Nr. 4 | August 2015 | ISSN 2364-7191 | www.doag.org

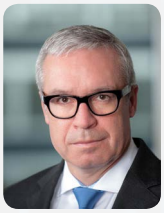
DOAG SOUG News

Enterprise-Architekturen

flexibel & leistungsstark

Microservices und SOA
Zwei Architektur-Ansätze im Vergleich

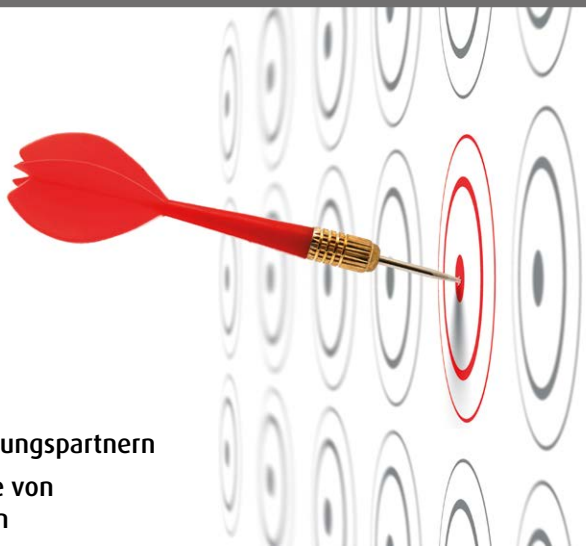
Im Interview
Hanspeter Kipfer, Vice President Technology Sales & Country Leader Oracle Schweiz



12c-New-Features
„SQL & PL/SQL"-Funktionalitäten

Berliner Expertenseminare

TERMINE 2015



- Wissensvertiefung für Oracle-Anwender
- Umfangreiches Seminarangebot
- Von Experten für Experten
- Mit ausgewählten Schulungspartnern
- Exklusive Gruppengröße von maximal 25 Teilnehmern

15.09.2015 - 16.09.2015	**Professionelle APEX-Entwicklung und APEX 5.0** *Oliver Lemm*
22.09.2015 – 23.09.2015	**Cost Based Optimizer & Trouble-shooting with Execution Plans** *Jonathan Lewis*
29.09.2015 - 30.09.2015	**OBIEE Repository und Reports Master Class** *Gerd Aiglstorfer*
06.10.2015 - 07.10.2015	**Oracle DB Hochverfügbarkeit** *Robert Bialek und Mathias Zarick*
13.10.2015 - 14.10.2015	**Beyond the Basics: PL/SQL-Objekte für jedermann** *Jürgen Sieben*

Robert Szilinski
Vorstand DOAG Development Community

Liebe Mitglieder, liebe Leserinnen und Leser,

ich habe in den letzten Monaten mit sehr vielen interessanten Menschen über die Zukunft der Software-Entwicklung diskutiert. Die immense Auswahl an verschiedensten Technologien, Frameworks und Methoden droht uns inzwischen zu überfordern und die Marketing-Abteilungen der großen Softwarehersteller und Dienstleister geben uns das Gefühl, dass wir gerade den nächsten Trend verschlafen.

Die Wahrheit liegt wahrscheinlich irgendwo in der Mitte. Tatsächlich habe ich allerdings derzeit den Eindruck, dass sich etwas nachhaltig verändert. Selbst in großen Konzernen werden gewachsene Strukturen hinterfragt, agile Methoden erhalten nahezu überall Einzug und schwergewichtige Produkt-Suiten werden vielerorts durch leichte Ansätze ersetzt. Dabei spielen heute insbesondere moderne Microservice-Architekturen eine immer größere Rolle. Ähnlich wie mit dem Entwurf auf der Titelseite lassen sich damit und mit webbasierten Technologien sehr schnell neue Geschäftsanforderungen umsetzen.

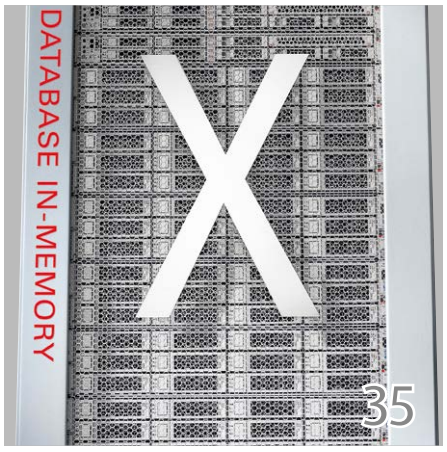
In der Development Community haben wir uns das Ziel gesetzt, diesen Aspekten zukünftig mehr Bedeutung beizumessen, uns hierfür weiter zu öffnen und die wichtigsten Fragestellungen für unsere Mitglieder aufzugreifen.

In diesem Sinne wünsche ich Ihnen viel Spaß beim Lesen dieser Ausgabe und einen intensiven Gedankenaustausch bei den kommenden Veranstaltungen der DOAG und der SOUG.

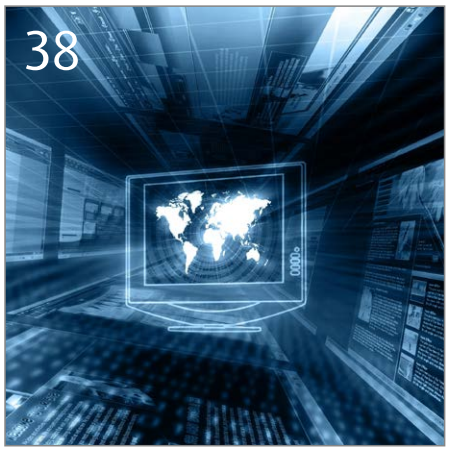
Ihr

R. Szilinski

Die Cloud gilt heute als Basis moderner Software-Entwicklung



Oracle Exadata im Vergleich mit den Angeboten anderer Hersteller



Grundlegende Maßnahmen beim Entwickeln von ETL-Prozessen

Einleitung

Enterprise-Architekturen

Engineered Systems

Data Warehouse

Entwicklung

Datenbank

DOAG/SOUG intern

DOAG/SOUG Timeline

28. April 2015

Michael Paege, stellvertretender Vorstandsvorsitzender der DOAG und Leiter Competence Center Lizenzierung, und Fried Saacke, DOAG-Vorstand und Geschäftsführer, sprechen mit Carsten J. Diercks, Rechtsanwalt der DOAG, über verschiedene Probleme bei der Oracle-Lizenzierung, da sich die Anfragen besorgter Anwender an die DOAG häufen.

6. Mai 2015

In Bern findet eine lokale SOUG-Veranstaltung „SOUG bi de Lüt" mit rund 35 Teilnehmern statt. Nach einer kurzen Begrüßung stellt Paolo Kreth den Gastgeber vor und leitet dann in den ersten Vortrag „Erfahrung mit der Oracle In-Memory-Option" über. Er zeigt das Vorgehen beim Wechsel von DB2 mit Nettezza auf Oracle und die In-Memory-Option. Der zweite Vortrag handelt vom „Performance Warehouse", vergleichbar mit dem von Oracle in Oracle Cloud Control 12.1.0.3 eingeführten AWR-Warehouse. Nach diesen sehr interessanten Referaten können die Teilnehmer bei einem Apéro noch fleißig ihr Netzwerk pflegen.

Eröffnung durch Paolo Kreth

6. Mai 2015

Das Organisationsteam der JavaLand 2016 trifft sich zum Kickoff-Meeting in der DOAG-Konferenzlounge in Berlin. Nach Auswertung des Feedbacks zur letztjährigen JavaLand werden die Konzepte für die JavaLand 2016 angepasst und erweitert, damit die Veranstaltung im nächsten Jahr noch erfolgreicher wird.

8. Mai 2015

Dr. Dietmar Neugebauer, Vorstandsvorsitzender der DOAG, begrüßt mehr als 70 DOAG-Aktive zum ersten Leitungskräfte-Forum. Er erläutert zu Beginn die Hintergründe, gewachsene Struktur und einzelnen Funktionen in der Organisationsstruktur des DOAG e.V. Wichtige Punkte auf der Agenda sind die Erfahrungen zur Satzungsreform sowie die themenorientierte Weiterentwicklung der Online-Services. Am Nachmittag steht die Arbeit der DOAG-Communities im Fokus. Jede Community präsentiert ihren aktuellen Status, die Zusammenarbeit mit anderen Communities, ihre Themen-Schwerpunkte für die DOAG-Medien, die Top-Events für das Jahr 2016 sowie die strategischen Ziele für die kommenden zwei Jahre. Die abschließende Feedback-Runde beschließt, das Leitungskräfte-Forum auch im kommenden Jahr im Vorfeld der Delegiertenversammlung abzuhalten.

9. Mai 2015

Die Delegiertenversammlung der DOAG tagt in Berlin und wählt ihren neuen Vorstand. Im Amt des Finanzvorstands und stellvertretenden Vorsitzenden gibt es mit der Wahl von Stefan Kinnen einen Wechsel. Dieser löst Urban Lankes ab, der nicht zur Wiederwahl steht. Die Leitung der beiden im letzten November neugegründeten Communities übernehmen Michael Klose (Business Intelligence Community) und Markus Eisele (Java Community). Auch bei der Infrastruktur und Middleware Community kommt es zu einem Wechsel der Leitung, Jan-Peter Timmermann löst Björn Bröhl ab, der ebenfalls nicht zur Wiederwahl steht. Dr. Dietmar Neugebauer wird erneut zum Vorstandsvorsitzenden gewählt. Ebenfalls in den Vorstand wiedergewählt werden Michael Paege als stellvertretender Vorsitzende mit dem Aufgabenbereich „Querschnittsgruppen", Christian Trieb als Leiter der Datenbank Community, Robert Szilinski als Leiter der Development Community und Dr. Frank Schönthaler als Leiter der Business Solutions Community. Fried Saacke, Geschäftsführer der DOAG-Geschäftsstelle, wird erneut als Vorstand mit dem Aufgabenbereich Geschäftsstelle kooptiert.

Ein intensiv diskutiertes Thema auf der Delegiertenversammlung ist die Gewinnung neuer Mitglieder und Referenten. Als eine Maßnahme der Mitglieder-Akquise sollen digitale Inhalte auf den Webseiten der DOAG zukünftig für jeden frei im Internet verfügbar sein. Die Delegierten versprechen sich davon die Generierung einer stärkeren Aufmerksamkeit im Internet, einen größeren Bekanntheitsgrad der DOAG und einen Anstieg der Mitgliedszahlen. Die neue, offenere Ausrichtung der DOAG wird unter dem Arbeitstitel „DOAG 2.0" von Robert Szilinski, Leiter der Development Community, vorgestellt. Besonderen Wert legen die Delegierten auch auf die Analyse und Verbesserung des Prozesses zur Gewinnung von Referenten für alle Veranstaltungen der DOAG. Hier wird eine Arbeitsgruppe mit fünf Delegierten eingerichtet, die zusammen mit je zwei Vertretern des Vorstands und der Geschäftsstelle den Prozess zur Referenten-Gewinnung analysieren und dem Vorstand bis Ende des Jahres Verbesserungsvorschläge präsentieren sollen.

Zudem stimmen die Delegierten einstimmig der Internationalisierungsstrategie des Vorstands zu. Insbesondere die Jahreskonferenz der DOAG soll internationaler werden. Auch eine noch intensivere Vernetzung mit den Usergroups aus den USA und aus Europa ist geplant.

Weitere Informationen unter „http://www.doag.org/home/aktuelle-news/article/delegiertenversammlung-waehlt-neuen-vorstand-und-beschliesst-doag-20.html"

Der neue DOAG-Vorstand: (von links nach rechts) Michael Klose, Fried Saacke, Christian Trieb, Markus Eisele, Stefan Kinnen, Dr. Dietmar Neugebauer, Jan-Peter Timmermann, Robert Szilinski, Michael Paege, Dr. Frank Schönthaler

19. Mai 2015

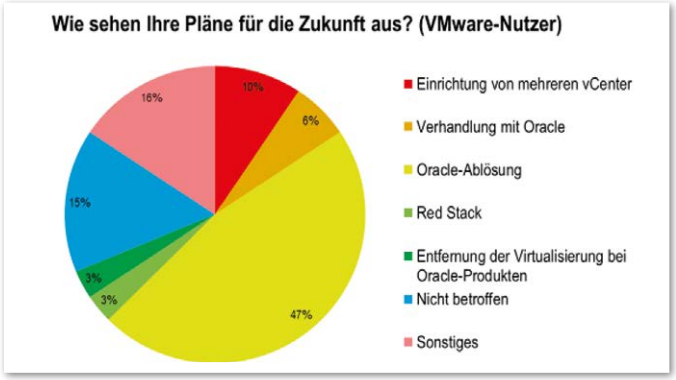
Peter Gübeli, Präsident der Swiss Oracle User Group (SOUG), stellt den 27. Jahresbericht des Präsidenten für das Jahr 2014 vor. Die SOUG hat viel initiiert, um seinen Mitgliedern in den kommenden Jahren einen modernen Service zu bieten und den Verein auch längerfristig attraktiv zu halten. Im Jahr 2014 werden acht Events durchgeführt: Drei Meetings der „Special Interest Groups" in der Deutschschweiz, zwei „SIGs" in der Romandie, ein „SOUG bi de Lüt", die „Real World Performance Tour" gemeinsam mit der DOAG und der AOUG in München und die „DOAG Jahreskonferenz" in Nürnberg. Aufgrund des beruflichen Wechsels zu Oracle kann Ansgar Wollnik seine Vorstandstätigkeit nicht mehr weiterführen. Zudem entscheidet sich Hansjörg Bütler nach vielen Jahren der Vorstandstätigkeit, seine Freizeit etwas abseits der Oracle-Themen zu verbringen. Glücklicherweise sind bereits kompetente und engagierte Nachfolger gefunden. Patrick Schäfer wird neu einen Event-Bereich für Architektur, Applikationen und Middleware aufbauen und Jürgen Vitek die Event-Koordination von Hansjörg Bütler übernehmen. Darüber hinaus ist geplant, den Vorstand künftig um eine Person zu erweitern.

Peter Gübeli stellt den Jahresbericht vor

20. Mai 2015

Die DOAG veröffentlicht ihre Umfrage zum Einsatz von Server-Virtualisierung. Demnach ist die Unzufriedenheit bezüglich der aktuellen Lizenzbedingungen hoch. Aus Sorge um Investitionssicherheit planen fast 35 Prozent der Nutzer von Virtualisierungs-Lösungen, sich von Oracle abzuwenden. Den Ernst der Lage verdeutlichen auch die mehr als 600 Teilnehmer, die sich an der anonymen Online-Umfrage beteiligt haben. Oracle verzichtet dennoch bisher auf eine offizielle Stellungnahme. Das Ergebnis der Umfrage zeigt sehr deutlich, dass viele Anwender die Lizenzierungs-Regelungen für Virtualisierungs-Umgebungen nicht mehr akzeptieren und sich der drohenden teuren Nachlizenzierung für Oracle-Datenbanken entziehen möchten. Neben den 35 Prozent der befragten Nutzer von Virtualisierungs-Lösungen, die Oracle ganz ablösen möchten, planen weitere 24 Prozent, dem Problem mit einem Workaround zu begegnen. Von den aktuellen Lizenzbedingungen „nicht betroffen" sehen sich lediglich 11 Prozent der Befragten.

Weitere Informationen unter „http://www.doag.org/home/aktuelle-news/article/doag-umfrage-zum-einsatz-von-server-virtualisierung-oracles-lizenzpolitik-hat-dramatische-konsequenzen.html"



Die Oracle-Lizenzpolitik hat dramatische Konsequenzen

21. Mai 2015

In Prangins am Ufer des Genfer Sees treffen sich die SOUG-R-Mitglieder zu einem spannenden Treffen mit Fokus auf der Oracle-Performance. Lucas Canali, CERN; Zbigniew Baranowski, CERN; Christian Antognini, Trivadis; Lothar Flatz, Trivadis; und Franck Pachot, dbi services führen mit ihre Vorträgen durch das interessante Thema. Das Hotel bietet mit Buffet und Service eine tolle Plattform für den weiteren Austausch.

Performance, wohin man schaut

29. Mai 2015

Fried Saacke, DOAG-Vorstand und Geschäftsführer, telefoniert mit Peter Gübeli, Präsident der Swiss Oracle User Group (SOUG). Es geht um die enge Zusammenarbeit der beiden Usergroups bei den operativen Aufgaben.

2./3. Juni 2015

Dr. Dietmar Neugebauer, Vorstandsvorsitzender der DOAG, Christian Trieb, DOAG-Vorstand und Leiter der Datenbank Community, und Ralf Kölling, Repräsentant der Regionalgruppe Bremen, vertreten die DOAG auf dem von Oracle veranstalteten EMEA-Usergroup-Treffen in Lissabon. Fried Saacke, DOAG-Vorstand und Geschäftsführer, komplettiert die DOAG-Delegation als Vertreter

des Interessenverbunds der Java User Groups e.V. (iJUG). Das wichtigste Thema seitens der DOAG ist die Verstärkung der internationalen Zusammenarbeit. Die konkrete Initiative, Vertreter der europäischen Usergroups am Vorabend der DOAG 2015 Konferenz + Ausstellung nach Nürnberg einzuladen, stößt auf sehr große Resonanz. Darüber hinaus wird ein Prozess festgelegt, um alle zwei Jahre einen Sprecher der EMEA-Usergroups zu wählen. Die DOAG und die UK Oracle User Group (UK OUG) bereiten die Wahl vor.

3./5. Juni 2015

Das Team der DOAG Dienstleistungen GmbH trifft sich zum jährlichen Workshop. Hauptthema ist diesmal ein Kommunikationstraining zur Klärung von Aufträgen und Konfliktgesprächen. In praktischen Übungen wird gelernt, Gespräche zu strukturieren, Bedarfe herauszufinden und mit Reklamationen besser umgehen zu können.

9./10. Juni 2015

Rund 170 Oracle-Spezialisten nehmen in Darmstadt an der diesjährigen Jahresveranstaltung der Business Solutions Community der DOAG teil und werden ermutigt, getreu Darwins Worten „the Survival of the Fittest" ihre Anpassungsfähigkeit und somit Wettbewerbsfähigkeit unter Beweis zu stellen. Sowohl Dirk Schmachtenberg, Geschäftsführer der Trevisto, als auch Ravi Pandit, Gründer und Firmenchef von KPIT Technologies, beide Keynote-Speaker, zeigen das rasche Vordrängen der sogenannten „vierten Revolution" anhand von Beispielen aus dem Alltag. Ob on-premise oder in der Cloud, auch die Oracle-Unternehmensapplikationen kommen während der zweitägigen Konferenz unter die Lupe. Während der Mittagspausen in der Ausstellung und auch am ersten Abend haben die Teilnehmer viele Gelegenheiten, in einer lockeren Atmosphäre zu netzwerken. Im Rahmen der Veranstaltung zeichnet die DOAG den E-Business-Spezialisten Sven Naegels, Applications Lead bei der Magna BDW Technologies Soest GmbH, für sein hohes Engagement innerhalb der Organisation als Botschafter in der Kategorie „Applications" aus.

Weitere Informationen unter „http://www.doag.org/home/aktuelle-news/article/doag-2015-bsk-der-digitalen-transformation-einen-schritt-naeher-kommen.html"

9./10. Juni 2015

Bei der ersten Apex-Konferenz der DOAG treffen sich rund 250 Begeisterte in Düsseldorf, um das Erscheinen der Version 5.0 zu feiern und viele spannende Vorträge zu erleben. Das Interesse und die Begeisterung für Apex sind riesig, die Veranstaltung ist bereits einige Zeit im Vorfeld vollständig ausgebucht. „Streckt doch mal alle eure Hände in die Luft", animiert Niels de Bruijn, Mitglied der Konferenzleitung, zu Beginn seiner Begrüßungsrede. Gleich darauf zückt er sein Smartphone, um den Moment bildlich festzuhalten. „Einfach genial diese #orclapex Community", twittert er im Anschluss und beschwört so das starke Community-Gefühl herauf, das die Teilnehmer sicher auch nach diesen zwei Tagen weiterhin begleiten wird. Für viele Lacher und gute Unterhaltung sorgt der Physik-Professor Metin Tolan mit seiner Keynote „Geschüttelt, nicht gerührt – James Bond im Visier der Physik". Er untersucht zahlreiche Filmszenen des Agentenklassikers auf ihren physikalischen Wahrheitsgehalt und beweist damit, dass Physik auch Spaß machen kann.

Weitere Informationen unter „http://www.doag.org/home/aktuelle-news/article/apex-connect-2015-grosser-ansturm-und-starkes-community-gefuehl.html"

Die Apex Community beim Aufwärmen

11. Juni 2015

Im Segelhof Baden findet die zweite SIG-Veranstaltung der SOUG im Jahr 2015 statt. Mehr als 40 Teilnehmer aus verschiedenen Bereichen rund um Oracle besuchen die zwei Streams „Oracle hochverfügbar" sowie „Oracle Security und weitere 12c-Features". Die insgesamt acht Vorträge sind ausgewogen und gut besucht. Es fällt auf, dass Live-Demonstrationen immer mehr zum Trend werden und die Sessions an Interaktivität gewinnen. So wird zum Beispiel Oracle SnapShot auf einem Laptop mit Oracle Linux, Virtual Box und ZFS Appliance Simulator gezeigt. Die Pause und der anschließende Apéro ermöglichen ein erstes Setzen der Materie sowie die Vertiefung der Themen untereinander und mit den Experten vor Ort.

12. Juni 2015

Die erste Vorstandssitzung des neu gewählten Vorstands stellt die letzten organisatorischen Weichen zur DOAG 2015 Konferenz + Ausstellung. Bezüglich der Veröffentlichung von Content ergeht der Beschluss, dass ab sofort Präsentationen und Manuskripte von Veranstaltungen im Web offen für alle sind. Der Zugriff auf Zeitschriften-Artikel ist bis acht Wochen nach Erscheinen nur für Mitglieder gestattet, ab der neunten Woche dann für alle Interessenten nach vorherigem Login. Auch die Teilnahme an den Live-Webinaren der DOAG soll zukünftig für Interessenten möglich sein.

16. Juni 2015

Die DOAG 2015 Datenbank in Düsseldorf ist erneut beliebter Treffpunkt zum Wissenstransfer für Datenbank-Administratoren und technisch Interessierte. Rund 200 Teilnehmer nutzen die Gelegenheit, um sich in den spannenden Erfahrungs- und Projektberichten über die aktuellen Trends im Datenbank-Umfeld zu informieren, rege zu diskutieren und neue Kontakte zu knüpfen. Aufgrund des großen Erfolgs soll die DOAG 2016 Datenbank erstmals zweitägig sein.

Weitere Informationen unter „http://www.doag.org/home/aktuelle-news/article/doag-2015-datenbank-beleuchtet-trends-und-chancen-im-datenbank-umfeld.html"

„Das traditionelle Geschäft von Oracle läuft in der Schweiz sehr gut …“

Der Schweizer Oracle-Markt hat seine Besonderheiten. Gaetano Bisaz, Mitglied des Vorstands der Swiss Oracle User Group (SOUG), und Wolfgang Taschner, Chefredakteur der DOAG/SOUG News, sprachen darüber mit Hanspeter Kipfer, Oracle Vice President Technology Sales & Country Lead Schweiz.

Hanspeter Kipfer (rechts) im Gespräch mit Gaetano Bisaz

Sie sind jetzt seit einem Jahr Country Leader von Oracle Schweiz. Was waren Ihre persönlichen Ziele in der neuen Funktion?
Kipfer: Meine persönlichen Ziele waren und sind nach wie vor, Oracle Schweiz weiterhin auf einem erfolgreichen Pfad zu führen. „Erfolgreich“ bedeutet für mich in erster Linie die positive Entwicklung von Kennzahlen wie Umsatz und Gewinn, aber auch die anerkennende Reflexion der Kunden über unser Unternehmen und unsere Produkte. Zudem kommt es mir darauf an, als Arbeitgeber erfolgreich zu sein, beispielsweise bei der Weiterentwicklung der Mitarbeiter und der Rekrutierung neuer Talente. Letztendlich definiere ich Erfolg auch an der guten Zusammenarbeit mit unseren Partnern.

Welche Ziele haben Sie bereits erreicht?
Kipfer: Mit dreißig Jahren Berufserfahrung, davon fünfundzwanzig Jahre im Vertrieb, ist es mir natürlich leicht gefallen, Prozesse im Unternehmen zu optimieren. Aufwändigere und mit größeren Auswirkungen verbundene Aufgaben, wie beispielsweise das Unternehmen als attraktiven Arbeitgeber zu präsentieren, sind natürlich nicht von heute auf morgen umzusetzen. Der Anspruch, in allen IT-Bereichen der Schweiz die Nummer eins oder zwei zu sein, ist uns weitgehend gelungen, wobei es auch hier immer noch ausreichend Potenzial zur Verbesserung gibt.

Wo steht Oracle Schweiz im Kontext mit den anderen europäischen Ländern?
Kipfer: Die Schweiz ist bei Oracle in der Nordic-Region eingebunden. Ich sehe es als meine Aufgabe, unser Land hier noch weiter nach vorn zu bringen.

Mit welchen Geschäftsfeldern sind Sie zufrieden, wo könnte es besser laufen?

Kipfer: Das traditionelle Geschäft von Oracle, also Datenbanken und Middleware, läuft in der Schweiz sehr gut. Mit den neuen Produkten, insbesondere der Cloud, haben wir unser Potenzial bei Weitem noch nicht ausgeschöpft. Das gilt auch für das Infrastruktur-Geschäft mit den Engineered Systems. Für die Bereiche „Platform as a Service", „Infrastructure as a Service" und „Software as a Service" sehe ich hohe Wachstums-Chancen.

Welche Bedeutung hat der Partner-Channel für Oracle in der Schweiz?

Kipfer: Der Partner-Channel ist fundamentaler Baustein unseres Goto-Market-Modells. Zum einen bieten wir gemeinsam mit den Partnern Komplett-Lösungen für die Kunden an und zum anderen sind unsere Partner in den Marktsegmenten aktiv, die wir selbst nicht bedienen.

Wie spüren Sie die aktuelle Situation des Schweizer Franken im Tagesgeschäft?

Kipfer: Unsere Referenz-Preisliste wird in US-Dollar geführt. Die Veränderungen des Schweizer Franken zum Dollar waren lange nicht so markant wie gegenüber dem Euro. Von daher kann man eher von einem Euro-Problem sprechen, das von der Schwäche des Euro gegenüber dem US-Dollar herrührt, was kurzfristig währungsbereinigt zu größeren Unterschieden zwischen der Euro- und der Schweizer-Franken-Preisliste geführt hat. Dies ist mittlerweile wieder korrigiert und an die aktuellen Wechselkurse angepasst worden. Den monatelangen Währungsvorteil aus der Stärke des US-Dollars haben wir jetzt einfach wieder abgegeben, sodass unter dem Strich alles ausgeglichen ist.

Welche speziellen Entwicklungen sehen Sie rund um die IT-Bedürfnisse der Schweizer Unternehmen?

Kipfer: Die Schweiz ist seit jeher sehr Technologie-affin. Ich erlebe das Land heute jedoch nicht mehr so innovativ wie früher. Unsere Unternehmen versuchen heute sehr stark, alle Risiken zu minimieren, insbesondere hinsichtlich Kosten-Reduzierung. In anderen europäischen Ländern liegt der Fokus hingegen mehr auf der Opportunität. Hier hat die Schweiz wieder Boden gutzumachen. Typische Beispiele sind Cloud oder Engineered Systems, die für mich eine evolutionäre Stufe der IT-Struktur darstellen. Von daher ist es für mich naheliegend, dass Unternehmen diese Technologien in Betracht ziehen. In vielen Gesprächen begegnet mir jedoch Skepsis bis hin zur Ablehnung gegenüber einer Veränderung. Wenn ich diese Haltung mit anderen europäischen Ländern vergleiche, sehe ich hier eine vergebene Chance für die Schweiz.

Die Einführung von Engineered Systems hat meist auch organisatorische Veränderungen im Unternehmen zur Folge. Wie reagieren Sie auf diesen Einwand?

Kipfer: Es gibt Whitepaper sowie eine von mir initiierte Presales-Taskforce, die unseren Kunden zur Verfügung stehen, um entsprechende Betriebsmodelle zu erarbeiten. Wenn wir es schaffen, die Hemmschwelle bei den Unternehmen zu überwinden, sind für deren Geschäfte auch wieder neue Innovationen möglich.

Sehen Sie diese Hemmschwelle eher im oberen Management oder auch an der Basis?

Kipfer: Nach meiner Einschätzung liegt es hauptsächlich am mittleren Management. Die oberen Führungskräfte haben die Vision und den Druck der Aktionäre, etwas umzusetzen, und die Basis ist durchaus an neuen Technologien interessiert.

Sie waren vor Ihrer Oracle-Zeit bei Siebel. Was haben Sie damals bei der Übernahme durch Oracle empfunden?

Kipfer: Meine ersten Gedanken waren absolut skeptisch, da Siebel durch die Übernahme Eigenständigkeit und Identität verlieren würde. Oracle hatte nach meiner Einschätzung keine Ahnung von Applikationen und ich wollte nur noch weg. Ich habe dann bei Oracle den jetzigen EMEA-Chef Loïc le Guisquet kennengelernt, der die Siebel-Integration geleitet hat. Das war eine sehr spannende Erfahrung für mich, da Loïc le Guisquet für mich die klare Absicht hatte, von Siebel zu lernen. Zudem wollte er Siebel zum Prototyp für das zukünftige Applikationsgeschäft von Oracle machen. Es hat mich sehr beeindruckt, wie er das konsequent durchgezogen hat. Während Loïc le Guisquet das Applikationsgeschäft für West-Europa leitete, habe ich es dann für Ost-Europa übernommen. Gemeinsam ist es uns gelungen, die heutige Basis für die Oracle Applications in Europa zu schaffen.

Wie stellen Sie sicher, dass lokale Anpassungen für Oracle Applications in dem kleinen Markt der Schweiz umgesetzt werden? Es gibt ja mit Deutsch, Französisch, Italienisch und Englisch gleich vier gebräuchliche Sprachen hier …

Kipfer: Zum Glück gibt es in Frankreich und Italien ja auch Oracle-Niederlassungen, die diese Sprachen abdecken und deren lokalisierte Produkte wir mit minimalen Änderungen übernehmen können. Das Anpassen lokaler legislativer Anforderungen hingegen ist durch unsere Partner abgedeckt. Ansonsten orientieren wir uns weitestgehend am internationalen Standard.

Was haben Sie aus Ihrer Zeit bei Sun in Erinnerung?

Kipfer: Ich war in den 1990er Jahren bei Sun. Damals war das Unternehmen stark im Wachsen und konnte große Erfolge verzeichnen. Zu Beginn der 2000er Jahre lief es dann nicht mehr so rund. Oracle hat durch die Sun-Übernahme sehr gute Mitarbeiter und mit SPARC, Solaris und Java auch sehr wertvolle Produkte erhalten. Dies war die Basis, um die heutigen Engineered Systems zu entwickeln. Von daher erlebe ich heute Sun innerhalb von Oracle wieder als Innovationsmotor. Durch die Weiterentwicklung im SPARC-Bereich werden wir bald Datenbanken auf Chip-Level betreiben können. Diese Innovationskraft hätte Sun alleine nicht aufbringen können.

Zunächst in Kooperation mit HP, später aus der Akquise von Sun sind die Exadata-Systeme entstanden. Wie läuft das Geschäft damit in der Schweiz?

Kipfer: Sehr namhafte Unternehmen wie Swiss Re setzen diese Systeme ein. Dennoch liegen wir im Vergleich zu den innovativsten Ländern in Europa noch um einige Zeit zurück. Von daher freue ich mich auf die kommenden Jahre. Die Basis ist geschaffen.

Neben Exadata und ODA gibt es ja noch andere Oracle Engineered Systems. Wie reagiert der Schweizer Markt auf diese?
Kipfer: Exadata ist hierzulande eindeutig am meisten im Einsatz. Für die Exalogic ist bei uns die installierte Basis im Applications-Bereich nicht so hoch wie in anderen Ländern, was die Verbreitung stark einschränkt. Erfolgreich sind wir hingegen mit Exalytics und der Database Appliance.

Manche Unternehmen möchten sich nicht von der Hardware bis zur Software in die Hand eines einzigen Anbieters wie Oracle begeben. Wie reagieren Sie darauf?
Kipfer: In erster Linie nehme ich diesen Einwand ernst. Es ist für mich aber auch eine Chance, dem Kunden aufzuzeigen, wie sich Oracle von anderen Anbietern unterscheidet. Oracle hat bei der Verbindung zwischen den einzelnen Layern immer auf offene Standards gesetzt, sodass der Kunde immer in der Lage ist, auch Produkte anderer Hersteller einzusetzen, sofern sie auf diesen offenen Standards aufsetzen. Unser Ziel war es nie, ein sogenanntes „Vendor Lock-in" zu betreiben.

Oracle fokussiert jetzt stärker auf die Cloud. Hat das Veränderungen in der Organisation von Oracle Schweiz zur Folge?
Kipfer: Im Applikationsgeschäft hat eine vollständige Transformation von „On Premise" zu „Software as a Service" stattgefunden. Dort ist heute auch das Neugeschäft angesiedelt. Im Technologie-Bereich hingegen gibt es diesen Wechsel nicht, dort kommen die Cloud-Aktivitäten inkrementell hinzu. Das bedeutet steigende Anforderungen für unseren Vertrieb, diese weitere Technologie zu vermarkten. Deshalb ist es mir auch sehr wichtig, neue Mitarbeiter zu finden, die quasi in der Cloud-Generation groß geworden sind, um unsere bewährten Kräfte zu ergänzen.

Hat beziehungsweise plant Oracle den Betrieb von Clouds in der Schweiz?
Kipfer: Vorstellen könnte man sich das schon, aktuell geplant ist so etwas nicht. Für unsere Unternehmen reicht es aus, wenn die Daten-Richtlinien innerhalb der EU erfüllt werden. Insofern ist es nicht erforderlich, die Daten ausschließlich direkt in der Schweiz zu halten. Unternehmen, die sehr sensible Daten besitzen, wie beispielsweise unsere Banken, würden ihre Daten sowieso nie nach draußen geben.

In der Presse ist von Entlassungen im Oracle-Support in Europa zu lesen. Inwiefern ist die Schweizer Niederlassung davon betroffen?
Kipfer: Für uns steht die Sicherung der Qualität des Supports im Vordergrund. Deshalb fahren wir seit geraumer Zeit das Servicekonzept „follow the sun", und dies rund um die Uhr. Bukarest ist für uns zur Sicherstellung dieses Ansatzes sehr wichtig geworden, dort beschäftigen wir bereits mehr als 1.500 Mitarbeitende. Wie die genaue Verteilung der Mitarbeiter auf unsere diversen Standorte sein wird, ist allerdings kein Punkt, den wir in der Öffentlichkeit debattieren.

Wo sehen Sie für Oracle Schweiz die Vorteile und wo die Nachteile einer starken Anwendervertretung wie der SOUG?
Kipfer: Ich sehe hier nur Vorteile und wünsche mir für die Zukunft eine noch engere Zusammenarbeit. Die Usergroup bietet mir die Möglichkeit, die Bedürfnisse unserer Kunden besser zu verstehen.

Die SOUG organisiert gemeinsam mit der DOAG Deutsche ORACLE-Anwendergruppe e.V. die DOAG 2015 Konferenz + Ausstellung vom 17. bis 20. November 2015 in Nürnberg. Am ersten Tag findet traditionell der Schweizer Abend zum Kennenlernen der Teilnehmer aus der Schweiz statt, zu dem wir Sie hiermit herzlich einladen. Werden Sie kommen?
Kipfer: Wenn es mir irgendwie möglich sein wird, werde ich natürlich teilnehmen.

Zur Person: Hanspeter Kipfer
Hanspeter Kipfer ist Vice President Technology Sales & Country Leader Oracle Schweiz. In dieser Rolle ist er verantwortlich für den gesamten Technologie-Produkte-Vertrieb sowie das abteilungsübergreifende Wachstum von Oracle in der Schweiz. Hanspeter Kipfer kam im April 2006 nach der Übernahme von Siebel Systems zu Oracle. Als General Manager bei Siebel Systems war er für den gesamten Vertrieb und die Service Operation für die Schweiz, Österreich, Osteuropa und die GUS-Staaten zuständig.

Vor seiner Zeit bei Siebel war Hanspeter Kipfer zwei Jahre in einem deutschen Start-up-Unternehmen in verschiedenen Führungsfunktionen tätig, unter anderem als COO und Executive Vice President Global Sales. Zuvor hatte er verschiedene Management-Positionen bei Sun inne. Während seiner letzten zweieinhalb Jahren im Unternehmen führte Kipfer das weltweite Geschäft, das durch die multilateralen Entwicklungsbanken wie die Weltbank, die Interamerikanische Entwicklungsbank (IDB) sowie die Europäische Bank für Wiederaufbau und Entwicklung (EBRD) von Washington D.C. generiert und mitfinanziert wurde.

Hanspeter Kipfer hat einen Abschluss in Telekommunikation-Wissenschaft, Wirtschaftsingenieurwesen sowie Business Administration und verfügt über mehr als fünfundzwanzig Jahre Erfahrung in der Technologie-Branche.

Microservices – Architekturmuster oder nur alter Wein in neuen Schläuchen?

Dr. Thomas Schuster und Carsten Wiesbaum, esentri AG

Microservices sind derzeit stark gefragt, doch was steckt eigentlich dahinter und wann lohnt sich der Einsatz von Microservices? Der Artikel zeigt, inwieweit Microservices von anderen Architektur-Stilen wie SOA abgegrenzt werden können sowie welche Chancen und Herausforderungen sich hieraus ergeben. Darüber hinaus geht es darum, ob sich Microservice-Konzepte auch unter Verwendung gängiger Werkzeuge wie der Oracle SOA Suite umsetzen lassen.

Wenn ein System neu entwickelt oder konzipiert werden soll, beginnt ein Entwicklungsteam in der Regel damit, Anforderungen zu identifizieren und sie auf eine geeignete Umgebung abzubilden – dies beinhaltet Server-Technologie, Frameworks und Programmiersprachen. Bereits die Tatsache, dass das Team diese allgemeinen Festlegungen trifft, impliziert oftmals eine schwergewichtige Makro-Architektur. Zu Beginn ist dies zumeist unkritisch – die entstehende Anwendung ist gut nachvollziehbar und anpassbar –, im Verlauf der Entwicklung ändert sich das mit zunehmender Anwendungsgröße. Aus einer monolithisch geprägten Makro-Architektur entsteht eine ebenso schwergewichtige Mikro-Architektur. Dies wirkt sich negativ auf die Weiterentwicklung des Gesamtsystems aus, unter anderem davon betroffen sind:

- *Wartbarkeit*
  Wird im Verlauf des Betriebs immer schwieriger
- *Skalierbarkeit*
  Wird eingeschränkt, schlimmstenfalls besteht diese lediglich aus der Replikation von Ressourcen und deren Load Balancing
- *Weiterentwicklung*
  rfordert einen hohen Aufwand (Abstimmung und Koordination zwischen den einzelnen Teams)
- *Deployment*
  Wird erschwert, insbesondere die unterbrechungsfreie Bereitstellung wird problematisch

Langfristig gilt, dass sich derartige Probleme verstärken, wenn die Anwendung wächst und neue oder geänderte funktionale sowie qualitative Anforderungen hinzukommen. Demgegenüber versprechen Microservices eine erhöhte Flexibilität und eine verbesserte Skalierbarkeit.

Microservices

Der Begriff „Microservice“ hat seit einiger Zeit eine Menge Aufmerksamkeit gewonnen *[1, 2]*. Der Microservice-Ansatz kann als ein Architektur-Stil für den Entwurf von verteilten Software-Systemen betrachtet werden. Kurz gesagt, sind Microservices ein Ansatz für die Implementierung eines Systems durch eine größere Menge von kleinen Diensten (Services). Jeder Dienst wird dabei unabhängig ausgeführt (eigener Prozessraum), verwendet seine eigenen Daten (Datenbank) und bietet leichtgewichtige Kommunikationsmechanismen gegenüber anderen Diensten (oft über HTTP oder HTTPS).

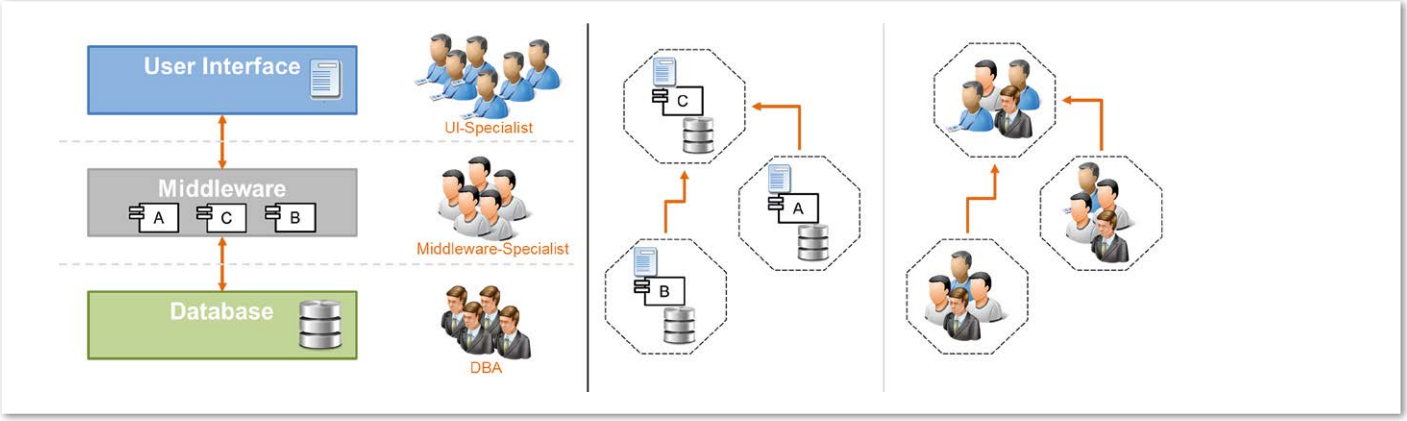


Abbildung 1: System-Entwicklung und Team-Organisation (ohne und mit Microservices)

Die Dienste beinhalten ausschließlich Funktionalitäten, die rund um eine Geschäftsfunktion (Business Capability) gruppiert werden können. Sie haben somit einen fachlich stark eingeschränkten Fokus, woraus in der Regel direkt die Grundprinzipien serviceorientierter Architekturen umgesetzt werden, insbesondere werden lose Kopplung, starke Kohäsion und die Trennung unterschiedlicher Belange (Separation of Concerns) erreicht.

