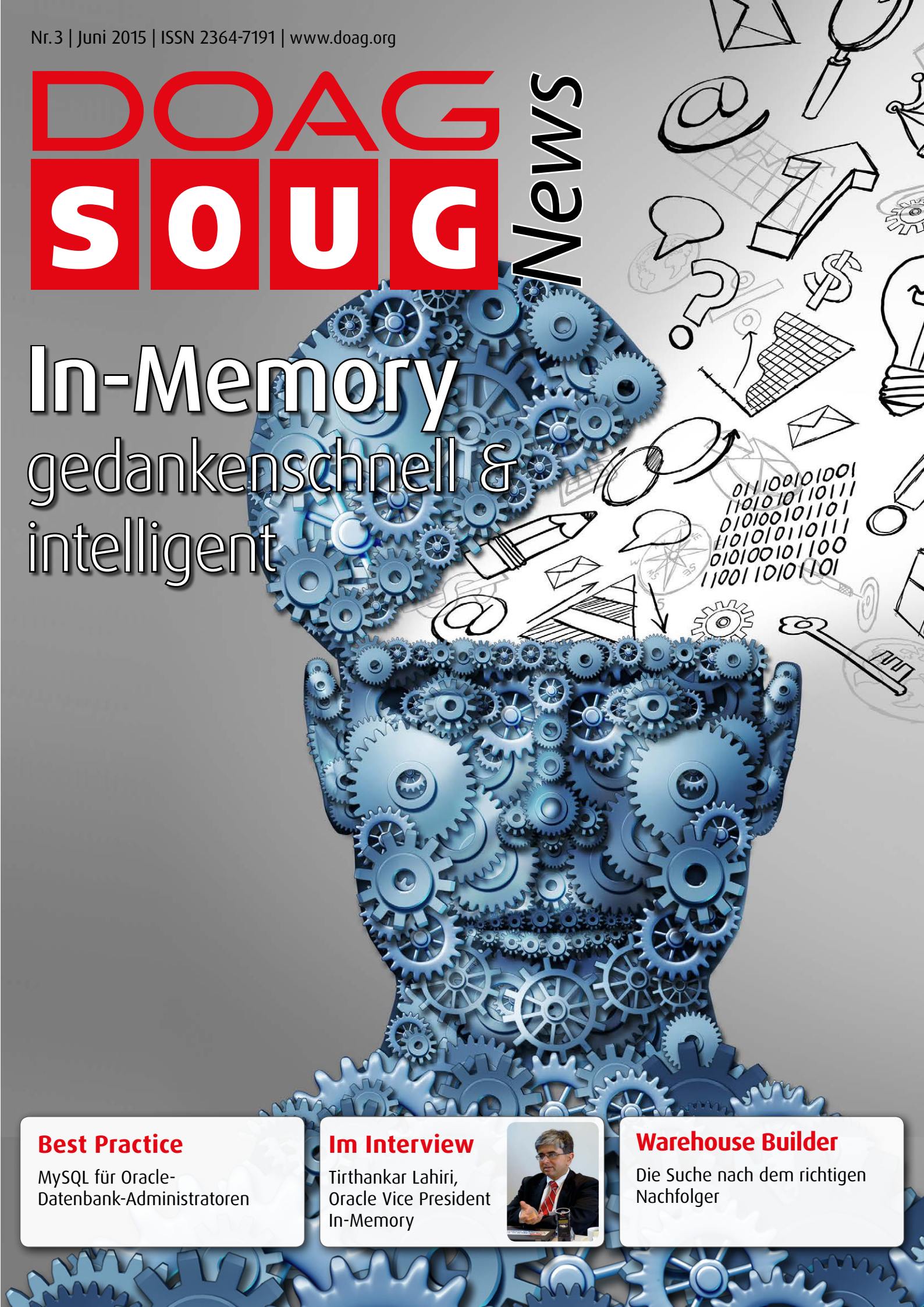
Nr. 3 | Juni 2015 | ISSN 2364-7191 | www.doag.org

DOAG SOUG News

In-Memory
gedankenschnell & intelligent



Best Practice
MySQL für Oracle-Datenbank-Administratoren

Im Interview
Tirthankar Lahiri, Oracle Vice President In-Memory

Warehouse Builder
Die Suche nach dem richtigen Nachfolger


bis 30.09.
Early Bird
Jetzt anmelden!
2015
DOAG
Konferenz + Ausstellung
17. - 20. November | Nürnberg
Experience Passion
Weil jedes IT-Projekt einer Rafting-Fahrt ähnelt.
Eventpartner:
AOUG
AUSTRIAN ORACLE USER GROUP
SOUG
Swiss Oracle User Group
2015.doag.org

Christian Trieb
Mitglied des Vorstands und Leiter der DOAG Datenbank Community

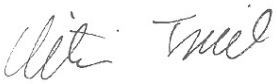
Liebe Mitglieder, liebe Leserinnen und Leser,

„In-Memory" nimmt heutzutage in der Datenverarbeitung und damit auch in der Oracle-Welt einen immer höheren Stellwert ein. Seit Mitte des Jahres 2014 ist die In-Memory-Option für die Oracle-Datenbank verfügbar. Ist es nur ein kurzfristiger Hype oder hat dies langfristige Auswirkungen, wie funktioniert diese Technologie überhaupt, wann ist es sinnvoll, In-Memory einzusetzen und ergibt dies bei einer Exadata-Datenbank-Maschine überhaupt Sinn? Diese und weitere Fragestellungen werden in dieser Ausgabe diskutiert und beantwortet.

Für mich persönlich ist die Frage nach dem Einsatz der In-Memory-Option im Exadata-Umfeld eine sehr spannende Fragestellung, denn eine Exadata-Datenbank-Maschine verfügt gerade in der neuesten Version X5 über sehr viel Flash-Speicher. Kann man die Performance mit der In-Memory-Technologie nochmal beschleunigen oder muss man hier andere Ansätze wählen? Ob Flash-Speicher und In-Memory ein Widerspruch oder eine Ergänzung darstellen, ist eine der praktischen Aufgabenstellungen, die es mit entsprechenden Tests zu bewältigen gilt. Diese Ausgabe der DOAG/SOUG-News gibt dafür entsprechende Hilfestellungen, führt in die Thematik ein, beschreibt die Funktionsweise und beantwortet damit die gestellten Fragen.

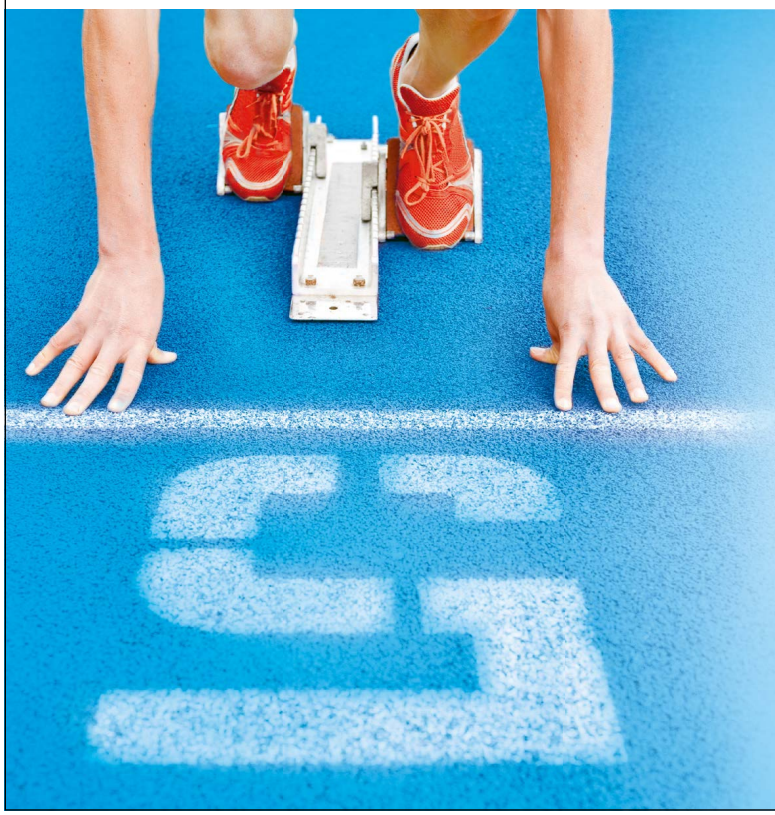
Ganz besonders freut es mich, das der Interviewpartner in diesem Heft, Tirthankar Lahiri, Oracle Vice President Data Technologies and Times Ten, auf der DOAG 2015 Konferenz + Ausstellung im November in Nürnberg teilnimmt, um über die neuesten Entwicklungen von Oracle in diesem Bereich zu informieren.

