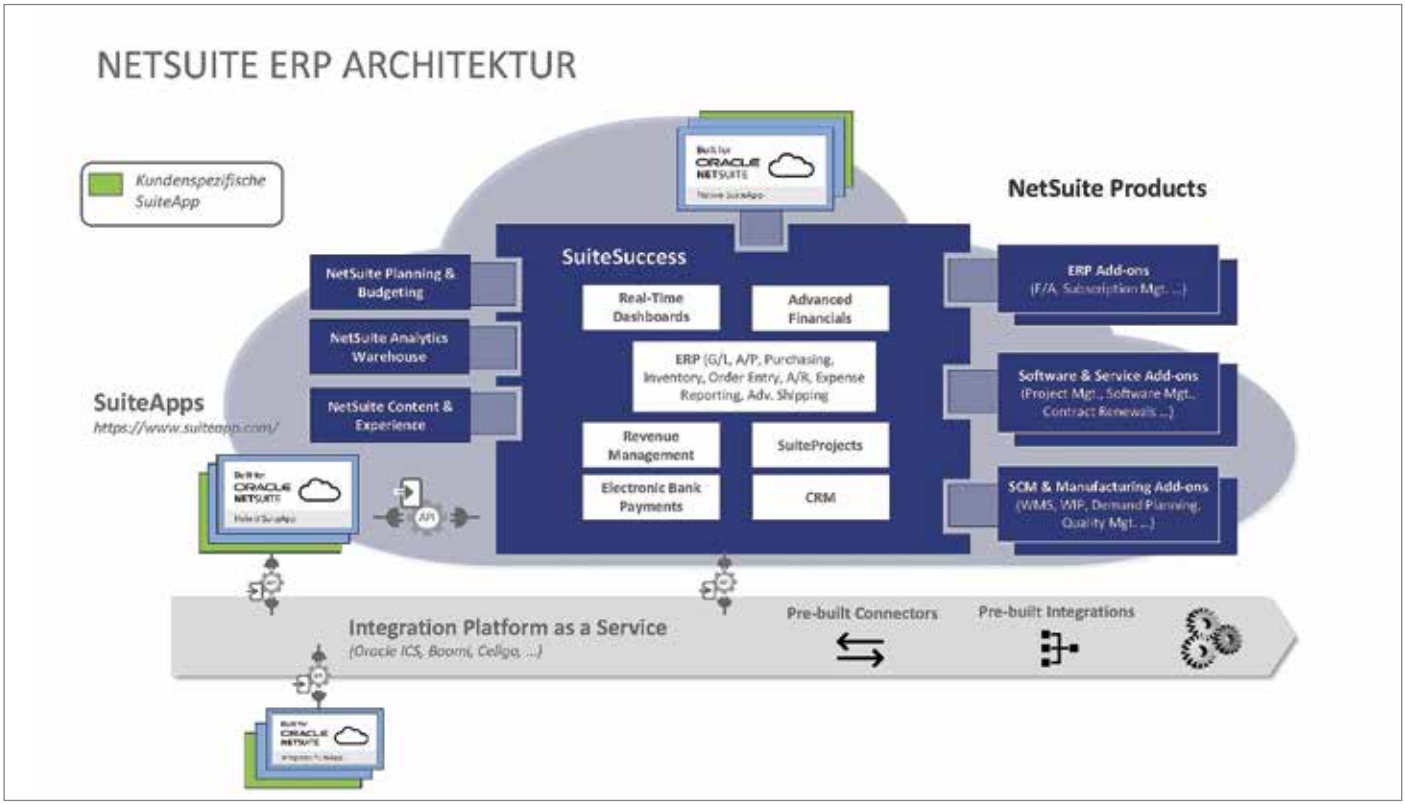


Abbildung 3: Architektur der NetSuite-ERP-Lösung (Quelle: PROMATIS)

weiterungsmodule und die SuiteApps dem Produktivsystem hinzugefügt.

Allerdings gestaltet sich dieses agile Implementierungsvorgehen in der Praxis nicht ganz so einfach. Oftmals begreifen Kunden NetSuite fälschlicherweise als ein System von Microservices, die dann in einem Continuous-Delivery-Verfahren im Unternehmen produktiv geschaltet werden. *Abbildung 4* zeigt die Struktur der NetSuite-ERP-Lösung als System von Geschäftsprozessen, die teils nebenläufig sind, vielfach aber durchaus komplexe kausale Abhängigkeiten aufweisen. Diese Abhängigkeiten führen bei der informationstechnischen Umsetzung zur modul- und prozessübergreifenden Nutzung von Stamm- und Bewegungsdaten und zu prozesslogischen Abhängigkeiten zwischen NetSuite-Modulen und SuiteApps, die sich in zahlreichen Daten- und Steuerinterfaces manifestieren. Oft sind es deshalb lediglich nicht-native SuiteApps, die sich in den Sprints als echte Microservices verwenden lassen.

Aus diesen Betrachtungen heraus kann eine modulweise Konfiguration und Entwicklung zwar Sinn ergeben, ein modulweises Delivery aber offensichtlich nicht. In der Praxis geht man deshalb bei der Festlegung der Sprint Backlogs von den Geschäftsprozessen aus. Gut geeignet sind Prozesse mit wenigen kausalen Abhängigkeiten von anderen Prozessen, insbesondere solchen, die noch nicht im aktuellen Produktivsystem implementiert sind. Sind bei der Sprintplanung auch aufwendige Datenmigrationen zu berücksichtigen, die sich über Prozessgrenzen hinweg erstrecken, fasst man die betroffenen Prozesse gerne zu Clustern zusammen, die dann den Sprint Backlog aufspannen.

Die Ausführungen zur agilen Implementierung machen deutlich, dass NetSuite-Einführungsprojekte in ihrer Komplexität nicht zu unterschätzen sind. Sie erfordern eine hohe Kompetenz des Projektteams und ein professionelles Projektmanagement. Leider werden NetSuite-Einführungsprojekte in der Praxis von den Stakeholdern im Kundenunternehmen immer wieder unterschätzt. In der Meinung, passend qualifizierte eigene Mitarbeiter in ausreichender Quantität und mit verwendbarer Praxiserfahrung zur Verfügung zu haben, kauft man dann „einige Beratertage nach Bedarf" ein und ist verwundert, wenn niemand das Projekt verantwortlich ins Ziel steuern kann und will.

Mitwirkungsleistungen des Kunden

In größeren Unternehmen ist zu beobachten, dass Unternehmenssoftware-Projekte heute nicht – wie dies im On-Premises-Zeitalter der Fall war – durch die IT-Abteilung getrieben werden, sondern durch die Fachabteilungen, die für „ihre" Cloud Services dann auch die Verantwortung übernehmen und dementsprechend engagiert die Implementierung unterstützen. Im Unterschied dazu werden in technologieorientierten Wachstumsunternehmen, deren Stakeholder zumeist eine ausgeprägte Technikaffinität zeigen, Unternehmenssoftware-Projekte immer noch sehr stark als IT-Projekte gesehen. Insofern ist die Bereitschaft des Kunden, eigene Kapazitäten in die Projektarbeit einzubringen, sehr zu begrüßen, geht aber zuweilen am echten Bedarf vorbei beziehungsweise beschränkt sich auf die Bereiche der Schnittstellenentwicklung und des Reportings. In puncto Mitwirkungsleistungen des Kunden besteht der höchste Bedarf in der Rolle der Geschäftsprozessexperten, die ihre betriebswirtschaftlichen Kenntnisse und branchenspezifischen Erfahrungen mit IT-Affinität verbinden und damit dem neuen Unternehmenssoftware-System den kundenspezifischen Stempel aufdrücken.

An dieser Stelle ist es wichtig, was im Zusammenhang mit NetSuite in Wachstumsunternehmen der Begriff Best Practice bedeutet. Die Verfügbarkeit einer großen Menge von Best-Practice-Prozessen und -Funktionalitäten ist ein wesentliches Alleinstellungsmerkmal von NetSuite im Wettbewerb. So ist ja bereits SuiteSuccess eine umfassend vorkonfigurierte Best-Practice-Lösung, die in kleineren Wachstumsunternehmen schon alle benötigten Prozesse und Funktionalitäten beinhaltet. Best Practice bedeutet aber nicht, dass dem Unternehmen eine „Standardlösung" zur Verfügung gestellt wird und in der Implementierung keine Mitwirkungsleistungen mehr zu erbringen wären. Im Übrigen müssen einige Unternehmen im Verlauf des Implementierungsprojekts die schmerzliche Erfahrung machen, dass ihre Geschäftsprozesse nicht einfach „Standard sind" beziehungsweise dass es einen Standard für Unternehmensprozesse schlichtweg nicht gibt. Was es aber gibt – und hier spielt NetSuite seine Stärken voll aus – ist eine auf Best Practice aufgebaute Software, die sich in Kundenprojekten mittels Konfiguration und Parametrisierung komfortabel und