Die Umsetzung von großen Systemen erfolgt in der Regel durch spezialisierte Teams. Das bedeutet zumeist, dass diese den Schichten der Software-Architektur entsprechend gebildet werden. Daher entstehen im Regelfall spezialisierte Teams je Anwendungsschicht (wie Benutzer-Interface, Middleware und Daten).

Die Organisation der Teams wird sich üblicherweise in der System-Architektur widerspiegeln *(siehe Abbildung 1)*; dieser Sachverhalt ist bereits aus Conways Gesetz bekannt: „Any organization that designs a system … will inevitably produce a design whose structure is a copy of the organization's communication structure." *[3]*

Sobald das System eine gewisse Größe überschritten hat, bedeuten Änderungen der Geschäftslogik (Middleware-Schicht) meistens einen hohen Aufwand, da aufgrund der Organisationstruktur und selbst bei serviceorientierten Systemen in der Regel viele Abhängigkeiten zwischen den Diensten einer Schicht bestehen.

Für die Entwickler bedeutet dies, dass die Änderung eines einzelnen Dienstes dazu führen kann, dass schlimmstenfalls eine ganz Schicht neu übersetzt und ausgerollt werden muss. Es wird also ein größerer Build-Prozess angestoßen, bei dem viele Komponenten integriert und getestet werden, die von den ursprünglichen Änderungen gar nicht betroffen gewesen sind.

Im Gegensatz hierzu fokussieren Microservices eine einzige Geschäftsfunktion und nutzen zu deren Umsetzung eine breite technologische Basis (einschließlich der Benutzerschnittstelle, der Datenhaltung und der externen Kommunikation). Dies bedingt, dass auch die Team-Organisation anders erfolgt. Teams werden typischerweise funktionsübergreifend organisiert, die Team-Mitglieder entstammen also unterschiedlichen Aufgabenbereichen (von der Gestaltung der Benutzer-Schnittstelle über die Geschäftslogik und Datenbank bis hin zum Projektmanagement).

Im Vergleich zu dem zuvor genannten Ansatz werden Dienste daher tatsächlich lose gekoppelt und das Team kann seine Dienste erneut bereitstellen, wann immer eine Änderung erforderlich wird und ohne dass hierbei zahlreiche andere Dienste ebenfalls neu ausgerollt werden müssen. Natürlich ist dies auch nicht ohne weitere Kosten möglich. Das Team verwendet in der Regel Mechanismen zur vollautomatischen Bereitstellung von Änderungen.

Da eine Änderung eines Dienstes auch andere Dienste betreffen kann und eine stärkere Verteilung der Funktionalität vorliegt, hat jedes Team darüber hinaus die Verantwortung, den Betrieb seiner Dienste zu überwachen, und wird hierzu Monitoring-Werkzeuge einsetzen. Die Entwicklungsteams sind somit in diesem Fall auch für den Betrieb der eigenen Dienste verantwortlich (oft als „DevOps" bezeichnet *[4]*).

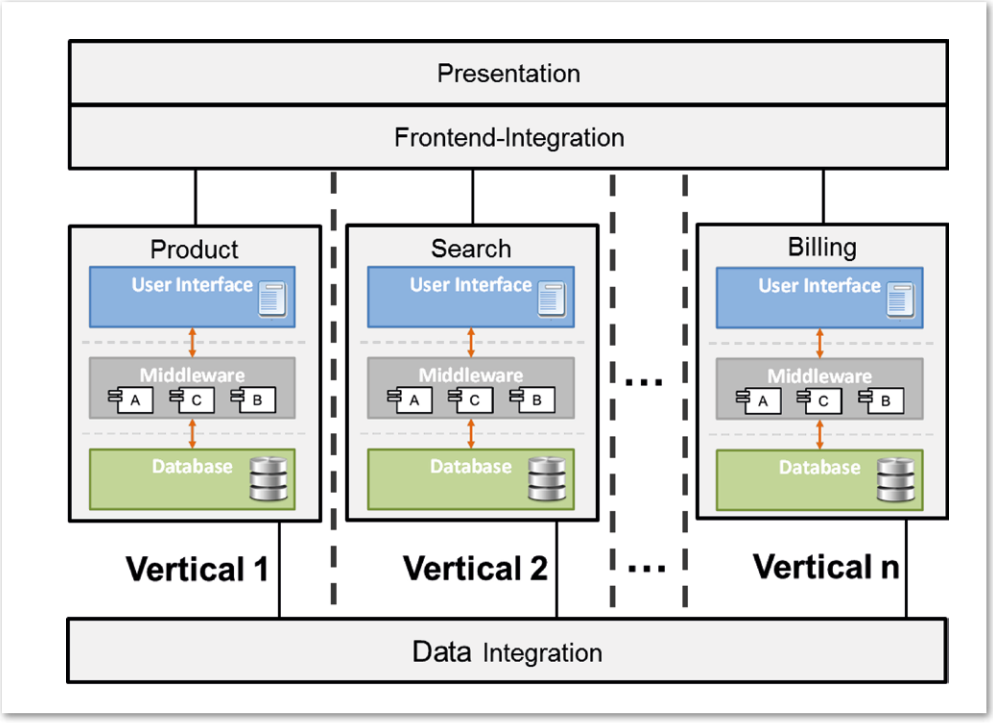


Abbildung 2: Vertikale Dekomposition

Grundprinzipien eines Microservice

Microservices fördern, neben einigen weiteren Dingen, somit die folgenden Grundsätze:

- Evolutionäres Design (Evolutionary Design)
- Strenge Kapselung (Shared Nothing)
- Intelligente Dienste und einfache Kommunikation (Smart Endpoints & Dumb Pipes)
- Dezentrale Governance
- Dezentrale Datenhaltung
- Automatisierung der Infrastruktur (Build-, Test- und Deployment-Prozesse)

Evolutionäres Design folgt praktisch unmittelbar aus der Tatsache, dass Änderungen jederzeit möglich sind und ein Team seine Dienste somit beliebig durch neue Dienste ersetzen kann. Die strenge Kapselung ist eine wichtige Voraussetzung, um evolutionäres Design zu ermöglichen. Erreicht wird diese Kapselung in der Regel durch das Konzept „Shared Nothing". Abhängigkeiten im Sinne gemeinsamer Quell-Codes und gemeinsamer Datenquellen werden hierbei vermieden; stattdessen werden externe Quellcode-Bibliotheken genutzt und jeder Dienst implementiert seine eigene Datenhaltung.

Im Gegensatz zur typischen SOA setzt der Microservice-Ansatz auf einfache Kommunikationsmechanismen nach dem Konzept „Smart Endpoints & Dumb Pipes".

SOA setzt oftmals einen Enterprise Service Bus (ESB) ein, um die Kommunikation zwischen den Services zu kontrollieren. Dieser bietet in der Regel anspruchsvolle Dienste wie für Nachrichten-Vermittlung

und -Transformation, zur Orchestrierung sowie zur Kontrolle von Geschäftsregeln.

Microservices verwenden das Architekturmuster „Pipes & Filters“ (Dienste im Stil von Unix). Daraus folgt, dass die intelligente Verarbeitung von Nachrichten innerhalb der Dienste („Smart Endpoint“) erfolgt, während die Kommunikation lediglich auf einfache Mechanismen („Dumb Pipe“) – etwa REST oder Messaging über eine leichtgewichtige, asynchrone Kommunikationsinfrastruktur wie RabbitMQ – setzt. Hierin liegt ein weiterer Grund für die einfache Änderung einzelner Dienste.

Herausforderungen

Eine der großen Herausforderungen in Bezug auf die Umsetzung von Microservice-Architekturen ist die Zerlegung des Systems, sodass in angemessener Weise zugeschnittene Dienste entstehen. Deshalb existieren hierzu verschiedene Strategien. Eine Möglichkeit, um eine Anwendung zu zerlegen, ist die sogenannte „vertikale Dekomposition“; das gesamte System wird hierbei vertikal in mehrere lose gekoppelte Anwendungen aufgeteilt. Jede dieser Vertikalen ist einem einzelnen Geschäftsbereich (etwa Bestellung, Zahlungsabwicklung oder Produkte) zugeordnet. Für jede Vertikale existiert eine eigene Präsentations-, Logik- und Datenhaltungsschicht *(siehe Abbildung 2)*.

Aus der Entwicklungsperspektive wird jede Vertikale durch ein funktionsübergreifend organisiertes Team (siehe oben) betreut und es besteht keine geteilte Code-Basis zwischen den Vertikalen (Shared Nothing). Grundsätzlich ist somit die lose Kopplung zwischen den Vertikalen gegeben. Wie die Teams die Entwicklung innerhalb einer Vertikalen organisieren, unterliegt hierdurch noch keiner Einschränkung. Insbesondere können die Teams den Technologie-Stack selbst wählen und selbstverständlich ihre Vertikale auch durch Microservices implementieren *(siehe Abbildung 3)*. Durch den klaren Fokus auf einen Geschäftsbereich fällt die Untergliederung in einzelne Microservices innerhalb einer Vertikalen jedoch deutlich leichter.

Um gegenüber dem Endanwender eine einheitliche Schnittstelle zu bieten, ruft man die Dienste der einzelnen Vertikalen dann in der Regel zunächst nach Bedarf durch einen Dienst auf (Frontend-Integration in *Abbildung 2)*. Anschließend werden die Anfrageergebnisse einheitlich gerendert. Da die Datenhaltung in den einzelnen Vertikalen ebenfalls unabhängig voneinander erfolgt, kann in jeder Vertikalen und bei Einsatz von Microservices für jeden Dienst die optimale Datenstruktur entworfen werden.

Sofern Daten aus einer anderen Vertikalen oder von einem anderen Microservice der gleichen Vertikalen genutzt werden müssen, können sie durch einen entsprechenden Service-Aufruf angefragt werden. Es ist ebenfalls denkbar, dass ein oder mehrere Dienste diesen Datenabgleich (Data Integration) ermöglichen. Die vertikale Dekomposition gepaart mit Microservices bietet somit eine gute Möglichkeit, Anwendungen zu entwerfen, die einem Höchstmaß an Flexibilität und Skalierbarkeit genügen.

Microservices und die Oracle SOA Suite

Bisher wurde im Artikel auf die Charakteristiken des Microservice-Architektur-Stils und die Abgrenzung zur klassischen SOA eingegangen. Zusammengefasst werden Microservices nach bestimmten Prinzipien entwickelt. Insbesondere die möglichst lose Kopplung und die isolierte Entwicklung der Microservices sind hervorzuheben. Jegliche Logik (Intelligenz) soll innerhalb der Microservices implementiert werden und allenfalls eine simple Integrationsschicht vorhanden sein, um die lose Kopplung nicht zu gefährden. Im klassischen SOA-Umfeld steht die Integration bestehender – zumeist komplexer, monolithischer und über die Zeit gewachsener – Geschäftsanwendungen (Alt-Systeme) im Vordergrund. Von deren Anpassung wird aufgrund ihrer Komplexität und zu erwartender Kosten eher abgesehen und stattdessen eine komplexe Integrationslogik genutzt.

Mit der Zeit sind so leistungsstarke SOA-Plattformen für die Entwicklung und die Integration von Diensten entstanden. Eine dieser Plattformen ist die Oracle SOA Suite. Deren zentrale Komponente ist ein Service-Component-Architecture-Projekt (SCA). Jedes SCA besitzt eine oder mehrere klar definierte

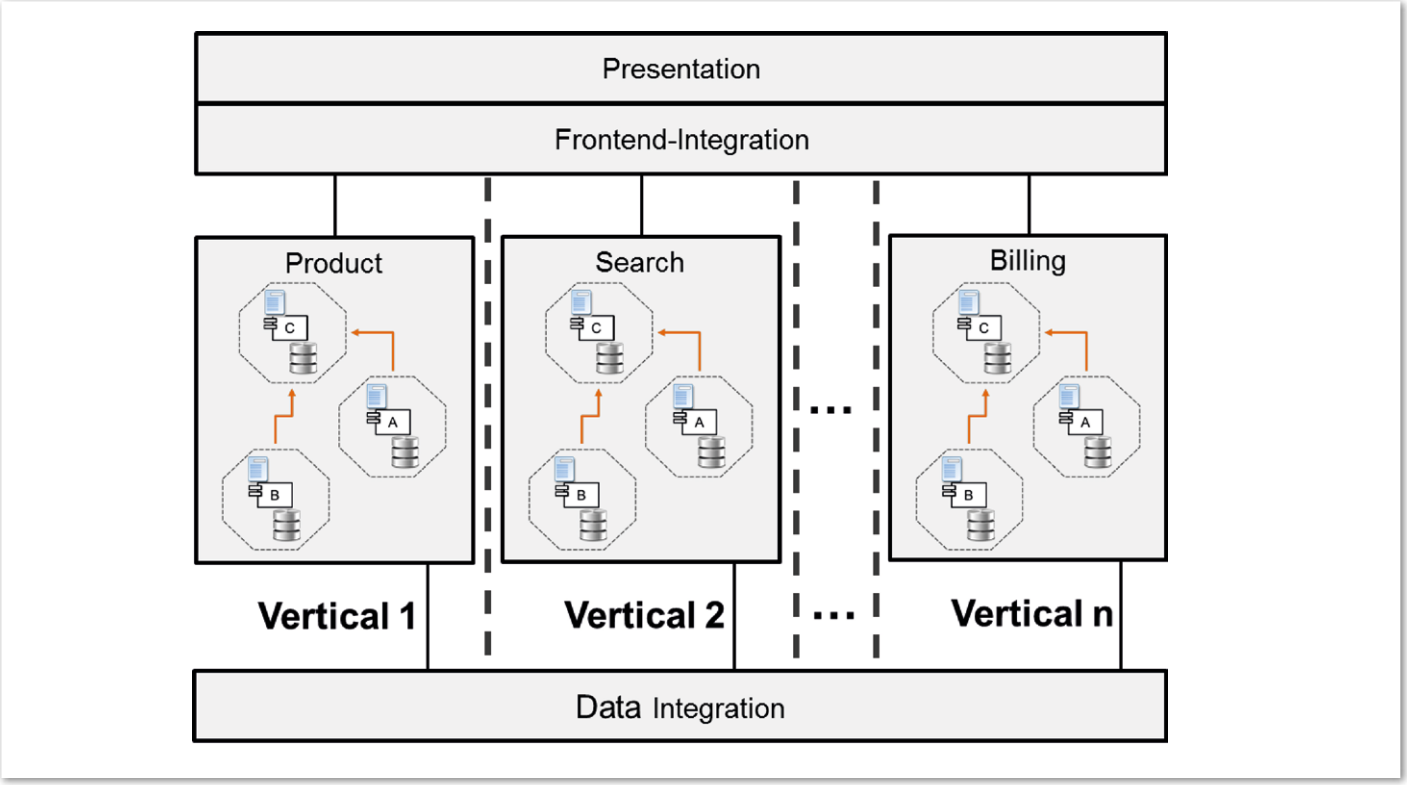


Abbildung 3: Vertikale Dekomposition mit Microservices

Abbildung 4: Konfiguration von Work-Manager-Gruppen pro Partition

Schnittstellen und wird separat auf der Oracle SOA Suite ausgeliefert.

Innerhalb eines SCAs lassen sich verschiedene Komponenten zur Implementation von Business- und Integrationslogik nutzen. Hierbei häufig verwendete Komponenten sind Mediatoren, BPEL-Prozesse oder auch Business Rules. Je nach Situation kann die für den Anwendungsfall am besten geeignete Komponente gewählt werden. Eine weitere Stärke der Plattform sind Technologie-Adapter, um die einfache Anbindung von Drittsystemen zu ermöglichen. Die Palette reicht von Datenbanken bis hin zu Cloud-Anwendungen.

Auch wenn beide Ansätze in vielerlei Hinsicht gegensätzlich wirken, zielen sie jeweils auf die Entwicklung und Integration von Diensten ab. Es stellt sich daher die Frage, ob eine Plattform wie die Oracle SOA Suite für die Entwicklung einer Microservice-Architektur genutzt werden kann. Dazu gibt es folgende Leitfragen:

- Welche der oben beschriebenen Microservice-Prinzipien könnten genutzt werden?
- Wie kann die Aufteilung eines Systems in Microservices und Vertikalen mit der Plattform umgesetzt werden?
- Bringt dabei die Nutzung der Oracle-Infrastruktur Vorteile in der Entwicklung?

Eine wesentliche Eigenschaft von Microservices ist, dass sie unabhängig voneinander entwickelt, ausgeliefert und skalierbar sind.

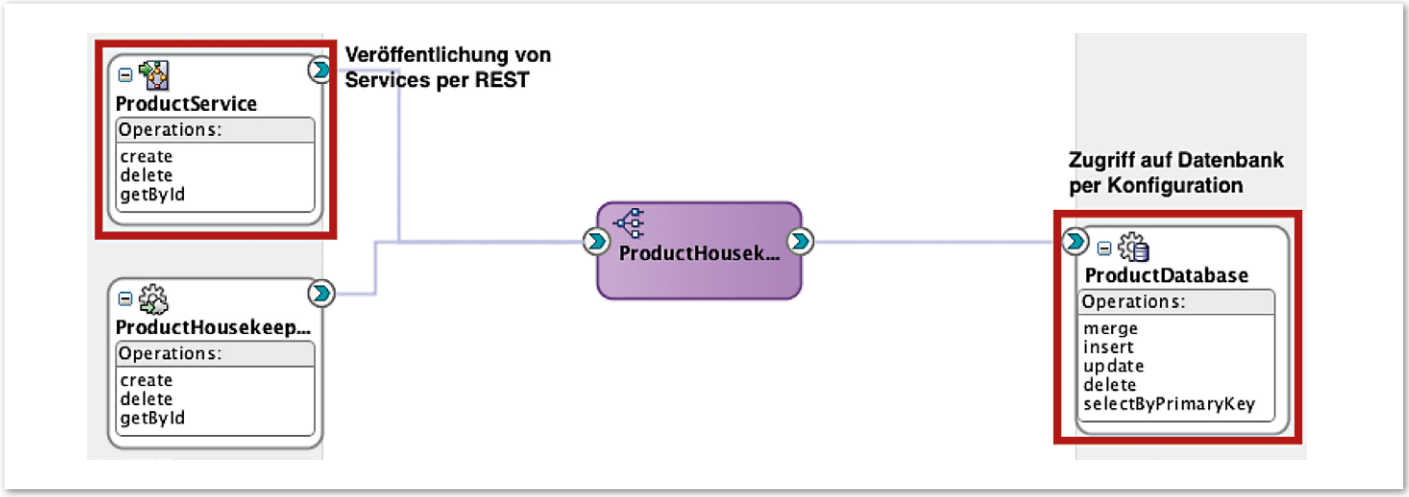


Abbildung 5: Einfacher CRUD-Microservice

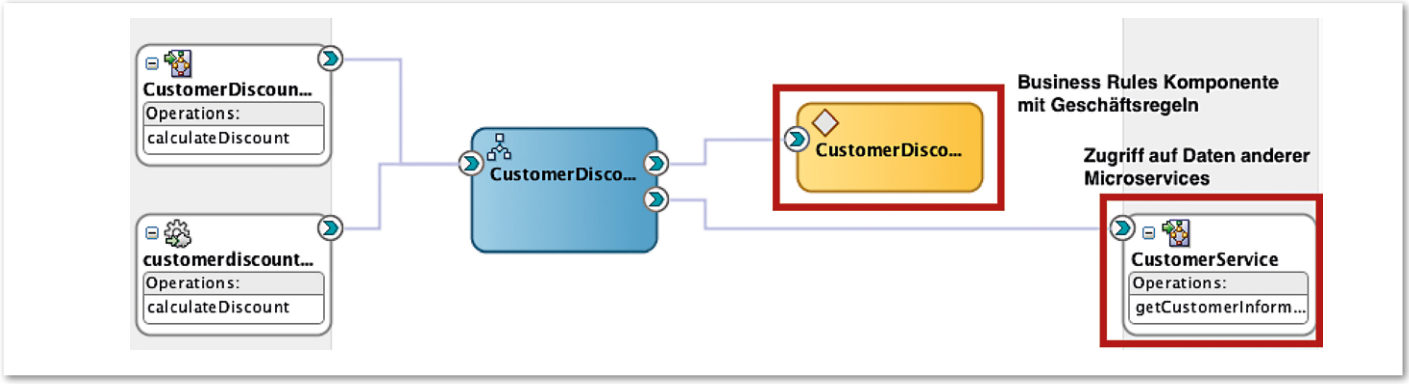


Abbildung 6: Microservice zur Umsetzung komplexerer Geschäftsregeln

Diese Eigenschaften können einzelne SCAs in der Oracle SOA Suite ebenfalls erfüllen. Sie sind in sich abgeschlossene Einheiten, die separat auf der Infrastruktur ausgeliefert werden können. Zur unabhängigen Skalierung lassen sie sich in der Laufzeitumgebung auf unterschiedliche Partitionen verteilen. Eine Partition bildet dabei eine logische Klammer über eine beliebige Anzahl von SCAs. Hinsichtlich des Skalierungsaspekts kann jeder Partition eine eigene Work-Manager-Gruppe zugewiesen werden. Mit deren Hilfe lassen sich einzelne Microservices höher priorisieren und gezielt skalieren *(siehe Abbildung 4)*.

Betrachtet man die in *Abbildung 3* skizzierte Architektur auf Basis von Microservices und Vertikalen, können einzelne Microservices als SCAs umgesetzt werden. Ein Microservice kann beispielsweise ein Dienst sein, der einfache CRUD-Operationen auf einer isolierten Produktdatenbank anbietet. Zur Umsetzung des Dienstes lassen sich in der Oracle SOA Suite Komponenten wie der Datenbank-Technologie-Adapter oder ein Mediator nutzen. Beide Komponenten stellen durch simple Konfiguration die gewünschten Funktionen zur Verfügung und basieren auf getesteter Standard-Technologie. Die so entwickelten Dienste können unter anderem nach dem REST-Ansatz zur Verfügung gestellt werden *(siehe Abbildung 5)*.

Um komplexere Logik in einem Microservice umzusetzen, können andere Komponenten aus der Palette der Oracle SOA Suite genutzt werden. Außerdem besteht die Möglichkeit, selbst entwickelte EJBs und Spring Beans zu nutzen.

Abbildung 6 zeigt den möglichen Aufbau eines Microservice, der einen kundenspezifischen Rabatt auf Basis komplexer Geschäftsregeln und Kundendaten berechnet. Neben den Eingangsdaten des Dienstes werden gegebenenfalls noch weitere Daten anderer Dienste benötigt. Im Beispiel werden Daten vom „CustomerService“ per REST-Anfrage bezogen. Die Eingangsdaten des „CustomerDiscount“-Service sowie die über eine REST-Anfrage erhaltenen Daten dienen als Eingabe für eine „BusinessRules“-Komponente. Dort sind die Geschäftsregeln hinterlegt, die den gewährten, spezifischen Kundenrabatt als Ergebnis liefern. Ein einfacher BPEL-Prozess koordiniert dies.

Die bereitgestellten REST-Dienste sind wiederum durch entsprechende Benutzeroberflächen erweiterbar und durch andere Dienste nutzbar. Die Kombination aus SCA und entwickelter Oberfläche kann entweder durch die JDeveloper-Build-Werkzeuge oder Maven zu einem Paket geschnürt und ausgeliefert werden. Durch die in Version 12*c* hinzugekommene Verwendung von Maven als Build-Werkzeug ist auch die Automatisierung der Infrastruktur sehr gut umsetzbar. Die isolierte Produkt-Datenbank, das SCA und die Benutzeroberfläche aus unserem Beispiel können so als abgeschlossene Einheit von einem Team entwickelt

und in einen automatisierten Build-Prozess überführt werden. Das Team kann dabei selbst entscheiden, welche Komponenten es zur Realisierung verwendet.

Das beschriebene Vorgehen zeigt eine Möglichkeit, wie Microservices mit der Oracle SOA Suite realisiert werden können. Wie beschrieben, ist durch die Einführung des Prinzips der vertikalen Dekomposition den einzelnen Microservices nun noch eine einheitliche Struktur gegeben. Dazu werden die einzelnen Vertikalen durch eine REST-Fassade ergänzt. Diese Fassade ist, im Sinne der Microservice-Eigenschaften, „dumm" und enthält keine komplexe Integrationslogik. Im Grunde werden die entsprechenden Anfragen nur an die zuständigen Microservices – in unserem Fall also einzelne SCAs – weitergeleitet. Die Dienste kommunizieren untereinander wiederum eigenständig, wodurch das Prinzip „Smart Endpoints and Dumb Pipes" umgesetzt ist. Hierzu können im einfachsten Fall Mediatoren zum Einsatz kommen.

Die Nutzung der Oracle-Plattform

Das beschriebene Vorgehen zeigt, dass die Entwicklung von Anwendungen, basierend auf Prinzipien des Microservice-Architektur-Stils und vertikaler Dekomposition, generell auch mit der Oracle SOA Suite möglich ist. Abschließend bleibt zu klären, ob der Einsatz der Oracle SOA Suite auch Vorteile mit sich bringt, die deren Einsatz rechtfertigen.

Um einer Klärung dieser Frage näher zu kommen, wird „Design for Failure" als eine weitere wichtige Charakteristik des Microservice-Paradigmas hinzugezogen. Darunter sind fortgeschrittene Überwachungsmöglichkeiten für Services und die Robustheit des Gesamtsystems gegenüber Ausfällen einzelner Dienste (sowie Netzwerk-Partitionen) subsummiert.

Gerade für den letzten Aspekt ist eine automatische Fehlerbehebung erstrebenswert *[1]*. Genau in diesen Bereichen bietet die Oracle SOA Suite vielfältige Funktionen und kann somit die Entwicklung von Anwendungen mit Microservices zusätzlich unterstützen. Die Infrastruktur unterstützt die Nachvollziehbarkeit für jede Dienst-Instanz („Audit-Trail"). Durch das Prinzip der „Flow-ID" kann die Abarbeitung einer Anfrage innerhalb, aber auch über die Grenzen eines Dienstes exakt nachvollzogen werden. Die Detailtiefe des Monitorings lässt sich über die SOA-Infrastruktur konfigurieren.

Im Hinblick auf das Auftreten und die Behandlung von Fehlersituationen kann man auf das „Fault Management Framework" der Oracle SOA Suite zurückgreifen. Diese Komponente ermöglicht die Konfiguration von automatischen Fehlerbehandlungsroutinen für SCAs, Komponenten oder auch Referenzen zu anderen Diensten. Die konfigurierbaren Maßnahmen reichen dabei von einem Abbruch der Abarbeitung über einen konfigurierbaren Retry-Mechanismus bis hin zur Aussteuerung über eine Workflow-Engine oder selbst geschriebene Java-Logik. Des Weiteren kann in definierten Fehlersituationen eine automatische Benachrichtigung („Alert") konfiguriert werden, etwa um das zuständige Entwicklerteam zu benachrichtigen und eine schnelle Fehlerbehebung zu gewährleisten.

Ein weiterer Aspekt, der für den Einsatz der Infrastruktur spricht, ist die Möglichkeit, die ausgereiften und leistungsstarken Komponenten des Frameworks zu nutzen. Häufig zu implementierende Standard-Funktionen, wie der Zugriff auf Datenbanken oder andere Drittsysteme, erfordern einen geringen Aufwand und können in den meisten Fällen konfiguriert werden.

Fazit

Insgesamt lässt sich festhalten, dass Anwendungen, entwickelt nach dem Architektur-Stil der Microservices, bestimmten Eigenschaften folgen. Der Fokus liegt in erster Linie auf der Entwicklung leichtgewichtiger, lose gekoppelter und skalierbarer Komponenten. Dabei werden in der Regel viele, jedoch selten alle der genannten Eigenschaften in Projekten umgesetzt *[1]*.

Auch wenn die Oracle SOA Suite mit ihrer Größe und dem umfangreichen Funktionsumfang als schwergewichtig angesehen werden kann, bietet sich auf ihrer Basis ein möglicher Ansatz zur Entwicklung von Microservices. Vorteilhaft sind dabei insbesondere konfigurierbare Standard-Komponenten und Mechanismen zur Fehlerkorrektur sowie im Bereich des Monitoring. Auch die Entwicklung und das Deployment der Microservices werden gut (Automatisierung) unterstützt. Generell ist zu sagen, dass sich die wichtigen Merkmale (Evolutionäres Design, Skalierbarkeit und Flexibilität) umsetzen lassen.

Neben den technischen und plattformabhängigen Aspekten muss jedoch auch immer ein Augenmerk auf die Projektstruktur geworfen werden. Die klare Definition sowie die Abgrenzung der Geschäftsbereiche und die Bildung von funktionsübergreifenden Projekt-Teams sind zentrale Grundprinzipien des Microservice-Architektur-Stils. Nur wenn diese gewährleistet sind, ist ein evolutionäres Design der einzelnen Dienste möglich. Gerade bei Verwendung der Oracle SOA Suite muss besonders auf die Umsetzung dieser Eigenschaften geachtet werden, da die zur Verfügung stehende Infrastruktur und deren umfangreiche Komponenten zur Entwicklung von zu komplexen Services oder Integrationslogik führen kann.

Weitere Informationen

[1] J. Lewis und M. Fowler, Microservices: http://martinfowler.com/articles/microservices.html

[2] C. Richardson, Microservices, Decomposing Applications for Deployability and Scalability, InfoQ: http://www.infoq.com/articles/micro-services-intro

[3] M. E. Conway, How Do Committees Invent?, Bd. Datamation magazine, 1968

[4] M. Loukides, What is DevOps?, O'Reilly Radar: http://radar.oreilly.com/2012/06/what-is-devops.html

Dr. Thomas Schuster
thomas.schuster@esentri.com

Carsten Wiesbaum
carsten.wiesbaum@esentri.com

Microservices und SOA: Zwei Architektur-Ansätze im Vergleich

Sven Bernhardt, OPITZ CONSULTING Deutschland GmbH

Viele Enterprise-Anwendungen, die Unternehmen bei der Abwicklung ihres Kerngeschäfts unterstützen, sind über die Jahre gewachsen und heute monolithisch aufgebaut. Diese sehr komplexen Anwendungen sind in der Regel nur noch schwer und entsprechend aufwändig anzupassen und zu erweitern.

Der noch junge Ansatz „Microservice-Architektur" greift das Problem auf und trägt dazu bei, dass monolithische Applikationen zukünftig der Vergangenheit angehören. Analog zum schon etwas betagteren Ansatz einer serviceorientierten Architektur (SOA) zielt er darauf ab, Software-Architekturen flexibler, besser ausbaufähig und skalierbarer zu machen.

Probleme monolithischer Geschäftsanwendungen

Historisch gewachsene, monolithische Anwendungen unterstützen fast immer eine Vielzahl von Geschäftsanwendungsfällen. Deren Architektur ist häufig in Form einer klassischen Schichten-Architektur mit einer Präsentations-, einer Geschäftslogik- und einer Persistenz-Schicht aufgebaut. Die Umsetzung der Anforderungen erfolgt horizontal, dabei arbeiten verschiedene Teams an einem Anwendungsfall. Dies führt zu erhöhten Abstimmungsbedarfen. Gemäß „Conway's Law" *(siehe „http://www.melconway.com/Home/Conways_Law.html")* bilden sich so innerhalb der Schichten unterschiedliche Implementierungsmuster heraus.

Die Implementierung erfolgt aus technologischer Sicht konsistent, für die Umsetzung wird also konsequent mit Technologien gearbeitet, die im Vorfeld festgelegt wurden. In der Konzeptionsphase entscheidet das Team, welche Technologien zum Einsatz kommen. Die Entscheidung fällt auf Basis der zu diesem Zeitpunkt vorhandenen funktionalen und nicht funktionalen Anforderungen. Ändern sich diese oder gibt es neue Voraussetzungen, gestalten sich die damit verbundenen Anpassungen bezogen auf die Technologien aufwändig oder sie werden gar nicht erst durchgeführt.

Die inhärente Komplexität einer monolithischen Anwendung macht die Änderungen und Erweiterungen aufgrund neuer fachlicher Anforderungen zu langwierig. Auch eine nicht vorhandene Test-Abdeckung, die generell schwierige Testbarkeit und komplexe Deployment-Mechanismen tragen dazu bei. Da Schnittstellen häufig nicht klar definiert wurden, ist es den Systemarchitekten kaum möglich, den Überblick über alle internen Abhängigkeiten zu behalten. Hinzu kommt die mangelnde Transparenz, die dazu führen kann, dass zuständige Service-Entwickler Funktionalitäten redundant implementieren, statt bereits Vorhandenes wiederzuverwenden. Da wie im Leben auch im Business nichts so beständig ist wie die Veränderung, müssen neue architektonische Mittel gefunden werden, um die Agilität von Unternehmen bezüglich sich ändernder Anforderungen langfristig sicherzustellen.

Jung und agil: Microservices

Einen Ansatz, um monolithische Applikationen und die damit verbundenen Probleme zu vermeiden, propagieren Microservice-Architekturen. Bei diesem Architektur-Stil wird die Funktionalität des Gesamtsystems über eine Summe autonomer Services abgebildet. Dabei bildet der einzelne Microservice einen dedizierten fachlichen Anwendungsfall ganzheitlich ab. Der Service implementiert also alle Komponenten, die für die Umsetzung der fachlichen Anforderungen notwendig sind. Dazu gehören beispielsweise Benutzer-Oberflächen, die Geschäftslogik oder entsprechende Persistenz-Komponenten. Microservices zeichnen sich daher durch eine hochgradige Kohäsion und Atomarität aus.

Im Gegensatz zum monolithischen Ansatz erfolgt die Implementierung bei Microservice-Architekturen nicht horizontal und schichtenorientiert, sondern auf der Basis des Anwendungsfalls, also vertikal von der GUI bis zur Persistenz. Dabei ist immer genau ein Team für die Umsetzung eines Microservice und des dazugehörigen Anwendungsfalls zuständig. Die klare Verantwortlichkeit des Teams minimiert den Aufwand für Abstimmungen und erhöht die Qualität der entwickelten Lösung. Die Gefahr, dass sich die Effekte nach Conway's Gesetz im implementierten Service manifestieren, wird damit so gut wie ausgeschlossen.

Aus technischer Sicht wird die Implementierung eines Microservice unter Verwendung der jeweils am besten geeigneten Technologie durchgeführt. Da ein Microservice nur einen abgeschlossen Anwendungsfall betrachtet, bildet sich bei Verwendung dieses Ansatzes ein Gesamtsystem heraus, das auf lange Sicht stabil ist. Dass dabei die Bewertung der Technologien für jeden Anwendungsfall individuell stattfindet, macht den Einsatz heterogener Technologien möglich und führt aus technischer Sicht zu einer optimalen Lösung.

Aufgrund ihrer hohen Kohäsion und Atomarität sind nachträgliche Anpassungen bei der Technologie-Auswahl in Systemen, die nach dem Microservice-Paradigma aufgebaut sind, einfacher als bisher möglich. Kommunikations-Beziehungen oder Abhängigkeiten zu anderen Microservices existieren entweder nicht oder sind über Schnittstellen definiert. Im letzten Fall werden Änderungen nur dann problematisch, wenn sich Schnittstellen ändern, was

bei Technologiewechseln im Allgemeinen nicht der Fall ist. Auch Änderungen an der Fachlogik wirken sich dank der Unabhängigkeit der Services nicht auf die Integrität und Stabilität des Gesamtsystems aus.

Da eine Enterprise-Architektur die Funktionsfähigkeit einzelner Services sowie der Gesamt-Applikation sicherstellen soll, gleichzeitig aber auch die schnelle Umsetzung fachlicher Änderungen und Erweiterungen ermöglichen möchte, bietet sich die Etablierung eines klaren Application-Lifecycle an. Dieser basiert auf automatisierten Tests sowie einer entsprechenden Continous-Delivery-Strategie und garantiert so jederzeit einen auslieferbaren, funktionsfähigen Stand. Ein solches Vorgehen kann zum Erfolg und zur Akzeptanz des Gesamtsystems beitragen.

Die bis hierhin angesprochenen Aspekte beziehen sich vor allem auf die Entwicklungszeit einer Microservice-Architektur. Damit die Integrität und Stabilität der Architektur auch zur Laufzeit sichergestellt ist, dürfen sich die Services nicht negativ beeinflussen. Um eine negative Wechselwirkung zu verhindern, kann ein Unternehmen so weit gehen, besonders hochfrequentierte Services innerhalb einer eigenen Laufzeitumgebung zu betreiben, beispielsweise in einer isolierten Java Virtual Machine (JVM). So bleiben die übrigen Systemkomponenten im Falle der Überlastung eines Service weitestgehend funktionsfähig.

Um Aspekte wie Durchsatz und Antwortzeiten zu optimieren, verzichten die Service-Entwickler soweit wie möglich auf Remote-Kommunikation zwischen den Services. In diesem Zusammenhang spielen auch Art und Menge der Daten, auf denen ein Microservice operiert, eine Rolle. Dabei sind im Kontext eines Service idealerweise nur die Daten verfügbar, die für den jeweiligen fachlichen Anwendungsfall relevant sind. Auch aus Compliance-Sicht kann dies von Interesse sein, wenn durch die Architektur die Integrität und Sichtbarkeit von sensiblen Daten, zum Beispiel von personenbezogenen Daten, sichergestellt wird.

Im Umkehrschluss hat dann jeder Service einen eigenen, isolierten Datenspeicher. Das kann sinnvoll sein, wenn hauptsächlich unstrukturierte Daten verarbeitet werden. Diese Daten werden dann am besten in einer NoSQL-Datenbank persistiert. Anders bei einem Service, der primär auf strukturierten Daten arbeitet, zum Beispiel auf Kundendaten. Hier verwenden die für die Umsetzung verantwortlichen Personen für die Datenpersistierung am effizientesten eine relationale Datenbank.

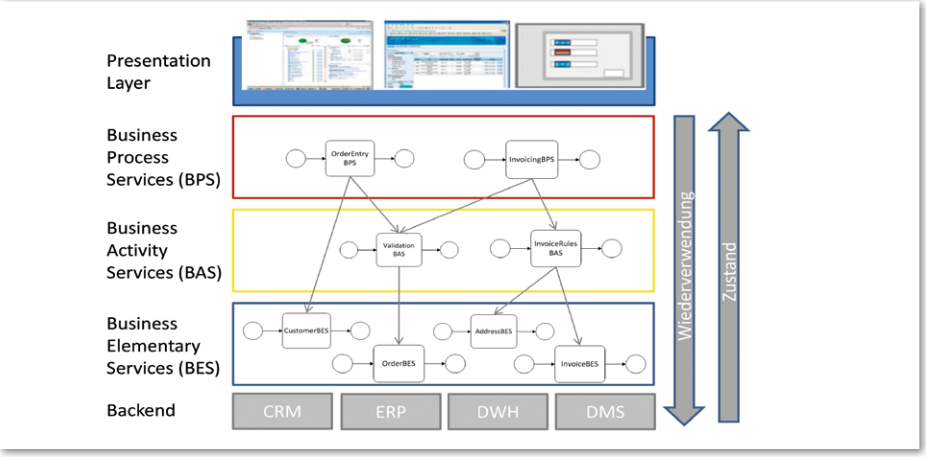


Abbildung 1: Beispiel für eine Service-Architektur

Neue Herausforderungen

Die Ausführungen zeigen, dass sich Microservice-Architekturen sehr gut dazu eignen, flexible, leicht anpassbare und skalierbare Systeme zu entwickeln. Auf der anderen Seite birgt der Architektur-Ansatz aber auch Herausforderungen. Diese gilt es frühzeitig zu adressieren. Zudem existieren diverse Fragen, auf die es aktuell noch keine einheitlichen Antworten gibt.

Eine Kernfrage bezieht sich auf die Charakterisierung von Microservices und zielt besonders darauf ab, ab wann beziehungsweise bis wann ein Service die Eigenschaft „Micro" erfüllt. Um die Antwort vorwegzunehmen: Sinnvolle, greifbare Eigenschaften gibt es derzeit nicht. Metriken, wie beispielsweise die Anzahl der Quellcode-Zeilen oder der Klassen, sind zwar gut messbar, stellen aber keine belastbare Maßeinheit für eine fundierte Bewertung dar. Auf der anderen Seite sind Kriterien, wie die Orientierung an fachlichen Funktionalitäten, zu unkonkret und weich, um auf ihrer Basis eine saubere Einordnung vornehmen zu können. Aus diesem Grund ist es umso entscheidender, dass der Business-Architekt im Vorfeld eindeutig und für alle Beteiligten verständlich vermittelt, wie sich ein Microservice definiert und wie der Service-Schnitt aussieht.

Der Wunsch nach Flexibilität, Erweiterbarkeit und Skalierbarkeit zieht ein gewisses Maß an Komplexität nach sich, aus dem Herausforderungen erwachsen, die es möglichst früh anzugehen gilt. Hierzu zählen unter anderem:

- Richtige Service-Granularität
- Transparentes Monitoring beziehungsweise Reporting bezüglich der Service-Verwendung, gerade vor dem Hintergrund der Service-Verteilung
- Konsistente und transparente Fehlerbehandlung (Definition von Timeouts, Etablierung von Retry-Mechanismen)
- Garantierte Nachrichten-Zustellung im Falle von Remote-Kommunikation
- Konsistentes Transaktions-Handling über Service-Grenzen hinweg
- Daten-Konsistenz und -Synchronität bei isolierten Datenspeichern
- Governance und Service Lifecycle Management

Diese Punkte bei den initialen Planungen zu berücksichtigen, ist ein wesentlicher Schlüssel für die erfolgreiche Implementierung einer Microservice-basierten Lösung.

Wie aktuell sind SOA-Services heute?

Mit der Etablierung einer SOA verfolgen Unternehmen die Absicht, ihre Flexibilität und Agilität für die Umsetzung von geänderten oder neuen fachlichen Anforderungen zu erhöhen. In diesem Zusammenhang gilt es, vorhandene monolithische Applikations-Silos schrittweise aufzulösen und ihre Funktionalitäten zu konsolidieren. Bei der Umsetzung sollten fachliche Zusammenhänge in Form autonomer Services gekapselt und an zentraler Stelle bereitgestellt werden. Um vorhandene Monolithen aufzubrechen, können Enter-

prise-Architekten bestehende Funktionalitäten weiterverwenden sowie in Services gekapselt bereitstellen. Grundlage für die Realisierung eines Service, unabhängig davon, ob es sich um neue oder vorhandene Funktionalitäten handelt, ist immer eine fachliche Anforderung.