Ich wünsche Ihnen viel Spaß beim Lesen dieser Ausgabe und eine gute Sommer- und Urlaubszeit.
Ihr



ORACLE Gold Partner
Specialized
Oracle Database

MUNiQSOFT
Datenbanken mit iQ

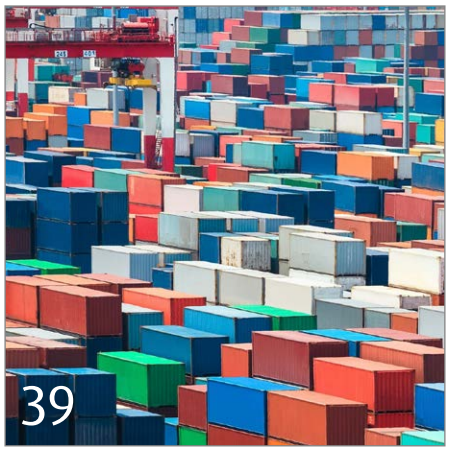


APEX 5 steht in den Startlöchern, wir auch ...

5 Gründe, die für uns sprechen:

1. 100 durchgeführte APEX-Schulungen
2. 10 Jahre APEX-Erfahrung
3. 300 geschulte Teilnehmer
4. 5 verschiedene APEX-Kurse im Programm
5. Mehr als 5 Oracle APEX-Consultants in unserem Haus

Besuchen Sie uns unter
www.muniqsoft.de/apex



Der Prototyp eines Docker-Containers zeigt was machbar ist und liefert praktische Erfahrungen

Erfahrene Oracle-DBAs kennen ihre Datenbanken aus dem Effeff, sind bei MySQL-Datenbanken aber etwas ratlos

Die Auswahl des Character Sets entscheidet beim Anlegen über die zur Verfügung stehenden Zeichen der Datenbank

Einleitung

In-Memory

Datenbank

Aktuell

Data Warehouse

Entwicklung

DOAG/SOUG intern

DOAG/SOUG Timeline

13. März 2015

Die letzte Vorstandssitzung der DOAG in dieser Wahlperiode bereitet die Delegiertenversammlung am 8. und 9. Mai in Berlin vor, auf der der neue Vorstand gewählt wird. Zweiter wichtiger Punkt ist die Programmgestaltung der DOAG 2015 Konferenz in Nürnberg. Dazu wird ein Team aufgestellt, dessen Mitglieder jeweils die Vortragsauswahl eines Streams betreuen.

18. März 2015

Beim Redaktionsmeeting im Berliner DOAG-Office geht es um die Feinabstimmung in der Herstellung der Zeitschriften sowie um Synergien zwischen DOAG Online und den Printmedien. Ein weiteres Thema ist die Übergabe der Presse- und Öffentlichkeitsarbeit von Wolfgang Taschner an Mylène Diacquenod. Wolfgang Taschner, der mehr als 15 Jahre die Presselandschaft für die DOAG betreut hat, ist weiterhin als Chefredakteur für alle Zeitschriften der DOAG zuständig.

19. März 2015

Andreas Ellerhoff, regionaler DOAG-Repräsentant in Hannover, lädt zum DOAG OEM Day auf der CeBIT ein. Im Convention Center auf dem Messegelände Hannover bieten erfahrene Experten eine optimale Kombination aus Information und Austausch zum Oracle Enterprise Manager sowie zu neuen Markttrends und Technologien auf der CeBIT 2015. In der verlängerten Mittagspause erhalten die Teilnehmer freien Zugang auf das CeBIT-Messegelände, um sich dort zu informieren. Bereits am Vorabend trifft sich die Regionalgruppe Hannover zum traditionellen Stammtisch auf der CeBIT.

CeBIT 2015 – ein Top-Event der digitalen Welt

19. März 2015

In Baden findet die Generalversammlung der Swiss Oracle User Group (SOUG) statt. Herauszuheben sind die Ehrungen von Ansgar Wollnik und Hans Jörg Bütler, die aus dem Vorstand austreten, sowie die „Speaker of the year"-Auszeichnung an Harro M. Wiersma von der Ernst & Young AG. Die im Anschluss an die Generalversammlung stattfindende SIG-Veranstaltung ist mit drei Streams und insgesamt zwölf Vorträgen schon fast eine Konferenz „light".

24. März 2015

Rund tausend Java-Begeisterte besiedeln das JavaLand im Phantasialand in Brühl. Neben mehr als hundert spannenden Java-Vorträgen und Community-Aktivitäten erleben sie die Attraktionen des Erlebnisparks. Die einmalige Mischung aus Spaß, Community und Technologie ist damit auch in der zweiten Auflage der Java-Konferenz ein voller Erfolg. Fried Saacke, DOAG-Geschäftsführer und Vorstandsvorsitzender des Interessenverbunds der Java User Groups e.V. (iJUG), eröffnet die Veranstaltung mit einem kräftigen „Jatumba"-Ruf und alle Teilnehmer erwidern begeistert die Grußformel der JavaLand-Besiedler.

Fried Saacke beim „Jatumba"

24. März 2015

Im Rahmen der JavaLand 2015 findet die erste JavaLand4Kids statt. Die Mädchen und Jungen der dritten und vierten Klasse der Max & Moritz Grundschule in St. Augustin lernen in mehreren Workshops die Grundlagen der Programmierung kennen und stellen schnell fest, dass man am PC weit mehr machen kann, als nur YouTube-Videos zu schauen. Jörn Diercks, der begleitende Lehrer, zeigt sich vom Konzept begeistert und kann sich gut vorstellen, im nächsten Jahr wieder mit seinen Kids dabei zu sein und die Zusammenarbeit noch zu intensivieren.

25. März 2015

Am letzten Abend der JavaLand 2015 feiert der iJUG den fünften Geburtstag der Zeitschrift Java aktuell. Bei Networking und gutem Essen lassen die Mitglieder der Usergroups, Vertreter von Oracle und einige Autoren, moderiert von Wolfgang Taschner, Chefredakteur der Java aktuell, die letzten fünf Jahre Revue passieren. Ein Film zeigt die Höhepunkte des Abends (siehe „www.doag.org/go/5jahre_javaaktuell").

25. März 2015

Der Vorstandsvorsitzende Dr. Dietmar Neugebauer vertritt die DOAG in Wien auf dem Event „Real World Performance Tour", den die Oracle Applications Users Group (OAUG) in verschiedenen europäischen Städten mit den Oracle-Experten Tom Kyte, Graham Wood und Andrew Holdsworth abhält. Rund 130 Anwender nehmen an der von den Oracle-Anwendergruppen aus Österreich, der Schweiz und Deutschland organisierten gemeinsamen Veranstaltung teil. Am Vorabend gibt es für alle eine Fahrt mit der Oldtimer-Tram durch Wien.