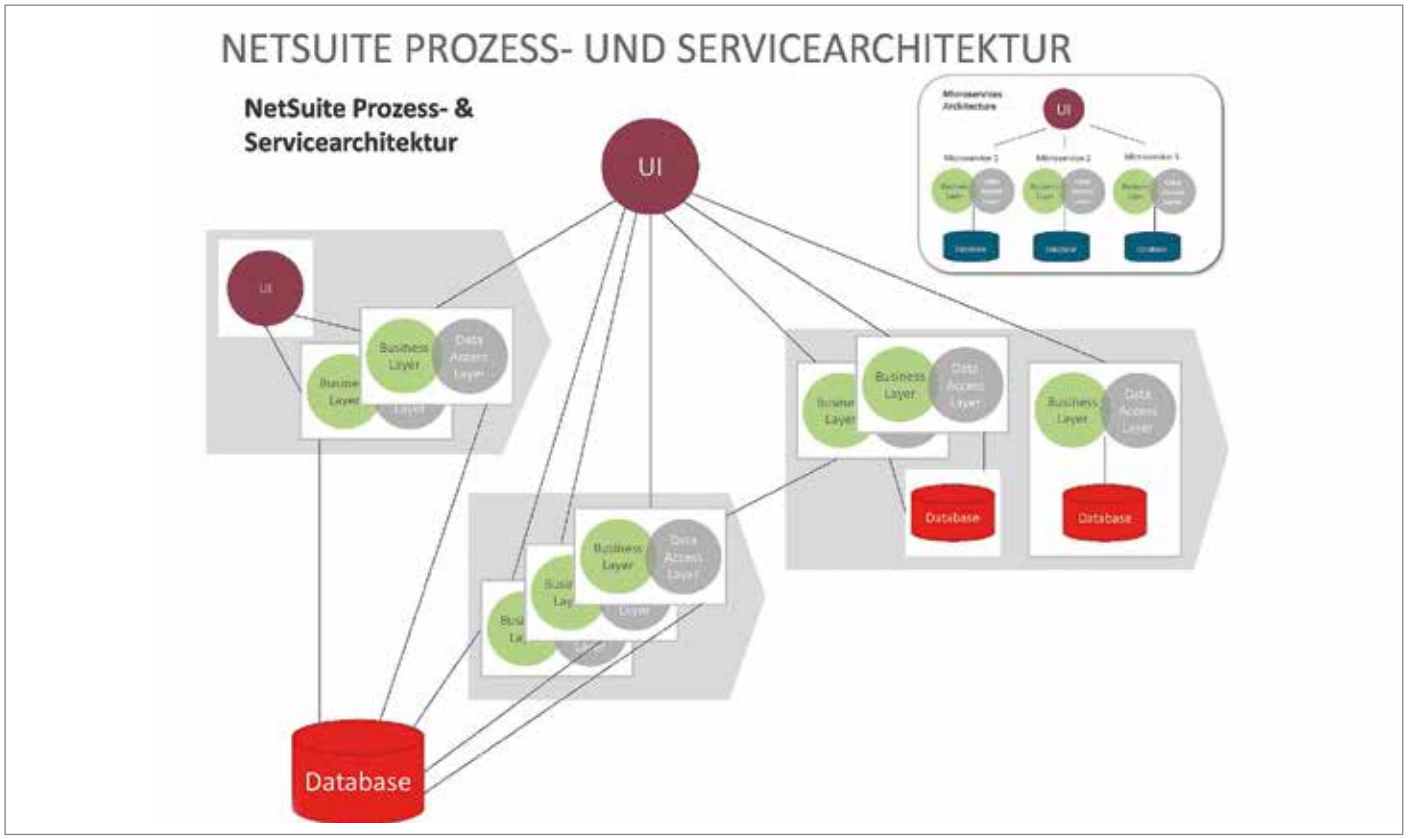


Abbildung 4: NetSuite Prozess- und Servicearchitektur (Quelle: PROMATIS)

wirksam an kundenspezifische Bedürfnisse adaptieren lässt.

Im Zusammenhang mit der Einführung einer NetSuite-Lösung ist die Beauftragung einer schlüsselfertigen Implementierung durch den Kunden äußerst selten. Dazu sind die Anforderungen zu Projektbeginn oft noch nicht ausreichend klar und vollständig spezifiziert und es liegt kein verbindliches Lastenheft vor. Typischerweise wird die Implementierung als Werkvertrag – in selteneren Fällen mit Festpreis- oder Kostendachelementen – beauftragt. Dies hat den Vorteil, dass ein zwischen Kunde und Implementierungspartner abgestimmtes Statement of Work als Pflichtenheft vorliegt, in dem Aufgaben und Verantwortlichkeiten der beteiligten Parteien einvernehmlich festgelegt sind. Interessant ist, dass gerade kleinere Wachstumsunternehmen nicht selten sehr selbstbewusst in die Einführungsprojekte gehen und lediglich einige Unterstützungstage „on demand" beauftragen. Für die Implementierung bis hin zur agilen Inbetriebnahme übernehmen sie dann selbst die Verantwortung und verzichten – bewusst oder unbewusst – auf die Absicherung, die die Gewährleistungspflichten eines Werkvertragspartners mit sich bringen würden.

Unabhängig von der Art der Beauftragung entstehen in einem NetSuite-Einführungsprojekt Mitwirkungspflichten, die durch geeignetes Personal des Kunden oder durch von ihm beauftragte Dritte erfüllt werden müssen:

- Governance
  Für das Einführungsprojekt oder ein komplettes Digitalisierungsprogramm muss eine geeignete Governance-Struktur aufgebaut werden. Das reicht von der Etablierung eines strategischen Gremiums oder Steuerungskomitees über die Bestimmung eines kundenseitigen Projektleiters bis hin zu den zugehörigen Governance-Prozessen einschließlich der Vorgabe eines verbindlichen Projektvorgehensmodells.
- Einbringen von Prozess- und betriebswirtschaftlichem Fachwissen
  Im Rahmen von Workshops und Interviews, der Bearbeitung von Fragenkatalogen, Reviews von Dokumenten und selbstverständlich praktischer Arbeit am Testsystem bringen Prozess- und Funktionalexperten des Kunden ihr Fachwissen ein. Da in NetSuite bereits Best-Practice-Prozesse und -Funktionen implementiert sind, beschränkt sich der Wissenstransfer in der Regel auf Gap-Analysen und die kundenspezifische Parametrisierung.
- Systemtests und Abnahmen
  Im Rahmen der Implementierung werden durch Kundenmitarbeiter Testfälle zusammengestellt und für die Implementierung und zur Nutzung in der nachfolgenden Betriebsphase dokumentiert. Im Unterschied zu Systemspezifikationen, für deren Erstellung ein gewisses Abstraktions- und Strukturierungsvermögen erforderlich ist *(siehe [3, 4])*, ist für die Testfallerstellung ausschließlich betriebswirtschaftliches Fachwissen erforderlich. Wichtig ist, dass die Bedeutung der Testfälle für die Qualität und Vollständigkeit des neuen Systems nicht unterschätzt wird. Die Testfälle bilden dann das Gerüst für System- und Abnahmetests, die durch die Kundenmitarbeiter – zumeist unterstützt durch Experten des Implementierungspartners – durchgeführt werden. Als Feedback entstehen Testprotokolle, die in die weitere Verbesserung des implementierten Systems einfließen.
- Kundenspezifische Entwicklung
  Gerade in technologieorientierten Wachstumsunternehmen besteht eine große Be-

reitschaft, sich in der kundenspezifischen Entwicklung von Zusatzmodulen, Integrationen, Auswertungs- und Berichtsfunktionen einzubringen. Dies hat den Vorteil, dass der Kunde nach erfolgreicher Inbetriebnahme des Systems Aufgaben der Betriebsunterstützung und Weiterentwicklung in Eigenregie oder mit nur geringer externer Unterstützung durchführen kann. Durch die Verwendung von Standardtechnologien lassen sich hierbei gerade mit NetSuite erhebliche Kostenvorteile und eine hohe Agilität im kontinuierlichen Verbesserungsprozess erzielen.

- Datenmigration
  NetSuites Standardtechnologien spielen auch in der Datenmigration ihre Vorteile aus, wenn es um die Entwicklung von Extraktions-, Transformations- und Ladefunktionalitäten geht. Da viele Wachstumsunternehmen bereits in den frühen Phasen der unternehmerischen Tätigkeit sehr große Datenmengen aufgebaut haben, wird die Möglichkeit, schnell und einfach Skripte für die Datenbereinigung und die Anreicherung von Daten bereitzustellen, auf Kundenseite als eine große Stärke der NetSuite-Lösung gesehen. Allerdings soll nicht unerwähnt bleiben, dass selbst in hochagilen Implementierungsprojekten auf eine intensive Mitwirkung der Anwender im Rahmen von Migrationstests und Cleansing-Aktivitäten nicht verzichtet werden kann.

Zusammenfassung und Ausblick

Im vorliegenden Beitrag ist das Phänomen NetSuite untersucht worden, das sich gerade in Wachstumsunternehmen, die NetSuite als ihr „Lieblings-ERP" begreifen, beobachten lässt. Im Vergleich mit Oracle Cloud Applications als exemplarischem Vertreter der Enterprise-SaaS-Lösungen wurden Aspekte der NetSuite-Architektur herausgearbeitet, die die besondere Eignung von NetSuite für Wachstumsunternehmen begründen. Mit den Ausführungen zur agilen Implementierung der NetSuite-Lösung wurde aufgezeigt, wie gerade technologieorientierte Wachstumsunternehmen in NetSuite eine hoch skalierbare Unternehmenssoftware-Lösung finden. Überlegungen zur Gestaltung des NetSuite-Implementierungsprojekts als „Joint Venture" des Kunden und des Implementierungspartners runden den Beitrag ab.