Überhaupt ist die Wiederverwendung bestehender Funktionalitäten, Komponenten oder Artefakte ein zentraler Aspekt bei einer SOA. Die Wiederverwendung trägt dazu bei, dass die IT neue Anforderungen effizient umsetzen und produktiv nehmen kann. Klar zu definierende Governance-Mechanismen stellen hierbei sicher, dass keine redundanten Services umgesetzt werden. Dies ist auch hinsichtlich Wartbarkeit und Erweiterbarkeit entscheidend, damit ein Service-Entwickler für die Änderung bestehender Komponenten Anpassungen lediglich an einer zentralen Stelle durchführen muss.

Bei der Implementierung definieren Enterprise-Architekten die Services innerhalb einer SOA auf der Grundlage etablierter Standards, um die Interoperabilität der Komponenten untereinander ebenso wie die Kompatibilität bei Interaktionen mit externen Partnern zu garantieren. Definiert wird die Kommunikation über entsprechende standardbasierte Service-Kontrakte, die neben der eigentlichen Schnittstellen-Beschreibung auch Informationen zu Qualitätsmerkmalen wie Antwortzeiten enthalten. Die Verwendung von Standards ist auch mit Blick auf die Zukunftssicherheit der Gesamt-Architektur förderlich. Aus technologischer Sicht gibt es für die Implementierung von Services keinerlei Restriktionen oder Vorgaben.

Bei der Architektur einer SOA sind Fragestellungen zur Service-Granularität und zum Zusammenspiel der Einzelkomponenten zu berücksichtigen. Dies ist auch aus Sicht der Governance interessant, um den Überblick über vorhandene Funktionalitäten zu behalten.

Damit die Architekten aufkommende Fragestellungen adäquat beantworten können, sollten sie möglichst frühzeitig eine Kategorisierung oder Klassifizierung von Services vornehmen. Die verschiedenen Kategorien sagen etwas über die Eigenschaften von Services aus. Weiterhin werden im Rahmen einer solchen Klassifizierung auch das Interaktionsverhalten und die dazugehörigen Regeln festgelegt. Aus all diesen Punkten resultiert am Ende eine Servicearchitektur, die normalerweise hierarchisch aufgebaut ist. Ziel dieses Vorgehens ist es, die Transparenz der Lösungs-Architektur und somit die Beherrschbarkeit der resultierenden SOA auch auf lange Sicht sicherzustellen.

Abbildung 1 zeigt ein Beispiel für eine Service-Architektur. Die Services auf der untersten Ebene haben nur wenige Abhängigkeiten, werden dafür aber häufig wiederverwendet. Bei den Services auf den oberen Ebenen verhält es sich umgekehrt. Die Kommunikation der Komponenten untereinander ist so geregelt, dass diese nur in eine Richtung, nämlich von oben nach unten erfolgt.

Microservice versus SOA

In der aktuellen Diskussion wird das junge Konzept „Microservice-Architektur" oftmals vom älteren SOA-Ansatz abgegrenzt. Dabei werden Unterscheidungsmerkmale genannt wie monolithische Deployments, der Einsatz einer entsprechend komplexen Middleware-Plattform, bestehend unter anderem aus Applikationsserver und Enterprise Service Bus (ESB), oder den Einsatz von Webservice-Standards (WS-*). Im Gegensatz zur SOA werden solche komplexen Strukturen bei Microservice-Architekturen nicht verwendet, sondern es wird Wert auf leichtgewichtige Komponenten und Technologien gelegt. Allerdings gibt es bei einer SOA gar keine Vorgaben bezüglich der zu verwendenden Plattformen oder Technologien. Auch hier könnte man also durchaus mit leichtgewichtigen Komponenten und Technologien arbeiten. Die Schwarz/Weiß-Sichtweise sollte somit der Vergangenheit angehören.

Beide Ansätze haben gemeinsam, dass es verschiedene Meinungen über ihre Definition gibt. Dazu kommt der unterschiedliche Ursprung, der die aktuelle Diskussion zusätzlich erschwert. Den SOA-Ansatz haben Vertreter der Enterprise-Architektur-Richtung ins Spiel gebracht, die komplexe Middleware-Plattformen seit jeher für Integrations- und Automatisierungsaspekte nutzen. Der Microservice-Ansatz hingegen wurde aus der Enterprise-Java-Richtung getrieben, in der komplexe Middleware-Lösungen eher die Ausnahme sind.

Beide Ansätze haben ihre Stärken und ihre Schwächen – und sicherlich können beide gewinnbringend eingesetzt werden, wenn sie zu dem zugehörigen Szenario passen. Auch bei den Zielsetzungen, Herausforderungen und Charakteristiken gibt es viele Gemeinsamkeiten:

- *Zielsetzung*
  Steigerung von Flexibilität und Agilität
- *Herausforderungen*
  Governance, Betrieb, Monitoring und Service-Schnitt
- *Charakteristika*
  Abbildung der Gesamtsystem-Funktionalität über ein Set autonomer Services

Eine Architektur auf der Basis von Microservices unterscheidet sich von einer SOA vor allem dadurch, dass sie benötigte GUI-Komponenten mit implementiert und ihre Funktionalitäten nicht per se nach außen über standardbasierte Schnittstellen exponiert.

Fazit

Grundsätzlich ist eine Koexistenz ebenso wie eine wertschöpfende Kombination beider Ansätze denkbar. Betrachtet man die Grafik zur Service-Architektur, so wäre es hier beispielsweise möglich, Microservices in Form einer eigenen Kategorie, als Private Services, in eine SOA zu integrieren. Sie würden dann zunächst nur innerhalb einer bestimmten Domäne verwendet, da die Funktionalitäten in anderen Bereichen nicht benötigt werden. In anderen Kontexten kämen diese Services dann erst einmal nicht zum Einsatz. Bei Bedarf wäre es dann allerdings jederzeit möglich, benötigte Funktionalitäten über Schnittstellen extern bereitzustellen. Es bleibt spannend zu beobachten, wie sich der Microservice-Ansatz weiterentwickelt und welchen Einfluss der neue Architekturstil auf bestehende Unternehmens-Architekturen haben wird.

Sven Bernhardt
sven.bernhardt@opitz-consulting.com

Cloud-Architekturen: Klassische Probleme – neu gelöst

Eberhard Wolff, innoQ Deutschland GmbH

Cloud ist mehr als nur ein Deployment-Modell: Es ist die Grundlage moderner Software-Entwicklung. Die effektive Nutzung der Cloud ist der Grund, aus dem Google und Amazon Software so effizient entwickeln können. Ursprung ist ein anderer Architektur-Ansatz – der auch zukunftsweisend für andere IT-Systeme sein kann.

Für den Begriff „Cloud" gibt es viele unterschiedliche Definitionen. Die wesentliche Eigenschaft von Cloud ist Self Service. Ressourcen aus der Cloud kann ein Nutzer einfach buchen – ohne dass dazu irgendjemand manuell eingreifen muss. Konkretes Beispiel: Ein virtueller Server in einer normalen IT-Landschaft ist nur mithilfe eines manuellen Prozesses möglich. Zu den manuellen Schritten kann neben der Einrichtung der Maschine und der Installation der Software auch ein Genehmigungsprozess gehören.

Eine Cloud-Umgebung bietet ein Portal, in das sich der Mitarbeiter einloggen kann. Anschließend startet er dort den Rechner mit einer bestimmten Basis-Installation. Im Hintergrund findet die Abrechnung statt. Statt des Portals kann ein API dem Starten und Stoppen der Maschinen dienen. Dann lässt sich beispielsweise als Reaktion auf eine hohe Last einfach eine neue virtuelle Maschine starten, die dann einen Teil der Last übernimmt.

Welche Ressourcen gebucht werden können, ist ein wesentliches Merkmal zur Unterscheidung der verschiedenen Cloud-Spielarten:

- *Infrastructure as a Service (IaaS)*
  Bietet Infrastruktur wie virtuelle Server oder Storage-Systeme. Beispiele sind Amazon EC2 (Server) oder EBS (Storage) *[1]*. Andere Anbieter wie Digital Ocean *[2]* sind auch in diesem Markt aktiv. Im eigenen Rechenzentrum können solche Lösungen durch Produkte wie OpenStack umgesetzt werden.

- *Platform as a Service (PaaS)*
  Erzeugt eine Infrastruktur, in denen Anwendungen eingerichtet werden können. Der Entwickler muss der Versionskontrolle nur noch Änderungen übergeben. PaaS liest die Änderungen aus und bringt die neue Version automatisch in Produktion. Öffentliche Clouds in diesem Bereich sind Amazon Elastic Beanstalk *[1]* oder Heroku *[3]*; für das eigene Rechenzentrum können OpenShift *[7]* oder Cloud Foundry *[8]* Ähnliches bieten.
- *Software as a Service (SaaS)*
  Bietet Software wie Office Suites oder Customer Relationship Management (CRM) an. Auch Wikis oder Continuous-Integration-Server können in den Bereich „SaaS" fallen. Selbst für Monitoring oder die Auswertung von Log-Dateien gibt es SaaS-Lösungen. Für Software-Entwicklung sind diese Lösungen oft sinnvoll, um eine Infrastruktur kostengünstig und schnell aufzustellen. Die Software-Architektur beeinflusst es aber nicht, sodass dieser Artikel SaaS nicht weiter betrachtet.

Für die Amazon-Cloud-Angebote gibt es ein kostenloses Kontingent für Neukunden. Ein Tutorial führt durch die ersten Schritte *[9]*. Ähnliches gilt für viele andere Clouds wie beispielsweise Heroku *[10]*.

Cloud und Software-Entwicklung

IaaS und PaaS scheinen nur eine Infrastruktur für Systeme anzubieten. Für eine Software-Architektur ist das nicht so wichtig. Schließlich sollte die Plattform, auf der die Software läuft, die Architektur nicht zu stark beeinflussen. Warum sollte sich also die Architektur durch Cloud entscheidend ändern? Die Cloud bietet eine viel flexiblere Infrastruktur. Hardware wird zu Software. Früher bedeutete ein neuer Server, dass ein Admin in einem Server Rack einen neuen Rechner eingebaut hat. In der Cloud ist ein neuer Rechner nur der Aufruf eines API. Dahinter steckt die Idee von „Infrastructure as Code". Auch die Infrastruktur wird also zu Code, der APIs zum Aufbau von Servern aufruft oder Skripte enthält, die Software auf den Servern installieren.

Continuous Delivery

Eine flexible Infrastruktur ermöglicht Optimierungen der Software-Entwicklung. Wenn es so einfach ist, neue Server zu starten und mit Software zu versehen, lassen sich Test-Systeme ohne viel Aufwand aufbauen. Ebenso können Produktionssysteme schnell entstehen. Das kommt dem Continuous-Delivery-Ansatz *[4]* zugute. Grundlage von Continuous Delivery und der Continuous Delivery Pipeline sind die verschiedenen Phasen *(siehe Abbildung 1)*:

- In der Commit-Phase werden die Software kompiliert, die Unit Tests ausgeführt und eine statische Code-Analyse durchgeführt. Diese Phase umfasst also die Schritte, die heutzutage ein Continuous-Integration-Server typischerweise durchführt.
- Die automatisierten Akzeptanztests überprüfen, ob die notwendigen Funktionalitäten korrekt implementiert worden sind.
- Kapazitätstests überprüfen, ob der notwendigen Anzahl Benutzer die erforderliche Performance geboten werden kann.
- Explorative Tests können neue Features oder bekannte Probleme ausleuchten.
- Schließlich wird die Software in Produktion übergeben.

Die Continuous Delivery Pipeline sollte mehrmals pro Tag durchlaufen werden. Dazu sind einige Aufwände notwendig. Die Ansprüche an die Infrastruktur sind sehr hoch. Cloud hilft in verschiedenen Bereichen:

- Für die Test-Phasen kann die Cloud-Infrastruktur ohne große Probleme bereitgestellt werden, was die Durchführung dieser Phasen entscheidend vereinfacht. Die notwendigen Systeme sind auch nur für die Tests notwendig. Das kommt der Cloud entgegen, weil dort nur die tatsächlich verbrauchte Rechenzeit berechnet wird.
- Für die Produktivstellung können verschiedene Ansätze genutzt werden, um das Risiko zu reduzieren. Blue/Green Deployment baut eine komplett neue Produktionsumgebung auf. Erst wenn sich diese Umgebung in Tests bewährt hat, wird auch der Traffic auf die Umgebung umgeleitet. Canary Releasing richtet zunächst Software auf einem Server im Cluster ein. Wenn die Software ihre Funktionstüchtigkeit unter Beweis gestellt hat, kommt sie auch auf die anderen Server. Die dafür notwendigen Infrastrukturen lassen sich mit Cloud-Ansätzen sehr einfach bereitstellen. Sie werden auch nicht besonders lange benötigt. Mit klassischen Ansätzen ist diese Flexibilität kaum zu erreichen. Einfach so eine zweite Produktionsumgebung aufzubauen, ist praktisch unmöglich.

Änderbarkeit

Continuous Delivery nutzt die Flexibilität der Cloud für bessere Deployment-Prozesse. So unterstützen Cloud und Continuous Delivery ein Qualitätsmerkmal einer guten Architektur – nämlich Änderbarkeit. Eine saubere Strukturierung der Software und damit eine gute Software-Architektur sollte die Software ebenfalls leichter änderbar machen.

Continuous Delivery geht Änderbarkeit anders an: Durch die Tests und das geringe Risiko beim Deployment können Änderungen schnell und sicher in Produktion gebracht werden. Dieser Aspekt ist für die Änderbarkeit sehr wichtig. Änderungen an

Abbildung 1: Continuous Delivery Pipeline im Überblick

einem System mit guter Architektur, aber ohne Tests, sind sicher schwieriger als Änderungen an einem schlecht strukturierten System mit vielen Tests. Also bietet Continuous Delivery einen ganz anderen Ansatz für die Änderbarkeit der Software, als die Software-Architektur das tut.

Skalierbarkeit

Ein weiteres wichtiges Qualitätsmerkmal von Software-Architekturen ist die Skalierbarkeit. Typischerweise werden diese Herausforderungen gelöst, indem die Software auf einem oder mehreren Systemen mit der notwendigen Leistung eingerichtet wird. Die Systeme müssen so dimensioniert sein, dass sie auch Lastspitzen handhaben können. Wenn die Vorhersage über die notwendige Leistung falsch ist, ist entweder Geld für Server verschwendet worden oder die Last kann nicht mehr gehandhabt werden – was zu Ausfällen und finanziellen Verlusten führen kann. Außerdem ist die Skalierung grobgranular: Wenn die Software auf zwei Servern läuft, ist eine Skalierung auf drei Servern machbar – das erhöht die Leistung allerdings gleich um fünfzig Prozent und ist wahrscheinlich zu viel.

In der Cloud kann das Problem gelöst werden, indem die Software abhängig von der Last neue Rechner startet und die Last aufteilt. So nutzt das System nur die Ressourcen, die gerade benötigt werden. Eine Vorhersage ist nicht notwendig. Die Skalierung kann feingranularer sein. Dazu nutzt man mehr, aber weniger leistungsfähige Server – also beispielsweise zehn Server für die normale Last. Dann kann die Leistung in Zehn-Prozent-Schritten erhöht werden.

Besonders gut funktioniert der Ansatz natürlich, wenn es um ein Internet-Angebot geht und die Anzahl der Nutzer nur schwer vorab festgestellt werden kann. Aber auch Anwendungen mit weniger Schwankungen profitieren, da eine Vorhersage über die Last nicht mehr notwendig ist und unvorhergesehene Lastspitzen abgefangen werden können.

Die Architektur kann so Skalierbarkeit anders umsetzen. Allerdings muss die Software auch dementsprechend eine Verteilung auf mehrere Systeme ermöglichen – dabei muss auch eine höhere Leistung erreicht werden. Wenn also das System einen Punkt hat, den jede Anfrage durchlaufen muss, entsteht ein Flaschenhals, der letztendlich die Skalierung gefährdet.

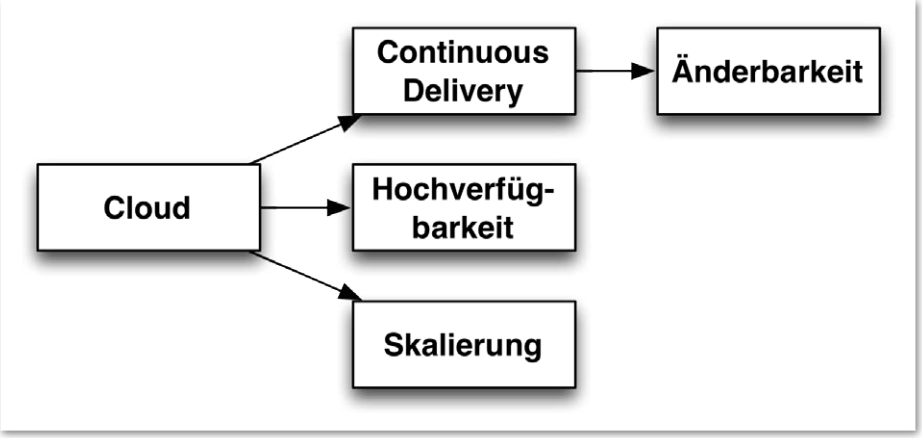


Abbildung 2: Vorteile der Cloud für Software-Architekturen

Hochverfügbarkeit

Klassische Systeme unterstützen Hochverfügbarkeit, indem die Hardware hochverfügbar gemacht wird – beispielsweise durch redundante Netzteile, Netzwerk-Interfaces etc. Zusätzlich wird die Software auf zwei Knoten betrieben, um gegen den Ausfall eines Knotens abgesichert zu sein. Wenn beide Server ausfallen, fällt auch das System aus – daher müssen die Server entsprechend hochverfügbar sein.

In der Cloud lassen sich zur Laufzeit neue Systeme starten. Daher kann der Ansatz zur Hochverfügbarkeit anders aussehen: Die Software startet beim Ausfall eines Knoten einen neuen. Dazu muss die Software mit dem Ausfall eines Knotens umgehen können – also dürfen in dem Knoten keine speziellen Informationen gespeichert sein, die beim Ausfall verloren wären. Dann kann dieser Ansatz sogar eine noch höhere Verfügbarkeit bieten als klassische Architekturen.

In einer klassischen Architektur wird ein großer Teil des Problems von der Software in die Hardware verlagert. Wenn diese ausfällt, funktioniert auch die Software nicht mehr. Dieser Fall tritt vielleicht nicht häufig auf; wenn dies jedoch der Fall ist, sind die Folgen katastrophal. In der Cloud sind Ansätze zum Umgang mit Hardware-Ausfall vorhanden, sodass der Ausfall keine ernsthaften Konsequenzen haben kann.

Die Software so zu strukturieren, dass sie mit dem Ausfall von Hardware umgehen kann, erhöht nicht nur die Verfügbarkeit, sondern senkt auch die Kosten. Schließlich ist keine hochverfügbare Hardware mehr notwendig. Die meisten Cloud-Anbieter haben auch mehrere Rechenzentren, sodass Software sich sogar gegen den Ausfall eines Rechenzentrums oder andere Katastrophen absichern kann, indem dann neue Server in anderen Rechenzentren gestartet werden. Unternehmen wie Google oder Amazon verwenden einen Ansatz für ihre hohe Verfügbarkeit, die sie tagtäglich im Internet unter Beweis stellen.

Warum Cloud?

Viele meinen, dass Cloud vor allem der Reduktion von Kosten dient. Das ist aber nur ein Grund. Vor der Entwicklung der Cloud-Infrastruktur hat Amazon *[5]* beobachtet, dass Entwickler sehr viel Zeit mit Skalierbarkeit verbringen, statt neue Features zu implementieren. Durch die Cloud hat Amazon für sich eine Basis für Skalierbarkeit geschaffen. Die dafür notwendige Software ist ein signifikantes Investment, das durch eine bessere Hardware-Auslastung sicher nicht zu rechtfertigen ist. So bietet die Cloud heute eine andere Basis für Skalierbarkeit und Ausfallsicherheit. Außerdem ist es die Grundlage für moderne Deployment-Verfahren wie Continuous Delivery *(siehe Abbildung 2)*.

Microservices = Architektur für die Cloud?

Um die Vorteile der Cloud auszunutzen, muss allerdings auch die Software passend strukturiert sein. In der letzten Zeit haben sich Microservices *[6]* als neuer Architektur-Ansatz positioniert. Ein Microservices-System ist in einzelne Deployment-Artefakte aufgeteilt, die einzeln und ohne Abstimmung mit anderen Artefakten in Produktion gebracht werden können. In einem E-Commer-

ce-Shop können die Verwaltung der Bestellungen, der Registrierungsprozess und der Katalog jeweils einzelne Microservices sein, die getrennt in Produktion kommen. Eine Änderung am Registrierungsprozess wird also in einem Microservice implementiert, der dann unabhängig von allen anderen Microservices in Produktion gebracht wird.

Jeder Microservice läuft in einer eigenen virtuellen Maschine. Die Kommunikation mit anderen Microservices erfolgt über das Netz mit Protokollen wie REST oder Messaging. So setzen sich Microservices klar von Deployment-Monolithen ab. Dieser Architektur-Stil hat Synergien mit der Cloud:

- Während die Cloud den Aufbau von Infrastrukturen für Continuous Delivery vereinfacht, sind Microservices kleine Deployment-Einheiten. Daher benötigt ein Microservices-System mehr Continuous Delivery Pipelines, die einfacher aufzubauen sind. Ebenso wird der Durchlauf durch die Pipelines schneller, weil für kleinere Deployment-Einheiten auch weniger Tests notwendig sind.
- Die Kommunikation zwischen Microservices erfolgt über das Netz. Jeder Microservice läuft auf einem eigenen Server. Daher ist der Ausfall eines Microservice viel wahrscheinlicher als der Ausfall eines Moduls in einer klassischen Architektur. Microservices müssen also gegen den Ausfall anderer Microservices abgesichert sein. Man spricht von „Resilience“. Beispielsweise können Default-Werte genutzt oder ein vereinfachter Algorithmus verwendet werden, wenn der eigentlich Microservice nicht zur Verfügung steht. Resilience verringert das Risiko des Ausfalls des Gesamtsystems und trägt damit zur Hochverfügbarkeit bei. Außerdem reduziert es das Risiko eines Deployments, da selbst der Ausfall des Microservice nach dem Deployment kaum Konsequenzen hat.
- Die Aufteilung in Microservices kommt der Verfügbarkeit zugute, weil der Ausfall eines Microservice keinen anderen beeinträchtigt – selbst wenn das Betriebssystem oder die ganze virtuelle Maschine in Mitleidenschaft gezogen wird. In einem Deployment-Monolithen kann ein Fehler, der zur Allokation von immer mehr Speicher führt, das gesamte System zum Absturz bringen. Alle Module laufen im gleichen Prozess. Wenn ein Fehler den Prozess zum Absturz bringt, ist das gesamte System ausgefallen. Microservices trennen das System in Module auf, bei denen der Ausfall eines Systems kein anderes System stört. Das ist natürlich auch eine gute zusätzliche Absicherung gegen den Ausfall einzelner Server in der Cloud.
- Microservices können einzeln durch das Starten zusätzlicher virtueller Maschinen skaliert werde. Auch das unterstützt den Skalierungsansatz der Cloud gut.

Fazit

Cloud ist mehr als nur eine Infrastruktur. Ein Architekturziel wie Flexibilität oder Anpassbarkeit kann durch Continuous Delivery und die damit einhergehenden Tests erreicht werden. Diesen Ansatz unterstützt Cloud als Infrastruktur ideal. Verfügbarkeit und Skalierbarkeit lassen sich auch anders umsetzen. Die Aufteilung in Microservices nutzt diese Vorteile ideal aus. So stellt die Cloud die Basis für moderne Architekturen dar.

Weitere Informationen

[1] http://aws.amazon.com/de
[2] https://www.digitalocean.com
[3] https://www.heroku.com
[4] Eberhard Wolff: Continuous Delivery: Der pragmatische Einstieg, dpunkt, 2014, ISBN 978-3864902086
[5] http://jandiandme.blogspot.com/2006/10/jaoo-2006-werner-vogels-cto-amazon.html
[6] Eberhard Wolff: Continuous Delivery: Grundlagen flexibler Software Architekturen, dpunkt, erscheint 2015
[7] https://www.cloudfoundry.org
[8] https://www.openshift.com
[9] http://aws.amazon.com/de/getting-started
[10] https://signup.heroku.com/www-home-top

Eberhard Wolff
eberhard.wolff@innoq.com

Internet of Things: Referenz-Architektur

Marcel Amende, ORACLE Deutschland B.V. & Co. KG

Vor zwei Jahren befand Kanzlerin Merkel noch, das Internet sei „... für uns alle Neuland“. In diesem Jahr findet sie Facebook bereits „... so schön, wie ... eine ordentliche Waschmaschine.“ Dabei ist die nächste tiefgehende Umwälzung in der IT bereits in vollem Gange, getrieben von einer explosionsartigen Verbreitung und Vernetzung kleiner, kostengünstiger und intelligenter Geräte.

In Deutschland wird häufig der Begriff „Industrie 4.0“ verwendet, der wie eine unnötige Selbstbeschränkung erscheint. Bei aufgeschlossener und kreativer Betrachtung der entstehenden technischen Möglichkeiten zeigt sich, dass es um weit mehr gehen kann als die Vernetzung von Fabriken und Maschinen. Tatsächlich werden alle Lebensbereiche durchdrungen, das uns bekannte Internet verändert sich zu einem „Internet der Dinge“ (IoT). Salopp gesagt, hat Merkels Waschmaschine heute Facebook. In Anbetracht der Bedeutung des Themas drängt es sich auf, belastbare Referenz-Architekturen für IoT-Umsetzungen zu entwickeln. Der Artikel zeigt dies aus einer On-Premise- und Cloud-Sicht.

Anforderungen einer IoT-Lösung

Eine vollständige IoT-Lösung *[1]* besteht aus verschiedenen Komponenten *(siehe Abbildung 1)*:

- *Geräte*
  Weit verteilte oder sogar mobile Geräte und Maschinen interagieren, mit einer Vielzahl von Sensoren und Aktoren ausgestattet, messend, steuernd und regelnd möglichst energieeffizient mit ihrer Umwelt. Sie sammeln dabei eine Vielzahl von Daten ein: Position, Bewegung, Umgebungstemperatur, Luftfeuchte, Vibrationen, aber auch Nutzungs- und Produktionskennzahlen. Dies kann vom schaltbaren Leuchtmittel über das regelbare Heizungs-Thermostat, die fernwartbare Waschmaschine bis hin zur Produktionsmaschine reichen. Apple Manager Jeff Williams bezeichnete jüngst das kommunizierende Auto als „ultimatives Mobilgerät“.

- *Gateways*
  Das Gateway sammelt und verarbeitet Daten von einem oder mehreren IoT-Geräten in seiner näheren Umgebung ein. Es nimmt eine intelligente Vorverarbeitung und Filterung der Daten vor und wandelt sie für eine sichere Übertragung an zentrale Systeme in ein möglichst standardisiertes Format. Zusätzlich werden dem Gateway Wartungs-, Verwaltungs- und Diagnose-Aufgaben übertragen: Es überwacht, (de-)aktiviert die Geräte der Umgebung und bietet Zugang etwa für die Verteilung von Software-Updates. Je nach Ökosystem des jeweiligen Anbieters kann die Gateway-Funktionalität in Netzwerkgeräte, Mediacenter oder Heizungssteuerungen eingebettet sein.

- *Netzwerk*
  Gateways kommunizieren üblicherweise in verschiedenen Netzwerken. Geräte können in persönlichen Netzen (PAN) per USB oder Low-Energy Bluetooth angebunden sein. Oft existieren bereits lokale drahtgebundene Ethernet- oder drahtlose WiFi-Netzwerke (LAN). Server sind in weitreichenden Netzen (WAN) über das öffentliche Internet, auch per Glasfaser- oder Satelliten-Verbindungen erreichbar. Dieses kann über sichere Tunnel zwischen Client und Server virtuell privat (VPN) erfolgen. Hier besteht die Herausforderung darin, die Netzwerk-Kommunikation zuverlässig, sicher, vertraulich, aber auch möglichst schnell und effizient zu gestalten.

- *Server*
  Die zentrale Infrastruktur übernimmt die weitere Filterung, Speicherung und

```
// I/O Port Nr. 23 für den Zugriff öffnen
GPIOPin myLED = DeviceManager.open(23);
// LED einschalten
myLED.setValue(true);
```

Listing 1

Abbildung 1: Die IoT-Lösungs-Komponenten

Analyse der übermittelten Daten. Diese können je nach Anforderung in unstrukturierter, strukturierter, anwendungsoptimierter und voraggregierter Form verwaltet werden. Oft werden hier mit Datenstrom-Analysen und Batchverfahren gleichzeitig parallel verlaufende Auswertepfade angewendet, um sowohl schnell Aktionen auslösen als auch zum Zeitpunkt der Erhebung unbekannte Fragestellungen auf historischen Daten und Rohdaten beantworten zu können.

- *Applikationen*
  Schließlich bleibt noch die Einbindung von Unternehmensapplikationen, um die aus der Vernetzung gewonnene Informationsfülle im Sinne der Geschäftsprozesse des Unternehmens zu nutzen. Neben Infrastruktur- und Lösungs-Design ist hier die unternehmerische Intelligenz und Kreativität gefragt, um aus innovativen IoT-Lösungsansätzen ein profitables Geschäft zu machen.

In der Folge werden die IoT-Komponenten-Bausteine mit Oracle-Technologien belegt, um zu einer tragfähigen Gesamtlösung zu kommen.

IoT-Geräte und Sensoren mit Java ME 8

Software für die Geräte im Internet der Dinge zu entwickeln, kann herausfordernd sein, da wegen weiter Verbreitung (auch räumlich), beschränkter Ressourcen, Variantenvielfalt und meist langer Lebensdauer manuelle Eingriffe in die Geräte-Software nicht möglich sind. Java-Entwickler sind hier in einer komfortablen Situation: Mit der Java Micro Edition (ME) bietet sich die Möglichkeit, auf eine standardisierte und wohlbekannte Entwicklungsplattform zu setzen, um die Abhängigkeiten von Hardware und Betriebssystem zu eliminieren. So lassen sich bereits erlernte Java-Fähigkeiten nutzen und proprietäre, teure Speziallösungen vermeiden. Dies beschleunigt und vereinfacht die Umsetzung neuer, eventuell noch experimenteller IoT-Anwendungen und Ideen mit unbekanntem Erfolgspotenzial.

Das Java ME 8 SDK ist ein für Embedded-Entwicklung optimierter Werkzeugkasten, der – unter Beachtung der limitierten Ressourcen und Bandbreiten von Embedded-Systemen – die Entwicklung von sicheren und zuverlässig betreibbaren Anwendungen ermöglicht. Das SDK zielt auf kleinere und mittlere Embedded-Systeme mit mindestens 128 KB RAM und 1 MB Flash oder ROM, wie etwa Sensoren, Netzwerk-, Mess- oder medizinische Geräte, ab.

Das Java-ME-8-API ist eine strikte Untermenge von Java SE 8, bietet aber viele Funktionalitäten, um spezifische Embedded-Anforderungen zu unterstützen: Eher ungewöhnlich für Java, ermöglicht das Device-I/O-API dem Entwickler einfachen Zugriff auf die Geräte-Peripherie und Bussysteme, um beispielsweise Sensoren und Micro Controller anzusteuern. Mit zwei Zeilen Java-Code lassen sich so Leuchtsignale über die I/O-Ports eines IoT-Geräts schalten *(siehe Listing 1)*.

Verschiedene Referenz-Implementierungen von Java ME 8 sind verfügbar, etwa die Raspberry-PI- und die Qualcom-IoE-Plattform. Beide sind mit ARM-Prozessoren ausgestattet. Zudem stehen Vorabversionen für Cortex-M4-basierte Systeme von Freescale und STM zum Download bereit. Die spezifischen Eigenschaften der Systeme werden unterstützt, indem zum Beispiel die I/O-Ports und Bussysteme der Geräte vorkonfiguriert sind. Beim Gerät von Freescale betrifft dies den Zugriff auf das integrierte Gyroskop und den Kompass, was die Entwicklung von Applikationen etwa für Datenbrillen erleichtert, die eine räumlichen Orientierung erfordern.

Java SE Embedded und Oracle Event Processing

Gateways sind leistungsfähigere Embedded-Systeme wie beispielsweise Industrie-PCs. Hier stehen mit der Java Standard Edition (SE) und Java SE Embedded ebenfalls die passenden und gegebenenfalls modularen Java-Versionen zur Verfügung. Sie übernehmen die dezentrale Kommunikation mit den lokalen Sensorsystemen und transportieren die relevanten Sensordaten auf sicherem Wege in das zentrale Rechenzentrum. Java SE ist in diesem Rahmen geeignet, komplexere Logik und Anwendungen zu betreiben. Es enthält eine Embedded-Java-Datenbank für die lokale Datenverwaltung und Auswertung mit SQL.

Auch das Oracle Event Processing (OEP) lässt sich mit Java SE Embedded betreiben. So kann komplexe Entscheidungslogik auf den eingehenden Datenströmen graphisch modelliert und abgebildet werden. Ein ty-

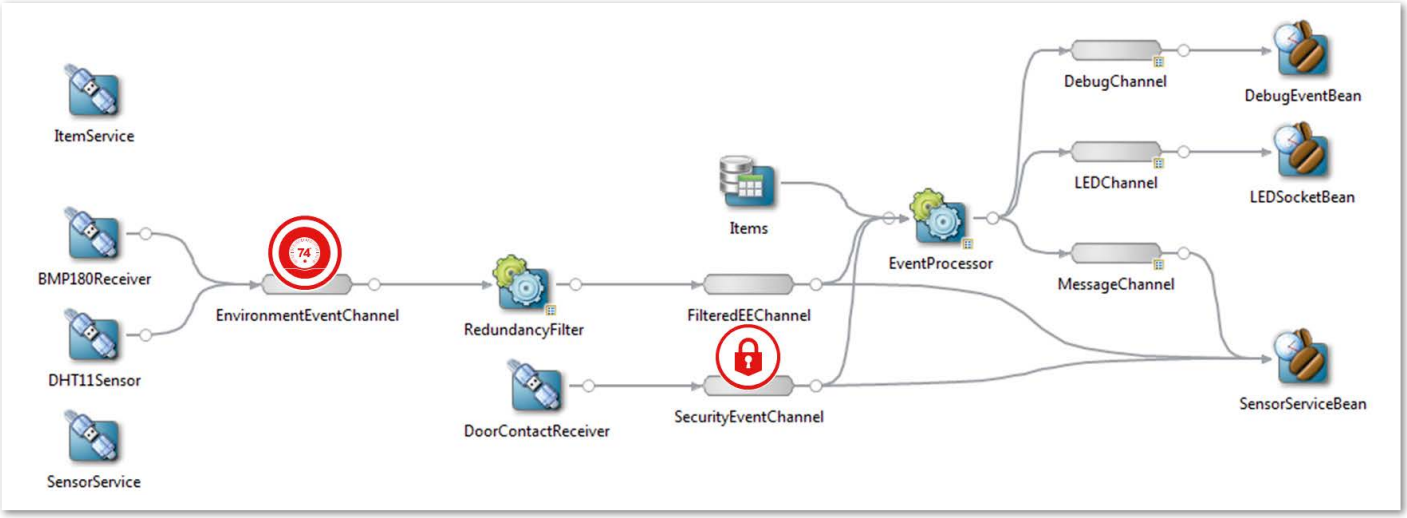


Abbildung 2: Ereignisverarbeitendes Netzwerk in Oracle Event Processing

pischer Einsatzbereich von Event Processing ist die Filterung und Vorverdichtung von Gerätedaten. Sie sorgt dafür, dass nur wirklich relevante Daten im Bedarfsfall über sichere Transportkanäle an zentrale Systeme übertragen werden. *Abbildung 2* zeigt ein in Oracle Event Processing modelliertes, ereignisverarbeitendes Netzwerk für ein Anwendungsbeispiel aus der Transport-Logistik.

Adapter empfangen die Sensordatenströme verschiedener Sensoren, die mit dem Gateway in einem Transport-Container verbaut sind. Temperatur, Luftfeuchtigkeit und Türschließkontakte werden kontinuierlich überwacht und mit einer – in einer lokalen Java-Datenbank abgelegten – Beladungsliste abgeglichen. Bei Zuständen, die auf Verderb der Ware durch Überschreiten der Temperatur-Schwellwerte oder auf einen möglichen Diebstahl bei unerlaubtem Öffnen der Türen hindeuten, werden in Echtzeit Alarme und Benachrichtigungen ausgelöst, die an ein Mobilgerät oder die zentrale Unternehmens-IT für die Weiterverarbeitung gesendet werden *(siehe Abbildung 3)*.

Die Demonstration eines Prototyps, der Java ME, Java Embedded und OEP in Kombination nutzt, ist in YouTube eingestellt *(siehe „https://www.youtube.com/watch?v=WBmZnOoRWhI“)*.

Abbildung 3: Demo einer sensorgetriebenen Transportüberwachung

Netzwerk und Sicherheit

Weitverbreitete Sensoren und Geräte bilden ein IoT-Netzwerk. Am zentralen Eingangspunkt für die Datenströme dieser Geräte, etwa in der DMZ eines Rechenzentrums, ist ein Security-Gateway wie das Oracle API Gateway erforderlich. Es dient der gesicherten Kommunikation über potenziell unsichere IP-Netze wie das Internet und ist vollständig transparent für die Anwendungen. Dafür kann ein sicherer Tunnel eingerichtet oder ein sicherer Endpunkt bereitgestellt werden. Zwei wichtige Aufgaben werden vom Gateway übernommen: die Schnittstellen-Adaption, also die Bereitstellung von Zugangspunkten für verschiedene Protokolle (SOAP, REST) und Datenformate (XML, JSON), sowie die Sicherheitskonfiguration. Die Grundfunktionalitäten in Sachen Sicherheit sind die Authentifizierung (Wer?), die Autorisierung (Was?), die Sicherstellung von Nachrichtenintegrität (Woher? Unverändert?) und die Verschlüsselung (Vertraulichkeit). Das ist vergleichbar mit der Bordkartenkontrolle an einem Flughafen, bei der sichergestellt wird, dass die richtige Person ins richtige Flugzeug steigt.

Das Oracle API Gateway bietet aber noch weitere Funktionalitäten. Entsprechend einer Sicherheitskontrolle am Flughafen, bei der Gepäck und Person eingehend durchsucht werden, kann es in den Nachrichten nach bedrohlichen Inhalten suchen. Angreifer versuchen typischerweise, Schadcode in die übermittelten Daten einzubauen (Viren), durch das Variieren der Anfrage-Parameter mehr Daten zu erhalten ((SQL-)Injection) als vorgesehen oder den Dienst oder Server durch gezieltes Überlasten außer Funktion zu setzen (Denial of Service (DOS), XML Bombs).

Man schützt sich vor Angreifern, indem man sicherstellt, dass ein bereitgestellter Dienst genauso wie vorgesehen genutzt wird. Noch vor der Weiterleitung an den eigentlichen Dienst sucht das Gateway nach Viren, injizierten SQL-Statements, überprüft Parameter-Signaturen auf ihre Gültigkeit, schränkt Größe und Komplexität von Nachrichten ein, gleicht Absender-Adressen mit Black- und White-Listen ab und schränkt die erlaubte Nutzungshäufigkeit in Abhängigkeit des Aufrufers (Throttling) ein.

Serverseitige Verarbeitung von IoT-Daten

Milliarden von Menschen interagieren heute im Internet. Die elektronische Ausstattung und Vernetzung von Dingen wird die Zahl der Sensoren und Aktoren im Internet der Dinge beträchtlich erhöhen. Eine wahre Flut von Daten ist zu beherrschen. Für das Daten-Management bieten sich heute vier Varianten an, die in beliebiger Kombination nutzbar sind:

- *Relationale Datenbanken*
  Die klassische relationale Oracle-Datenbank stellt die komfortabelste und flexibelste Möglichkeit zur Datenverwaltung dar. Sie eignet sich für die Speicherung von strukturierten und unstrukturierten Daten und für alle Last-Szenarien, von transaktionaler bis zu In-Memory-Verarbeitung.
- *Data Warehouse*
  Im Data Warehouse werden die Daten nach bestimmten Dimensionen voraggregiert und somit auswertungsorientiert vorgehalten. Das spart Speicherplatz und beschleunigt das Abfragen der Daten für bekannte Fragestellun-

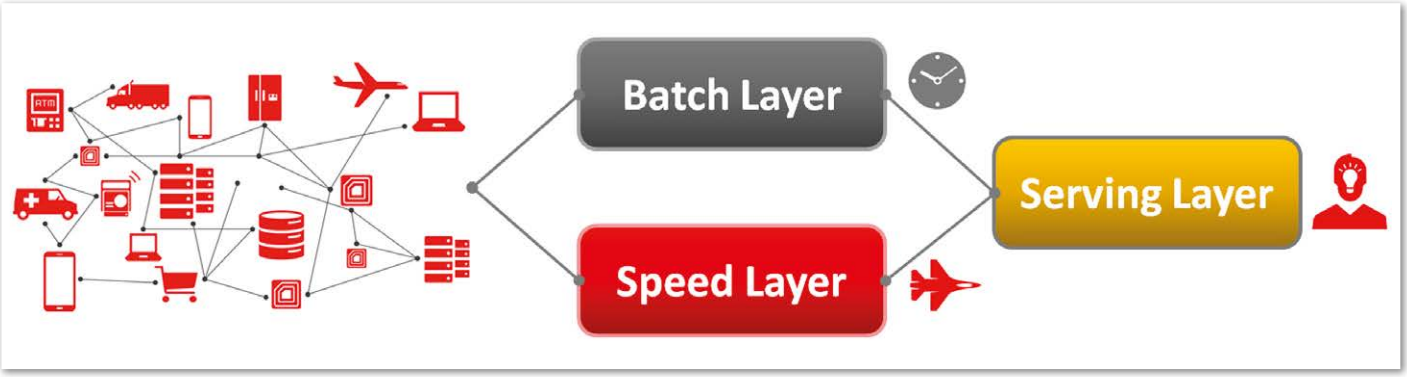


Abbildung 4: Lambda-Architektur

gen. Die Daten liegen allerdings nicht mehr in Rohform vor. Neue Fragestellungen bedürfen gegebenenfalls einer neuen Modellierung der Daten.