26. März 2015

Ein Schulungstag rundet die JavaLand 2015 ab. Parallel dazu tagt die Mitgliederversammlung des Interessenverbunds der Java User Groups e.V. (iJUG) und stellt bereits die Weichen für die JavaLand 2016, die vom 8. bis 10. März 2016 an gleicher Stelle stattfinden wird.

Die Mitgliederversammlung des iJUG

27. März 2015

Die DOAG startet eine große Umfrage zum Thema „Oracle und Virtualisierungstechnologien", nachdem Oracle die Bewertung des Lizenzbedarfs beim Einsatz der Virtualisierungssoftware VMware vSphere seit der Version 5.1 geändert hat. Die anonyme Online-Umfrage soll Klarheit über die Konsequenzen dieser Umstellung im deutschsprachigen Raum schaffen. Das Ergebnis wird auf der DOAG-Website vorgestellt.

9. April 2015

Die DOAG-Vorstände Dr. Dietmar Neugebauer, Fried Saacke und Christian Trieb telefonieren mit Tom Schiersen, Oracle EMEA User Groups Relationship Manager, über die Zusammenarbeit der Usergroups mit Oracle. Im Mittelpunkt des Gesprächs steht die langjährige Forderung der DOAG, das Programm der von Oracle organisierten internationalen Usergroup-Events deutlich mehr auf die Anforderungen und Bedürfnisse der Anwendergruppen hin abzustimmen, damit sich der Aufwand für die Teilnehmer der Usergroups überhaupt lohnt.

12. April 2015

Die Collaborate 2015, mit rund 5.500 Besuchern eine der größten Oracle-Anwenderkonferenzen der Welt, öffnet in Las Vegas ihre Pforten. Die DOAG ist durch die drei Vorstandsmitglieder Dr. Dietmar Neugebauer, Dr. Frank Schönthaler und Fried Saacke sowie durch Kasi Färcher-Haag, Themenverantwortlicher JD Edwards, vertreten. Sie tauschen sich intensiv mit den Ansprechpartnern der veranstaltenden Independent Oracle User Group (IOUG), Oracle Applications Users Group (OAUG) und Quest International User Group aus. Ergebnis ist die Planung gemeinsamer Projekte auf europäischer und internationaler Ebene mit dem Ziel, den Nutzen für alle Mitglieder der einzelnen Usergroups zu erhöhen. Darüber hinaus werden erste Kontakte zur Oracle Development Tools User Group (ODTUG) aufgebaut und der Austausch mit weiteren europäischen Usergroups intensiviert. Auch die neue Lizenzierungspolitik hinsichtlich der Oracle-Produkte in virtualisierten Umgebungen ist auf der Veranstaltung ein großes Thema.

13. April 2015

Im Rahmen der Collaborate 2015 findet ein Arbeitstreffen zwischen der DOAG-Delegation und der Oracle Applications Users Group (OAUG) statt. Es geht um die Zusammenarbeit auf der DOAG 2015 Business Solutions Konferenz. George Somogyi vom OAUG Board of Directors kündigt seine Anwesenheit auf der Veranstaltung vom 9. bis 11. Juni 2015 in Darmstadt an.

15. April 2015

Das „Just in Time"-Konzept von Oracle auf der Collaborate 2015 funktioniert nicht, denn die Keynote von Mark Hurd, Chief Executive Officer of Oracle Corporation, fällt wegen einer Verzögerung am Flughafen aus. Trotz besten Flugwetters wird das im Anschluss an die Keynote zugesagte Treffen von Mark Hurd mit den Vertretern der Usergroups ebenfalls abgesagt.

21. April 2015

Um dem Ruf der DOAG 2015 Konferenz als führende Datenbank- und Technologie-Konferenz gerecht zu werden, lädt die DOAG am Vortag der Veranstaltung alle Vorsitzenden der europäischen Oracle-Anwendergruppen zum ersten EMEA Usergroup Forum ein. Ziel ist die Förderung der Zusammenarbeit auf internationaler Ebene. Das Vorbereitungskomitee bilden Heli Helskyaho von der Finnischen Oracle Usergroup sowie Ralf Kölling von der DOAG.

22. April 2015

Dr. Dietmar Neugebauer, Vorstandsvorsitzender der DOAG, und Fried Saacke, DOAG-Vorstand und Geschäftsführer, folgen der Einladung von Oracle in den Übersee-Club nach Hamburg. Neben interessanten Vorträgen zu Big Data gibt es eine spannende Rede des Bestsellerautors Edgar K. Geffroy.

23. April 2015

Die Teilnehmer der DOAG 2015 BI informieren sich in München über die neuesten Trends im Bereich Business Intelligence sowie Data Warehouse und tauschen ihre Erfahrungen aus. Dirk Schmachtenberg führt in seiner Keynote mit eindrucksvollen Beispielen vor, wie sich die Welt immer wieder aufs Neue verändert. Erstmals neu im Programm ist ein Grundlagen-Stream für Teilnehmer, die sich noch nicht unmittelbar mit den Themen der Konferenz beschäftigt haben, um das Wichtigste zum Einstieg in die BI- und DWH-Thematik zu erhalten. Michael Klose, Leiter der neu gegründeten BI Community der DOAG, stellt zu Beginn der Veranstaltung seine Ziele vor. Da die Arbeit mittlerweile auf viele Schultern verteilt ist, lassen sich jetzt weitaus mehr Themen angehen, mehr Infos aufbereiten und mehr Veranstaltungen organisieren. Ein Ziel ist es, die DOAG Business Intelligence Konferenz künftig auf zwei Tage auszuweiten.

„Die In-Memory-Option ist vollständig in die Oracle-Datenbank integriert …“

In-Memory ist eine zukünftige Kerntechnologie bei Datenbanken. Christian Trieb, DOAG-Vorstand und Leiter der Datenbank Community, und Martin Klier, DOAG-Themenverantwortlicher für In-Memory, sprachen darüber mit Tirthankar Lahiri, Oracle Vice President Data Technologies and TimesTen.

Fotos: Wolfgang Taschner

Tirthankar Lahiri

Wir gehen davon aus, dass die Entwicklung der In-Memory-Option ein großes Projekt bei Oracle war. Was haben Sie persönlich dabei empfunden?

Lahiri: Es war in der Tat ein großes Projekt innerhalb von Oracle, in das viele unterschiedliche Teams innerhalb von Server Technologies, das ist die Datenbank-Entwicklungsgruppe, eingebunden waren. Was mich am meisten fasziniert hat, war die Tatsache, wie gut wir alle zusammengearbeitet haben. Die In-Memory-Option ist sehr transparent für die Anwendungen geworden und die Performance hat unsere höchsten Erwartungen übertroffen. Die ersten Tests mit dem fertigen Produkt waren überwältigend.

Wann ist die Idee für die In-Memory-Option entstanden?

Lahiri: Wir hatten mit TimesTen bereits seit dem Jahr 2005 eine In-Memory-Datenbank, die zu den Pionier-Produkten für OLTP zählte. Die Datenbank-In-Memory-Option war eine unserer wichtigsten Ideen, um die Leistungsfähigkeit der Oracle-Datenbank zu steigern. Als wir mit den Überlegungen für das Datenbank-Release 12 begannen, war uns klar, dass sich In-Memory zu einer Mainstream-Technologie entwickeln wird.

Welche Rolle hat SAP HANA dabei gespielt?

Lahiri: SAP HANA hatte keinerlei Einfluss auf unsere Überlegungen. Es hat lediglich gezeigt, dass es einen Markt für In-Memory-Technologien gibt. Wir schauen aber nicht auf andere Produkte, wenn wir unsere Roadmap festlegen.

Die Oracle-Datenbank wird in den unterschiedlichsten Umgebungen eingesetzt. Welche Zielgruppe ist die größte für die In-Memory-Option?