Für den Hersteller Oracle ist NetSuite zweifellos nicht nur eine der erfolgreichsten, sondern vor allem eine der dynamischsten Produktlinien. Dies zeigt sich im rasanten Ausbau der NetSuite-Produktfunktionalität, die dem Kunden immer mehr branchenspezifische Funktionalität bietet, aber auch immer mehr SuiteApps in NetSuite-Standardprodukte gießt. Letzteres hilft, die nicht zu unterschätzende Komplexität im Rahmen zu halten, die die Verwendung einer großen Zahl von SuiteApps unterschiedlicher Fremdhersteller mit sich bringt. Interessant ist weiterhin, wie sich die beiden Produktlinien NetSuite und Oracle Cloud Applications aufeinander zubewegen, indem im Oracle Integration Cloud Service immer mehr vorgefertigte Integrationen zwischen Modulen der beiden Produktlinien mitgeliefert werden. Zudem werden Produkte aus der Oracle-Cloud-Applications-Welt, die auch für NetSuite-Anwender hohe Nutzwerte versprechen, in der NetSuite-Produktlinie assimiliert, zum Beispiel die Cloud Services NetSuite Planning & Budgeting und NetSuite Content & Experience.

Referenzen

[1] Gartner Magic Quadrant 2021 for Cloud Core Financial Management Suites for Midsize, Large and Global Enterprises.

[2] Gartner Magic Quadrant 2021 for Cloud ERP for Product-Centric Enterprises.

[3] F. Schönthaler, G. Vossen, A. Oberweis, T. Karle (2011): Geschäftsprozesse für Business Communities: Modellierungssprachen, Methoden, Werkzeuge. Oldenbourg Wissenschaftsverlag, München.

[4] G. Vossen, F. Schönthaler, S. Dillon (2017): The Web at Graduation and Beyond: Business Impacts and Developments. Springer International Publishing, Cham, CH.

Dr. Frank Schönthaler
frank.schoenthaler@promatis.de

Als geschäftsführender Gesellschafter der in der D.A.CH-Region und USA tätigen PROMATIS Gruppe verantwortet Dr. Schönthaler die Unternehmens- und Produktstrategie und führt das operative Kerngeschäft. Mit mehr als 30 Jahren internationaler Erfahrung in der Beratungs- und Softwarebranche ist er ein ausgewiesener Experte für prozessorientierte Oracle-Digitalisierungslösungen. Mehr Lebensqualität durch IT ist das Ziel, das er in seiner ehrenamtlichen Aufgabe als Präsident der Integrata-Stiftung für humane Nutzung der Informationstechnologie verfolgt.

NetSuite wächst und wächst – und steht doch erst am Anfang

Von Martin Kalkuhl, Alta Via

Die Disruption im Cloud-ERP-Markt hat in Deutschland noch nicht wirklich begonnen. Warum das so ist, wird im folgenden Text beschrieben.

Ohne Server, immer und überall

NetSuite wurde 1998 – damals noch unter dem Namen NetLedger – gegründet. Ziel war es, eine eine Buchhaltungslösung zu schaffen, die völlig ohne lokale Server auskommt und von überall und egal mit welchem Gerät nutzbar ist – Public Cloud eben. SalesForce CRM wurde im selben Jahr gegründet und beide Start-ups wurden von Larry Ellison finanziert. Ein CRM in der Cloud konnte sich schneller global entwickeln, weil es keine Transaktionen mit buchhalterischen Auswirkungen hat und das Bauchgefühl gerade hier im zentraleuropäischen Raum der Cloud gegenüber im Finanzbereich eher skeptisch war.

ERP-Marktführer NetSuite

Im Jahr 2010 war das Leitthema der CeBIT „CLOUD". Google, Facebook und iPod/iPhone/iTunes waren schon ein paar Jahre auf dem Markt. Die deutsche Welt hat damals schon gedacht: Cloud – alles klar, ich kenne mich aus. Das gilt Stand heute tatsächlich für einige Bereiche, aber noch nicht im Enterprise-Resource-Planning-Bereich (ERP). Die Anbieter von vollumfänglichen Business Suites inklusive Customizing-Umgebung, sicheren Release-Updates, stabilen APIs (Application Programming Interfaces) sowie Verfügbarkeit hinsichtlich Steuern, Reports und der lokalen Anforderungen aller Länder der Welt sind sehr rar. NetSuite ist hier mit ca. 31.000 Kunden *[1]* mit Abstand der Marktführer, wobei die Mehrheit der Kunden aus den USA stammt – noch.

„Cooles Gütesiegel"

NetSuite hat in seinem App Store „suiteapp.com" ein ziemlich umfangreiches Angebot an nativen und hybriden Add-ons zur Verfügung. Die nativen Apps laufen tatsächlich zu 100 % im Kunden-Account und halten natürlich auch den Release-Wechseln – zwei pro Jahr – stand. Dafür sorgt das BFN-Siegel „Built-for-NetSuite". Das ist schon cool und diese Stärke wird den meisten Kunden und Usern auch über die Zeit bewusst.

Globale Skalierung, lokale Experten

Hersteller dieser Apps sind die NetSuite Solution Provider, Alliance oder SDN (SuiteCloud Developer Network) Partner. Wenn ein Kunde global skaliert, braucht man als Steering-Partner die Unterstützung von lokalen Experten, die zum Beispiel die brasilianischen, italienischen oder thailändischen Lokalisierungsanforderungen kennen und ein sogenanntes „Bundle" gebaut haben. Die Zusammenarbeit unter den Partnern funktioniert in der Regel sehr gut. Natürlich ist auch das NetSuite Professional Service Team ein guter Partner. Aber je nach Land und Anforderung muss man schauen, wer da am besten mit wem kooperiert und welcher Partner sich mit den jeweiligen

Kundenanforderungen aus dem umfangreichen Produktportfolio von NetSuite am besten auskennt.

Tausendfacher „Proof of Concept" (PoC)

Es muss oft auch alles recht schnell gehen. Nicht selten beginnt die Reise eines Start-ups mit einer kleinen Truppe… 15 Leute, die sich aus ein paar Gründern, Software-Entwicklern und Marketing/Vertrieb/Operations zusammensetzen. Die brauchen am Anfang eine bezahlbare Lösung, die dann mit dem Wachstum der Company Schritt hält. Da kommt es schnell vor, dass zwei, drei, vier Mitarbeiter pro Woche eingestellt werden, die alle in die Firmenprozesse eingearbeitet werden müssen. Dabei kann es auch passieren, dass die Prozesse zusätzlich im laufenden Betrieb angepasst werden. Bei einer derartig agilen Vorgehensweise ist die Stabilität der Performance des ERP wichtig: User, Transaktionen, Niederlassungen, Länder und eventuell noch parallele Schnittstellen mit den Vertriebs-Channels, alles wächst gleichzeitig. Das muss das ERP hinkriegen. NetSuite hat das tausendfach bewiesen – Airbnb, Snapchat, Spotify, BOX, GoPro, Byrd, Spryker, SellerX, MAMBU oder SodaStream sind hier nur einige der beispielhaften Success Stories.

„Customizing is not a buzz word anymore"

In der Vergangenheit haben in der Client-Server- beziehungsweise On-Premises-Welt die Anpassungen/Customizations die IT-Abteilungen spätestens zum Major-Release-Wechsel herausgefordert, um nicht zu sagen „in den Wahnsinn" getrieben. Denn das Einspielen der neuesten Softwareversion kann nicht einfach so geschehen, weil die vorherigen Anpassungen nicht in das höhere Release übernommen werden können und somit alle erneut umgesetzt werden müssen. Das ist fast gleichbedeutend damit, ein neues Projekt aufzusetzen. „Da kann ich mir ja auch gleich eine neue Software kaufen" steht dann plötzlich auch als Alternative auf der Tagesordnung. Dieser Option, in eine reine Cloud-Lösung zu wechseln, stehen dann einige sehr offen gegenüber, andere bleiben allerdings skeptisch. Fakt ist: Für einige Branchen mit besonderen Compliance-Anforderungen ist ein Cloud-Hosting schon gegenüber Behörden keine Option, ebenso in der Life-Science-Industrie wie Pharma-, Chemie- oder API (Active Pharmaceutical Ingredients) sowie Nahrungsmittel-Produktion. Für die allermeisten Industrien gilt das jedoch nicht.

Die „Eh-da-Kosten"

Bei der TCO-Rechnung (Total Cost of Ownership) sollte jeder wirklich mal ganz ehrlich zu sich sein und wirklich alle Kosten einrechnen, die zum Betrieb der aktuellen Softwarelandschaft gehören: Rechenzentrum, Räumlichkeiten, Server-Hardware, Server- und Client-Lizenzen, Netzwerk, Hardware-Infrastruktur, Backup, Strom, Klimaanlage, Versicherungen und Sicherheitsmaßnahmen, aber auch die anderen „Eh-da-Kosten" wie IT-Mitarbeiter, die man ja braucht, um diese ganze Softwarelandschaft aufrechtzuerhalten. Bei der Überlegung und dem Kostenvergleich mit einer Public-Cloud-Lösung werden diese Kosten oft nicht ehrlich gerechnet. Kurz und knapp kann man jedoch sagen: Bei einer On-Premises-/Client-Server-Landschaft sollte die Zeit, die man aufwenden muss, um den Betrieb der Soft- und Hardware aufrechtzuerhalten, nicht unterschätzt beziehungsweise vergessen werden. Wertvolle Zeit, die man lieber nutzen sollte, um seine Prozesse zu automatisieren. Bei einer Public-Cloud-Lösung kann man sich vollkommen auf die Optimierung der Prozesse konzentrieren.

Gibt es eigentlich ein Projekterfolgsrezept?