- *NoSQL-Datenbanken*
  Hier werden die Daten als Anwendungsobjekte in Schlüssel-Wert-Paaren vorgehalten. Auf die Objekte wird von Anwendungen in direkt konsumierbarer beziehungsweise instanziierbarer Form zugegriffen.
- *Verteilte Dateisysteme (Big Data/Hadoop)*
  Ist der Wert der zu verwaltenden Daten eher unbestimmt und sollen Daten in ihrer Rohform einschließlich aller Redundanzen gespeichert werden, greift man auf kostengünstige, Datei-basierte Verfahren zurück. Hier werden die Daten unstrukturiert und in großen Blöcken (Hunderte MB) gespeichert. Dadurch lassen sich zu späteren Zeitpunkten beliebige, auch bei der Erfassung der Daten völlig unbekannte Fragestellungen beantworten. Dafür müssen aber verteilte und hochparallele Verarbeitungsprozesse (Map/Reduce) über die Datenblöcke laufen, die in kaskadierenden Batchläufen die unstrukturierten Daten interpretieren, sortieren und verdichten, bis eine Ergebnismenge als Liste von Schlüssel-Wert-Paaren ermittelt ist.Alle genannten Verfahren zur Datenverwaltung haben ihre spezifischen Vor- und Nachteile, weshalb sie oft in Kombination angewendet werden. Sie teilen den Nachteil, dass eine Auswertung der Daten in Echtzeit nicht möglich ist. Im Internet der Dinge verlieren Informationen jedoch schnell an Wert: Überhitzt eine Maschine, muss sie sofort abgeschaltet werden, bevor ein Lagerschaden zu größeren Beschädigungen führt. Braucht die nachgelagerte Daten-Analyse in Form eines Batchlaufs zu lange, lässt sich der Schaden nicht verhindern. Für Reaktionen in Echtzeit sind in IoT-Architekturen daher parallele Verarbeitungsprozesse vorgesehen.

Lambda-Architektur für parallele Echtzeitverarbeitung

Bei der Lambda-Architektur führt man parallel zu einem Batch Layer, der alle verfügbaren Daten in Form von Batch-Prozessen verarbeitet, einen Speed Layer ein. Dieser füllt durch eine Echtzeitverarbeitung der Eingangs-Datenströme die zeitliche Lücke, die der Batch Layer hinterlässt *(siehe Abbildung 4)*.

Für die Verarbeitung hochvolumiger IoT-Eingangsdatenströme eignet sich der Oracle Stream Explorer. Er ermöglicht die Korrelation, Filterung und Verarbeitung von Datenströmen auch ohne IT-Kenntnisse, beispielsweise durch Fachabteilungen. Immer ausgehend von einer Vorschau des Live-Datenstroms, können so Muster in der Abfolge der Ereignisse eines Datenstroms erkannt werden *(siehe Abbildung 5)*.

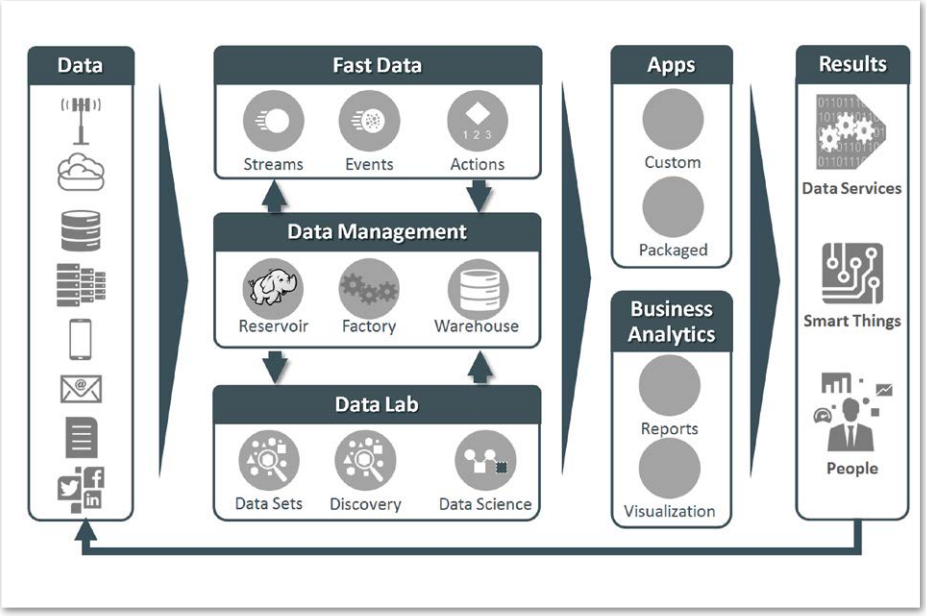


Abbildung 6: Oracle-Big-Data-Plattform

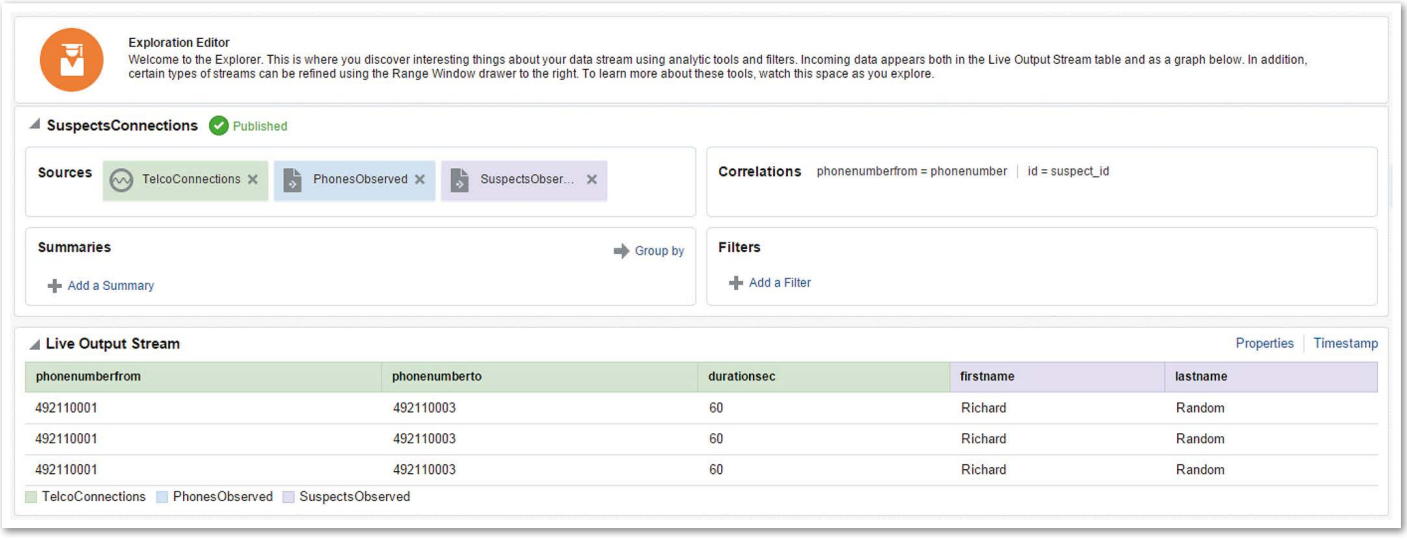


Abbildung 5: Stream Explorer, Analyse ausgehend vom Live-Datenstrom

Abbildung 7: Die Oracle-IoT-Cloud

Die Besonderheit ist, dass auch die zeitliche Beziehung zwischen Einzel-Ereignissen in die Analyse einfließen kann. Damit ist es möglich, weit komplexere Mustererkennungen anzuwenden als reine Schwellwert-Betrachtungen: Trends (kontinuierliche Auf- und Abwärtsbewegungen), Fluktuationen (Abweichungen vom Mittel), ausbleibende Ereignisse (Nicht-Ereignisse) oder „W"- und „M"-Formationen werden erkannt und Duplikate herausgefiltert.

Ein nachfolgender Serving Layer stellt sicher, dass es einen einheitlichen Zugang zu Daten und Analysen gibt. Neue Funktionalitäten der Oracle-Datenbank wie Big Data SQL ermöglichen, verteilte Hadoop-Filesysteme (HDFS) als externe Tabellen in der Oracle-Datenbank zu registrieren und per gewohnter SQL darauf zuzugreifen. Datenbank-Abfragen können sich über relational und blockweise gespeicherte Daten hinweg erstrecken. Spezialkenntnisse, wie die Programmierung verteilter Map/Reduce-Verarbeitungsprozesse in Java, sind nicht nötig.

Oracle Big Data Discovery geht den nächsten Schritt hin zu einfachen Daten-Analysen: Es bietet eine intuitive und einfache Benutzerschnittstelle für Hadoop. Daten aus Hadoop-Systemen können katalogisiert und grafisch analysiert werden. Durch die interaktiven Visualisierungen der Daten werden dabei immer neue Perspektiven eingenommen, um Muster in den Daten auch für Nicht-Mathematiker und -Statistiker erkennbar zu machen.

Das sich daraus ergebende Gesamtbild zeigt, dass die Lambda-Architektur in völligem Einklang mit der Oracle-Big-Data-Plattform ist *(siehe Abbildung 6)*. Der Batch Layer entspricht dem Block der Data-Management-Werkzeuge, von der Datenbank über Warehouse bis hin zu Hadoop. Der Speed Layer spiegelt die Fast-Data-Kategorie mit Stream Explorer wider und der Serving Layer wird durch das Data Lab mit Big Data Discovery repräsentiert.

Oracle-IoT-Cloud-Service

In einer aktuellen Umfrage *[2]* unter hundert Unternehmen zu den größten Frustrationspunkten in IoT-Projekten wurden neben einer generellen Verunsicherung um den entstehenden Hype vor allem Implementierungsschwierigkeiten, Datensicherheit, fehlende Standards und hohe Kosten als Hemmnis genannt. All diese Punkte adressiert der ab Sommer 2015 verfügbare Oracle-IoT-Cloud-Service *[3]*. Dieser wird von der Daten-Akquirierung über die Daten-Analyse bis zur Integration der Geschäftsprozesse und Unternehmens-Anwendungen alle benötigten Funktionen schnell, flexibel und bei deutlich reduziertem Risiko als Plattformdienst in der Oracle-Cloud bereitstellen. Die Schichten übernehmen dabei folgende Aufgaben *(siehe Abbildung 7)*:

- *Daten-Akquirierung*
  Für die Anbindung, Kontrolle und Verwaltung von IoT-Geräten stehen die IoT-Cloud-Service-SDKs und -APIs für verschiedene Programmier-Umgebungen und das IoT-Cloud-Service-Gateway zur Verfügung. Sie sorgen für eine sichere und vertrauenswürdige bidirektionale Kommunikation der Geräte mit der Cloud.
- *Daten-Analyse*
  Die eingehenden Datenströme können, in Echtzeit aggregiert, gefiltert, miteinander korreliert und analysiert werden. Wie vom Oracle Stream Explorer bekannt, können auch prädiktive Analysen und Trend-Erkennungen ausgeführt werden, die regelbasiert Alarme und Benachrichtigungen auslösen. Zudem stehen mit Big Data für IoT und dem Oracle Business Intelligence Cloud Service Lösungen für die Abfrage, Analyse und Visualisierung von IoT-Massendaten zur Verfügung.
- *Integration*
  Natürlich können vorhandene Unternehmens-Applikationen dynamisch mit Daten aus der IoT-Cloud versorgt werden, um Prozesslücken zu schließen und schnell auf kritische Ereignisse reagieren zu können.

Fazit

Mit Oracle-Standard-Produkten lässt sich eine komplette, durch die Nutzung von Java offene und zukunftssichere sowie in allen Schichten belastbare IoT-Architektur abbilden. Wenn man keine eigene IoT-Infrastruktur aufbauen möchte oder langfristige Investitionen in ein neuartiges Geschäftsumfeld scheut, bietet die Oracle IoT Cloud ab diesem Sommer die Möglichkeit, schnell und risikofrei eigene IoT-Anwendungen zu starten.

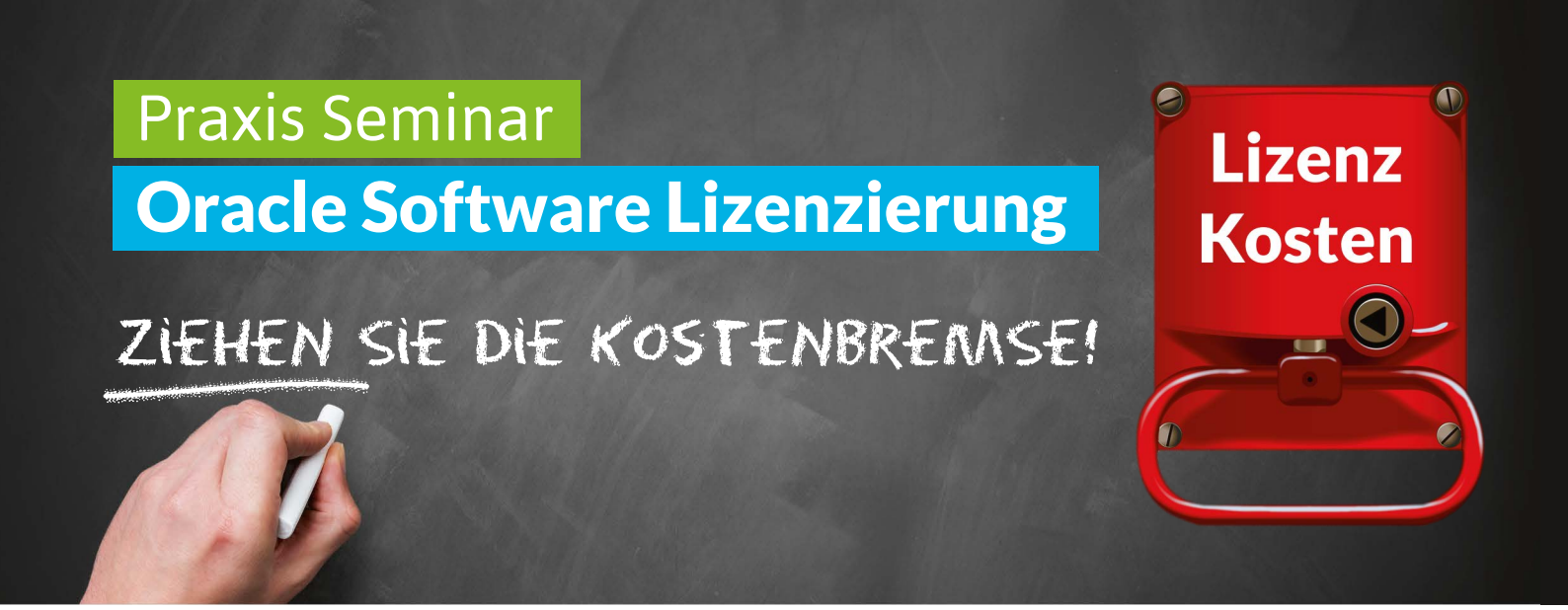
Weitere Informationen

[1] JD Edwards EnterpriseOne Internet of Things Platform, Oracle White Paper, 2015
[2] https://www.linkedin.com/grp/post/108418-6016658510509592578
[3] https://cloud.oracle.com/iot

Marcel Amende
marcel.amende@oracle.com

Praxis Seminar

Oracle Software Lizenzierung



In diesem Seminar vermitteln wir unser detailliertes Expertenwissen über die optimale Lizenzierung von Oracle Datenbank Software. Sie lernen zahlreiche Fehlerquellen zu erkennen und zu vermeiden, sowie Einsparungspotenziale im Bereich der Lizenzierung zu identifizieren und umzusetzen.

Endlich Klarheit bei allen Fragen zur Oracle Datenbank Lizenzierung

Teure Fehler vermeiden **Lizenzkosten optimieren** **Klare Antworten** **Zahlreiche Praxisbeispiele**

Seminar Inhalte

- Oracle Infrastrukturkonzepte und deren Lizenzierung
- Oracle Datenbanken in virtualisierten Umgebungen lizenzieren
- Oracle Database Special-/Restricted- Use Lizenzen einsetzen
- Auswirkungen von Oracle Enterprise Management Packs
- Lizenzierung von Test- und Entwicklungssystemen
- Teure Fehler und Lizenzstolpersteine erkennen und beheben
- Aufräumen von Mythen, Legenden und falschen Auslegungen
- Wie man sich auf ein Oracle Lizenz Audit richtig vorbereiten

und viele, viele weitere Themen ...

Termine:

29. September, Wien
01. Oktober, München
06. Oktober, Hamburg

Vortragender
Ing. Klaus-Michael Hatzinger
CEO DBConcepts GesmbH.

Präsident Austrian Oracle User Group (AOUG)
Member der DOAG Arbeitsgruppe Lizenzierung
Herausgeber "Oracle Quick Pocket Licence Guide"
Author der Email Reihe: „Oracle Lizenzierung - Teure Fehler vermeiden."

Informationen & Anmeldung:

http://www.dbconcepts.at/seminar

DBCONCEPTS
Die Oracle Experten

DBConcepts GesmbH. Lassallestraße 7a, 1020 Wien Tel: +43 1 8908999-0 office@dbconcepts.at

Teilnehmer/innen Feedback:

„Sehr gut, knappe und klare Informationsweiterleitung. Sehr kompetenter Trainer!"

„Die Veranstaltung hat mir sehr gut gefallen! Der Vortragende war sehr fachkundig und erfahren."

„Sehr gute Veranstaltung, sicherlich einzigartig! Sonst erfährt man die Lizenzprobleme nie in einer derart kompakten Form."

„Das Wissen von DBConcepts und die praktische Erfahrung wurden sehr gut vermittelt. Die ausgehändigten Unterlagen sind ausgezeichnet."

„Es wurde extrem verdichtet und dennoch auch für Anfänger das Lizenzierungs Know-how sehr verständlich dargelegt."

Schlanke Architekturen durch smarte Modellierung

Florian Kurz, Marco Mevius und Peter Wiedmann, Konstanzer Institut für Prozesssteuerung (kips), Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)

Die richtige Balance zwischen Flexibilität und Stabilität von Geschäftsprozessen ist ein essenzieller Faktor für Unternehmen, um den Anforderungen zunehmend dynamisch werdender Märkte gerecht zu werden. Die durch die digitale Revolution stetig steigende IT-Durchdringung fordert eine noch besser abgestimmte Zusammenarbeit zwischen den IT- und Fachabteilungen.

Müssen Organisationen schwergewichtige IT- und Prozess-Architekturen pflegen, können diese Anforderungen in vielen Fällen nicht adäquat erfüllt werden. Zu lange Zeitfenster der Aktualisierungszyklen führen zu signifikanten Differenzen zwischen den modellierten und den gelebten Geschäftsprozessen. Die Autoren dieses Beitrags propagieren eine Lösung durch innovative, agile Methoden, mit deren Hilfe es möglich wird, schlanke, maßgeschneiderte Architekturen unmittelbar zu implementieren und kontinuierlich zu pflegen.

Der Artikel zeigt einleitend die Herausforderungen schlanker Architekturen. Darüber hinaus ist beschrieben, warum klassische Methoden diesen Anforderungen nur schwer genügen. Mit einer ganzheitlichen Methode des agilen Geschäftsprozessmanagements (GPM), bestehend aus der Vorgehensweise M.E.M.O., der mobilen Applikation BPM Touch und der Sprache BPMN Easy, werden die Potenziale einer smarten Modellierung für schlanke Architekturen dargestellt.

Schlanke Architekturen

Übergeordnete Zielsetzung des Einsatzes von IT und des strukturierten Designs von Prozess-Architekturen ist es, einen signifikanten Beitrag zum „Business-Value“, also dem langfristigen Unternehmenswert und dessen Steigerung zu generieren *[1]*. Neben dem operational betrieblichen Fokus müssen sowohl IT- als auch Prozess-Architekturen dezidierten Anforderungen etwa an die IT-Sicherheit und die Resilienz (also beispielsweise sich innerhalb kurzer Zeit an neue, ändernde Rahmenbedingungen anpassen zu können) adäquat gerecht werden *[2]*. Insbesondere intransparente Geschäftsprozesse und unberücksichtigte Anforderungen der Fachbereiche sind Gründe für den Wunsch zur Flexibilisierung und Auflösung um fangreicher IT-Anwendungslandschaften – sogenannten „schwergewichtiger Architekturen“.

Bereits nach wenigen Jahren sind in der IT-Infrastruktur und der IT-Anwendungslandschaft eine Vielzahl unterschiedlicher Technologien (Clients, Server, Betriebssysteme, Netzwerke etc.), Werkzeuge und Entwicklungs-Paradigmen zu finden. Daraus resultieren Redundanzen in der Daten-Architektur, der Funktions-Abdeckung und den Schnittstellen zwischen den unterschiedlichen IT-Anwendungen und übergeordneten Geschäftsprozessen. Auf Basis schwergewichtiger Architekturen kann häufig nur mit erheblichem personellen und finanziellen Aufwand auf neue Technologien und Paradigmen (wie Industrie 4.0, Cloud Computing, In Memory) reagiert werden. Eine schnelle und strukturierte Bereitstellung von innovativen IT-Lösungen ist somit unzureichend möglich *[3]*. Um daher einen kontinuierlichen Abgleich der Anforderungen des Fachbereichs unter Berücksichtigung der Möglichkeiten des Aufbaus und der Pflege einer schlanken flexiblen Architektur zu ermöglichen, müssen agile Vorgehen zur Synchronisation von Fachbereich und IT im Unternehmen implementiert werden.

Agiles, anwenderzentriertes Geschäftsprozessmanagement

Klassische Methoden des Geschäftsprozessmanagements orientieren sich am

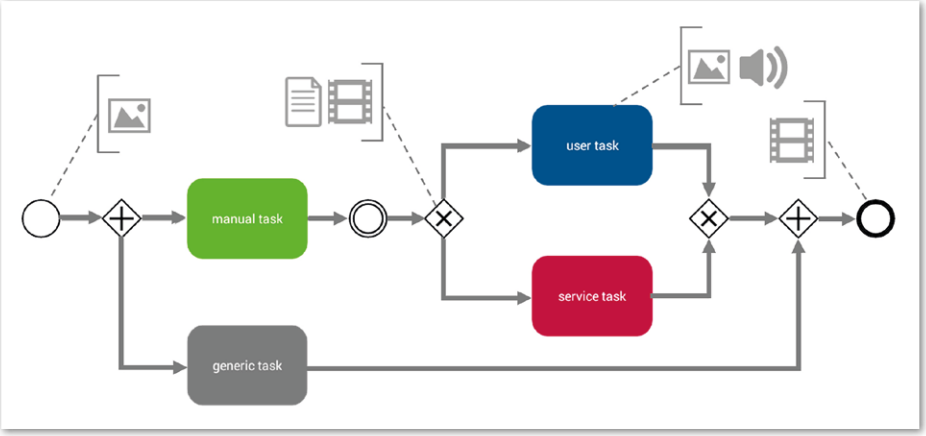


Abbildung 1: Notations-Elemente der Sprache BPMN Easy mit multimedialen Annotationen

wasserfallartigen Vorgehen *[4]*. Voraussetzung für diese Art des sequenziellen Vorgehens sind bekannte und präzise beschriebene Anforderungen der Fachabteilungen an die IT-Anwendungs-Landschaft. Die adäquate Reaktion auf kurzfristig identifizierte Änderungen oder nicht präzise formulierte Anforderungen ist häufig nur mit großem Aufwand möglich. Durch die Übertragung agiler Prinzipien (hohe Interaktion aller Beteiligten, kurze Iterationszyklen etc.) auf das Geschäftsprozessmanagement können diese Herausforderungen gemeistert werden *[4]*.

Iteratives und inkrementelles Vorgehen im GPM ermöglichen es, Problemstellungen und Optimierungspotenziale während der Erfassung, Implementierung und Ausführung von Geschäftsprozessen unmittelbar zu berücksichtigen. Fachanwender können somit auf neuartige Weise direkt in die unterschiedlichen Schritte des Geschäftsprozessmanagements integriert werden *[5]*.

In enger Zusammenarbeit des Konstanzer Instituts für Prozessteuerung (kips) und der bamero AG *[6]* wurde die hier vorgestellte agile Methode entwickelt. Die Methode besteht aus der Sprache BPMN EASY, der agilen Vorgehensweise M.E.M.O. und der Applikation BPM Touch zur intuitiven, Software-unterstützten Geschäftsprozessmodellierung.

BPMN Easy

Bestehende Modellierungssprachen wie BPMN 2.0, EPK oder UML können durch BPM-Experten aufgrund der umfassenden Element-Sets zur Dokumentation von komplexen Sachverhalten unterschiedlicher Detaillierungsgrade eingesetzt werden. Für Fachanwender ohne Vorkenntnisse ist ein intuitiver Zugang zur Geschäftsprozessmodellierung nicht möglich *[7]*. Motiviert aus dieser Problemstellung, reduziert BPMN Easy das Elementset des aktuellen Standards BPMN 2.0.

Das synonymfreie Elementset von BPMN Easy umfasst Ereignisse (Start-, Zwischen- und End-Ereignisse), Gateways (exklusiv und parallel) und Aktivitäten. Erfahrungen aus Anwendungsprojekten *[8]* zeigen, dass diese Elemente für ein erfolgreiches Geschäftsprozessmanagement vollständig ausreichend sind.

Abbildung 1 zeigt einen beispielhaften Geschäftsprozess, modelliert mit BPMN Easy. Die intuitive, visuelle Unterscheidung von Aktivitäten erfolgt durch farbliche Kennzeichnung. Manuelle Aktivitäten sind „grün", teilautomatisierte Aktivitäten „blau" und automatisierte Aktivitäten „rot" gekennzeichnet. Um eine größtmögliche Flexibilität bei der Modellierung zu erreichen, können neben den beschriebenen Aktivitäten auch generische, „grau" gekennzeichnete Aktivitäten modelliert werden.

Im Vergleich zu anderen Modellierungssprachen können Fachanwender durch die farbliche Kennzeichnung der Aktivitäten in BPMN Easy sehr einfach zwischen den Aktivitätstypen unterscheiden und beim Blick auf den gesamten Geschäftsprozess anhand der dominierenden Farbe den Automatisierungsgrad des Geschäftsprozesses erkennen *[7]*. Trotz der Einfachheit und Klarheit erfüllt BPMN Easy die Anforderung der formalen Beschreibbarkeit von Geschäftsprozessmodellierungssprachen nach van der Aalst et al. *[9]* vollständig. Ein signifikanter Mehrwert entsteht durch die Möglichkeit, die Elemente von BPMN Easy mit Mediadateien zu annotieren. Dies ermöglicht eine Anreicherung der Geschäftsprozessmodelle mit Audio-, Video-, Bild- und allgemeinen Dateien.

Aufgrund des intuitiven Elementsets und der Multimedia-Annotation kann ein

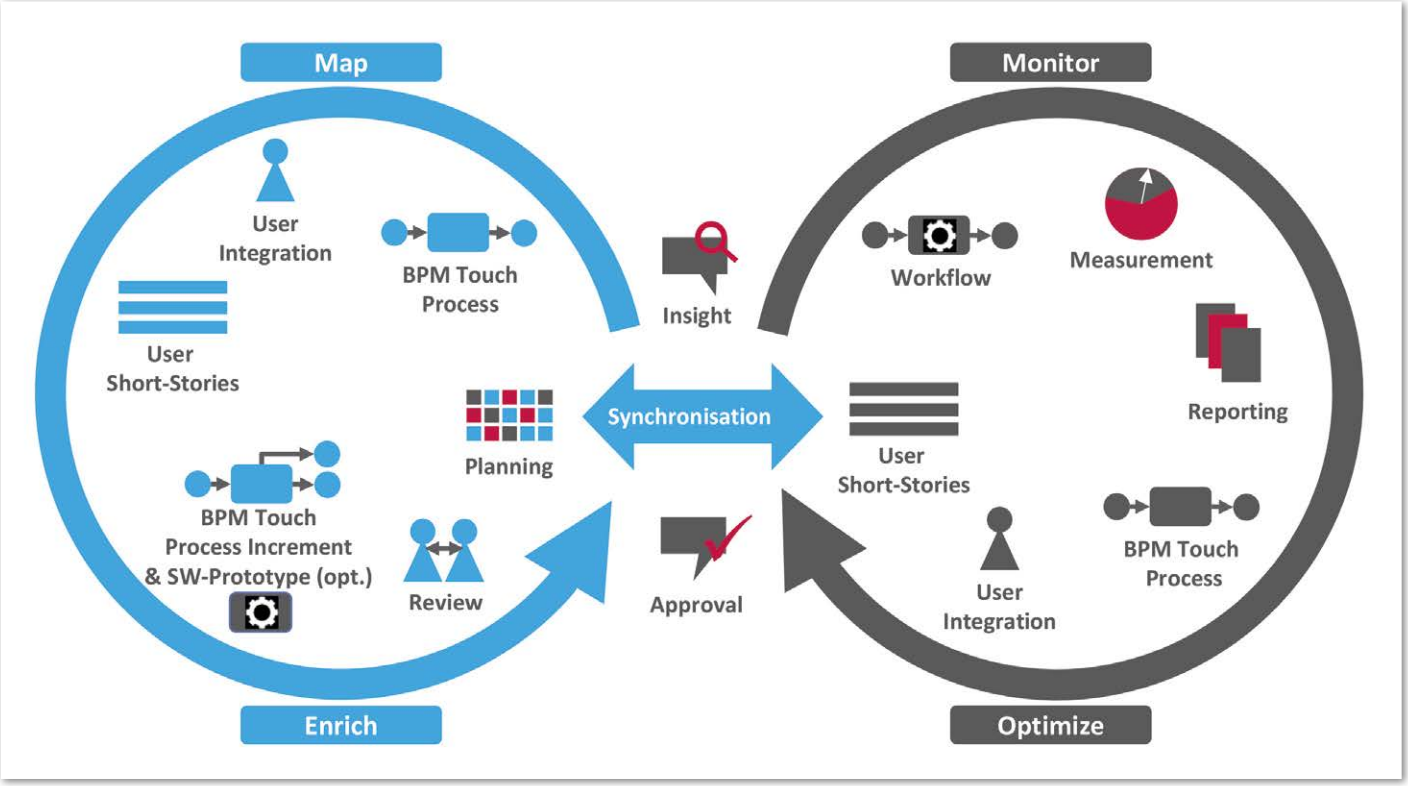


Abbildung 2: Agile Vorgehensweise M.E.M.O.

BPMN-Easy-Modell sowohl von nicht IT-affinen Anwendern als auch von IT-Experten gleichermaßen modelliert, überprüft und analysiert werden *[10, 11]*. Neben der Erhöhung des Verständnisses aller Beteiligten hinsichtlich der Geschäftsprozessmodelle werden durch die kompakte Erfassung und Darstellung von Informationen der Aufbau und die Pflege einer schlanken Prozess-Landschaft unterstützt.

M.E.M.O.

Die in *Abbildung 2* dargestellte Vorgehensweise M.E.M.O erweitert traditionelles Geschäftsprozessmanagement um agile Konzepte in den vier Schritten:

- Map (Process Mapping)
- Enrich (Prozessanreicherung)
- Monitor (Prozessmonitoring)
- Optimize (Prozessoptimierung)

Damit werden Effektivität und Effizienz im Rahmen von Geschäftsprozessmanagementprojekten signifikant erhöht. Die Schritte „Map“ und „Enrich“ bilden den ersten Zyklus der Vorgehensweise M.E.M.O. Während der „Map“-Phase werden Geschäftsprozesse mithilfe der Applikation BPM Touch von den Fachanwendern modelliert.

Durch die Möglichkeit der multimedialen Modellierung können Anforderungen, Verbesserungsideen oder Ausnahmen direkt und gebrauchssprachlich von Fachanwendern an die entsprechenden Notationselemente (z.B. annotierte BPMN-Easy-Aktivität) in Form von sogenannten „Kurzgeschichten“ angehängt werden. Kurzgeschichten werden (etwa durch eine Beschreibung in Form von maximal zwei Sätzen oder in 30-Sekunden Statements) multimedial im Fachterminus der Anwender gebrauchssprachlich dokumentiert. Mit der präzisen multimedialen Aufzeichnung von Anforderungen, Problemstellungen und Optimierungspotenzial durch die Fachanwender wird eine neuartige, strukturierte Wissensdokumentation ermöglicht.

Der „Enrich“-Schritt bietet die Möglichkeit, fachliche Modelle in Hinblick auf die spätere (automatisierte) Ausführung beziehungsweise die spätere Verwendung des Geschäftsprozessmodells zu erweitern. Im Zuge dessen werden die fachlichen Geschäftsprozessmodelle der ersten Phase beispielsweise um technische, zur Ausführung benötigte Details angereichert. In weiteren Iterationen wird die „Enrich“-Phase genutzt, um Benutzeroberflächen der im Geschäftsprozess eingesetzten Software oder des Workflow-Systems in enger Zusammenarbeit mit den Fachanwendern zu erstellen und zu optimieren.

Die erarbeiteten und angereicherten Geschäftsprozessmodelle werden in jeder Iteration mit allen prozessbeteiligten Stakeholdern (GPM- und IT-Experten, Anwender) synchronisiert. Mit der kontinuierlichen Synchronisation in Abnahmetreffen wird eine bestmögliche Integration der Fachanwender und der IT-Experten in das GPM erreicht. Während der Abnahmetreffen wird geprüft, ob die selektierten Kurzgeschichten korrespondierend zu den definierten Projektzielen abgearbeitet wurden.

Die Schritte „Monitor“ und „Optimize“ dienen im zweiten Abschnitt des Zyklus der kontinuierlichen Analyse und Optimierung der operativ ausgeführten Geschäftsprozesse. Die fachlich modellierten und technisch angereicherten Geschäftsprozessmodelle können auf Basis des Standards BPMN 2.0 direkt und medienbruchfrei in Workflow-Systeme übertragen werden.

Fachanwender haben die Möglichkeit, Informationen und Hinweise hinsichtlich der Kennzahlen des Geschäftsprozesses bereits während der fachlichen Modellierung in den Kurzgeschichten zu dokumentieren. Mit der agilen BPM-Vorgehensweise M.E.M.O. wird ein kontinuierliches Alignement zwischen Fachbereich und IT auf integrative Art und Weise ermöglicht. Die gebrauchssprachliche Dokumentation in Form von Bildern, Audio- und Videoaufnahmen sowie Dokumenten direkt im Modell des Geschäftsprozesses befähigt Unternehmen zu einer neuen, innovativen Strukturierung internen Wissens.

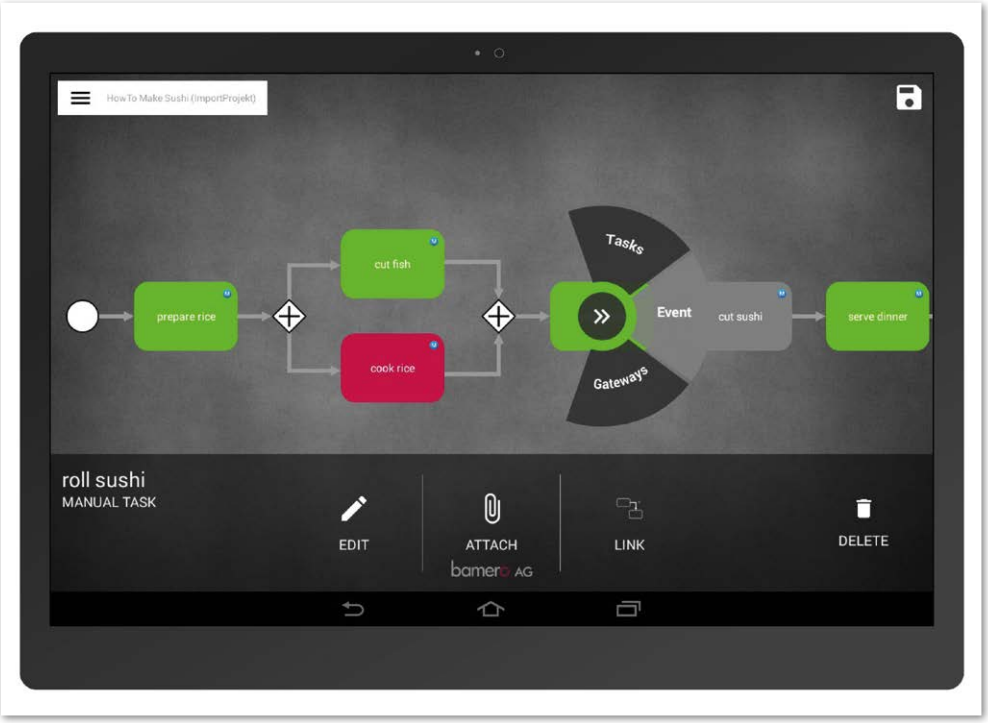


Abbildung 3: Mobile Applikation BPM Touch

BPM Touch

Software-Werkzeuge dienen der Unterstützung des Geschäftsprozessmanagements, wodurch beispielsweise die Modellierung und digitale Verwaltung von Geschäftsprozessmodellen ermöglicht wird. Mit der Applikation BPM Touch der bamero AG können Geschäftsprozesse intuitiv von Fachanwendern sowie IT- und BPM-Experten in der Sprache BPMN Easy modelliert werden. Native Geräte-Funktionen wie Kamera und Mikrofon mobiler Endgeräte (Tablets) ermöglichen es, Bilder, Videos und Audioaufnahmen

der zu beschreibenden Sachverhalte an einzelne Elemente anzuhängen.

In Kombination mit intuitiven Touch-Gesten für die Modellierung, wie sie in *Abbildung 3* dargestellt sind, wird mit der gebrauchssprachlichen Annotation der grafischen Modellierungssprache ein einfacher Einstieg in die Geschäftsprozessmodellierung sowohl für Laien als auch für Experten gewährleistet. Neben standardisierten technischen Exportformaten wie BPMN 2.0 können alle modellierten Geschäftsprozesse als Bilddateien oder gepackt inklusive der annotierten Multimediadateien exportiert werden.

Eine innovative Möglichkeit der detaillierten Geschäftsprozessanalyse ist beispielsweise mithilfe eines Microsoft-PowerPoint-Exports ermöglicht. Mit der Transformation der Geschäftsprozessmodelle in unterschiedliche Sichtweisen inklusive aller multimedialen Annotationen in eines der weitverbreitetsten Office-Formate ermöglicht BPM Touch eine unternehmensweite Geschäftsprozesssynchronisation und -optimierung über die Grenzen der Fachabteilung hinweg. Zudem zeigt die mobile Business-Applikation BPM Touch mit kollaborativen Features zur gemeinsamen Modellierung, wie sich Social-BPM-Aspekte direkt in die Fach- und IT-Abteilungen integrieren lassen und damit ein unternehmensweiter „BPM Spirit“ möglich ist.

Smarte Modellierung

Der Begriff „Smarte Modellierung” beschreibt den schnellen und intuitiven Zugang aller GPM-Beteiligten zur Geschäftsprozessmodellierung. Das Akronym „smart“ repräsentiert hierbei die Kernziele der ganzheitlichen Methode. Fokussiert werden mit der Methode die Eigenschaften:

- **S**peed (beschleunigtes Modellieren)
- **M**ultimedial (Gebrauchssprachlich annotierte Modellierung)
- **A**gile (agile, anwenderzentrierte Vorgehensweise)
- **R**eal (real gelebte Geschäftsprozessmodelle)
- **T**ransferable (medienbruchfreie Weiterverarbeitung)

Die Tandem-Modellierung von Fachanwendern und (IT-)Experten während der Erfassung von Geschäftsprozessmodellen reduziert den benötigten zeitlichen Aufwand signifikant („Speed“). Alle Notationselemente der Sprache BPMN Easy können mit Multimediadateien annotiert werden. Durch die Annotation wird der Geschäftsprozess mit wichtigen Informationen in Bildern, Audio- und Video-Aufnahmen sowie Dokumenten angereichert. Fachanwender können mithilfe der

multimediabasierten Annotation komplexe Sachverhalte ohne aufwändige Schulung in ihrem eigenen Fach-Terminus im Geschäftsprozessmodell dokumentieren („Multimedial").

Die zugrunde liegende agile Vorgehensweise M.E.M.O. ermöglicht es, Geschäftsprozessmodelle iterativ und inkrementell zu erarbeiten. Im Fokus steht der Mensch, weshalb der Synchronisation aller Beteiligten eine zentrale Rolle zukommt („Agile"). Weiterer Vorteil der multimedialen Annotation von Geschäftsprozessmodellen mit gebrauchssprachlich formulierten Sachverhalten ist die Sicherung und Steigerung der Qualität der Geschäftsprozesse *[12]*.

Ein intuitiver Zugang der Fachanwender zur Geschäftsprozessmodellierung ermöglicht es, Geschäftsprozesse über die Erfassung hinaus aktuell an den realen Anforderungen auszurichten, und prägt eine prozessorientierte Denkweise in den einzelnen Fachabteilungen („Real"). Zudem können alle Geschäftsprozessmodelle medienbruchfrei in die Anwendungen der Oracle Business Process Management Suite oder in andere Fremdsysteme übertragen werden („Transferable").

Ermöglicht wird die medienbruchfreie Übertragung der Geschäftsprozessmodelle dadurch, dass die Notation der Sprache BPMN Easy aus einem stark reduzierten Elementset des Standards BPMN 2.0 besteht *[11]*. Aus der Transformation der fachlichen Geschäftsprozessmodelle in beispielsweise Workflow-Systeme werden fachliche Anforderungen direkt in die operativ (teil-)automatisierten, ausgeführten Geschäftsprozesse übertragen.

Mithilfe einer smarten Modellierung wird gewährleistet, dass Architekturen in der IT-und Geschäftsprozesslandschaft kontinuierlich an die Anforderungen der Fachbereiche angepasst werden, ohne dabei schwergewichtige, unflexible Strukturen aufzubauen. Zudem bietet die regelmäßige Synchronisation zwischen Fachbereichen und IT auf Basis der operativen Geschäftsprozesse einen idealen Anhaltspunkt, um neue, innovative Technologien aus der IT heraus in die Fachbereiche zu integrieren.

Fazit

Schlanke Architekturen bedingen permanente Pflege. Analog zum menschlichen Körper „bleibt nur fit, wer sich fit hält". Die vorgestellte Methode – bestehend aus der intuitiven Sprache zur Geschäftsprozessmodellierung BPMN Easy, der agilen BPM-Vorgehensweise M.E.M.O. und der mobilen Applikation BPM Touch – ermöglicht ein anwenderorientiertes Alignment zwischen Fachbereich und IT.

Die kontinuierliche Anwender-Integration in kurzen Iterationen und eine intuitive Geschäftsprozessmodellierung erlauben es, Anforderungen und Optimierungspotenzial direkt und strukturiert zu dokumentieren. Schlanke Architekturen sind durch reduzierten Overhead (etwa ohne redundante Funktionen und Daten), Anpassungsfähigkeit und Resilienz gekennzeichnet. Das smarte Modellieren von Geschäftsprozessen befähigt Unternehmen, ihre Architekturen an den gelebten Geschäftsprozessen auszurichten.