Lahiri: Die Haupt-Zielgruppe sind Unternehmen, die große Datenmengen, also Data Warehouses oder Data Marts, in kürzes-

ter Zeit analytisch verarbeiten möchten. Auch große OLTP-Umgebungen zählen dazu. Die In-Memory-Option ist auch ideal für Applikationen wie PeopleSoft oder Siebel CRM, die ebenfalls umfangreiche analytische Arbeitsanforderungen haben.

Was ist der entscheidende Unterschied zwischen TimesTen und der Datenbank-In-Memory-Option?

Lahiri: Der Fokus von TimesTen liegt auf der OLTP-Verarbeitung ohne große Zeitverzögerung – also Transaktionen im Mikrosekunden-Bereich. Die Datenbank-In-Memory-Option ist weniger auf die Zeitverzögerung ausgelegt, sondern auf den hohen Durchsatz der Datenverarbeitung. Beide Lösungen nutzen gleichermaßen den Hauptspeicher, allerdings auf unterschiedliche Weise. Die Stärke von TimesTen besteht darin, dass die komplette Datenbank in die Anwendung eingebettet sein kann und dadurch die Client-Server-Kommunikation über das Netzwerk entfällt. Die In-Memory-Option läuft innerhalb der Datenbank mit dem Fokus auf extrem schnelle Abfragen.

Ist die In-Memory-Option für SAP-Applikationen bereits freigegeben?

Lahiri: Seit dem 31. März 2015 ist die Oracle-Datenbank 12.1.0.2 für SAP freigegeben. Allerdings bedarf es noch weiterer Zertifizierungsaktivitäten im Umfeld von In-Memory. Wir gehen davon aus, dass die Zertifizierung von In-Memory zum 30. Juni 2015 abgeschlossen sein wird. Ich verweise gerne auf die von SAP und Oracle gemeinsam betriebene Informationsplattform zu allen Entwicklungsthemen rund um Oracle und SAP unter http://scn.sap.com/community/oracle. Dort sind alle Daten veröffentlicht.

Sollte ein SAP-Anwender zur Steigerung der Performance HANA benutzen oder die In-Memory-Option der Oracle-Datenbank?

Lahiri: Die In-Memory-Option ist vollständig in die Oracle-Datenbank integriert. Das bedeutet, dass jedes einzelne Datenbank-Feature des Oracle-Stacks ohne Änderung mit der In-Memory-Option funktioniert. Viele andere Anbieter, einschließlich SAP, haben ihre In-Memory-Lösung durch die Kombination verschiedener Produkte aufgebaut. Die Oracle-In-Memory-Option hingegen ist nur eine andere Art der Tabellenorganisation, die auch weiterhin ohne Einschränkung mit jeder Art von Features des Oracle-Ökosystems wie Data Guard, Flashback Query, Replication, Golden Gate, RAC etc. arbeitet. Deshalb würde ich zur In-Memory-Option raten.

Wie viel Know-how ist erforderlich, um die In-Memory-Option effizient einsetzen zu können?

Lahiri: Ein Datenbank-Anwender kann sehr schnell damit umgehen. Es ist genauso einfach, wie beispielsweise einen Index zu erstellen. Man muss nur den Speicher freigeben und die Tabelle auswählen, die In-Memory verarbeitet werden soll. Für die In-Memory-Option sind Änderungen weder an der Applikation noch an der Datenstruktur erforderlich.

Soll ein Kunde, der mehr Performance benötigt, die In-Memory-Option einsetzen oder eine Exadata mit viel Flash-Speicher?

Lahiri: Exadata ist definitiv die beste Plattform für die In-Memory-Option. Es gibt mehrere Gründe dafür. Exadata bietet verschiedene Speicher-Ebenen für die Datenbank. So kann man seine dringend gebrauchten Daten im Arbeitsspeicher halten, um schnelle Abfragen zu ermöglichen. InfiniBand sorgt hier für eine extrem performante Anbindung. Weniger häufig gebrauchte Daten sind im Flash-Speicher untergebracht, wo sie ebenfalls relativ schnell im Zugriff sind. Alle anderen Daten sind auf der Festplatte. Mit der In-Memory-Option ist die Performance natürlich noch um einiges höher. Hier kommen die neuen Möglichkeiten bei den Scans und Joins sowie weitere Optimierungen, wie die des Vector Group By, zum Tragen. Exadata und In-Memory sind das absolute Optimum der heutigen Datenbank-Technologie. Mir ist wichtig, eindeutig klarzustellen, dass unsere In-Memory-Technologie auf allen von Oracle unterstützen Plattformen eingesetzt werden kann. Dies schließt neben Linux und Solaris AIX auch HP-UX, Windows und sogar z-Linux ein.

Zur Person: Tirthankar Lahiri

Tirthankar Lahiri ist Vice President in den Teams für die Oracle RDBMS Data Technologies sowie für die Oracle TimesTen In-Memory Database. Das Data-Technologies-Team ist verantwortlich für Datenformate (wie Tabellen, Index Organized Tables (IOT) oder Indizes), Transaktionen, Space Management sowie Advanced Compression, Flashback Query und Total Recall für die Oracle-Datenbank. Diese Gruppe hat auch die neue In-Memory-Option für die Oracle-Datenbank entwickelt. Tirthankar Lahiri verfügt über 19 Jahre Erfahrung in der Oracle-Datenbank-Organisation und besitzt 18 Patente. Er spielte eine führende Rolle in einer Reihe weiterer Datenbank-Initiativen wie Automatic Memory Management, Active Dataguard, Transparent Data Encryption und TimesTen für Exalytics. Tirthankar Lahiri ist Bachelor of Technology in Computer-Wissenschaft an der IIT, Kharagpur (Indien), und Master of Science für Electrical Engineering an der Stanford University. Er nahm in Stanford am PhD-Programm teil und forschte unter anderem in den Bereichen „Multiprozessor-Betriebssysteme“ und „Semistrukturierte Daten“.

Wenn jemand eine Exadata hat und mehr Performance braucht, sollte er dann sein Geld in die In-Memory-Lizenz stecken oder mehr Flash-Speicher kaufen?
Lahiri: Das hängt von den Daten ab. Wenn jemand riesige Datenmengen hat, ist zusätzlicher Flash-Speicher der richtige Weg. Bei kritischen Daten, die in den Speicher passen, sollte man die In-Memory-Option im Auge haben, vor allem wenn es um Real-Time-Analytics geht. Letztendlich macht die Kombination aus beidem den Charme aus.

Wie ist sichergestellt, dass die Daten im Column Store und im Buffer Cache permanent konsistent sind?
Lahiri: Speziell im Column Store gibt es verschiedene Mechanismen, die feststellen, wenn Daten verändert werden. Deshalb werden die Abfragen im Column Store immer das gleiche Ergebnis liefern wie im Buffer Cache.

In welche Richtung wird sich die In-Memory-Option weiterentwickeln?
Lahiri: Die In-Memory-Technologie wird immer wichtiger, da die Speicherkapazitäten immer größer werden. Normalerweise verdoppelt sich die Speicherkapazität alle drei Jahre, sodass eine Datenbank immer einfacher in den Speicher passt. Heute schon bietet die neue M6-Maschine mit 32 TB Hauptspeicher ungeahnte Möglichkeiten. Auch der Umgang mit In-Memory-Funktionen wird immer einfacher und es werden weitere Optimierungsmechanismen hinzukommen.

10 wichtige Fragen und Antworten zur Oracle-Datenbank-12c-In-Memory-Option

Markus Kißling, ORACLE Deutschland B.V. & Co. KG

Die neue In-Memory-Option ist bei Kunden und Partnern ein heiß diskutiertes Thema – gleichgültig, ob bei einem Oracle Database Day, Oracle Partner Day oder auch bei einem der Kunden-Workshops. Zu zehn häufig gestellten Fragen gibt es hier die passenden Antworten.