So einfach lässt sich das sicherlich nicht sagen. Aber wie in jedem Software-Projekt – Cloud hin oder her – ist die grundsätzliche Projektvorgehensweise, die Qualität der Projektverantwortlichen, die Chemie aller Beteiligten untereinander sowie ein transparentes und offenes Verhalten in den kniffeligen Situationen sehr entscheidend, um am Ende den Sack zuzumachen = den GoLive erfolgreich hinzubekommen. Das reine Versprechen auf Präsentations-Slides schafft anfangs vielleicht ein wenig Vertrauen, aber bereits dann, wenn es Richtung GoLive geht, braucht es die Bereitschaft aller Beteiligten „mitzumachen" und keine Slide-Versprechen mehr: Der eine arbeitet dann mal etwas länger, der andere hält sich dann halt nicht mehr an die hundertprozentige Umsetzung aller gewünschten Features. Diese müssen dann im laufenden Betrieb umgesetzt werden – genauso wie die vermutlich weiter folgenden funktionalen Wünsche der zweiten Projektphase.

Disruption – bin ich eigentlich auch davon betroffen?

Ganz klar: ja. Warum hat die Disruption aber noch gar nicht richtig begonnen? Es gibt in Europa zum Beispiel ca. 450 Millionen Menschen und Millionen von Firmen, die alle mittelfristig „in die Cloud" wollen. Auf die eine oder andere Art haben die das ja auch schon gemacht. Doch mal zurückgerechnet: Wenn NetSuite in den vergangenen zwölf Jahren von ca. 6.000 Kunden auf 31.000 gewachsen ist und es allein in Europa Millionen von potenziellen Firmen und Kunden gibt – und eben noch nicht so viele vollumfassende Anbieter wie NetSuite, dann ist da noch Luft nach oben, für NetSuite und seine Partner, die Kunden zu überzeugen, schnellstmöglich in ein wachstumsfähiges, anpassbares System zu investieren. Diese Investition wird sich doppelt lohnen, denn man sagt ja, dass sich in der Regel jedes Business alle drei bis vier Jahre verändert. Da muss dann auch die Software mitgehen, indem diese entweder das neue Business bereits beherrscht oder sie wieder angepasst werden muss. Hier ist also die Customization sehr eng mit der Disruption verbunden.

Wenn man nicht mit der Zukunft geht, geht ein anderer. Also: einfach machen. Schnell. Jetzt.

Quellen

[1] *https://www.netsuite.com/portal/customer-testimonials.shtml*

Martin Kalkuhl
mkalkuhl@altavia.de

Martin ist seit 2011 bei Alta Via maßgeblich für den Vertrieb und das NetSuite Partnergeschäft verantwortlich. Die technischen Grundlagen für das beratungsintensive Anwendersoftware-Geschäft hat er im Studium mit Schwerpunkt Software Engineering in Berlin gelegt. Langjährige Auslandsaufenthalte in kleinen mittelständischen Firmen, viele Reisen und Unternehmergeist haben seinen Charakter mit viel Verständnis für die Sicht des Kunden geprägt. Er lebt mit seiner Frau und seinen beiden Kindern seit 1998 in Berlin, spielt leidenschaftlich Tennis, fährt sehr gerne Ski und – als ehemaliger DJ – liebt die Musik.

ACT GLOBAL, BE LOCAL? Deutsche Compliance im internationalen Umfeld

Katharina Schraft, PROMATIS Gruppe, Ettlingen (TechnologieRegion Karlsruhe)

Oracle NetSuite – die weltweite Nr. 1 für cloudbasierte Unternehmenssoftware – erfreut sich weiter steigender Beliebtheit, insbesondere seit der Akquisition durch Oracle im Jahr 2016. Das System bietet besonders für kleine und mittelständische Unternehmen enorme Vorteile. Gerade die eingebaute Internationalität mit dem Produkt NetSuite OneWorld ist oft ein Entscheidungskriterium für Unternehmen mit großen Wachstumsplänen. Demnach ist es nicht verwunderlich, dass sich NetSuite den schlagkräftigen Slogan „ACT GLOBAL, BE LOCAL" sogar als Trademark gesichert hat [1]. Teilweise werden auch Zweifel laut: Eignet sich ein amerikanisches System tatsächlich für den deutschen Markt? Bei internationalen Organisationen mit Niederlassungen in Deutschland ist NetSuite allerdings schon seit Jahren im Einsatz und weiter auf dem Vormarsch, um bewährte lokale Lösungen zu ersetzen. Dieser Beitrag wird einen Einblick in verschiedene Projekte dieser Art geben, die wir in den letzten Jahren begleitet haben.

Häufig fängt es damit an, dass Kunden ihre Steuermeldung aus NetSuite heraus erstellen möchten oder von einer deutschen Lokalisierung und dem ominösen Multibook – wodurch sich NetSuite „compliant" verhält – gehört haben. In vielen Fällen ist es auch das globale Management, das nicht verstehen kann, dass in Deutschland eine lokale Lösung im Einsatz ist, während der Rest der Unternehmensgruppe bereits glücklich mit NetSuite arbeitet und die Zahlen somit immer für globale Auswertungen vorliegen – tagesgenau (schön bunt auf dem C-Level Dashboard) und nachvollziehbar. Aber man stellt schnell fest, deutsche Compliance ist keine Komponente, die einfach „eingeschaltet" wird. Die GoBD-konforme Buchhaltung (Grundsätze zur ordnungsmäßigen Führung und Aufbewahrung von Büchern, Aufzeichnungen und Unterlagen in elektronischer Form sowie zum Datenzugriff) sieht vor, dass betriebliche Geschäftsvorfälle nachvollziehbar, vollständig, richtig, zeitgerecht und geordnet zu erfassen sind *[2]*. Es geht also hauptsächlich um die Umsetzung von Geschäftsprozessen auf Verwaltungsebene. In Compliance-Projekten steht natürlich oft das ERP-System im Vordergrund. Das ist aber im Normalfall nicht ausreichend. Themen, die neben den offensichtlichen Faktoren wie Steuern und Reporting in einem System-Review zu Beginn überprüft werden, sind beispielsweise die Belegnummerierung, der Aufbau des Kontenplans, das Berechtigungskonzept sowie die vorliegenden installierten Komponenten, auch Bundles oder SuiteApps genannt. Insgesamt führt eine derartige Analyse zu einer kundenspezifischen „Road to German Compliance", also einer Empfehlung für die nächsten Schritte *(siehe Abbildung 1)*.

System Compliance – die Basis

Zu Beginn ist es wichtig, ein paar Details über NetSuite zu wissen, die einen großen Einfluss auf ein Compliance-Projekt haben können. Unserer Erfahrung nach nimmt in multinational genutzten NetSuite-Systemen das globale Setup einen sehr großen Raum ein. Einstellungen, die in den Features und Accounting Preferences getroffen werden, haben einen Einfluss auf alle Ländergesellschaften, die dieses System nutzen. Es gibt auch einige niederlassungsspezifische Einstellungen; jedoch wird das Grundgerüst global verwendet. Aus diesem Grund ist es von hoher Notwendigkeit, bei der Planung und Umsetzung von nationalen Initiativen stets einen Administrator mit einem Gesamtüberblick einzubeziehen. Sicher kann jeder erfahrene Berater mit Admin-Rechten nahezu alles herausfinden, jedoch geht es auch darum, Design-Entscheidungen richtig zu verstehen oder die tatsächlichen Stakeholder zu erreichen.

NetSuite ist GoBD-zertifiziert; ein IDW PS880 Zertifikat kann über den zuständigen Ansprechpartner/Account Manager auf der NetSuite-Webseite eingeholt werden, falls diese Unterlagen zum Beispiel für eine Verfahrensdokumentation oder Ähnliches benötigt werden. Folglich kann das System so aufgesetzt und genutzt werden, dass auch die DSGVO abgedeckt wird. Globale Einstellungen, die sich herausgestellt haben, sind beispielsweise:

- Lückenlose Belegnummerierung
- Unveränderbarkeit von Transaktionen
- Genehmigungsworkflows (Vier-Augen-Prinzip)
- Berechtigungskonzept

Zur Belegnummerierung bietet NetSuite drei Typen:

- Dokumentennummer – ist automatisch aktiv, editierbar und wird für die Außenkommunikation verwendet, zum Beispiel eine Rechnungsnummer.
- Transaktionsnummer – kann separat aktiviert werden, ist nicht änderbar und lückenlos. Diese beiden Nummern werden zum Zeitpunkt der Transaktionserstellung erzeugt.
- GL-Audit-Nummer – kann im Zuge des Periodenabschlusses auf alle buchenden Transaktionen vergeben werden.

Abbildung 1: Road to German Compliance (Bildquelle: https://www.visitcalifornia.com, Anpassung durch PROMATIS)

Hier können Nummerierungssequenzen sehr feingranular definiert werden.

Transaction Locking ist ein Thema, das in NetSuite eine Herausforderung darstellt. Prinzipiell sind Transaktionen für Änderungen gesperrt, wenn die zugehörige Periode geschlossen wurde. Dies ist jedoch für die Nutzung in Deutschland nicht ausreichend. Hierbei muss ein Beleg gesperrt werden, sobald er gebucht wurde – was in NetSuite bedeutet, dass es bei der finalen Genehmigung eintreten muss. Das kann durch die Definition von Workflows oder die Installation von Bundles (die dann ebenfalls Workflows verwenden) realisiert werden.