Weitere Informationen

[1] Bieberstein, N.: Service-oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap. IBM Press, 2006

[2] Kaisler, S. H.; Armour, F.; Valivullah, M.: Enterprise Architecting: Critical Problems: 38th Annual Hawaii International Conference on System Sciences, 2005; S. 224b

[3] Zimmermann, A. et al.: Enterprise Architecture Management for the Internet of Things. In (Zimmermann, A.; Rossmann, A. Hrsg.): GI LNI Proceedings Band 244 Digital Enterprise Computing (DEC 2015), 2015

[4] Rosing, M. von; Scheel, J. von; Gill, A. Q.: Applying Agile Principles to BPM: The Complete Business Process Handbook. Elsevier, 2015; S. 553–577

[5] Serrador, P.; Pinto, J. K.: Does Agile work? — A quantitative analysis of agile project success. In International Journal of Project Management, 2015, 33; S. 1040–1051

[6] bamero AG. http://www.bamero.de, 29.06.2015

[7] Genon, N.; Heymans, P.; Amyot, D.: Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In (Hutchison, D. et al. Hrsg.): Software Language Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011; S. 377–396

[8] Brucker-Kley, E.; Kykalova, D. Hrsg.: Business Process Management (BPM) Studie 2015. Prozessintelligenz - Anwendungsszenarien, Methoden und Technologien, 2015

[9] van der Aalst,Wil M. P.; ter Hofstede,Arthur H. M.; Weske, M.: Business Process Management: A Survey. In (Goos, G. et al. Hrsg.): Business Process Management. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003; S. 1–12

[10] Mevius, M.; Ortner, E.; Wiedmann, P.: Gebrauchssprachliche Modellierung als Grundlage für agiles Geschäftsprozessmanagement. In (Fill, H.-G. et al. Hrsg.): GI-Edition Proceedings Band 225 - Modellierung 2014 -. 19.-21. März 2014 in Wien. Köllen, Bonn, 2014; S. 169–184

[11] Mevius, M.; Wiedmann, P.; Kurz, F.: Nutzerorientierte Multimedia-Geschäftsprozessmodelle als Basis der Serviceorchestrierung. In (Schmietendorf, A.; Simon, F. Hrsg.): BSOA/BCloud 2014. 9. Workshop Bewertungsaspekte service- und cloudbasierter Architekturen, 04. November 2014, Frankfurt/Main. Shaker, Herzogenrath, 2014; S. 49–62

[12] Mevius, M. et al.: Ausgesprochen hochwertig: Hybride Qualitätskontrolle in agilem BPM. In OBJEKTSpektrum, 2014, 05; S. 66–71

M.Sc. Peter Wiedmann
peter.wiedmann@htwg-konstanz.de

B.Sc. Florian Kurz
florian.kurz@htwg-konstanz.de

Prof. Dr. Marco Mevius
marco.mevius@htwg-konstanz.de

Gibt es eine Alternative zur Exadata?

Manfred Drozd, Benchware AG

Oracle hat im Januar 2015 die neue Exadata-X5-Generation eingeführt. Auffällig sind sehr viele Detailverbesserungen, die das Einsatzgebiet der Exadata erweitern. In diesem Artikel werden einige Neuerungen aus neutraler Sicht beschrieben und mit den Angeboten anderer Hersteller verglichen.

Der Inhalt des Artikels wurde bereits am DOAG Exaday 2015 in Frankfurt präsentiert und steht bei der DOAG für Mitglieder zum Download zur Verfügung. In dieser Präsentation sind auch konkrete Performance-Zahlen verschiedener Systeme genannt.

Oracle VM

Exadata unterstützt nun die Oracle-eigene Virtualisierungs-Lösung Oracle VM. Sie ist in erster Linie für die Trennung von Netzwerk-Zonen, Service-Klassen (Produktion, Test), Lastprofile (OLTP, DWH) und Applikationen (SAP, Siebel etc.) gedacht. Auch für eine Lizenzierung von Oracle-Enterprise-Editionen-Optionen auf einem reduzierten Bestand von Cores ist diese Technologie bestens geeignet.

Wo früher separate physische Server notwendig waren, kann diese Trennung nun mit virtuellen Servern erfolgen. Dadurch kann die Anzahl der notwendigen physischen Datenbank-Server reduziert werden. Nur IBM bietet mit seinen POWER/AIX-Systemen eine entsprechende Virtualisierungs-Technologie an. Die von Hitachi Data Systems (HDS) angebotene Hardware-Virtualisierung für x86-Systeme wird vom Oracle-Lizenzmodell nicht akzeptiert.

Capacity on Demand

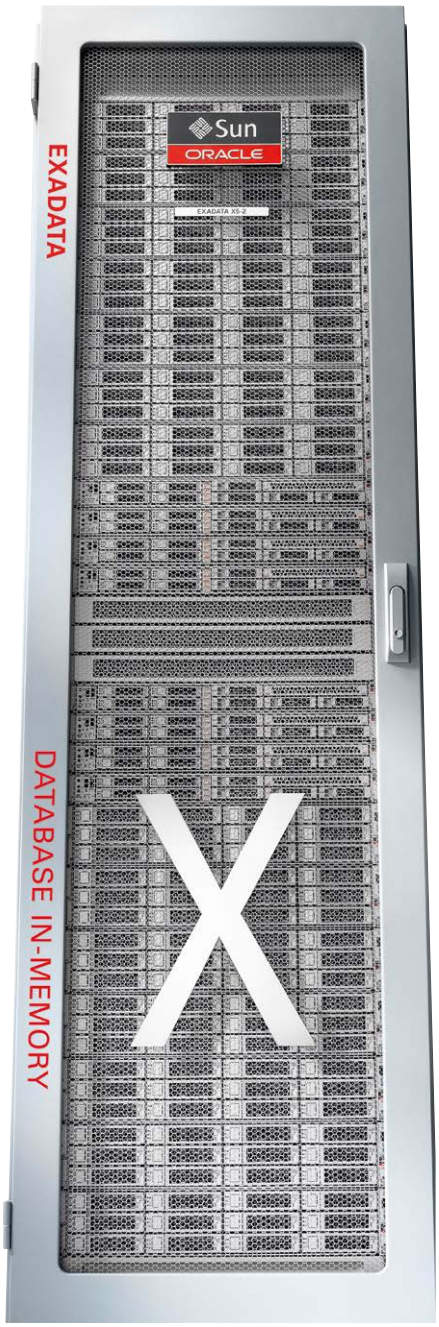
Der Trend in der IT-Industrie geht zu Prozessoren mit immer mehr Cores pro Socket. Dies hat eine dramatische Auswirkung auf die Oracle-Lizenzkosten. Mit Capacity on Demand (CoD) müssen erstmals nicht alle Cores lizenziert werden, sondern nur ein Teil (mindestens 40 Prozent). Mit dieser Technologie und im Zusammenspiel mit der Oracle-VM-Technologie lassen sich die Oracle-Lizenzkosten feingranular kontrollieren.

Scale-up versus Scale-out

Exadata X5-2 Database Server bieten nur wenige Möglichkeiten für den Ausbau, lediglich die Hauptspeicher-Kapazität kann zwischen 256 und 768 GByte gewählt werden. Für monolithische Applikationen, die mehr als 36 Intel x86-Cores oder eine größere Hauptspeicher-Kapazität benötigen, muss das Modell X4-8 mit acht Sockets, 80 Cores und einer maximalen Hauptspeicher-Kapazität von sechs TByte pro Database Server gewählt werden. Im Gegensatz zu den klassischen RISC-Prozessoren SPARC und POWER ist Intel bis heute allerdings den Nachweis schuldig geblieben, auch bei Servern mit mehr als vier Sockets im Oracle-Betrieb linear zu skalieren. SPARC- und POWER-Prozessoren bieten auch deutlich mehr Hauptspeicher-Kapazität pro Socket (SPARC T7 vier TByte bei zwei Socket-Servern, POWER8 demnächst zwei TByte bei zwei Socket-Servern).

Hohe Hauptspeicher-Kapazitäten garantieren ultimative Oracle-Performance, auch ohne die neue In-Memory-Option. Ein Blockzugriff im Hauptspeicher (100 ns) ist immer noch 5.000 Mal schneller als ein Blockzugriff auf einem Flash-Storage (500 µs). IBM bietet mit der Option Active Memory Expansion (AME) obendrein die Möglichkeit, die Hauptspeicher-Kapazität um bis zu Faktor drei ohne nennenswerte Performance-Einbuße zu erweitern. Gerade für Benutzer der In-Memory-Option ein nicht zu unterschätzender Vorteil.

Ein Schweizer Versicherungsunternehmen hat gerade einen umfangreichen Proof of Concepts mit der Oracle-12.1-



In-Memory-Option abgeschlossen und für typische Workloads Performance-Vorteile von bis zu Faktor dreizehn gegenüber einer konventionellen Oracle-11.2-Installation erreicht. Bei solchen Performance-Verbesserungen rechnen sich die zusätzlichen Lizenzkosten schnell. Die In-Memory-Option verbessert die Performance allerdings nur für Applikationen, die nicht prozedural, sondern stark mengenorientiert arbeiten.

Eine hohe Hauptspeicher-Kapazität ist und bleibt somit die günstigste Technologie, um eine herausragende Oracle-Performance zu erreichen (ein TByte in 32 GByte DIMMs für x86-Server kosten rund 40.000 US-Dollar, für RISC-Prozessoren etwa 70.000 US-Dollar). Für eine Kapazitätsplanung von Exadata-Systemen ist zu beachten, das Exadata Database Server komplett von I/O-Operationen entlastet werden, was dazu führt, dass Exadata Database Server circa 25 Prozent mehr Last übernehmen können als konventionelle x86-Database-Server.

Intel-Prozessoren

Das Unternehmen des Autors hat die Performance der Intel-Prozessoren seit der Exadata V2 (2010) systematisch im Oracle-Betrieb gemessen und musste feststellen, dass sich die Geschwindigkeit der Prozessoren (single thread speed) nur marginal verbessert hat. Dies steht teilweise im Widerspruch zu den publizierten SPEC-Zahlen, wobei SPEC-Algorithmen in C und Fortran geschrieben sind, der Autor hingegen bei den Tests Oracle-Datentypen verwendet.

Andererseits hat sich jedoch der Durchsatz der Datenbank-Server durch die ständige Erhöhung der Anzahl von Cores pro Socket deutlich verbessert. Der Durchsatz pro Core ist hingegen nahezu konstant geblieben. Ganz anders bei den POWER8-Prozessoren von IBM. Dort hat sich der Durchsatz pro Core von POWER7 auf POWER8 in unseren Oracle-Benchmarks mehr als verdoppelt. Ein IBM-Kunde kann sich also freuen, dass er beim Wechsel von POWER7 auf POWER8 die Hälfte seiner Oracle-Lizenzen einspart. Im Vergleich zu x86-Servern der Xeon-V3-Generation benötigen POWER8-Server nur halb so viele Cores, sodass der Unterschied bei den Lizenzkosten (Oracle Core Faktor) gerechtfertigt ist. Die Intel-Xeon-V3-Prozessoren in der Exadata X5-2 verwenden bereits DDR4-Hauptspeicher-Bausteine, was sich sowohl beim Durchsatz als auch bei den SQL-Transaktionszeiten äußerst positiv auswirkt.

Storage-Server

Mit der Exadata X5 werden endlich auch All-Flash-Storage-Server angeboten. Oracle verwendet Intel SSDs, die direkt an einen PCI-Bus angeschlossen sind und über das moderne NVMe-Protokoll angesteuert werden. Gerade bei OLTP-Systemen wirkt sich die All-Flash-Technologie positiv auf die SQL-Transaktionenzeiten aus, sowohl bei lesenden als auch bei schreibenden Transaktionen. Außerdem garantieren nur All-Flash-Storage-Systeme eine prognostizierbare (predictable) Oracle-OLTP-Performance. Hybride-Storage-Systeme mit Auto-Tiering-Verfahren leisten sich da zu viele Ausreißer.

Oracle reduziert mit den Exadata-X5-Systemen den Vorsprung konventioneller OLTP-Systeme (siehe dazu die Benchmark-Ergebnisse des Autors mit IBM Server 822 und IBM Flashsystem 840 oder auch HDS-UCP-Servern mit HDS-HUS-VM- oder HDS-VSP-Storage-Systemen). Für aktive Oracle-Daten empfiehlt sich immer der Einsatz von Flash-Technologie, um Prozessoren höher auszulasten und so die Anzahl von Cores und damit die Oracle-Lizenzkosten zu reduzieren.

Oracle bezeichnet die neuen Storage-Server als „Extreme Flash“ (EF). Sie lösen die High-Performance-Storage-Server ab. EF könnte aber auch für „Extreme Expensive“ stehen, denn Oracle verlangt für einen Storage-Server neben dem Hardware-Preis zusätzliche Software-Lizenzkosten in Höhe von 160.000 US-Dollar (20.000 US-Dollar pro Flash Modul; Listenpreis). Innovative Flash-Hersteller wie Pure Storage, die in jüngster Zeit von Gartner durch eine Platzierung im rechten oberen Quadranten geadelt wurden, bieten neben der guten Performance auch noch standardmäßig und ohne Aufpreis Deduplication, Compression und Encryption an, was bei Oracle alles zusätzlich lizenziert werden muss.

Storage-Software

Das Besondere an Exadata-Systemen ist die Storage-Server-Software. Herausragendes Beispiel ist der Smart Scan. Diese Operation ist vor allem für Data-Warehouse-Applikationen nützlich. Ein Exadata-Full-Rack-System erreicht beim sequenziellen Durchsuchen eine unglaubliche Geschwindigkeit von mehr als 250 GBps beim Einsatz von vierzehn Extreme-Flash-Storage-Servern und immerhin noch 140 GBps bei vierzehn High-Capacity-Storage-Servern. Zu beachten ist allerdings, dass diese Datenmenge durchsucht, aber nicht übertragen werden kann.

Jede Komponente in einer Exadata ist über ein InfiniBand-Netzwerk angeschlossen, das maximal eine Übertragung von vier GBps pro Komponente zulässt. Zum Vergleich: Die besten High-End-Storage-Systeme liefern heute einen Durchsatz von rund 35 GBps. Aber dieser Durchsatz muss erst einmal zu den Servern übertragen werden, denn der Scan wird von den Datenbank-Servern durchgeführt. Die Datenbank-Server der Exadata werden während eines solchen Scans nicht belastet.

Elastische Konfigurationen

Exadata-Systeme können neu modularer zusammengesetzt und ausgebaut werden. Ausgangspunkt ist immer noch das Achtel-Rack. Aber Datenbank- und auch Storage-Server lassen sich dann nach Bedarf hinzufügen. Dadurch können die Ausbaukosten eines bestehenden Exadata-Systems besser den Bedürfnissen angepasst werden.

Von Vorteil ist auch, dass in Exadata-Systemen die Komponenten verschiedener Generationen kombiniert werden können. Selbst X2-Systeme werden von der neuesten Exadata-Software unterstützt und können so um aktuelle Komponenten ergänzt werden. Und das Schönste daran ist, dass Oracle sich um die Kompatibilität aller Komponenten kümmert. Das wirkt sich günstig auf die Betriebskosten aus. Typischerweise patchen Kunden zweimal pro Jahr ihre Exadata-Systeme.

Oracle Enterprise Edition

Oracle-Exadata-Server werden ausschließlich mit der Oracle Enterprise Edition angeboten, was zwangsläufig zu hohen Lizenzkosten führt. Die Standard Edition wird zunehmend als kostengünstige Alternative gesehen, da sie in Verbindung mit leistungsfähigen Zwei- und Vier-Socket-Servern, großen Hauptspeicher-Kapazitäten und Flash-Technologie auch ohne die häufig vermissten Technologien wie Partitioning und Parallel Query extrem effiziente Datenbank-Plattformen ermöglicht. In diesem Zusammenhang sei auf verschiedene Publikationen und Vorträge *[1]* zum Thema „Oracle Standard Edition“ hingewiesen.

Anschaffungs- und Betriebskosten

Da Hersteller-Angaben immer mit Vorsicht zu genießen sind, greift der Autor lieber auf eine interne Langzeitstudie eines Exadata-Kunden zurück *[2]*. Vor allem bei der Zuverlässigkeit und den Betriebskosten übertreffen Exadata-Systeme alle Erwartungen. Beim Plattform-Engineering konnte der Aufwand deutlich reduziert werden: Oracle 60 Prozent, Unix 90 Prozent, Storage 100 Prozent.

Exadata-Systeme sind in der Regel innerhalb von zwei Wochen betriebsbereit, herkömmliche Plattformen selbst bei hohem Standardisierungsgrad der Verfahren und Prozesse häufig erst nach einigen Monaten. Ähnliche Einsparungen konnten auch bei den Betriebskosten erreicht werden: Oracle 40 Prozent, Unix 70 Prozent, Storage 100 Prozent.

Alte Applikationssoftware

Die Hardware-Technologie hat sich in den letzten Jahren kontinuierlich weiterentwickelt und bietet heute eine außergewöhnliche Leistungsfähigkeit. Leider kann die Leistungsfähigkeit häufig von Applikationen nicht genutzt werden, da diese vor vielen Jahren entwickelt und seitdem nicht an die Hardware-Technologie angepasst wurden.

Viele Oracle Features, die entwickelt wurden, um Leistungsengpässe der Hardware zu überwinden, sind heute eher Ballast. Viele andere wichtige Features werden nicht genutzt, oft aus Unkenntnis oder weil man den Aufwand für eine Anpassung der Software scheut. Bei einem Technologie-Wechsel, sei es auf Exadata oder auf konventionelle Plattformen, empfiehlt es sich, auch die Applikationssoftware zu überprüfen, um ein Maximum an Nutzen zu erzielen.

Fazit

Oracle bietet mit seinen Engineered Systems die mit weitem Abstand kompletteesten Systeme im Markt an. Die Lösungen der Mitbewerber sind häufig nur Referenz-Architekturen oder betreffen nur den Hardware-Stack. Oracle verwendet häufig Standard-Komponenten und veredelt sie mit Software, wenn es für den Oracle-Datenbank-Betrieb nützlich ist.

Mitbewerber versuchen eher, ihre Komponenten zu verbessern, haben aber oft kein umfassendes Angebot für den Oracle-Datenbank-Kunden. Oracle hat bei der Exadata X5-2 sehr viele Detail-Verbesserungen eingebracht, die diese Lösung erheblich attraktiver machen, etwa die neuen Möglichkeiten der Lizenzierung und die Modularität und Aufwärtskompatibilität beim Ausbau der Systeme.

Mit Einführung der Flash-Storage-Server und Verbesserung der Protokolle kann die Exadata jetzt auch bei OLTP-Systemen mit den meisten Mitbewerbern mithalten. Bei Data-Warehouse-Anwendungen ist die Performance von Exadata-Systemen im Vergleich zu konventionellen Plattformen dank der intelligenten Storage-Server-Software überragend. Konventionelle Plattformen sind in dieser Disziplin weit abgeschlagen.

Erste Langzeit-Studien bei Kunden bestätigen die Kostenvorteile bei der Beschaffung und im tagtäglichen Betrieb. Einziger Wermutstropfen bleiben die hohen Lizenzkosten, insbesondere für den Flash-Storage. Jeder Oracle-Kunde, der neue Plattformen evaluiert, sollte sich auch Exadata-Systeme anschauen. Organisatorische Vorkehrungen bei deren Einführung helfen, die Inbetriebnahme und die Migration von Applikationen zu beschleunigen und somit die Akzeptanz schnell zu erhöhen.

Zusammenfassend kann gesagt werden, dass es je nach Anwendungsgebiet Alternativen zur Exadata gibt, Oracle es aber seinen Mitbewerbern immer schwerer macht.

Referenzen

[1] http://www.carajandb.com

[2] The Exadata case; SwissRe internal study of Exadata X2-2 Full Rack over 2 years.

Manfred Drozd
manfred.drozd@benchware.ch

ETL-Prozesse beschleunigen

Dani Schnider, Trivadis AG

Man erlebt hin und wieder die Situation, dass die Nacht zu kurz ist. Oder mit anderen Worten: Der nächtliche ETL-Lauf dauert so lange, dass die Daten am Morgen nicht rechtzeitig in den Data Marts zur Verfügung stehen. Um für diese Problematik Abhilfe zu schaffen, gibt es ein paar grundlegenden Maßnahmen, die beim Entwickeln von ETL-Prozessen beachtet werden sollten.

ETL-Tools und Datenbank-Systeme stellen verschiedene Features und Möglichkeiten zur Verfügung, die ein effizientes Laden von Daten ins Data Warehouse erlauben oder mit denen die bestehenden Ladeprozesse beschleunigt werden können. Doch leider werden diese Möglichkeiten oft nicht optimal ausgenutzt. Immer wieder muss der Autor im Rahmen von Performance-Einsätzen und Reviews von Data Warehouses feststellen, dass teilweise elementare Regeln zur effizienten Realisierung von ETL-Prozessen missachtet werden. Ohne hier zu stark auf die Spezialitäten der einzelnen Tools einzugehen, erläutert der Artikel einige wichtige Grundlagen, die Voraussetzung für eine gute Performance der Ladeprozesse sind.

ELT statt ETL

Je nach verwendeter Technologie der ETL-Tools werden die Transformationen auf einem ETL-Server durchgeführt oder finden innerhalb der Datenbank statt. Im zweiten Fall wird auch oft der Begriff „ELT" (Extract, Load, Transform) statt „ETL" (Extract, Transform, Load) verwendet. Insbesondere bei großen Datenmengen sowie langsamen Netz-Verbindungen zwischen ETL-Server und Datenbank können diese zwei Arbeitsweisen markante Unterschiede in der Laufzeit von ETL-Prozessen zur Folge haben. In der Regel sind ELT-Verarbeitungen schneller, da der Datentransfer zwischen ETL-Server und Datenbank entfällt. Durch geeignete Maßnahmen wie schnelle Netzwerk-Verbindungen oder Bulk-Verarbeitung zwischen ETL-Server und Datenbank kann die Verarbeitungszeit soweit optimiert werden, dass sie in einer ähnlichen Größenordnung wie die ELT-Verarbeitung liegt *(siehe Abbildung 1)*.

In vielen Data Warehouses kommen Integrationswerkzeuge zum Einsatz, die ausschließlich ETL-Funktionalität verwenden, ohne dass dies zu Performance-Problemen führt. Solange die zu ladenden Datenmengen genügend klein sind, fällt die Verarbeitungszeit meistens nicht ins Gewicht. Für größere Datenmengen stehen zum Teil spezielle ELT-Features zur Verfügung, die es erlauben, die Transformationsprozesse auf der Ziel-Datenbank durchzuführen (beispielsweise Pushdown Optimization in Informatica PowerCenter oder ELT-Komponenten in Talend Open Studio). Bietet ein ETL-Tool keine entsprechende Funktionalität an, wird oft die Möglichkeit gewählt, zeitkritische Verarbeitungen mittels SQL in der Datenbank zu implementieren (etwa in PL/SQL-Packages) und aus dem ETL-Tool aufzurufen.

Bei den Integrationstools von Oracle ist dies nicht notwendig: Sowohl Oracle Warehouse Builder (OWB) als auch Oracle Data Integrator (ODI) arbeiten standardmäßig als ELT-Werkzeuge. OWB generiert PL/SQL-Packages in der Ziel-Datenbank. ODI führt die Transformationen in den meisten

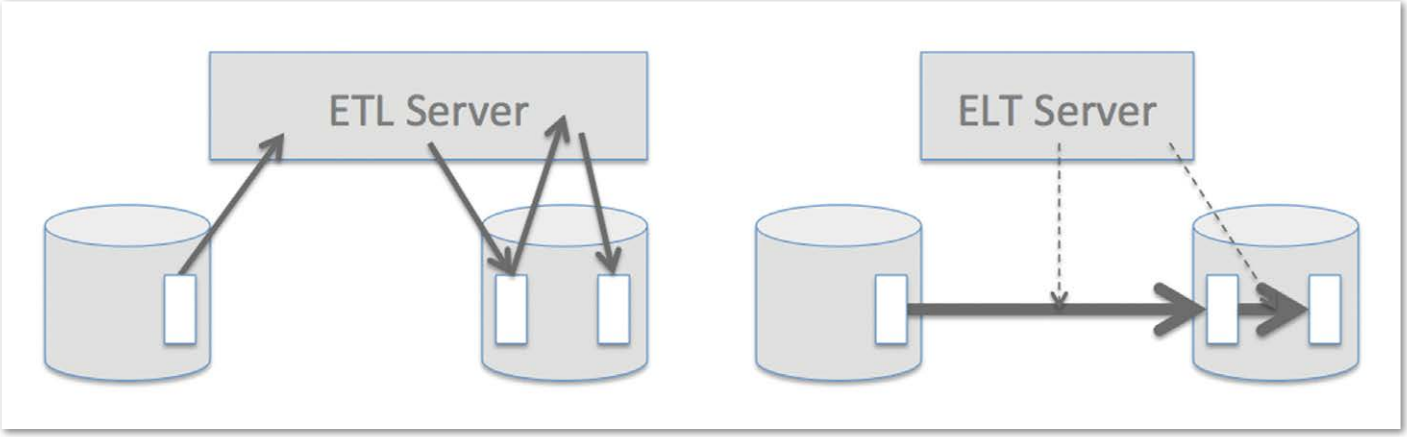


Abbildung 1: Arbeitsweise von ETL (links) und ELT (rechts)

Fällen als SQL-Befehle in der Zieldatenbank aus, unterstützt aber in heterogenen Umgebungen auch ETL-Verarbeitungen.

Mengen- statt datensatzbasierter Verarbeitung

Die Verarbeitung eines Transformationsprozesses – sei es als ETL oder ELT – kann entweder mengen- („set-based") oder datensatzbasiert („row-based") erfolgen. Wer den OWB einsetzt, kennt diese Unterscheidung, denn dieser unterstützt beide Ausführungsarten. Beim ODI kommt in den meisten Knowledge-Modulen eine mengenbasierte Verarbeitung vor. Verschiedene ETL-Tools von Fremdherstellern arbeiten datensatzbasiert, erlauben aber – mit mehr oder weniger Komfort – auch die Ausführung mengenbasierter Ladeprozesse.

Die unterschiedliche Verarbeitungsweise soll an einem einfachen Beispiel mit PL/SQL und SQL aufgezeigt werden. Eine Stage-Tabelle „STG_SALES" soll in eine Cleanse-Tabelle „CLS_SALES" geladen werden, wobei folgende Transformationen erfolgen sollen:

- Den Datumschlüssel „DATE_ID" aus dem Verkaufsdatum (ohne Uhrzeit) ermitteln
- Den Produktschlüssel „PRODUCT_ID" mittels Key Lookup auf der Core-Tabelle „COR_PRODUCT" ermitteln. Ist kein entsprechendes Produkt vorhanden, soll ein Singleton-Wert (-1) eingefügt werden
- Den Verkaufskanal „CHANNEL_ID" anhand eines Flags „ONLINE_FLAG" ermitteln
- Fehlt die Mengen-Angabe für einen Verkauf, wird für das Attribut „QUANTITY" der Wert „1" verwendet

```
DECLARE
   CURSOR cur_stage IS
      SELECT sales_date, product_code, online_flag, quantity
      FROM stg_sales;
   v_cls  cls_product%ROWTYPE;
BEGIN
   FOR v_stg IN cur_stage LOOP
      -- convert sales date
      v_cls.date_id := TRUNC(v_stg.sales_date);
      -- lookup product id
      BEGIN
         SELECT dwh_id
         INTO v_cls.product_id
         FROM cor_product
         WHERE product_code = v_stg.product_code;
      EXCEPTION
         WHEN NO_DATA_FOUND THEN
            v_cls.product_id := -1;
      END;
      -- define sales channel
      IF v_stg.online_flag = 'Y' THEN
         v_cls.channel_id = 1; -- internet
      ELSIF v_stg.online_flag = 'N' THEN
         v_cls.channel_id = 2; -- shop
      ELSE
         v_cls.channel_id = -1; -- unknown
      END IF;
      -- cleansing action for quantity
      IF v_stg.quantity IS NULL THEN
         v_cls.quantity := 1;
      ELSE
         v_cls.quantity := v_stg.quantity;
      END IF;
      -- insert row in cleanse table
      INSERT INTO cls_sales VALUES v_cls;
      COMMIT;
   END LOOP;
END;
```

Listing 1: Datensatzbasierte Verarbeitung („row-based")

Listing 1 zeigt, wie diese Anforderungen mittels PL/SQL-Block als datensatzbasierte Verarbeitung implementiert werden könnte. Über einen SQL-Cursor wird jeder Datensatz aus der Stage-Tabelle gelesen, transformiert und in die Cleanse-Tabelle geschrieben. Die Lösung funktioniert, ist aber alles andere als schnell. Insbesonde-

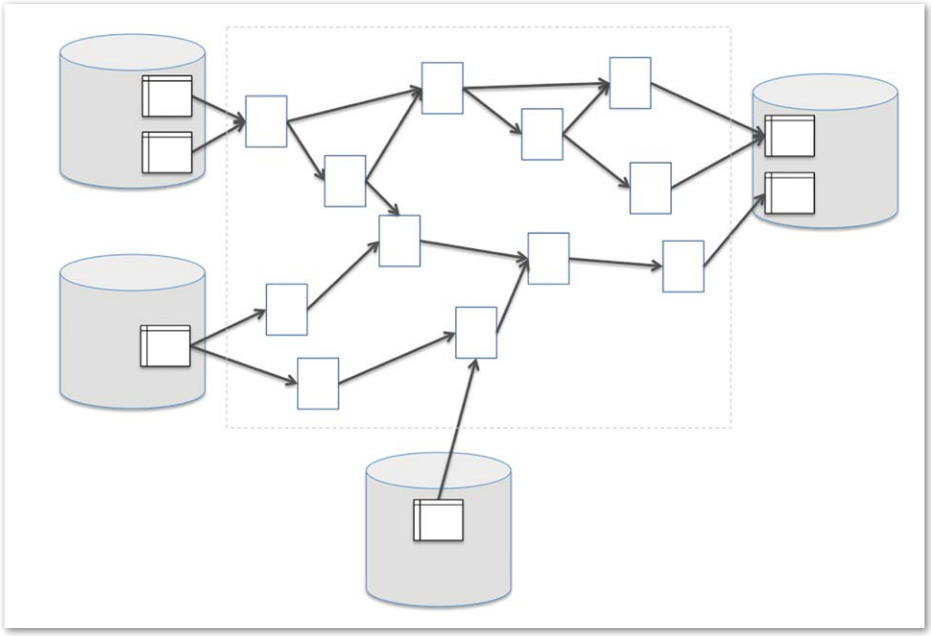


Abbildung 2: Beispiel für komplexen ETL-Prozess

```
INSERT /*+ append */ INTO cls_
sales
       ( date_id
       , product_id
       , channel_id
       , quantity)
SELECT TRUNC(stg.sales_date)
       , NVL(lkp.dwh_id, -1)
       , CASE stg.online_flag
           WHEN 'Y' THEN  1
           WHEN 'N' THEN  2
           ELSE           1
   END
   , NVL(stg.quantity, 1)
   FROM stg_sales stg
   LEFT OUTER JOIN cor_product
lkp
       ON (stg.store_number = lkp.
store_number);
COMMIT;
```

Listing 2: Mengenbasierte Verarbeitung („set-based“)

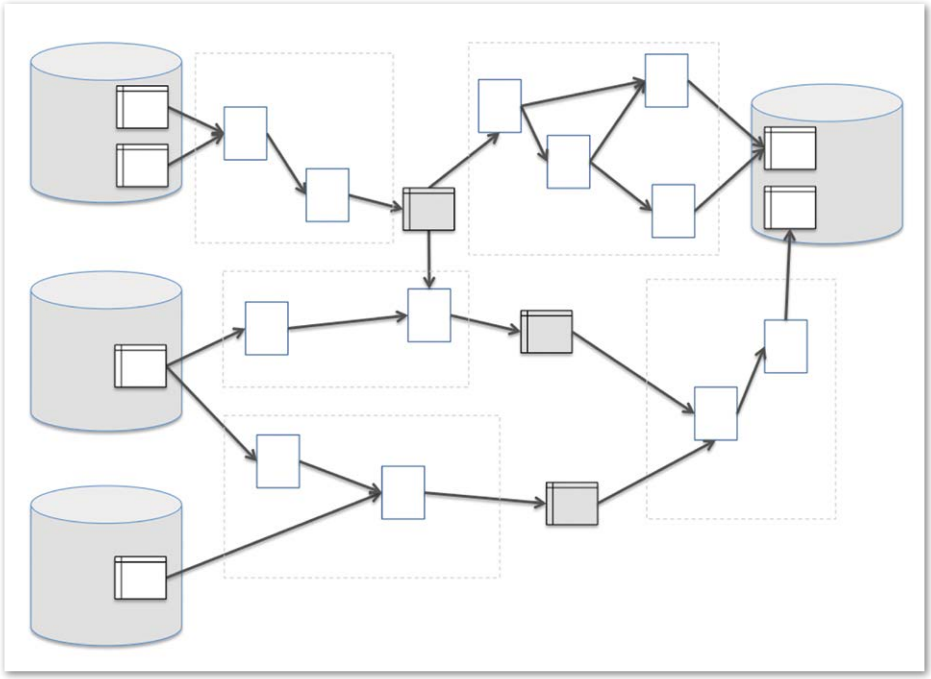


Abbildung 3: Aufteilung in fünf überschaubare Sub-Prozesse

re der ausprogrammierte Key Lookup und das „COMMIT“ nach jedem Datensatz können als Performance-technische Todsünden bezeichnet werden.

Deutlich effizienter ist es, die gleiche Logik mit mengenbasiertem SQL-Befehl zu formulieren. Dies ermöglicht das effiziente Laden von Daten aus einer oder mehreren Quell-Tabellen in eine Ziel-Tabelle durch SQL-Befehle („INSERT“ oder „MERGE“). Das SQL-Statement in *Listing 2* umfasst die gleiche Funktionalität wie der zuvor beschriebene PL/SQL-Block, wird aber bei großen Datenmengen viel schneller ausgeführt. Um die Verarbeitung zusätzlich zu beschleunigen, wird in Oracle ein Direct-Load „INSERT“ (mit dem Hint „/*+ append */“) verwendet.

Was hier mit PL/SQL und SQL aufgezeigt wird, funktioniert auch in zahlreichen Integrationswerkzeugen. Bei OWB und ODI entspricht die mengenbasierte Verarbeitung der (empfohlenen) Standardausführung. Andere ETL-Tools bieten zum Teil Operatoren zur Ausführung von SQL-Befehlen an oder ermöglichen es, die Datensätze mithilfe von Array-Verarbeitung aufzubereiten und mittels Bulk-Operationen in die Datenbank zu schreiben. Werden beispielsweise jeweils 1.000 oder 10.000 Datensätze mit einem „INSERT“-Befehl in die Datenbank geschrieben, ist dies schon deutlich schneller, als wenn für jeden einzelnen Datensatz ein SQL-Befehl ausgeführt werden muss.

Reduktion der Komplexität

Unabhängig von der eingesetzten Technologie ist folgende Performance-Maßnahme immer zu empfehlen: Komplexe ETL-Prozesse sollten, wann immer möglich, in mehrere, überschaubare Einzelschritte aufgeteilt sein. Dies gilt sowohl bei der Implementierung mit SQL oder mit einer prozeduralen Programmiersprache als auch bei der Realisierung mit einem ETL-Tool. Bei prozeduralen Sprachen gehört es zum guten Programmierstil, komplexe Abläufe zu modularisieren und in mehrere Subprogramme oder Prozeduren aufzuteilen. Das gleiche Prinzip gilt auch bei der Entwicklung von komplexen ETL-Prozessen, die als Mappings oder Jobs mit einem ETL-Tool implementiert werden. Der in *Abbildung 2* schematisch dargestellte ETL-Prozess kann – und soll – in mehrere Teilschritte aufgeteilt sein, die

nacheinander oder parallel ausgeführt werden können.

Anstatt den komplexen ETL-Prozess in einem umfangreichen Mapping zu implementieren, wird der Ablauf in mehrere Subprozesse unterteilt, die als separate Mappings implementiert sind *(siehe Abbildung 3)*. Dies hat mehrere Vorteile:

- Die Summe der Ausführungszeiten der einzelnen Sub-Prozesse ist viel kürzer als die Ausführungszeit des gesamten Prozesses, da für die einfacheren Operationen viel weniger Ressourcen benötigt werden. Bei ELT-Tools oder SQL-Statements, die direkt in der Datenbank ausgeführt werden, kommt hinzu, dass der Query Optimizer der Datenbank die einfacheren Statements leichter optimieren kann.
- Die einzelnen Mappings sind einfacher überschaubar, was die Weiterentwicklung bei zukünftigen Erweiterungen sowie die Einarbeitung neuer ETL-Entwickler vereinfacht. Auch hier gilt das Gleiche wie bei Programmiersprachen: Überschaubare Programme sind leichter zu verstehen als „Spaghetti-Code".
- Schließlich ist auch die Fehlersuche einfacher, da die Zwischen-Resultate in Stage-Tabellen oder weiteren Zwischen-Tabellen abgespeichert und dort vom nächsten Verarbeitungsschritt wieder gelesen werden. Somit lässt sich im Falle von fehlerhaften Daten einfacher nachvollziehen, in welchem Teilschritt der Fehler auftritt, da man die Zwischen-Resultate analysieren kann.

Die Reduktion der Komplexität bewirkt somit nicht nur kürzere Laufzeiten der ETL-Prozesse, sondern führt auch zu Zeiteinsparungen bei der Entwicklung und beim Testen der Ladestrecken. Wir haben also durch diese Maßnahme auch Performance-Verbesserungen bei der Realisierungszeit erreicht.

Frühzeitige Mengen-Einschränkung

Je kleiner die zu verarbeitende Datenmenge ist, desto schneller lässt sich ein ETL-Prozess ausführen. Deshalb ist es wichtig, bei der ETL-Entwicklung darauf zu achten, dass die Datenmenge so früh wie möglich eingeschränkt wird. Ist ein Mapping so aufgebaut, wie in *Abbildung 4* gezeigt,

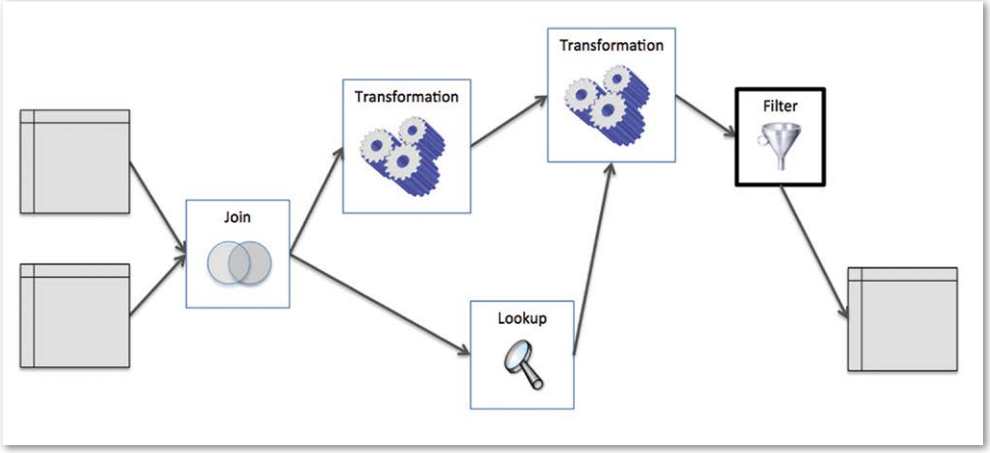


Abbildung 4: Mengen-Einschränkung am Ende des ETL-Prozesses

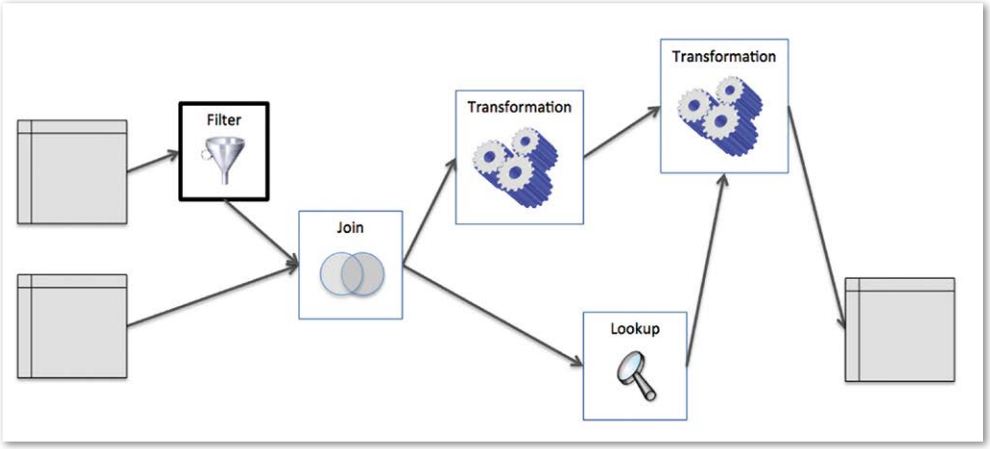


Abbildung 5: Frühzeitige Mengeneinschränkung am Anfang des ETL-Prozesses

kann dies negative Auswirkungen auf die Performance des ETL-Prozesses haben, denn bei ELT-Tools ist es möglich, dass der Query Optimizer der Datenbank die ausgeführten SQL-Befehle so optimieren kann, dass trotz schlechten Designs des Mappings die frühzeitige Mengeneinschränkung funktioniert. Nach aufwändigen Zwischenschritten und Transformationen wird am Ende ein Filter eingefügt, der nur eine Teilmenge der Daten in die Zieltabelle schreibt. Das bedeutet, dass für alle nicht relevanten Datensätze die Transformationsschritte ebenfalls ausgeführt wurden – und zwar vergeblich. Dies sollte möglichst vermieden werden.

Besser ist es, die Filterkriterien so früh wie möglich anzuwenden und so die Datenmenge für die nachfolgenden Transformationsschritte zu reduzieren. Das Beispiel in *Abbildung 5* ist ein optimaler Fall, da eine Filterung der Daten bereits auf einer der beiden Quell-Tabellen erfolgen kann (etwa das Lesen der aktuellen Version aus einer Dimensions-Tabelle). Das ist nicht immer möglich. Unter Umständen kann ein Filter-Kriterium erst aus dem Ergebnis eines Joins, eines Key Lookups oder einer Transformation ermittelt werden (etwa das Eliminieren aller Datensätze, für die beim Key Lookup kein passender Schlüssel gefunden wurde). Aber auch dann sollte die Filterung so früh wie möglich stattfinden und nicht erst vor dem Schreiben in die Ziel-Tabelle.

Parallelisierung

Um die Ladezeit ins Data Warehouse zu verkürzen, besteht die Möglichkeit, die ETL-Prozesse zu parallelisieren. Dabei stehen verschiedene Varianten zur Verfügung. Einerseits können die einzelnen SQL-Befehle zum Lesen und Schreiben der Datenbank-Tabellen parallelisiert werden. Andrerseits lassen sich mehrere separate ETL-Prozesse gleichzeitig ausführen.