Frage 1: *Ich möchte gerne die neue In-Memory-Option testen, im ersten Schritt aber nur die neuen Funktionalitäten kennenlernen. Benötige ich dazu eine separate Hardware?*
Antwort: Man kann sofort mit einer 12c-Installation beginnen – das passende Release vorausgesetzt. Für erste Schritte kann man die Oracle-Datenbank 12c Enterprise Edition, Version 12.1.0.2 (Voraussetzung), direkt auf einem PC oder Notebook mit Linux oder Windows installieren. Wer eine isolierte virtuelle Umgebung bevorzugt, kann beispielsweise Oracle Virtual Box nutzen. In beiden Fällen lässt sich die Funktionsweise der In-Memory-Option sehr gut testen. Auf Oracle Technology Network (OTN) steht die Software im Rahmen des OTN License Agreements zum Download zur Verfügung *(siehe „http://www.oracle.com/us/downloads/index.html")*. Für diejenigen, die ihre eigene Umgebung unter realen Bedingungen mit der In-Memory-Option testen möchten, bietet sich ein separater Test-Server an. Wie bei allen neuen Versionen sollte man sich neben der Produktions-Umgebung eine entsprechende Test-Umgebung aufbauen. Neben der Oracle Database 12c Enterprise Edition Version 12.1.0.2.0 sollte dafür immer das aktuellste Bundle-Patch eingespielt sein (weitere Informationen siehe „Frage 10"). Tests können auch mit dem Oracle-Vertrieb abgesprochen werden.

Frage 2: *Wie aufwändig ist es, die In-Memory-Option im eigenen Unternehmen zu testen? Bietet Oracle hier Unterstützung an?*
Antwort: Es sind nur ein Server mit aktueller Hardware, ein entsprechend großer Hauptspeicher und eine ausreichende Anzahl an CPU-Cores – die die Basis für die In-Memory-Verarbeitung darstellen (Stichwort: SIMD Vector Processing) – erforderlich. Die In-Memory-Option wird über den

„init.ora"-Parameter „inmemory_size" aktiviert und steht sofort nach dem Starten der Oracle-Instanz zur Verfügung *(siehe Abbildung 1)*.

Der Oracle-Vertrieb und die betreuende Systemberatung unterstützen gerne, um die Rahmenbedingungen im eigenen Haus abzuklären. Eine Unterstützung bei einem Proof of Concept ist gegebenenfalls auch möglich.

Frage 3: *Stimmt es, dass Oracle Linux und Solaris die besten Plattformen für die In-Memory-Option sind?*

Antwort: Entscheidend ist, dass es sich bei allen Plattformen um ein und dieselbe Oracle-Datenbank handelt. Neben Oracle Linux und Solaris werden wie gewohnt auch alle anderen Plattformen wie AIX, HP-UX, Windows und z-Linux unterstützt.

Frage 4: *Welchen Vorteil bringt Exadata für die In-Memory-Option?*

Antwort: Der gemeinsame Einsatz der In-Memory-Option mit einer Exadata ist das Nonplusultra in Sachen „Performance". Exadata bietet neben der I/O-Optimierung über die Storage-Server die unterschiedlichen Speicher-Ebenen für die Datenbank an. Die InfiniBand-Anbindung bringt weitere Vorteile. Speziell bei RAC kann dadurch der Column-Store redundant gehalten werden, Stichwort: Fault Tolerance. Dazu gibt es die „DUPLICATE"-Subclause beim Festlegen, welche Objekte in den Column-Store geladen und redundant auf allen Knoten gehalten werden sollen. Zusätzlich bieten die Storage-Server einen enormen Durchsatz, der auch beim parallelen Beladen des Column-Stores hilfreich ist.

Frage 5: *Ist es geplant, dass die In-Memory-Option Bestandteil der Standard Edition wird beziehungsweise ohne Option innerhalb der Enterprise Edition?*

Antwort: Momentan sind keine Pläne dafür bekannt. Wie die Bezeichnung „Option" aussagt, wird In-Memory wohl auch zukünftig separat zur Enterprise Edition zu lizenzieren sein.

Frage 6: *Was geschieht, wenn der Column-Store voll ist?*

Antwort: Man kann trotzdem wie gewohnt weiterarbeiten. Es ist lediglich nicht mehr möglich, weitere Objekte in den Column-Store einzufügen. Während Objekte, die dann darin enthalten sind, auch aus diesem genutzt werden können, werden diejenigen Objekte, die nicht im Column-Store enthalten sind – wie seither auch – über den Buffer-Cache genutzt. Für diesen Fall gibt es im Alert-Log einen Hinweis („Insufficient memory to populate table to inmemory area"). Zudem erscheint beim Aufruf der View „v$inmemory_area" innerhalb des Attributs „POPULATE_STATUS" der Eintrag „OUT OF MEMORY" unter dem 1-MB-Pool. Da es sich beim Column-Store im Gegensatz zum Buffer-Cache um keinen Cache mit einem LRU-Algorithmus handelt, muss sich der DBA nur um die passende Belegung des Column-Stores kümmern.

Frage 7: *Profitiert man trotzdem von der In-Memory-Option, auch wenn nur Teile von Tabellen in den Column-Store geladen sind?*

Antwort: Ja, auf jeden Fall. Der Oracle-Optimizer erstellt einen Plan, der sowohl die geladenen Teile des Column-Stores als auch die übrigen Teile des Objekts aus dem Buffer-Cache nutzt. Der Optimizer Trace („10053 Trace Event") enthält Details über die Nutzung des Column-Stores. Unter „BASE STATISTICAL INFORMATION" gibt es den Eintrag „IMCQuotient", der das Verhältnis aus der Gesamtzahl der Spalten („#Rows") und die Anzahl der Rows, die in den Column-Store geladen wurden („IMCRowCnt"), angibt. Ein „IMCQuotient" von 0.38 bedeutet beispielsweise, dass 38 Prozent der Rows in den Column-Store geladen sind. Die geladenen Rows haben auch Auswirkungen auf schnellere Antwortzeiten, insbesondere, wenn Tabellen über Full-Table-Scans abgefragt werden, da für diese Teile der Column-Store genutzt wird. Noch eine Anmerkung zum Oracle-Optimizer: Dieser ist und bleibt auch im Zeitalter von In-Memory die höchste Instanz. Das regelmäßige Aktualisieren der Statistiken stellt sicher, dass der Optimizer über das Kostenmodell den besten Ausführungsplan aufbauen kann. Der Abgleich mit den neuen In-Memory-Statistiken, die on-the-fly generiert werden, erfolgt dann zur Laufzeit.

Frage 8: *Können SQL-Statements Tabellen sowohl im Column-Store als auch im Buffer-Cache abfragen?*

Antwort: Ja, unbedingt. Dies ist ein Alleinstellungsmerkmal von Oracle. In den Ausführungsplänen ist zu erkennen, dass Tabellen oftmals gemischt genutzt werden. Eine kleine Dimensions-Tabelle kann demnach im Buffer-Cache liegen (Zugriff über „Table Access (FULL)") und die große Fakten-Tabelle im Column-Store (Zugriff über „Table Access (INMEMORY FULL)"). Sämtliche Optimierungen, die der Column Store mit sich bringt, werden dabei angewendet, etwa das Erzeugen und Anwenden von Bloom-Filtern anstelle von Joins oder der Vector „Group By" als weitere Optimizer-Transformation. Zu beachten ist aber, dass diese neuen Möglichkeiten nur mit aktivierter In-Memory-Option („INMEMORY_SIZE > 100 MB") zur Verfügung stehen. Der Mischbetrieb hat den Vorteil, dass man die Abfragen über die Speichergrenze des Column-Stores hinweg höchst effizient ausführen kann, ohne an die starre Größe des Hauptspeichers gebunden zu sein. Besonders interessant ist auch die Table-Expan-

```
[oracle@host-8-217 ~]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0 Production on Sat Aug 9 07:20:49 2014
Copyright (c) 1982, 2014, Oracle. All rights reserved.
Connected to an idle instance.

SQL> startup
ORACLE instance started.
Total System Global Area  6.9793E+11 bytes
Fixed Size                7701464 bytes
Variable Size             4.8855E+10 bytes
Database Buffers          3.2642E+11 bytes
Rede Buffers              529166336 bytes
In-Memory Area            3.2212E+11 bytes
Database mounted.
Database opened.
SQL>
```

Abbildung 1: Die neue In-Memory-Option ist aktiviert

sion-Transformation, die mit der Version 11*g* R2 eingeführt wurde. Dabei werden im Ausführungsplan Partitionen im Column-Store (etwa Daten des aktuellen Jahres) mit denen älterer Jahre kombiniert, die nur im Buffer-Cache vorliegen und auf die über einen Index zugegriffen wird.