Workflows werden häufig auch bei der Abbildung eines Vier-Augen-Prinzips im System genutzt. Hier können Genehmigungsschritte definiert werden, sodass Belege anhand verschiedener Kriterien mitunter einen mehrstufigen Prozess durchlaufen. Die einzelnen Schritte werden an der jeweiligen Transaktion transparent dokumentiert. Wichtig ist hier, dass manuelle Eingriffe möglichst auf ein Minimum reduziert werden. Außerdem sollten die Workflows so einfach wie möglich und so komplex wie nötig sein. In der Design-Phase sind die Anforderungen teilweise sehr umfassend und gehen deutlich über das hinaus, was für die Compliance notwendig ist. NetSuite unterstützt bei der Definition von Workflows mit einer intuitiven grafischen Benutzeroberfläche, die exemplarisch in *Abbildung 2* gezeigt wird.

Im Bereich Berechtigungskonzept gibt es ein paar grundlegende Anforderungen, die gewährleistet sein müssen. Die Sichtbarkeit von Daten muss den DSGVO-Richtlinien entsprechen, nicht jeder darf alle Daten (beispielsweise von Mitarbeiterdatensätzen) einsehen. Zusätzlich muss das System so eingestellt werden, dass insbesondere buchende Transaktionen nicht gelöscht werden können. Rein technisch gesehen ist das in NetSuite zwar möglich – hier ist allerdings die Berechtigungsebene für die verwendeten Rollen ausschlaggebend. Das Setzen der „Edits vs. Full" macht den entscheidenden Unterschied.

Diese Themen obliegen der Verwaltung durch globale Administratoren. Wenn in einer Analyse Änderungspotenziale identifiziert werden, hat deren Änderung oft Auswirkungen auf alle beteiligten Niederlassungen. Darüber hinaus gibt es „lokale" Anforderungen, die den Fachabteilungen bekannt sind und deren Inhalt den meisten Anfragen an Beratungshäuser zugrunde liegen:

Steuer

Wie eingangs schon erwähnt, ist die Umsatzsteuervoreinstellung in den meisten Ländern kein technisches Problem. NetSuite liefert mit Bundles, zum Beispiel „International Tax Reports", den meisten Unternehmen mehr als genug verfügbare Steuerschlüssel, die meistens auch direkt in der Umsatzsteuervoranmeldung (UVA) reflektiert werden. Häufig ist es allerdings so, dass diese Schlüssel entweder nicht installiert oder falsch eingesetzt werden. Somit kann keine korrekte Meldung der Steuer erfolgen. Hierzu stellt NetSuite Steuerfälle bereit – so zeigt *Abbildung 3* eine Steuermatrix, die im Zuge der Projektarbeit zusätzlich erstellt wurde. Sie unterscheidet die Steuerfälle nach Dimension, Produktart und Land des Kunden:

Wenn die Schlüssel in dieser Art verwendet werden, ist der Nutzer seiner korrekten UVA schon etwas näher, da die Schlüssel auf die UVA-Positionen geschlüsselt werden. Informationen liefert NetSuite hier fortwährend aktuell in Suite Answers, zum Beispiel unter der „Answer ID 22253". Diese können auch durch einen Klick auf „Click here for Germany VAT Help Topics" in der UVA-Ansicht erreicht werden, sichtbar am linken oberen Rand in *Abbildung 4*.

Mit dem klassischen Steuer-Reporting, ohne das neue Steuer- und Reporting-Framework SuiteTax – momentan in Deutschland noch nicht verbreitet – ist es nicht möglich, die Zuordnung Steuerschlüssel UVA zu beeinflussen. Es gibt ebenfalls Szenarien im Raum D.A.CH, bei denen Steuerschlüssel existieren – diese aber nicht automatisch überführt werden, zum Beispiel für die Steuer auf Umsätze nach §24 UStG für die Position 80 (Deutschland). In diesen Fällen muss die UVA dann vor dem Hochladen via ELSTER geprüft und gegebenenfalls ergänzt werden.

Kontenplan & Multibook

In Deutschland gibt es sehr präzise Vorgaben, wie im System gebucht und im Anschluss ausgewertet wird. Oftmals gibt es auch lokale Steuerberater, die den Buchungsstoff empfangen und dann im Anschluss beispielsweise eine eBilanz erstellen. Um transparent im deutschen Mandanten buchen zu können, bietet sich die Multibook-Funktionalität von NetSuite an. Sie ermöglicht, weitere Hauptbücher zu pflegen und nach Regeln automatisch sowie manuell zu buchen. *Abbildung 5* bietet einen Überblick über die Verwaltung der Vorgänge.

Aber warum muss es hier Regeln geben? Diese Regeln können sehr einfach sein – zum Beispiel Konto 063500 im IFRS-Buch (International Financial Reporting Standards) entspricht Konto 0635 im HGB-Buch nach SKR04. Es ist jedoch auch möglich, durch das Hinzufügen weiterer Faktoren (zum Beispiel ein Segment „Kostenstelle") komplexere Regeln aufzubauen. Einige Transaktionen bieten die Definition eigener Regeln an, um Abschreibungen im HGB-Buch nach vollkommen anderen Regeln auf andere Konten abzuschreiben oder einen Umsatz nach IFRS vollständig direkt zu realisieren. Diese Funktionalitäten können eingerichtet sowie miteinander orchestriert werden und stellen aufgrund der Komplexität einige Herausforderungen dar. Daher ist die Nutzung von Multibook nicht für jeden User möglich. Die NetSuite-Consulting-Organisation implementiert hier selbst oder bietet ein „Partner Enablement" an. So werden die Partner geschult und geprüft, um dann den Status Multi-Book Authorized zu erhalten, damit im Anschluss (nach Beantragung bei NetSuite) beim Kunden das Multibook Feature eingerichtet werden kann. Bei großen Systemen mit einer Vielzahl von internationalen Niederlassungen kann auch eine komplexere Struktur abgebildet werden, wie dies in *Abbildung 6* veranschaulicht wird.

In diesem Fall muss über die Holding in IFRS ausgewertet werden, während die Tochterunternehmen in DE, IT und US jeweils nach lokalem Recht berichten. Hierzu kann ein primäres Buch für IFRS verwendet werden und das sekundäre Buch kann lokales GAAP (Generally Accepted Accounting Principles) pro Land abdecken. Die verschiedenen Währungen sind dabei kein Problem. Auch separate Bücher sind für die Tochterunternehmen möglich. Insgesamt gibt es ein technisches Limit von fünf aktiven Büchern in NetSuite. Ein Beispiel für die Definition eines Buchs für das lokale GAAP, mit verschiedenen Ländern und Währungen, findet sich in *Abbildung 7*.

Eine Information darüber, welche Bücher die Anwender sehen und verwenden dürfen, wird im System hinterlegt. Wenn die Unternehmensstruktur deutlich ein-

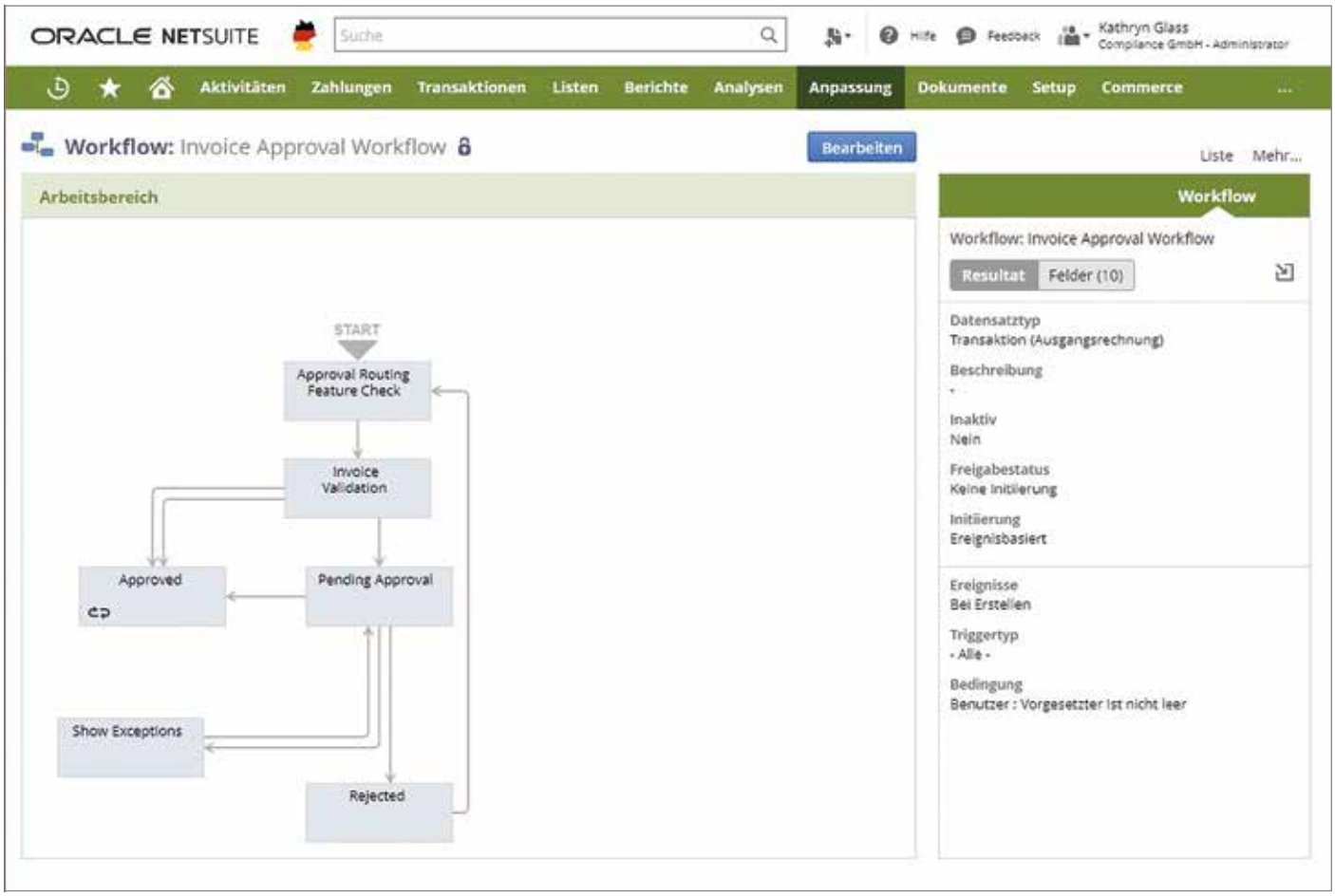


Abbildung 2: Genehmigungs-Workflow (© PROMATIS)