Idealerweise sollte die Parallelisierung den gesamten ETL-Ablauf umfassen: Die Da-

ten werden zum Beispiel mit mehreren parallelen Sub-Prozessen aus der Stage-Tabelle gelesen, transformiert und anschließend in die Cleanse-Tabelle geschrieben. Erfolgt einer der Schritte (beispielsweise die Transformation) seriell, so führt dies zu einem Flaschenhals in der Verarbeitung und somit zu einer längeren Ausführungszeit. Autofahrer kennen die Situation, wenn sie auf einer mehrspurigen Autobahn unterwegs sind. Eine Reduktion der Spuren – zum Beispiel durch eine Baustelle – führt unweigerlich zu einem Rückstau und somit zu „Performance-Problemen" im Straßenverkehr.

Zurück von der Autobahn ins Data Warehouse: Idealerweise erfolgt die Parallelisierung der Ladeprozesse durch ELT-Technologien, es werden also die Parallelisierungsmöglichkeiten der Datenbank ausgenutzt. Bei einer mengenbasierten Ausführung mit Datenbank-Features wie Parallel-Query- und Parallel-DML-Operationen lässt sich ein optimaler Datendurchsatz erreichen. *Listing 3* zeigt das bereits beschriebene Beispiel einer mengenbasierten Verarbeitung, allerdings achtfach parallelisiert. Je acht SQL-Subprozesse lesen die Daten aus der Stage-Tabelle „STG_SALES" und schreiben sie in die Cleanse-Tabelle „CLS_SALES".

Eine andere Variante der Parallelisierung besteht darin, mehrere ETL-Prozesse gleichzeitig auszuführen. Solange die einzelnen Abläufe unabhängig voneinander sind und verschiedene Quellen und Ziele haben, ist dies eine einfache Maßnahme, um die Ausführungszeit eines ETL-Laufs zu reduzieren. Ein typischer Anwendungsfall ist beispielsweise das parallele Laden aller Dimensionen. Das Laden der Fakten-Tabelle kann hingegen erst beginnen, wenn alle Dimensions-Tabellen vollständig geladen sind.

Vermieden werden sollte hingegen die gleichzeitige Ausführung von ETL-Prozessen, die in dieselbe Ziel-Tabelle schreiben. Dies kann zu Locking-Problemen auf der Datenbank führen, wenn mehrere Client-Prozesse in die gleichen Tabellen oder Partitionen schreiben. Bei Oracle ist dies vor allem dann der Fall, wenn auf der Ziel-Tabelle Bitmap-Indizes vorhanden sind.

Aus Sicht der Datenbank ist dieses Verfahren vergleichbar mit einem Multi-User-Betrieb in einem OLTP-System und nicht geeignet für die Massenverarbeitung von großen Datenmengen. Möglich ist dieses Prinzip höchstens in Kombination mit partitionierten Tabellen. Kann sichergestellt werden, dass jeder Prozess in eine separate Ziel-Partition schreibt, so können mehrere ETL-Prozesse parallel ablaufen, um Daten in die gleiche Tabelle zu schreiben.

```
INSERT /*+ parallel(cls, 8) */
INTO cls_sales cls
      ( date_id
      , product_id
      , channel_id
      , quantity)
SELECT /*+ parallel (stg, 8) */
         TRUNC(stg.sales_date)
      , NVL(lkp.dwh_id, -1)
      , CASE stg.online_flag
          WHEN 'Y' THEN  1
          WHEN 'N' THEN  2
          ELSE           -1
        END
      , NVL(stg.quantity, 1)
  FROM stg_sales stg
  LEFT OUTER JOIN cor_product
lkp
      ON (stg.store_number = lkp.
store_number);
COMMIT;
```

Listing 3: Parallele Ausführung von SQL-Befehlen

Datenbank-Konfiguration

Die bisher beschriebenen Performance-Maßnahmen betreffen hauptsächlich Design und Entwicklung der ETL-Prozesse. Daneben sind aber auch die korrekte Konfiguration der Datenbank sowie das physische Datenbank-Design wichtig, um die Ladezeiten möglichst klein zu halten.

- In Data Warehouses gelten andere Regeln als in OLTP-Systemen. Dies hat Auswirkungen auf die Konfiguration der Datenbanken. Eine Data-Warehouse-Datenbank wird so konfiguriert, dass sie für nicht-selektive Abfragen optimiert ist, genügend Arbeitsspeicher für Massenverarbeitungen zur Verfügung hat und die parallele Ausführung von ETL-Prozessen und Auswertungen erlaubt *(„siehe https://danischnider.wordpress.com/2015/04/29/oracle-initialization-parameters-for-data-warehouses")*.
- Auch die Indexierung ist in einem Data Warehouse unterschiedlich zu einem operativen System. Für effiziente ETL-Prozesse ist es zweckmäßig, so wenige Indizes wie möglich anzulegen. Einzig auf den Data Marts, die für Benutzerabfragen optimiert sind, werden Indizes angelegt – in der Regel Bitmap-Indizes. In allen anderen DWH-Schichten werden nur wenige oder gar keine Indizes erstellt *(siehe „http://www.slideshare.net/trivadis/indexierungsstrategie-im-data-warehouse-zwischen-albtraum-und-optimaler-performance-39738594")*. Je nach Art der Ladeverarbeitung (initiale oder inkrementelle Ladeprozesse) kann es sogar sinnvoll sein, die Indizes vor dem Laden auf „UNUSABLE" zu setzen und nach dem Laden mit „REBUILD" neu zu erstellen.
- Für die Abfragen, aber auch für nachfolgende ETL-Prozesse, ist es wichtig, dass man nach dem Laden von Daten die Optimizer-Statistiken der geladenen Tabellen und Indizes aktualisiert. Dies sollte nicht erst am Ende eines kompletten Ladelaufs erfolgen, sondern nach jedem Zwischenschritt. Die zuerst geladenen Tabellen werden in weiteren ETL-Prozessen als Quellen verwendet. Insbesondere bei der mengenbasierten ELT-Verarbeitung ist es wichtig, dass der Query Optimizer korrekte Statistiken für die Optimierung der nachfolgenden Transformationsschritte verwenden kann *(siehe „https://danischnider.wordpress.com")*.

Sind Konfiguration und Design der Datenbank für ein Data Warehouse ausgelegt und werden die ETL-Prozesse gemäß den hier beschriebenen Grundprinzipien implementiert, steht einem erfolgreichen und effizienten Laden des Data Warehouse nichts mehr im Wege. Die Nacht ist somit nicht mehr zu kurz und die Daten im Data Warehouse stehen am Morgen für die Endanwender rechtzeitig zur Verfügung.

Dani Schnider
dani.schnider@trivadis.com

Tuning von Oracle-Web-Applikationen im WebLogic Server 12c

Markus Klenke, Team GmbH

Beim Entwickeln der ersten prototypischen Applikationen mit JSF oder Oracle ADF wird häufig nicht direkt auf die Laufzeit und Performance der Applikation geschaut. Fehlende Optimierungen fallen meist erst bei größeren Applikationen, höheren Nutzerzahlen oder dem allgemeinen Produktivgang auf.

Es kommt nicht infrage, die Applikation neu aufzubauen. Das ist auch meist nicht notwendig, da einige Stellschrauben vorhanden sind, mit denen es häufig möglich ist, die notwendige Leistung aus der Anwendung und dem Server herauszukitzeln. Der Artikel zeigt verschiedene dieser Stellschrauben an der Applikation, dem Deployment und dem WebLogic Server 12*c* auf.

Das Erstellen von Geschäfts-Applikationen mit einem neuen Framework oder einer neuen Programmiersprache folgt meistens einem ähnlichen Schema: Schulungen erhalten, Prototypen bauen, Prototypen einstampfen, eigentliches Projekt beginnen.

Dieser Ansatz klingt nach einem guten Plan, er ist es in den meisten Fällen auch. Leider kann ein Prototyp bei Weitem nicht alle Anforderungen an das Echt-Projekt abbilden. In vielen Fällen wird mit Mock-Daten gearbeitet, höchstens mit einer oder zwei Personen getestet oder mal eben schnell die erste Lösung für ein Problem gewählt.

Beginnt dann die eigentliche Implementierung des Projekts, treten bei erstem Last-Verhalten die Schweißperlen auf die Stirn der Entwickler/Administratoren. Die Applikation ist auf einmal sehr langsam, bei vielen Benutzern bricht der Server zusammen, Benutzer sperren sich gegenseitig Datensätze etc. Woran kann das liegen und wo kann man nach diesen Problemfällen schauen?

Tendenziell gibt es mindestens drei Bereiche für potenzielle Applikations-Schwachpunkte: Die Applikation selbst, das Deployment und der Web-Server. Zusätzlich gibt es diverse Datenbank-Optimierungen, auf die dieser Artikel nicht eingehen wird.

Oracle ADF

Das Application Development Framework (ADF) ist das zentrale Java-Framework von Oracle zur Erstellung von Enterprise-Applikationen. Technisch gesehen bietet es diverse Interfaces, die beliebige Datenquellen über eine Abstraktionsschicht an eine Model-View-Controller-Muster-getriebene Web-Applikation binden. Dadurch ergeben sich innerhalb der ADF-Applikation mehrere Schichten, die für eine Performance-Analyse begutachtet werden müssen. *Abbildung 1* zeigt eine Übersicht über die verschiedenen Schichten. Dieser Artikel analysiert die Standard-Oracle-Implementierungen auf Performance-Optimierung.

ADF Business Components – ein Ebenbild der Datenbank

Die ADF Business Components sind das Persistenz-Framework der Daten für die ADF-Applikation. Aufgabe der Schicht ist es, Daten aus der Datenbank für Caching-Zwecke zu replizieren und diese den Oberflächen bereitzustellen. Um die Cache-Elemente zu erstellen und zu füllen, sind Queries an die Datenbank in den sogenannten „View-Objekten" gekapselt. Diese lassen sich ähnlich

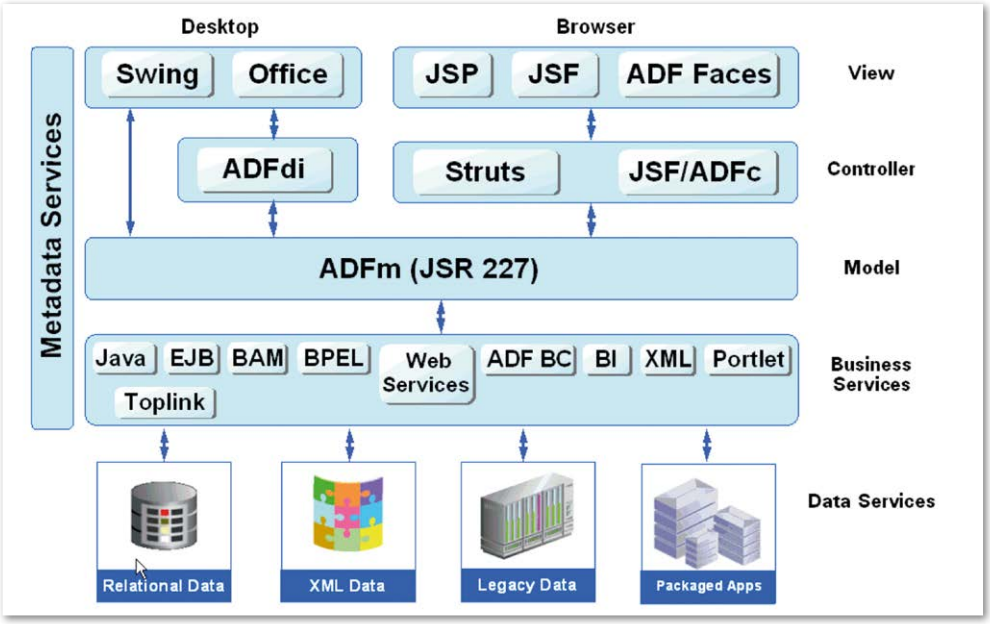


Abbildung 1: Schichten in der ADF-Entwicklung

wie allgemeine Datenbank-Queries optimieren. So können am View-Objekt selbst Optimizer Hints genauso benutzt werden wie bei der Datenbank, um der Datenbank bei der Abfrage direkt mitzugeben, worauf die Abfrage optimiert sein soll.

In vielen Fällen ist es absolut unnötig, alle Werte aus der Rückgabe-Menge sofort zu laden, da entweder eine Formular- oder eine Tabellen-Darstellung mit weniger als „n" Zeilen auf der Oberfläche vorliegt. Um nicht nur die Datenbank Queries, sondern auch das Instanziierungs-Verhalten der Objekte zu optimieren, bietet ADF an den View-Objekten weitere Einstellungen. So ist es zum Beispiel möglich, ein View-Objekt rein für die Erstellung eines Datensatzes zu benutzen. Die Query dient daher nur der Information, wo dieser Satz später in der Datenbank abgespeichert wird *(siehe Abbildung 2)*.

Application Modules sind die Services der Business Components, um die Datenabfragen und Rückgabewerte an die Außenwelt zu propagieren. Diese Objekte organisieren zusätzlich die Instanziierung von View-Objekten. Im Normalfall wird für jeden Anwendungsbenutzer mindestens ein Application Module erzeugt, um für die Nutzungszeit der Applikation eine Verbindung mit der Datenbank aufrechtzuerhalten.

Es gibt allerdings View-Objekte, bei denen es wenig Sinn ergibt, sie für jeden Benutzer neu zu instanziieren. Ein Beispiel ist ein View-Objekt, das etwa Übersetzungstexte aus der Datenbank liest. Diese Texte ändern sich für gewöhnlich nicht zwischen den Release-Zyklen. Um diese Objekte als Singleton zu erstellen, kann das Application Module als „Shared Application Module" deklariert sein. Dieser Modus sorgt dafür, dass nur ein einziges Modul dieses Typs erstellt und von allen Benutzern gleichermaßen verwendet wird. Man sollte dazu sagen, dass die Query des View-Objekts zur Laufzeit für jeden Benutzer individuell verändert werden kann, etwa für Sprach-Präferenzen. Das sorgt dafür, dass die abgefragten Elemente auch nur ein einziges Mal gecacht werden, was gerade bei Anwendungen mit vielen Benutzern einen enormen Speichervorteil mit sich bringt. Zusätzlich wird die Geschwindigkeit der Anwendung gesteigert, da bei schon gecachten Objekten die Query auf der Middleware ausgeführt wird und nicht auf der Datenbank.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<deployment-plan …>
  <application-name>MyHRApp</application-name>
      <variable-definition>
          <variable>
              <name>Disable_Content_Compression</name>
              <value xsi:nil="false">false</value>
      </variable>
      </variable-definition>
      <module-override>
          <module-name>MyHRApp.war</module-name>
          <module-type>war</module-type>
          <module-descriptor external="false">
              <root-element>web-app</root-element>
              <uri>WEB-INF/web.xml</uri>
              <variable-assignment>
                  <name>Disable_Content_Compression</name>
                  <xpath>/web-app/context-param/[param-name="org.apache.
myfaces.trinidad.DISABLE_CONTENT_COMPRESSION"]/param-value</xpath>
              <operation>replace</operation>
          </variable-assignment>
          </module-descriptor>
      </module-override>
</deployment-plan>
```

Listing 1: Deployment-Plan zur Veränderung von „web.xml"-Parametern

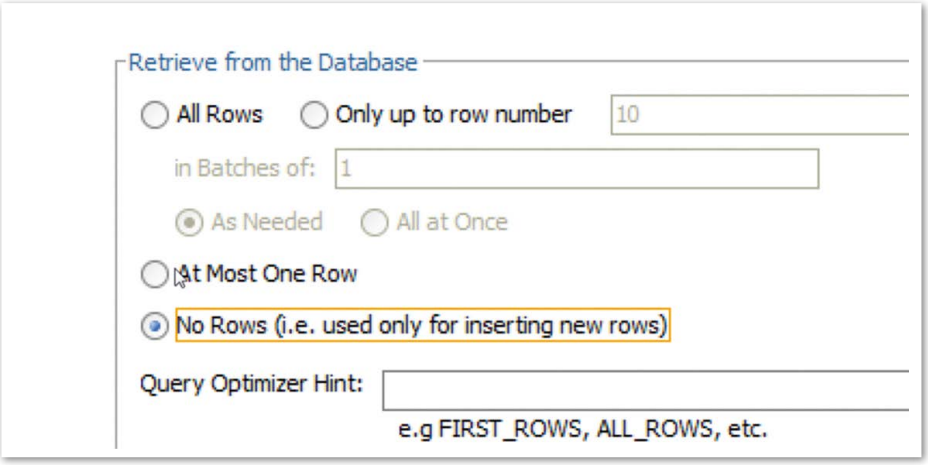


Abbildung 2: View-Objekt – Tuning-Einstellungen

Volle Kontrolle – Applikationsfluss über Managed Beans steuern

In Oracle ADF wird wie bei JSF der Kontext der Applikation in Managed Beans gespeichert. Dies sind Java-Klassen, die vom Framework gesteuert instanziiert werden und über einen bestimmten Zeitrahmen, den Scope der Bean, am Leben erhalten werden. Dadurch wird dem Entwickler einiges an Instanziierungs-Arbeit abgenommen. Managed Beans können in ADF in einem von sechs verschiedenen Scopes untergebracht sein, die unterschiedliche Lebenszyklen für die Bean bedeuten *(siehe Abbildung 3)*.

Ist eine Bean in einem kurzen Scope angesiedelt, wird sie entsprechend häufig aufgeräumt und neu instanziiert. Eine Bean im Application Scope würde hingegen ein einziges Mal beim Starten der Applikation instanziiert werden und dann für alle Benutzer der Applikation gültig sein.

Auf diesem Level der Applikation finden sich meistens zwei Extrema, die zu einer Performance-Verschlechterung oder einer System-Instabilität führen können. So kommt es immer mal vor, dass Beans mit einem hohen Kostenfaktor (lange Instanziierung, komplexe Berechnung etc.) in einem zu kleinen Scope liegen oder, anders herum gesagt, Java-Objekte mit einer großen Datenmenge für eine temporäre Berechnung im einem zu großen Scope.

Um die Auswirkung dieser beiden Fälle zu verdeutlichen, zwei Beispiele.

Erstes Beispiel: Eine Bean im Request-Scope (Lebenszyklus pro HTTP-Server-Anfrage) wird benutzt, um einen Wert aus der Datenbank zu lesen. Dieser wird zum Instanziierungs-Zeitpunkt über eine Datenbank-Funktion berechnet, die wiederum drei Sekunden für die Berechnung benötigt. Benutzer „A" lädt die Seite, die das Ergebnis der Funktion anzeigen soll. Er muss also entsprechend drei Sekunden warten, bis die Seite erscheint.

Benutzer „A" klickt auf der Seite auf einen Button, um eine weitere, aber sehr simple Funktion aufzurufen, die allerdings ebenfalls auf der Server-Seite ausgeführt wird. Statt einer kurzen Wartezeit muss Benutzer „A" nochmal drei Sekunden warten, da die Bean im Hintergrund neu instanziiert wird. Dies würde bei jeder weiteren Server-Anfrage wieder passieren. Wird die Bean in einen größeren Scope eingehängt (etwa „ViewScope", also Lebenszyklus einer Ansicht, die sich in unserem Beispiel nicht ändert), würde die erste Anfrage drei Sekunden benötigen, die anderen Anfragen könnten allerdings auf den bereits geladenen Wert zugreifen.

Zweites Beispiel: Eine Bean im Session Scope (Lebenszyklus pro User-Log-in) wird benutzt, um einen Summenwert über eine Liste von Artefakten zu generieren. Diese Liste besitzt 100.000 Einträge, die Daten-Akquise sowie die Summen-Berechnung sind sehr schnell. Ein Benutzer kommt auf die Seite, die das Summen-Ergebnis darstellt. Die Seite wird mehr oder weniger sofort angezeigt.

Für den End-Anwender läuft alles glatt und er beendet am Ende seines Arbeitstages die Session. Während der gesamten Zeit hält sich aber die Liste mit Einträgen im Speicher. Ist die Anzahl der Benutzer über den Tag auf eine bestimmte Anzahl angewachsen, ist die Gefahr, einen „Out of Memory"-Fehler zu erhalten, drastisch hoch, was zu einem Komplett-Ausfall des Servers führt. Wird die Bean hingegen in einem kleineren Scope gespeichert (wie „Request Scope"), muss zwar der Wert immer wieder neu berechnet werden, durch die geringe Komplexität fällt dies allerdings Performance-seitig nicht ins Gewicht; außerdem wird die Liste sehr zuverlässig vom Garbage Collector weggeräumt, was den Speicher-Lasten des Servers sehr guttut.

Von der Entwicklung zur Produktion – Deployment-Pläne

Ist die Applikation selbst so gut es geht optimiert, können Änderungen an Applikations-Parametern Performance-Gewinn einbringen. Im Entwicklungsmodus werden die Oberflächen-Komponenten von ADF meist unkomprimiert gerendert, was für Oberflächen-Tests auf Style-Klassen-Basis notwendig ist. Für den Produktiv-Einsatz ist dieser Modus allerdings vollkommen unnötig und verlangsamt nur die Renderzeit der Seite.

Muss man jetzt für den Entwicklungsmodus und den Produktivmodus zwei Applikationen simultan entwickeln? Glücklicherweise nicht, es besteht die Möglichkeit, einzelne XML-Strukturen der Applikation während des Deployments mithilfe eines Deployment-Plans an eine Umgebung anzupassen. So kann beispielsweise der Parameter zur Kompression der Oberflächen-Komponenten zu diesem Zeitpunkt auf „true" gestellt werden, damit auf der Produktion die Kompression der Klassennamen durchgeführt wird.

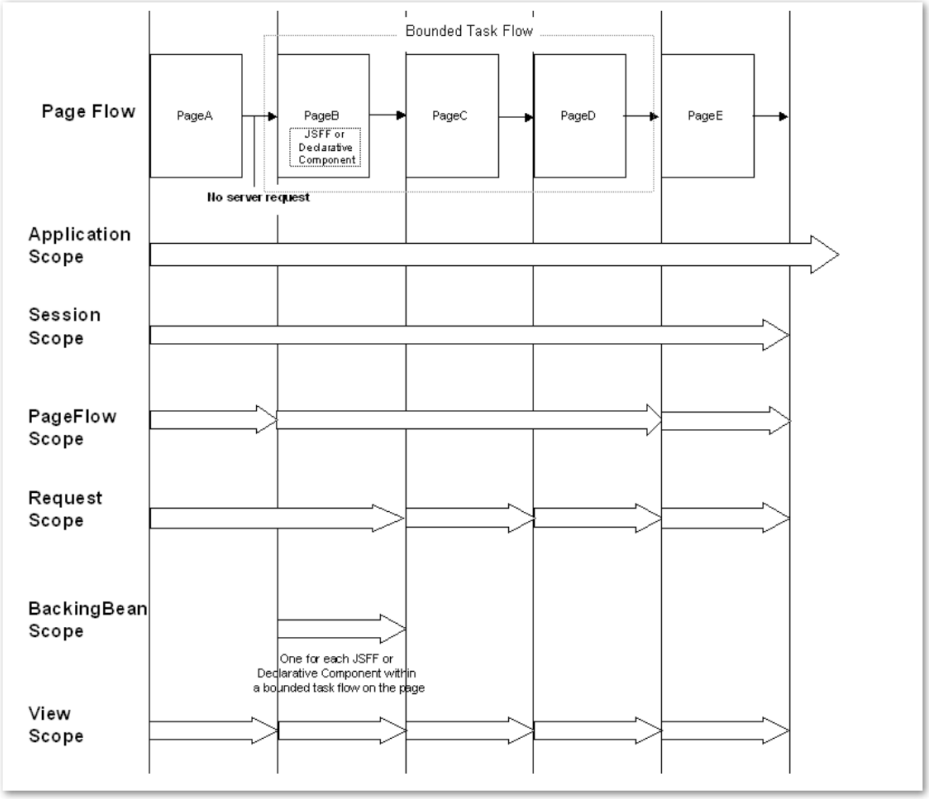


Abbildung 3: ADF Managed Bean Scopes

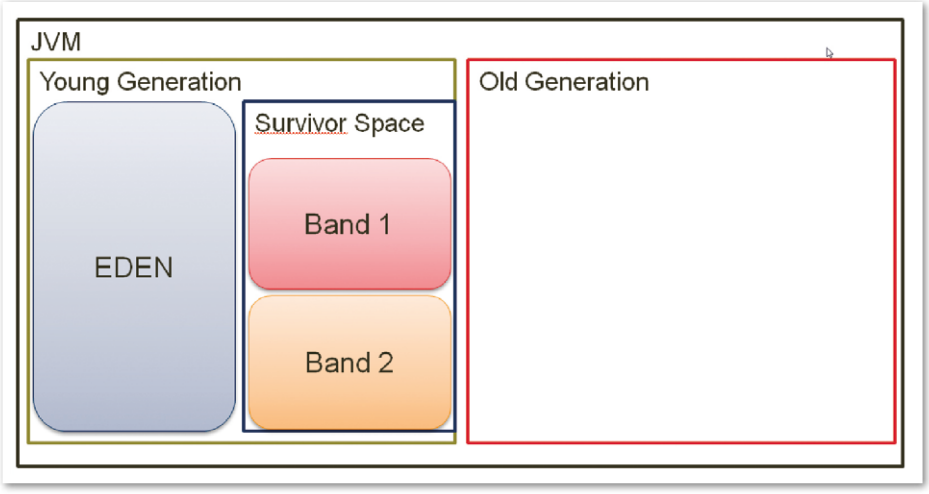


Abbildung 4: Visualisierung einer JVM

Es können insbesondere auch Parameter verändert werden, die das Applikationsverhalten alternieren. So lässt sich beispielsweise im Produktions-Modus eine Abfrage nicht mehr gegen eine Datenbank, sondern gegen einen parallel gestarteten Web-Service steuern, um Performance zu gewinnen. *Listing 1* zeigt ein Beispiel für einen Deployment-Plan, der die Kompression der Oberflächen-Komponenten aktiv schaltet:

JVM – Optimierung auf dem Server

Auf dem Server angekommen, gibt es immer noch diverse Möglichkeiten, die Applikations-Performance zu beeinflussen. Primär-Elemente dafür sind die Konfiguration der JVM des Servers und die Konfiguration der Datenquell-Pools. Dieser Artikel zeigt nur die JVM-Performance-Optimierungen.

Bei den Einstellungen der JVM scheiden sich die Geister. Daher sind die folgenden Einstellungen aus den Erfahrungen der ADF-Entwicklung entstanden und sollten nicht als Allheil-Werkzeug für die Entwicklung jeglicher Web-Applikationen gesehen werden. Wie beschrieben, muss die JVM großteilig mit Objekten aus den Managed Beans interagieren. Daher sollten die Speicher-Konfigurationsparameter auf genau dieses Szenario optimiert sein.

Generell benötigen Starts der Applikation mit einem noch nicht aktiven Benutzer einen relativ großen Anteil an „bootstrap"-Speicher. Bei einer Neuanmeldung wird also der größte Teil des gesamten vom Benutzer benötigten Speichers sofort benötigt. Es ist daher nicht sinnvoll, die initiale Heap-Size deutlich kleiner zu halten als die maximale. Als Best Practice für ADF-Applikationen hat sich ergeben, die JVM-Parameter „Xms" (initialer Heap Size) = „Xmx" (maximaler Heap Size) zu setzen, um das Nachladen von Speicherblöcken auf dem Arbeitsspeicher des Servers zu vermeiden.

Ein weiteres wichtiges Thema im Bereich „Speicherverwaltung" ist die Konfiguration der Garbage Collection (GC). Für ADF sind insbesondere die Parameter „XX:NewRatio" und „XX:SurvivorRatio" interessant, da sich diese mit den jeweiligen Teilbereichen des permanenten Heap auseinandersetzen, der „Young Generation" und der „Old Generation" *(siehe Abbildung 4)*.

```
Desired survivor size 255379422 bytes, new threshold 55 (max 55)
- age     1:  203476204 bytes,      203476204 total
- age     2:     4037572 bytes,     207513776 total
- age     3:    39200462 bytes,      246714238 total
```

Listing 2: Ausgabe der JVM Garbage Collection

Die Old Generation enthält Objekte, die entweder initial zu groß für die Young Generation gewesen sind, oder Objekte, die eine bestimmte Anzahl an GCs überlebt haben. Während einer GC ist der Young-Generation-Anteil der spannendere: Neu instanziierte Objekte sind generell in „Eden" gespeichert, überlebende Objekte in einem der beiden „Bands", sodass das jeweils andere leer bleibt. Es entstehen also ein Content Band und ein Copy Band.

Sind Objekte bei der GC nicht mehr erforderlich, werden sie einfach gelöscht und Speicher freigegeben. Sind in Eden und dem Content Band noch Referenzen aktiv, so werden diese Objekte als Survivor markiert und in dem Copy Band gespeichert. Sollte das Copy Band noch nicht gefüllt sein, tauschen Copy Band und Content Band die Rollen für die nächste GC. Ist das Band voll, werden die Objekte mit der längsten Lebenszeit in die Old Generation verschoben. Das Löschen von Objekten in der Old Generation ist aufgrund des fehlenden Copy Bands komplexer, da bei jedem Aufräumen zusätzlich eine Defragmentierung des Bereichs erfolgt.

„XX:NewRatio" stellt die Größe der Young Generation in Relation zur Old Generation dar. Ein Wert von „drei" ergibt, dass die Old Generation drei Mal so groß ist wie die Young Generation. Das Minimum ist eins, da die Old Generation mindestens so groß sein muss wie die Young Generation.

Die SurvivorRatio gibt die Relation von Eden zum Survivor-Bereich an. In ADF werden viele Java-Objekte einem Benutzer zur Session zugeordnet. Sie haben also im Allgemeinen einen hohen Lebenszyklus, während andere nur für einen Request benutzt und direkt danach wieder aufgeräumt werden sollten. Es ergibt also Sinn, die Session-Objekte in den Langzeit-Speicher zu verschieben. Daher sollte man, um die GC-Zyklen so kurz wie möglich zu halten, die Young Generation der JVM eher kleiner einkalkulieren.

Um eine Übersicht über die GC Zyklen zu bekommen, sollte in der Entwicklungsumgebung der Parameter „XX:+PrintTenuringDistribution" benutzt werden. Dieser sorgt dafür, dass bei jeder GC Informationen darüber in das Log geschrieben werden, wie lange sich Elemente schon in der JVM befinden und wie viele GC-Durchläufe sie schon überlebt haben. *Listing 2* gibt ein Beispiel:

Die Ausgabe zeigt, dass ca. 200 MB an Objekten einen GC-Zyklus überlebt haben, ca. 4 MB zwei Zyklen und ungefähr 40 MB schon drei Zyklen. Die Survivor Size ist 256 MB, daher ist es im nächsten GC-Zyklus nicht notwendig, Daten in die Old

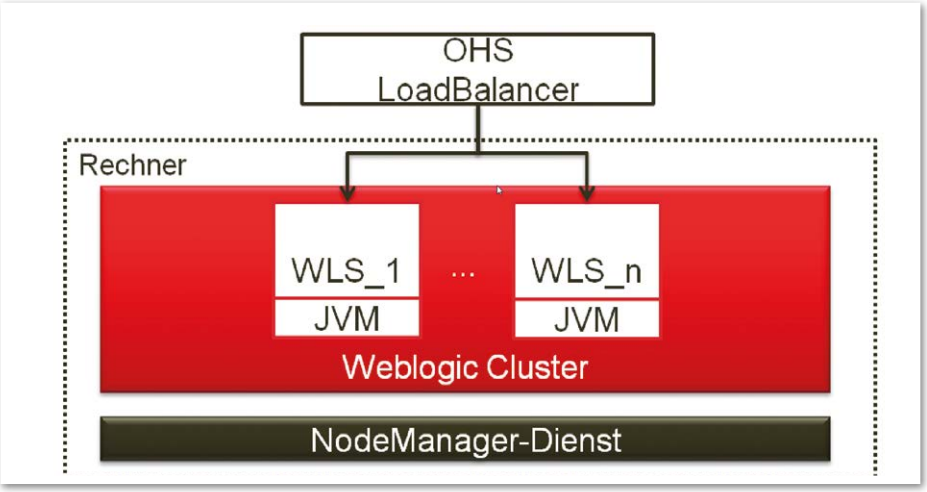


Abbildung 5: WLS Cluster für Performance-Optimierung

Generation zu verschieben. Aus der Erfahrung haben sich folgende JVM-Parameter für ADF-Web-Applikationen bewährt:

- Xmx = Xms
- XX:NewRatio 3
- XX:SurvivorRatio 6
- XX:+UseISM (allokiert 4 MB Blöcke statt 8 KB Blöcken)
- XX:+AggressiveHeap (dynamische Vergrößerung der maxHeap; nicht verwenden mit Xmx-Parametern)

Wenn nichts mehr geht – WebLogic Cluster

Es kann Fälle geben, in denen weder an der Applikation noch am Deployment oder an den Server-Einstellungen etwas optimiert werden kann. Eventuell ist die Last der User auf einem Server und auf einer Applikation einfach zu hoch. Spätestens jetzt muss man Fragen nach der Replikation stellen und Oracle bietet mit dem Oracle WebLogic Cluster einen mächtigen Baukasten für diverse Herausforderungen.

Um die Performance auf einem System mithilfe einer einzelnen Hardware zu verbessern, kann die Applikation durch eine simple Lastverteilung über einen Oracle-HTTP-Server auf einen Cluster von Managed-WebLogic-Servern repliziert werden *(siehe Abbildung 5)*.

Fazit

Durch einfache Konfiguration über die WebLogic-Konsole oder WLST kann man in kürzester Zeit eine (der User entsprechenden) Anzahl an JVMs auf einem Rechner zusammenschließen. Allerdings muss man fairerweise sagen, dass das auch seinen Preis hat (WebLogic Enterprise Edition).

Das Thema „Performance-Optimierung" ist ein spannendes Thema, weil in jedem Projekt individuelle Anforderungen an Applikationen, Deployments und Server gestellt werden. Der Artikel hat einen kleinen Einblick in die Vielfältigkeit der Optionen zur Performance-Optimierung gegeben.

Markus Klenke
mke@team-pb.de

12c-New-Features im Praxis-Einsatz

Roger Troller, Trivadis AG

Zwei Jahre sind vergangen, seit Oracle die Version 12*c* ihrer Datenbank ausgeliefert hat. Die ersten Unternehmen haben den Schritt zum produktiven Einsatz der jüngsten Datenbank-Version gewagt und es ist an der Zeit, ein erstes Fazit zu den neuen Funktionalitäten im Bereich „SQL & PL/SQL“ zu ziehen.

Die folgende Zusammenstellung ist eine persönliche Einschätzung des Autors zu den neuen Funktionalitäten, basierend auf Erfahrungen in verschiedenen Projekten.

Invisible Columns

Mit der Möglichkeit, Spalten zu verstecken, wird ein älteres Feature von Oracle (virtuelle Spalten) enorm aufgewertet. Bisher musste man beim Einsatz von virtuellen Spalten mit verschiedenen Problemen kämpfen wie zum Beispiel:

- Virtuelle Spalten sind in „%ROWTYPE“-Variablen enthalten, wodurch keine „ROW“-Operationen möglich sind *(siehe Listing 1 und 2)*
- Insert ohne Angabe der Spalten (Ausschluss der virtuellen Spalten) führt zu Fehlern, wenn die Ziel-Tabelle virtuelle Spalten enthält

Die Möglichkeit, eine virtuelle Spalte unsichtbar zu definieren, umgeht diesen Fehler.

Ein wenig störend an den unsichtbaren Spalten ist, dass man keine „%TYPE“-Deklaration auf einer solchen Spalte durchführen kann *(siehe Listing 3)*.

Row Limiting Clause

Die Row Limiting Clause stellt eine wesentliche Vereinfachung im Bereich der Paginierung und der „Top-N“-Abfragen dar. Es ist nicht so, dass diese Problemstellungen vorher nicht lösbar waren, aber die mehrstufigen Ansätze unter Verwendung von „ROWNUM“ *(siehe Listing 4)* oder analytischen Funktionen *(siehe Listing 5)* werden durch die Row Limiting Clause überflüssig.

Ein Vergleich herkömmlicher Lösungen, um die zweite Seite eines Produktkatalogs (Produkt 11-20, nach „product_id“ sortiert) anzuzeigen, veranschaulicht mit einer 12*c*-Implementation, wie viel einfacher sich solche Abfragen zukünftig mit dem neuen Feature definieren lassen. Mit der Row Limiting Clause fällt die Notwendigkeit der geschachtelten Abfragen weg. Es werden eine Sortierung, eine Anzahl zu überspringende Datensätze sowie eine Anzahl der zu lesenden Datensätze definiert *(siehe Listing 6)*.

Neben der verbesserten Lesbarkeit des SQL-Statements erhalten wir mit der Row Limiting Clause auch die Möglichkeit, prozentuale „Top-N“-Abfragen (die ersten 5 Prozent der Datensätze anzeigen, *siehe Listing 7)* sowie eine Behandlung von Duplikaten *(siehe Listing 8)* an der Anzeigegrenze durchzuführen.

Zusammengefasst vereinfacht die Row Limiting Clause die Formulierung von „Top-N“- und Paginierungs-Abfragen wesentlich und erhöht dadurch sowohl die Wart- wie auch die Lesbarkeit solcher Queries.

Pattern Matching

Die neue Mustererkennungs-Funktionalität hat angesichts der Tatsache, dass in den Daten an verschiedensten Orten Muster zu finden sind, das Potenzial, eine Nische zu besetzen.

Beispiele für ein Muster in den Daten sind:

- *Finanzsektor*
  Aktienkurse, Fraud-Detection, Geldwäsche
- *Handel*
  Kaufgewohnheit, Preisentwicklung
- *Telekom*
  Netzaktivität, Ticketing
- *Verkehr*
  Sensorketten, Prozessketten

```
ALTER TABLE emp ADD total_income
   GENERATED ALWAYS AS (nvl(sal,0) + nvl(comm,0));

CREATE PROCEDURE upd_emp (in_emp_row IN emp%rowtype)
is
BEGIN
   UPDATE emp
       SET ROW = in_emp_row
   where empno = in_emp_row.empno;
END upd_emp;
/

DECLARE
   r_emp emp%rowtype;
BEGIN
   SELECT *
       INTO r_emp
       FROM emp
       WHERE empno = 7839; -- King

   r_emp.sal := r_emp.sal * 1.1;

   upd_emp(in_emp_row => r_emp);
end;
/
ORA-54017: UPDATE operation disallowed on virtual columns
```

Listing 1: „UPDATE SET ROW“ mit einer virtuellen sichtbaren Spalte

Viele der über die „Pattern Matching"-Klausel lösbaren Fragestellungen werden heute mit Joins, Subqueries, Pipelined Functions etc. gelöst. Dies dürfte erfahrungsgemäß auch für einige Jahre so bleiben, da neue Ansätze es immer schwer haben, Nutzer und Fürsprecher zu finden. Ein Einsatzgebiet der „Pattern Matching"-Klausel ist das Suchen nach fortlaufenden Bereichen beziehungsweise das Auffinden von Lücken. *Listing 9* zeigt die herkömmliche „Tabibitosan"-Lösung und *Listing 10* die gleiche Problemstellung, gelöst über die neue „MATCH_RECOGNIZE"-Klausel.

Beim Pattern Matching sind die Muster mit dem Keyword „PATTERN" beschrieben. Im Beispiel wird definiert, dass einem Start „0 – n" Weiterführungen folgen „[PATTERN (strt cont*)]". Die Beschreibung des Musters ist syntaktisch an reguläre Ausdrucke angelehnt. Später wird in der „MATCH_RECOGNIZE"-Klausel definiert, woran man eine Weiterführung erkennt „[DEFINE cont AS id = PREV(id) + 1]". Die Aussage hinter dieser Definition ist, dass eine Weiterführung dann vorliegt, wenn der aktuelle Wert von „id" den Wert der „id"-Spalte des vorhergegangenen Datensatzes um „1" übersteigt. Ebenfalls kann definiert werden, welche Werte („MEASURES") ermittelt und wie viele Resultatzeilen pro gefundenes Muster zurückgegeben werden sollen „[ONE ROW PER MATCH]".

Seine wahren Stärken kann „MATCH _RECOGNIZE" ausspielen, wenn es darum geht, komplexere Muster zu erkennen und zusammenzuführen. Mittelfristig dürfte diese Klausel jedoch ein Schattendasein führen, wie etwa die „Model"- oder „Pivot"-Klauseln, die zwar vorhanden, aber kaum bekannt sind.