Frage 9: *Wie lassen sich Use Cases identifizieren?*

Antwort: Hierzu gibt es von Tirthankar Lahiri, Vice President Oracle Data Technologies and TimesTen, ein aktuelles White Paper auf Oracle Technology Network, das beim Identifizieren helfen kann *(siehe „http://www.oracle.com/technetwork/database/in-memory/overview/twp-dbim-usage-2441076.html“)*. Neben einigen Richtlinien für die Erstellung von effizienten Applikationen im Kontext der In-Memory-Option beschreibt es auch geeignete und weniger geeignete SQL-Abfragen. Grundsätzlich gilt, dass die Applikation für die In-Memory-Option nicht geändert werden muss.

Frage 10: *Welche Tools, Hilfsmittel und weiterführenden Informationen stehen zur Verfügung?*

Antwort: Der Oracle-Compression-Advisor (siehe PL/SQL-Package „DBMS_COMPRESSION“) hilft bei der Abschätzung der Kompressionsrate, die durch den Einsatz der In-Memory-Option erreicht werden kann. Damit kann man die voraussichtliche Größe der Objekte im Column-Store bestimmen. Daneben gibt es den neuen In-Memory-Advisor, der über My Oracle Support (MOS Note 1965343.1) heruntergeladen werden kann (siehe Seite 12). Er hilft bei der Ermittlung von geeigneten Kandidaten für den IM-Column-Store und erzeugt die erforderlichen SQL-Skripte anhand des Workloads der Applikation. Dieser Advisor kann sogar für eine Oracle-11*g*-Datenbank eine Empfehlung abgeben. Oracle Enterprise Manager Cloud Control bietet mit IM-Central eine gute Übersichtsseite über die Belegung des IM-Column-Stores. Auch der Oracle SQL Developer hat eine tiefgehende Integration für die In-Memory-Option erhalten. Dort lassen sich zum einen die Ausführungspläne anzeigen, zum anderen geben die Session-Statistiken Auskunft über die Verwendung des Column-Stores in der jeweiligen Session. Im AWR-Report stehen die IM-Statistiken für die Ebene der Datenbank zur Verfügung.

Weitere Informationen

Die Objekt-Kandidaten für den Column-Store lassen sich auch anhand der Top-SQL-Statements ermitteln, beispielsweise bei einer hohen „Elapsed Time“. Dazu sei auf das wichtige White Paper von Maria Colgan, Master Product Manager, verwiesen, das einen Gesamtüberblick über die In-Memory-Option gibt *(siehe „http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html“)*.

Der „Oracle Database In-Memory-Blog“ *(siehe „https://blogs.oracle.com/In-Memory“)* bietet neben aktuellen Tipps und Tricks auch Informationen zu dem jeweils aktuellen Bundle Patch. Die Patches werden in regelmäßigen Abständen durch My Oracle Support (MOS) zur Verfügung gestellt.

Der „Oracle Database In-Memory Bundle Patch 7“ (April 2015) ist derzeit aktuell und steht für die Datenbank-Version 12.1.0.2.7. Dabei handelt es sich um einen Datenbank-Patch für Engineered Systems und Database-In-Memory (gilt auch für Non-Engineered-Systems-Umgebungen). Zudem ist es kumulativ und beinhaltet deshalb auch alle Fixes der Bundle-Patches 1 bis 6.

Weitere Informationen stehen in der MOS-Note (Doc ID 20698050.8). Außerdem bietet die deutsche Oracle Business Unit „DB Tec“ regelmäßig Webcasts und Veranstaltungen, wie die Oracle Database Days, zum Thema „In-Memory-Option“ und zu allen weiteren relevanten Datenbank-Themen an *(siehe „tinyurl.com/odd12c“ und „tinyurl.com/oraclebudb“)*.

Markus Kißling
markus.kissling@oracle.com

Spiegelung kompletter Systemumgebungen

Libelle Business*Shadow*®

Erfahren Sie mehr:
www.Libelle.com/business

ORACLE Gold Partner

Libelle

Libelle AG
Gewerbestr. 42 • 70565 Stuttgart, Germany
T +49 711 / 78335-0 • F +49 711 / 78335-148
www.Libelle.com • sales@libelle.com

Mit Rat und Tat: Der Database-In-Memory-Advisor

Ileana Someșan, ORACLE Deutschland B.V. & Co. KG

Die 12*c*-In-Memory-Option ist eine neue Möglichkeit, analytische Abfragen zu beschleunigen. Der seit Februar 2015 verfügbare In-Memory-Advisor kann bereits in 11*g*-Datenbanken eingesetzt werden, um das Potenzial der In-Memory-Option aufzudecken. Darüber hinaus gibt der Advisor nützliche Empfehlungen zur Konfiguration der In-Memory-Option und hilft bei der Implementierung. Dieser Artikel beschreibt den In-Memory-Advisor und gibt Tipps für den Einsatz.

Die In-Memory-Option im Datenbank-Release 12.1.0.2 optimiert analytische Arbeitslasten sehr stark. Hundertfach schnellere Analysen und Reports sind nicht ungewöhnlich. Es stellt sich die Frage, ob ein Data Warehouse von der Umsetzung der In-Memory-Option profitieren kann und, wenn ja, welcher Performance-Verbesserungsfaktor zu erwarten ist. Der neue In-Memory-Advisor liefert entsprechende Antworten.

Der Advisor untersucht die analytische Arbeitslast der Datenbank mithilfe von Statistiken aus dem Automatic Workload Repository (AWR) sowie der Active Session History (ASH) und berechnet die voraussichtliche Performance-Steigerung, die sich durch den Einsatz der In-Memory-Option erzielen lässt. Gleichzeitig ermittelt er die optimale Größe des In-Memory Column Store und die Datenbank-Objekte, die von der Spalten-Technologie am meisten profitieren.

Die Ergebnisse des Advisor sind als Schätzungen zu betrachten, da die In-Memory-Option selbst nicht zum Einsatz kommt; vielmehr simuliert der Advisor die Vorteile dieser Technologie. Das Schätzmodell berücksichtigt unter anderem die Eliminierung der Wartezeiten für das Lesen von Festplatte (durch die In-Memory-Datenhaltung), die schnellere Verarbeitung des Spaltenformats (durch SIMD-fähige Prozessoren) oder die positive Auswirkung bestimmter In-Memory-Komprimierungsverfahren auf die Abfragegeschwindigkeit.

Installation und Ausführung

Der In-Memory-Advisor ist im Package „DBMS_INMEMORY_ADVISOR" enthalten. Es gehört nicht zum Standard-Lieferumfang der Datenbank und muss von MyOracle Support (Support Note 1965343.1) heruntergeladen und in der Datenbank 11.2.0.3 oder höher installiert werden (siehe auch das Kapitel „Lizenzierung"). Das erste Release unterstützt noch keine Multitenant-Datenbanken, daher müssen 12*c*-Datenbanken die klassische Non-CDB-Architektur aufweisen. Die Installation ist sehr einfach *(siehe Listing 1)*. Das mitgelieferte Skript führt durch den Installationsprozess und erstellt die erforderlichen User und Schema-Objekte.