		Waren	Dienstleistungen
Inland	Standard	S2-DE (16%) / S-DE (19%)	
	Reduziert	R1-DE (5%) / R-DE (7%)	
	Steuerfrei	Z-DE	
EU	Standard	ES-DE	ESSS-DE
	Reduziert	ER-DE	
Drittland	Standard	ZX-DE	OS-DE

Abbildung 3: Beispiel debitorischer Steuerfälle (© PROMATIS)

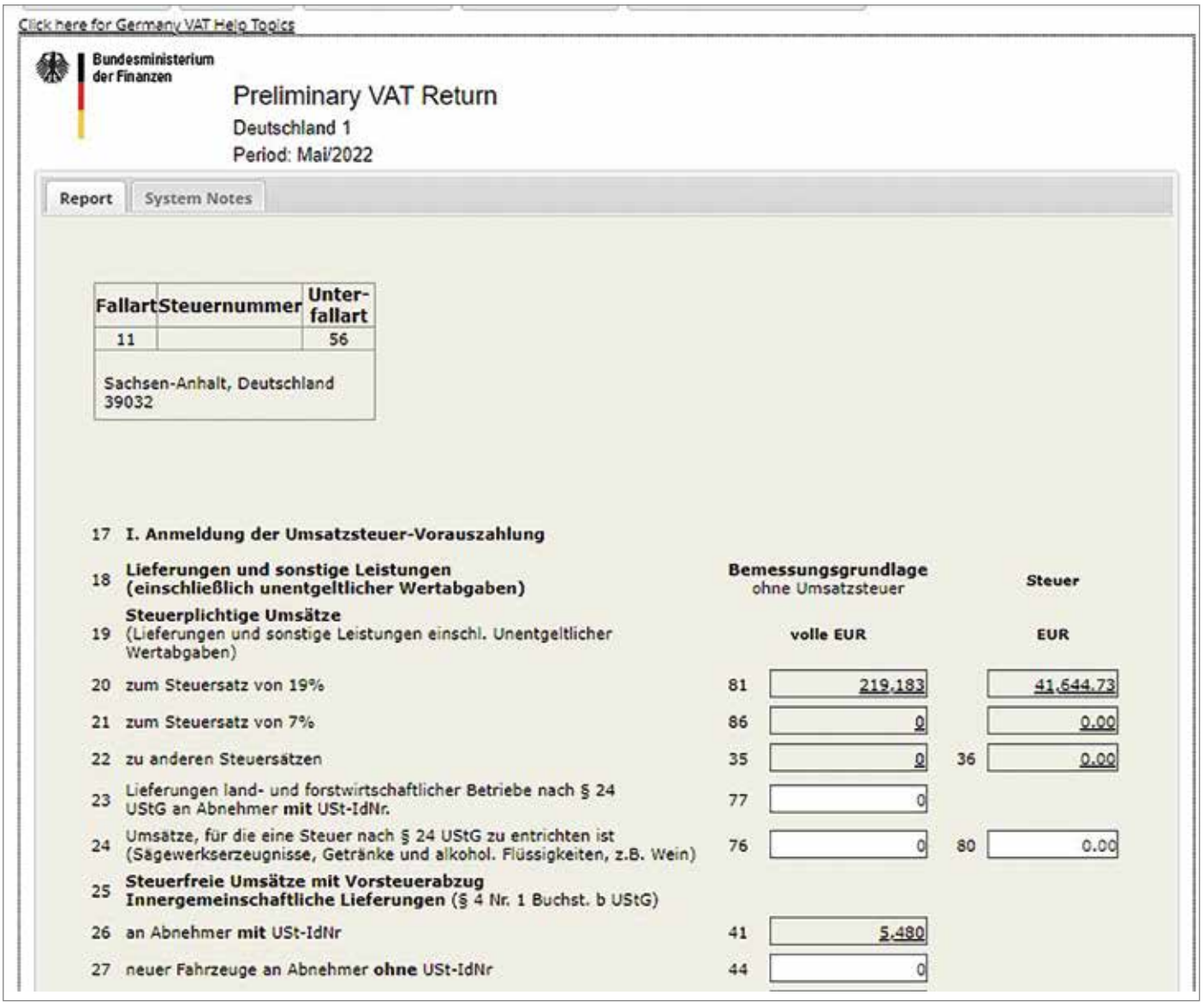


Abbildung 4: Umsatzsteuervoranmeldung im NetSuite-Standard (© PROMATIS)

Business Transactions (enter once)	Accounting Rules	Books of Record
Accounts Receivable	GAAP	US Based
Accounts Payable	IFRS	International
Invoices	Taxes	Tax
Sales Orders	Local	Fixed Assets
Journal Entries	Statutory	Country
Purchase Orders		Territory
Depreciation		Community
Amortization		Joint Venture

Abbildung 5: Multibook (via https://www.netsuite.com/portal/assets/pdf/ds-netsuite-multi-book-accounting.pdf)

facher ist und es sich lediglich um die „Übersetzung" des Kontenplans handelt, bietet NetSuite auch die Funktionalität des Buchhaltungskontexts (Accounting Context) an. Dieser kann an jedem Konto hinterlegt werden. So können die Konten bei bestimmten Usern oder Berichten mit einer anderen Nummer und einem anderen Namen angezeigt werden. Hierbei ist jedoch zu beachten, dass ein Accounting Context nur einmalig vergeben werden darf. So muss für jedes Konto ein einzigartiger Kontext erstellt werden. Die Definition an den Kontenstammdaten ist in *Abbildung 8* ersichtlich. Es ist möglich, sowohl den Namen als auch die Nummer abweichend zu definieren.

Auch Mischformen sind durchaus eine pragmatische Lösung. Es ist nicht auszuschließen, dass weitere Konten für die deutsche Niederlassung angelegt werden müssen. Diese können in der Sichtbarkeit eingeschränkt werden, sodass kein globaler Impact entsteht.

Reporting – SuSa, G&V, Bilanz bis hin zur eBilanz

Wenn man das System „davon überzeugt" hat, korrekt zu buchen und die Steuerschlüssel richtig einzusetzen, sowie die Buchungen genehmigt sind, ist das Thema Reporting an der Reihe. Wir unterscheiden ganz klassisch zwischen Nebenbuch- und

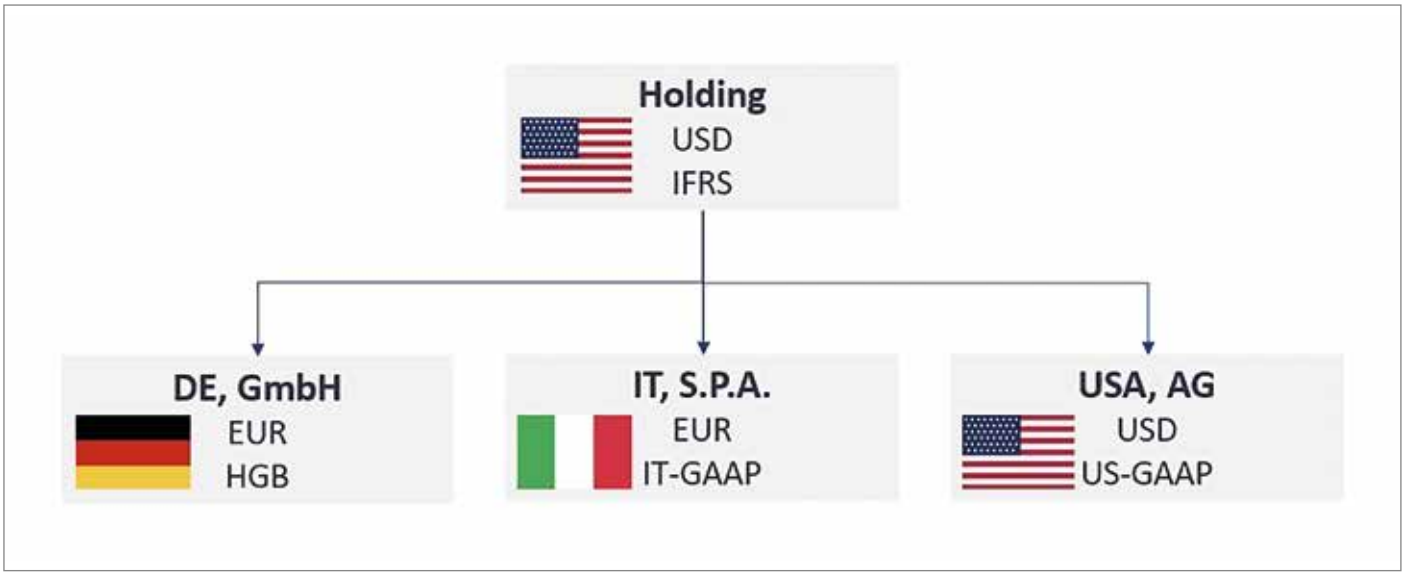


Abbildung 6: Beispiel internationale Struktur (© PROMATIS)