```
ALTER TABLE emp ADD total_income INVISIBLE
   GENERATED ALWAYS AS (nvl(sal,0) + nvl(comm,0));

DECLARE
   r_emp emp%rowtype;
BEGIN
      SELECT *
          INTO r_emp
          FROM emp
      WHERE empno = 7839; -- King

   r_emp.sal := r_emp.sal * 1.1;

   upd_emp(in_emp_row => r_emp);
END;
/
anonymous block completed
```

Listing 2: „UPDATE SET ROW" mit einer virtuellen unsichtbaren Spalte

```
ALTER TABLE emp MODIFY comm invisible;

DECLARE
   l_comm emp.comm%type;
BEGIN
   NULL;
END;
/

Error report -
ORA-06550: line 2, column 15:
PLS-00302: component 'COMM' must be declared
```

Listing 3: „%TYPE"-Deklaration gegen eine unsichtbare Spalte

```
SELECT y.product_id, translated_name
   FROM (SELECT ROWNUM rn
               ,x.*
          FROM (SELECT product_id
                      ,translated_name
                  FROM product_descriptions
                 WHERE language_id = 'US'
                 ORDER BY product_id) x
      ) y
 WHERE y.rn BETWEEN 11 AND 20
/
```

Listing 4: Paginierung mittels „ROWNUM"

```
SELECT y.product_id, translated_name
   FROM (SELECT product_id
          ,translated_name
          ,row_number() OVER (ORDER BY product_id) rn
      FROM product_descriptions
      WHERE language_id = 'US') y
 WHERE y.rn BETWEEN 11 AND 20
 /
```

Listing 5: Paginierung durch analytische Funktion

```
SELECT product_id
       ,translated_name
   FROM product_descriptions
 WHERE language_id = 'US'
 ORDER BY product_id
 OFFSET 10 ROWS FETCH NEXT 10
ROWS ONLY
/
```

Listing 6: Pagination mit Row Limiting Clause

```
SELECT product_id
       ,translated_name
   FROM product_descriptions
 WHERE language_id = 'US'
 ORDER BY product_id
FETCH FIRST 5 PERCENT ROWS ONLY
/
```

Listing 7: Prozentuale „Top-N"-Abfrage

UTL_CALL_STACK

Mit „UTL_CALL_STACK“ erhalten wir die Möglichkeit, auf den Namen der Program Unit und der darin aufgerufenen Prozedur/Funktion bis hinunter zu lokalen Prozeduren und Funktionen zuzugreifen. Die Verwaltung geschieht innerhalb von Oracle in einem mehrdimensionalen Feld, wobei auf der Y-Achse die Aufrufsequenz (letzte aufgerufene Program Unit auf Position 1, die vorherigen mit einer entsprechend höheren Nummer) und auf der X-Achse die Program-Unit-Struktur abgelegt ist, in der wir uns befinden *(siehe Listing 11)*. In diesem Beispiel würde der Call-Stack zu dem Zeitpunkt, zu dem wir uns in der lokalen Prozedur befinden, wie in *Listing 12* aussehen.

Wichtig dabei ist, dass sich die aktuelle Position innerhalb des Call-Stacks immer auf Position 1 befindet. Dadurch lässt sich einfach eine Library-Funktion erzeugen, die es erlaubt, die aktuelle Position als String zurückzugeben *(siehe Listing 13)*. Als aktuelle Position wird dabei der Ort zurückgegeben, an dem die Library-Funktion aufgerufen wird (somit Level 2).

Diese Funktion macht sich das Wissen zunutze, dass sich der Aufrufer der Funktion im Call-Stack auf Level 2 befindet. Dadurch kann nun innerhalb der eigenen Program Units diese kleine Library-Funktion aufgerufen werden, um die aktuelle Position in Form eines Strings zu erhalten *(siehe Listing 14)*.

Der Call-Stack funktioniert auch auf gewrapptem PL/SQL-Code und leistet gute Dienste, wenn es darum geht, ein Logging oder ein applikationsspezifisches Tracing durchzuführen, bei dem der Zugriff auf den Namen der aktuell ausgeführten Program Unit erforderlich ist.

ACCESSIBLE_BY Clause

Im Oracle Magazine, Ausgabe 1/2015, hat Steven Feuerstein aufgezeigt, wie die „ACCESSIBLE_BY“-Klausel verwendet werden kann, wenn man bestehenden PL/SQL-Code („Packages“) restrukturieren muss. In dem Artikel „When Packages Need to Lose Weight“ beschrieb er die Trennung eines Package in zwei – unter Beibehaltung der zuvor privaten Methoden, die nun neu von zwei verschiedenen (neuen) Packages benutzt, aber nur einmal definiert und gleichzeitig vor anderen unbe-

```
SELECT product_id
      ,translated_name
  FROM product_descriptions
 WHERE language_id = 'US'
 ORDER BY product_id
FETCH FIRST 5 PERCENT ROWS ONLY
/
```

Listing 7: Prozentuale „Top-N”-Abfrage

```
SELECT first_name, last_name, salary
  FROM employees
 ORDER BY salary DESC
 FETCH FIRST 2 ROWS WITH TIES
/
FIRST_NAME           LAST_NAME                 SALARY
-------------------- ------------------------- ----------
Steven               King                      24000
Neena                Kochhar                   17000
Lex                  De Haan                   17000
```

Listing 8: Behandlung von Duplikaten, Anzeige von drei Datensätzen bei „FIRST 2“

```
WITH data (id) AS (SELECT column_value
                   FROM TABLE(nc(1,2,3,5,6,7,11,12,13)))
SELECT MIN(id) start_of_range
      ,MAX(id) end_of_range
  FROM (SELECT id
              ,id - ROW_NUMBER() OVER (ORDER BY id) distance
          FROM data)
 GROUP BY distance
 ORDER BY distance
/
```

Listing 9: „Tabibitosan“-Lösung (Quelle: https://community.oracle.com/thread/1007478?start=0&tstart=0)

```
WITH DATA (ID) AS (SELECT COLUMN_VALUE
                   FROM TABLE(nc(1,2,3,5,6,7,11,12,13)))
SELECT *
  FROM data
      MATCH_RECOGNIZE(ORDER BY id
                      MEASURES FIRST(id) start_of_range
                             ,LAST(id)end_of_range
                      ONE ROW PER MATCH
                      PATTERN (strt cont*)
                          DEFINE cont AS id = PREV(id)+1
                          )
/
START_OF_RANGE END_OF_RANGE
-------------- ------------
             1            3
             5            7
            11           13
```

Listing 10: Pattern Matching Clause für fortlaufende Nummern

```
Anonymer Bock
  Packaged Procedure (TEST_PCK.
PROC1)
      Local Procedure    (TEST_
PCK.PROC1.LOCAL_PROC)
```

Listing 11: Beispiel einer Aufrufsequenz

```
Level Element 1  Element 2  Ele-
ment 3
1 TEST_PCK   PROC1   LOCAL_PROC
2 TEST_PCK   PROC1
3 __anonymous_block
```

Listing 12: Call-Stack für Aufrufsequenz von Listing 11

rechtigten Zugriffen geschützt werden sollen. Da dies zur Folge hat, dass die ehemals privaten Methoden neu öffentlich in einem Library-Package deklariert sein müssen, kommt die „ACCESSIBLE_BY"-Klausel zum Einsatz, um unberechtigte Zugriffe auf das neue Library-Package zu verhindern.

```
CREATE OR REPLACE PACKAGE BODY TVDUTIL
IS
   FUNCTION whocalled RETURN VARCHAR2
   IS
   BEGIN
       RETURN sys.utl_call_stack.concatenate_subprogram(
                      sys.utl_call_stack.subprogram(2));
   END whocalled;
END TVDUTIL;
```

Listing 13: Library-Funktion für die Rückgabe der aktuellen Position im Call-Stack

```
CREATE OR REPLACE PACKAGE BODY test_pck
IS
   PROCEDURE proc1 IS
       PROCEDURE local_proc IS
       BEGIN
           sys.dbms_output.put_line('[2] : '
                                    || tvdutil.whocalled());
       END local_proc;
   BEGIN
       sys.dbms_output.put_Line('[1] : '
                                    || tvdutil.whocalled());
       local_proc();
        END proc1;
END;
/

BEGIN
   test_pck.proc1();
END;
/

[1] : TEST_PCK.PROC1
[2] : TEST_PCK.PROC1.LOCAL_PROC
```

Listing 14: Verwendung der Library-Funktion

```
CREATE TABLE employees (
   emp_id     NUMBER DEFAULT ON NULL emp_seq.nextval NOT NULL
   ,last_name VARCHAR2(30)
);
```

Listing 15: DEFAULT ON NULL

```
CREATE TABLE employees (
   emp_id     NUMBER GENERATED ALWAYS AS IDENTITY
   ,last_name VARCHAR2(30)
);
```

Listing 16: Identity Column (hier „generated always")

Identity Columns/Default ON NULL

Wichtiger als die neuen „Identity Columns" ist der Umstand, dass es nun möglich ist, als Default-Wert einer Spalte eine Sequenz zu hinterlegen und zu steuern, dass der Default-Wert auch dann verwendet werden soll, wenn „NULL" für eine Spalte übergeben wird. Diese Möglichkeit wertet die Default-Klausel extrem auf. Dadurch kann man öfter die Default-Zuweisung über diese Klausel statt über einen Trigger erfolgen lassen *(siehe Listing 15)*.

Den größten Nutzen, den wir aus den Identity Columns und der Möglichkeit, Sequenzen als Default-Wert zu verwenden, ziehen können, ist der Zeitgewinn gegenüber der herkömmlichen Lösung über Sequenzen und Datenbank-Trigger. Dieser ist dem Umstand geschuldet, dass die Context Switches zwischen SQL und PL/SQL (Trigger) wegfallen. Zudem reduziert sich der Implementationsaufwand, da weder ein Trigger noch eine Sequenz erstellt werden muss *(siehe Listing 16)*.

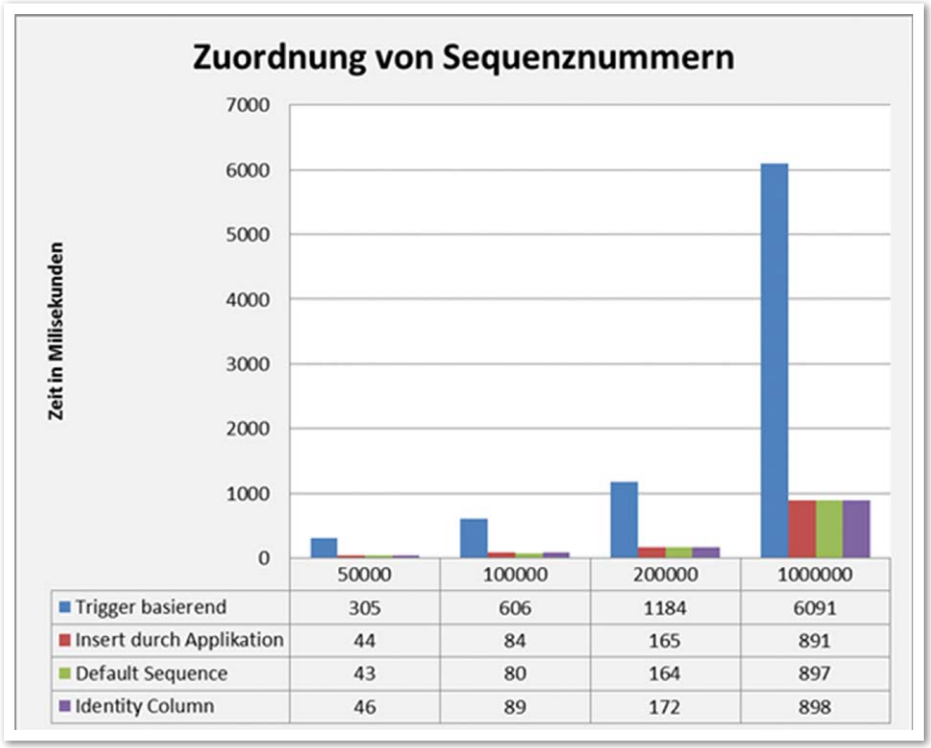
Der folgende Vergleich zeigt vier verschiedene Szenarien dafür, wie ein Datenbank-Feld mit einem Sequenz-Wert gefüllt werden kann:

- *Trigger-Lösung*
  In einem „PRE INSERT"-Trigger auf Row-Ebene wird die Sequenz gelesen und dem Datenbank-Feld zugewiesen
- *Applikatorischer Insert*
  Die Applikation liest beim Einfügen den nächsten Sequenzwert und liefert diesen beim Insert mit
- *Sequenz als Default-Wert*
  Die Sequenz ist als „Default ON NULL" in der Datenbank-Spalte hinterlegt
- *Identity Column*
  Wir bedienen uns einer Identity Column (technisch identisch mit Punkt 3)

Beim vorliegenden Test wurden 50.000, 100.000, 200.000 und eine Million Datensätze erzeugt. Die Auswertung zeigt, dass die Varianten 2 bis 4 vergleichbare Performance erreichen, während die Variante 1 (Trigger) deutlich hinterherhinkt *(siehe Abbildung 1)*.

Fazit

Auch wenn die Datenbank-Version 12*c* nicht als Entwickler-Release in die Geschichte eingehen wird, sind etliche Features enthalten, die wir in Zukunft nützen können. Neben den hier aufgeführten gäbe es viele weitere interessante Neuerungen, die es wert wären, genauer beleuchtet zu werden:



	50000	100000	200000	1000000
Trigger basierend	305	606	1184	6091
Insert durch Applikation	44	84	165	891
Default Sequence	43	80	164	897
Identity Column	46	89	172	898

Abbildung 1: Gegenüberstellung der Performance

- Erweiterungen im Bereich der Partitionierung
- „WITH"-Klausel mit PL/SQL-Support
- Integration von PL/SQL-Datentypen in dynamisches SQL
- In-Memory-Funktionalität

Roger Troller
roger.troller@trivadis.com

Umgebungswechsel vermeiden

Jürgen Sieben, ConDeS GmbH & Co. KG

Performance-Tuning ist ein weites Feld mit vielen Facetten. Doch es gibt Best Practices als Grundlage für weitere Tuning-Maßnahmen. Oft werden diese nicht ausreichend berücksichtigt, sondern die Hoffnung auf eher exotische Optimierungen gelegt. Eine besondere Rolle fällt in diesem Zusammenhang den Umgebungswechseln zwischen SQL und PL/SQL zu. Sie sind – meist unbemerkt – für erhebliche Einbußen der Performance verantwortlich. Dieser Artikel zeigt die grundlegenden Mechanismen auf und erläutert, wie ungewollte Umgebungswechsel erkannt und vermieden werden können.

Seit vielen Versionen der Datenbank enthält die Programmiersprache PL/SQL keine eigene SQL-Implementierung mehr, sondern ruft die SQL-Implementierung der Datenbank auf, wenn entsprechende Anweisungen im Code auftreten. Wechselt die Kontrolle von PL/SQL zu SQL oder umgekehrt, entsteht ein Umgebungswechsel, der aufgrund der Einrichtung der Umgebungsvariablen etc. einen erheblichen Aufwand für die Datenbank darstellt.

Diese Umgebungswechsel haben enormen Einfluss auf die Gesamtleistung der Anwendung und sollten daher so selten wie möglich auftreten. Vor der Vermeidung steht jedoch das Erkennen von Umgebungswechseln und das kann sehr einfach, aber auch sehr vertrackt sein. Zunächst die einfachen Beispiele. Beim Umgebungswechsel von SQL nach PL/SQL sind es vor allem zwei Szenarien, die immer wieder anzutreffen sind:

- Aufruf von PL/SQL-Funktionen aus SQL
- Zeilen-Trigger auf Tabellen

Listing 1 zeigt ein einfaches Beispiel für einen Funktionsaufruf aus SQL. Stellvertretend für alle PL/SQL-Funktionen wird hier

```
select *
  from emp
  where deptno = v('P10_DEPTNO');
```

Listing 1

```
select *
  from emp
  where deptno = (select v('P10_DEPTNO') from dual);
```

Listing 2

```
  with params as(
      select v('P10_DEPTNO') deptno,
          v(('P10_JOB') job
      from dual)
select e.*
  from emp e natural join params p;
```

Listing 3

```
create or replace trigger trg_emp_briu
before insert or update on emp
for each row
declare
  l_now date := sysdate;
begin
  :new.empno := coalesce(:new.empno, emp_seq.nextval);
  :new.ename := upper(:new.ename);
  if updating then
      :new.empno := :old.empno;
  end if;
end;
/
```

Listing 4

die Funktion „v“ gezeigt. Sie ist im Schema „APEX_nnnnn“ definiert und bietet dem Entwickler Zugriff auf Informationen, die sich im Session State von Apex befinden. Der Parameter „P10_DEPTNO“ bezeichnet ein Element auf einer Apex-Anwendungsseite, er ist eine Konstante und die Funktion mit Sicherheit für die Dauer der Abfrage deterministisch, das heißt, sie wird für den gleichen Eingangsparameter im Kontext dieser „select“-Abfrage das gleiche Ergebnis liefern. Sollte SQL nicht so clever sein, die Funktion nur einmal aufzurufen anstatt für jede Zeile?

Die Antwort lautet: Nein. Die Funktion selbst ist nicht deterministisch: Ein anderer Benutzer mit anderem Session-Identifier wird einen anderen Wert für das Element „P10_DEPTNO“ erhalten können, und auch zwischen zwei Abfragen innerhalb der gleichen Benutzersession können die Ergebnisse für den Funktionsaufruf mit diesem Parameter differieren. Die Datenbank weiß also nicht, welches Ergebnis die Funktion liefert, sondern muss es aktuell erfragen. Daher hätte noch nicht einmal PL/SQL die Möglichkeit, das Ergebnis im Cache zwischenzuspeichern.

Viel wichtiger ist jedoch, dass SQL nicht weiß, welche Werte die Funktion zurückliefern wird. Daher muss für jede Zeile die Funktion erneut aufgerufen und von PL/SQL neu berechnet werden. Dies ist besonders fatal, weil die Funktion in der „where“-Klausel der Abfrage verwendet wird, denn sie wird in jedem Fall für jede Zeile der Tabelle aufgerufen: Beinhaltet die Tabelle eine Million Zeilen, von denen nach der Filterung nur noch hundert übrigbleiben, hat die Anfrage dennoch eine Million Umgebungswechsel durchführen müssen. Was also tun? Die Lösung liegt darin, den Aufruf der Funktion in einer skalaren Unterabfrage zu schachteln *(siehe Listing 2)*.

Das sieht auf den ersten Blick zwar komisch aus, hat aber eine Reihe von Vorteilen:

- Weil eine skalare Unterabfrage von SQL wie eine Konstante behandelt wird, erfolgt die Berechnung nur einmal für jeden unterschiedlichen Parameter der Funktion, in unserem Fall also genau einmal.
- Es treten nach der ersten Berechnung keine weiteren Umgebungswechsel mehr auf.
- Diese Optimierung funktioniert mit deterministischen Funktionen ebenso wie mit nicht deterministischen Funktionen.

Gerade der letzte Punkt ist interessant: Die Denkweise ist, dass SQL eine Unterabfrage lesekonsistent zum Zeitpunkt der Abfrage einmal pro unterschiedlichem Parameter beantwortet und das Ergebnis der Abfrage nachfolgend als Konstante betrachtet. Daher ist es nicht erforderlich, dass die Funktion „v“ deterministisch ist; wir verwenden das Ergebnis, das zum Zeitpunkt der Abfrage geliefert wurde.

Da wir mit diesem Aufruf nun eine Million Umgebungswechsel einsparen, sinkt die Ausführungszeit der Abfrage auf die gleichen Werte (nahezu) wie bei Verwendung einer Konstanten in der Abfrage, sie wird förmlich pulverisiert und sinkt unter den sinnvoll messbaren Bereich. Ein erweiterter Lösungsansatz besteht darin, mehrere Funktionsaufrufe in einer faktorisierten Unterabfrage der eigentlichen Abfrage voranzustellen *(siehe Listing 3)*.

Das ist auch einmal eine sinnvolle Anwendung des Natural Join, der ja Spalten gleichen Namens zweier Tabellen in eine automatische „Inner Join“-Beziehung einbezieht. Da man die „PARAMS“-Abfrage über Aliase im Griff hat, kann man sich eine explizite Join-Klausel ersparen. Doch ist dieser Aufwand nötig? Haben wir nicht von „RESULT_CACHE“, „QUERY_RESULT_CACHE“ (für SQL-Abfragen) und in Version 12*c* vom Pragma „UDF“ (User Defined Function) gelesen, das speziell für Funktionen da ist, die in SQL verwendet werden sollen? Was ist mit dem Pragma „DETERMINISTIC“?

Alle diese Lösungen beziehen sich auf die jeweilige Umgebung, also auf PL/SQL oder auf SQL, und können dort unnötige Neuberechnungen ersparen. Viele dieser Optionen sind zudem an die Enterprise Edition gebunden und haben keine Auswirkung auf SE oder XE. Zudem kann keine dieser Optimierungen unnötige Umgebungswechsel minimieren; das kann nur die Kapselung in einer Unterabfrage, die zudem auch noch in jeder Datenbank-Version und -Edition funktioniert.

Grund genug für die erste Best Practice: „Funktionsaufrufe in SQL gehören in eine skalare Unterabfrage. Jedenfalls dann, wenn Sie nicht für jede Zeile ein unterschiedliches Ergebnis benötigen (zum Beispiel „dbms_random"). Diese Funktionsaufrufe müssen pro Zeile gerechnet werden und dürfen daher nicht in eine skalare Unterabfrage."

Zeilen-Trigger

Ein beliebtes Thema und für viele die Einstiegsdroge in die Programmierung von PL/SQL sind Trigger auf Tabellen, um zum Beispiel Primärschlüssel-Spalten zu berechnen, Großschreibung von Namen zu garantieren oder sonstige Datenprüfungen vorzunehmen. *Listing 4* zeigt einen einfachen Zeilen-Trigger auf die Tabelle „EMP".

Zeilen-Trigger, die also für jede betroffene Zeile einer DML-Anweisung einmal ausgeführt werden („Klausel for each row"), haben für jede Zeile einen Umgebungswechsel zwischen SQL und PL/SQL zur Folge: Der Trigger-Körper ist ein anonymer PL/SQL-Block. Wann der Trigger ausgelöst wird und auf welche Tabelle er sich bezieht, ist in SQL definiert. Die Logik wird aber in PL/SQL implementiert und die Schnittstelle zwischen beiden ist der Aufruf eines anonymen PL/SQL-Blocks für jede Zeile, für die der Trigger ausgelöst wird.

Was tun wir dagegen? Die Antwort ist einfach, die Umsetzung nicht: Den Zeilen-Trigger weglassen. Ab Version 12*c* gibt es mit der „column identity"-Klausel (endlich) einen Weg, Trigger zur Generierung neuer Primärschlüssel zu ersetzen. Das muss dann auch konsequent gemacht werden, wenn man auf Version 12*c* umsteigt und keine Rückwärts-Kompatibilität benötigt.

Wie ersetzt man den Rest? Wer die Zeilen-Trigger weglässt, verlagert die Logik in ein PL/SQL-Package. Toll, dann haben wir halt die Umgebungswechsel umgekehrt. Richtig und auch wieder nicht, denn wie der entsprechende Abschnitt zeigt, bietet ein Package immer die Möglichkeit, solche Arbeiten in einer Mengenverarbeitungsroutine in PL/SQL zunächst aufzubereiten und dann mit wenigen Umgebungswechseln als Menge an SQL zu übergeben.

Doch ein Vorteil des Triggers ist, dass an ihm keiner vorbeikommt. Ist jederzeit bekannt, welche Teile des extern programmierten Anwendungscodes auf die Tabelle schreiben? Falls nicht, aber dennoch sichergestellt werden muss, dass immer ein Primär-Schlüssel berechnet und der Name in Großbuchstaben gespeichert wird, scheint man um den Trigger nicht herumzukommen. Das könnte man nur verhindern, wenn nicht jeder beliebig auf die Tabelle schreiben, sondern nur ein bestimmtes Package nutzen darf.

Hier beginnt das Problem: Wenn man sagt: „Danke, das war's dann mit diesem Konzept, das kriegen wir nie durch", lässt sich das zwar gut nachvollziehen, es ist aber auch ein Indiz für eine zu starke Kopplung zwischen Datenbank und Anwendungscode. Sagen wir so: Es ist eine architektonische Entscheidung. Wenn man die nicht treffen möchte oder kann, sind die durch die Trigger verursachten Umgebungswechsel eine unausweichliche Folge und die damit verbundenen Performance-Probleme nur äußerst schwer in den Griff zu bekommen.

Fassen wir also zusammen: „Zeilentrigger sind mit einer performanten Anwendung nicht in Einklang zu bringen, wenn sie in Bewegungs-Tabellen verwendet werden. Je größer die Transaktionslast auf einer Tabelle, desto stärker die Begründung, auf Zeilentrigger zu verzichten und alternative Programmiermodelle zu etablieren."

Umgebungswechsel von PL/SQL nach SQL

Auch umgekehrt sind Umgebungswechsel von PL/SQL nach SQL möglich. Dazu ein Beispiel mit einem Aufruf einer DML-An-

```
create or replace procedure sql_
performance_test_A
as
begin
   for i in 1..10000 loop
       insert into test_table
values(i);
   end loop;
   commit;
end sql_performance_test_A;
/
```

Listing 5

```
create or replace procedure sql_
performance_test_B
as
   type value_table_type is table
of pls_integer index by binary_
integer;
   l_value_table value_table_
type;
   l_iterations integer := 10000;
begin
   for i in 1..l_iterations loop
       l_value_table(i) := i;
   end loop;
   forall idx in 1 .. l_itera-
tions
   insert into test_table
values(l_value_table(idx));
   commit;
end sql_performance_test_B;
/
```

Listing 6

```
scott@condes> begin
   2       Compare_Implementation('A', 'B');
   3   end;
   4   /
Run1 ran in 307 hsecs
Run2 ran in 4 hsecs
run 1 ran in 7675% of the time

Run1 latches total versus runs -- difference and pct
Run1          Run2           Diff    Pct
120,409       6,072      -114,337   1,983.02%
PL/SQL procedure successfully completed.
```

Listing 7

weisung in einer Loop. Einfach zu sehen ist dies in der Prozedur, die 10.000 Zeilen in eine Tabelle schreibt *(siehe Listing 5)*.

Diese Prozedur macht für jeden Schleifen-Durchlauf einen Umgebungswechsel. Ärgerlich daran ist, dass die elegante und einfache Art der Programmierung eigentlich suggeriert, dass man alles richtig gemacht hat. Um die Aufgabe ohne unnötige Umgebungswechsel zu lösen, wird die mengenorientierte Programmierung eingesetzt, wie im folgenden Beispiel *(siehe Listing 6)*.

Zunächst wird die PL/SQL-Tabelle „value_table" mit Daten geladen, was einen erhöhten Speicherverbrauch (schließlich müssen statt einer alle 10.000 Zeilen im Arbeitsspeicher gehalten werden), nicht aber Umgebungswechsel zur Folge hat. Anschließend wird in einer „FORALL"-Anweisung die lokal gefüllte Variable an SQL übergeben und dort in einer einzigen SQL-Anweisung in die Tabelle eingefügt.

Der Autor hat die beiden Prozeduren gegeneinander ins Rennen geschickt. Er hat dabei nicht nur die Zeit der Ausführung, sondern auch die Locks und Latches, die beide Prozeduren innerhalb der Datenbank allokieren, mit Tom Kytes „RUN_STATS"-Skript gemessen. *Listing 7* zeigt das beeindruckende Ergebnis.

Der Vergleich zeigt, dass die mengenorientierte Arbeitsweise um den Faktor 75 schneller ist, die Anzahl der Locks und Latches sinkt um den Faktor 20 (und damit steigt die Skalierbarkeit der Anwendung um diesen Faktor). Nachteil ist, wie gesagt, der erhöhte Speicherverbrauch dieser Lösung. Um diesen zu reduzieren, kann man die Anzahl der Zeilen steuern, die im Bulk an SQL übergeben werden, allerdings erfolgen nun mehr Umgebungswechsel.

Wo ist hier also die richtige Balance? Hier gibt es gute Nachrichten, denn bereits kleine Zeilenmengen haben erheblichen Einfluss auf die Geschwindigkeit. *Abbildung 1* zeigt den obigen Vergleich für die Bulk-Größen von 1 bis 128 Zeilen.

Die Ausführungszeit sinkt anfangs rapide (Angaben in hsec) und reduziert sich ab etwa hundert Zeilen nicht mehr relevant weiter. Daher ist diese Zeilenzahl ein guter Start für eigene Versuche. *Listing 8* zeigt, wie man die Bulk-Größe kontrolliert.

Auch hier die Zusammenfassung als Best Practice: „Datenbanken werden

```
create or replace procedure sql_bulk_test(
   p_bulk_size in integer)
as
   type value_table_type is table of pls_integer index by binary_inte-
ger;
   l_value_table value_table_type;
   c_iterations constant integer := 100000;
   l_idx integer := 1;
   l_value integer;
   l_time integer;
begin
   l_time := dbms_utility.get_time;
   while l_idx <= ceil(c_iterations / p_bulk_size) + 1 loop
       -- Daten in PL/SQL-Tabelle laden
       l_value_table.delete;
       for i in 1..p_bulk_size loop
           l_value := ((l_idx - 1) * p_bulk_size) + i;
           if l_value <= c_iterations then
           l_value_table(i) := l_value;
       else
       exit;
       end if;
       end loop;
       -- BULK-Insert in die Datenbank
       forall idx in 1 .. l_value_table.count
           insert /*+ append */ into test_table values(l_value_
table(idx));
       l_idx := l_idx + 1;
   end loop;
   l_time := dbms_utility.get_time - l_time;
   dbms_output.put_line('Bulk size: ' || p_bulk_size || ', Time: ' ||
l_time || 'hsec');
   execute immediate 'truncate table test_table';
end sql_bulk_test;
/
```

Listing 8

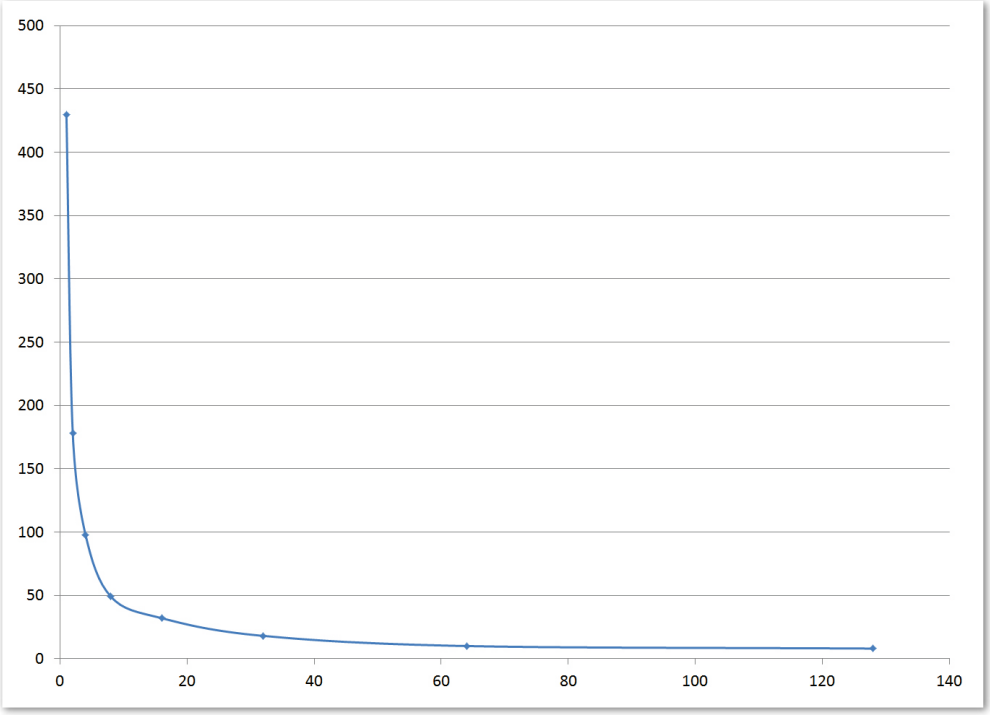


Abbildung 1: Zeitverbrauch in Abhängigkeit von der Bulk-Größe

grundsätzlich mengenorientiert programmiert und nicht zeilenorientiert. Bei einer Bearbeitung von Daten Zeile für Zeile (Tom Kyte übersetzt „row by row“ durch „slow by slow“) führen die häufigen Umgebungswechsel zu schlechter Performance. Selbst höherer Programmieraufwand ist gerechtfertigt, wenn dadurch mengenorientiert gearbeitet werden kann.“

Öffnen eines Cursors

Ebenso offensichtlich ist, dass von PL/SQL nach SQL gewechselt werden muss, wenn ein Cursor geöffnet wird. Auch hier gilt die Empfehlung, mit Mengenverarbeitung (beim Lesen von Daten allerdings mit der Anweisung „bulk collect into“) zu arbeiten, falls lediglich Daten-Strukturen in PL/SQL-Variablen umkopiert werden sollen.

Arbeitet man mit „cursor for“-Schleifen nd ist der Parameter „PLSQL_OPTIMIZE_LEVEL“ mindestens auf den Wert „2“ eingestellt, wird der Compiler beim Kompilieren des PL/SQL-Codes diese automatisch zu „bulk collect into“-Schleifen umbauen, ebenfalls mit einer reduzierten Bulk-Größe von hundert Zeilen. Ein Beispiel dafür können wir sparen, denn es handelt sich um das Standardverfahren zum Arbeiten mit Datenmengen in PL/SQL.

Interessant wird es allerdings, wenn die Situation nicht ganz so einfach ist. Man stelle sich folgenden Klassiker vor: Es wird ein Cursor benötigt, der über alle Abteilungen iteriert, um für eine Abteilung in einem zweiten Cursor alle Mitarbeiter zu bearbeiten. *Listing 9* zeigt den Code dafür.

```
declare
   cursor dept_cur is
       select deptno, dname, loc
           from dept;
   cursor emp_cur(p_deptno in number) is
       select empno, ename, job, sal
       from emp
       where deptno = p_deptno
       order by ename;
begin
   for dept in dept_cur loop
       dbms_output.put_line('Abteilung ' || dept.dname);
       for emp in emp_cur(dept.deptno) loop
           dbms_output.put_line(
           '. ' || emp.ename || ', ' || emp.job);
       end loop;
   end loop;
 end;
/
```

Listing 9

Dieser Code hat gleich zwei massive Probleme: Zum einen, das ist aus dem bisher Gesagten klar, werden nun pro Durchlauf der äußeren Schleife Umgebungswechsel für das Öffnen jedes inneren Cursors notwendig. Das allein wäre nur lästig beziehungsweise langsam. Schlimmer ist jedoch, dass der innere Cursor nicht lesekonsistent zum äußeren Cursor ist: Ändern sich die Zeilen der Tabelle „EMP“ während des Laufs des PL/SQL-Codes, sehen nachfolgende Durchläufe der inneren Schleife die geänderten Daten. Dies liegt daran, dass „select“-Anweisungen keine Sperren in der Datenbank hinterlassen und daher nicht den Regeln der Lesekonsistenz unterliegen.
Ein rabiater Ausweg wäre, vor der Ausführung des Codes den Serialization Level der Session auf „serializable“ zu stellen und anschließend explizit eine Transaktion anzufordern. Wenn jemand die Anweisungen dafür nicht kennt, dann wohl auch deshalb, weil dies bei Oracle so gut wie nie unbedingt nötig ist.

Was aber ist der Ausweg aus dieser Situation? Wieder wird die Programmierung etwas umfänglicher, aber eben auch sowohl lesekonsistent als auch schnell. Wir benötigen einen Cursor-Ausdruck in der „select“-Anweisung *(siehe Listing 10)*.

Man übernimmt explizit die Kontrolle über den Cursor („cursor for“-Schleifen können mit Cursor-Ausdrücken nicht genutzt werden) und deklariert im Cursor mit einem Cursor-Ausdruck die „n“-Seite der Abfrage. Auf diese Weise wird die gesamte Ergebnismenge in einem Roundtrip zur SQL-Seite ermittelt und lesekonsistent zur Verfügung gestellt.

Die Ergebnisse der Unterabfrage werden durch den Cursor-Ausdruck in der Schleife in einen lokalen Cursor vom Typ „SYS_REFCURSOR“ übernommen (andere Varianten wären möglich, aber komplexer) und in der äußeren Schleife in eine Variable dieses Typs. Über diesen Cursor „L_EMP_CUR“ kann nun eine zweite Schleife iterieren und die Werte ausgeben.

Die Lehre aus der Geschichte: „Ein Cursor ist ein unvermeidbarer Umgebungswechsel zwischen SQL und PL/SQL und daher nicht schlimm. Wenn aber geschachtelte Cursor verwendet werden, ist es erforderlich, die Daten in einem Roundtrip zu generieren und mit Cursor-Ausdrücken auszuwerten, um nicht in Lesekonsistenz-Probleme und schlechte Performance zu schlittern.“

Verdeckte Umgebungswechsel

Etwas weniger durchsichtig wird es, wenn Umgebungswechsel nicht offensichtlich sind. Wie auch schon in den gezeigten Beispielen liegt die Gefahr darin, dass die Programmierung verdeckter Umgebungswechsel mit elegant erscheinendem Code möglich ist und daher für richtig erachtet wird.

Dabei fällt vor allem die Instanziierung von Objekttypen beziehungsweise die Verwendung von „member“-Funktionen dieser Typen auf. Hat man eigene Objekt-Typen erstellt und hierfür Konstruktor- oder „member“-Funktionen angelegt, wird der Gebrauch dieser Funktionen zu Umgebungswechseln führen. Das ist schwer zu verhindern, am ehesten noch dadurch, dass man die Objekte in PL/SQL erstellt und mit „bulk“-Operationen an SQL übergibt. Ob der Aufwand hierfür gerechtfertigt ist, ist allerdings von Fall zu Fall zu entscheiden.

Etwas hinterhältiger verhält sich PL/SQL. Zum einen gibt es Funktionen, die eine Fallback-Lösung auf eine SQL-Abfrage besitzen und daher potenziell für Umgebungswechsel sorgen können. Dazu als Beispiel einmal die Implementierung der Funktion „sysdate“ im Package „STANDARD“ *(siehe Listing 11)*.

Die Funktion ist als externe Funktion „PESSDT“ in C implementiert. Sollte dies aber aus irgendeinem Grund nicht gehen

```
declare
   cursor dept_cur is
       select dname,
              cursor(
              select ename, job
                  from emp e
              where e.deptno = d.deptno
              order by e.ename) emp_cur
          from dept d;
   l_dname dept.dname%type;
   l_emp_cur sys_refcursor;
   l_ename emp.ename%type;
   l_job emp.job%type;
begin
   open dept_cur;
   loop
       fetch dept_cur into l_dname, l_emp_cur;
       exit when dept_cur%notfound;
       dbms_output.put_line('Abteilung ' || l_dname);
       loop
           fetch l_emp_cur into l_ename, l_job;
           exit when l_emp_cur%notfound;
           dbms_output.put_line(
           '. ' || l_ename || ', ' || l_job);
       end loop;
   end loop;
end;
/
```

Listing 10