Der Advisor wird über das Package „DBMS_INMEMORY_ADVISOR" eingesetzt. Eine grafische Implementierung im Enterprise Manager existiert noch nicht. Für die Ausführung kann man eigene PL/SQL-Skripte entwickeln; die Schnittstelle des „DBMS_INMEMORY_ADVISOR"-Package ist in *[1]* gut dokumentiert.

Schneller und einfacher ist es jedoch, wenn man das mitgelieferte Ausführungsskript nutzt *(siehe Listing 2)*. Es benötigt als Eingabe einen „Task"-Namen, die Beginnzeit und die Dauer der analysierten Arbeitslast. Ein wichtiger Hinweis an dieser Stelle: Der Advisor läuft nicht in Echtzeit mit, sondern analysiert die Datenbank-Aktivität aus einem Zeitfenster in der Vergangenheit, beispielsweise vom 01.04.2015 zwischen 10 und 12 Uhr.

Das Ergebnis

Der Advisor generiert einen HTML-Bericht mit Empfehlungen sowie ein SQL-Skript zu deren Implementierung. Der Advisor-Bericht besteht aus mehreren Abschnitten. Der erste Abschnitt *(siehe Abbildung 1)* gibt einen Überblick über die untersuchte Arbeitslast und zeigt den Anteil der analytischen Anfragen an der gesamten Datenbank-Zeit. Die analytische Datenbank-Aktivität steht im Fokus des Advisor und wird über die nachfolgenden Empfehlungen optimiert.

Der nächste Abschnitt zeigt die optimale Größe des In-Memory Column Store („INMEMORY_SIZE"). Der Advisor führt eine Kalibrierungsübung durch, die den In-Memory Column Store schrittweise vergrö-

```
$ sqlplus sys/<pw> as sysdba
SQL> @instimadv
```

Listing 1: Installation des In-Memory-Advisor

```
$ sqlplus sys/<pw> as sysdba
SQL> @imadvisor_analyze_and_report
```

Listing 2: Ausführung des In-Memory-Advisor

ßert, bis ein Schwellenwert oder der maximale Performance-Verbesserungsfaktor erreicht ist *(siehe Abbildung 2)*. Grundsätzlich gilt: Je größer der In-Memory Column Store, desto größer der Performance-Gewinn.

Total Database Time (Seconds)	Analytics Processing Time (Seconds)	Analytics Processing Percentage
20286	6696	33%

Abbildung 1: Untersuchte Datenbank-Zeit insgesamt, in diesem Beispiel ist ein Drittel als analytische Aktivität eingestuft

Percentage of Maximum Recommended In-Memory Size	Percentage of Current SGA Size (640MB)	In-Memory Size	Estimated Analytics Processing Time Reduction (Seconds)	Estimated Analytics Processing Performance Improvement Factor
100%	45%	286MB	6443	26.4X
95%	43%	272MB	2414	1.6X
90%	40%	258MB	2414	1.6X
85%	38%	243MB	2414	1.6X
80%	36%	229MB	2414	1.6X
75%	34%	215MB	2414	1.6X
70%	31%	200MB	2414	1.6X
65%	29%	186MB	2414	1.6X
60%	27%	172MB	2414	1.6X
55%	25%	157MB	2414	1.6X
50%	22%	143MB	2414	1.6X
45%	20%	129MB	2414	1.6X
40%	18%	115MB	2414	1.6X
35%	16%	100MB	2414	1.6X
30%	13%	86MB	2414	1.6X
25%	11%	72MB	237	1.0X
20%	9%	57MB	237	1.0X
15%	7%	43MB	237	1.0X
10%	4%	29MB	237	1.0X
5%	2%	14MB	237	1.0X

Abbildung 2: Größe des In-Memory Column Store und der erwartete Performance-Vorteil

Es gibt jedoch eine applikationsspezifische Grenze, ab der keine Verbesserung mehr möglich ist. Der Schwellenwert, also die maximale „INMEMORY_SIZE", beträgt standardmäßig die doppelte SGA-Größe und lässt sich über die „GENERATE_RECOMMENDATIONS"-Prozedur anpassen.

Für „INMEMORY_SIZE" mit den besten Performance-Aussichten sind die SQL-Statements aufgelistet, bei denen die größte Zeiteinsparung zu erwarten ist *(siehe Abbildung 3)*. Der nächste Abschnitt ist einer der interessantesten *(siehe Abbildung 4)*. Es zeigt die Tabellen, Partitionen und Sub-Partitionen, die in den In-Memory Column Store geladen werden sollten, damit die vorhergesagte Performance-Verbesserung eintritt. Für jedes Datenbankobjekt wird ein Komprimierungsalgorithmus empfohlen und geschätzt, wie viel Platz es im spaltenbasierten Column Store verbraucht. Der In-Memory-Advisor generiert auch das passende SQL-Skript zur Konfiguration der In-Memory-Objekte *(siehe Listing 3)*. Das Skript kann in der Oracle-Datenbank-Version 12.1.0.2 ausgeführt werden.

Einige Tipps und Bemerkungen

Die In-Memory-Option eignet sich am besten für analytische Arbeitslasten. Je ausgeprägter der analytische Charakter der Datenbank, desto größer die Performance-Vorteile, die der Advisor aufdecken wird.

Der In-Memory-Advisor benötigt unbedingt Datenbank-Statistiken für den Zeitraum, der ausgewertet werden soll. Falls keine oder unzureichende Statistiken vorhanden sind, lassen sich keine Empfehlungen generieren oder die Ergebnisse sind

SQL Id	SQL Text	Analytics Processing Time Used (Seconds)	Estimated Analytics Processing Time Improvement (Seconds) With Unlimited Memory	Estimated Analytics Processing Performance Improvement Factor With Unlimited Memory	Estimated Analytics Processing Time Improvement (Seconds) With 286MB	Estimated Analytics Processing Performance Improvement Factor With 286MB
akabhw7s1rdjm	SELECT t.time_id, to_char(SUM(amount_sold), '9,999,999,999') AS sales, to_char(AVG(SUM(amou...	16	5	1.4X	5	1.4X

Abbildung 3: SQL-Statements, die optimiert werden können

Object Type	Object	Compression Type	Estimated In-Memory Size	Analytics Processing Seconds	Estimated Reduced Analytics Processing Seconds	Estimated Analytics Processing Performance Improvement Factor	Benefit / Cost Ratio (Reduced Analytics Processing / In-Memory Size)
TABLE	SH.TIMES	Memory compress for query low	1MB	356	237	3.0X	153 : 1
TABLE	SH.CUSTOMERS	Memory compress for query low	69MB	2822	2177	4.4X	2 : 1
TABLE	SH.SALES	Memory compress for query low	169MB	5277	4029	4.2X	1 : 1

Abbildung 4: Datenbank-Objekte, die von der In-Memory-Option am meisten profitieren

unzuverlässig. Diagnostics Pack und Tuning Pack müssen aktiv sein, „show parameter control_management_pack_access" zeigt dann 'DIAGNOSTIC+TUNING' an.

Der Advisor sollte auf relevanter Arbeitslast ausgeführt werden, zum Beispiel am Monatsende mit intensiver Reporting-Aktivität. Das optimale Zeitfenster hängt von der Datenbank-Aktivität ab (etwa lang andauernde Abfragen) und kann schrittweise vergrößert werden.

Nach der Umsetzung der Advisor-Empfehlungen sollte der tatsächliche Performance-Gewinn validiert werden. Der SQL Performance Analyzer hilft dabei, die Ausführungszeit von SQL-Abfragen vor und nach der Implementierung der In-Memory-Option zu vergleichen *[2]*.