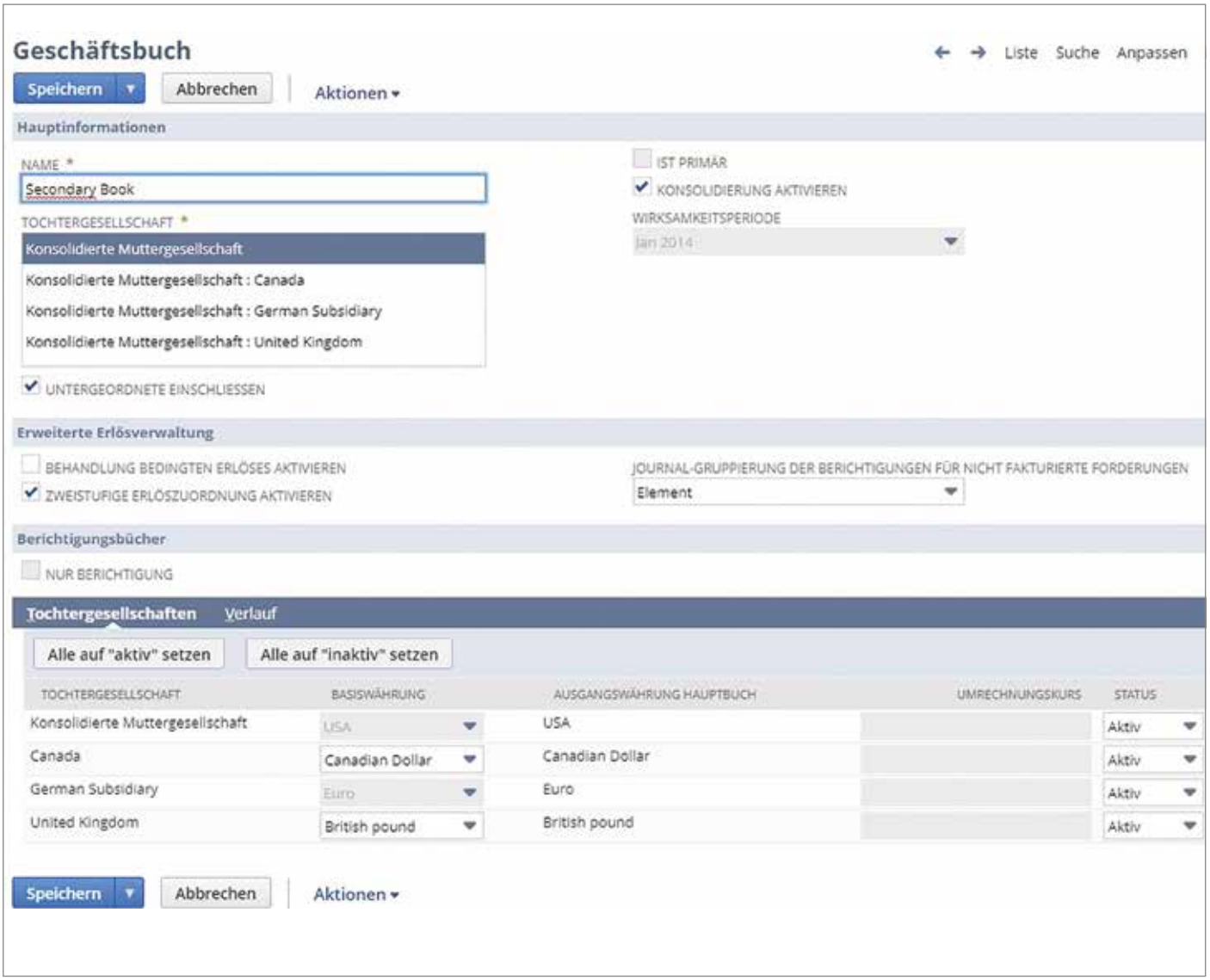


Abbildung 7: Sekundäres Geschäftsbuch (© PROMATIS)

Hauptbuch-Reporting. Im Nebenbuch werten wir Transaktionen und Ereignisse aus. Diese liefern dann Buchungen auf den Hauptbuchkonten, die durch Nebenbuchdetails verprobt werden können. So wird zum Beispiel ein Forderungskonto mit einem Aging-Bericht abgestimmt oder ein Anlagenbestandskonto mit einem Anlagenspiegel. Damit können pro Bilanzposition die passenden Nebenbuchdetails abgerufen werden, um die Konten im Monatsabschluss abzugleichen. Eine gute Basis hierfür ist eine Summen- und Saldenliste (SuSa). Wenn die Kontensalden nachvollziehbar sind, kann die Abstimmung der Gewinn- und Verlustrechnung – kurz G&V – stattfinden. Hier werden beispielsweise Aufwendungen und Erträge auf ihre periodengerechte Zuordnung geprüft. Wenn das Betriebsergebnis vorliegt, wird dieses mit in die Bilanz einbezogen. Spätestens hier sollten dann, bei korrekter Erstellung der Berichte, Aktiva mit Passiva übereinstimmen.

Diese Berichte lassen sich in NetSuite auf verschiedene Art leicht darstellen: durch Anpassung von Standardberichten, Nutzung von Lokalisierungs-Bundles oder Reporting-Paketen von Partnern beziehungsweise anderen Anbietern. Teilweise ist eine Anpassung der Zuordnungen von Konten zu Positionen im Report notwendig. Das gilt insbesondere, wenn kein standardisierter Kontenrahmen wie SKR04 oder SKR03 verwendet wird. Den Königsweg stellt häufig die Erstellung einer eBilanz dar, die momentan nicht auf dem direkten Weg aus NetSuite ableitbar ist. Dies hat verschiedene Gründe, zum Beispiel dass die eBilanz eine abweichende Struktur basierend auf der anzuwendenden Taxonomie hat und diese Zuordnung in NetSuite nur mit erhöhtem Aufwand wartbar ist. Viele Kunden entscheiden sich deshalb dafür, die eBilanz durch ihren Steuerberater erstellen zu lassen. Hierfür sowie für die Prüfungen werden Daten von NetSuite an die Steuerberatung übergeben, sodass anschließend die Meldung erfolgen kann.

Zusammenfassung

Compliance ist nicht einfach ein Bundle oder etwas, das man einstellt – am Ende sind es systemische Abbildungen und begleitende GoBD-konforme Geschäftsprozesse. In NetSuite hat man allerdings durch die einfache und transparente Konfiguration, die verfügbaren Zusatzpakete sowie die einfache Erweiterbarkeit eine sehr gute Grundlage. Man sollte sich auch immer bewusst sein, dass ein „deutsches Compliance-Projekt“ erhebliche Auswir-

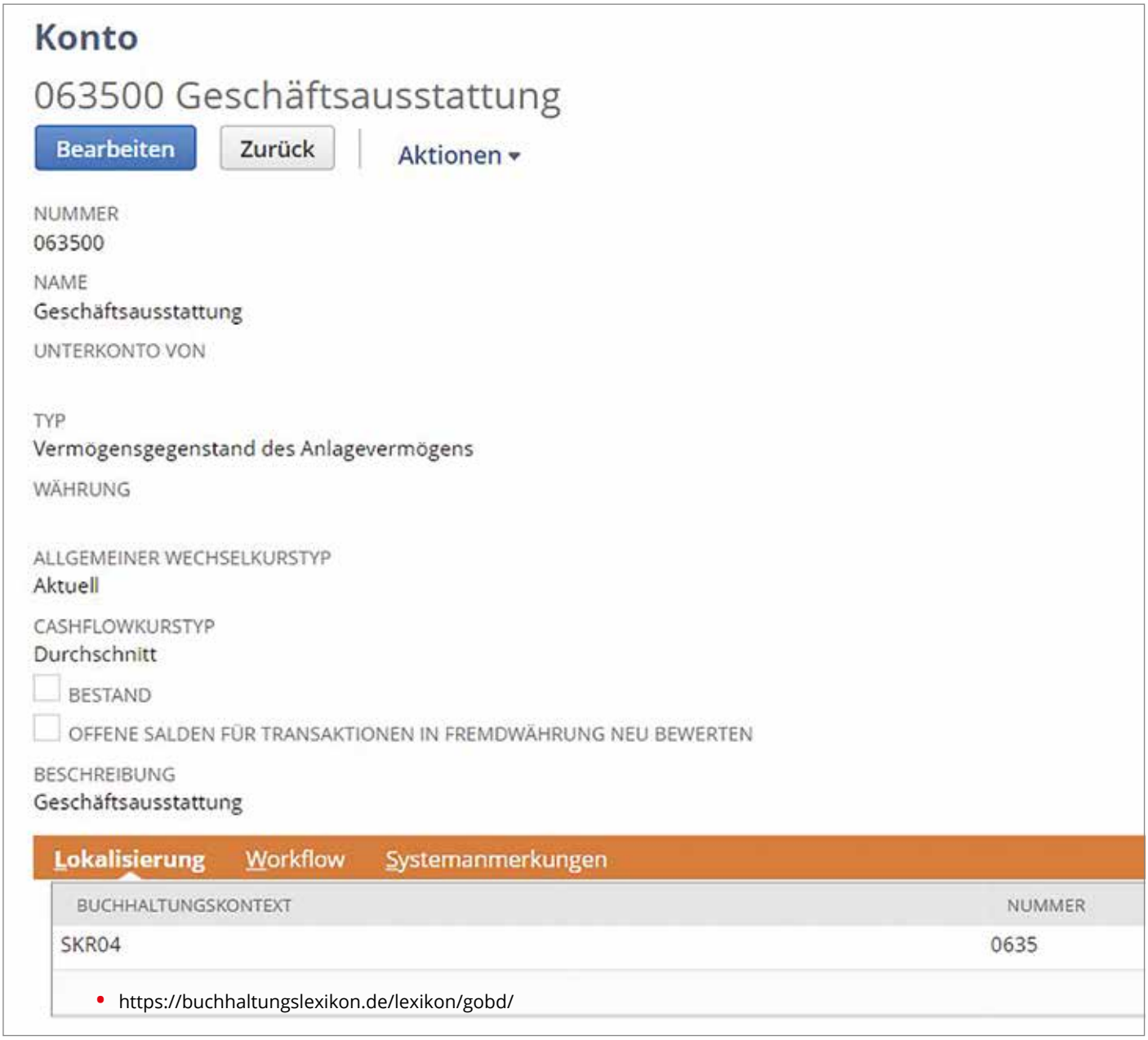


Abbildung 8: Accounting Context (© PROMATIS)