```
function pessdt return DATE;
      pragma interface (c,pessdt);
   -- Bug 1287775: back to calling ICD.
   -- Special: if the ICD raises ICD_UNABLE_TO_COMPUTE, that means we
should do
   -- the old 'SELECT SYSDATE FROM DUAL;' thing.      This allows us to
do the
   -- SELECT from PL/SQL rather than having to do it from C (within the
ICD.)
   function sysdate return date is
       d date;
   begin
       d := pessdt;
       return d;
   exception
       when ICD_UNABLE_TO_COMPUTE then
           select sysdate into d from sys.dual;
       return d;
   end;
```

Listing 11

(„exception ICD_UNABLE_TO_COMPUTE“), erfolgt „the old 'SELECT SYSDATE FROM DUAL' thing“, also ein Umgebungswechsel zu SQL. Das ist vielleicht etwas exotisch, aber es spricht dafür, auch „sysdate“ nicht in einer Schleife zu verwenden, sondern lieber eine lokale Variable „l_now date := sysdate;“ zu vereinbaren und in der Methode zu benutzen. Auf diese Weise ist auch sichergestellt, dass alle Iterationen einer Schleife zur gleichen Zeit ausgeführt werden (so dies denn nicht ausdrücklich vermieden werden soll).

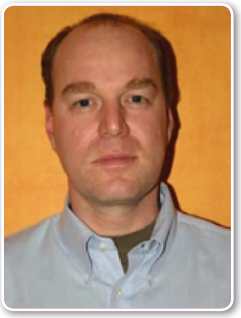
Wichtiger ist sicher noch das Beispiel der Funktion „USER“, die viele für eine Pseudo-Spalte halten. Das ist allerdings eine „select“-Abfrage gegen die Datenbank, wie ein Blick in das Package „STANDARD“ zeigt. Daher verbietet sich der Einsatz von „USER“ in Schleifen, zumal es doch eher unwahrscheinlich ist, dass sich der angemeldete Datenbank-Benutzer während eines laufenden PL/SQL-Blocks ändert. Auch hier lautet die Empfehlung, sich den aktuell angemeldeten Benutzer als Konstante in den Code zu holen, vielleicht in einem Konstanten-Package, das den Namen beim Initialisieren einmalig erfragt.

Fazit

Umgebungswechsel sind ein ganz wesentliches Problem der Programmierung in PL/SQL und der Abfrage-Gestaltung in SQL. Daher sollte man einen Blick für mögliche Umgebungswechsel entwickeln. Umgebungswechsel sind natürlich auch über die bekannten Hilfsmittel „TKPROF“ oder den hierarchischen Profiler „HPROF“ zu finden.

Die in diesem Artikel zusammengetragenen Best Practices sind nicht umfassend, nicht einmal bezüglich der Umgebungswechsel, stellen jedoch einen wesentlichen Teil der Strategien dar, die ein PL/SQL-Entwickler im Hinterkopf haben sollte, um grundsätzlich performanten und skalierbaren Code zu schreiben.

Performance-Tuning beginnt ja nicht erst, wenn ein Problem existiert, sondern manifestiert sich zunächst einmal darin, dass die groben und bekannten Fehler von vornherein vermieden werden. Dies gilt insbesondere für Code, von dem man annimmt, dass er nicht Performance-kritisch sei und von daher ja auch nachlässiger programmiert werden könnte. Erfahrungsgemäß lebt solcher Code ewig und zwar nach Murphys Gesetz genau dort, wo er erheblich im Wege steht.



Jürgen Sieben
j.sieben@condes.de

Oracle SQL – das umfassende Handbuch

gelesen von Christian Piasecki

1.011 Seiten und 2 kg Gewicht: So viel muss ein Buch heutzutage aufbieten, um einen guten Überblick über Oracle SQL zu verschaffen.

Genau dies gelingt dem Autor Jürgen Sieben durch sein spiralartiges Konzept im Aufbau des Buchs. Einsteiger werden zuallererst mit den Grundlagen der Oracle-Datenbank und den Basics von Oracle-SQL vertraut gemacht. Spätestens nach dem Lesen der ersten zehn Kapitel rund um die „SELECT"-Anweisung mit Themen wie „Auswahl und Projektion", „Sortieren von Daten", „Join von Tabellen", „Nutzen von Funktionen" (Zeilen-, Gruppen-, analytische Funktionen) und „Unterabfragen" sind auch absolute Neulinge in der Lage, Abfragen für die Oracle-Datenbank zu erstellen, die für viele Einsatz-Zwecke im Alltag ausreichen.

Sind diese Grundlagen zum Abfragen von Daten erstmal vorhanden, wird der Leser an die Daten-Manipulation herangeführt. Hier merkt man besonders, dass der Autor nicht nur theoretische Kenntnisse hat, sondern Praxiserfahrung aus fünfzehn Jahren in Oracle-Projekten mit in das Buch einfließen lässt. So werden unter anderem nicht nur Befehle wie „insert" oder „update" erläutert, sondern auch Statements wie „insert first", „insert all" „merge", die ein hohes Maß an Funktionalität mit sich bringen, aber vielen alten Hasen oft nicht bekannt sind.

Auch in den Kapiteln rund um das Thema „Erstellen von Datenobjekten" ergänzt Jürgen Sieben die fachlich guten und korrekten Ausführungen immer wieder mit Beispielen aus der Praxis, sodass die Anwendung der einzelnen Befehle und Funktionen klar ist und eine Einschätzung dessen, was in der Praxis relevant und was nachgeschlagen werden kann, erleichtert wird.

Doch nicht nur für SQL-Einsteiger ist das Buch gedacht, denn es werden viele Themen behandelt, die auch erfahrene SQL-Entwickler vielleicht noch nicht wahrgenommen haben, weil sie diese im Alltag bis jetzt nicht gebraucht haben oder den Mehrwert dieser erst in den letzten Datenbank-Versionen eingeführten Features nicht kennen. So werden brandneue Möglichkeiten der Datenbank-Version 12*c* wie Arbeiten mit JSON oder „Row Pattern Matching" genauso detailliert betrachtet wie die alten Themen „Model Clause", „With-Abfrage", „XML in der DB", „Pivot" und „Unpivot". Die hier aufgeführten Themen sind nur ein Auszug aus dem Buch und werden durch viele andere abgerundet, die das Buch zu einer ganzen Sache machen.

Fazit

Der lebendige Stil des Autors und seine guten, detaillierten Ausführungen empfehlen das Buch nicht nur als Referenz oder für Einsteiger, sondern für jeden, der auch gerne mal querliest und sein Wissen erweitern will. Wer ein Buch zum Thema „Oracle SQL" in deutscher Sprache und vor allem passend zur Datenbank-Version 12*c* sucht, kann hier bedenkenlos zugreifen.

Autor: Jürgen Sieben
Titel: Oracle SQL – das umfassende Handbuch
Verlag: Rheinwerk Computing
Umfang: 1.011 Seiten
Preis: 69,90 Euro
ISBN: 978-3-8362-3717-8

Christian Piasecki
cpiasecki@pitss.com

Data Subsetting – konsistente Daten für Testsysteme

Oliver Gehlert, Ventum Consulting GmbH & Co KG

Für Test- und Entwicklungssysteme sind konsistente Daten erforderlich. Die einfachste und aus Sicht eines Entwicklers beste Variante wäre eine komplette Kopie der Produktionsdaten. Aus Kosten- und Datenschutzgründen ist dieses Vorgehen jedoch selten.

Da Test- und Entwicklungssysteme meist schwächer dimensioniert sind als Produktivsysteme, können sie nur eine Teilmenge der Produktivdaten aufnehmen. Dabei ist die Konsistenz der Daten für eine sinnvolle Nutzung unumgänglich. Dieser Artikel stellt unterschiedliche Vorgehensweisen für die Erzeugung konsistenter Teilmengen vor. Das Thema „Datenschutz" wird nur am Rande betrachtet.

Entwickler und Tester benötigen konsistente Daten, um entwickeln beziehungsweise testen zu können. Nicht in allen Fällen ist es möglich, die gesamte Datenbasis zu kopieren. Stattdessen werden passende Subsets benötigt. Die Anforderungen an die Subsets unterscheiden sich je nach Anwendungsfall, Beispiele sind:

- Ein Entwickler benötigt zehn Prozent der Daten
- Für Tests werden alle Daten für einen Kunden/ein Land benötigt
- Um einen Bug nachzustellen, ist ein spezifischer Teil der Daten erforderlich

Für den Vergleich der Methoden werden wir das SH-Schema verwenden. Dieses bildet zwar nicht alle, aber die meisten Relationen zwischen den Tabellen durch Foreign Keys ab *(siehe Abbildung 1)*.

Testszenarien

Folgende Tools und Methoden sind ausgewählt, um die Generierung von Sub-

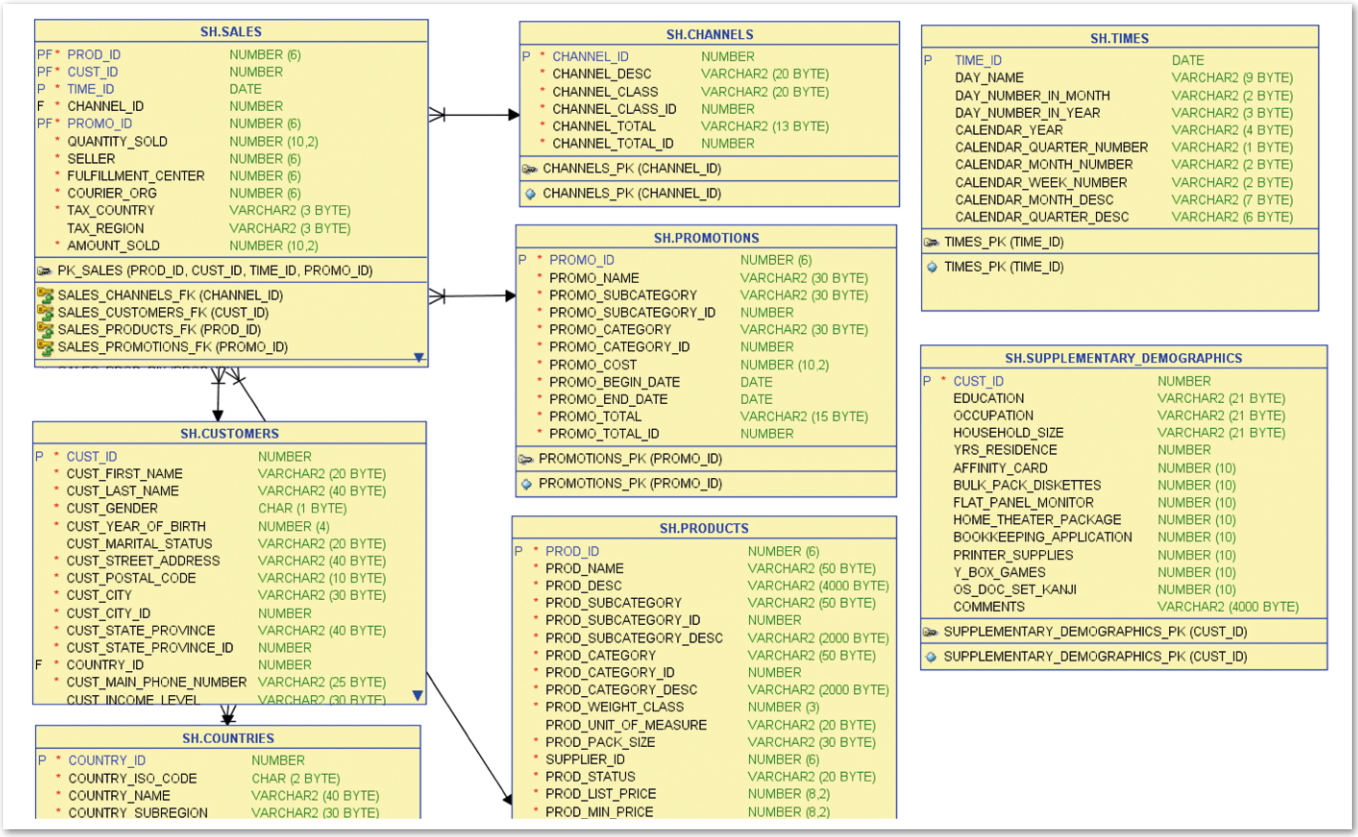


Abbildung 1: Das SH-Schema

sets zu testen: SQL, Oracle Datapump und toolbasiertes Subsetting. Um die Tools und Methoden besser vergleichen zu können, betrachten wir sie anhand der folgenden drei Aufgabenstellungen:

- Alle Daten für Kunden aus Japan
- Alle Kunden mit Umsatz größer als 500.000 Euro und die zugehörigen Details
- Zehn Prozent der Gesamt-Datenmenge

Subsetting per SQL

Die ersten beiden Aufgabenstellungen lassen sich mit SQL-Standardabfragen beantworten. Bei der dritten muss definiert sein, wie die zehn Prozent der Daten in SQL übersetzt werden. Als Näherung könnte man zehn Prozent der Fakten und die zugehörigen Dimensionswerte wählen oder zehn Prozent der Dimensionswerte und die zugehörigen Fakten. Über genauere Analysen können Regeln aufgestellt werden, die eine genaue Extraktion möglich machen. Um regelmäßig zehn Prozent der Daten abzuziehen, ist eine Näherung wegen des geringeren Aufwands aber besser geeignet.

Die Beziehungen zwischen den Tabellen über Foreign Keys können genutzt werden, um Statements zur Extraktion zu generieren. Das Schreiben dieses Codes ist jedoch aufwändig und deckt die zusätzlichen Verbindungen nicht ab. Bei einem einfachen Datenmodell wie dem SH-Schema ist das direkte Schreiben der SQL-Statements weniger aufwändig.

Um die Daten performant in die Zieldatenbanken zu importieren, sollte der Extrakt mit dem SQL*Loader erfolgen. Mit dem Unloader Utility von Tom Kyte wird zusätzlich das passende Controlfile generiert und somit der gesamte Prozess automatisiert. Für einmalige Aktionen oder erste Tests kann ein Export als SQL*Loader-Datei auch mit graphischen Tools wie dem SQL Developer vorgenommen werden *(siehe Tabelle 1)*.

Subsetting mit Oracle Datapump

Oracle Datapump ist nicht das Erste, das einem zum Thema „Subsetting" in den Sinn kommt. Dabei bietet sich Datapump gerade für große Datenmengen an. Bereits Oracle Export enthielt einen Query-Parameter, der es erlaubte, Teilmengen einer Tabelle zu exportieren.

Der Query-Parameter für Datapump in Version 11.2 und 12.1 bietet die Möglichkeit, komplexe Abfragen zu integrieren. Es kann nicht nur eine „Where"-Bedingung für die zu exportierende Tabelle angegeben werden, sondern auch Lookups auf andere Tabellen. Die Verwendung eines Parameterfiles ist zu empfehlen. Es gibt einfache *(siehe Listing 1)* und komplexe Queries *(siehe Listing 2)*.

Wichtig ist der Alias „ku$" für die Tabelle, auf die sich die Query bezieht, sowie die Angabe des vollqualifizierten Tabellen-Namens, wenn man den Export nicht als Tabellen-Owner durchführt. Werden mehrere Tabellen beziehungsweise ein ganzes Schema extrahiert, so kann für jede Tabelle eine eigene Query angegeben werden, wobei jede Query aus maximal 4.000 Zeichen inklusive Anführungszeichen bestehen darf.

Aufgabe 1 der oben genannten Testszenarien ist mit Datapump leicht zu beantworten, indem die passenden Queries in das Parameterfile eingetragen werden.

```
QUERY=PROMOTIONS:"WHERE promo_
category = 'TV'"
```

Listing 1

```
QUERY=SALES:"WHERE EXISTS ( se-
lect cust_id from SH.CUSTOMERS c
where c.country_id = 52782  AND
c.cust_id=ku$.cust_id)
AND ku$.prod_id in (SELECT prod_
id from SH.PRODUCTS p where
p.prod_category='Photo')
AND exists (select promo_id from
SH.PROMOTIONS pr where
pr.promo_category = 'TV' and
pr.promo_id = ku$.promo_id)"
```

Listing 2

+	-
Performance beim Import über sqlldr	Exportgeschwindigkeit
Abhängigkeiten zwischen Tabellen abbildbar	Manuelle Erstellung und Pflege
Beliebig komplexe Queries möglich	Nicht geeignet für Binä rdaten
Plattformübergreifend Dateigröße	*Dateigröße*
Versionsübergreifend	
Scriptfähig	

Tabelle 1: Subsetting per SQL

Um die Aufgabe 2 zu lösen, muss man jedoch eine zusätzliche View anlegen, da innerhalb der Query keine Selbstreferenz angegeben werden kann. Aufgabenstellung 3 kann analog zum Vorgehen bei SQL beantwortet werden.

Das Verfahren ist gut automatisierbar und für größere Datenmengen geeignet *(siehe Tabelle 2)*.

Toolbasiertes Subsetting

Es gibt einige Tools, die das Erzeugen von Testdaten vereinfachen wollen. Nachfolgend werden zwei unterschiedliche Tools vorgestellt, zum einen das Open-Source-Tool Jailer, das auf Subsetting spezialisiert ist und für zahlreiche Datenbanken verwendet werden kann, und zum anderen die Subsetting-Funktionalität von Oracle Grid Control.

Jailer *(siehe „http://jailer.sourceforge.net")* ist ein Open-Source-Tool für die Generierung von Subsets, das aktiv gepflegt wird. Aktuell ist die Version 4.3.3 von Januar 2015. Jailer verwendet eine mehrstufige Vorgehensweise:

- Generierung eines Datenmodells, ausgehend vom Data Dictionary
- Hinzufügen von Beziehungen, die nicht durch Foreign Keys beschrieben sind
- Erstellung eines Extraktionsmodells
- Export des Subset

Unter Windows muss vor der Verwendung von Jailer die zentrale Batch-Datei „jailerGUI.bat" angepasst werden *(siehe Listing 3)*. Möchte man Jailer per Skript ansprechen, müssen die entsprechenden Anpassungen auch in der Datei „jailer.bat" vorgenommen werden.

Nach dem Start von Jailer wird zuerst ein Datenmodell angelegt. Man kann dies manuell vornehmen oder ein Schema aus der Datenbank per Reverse Engineering auslesen. Nach dem Auslesen des Datenmodells aus der Datenbank lassen sich fehlende Referenzen manuell nachpflegen *(siehe Abbildung 2)*.

Jailer liest keine Views, sondern nur Tabellen ein und erwartet, dass alle Tabellen einen Primary oder Unique Key aufweisen. Ist kein Primary Key auf der Datenbank implementiert, so kann man diesen im Datenmodell-Editor logisch hinzufügen. Nur mit Primary oder Unique Key in allen Tabellen kann ein Datenexport erfolgen. Im nächsten Schritt wird ein Extrak-

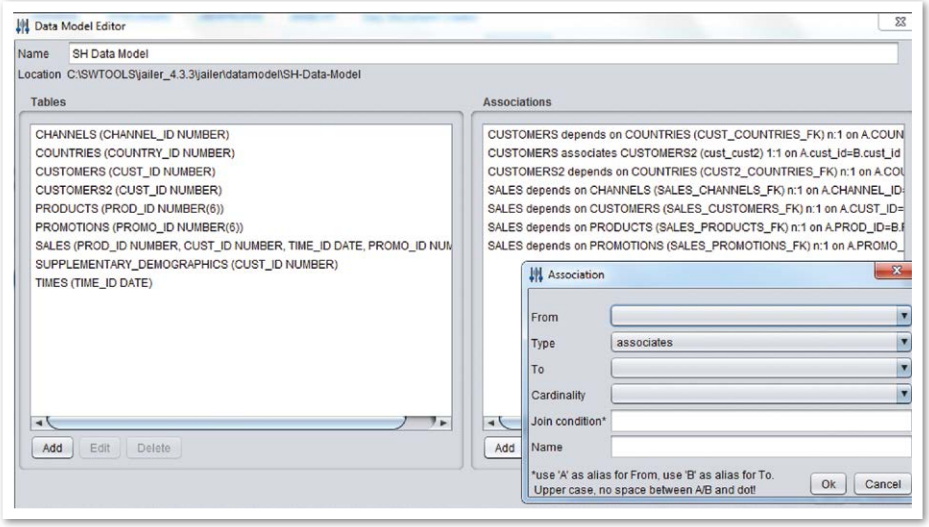


Abbildung 2: Relationen hinzufügen

```
set LIB=lib
set JAVA_HOME=C:\Oracle\product\
db\12.1.0\jdk\jre

rem JDBC-driver
rem set CP=%CP%;<jdbc-driver>.jar

rem configuration files in the con-
fig directory
set CP=%CP%C:\Oracle\product\
db\12.1.0\jdbc\lib\ojdbc6.
jar;config
```

Listing 3: Jailer-Konfiguration

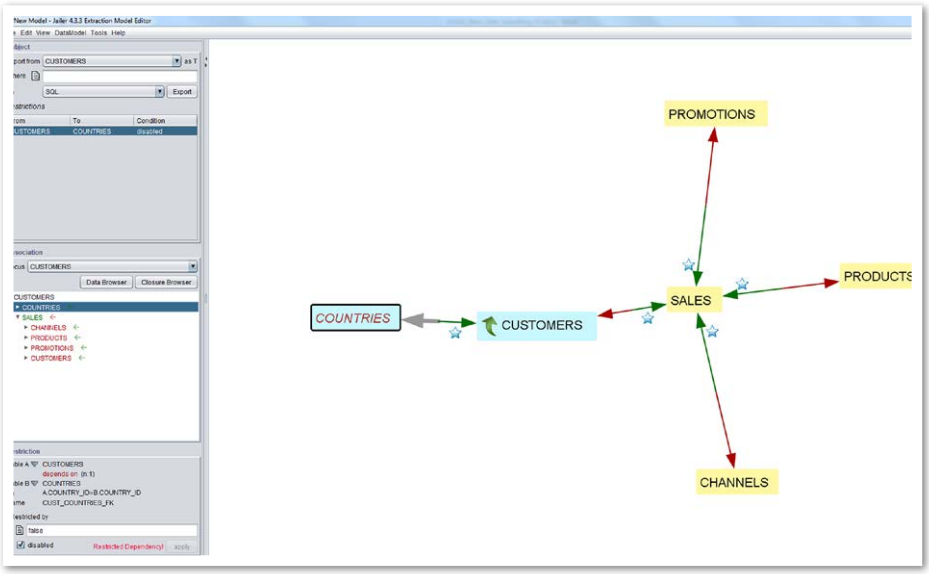


Abbildung 3: Das Extraktionsmodell

tionsmodell erstellt. Hier wird festgelegt, welche die führende Tabelle für den Export ist und welche Einschränkungen vorgenommen werden *(siehe Abbildung 3)*.

Im Extraktionsmodell muss zudem festgelegt sein, wie der Umgang mit den Relationen erfolgt. Es gibt die beiden Modi „restricted" und „unrestricted". Im „restricted"-Modus geht man von der führenden Tabelle (etwa Sales) aus und exportiert die Sales-Daten und die zugehörigen Daten aus den Tabellen „Promotions", „Products", „Channels", „Customers" und „Countries".

+	-
Performance	Import nur in die identische Datenbank-Version oder in höhere Versionen
Abhängigkeiten zwischen Tabellen abbildbar	Queries auf 4.000 Zeichen beschränkt
Plattformübergreifend	Keine Möglichkeit von Gruppen-Operationen in den Queries
	Manuelle Erstellung und Pflege

Tabelle 2: Subsetting mit Oracle Datapump

Im „unrestricted"-Modus werden die Relationen in Rückrichtung aufgelöst und beispielsweise zu dem ausgewählten Kunden alle Sales-Datensätze exportiert. Dies kann dazu führen, dass deutlich mehr Daten exportiert werden, als geplant.

Für unsere drei Aufgabenstellungen ist der „restricted"-Modus sinnvoll. Die Aufgabenstellung 1 ist einfach zu beantworten, indem man die Tabelle „Countries" als führende Tabelle wählt. Um die Aufgabenstellung 2 zu lösen, muss eine View angelegt werden, die die gewünschten Kunden enthält. Diese kann man bei der Extraktion manuell referenzieren *(siehe Abbildung 4)*.

Aufgabenstellung 3 kann nicht direkt beantwortet werden. Es gibt keine Mög-

lichkeit, in der GUI auf x Prozent der Quelldaten einzuschränken. Hier muss noch einmal der Umweg über eine View oder Tabelle genommen werden. Jailer bietet für den Export vier verschiedene Formate an:

- Insert
- XML
- DbUnit Flat File
- Liquibase.

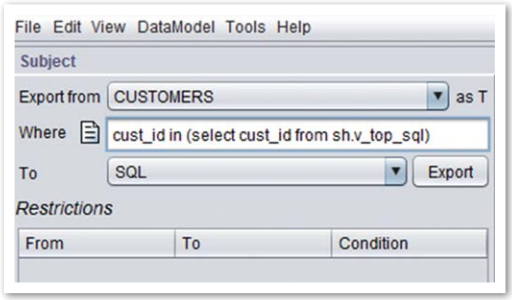


Abbildung 4: View als Referenz

```
Insert into SALES(PROD_ID,
CUST_ID, TIME_ID, CHANNEL_ID,
PROMO_ID, QUANTITY_SOLD, SELLER,
FULFILLMENT_CENTER, COURIER_ORG,
TAX_COUNTRY, TAX_REGION, AMOUNT_
SOLD)
Select 26, 376675, to_timestamp
('2011-12-12 00.00.00.0', 'YYYY-
MM-DD HH24.MI.SS.FF1'), 9, 132,
95, 10230, 13566, 1506, 'WL',
'KH', 71 From DUAL Onion all
Select 52, 1287693, to_times-
tamp('2011-12-16 00.00.00.0',
1YYYY-MM-DD HH24.MI.SS.FF1'),
3, 394, 48, 10230, 13346, 1212,
'IF', 'CG', 32 From DUAL Union
all
Select 67, 743120, to_times-
tamp('2011-10-09 00.00.00.0',
'YYYY-MM-DD HH24.MI.SS.FF1'),
3, 36, 42, 10230, 13469, 1244,
'KM', 'CC', 21 From DUAL Union
all
Select 17, 1303887, to_times-
tamp('2011-12-14 00.00.00.0',
'YYYY-MM-DD HH24.MI.SS.FF1'),
5, 342, 61, 10230, 14408, 1044,
'RX', 'YF', 12 From DUAL Union
all
Select 31, 208797, to_times-
tamp('2011-12-07 00.00.00.0',
'YYYY-MM-DD HH24.MI.SS.FF1'),
5, 197, 81, 10230, 14848, 1495,
'QQ', 'LN', 17 From DUAL Union
all
```

Listing 4: Insert Statements von Jailer

Die generierten Insert Statements sehen etwas ungewöhnlich aus, funktionieren aber *(siehe Listing 4)*.

Der Export als Insert Statement bietet Vorteile beim plattform- und versionsübergreifenden Transfer, eignet sich aus Performance-Gründen jedoch nicht für große Datenmengen *(siehe Tabelle 3)*. Jailer verfügt auch über ein Kommandozeilen-Interface. Nach der Erstellung eines Extraktionsmodells in der GUI können die Extrakte per Skript erzeugt und Jailer so automatisiert werden. Über sogenannte „Filter" kann Jailer auch einfache Maskierungen der Quelldaten zur Anonymisierung durchführen.

Oracle Grid Control

Oracle Grid Control bietet einen mächtigen Funktionsumfang für die Verwaltung größerer Datenbank-Umgebungen. Seit dem Release 12*c* ist auch das Erstellen von Subsets und maskierten Daten möglich. Hierfür muss allerdings das „Oracle Data Masking and Subsetting Pack" lizenziert sein. Das Vorgehen ist weitgehend analog zu Jailer:

- Generierung eines Datenmodells, ausgehend vom Data Dictionary
- Hinzufügen von Beziehungen, die nicht durch Foreign Keys beschrieben sind
- Definition von Einschränkungen und Regeln
- Export des Subset

Das Datenmodell ist sowohl die Basis für Subsetting, als auch für Data Masking *(siehe Abbildung 5)*.

Im Applikationsmodell können weitere Verknüpfungen hinterlegt sein. Im Gegensatz zu Jailer müssen die Tabellen keinen Primärschlüssel aufweisen.

Um das Subset zu beschreiben, stehen unterschiedliche Vorgaben zur Verfügung *(siehe Abbildung 6)*.

Es kann auch eine Mengenangabe in Prozent erfolgen. Die Aufgabenstellung 1 kann erledigt werden, indem man die Tabelle „Countries" als führende Tabelle wählt. Um die Aufgabenstellung 2 zu bearbeiten, muss eine View anlegt sein, die die gewünschten Kunden enthält. Diese View kann man bei der Extraktion manuell referenzieren. Aufgabe 3 kann in Grid Con-

+	-
Abhängigkeiten zwischen Tabellen abbildbar	Primary oder Unique Key in allen Tabellen notwendig
Datenbankübergreifend	Nicht geeignet für Binärdaten
Plattformübergreifend	Dateigröße
Versionsübergreifend	Eignung für große Datenmengen
Kosten	Gefahr von Schleifen
Scriptfähig	Anlage von Metadaten-Tabellen in der Datenbank
Export als Input für DBUnit verwendbar	

Tabelle 3: Export als Insert Statement

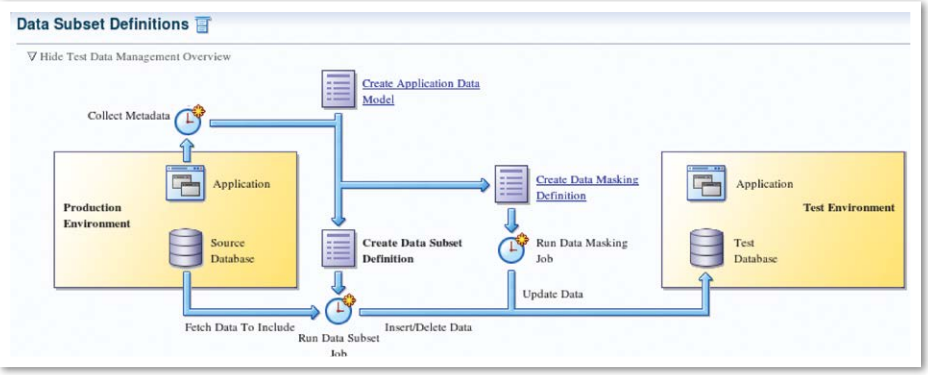


Abbildung 5: Applikationsmodell

+	-
Abhängigkeiten zwischen Tabellen abbildbar	Hardware-Bedarf
Datenmaskierung möglich	Lizenzkosten
Plattformübergreifend	Komplexität
Für komplexe Umgebungen geeignet	
Automatisierbar	

Tabelle 4: Export mit Oracle Grid Control

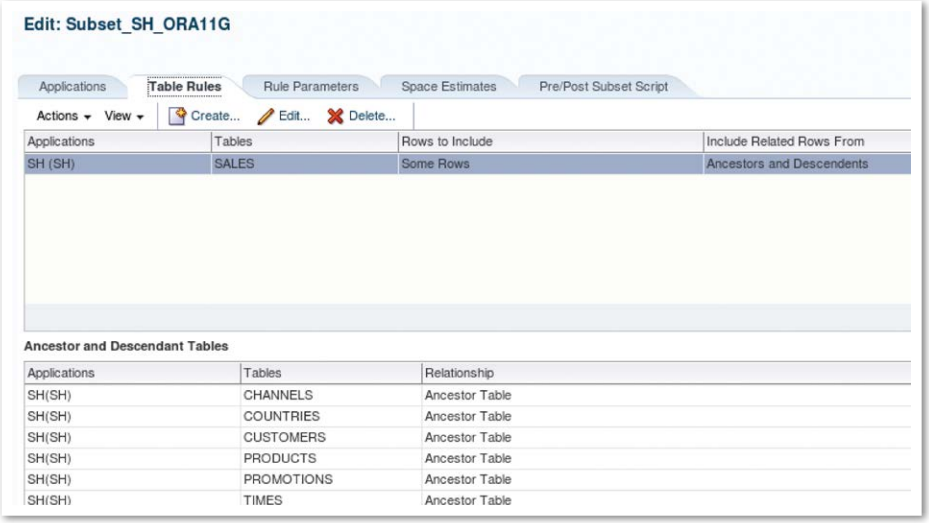


Abbildung 6: Beschreibung der Subsets

trol ohne Schwierigkeiten gelöst werden, indem zehn Prozent der Gesamtmenge als Bedingung angeben wird. Grid Control gibt vor dem Export aus, wie viele Zeilen je Tabelle zu erwarten sind.

Grid Control exportiert die Daten per Datapump. Daher lassen sich die Daten performant in andere Oracle-Datenbanken, mit demselben Release oder höher, einspielen. Über Grid Control Jobs kann das Subsetting gut automatisiert und ohne manuelle Eingriffe durchgeführt werden. Mit dem Patchset 3 von Grid Control ist es möglich, Data Masking und Subsetting zu kombinieren. So können auch personenbezogene oder sensible Daten vom Produktivsystem kopiert werden *(siehe Tabelle 4)*.

Fazit

Data Subsetting ist ein komplexes Thema und erfordert auch mit Tool-Unterstützung ein detailliertes Wissen über das Datenmodell. Jailer und Grid Control ermöglichen es, schnell erste Subsets zu erzeugen; SQL und Datapump erfordern einen höheren Einarbeitungsaufwand, bieten aber auch mehr Flexibilität und ermöglichen komplexe Bedingungen. Aufgrund der Lizenzkosten bietet sich Grid Control nur an, wenn auch Data Masking und Automatisierung in komplexeren Umgebungen gefordert sind.

Oliver Gehlert
oliver.gehlert@ventum.de

Oracle Hidden Secrets:

Zeichensatz-Konvertierung beim Datenbank-Upgrade leicht gemacht

Ralf Durben, ORACLE Deutschland B.V. & Co. KG

Im Rahmen eines Datenbank-Upgrades auf die aktuelle Version 12*c* ergibt sich für die historisch gewachsenen Datenbanken oft die Frage, inwieweit diese in ein Konsolidierungskonzept auf Basis der Multitenant-Architektur passen. Schließlich müssen alle Pluggable-Datenbanken innerhalb einer Container-Datenbank mit dem gleichen Zeichensatz betrieben werden. Das kleine Tool „Database Migration Assistant for Unicode" ist in diesen Situationen die Lösung.

Schon seit geraumer Zeit empfiehlt Oracle für die Datenbank den Einsatz von Unicode-Zeichensätzen. Ältere Datenbanken jedoch verwenden oft noch die länder- oder regionsspezifischen Zeichensätze wie zum Beispiel „WE8ISO8859P1". Typischerweise wurde für diese älteren Datenbanken bei früheren Upgrades keine Konvertierung des Zeichensatzes durchgeführt, solange es keinen Grund seitens der Anwendung dazu gab.

Wer eine solche Datenbank per Upgrade auf die Version 12*c* aktualisiert, für den ist das generell auch weiterhin nicht notwendig. Man kann die Datenbank weiterhin mit dem alten Zeichensatz betreiben, solange die Datenbank isoliert läuft. Wer jedoch zusätzlich eine Konsolidierung seiner Datenbanken in einer Multitenant-Architektur in Betracht zieht, für den ist ein einheitlicher Zeichensatz zwingend, denn alle Pluggable-Datenbanken in einer Container-Datenbank müssen den gleichen Zeichensatz verwenden.

Wer darüber hinaus die Vorteile der Portabilität von Pluggable-Datenbanken nutzen möchte, sollte für seine Container-Datenbanken einen einheitlichen Zeichensatz verwenden. Da bietet sich die Verwendung eines Unicode-Zeichensatzes an, der auch von Oracle empfohlen wird.

Der Wechsel des Datenbank-Zeichensatzes erfordert unter Umständen eine Modifizierung der Daten und ist nicht einfach nur eine Änderung eines Parameters. Dabei ist zu beachten, dass die Daten nach der Konvertierung unter Umständen mehr Speicherplatz benötigen und daher eventuell die Datentyp-Länge der Tabellenspalte überschreiten. Oracle bietet dazu das kostenlose grafische Tool „Database Migration Assistant for Unicode" (DMU) an. Es prüft zunächst, ob die Konvertierung problemlos durchführbar ist und führt diese anschließend auch durch. Dabei ist zu beachten, dass das Tool keine Versions-Veränderung an der Datenbank vornimmt, auch wenn der Name durch das Wort „Migration" diese Assoziation erweckt. Das Tool analysiert eine Oracle-Datenbank und zeigt Folgendes an *(siehe Abbildung 1)*:

- Welcher Zeichensatz von der Datenbank verwendet wird
- Welche Datenbank-Objekte analysiert und angepasst werden müssen
- Ob eine Konvertierung ohne Probleme durchgeführt werden kann

Die jeweils aktuelle Version von DMU lässt sich von den Oracle-Webseiten „OTN“ und „My Oracle Support“ herunterladen. Wer den Zeichensatz einer Datenbank, die älter als 11.2.0.3 ist, auf Unicode umstellen möchte, muss eventuell noch einen Patch einspielen. Details dazu stehen in der ausführlichen MOS-Note „The Database Migration Assistant for Unicode (DMU) Tool (Doc ID 1272374.1)“ und in der Dokumentation von DMU.

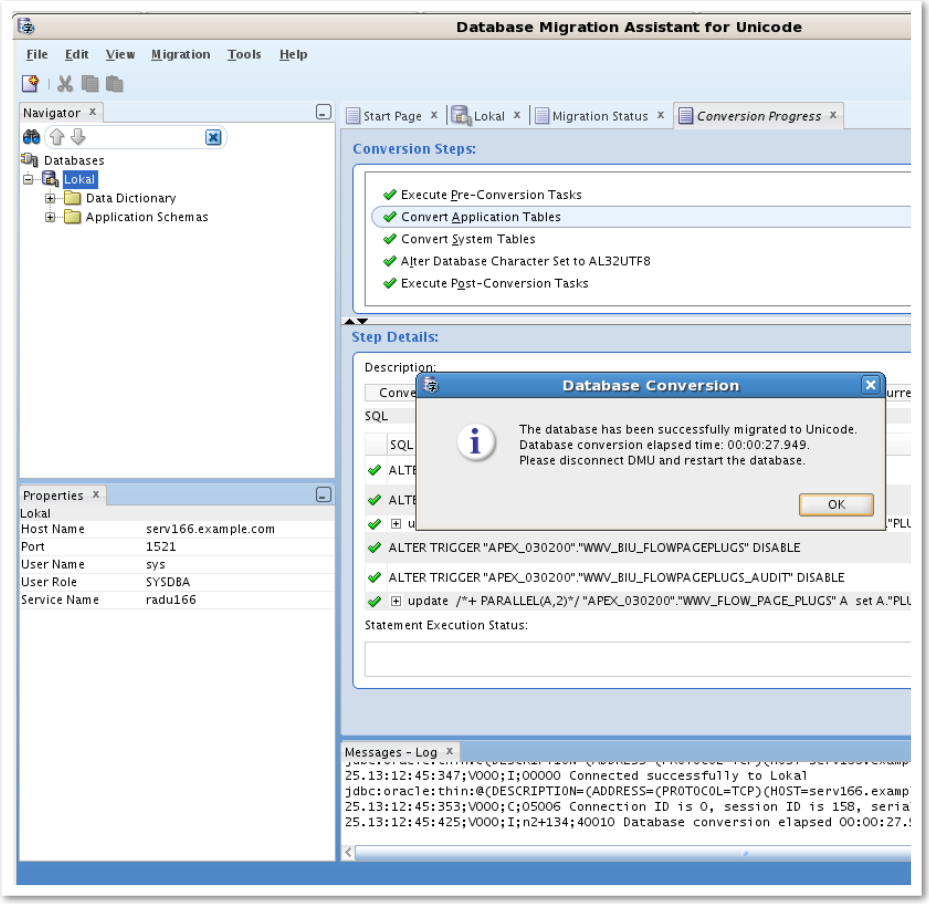


Abbildung 1: Zeichensatz-Konvertierung mit DMU

Fazit

Database Migration Assistant for Unicode (DMU) kann eine Oracle-Datenbank leicht in eine Unicode-Datenbank konvertieren. Die Umstellung des Zeichensatzes ist ein Spezialfall beim Upgrade von Datenbanken. In der aktuellen Kampagne „12cjetzt“ zeigen sieben Webinare, warum das Upgrade auf Database 12*c* gerade jetzt interessant ist, wie das Upgrade erfolgt und dass der Weg zu 12*c* in den meisten Fällen kein Hexenwerk ist *(siehe „http://tinyurl.com/12cjetzt“).*

Ralf Durben
ralf.durben@oracle.com

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

Matthias Schmidt
Gebhard Przyrembel
Martin Moje
Ankita Khandelwal
Frank Bitzer
Stefan Koehler
Jörg Maaßen
Alexander Pedde
Eike Zink
Carsten Sensler
John Kelly
Michael Kasten
Rainer Lehrbach
Osama MustafaMountaga Sall
Alexander Cora
Stephan Gneist
Thilo Schwinge
Bernd Rosenau
Dirk Blask
Gerhard Knarr
Andreas Rein

Firmenmitglieder DOAG

Andreas Weitzel, SCHUFA Holding AG
Karsten Besserdich, Besserdich Sustainable IT Solutions GmbH
Kathrin Griesel, Landesamt für Digitalisierung, Breitband und Vermessung - IT14
Markus Bremer, IPartake Consulting Gmb-HMartin Waiblinger, TransnetBW GmbH
Sandra Happel F; Aareon Deutschland GmbH

Neumitglieder SOUG

Michael Krebs, esentri AG

08.09.2015
Regionaltreffen Dresden/Sachsen
Helmut Marten
regio-sachsen@doag.org

09.09.2015
Regionaltreffen Berlin/Brandenburg
Michel Keemers
regio-bb@doag.org

10.09.2015
Regionaltreffen Trier/Saar/Luxemburg
Bernd Tuba, Holger Fuchs
regio-trier@doag.org

10.09.2015
Regionaltreffen Bremen – Nordlichtertreffen
Ralf Kölling
regio-bremen@doag.org

10.9.2015
SOUG SIG
sekretariat@soug.ch

11.09.2015
Webinar: Zentrale Steuerung von Installationen und Updates - Linux Spacewalk
Christian Trieb
ctr@doag.org

14.09.2015
Regionaltreffen Osnabrück/Bielefeld/Münster
Andreas Kother, Klaus Günther
regio-osnabrueck@doag.org

14.09.2015
Regionaltreffen Halle/Leipzig
Matthias Reimann
regio-halle@doag.org

15.-16.09.2015
Berliner Expertenseminar mit Oliver Lemm ProfessionelleEntwicklung mit APEX 5.0
Cornel Albert
expertenseminare@doag.org

17.09.2015
Regionaltreffen Nürnberg/Franken – Oracle Middleware für DBAs
André Sept, Martin Klier
regio-franken@doag.org

17.-18.09.2015
DOAG Big Data Days: SQL meets Big Data (mit Hands-On) | Hannover
Peter Welker, Jens Bleiholder, Christian Schwitalla
office@doag.org

21.09.2015
DOAG Cloning Day | Leipzig
Johannes Ahrends, Christian Trieb
sig-database@doag.org

22.09.2015
Regionaltreffen Hamburg/Nord
Jan-Peter Timmermann
regio-nord@doag.org

22.09.2015
DOAG Security Day | Leipzig
Bruno Cirone
sig-security@doag.org

23.09.2015
DOAG Middleware Day | Stuttgart
Jan-Peter Timmermann
sig-middleware@doag.org

24.09.2015
DOAG OpenStack Day 2015 | Stuttgart
Jan-Peter Timmermann, Torsten Winterberg, Heiko Stein
sig-middleware@doag.org

24.09.2015
Regionaltreffen München/Südbayern
Franz Hüll, Andreas Ströbel
regio-muenchen@doag.org

29.09.2015
Regionaltreffen NRW (Apex Community)
Stefan Kinnen, Andreas Stephan
regio-nrw@doag.org

29.-30.09.2015
Berliner Expertenseminar mit Gerd Aiglstorfer zum Thema "OBIEE Repository und Reports Master Class"
Cornel Albert
expertenseminare@doag.org

06.10.2015
Regionaltreffen Rhein-Neckar Thema: Hochverfügbarkeit
Frank Stöcker
regio-rhein-neckar@doag.org

Weitere Termine und Informationen unter www.doag.org/termine/calendar.php sowie unter www.soug.ch

Impressum

Herausgeber:
DOAG Deutsche ORACLE-Anwendergruppe e.V.
Tempelhofer Weg 64, 12347 Berlin
Tel.: 0700 11 36 24 38
www.doag.org

SOUG Swiss Oracle User Group
Im Gundelinger Feld/Bau 5
Dornacherstrasse 192
CH-4053 Basel
Tel.: +41 (0)61 367 93 30
www.soug.ch

Verlag:
DOAG Dienstleistungen GmbH
Fried Saacke, Geschäftsführer
info@doag-dienstleistungen.de

Chefredakteur (ViSdP):
Wolfgang Taschner, redaktion@doag.org

Redaktion:
Fried Saacke, Julia Bartzik, Marina Fischer, Mylène Diacquenod, Marius Fiedler, Dr. Dietmar Neugebauer, Gaetano Bisaz

Titel, Gestaltung und Satz:
Alexander Kermas, DOAG Dienstleistungen GmbH

Titel: © 1xpert / fotolia.com
Foto S. 20: © peshkova / fotolia.com
Foto S. 38: © Nmedia / fotolia.com
Foto S. 64: © Oracle / oracle.com

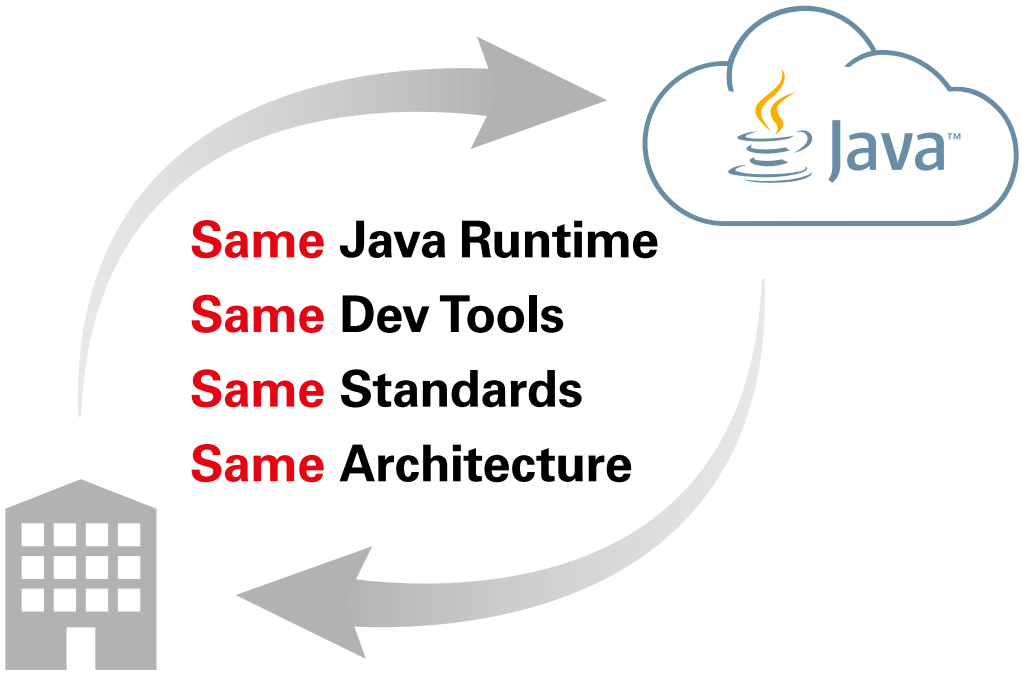
Anzeigen:
Simone Fischer, anzeigen@doag.org
DOAG Dienstleistungen GmbH
Mediadaten und Preise finden Sie unter: www.doag.org/go/mediadaten

Druck:
Druckerei Rindt GmbH & Co. KG
www.rindt-druck.de

Inserentenverzeichnis

Push a Button

Move Your Java Apps to the Oracle Cloud



... or Back to Your Data Center

ORACLE®

cloud.oracle.com/java