kungen hat – auf die Systemnutzung und internen Prozesse wie auch auf das globale Setup, denn dies betrifft einen erheblich größeren Kreis als „nur" die Anwender in Deutschland. Wir empfehlen, eine derart „lokale Initiative" im Unternehmen als ernstzunehmendes Projekt wahrzunehmen und auch globale Administratoren und Stakeholder mit einzubeziehen.

Referenzen

[1] Trademark Details (*https://trademarks.justia.com/876/54/act-global-be-87654908.html*, abgerufen am 03.06.2022)

[2] Grundsätze zur ordnungsmäßigen Führung und Aufbewahrung von Büchern, Aufzeichnungen und Unterlagen in elektronischer Form sowie zum Datenzugriff (*https://buchhaltungslexikon.de/lexikon/gobd/*, abgerufen am 03.06.2022)

Katharina Schraft
katharina.schraft@promatis.de

Katharina Schraft ist seit 2010 bei der PROMATIS software GmbH am Hauptsitz in Ettlingen (TechnologieRegion Karlsruhe) tätig und hat ihren Schwerpunkt in der ERP-Beratung mit Fokus auf Finance gelegt. Seit 2018 ist sie Vice President sowie Leiterin der Business Unit NetSuite und berät Kunden unter anderem in strategischen Buchhaltungs- und Compliance-Fragestellungen.

BEST OF DOAG ONLINE

Eine Auswahl der besten DOAG News Juli/August 2022

On-demand-Ticket – das neue Angebot der DOAG

Verhindert, erkrankt, eingespannt? Allen, denen eine große DOAG-Konferenz durch die Lappen gegangen ist, kann geholfen werden.

https://www.doag.org/de/home/news/on-demand-ticket-das-neue-angebot-der-doag/

Erfolgreiche Premiere der CloudLand 2022

Ein viertägiges Festival mit Cloud-Themen in einem Freizeitpark – unser Recap zur fantastischen CloudLand 2022.

https://www.doag.org/de/home/news/erfolgreiche-premiere-der-cloudland-2022/

DOAG Datenbank Kolumne: "Wie wichtig gute Kommunikation ist"

Jürgen Vitek erläutert an einem praktischen Beispiel den Wert von rechtzeitiger, guter und direkter Kommunikation.

https://www.doag.org/de/home/news/datenbank-kolumne-wie-wichtig-gute-kommunikation-ist/

DOAG-Studio-Interview: "Ronny trifft..." Scott Spendolini

Open Source Developer und Mitglied des Oracle APEX Development Teams Ronny Weiß spricht mit dem Covid-App-Entwickler Scott Spendolini.

https://www.doag.org/de/home/news/doag-studio-interview-ronny-trifft-scott-spendolini/

DOAG Datenbank Kolumne: "Generation X, Y, Z – und was kommt dann?"

Axel vom Stein über Generationsbezeichnungen und was dies mit der Datenbank-Community zu tun hat.

https://www.doag.org/de/home/news/datenbank-kolumne-generation-x-y-z-und-was-kommt-dann/

DOAG DB WebSession: "New Features in Multitenant mit 21c"

In der 60-minütigen WebSession vom 8. Juli 2022 erfahren Sie alles rund um das Thema "New Features in Multitenant mit 21c".

https://www.doag.org/de/home/news/doag-db-websession-new-features-in-multitenant-mit-21c/

Devs on Tape: Florian Heubeck über GitOps, MediaMarktSaturn und Deployments an Freitagen

In dieser Episode des Podcasts war Florian Heubeck zu Gast bei Kai Donato und Carolin Hagemann im Studio auf der CloudLand 2022.

https://www.doag.org/de/home/news/devs-on-tape-florian-heubeck-ueber-gitops-mediamarktsaturn-und-deployments-an-freitagen/

Sicherheitslücke im Postgres-Core-Server

Die Postgres-Community hat eine neu entdeckte Sicherheitslücke im Postgres-Core-Server identifiziert und behoben.

https://www.doag.org/de/home/news/sicherheitsluecke-im-postgres-core-server/

Wir begrüßen unsere neuen Mitglieder

Natürliche Mitglieder:

- Jürgen Marquardt
- Noelani Thalbauer
- Roland Bon
- Florian Heubeck
- Kerstin Dolz
- Thorsten Bachmann
- Fabian Lehn
- Daniel Stollenwerk
- Kersten Penni
- Modjibullah Moheby
- Felix Specht

Korporative Mitglieder:

- Dirk Rossmann GmbH, Repräsentant: Christian Thieliant
- MTU Aero Engines AG, Repräsentant: Roland Hauk
- DZ BANK AG, Repräsentant: Jörg Kettlitz
- TDM Systems GmbH, Repräsentant: Uwe Schütze
- KZV Niedersachsen, Repräsentant Stefan Wiese

Termine

September 09

15.09.2022
DB-Programmierung
DevTalk mit Jürgen Sieben, Ulrike Schwinn, Sabine Heimsath und Christian Schwitalla
WebSession

19.09.2022
European NetSuite User Days
im Rahmen der DOAG 2022 Konferenz + Ausstellung in Nürnberg
Nürnberg Convention Center Ost

20.09.2022
DOAG 2022 Konferenz + Ausstellung
Zwei Tage Konferenzerlebnis + Thementag
Nürnberg Convention Center Ost

29.09.2022
Rückblick K&A
DevTalk mit Carolin Hagemann und Niels de Bruijn
Online

Oktober 10

11.10.2022
Einführung in Docker als Basis für Entwicklung und Produktion.
Expertenseminar mit Kai Donato und Philipp Hartenfeller
Berlin

13.10.2022
Flyway vs. Liquibase
DevTalk mit Sabine Heimsath und Oliver Lemm
WebSession

19.10.2022
LDAP-basierte Namesauflösung von Oracle Datenbanken
Online DAC WebSession

25.10.2022
Professionelle PL/SQL Datenbankprogrammierung
Expertenseminar mit Jürgen Sieben
Berlin

27.10.2022
Q&A mit dem APEX Team
DevTalk mit Carolin Hagemann, Carsten Czarski und Niels de Bruijn
Online

November 11

08.11.2022
Good APEX Practices for Enterprise Projects – incl. APEX 22.1
Expertenseminar mit Moritz Klein und Markus Dötsch
Berlin

17.11.2022
APEX & JavaScript
DevTalk mit Carolin Hagemann und Kai Donato
Online

22.11.2022
Oracle BI/Analytics Publisher als Reporting-Lösung für Forms- und APEX-Anwendungen
Expertenseminar mit Jürgen Menge
Berlin

29.11.2022
Oracle PL/SQL Performance Tuning
Expertenseminar mit Jürgen Sieben
Berlin

Impressum

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

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

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

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

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
ViSdP: Fried Saacke
Redaktionsleitung Red Stack Magazin:
Martin Meyer
Redaktionsleitung Business News:
Marcos López
Kontakt: redaktion@doag.org
Weitere Redakteure (in alphabetischer Reihenfolge): Clemens Bleile, Manfred Drozd, Markus Flechtner, Marco Friebe, Martin Kalkuhl, Markus Kißling, Jonas Matthes, Raphael Salguero Aragón, Dr. Frank Schönthaler, Katharina Schraft, Jürgen Sieben, Günther Stürner, Frank Weise, Ronny Weiß.

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

Fotonachweis:
Titel: © Storyset | *www.freepik.com*
S. 12: © AaronJOlson | *www.pixabay.com*
S. 14: © Pisauikan | *www.pixabay.com*
S. 18: © ID 27707 | *www.pixabay.com*
S. 24: © Atlantios | *www.pixabay.com*
S. 30: © ValdasMiskinis | *www.pixabay.com*
S. 36: © AJEL | *www.pixabay.com*
S. 42: © Pexels | *www.pixabay.com*
S. 52: © DesignDrawArtes | *www.pixabay.com*
Titel S. 54: © Upklyak | *www.freepik.com*
S. 70: © Geralt | *www.pixabay.com*
S. 72: @ SamEdwards | *www.istockphoto.com*
S. 81: © Pch Vector | *www.freepik.com*

Anzeigen:
sponsoring@doag.org

Mediadaten und Preise:
www.doag.org/go/mediadaten

Druck:
WIRmachenDRUCK GmbH,
www.wir-machen-druck.de

Inserentenverzeichnis