Der Overhead für die Ausführung des Advisor ist gering, ähnlich wie bei anderen Oracle-Advisors. Der Aufwand kann komplett eliminiert werden, indem man den Advisor auf einem Testsystem ausführt, in das vorher die relevanten AWR-Statistiken importiert wurden. Die Vorgehensweise ist in *[1]* beschrieben.

```
Rem Copyright (c) 2014, 2015, Oracle and/or its affiliates. All rights
reserved.
ALTER TABLE „SH“.“CUSTOMERS“ INMEMORY MEMCOMPRESS FOR QUERY LOW;
ALTER TABLE „SH“.“SALES“ INMEMORY MEMCOMPRESS FOR QUERY LOW;
ALTER TABLE „SH“.“TIMES“ INMEMORY MEMCOMPRESS FOR QUERY LOW;
```

Listing 3: SQL-Befehle für die Konfiguration der In-Memory-Objekte

Lizenzierung

Der In-Memory-Advisor ist Bestandteil des Oracle-Tuning-Packs und verwendet Informationen aus dem Diagnostics Pack. Somit sind Lizenzen für beide Packs sowie für die Oracle Database Enterprise Edition erforderlich.

Weitere Informationen

[1] Oracle Database In-Memory Advisor, J. Raitto und K. Engeleiter, 02.2015
[2] Oracle Database In-Memory Advisor Best Practices, K. Engeleiter, 02.2015

Ileana Someșan
ileana.somesan@oracle.com

Der neue DOAG-Vorstand

Die Delegiertenversammlung hat am 9. Mai 2015 einen neuen Vorstand gewählt. Das Ergebnis:

- Dr. Dietmar Neugebauer, Vorstandsvorsitzender
- Stefan Kinnen, stellv. Vorsitzender, Finanzen
- Michael Paege, stellv. Vorsitzender, Querschnittsgruppen
- Christian Trieb, Datenbank Community
- Robert Szilinski, Development Community
- Jan-Peter Timmermann, Infrastruktur & Middleware Community
- Dr. Frank Schönthaler, Business Solutions Community
- Markus Eisele, Java Community
- Michael Klose, Business Intelligence Community
- Fried Saacke, Geschäftsstelle

v.l.n.r.: M. Klose, F. Saacke, C. Trieb. M. Eisele. S. Kinnen, D. Neugebauer, J. Timmermann, R. Szilinski, M. Paege, F. Schönthaler

Oracle Database In-Memory vs. SAP HANA

Stefan Menschel, ORACLE Deutschland B.V. & Co. KG

Vieles wurde in der jüngsten Vergangenheit zum Thema „In-Memory" veröffentlicht. Allen voran hat es SAP geschafft, durch ein enormes Marketing die Aufmerksamkeit auf ihre SAP-HANA-Technologie zu lenken. So ist es nicht verwunderlich, dass ein großer Teil der Anwender In-Memory Computing zunächst mit dem Produkt der SAP verbindet. Aus meiner Sicht gibt es in der Technologie des In-Memory Computing grundsätzliche Unterschiede. SAP spricht bei SAP HANA von einer „In-Memory-Datenbank", während bei Oracle von „Database In-Memory" die Rede ist.

Die Entwicklung der SAP-HANA-In-Memory-Datenbank, auf Basis der am Hasso-Plattner-Institut in Potsdam entwickelten Sanssouci-Datenbank, ist eine Verschmelzung von drei unterschiedlichen Technologien. Der zeilenorientierte Anteil stammt von „P*Time", einem kleinen, Memory-optimierten OLTP-RDBMS; der spaltenorientierte Anteil kommt vom SAP-eigenen Produkt TREX des SAP BWA, während der Persistence Layer von der SAP-eigenen MaxDB abstammt. Mit SAP HANA wird versucht, alle Daten einer Datenbank hochkomprimiert in der Form von indizierten Spalten im physikalischen Speicher eines Datenbank-Servers abzulegen.

Spaltenorientierte Ablage der Daten ist ein weiteres Merkmal des In-Memory Computing. Dies ist nicht neu, wurde jedoch aufgrund deutlicher Nachteile im OLTP-Umfeld von allen namhaften Datenbank-Herstellern bisher nicht eingesetzt. Erst das enorme Wachstum zu analysierender Daten und der Preisverfall für RAM haben hier zu Veränderungen/Ergänzungen der bisherigen Datenbank-Technologie geführt.

Die SAP-HANA-Lösung hält die Daten primär im Hauptspeicher und verarbeitet sie dort. SAP spricht deshalb von einer In-Memory-Datenbank. Sehr große Datenbanken benötigen hier trotz einer initial hohen Komprimierung enorm viel Hauptspeicher. Der erwähnte Preisverfall der RAM-Module ist nicht ausreichend, da hier doch hohe Kosten allein für die Hardware einer SAP HANA Appliance oder einer SAP Tailored Datacenter Integration (TDI) anfallen.

Da HANA eine reine In-Memory-Datenbank ist, erfolgt die Verfügbarkeit nach einem Server Crash erst nach dem Laden der Objekte in den Arbeitsspeicher. Das ist bei einer kritischen Anwendung wie der SAP Business Suite wichtig, da hier in Abhängigkeit von der Größe zum Teil mehrere Stunden für das Recovery anfallen. Alternativ können kostenintensive Disaster-Recovery-Lösungen vor einer längeren Auszeit schützen.

Die SAP-HANA-Plattform kann in verschiedenen Szenarien, etwa als Accelerator, als Datenbank für das SAP Business Warehouse oder die SAP Business Suite zum Einsatz kommen. Hierfür ist eine vollständige Migration der bestehenden Datenbank mit den SAP-Landscape-Transformation-Services notwendig. Einmal migriert, gibt es bislang keine technisch unterstützten Möglichkeiten, diesen Weg wieder rückgängig zu machen. Dies kommt einer Einbahnstraße gleich und sollte vorher bestens überlegt sein.

Gerade für sehr große Datenbanken im Business-Warehouse-Umfeld ist es nicht erforderlich und sinnvoll, die kompletten Daten dieses Systems in einer reinen In-Memory-Datenbank zu betreiben. Technisch bietet SAP hier ein Nearline-Storage-System auf Basis einer weiteren Datenbank in Form von Sybase IQ an, um diese an die bestehende HANA-Lösung anzudocken. In diesem Fall kann man nicht mehr von einer reinen In-Memory-Datenbank sprechen. Im Gegenteil, für die Haltung der Daten einer einzigen SAP-Anwendung (hier: Business Warehouse) sind zwei getrennt arbeitende Datenbanken notwendig. Das hat Konsequenzen in Form höherer Aufwände für Administration, Disaster-Technologien, Backup und Recovery.

In-Memory und OLTP

Rein spaltenorientierte Datenbank-Systeme – wie auch SAP HANA – haben deutliche Nachteile im Bereich der transaktionalen Prozesse gegenüber klassischen, zeilenorientierten Datenbank-Systemen, etwa wenn Daten angelegt, verändert und gelöscht werden. Hier werden wenige Zeilen mit vielen Spalten gleichzeitig verändert, während man im analytischen Umfeld wenige Spalten mit vielen Zeilen lesen möchte.

Eine grundlegende Veränderung findet nun in HANA in der Umsetzung von Update-oder Delete-Anforderungen statt. Die SAP-HANA-In-Memory-Datenbank arbeitet nach dem „INSERT-ONLY"-Ansatz, um alle bisher bekannten Performance-Probleme einer spaltenorientierten Datenbank zu umgehen. Das bedeutet, aus jedem „UPDATE" wird intern ein „INSERT". Diese oder auch richtige „INSERTs" werden in einen schreib- und zeilenorientierten, nicht komprimierten „Delta Store1" geladen. Von dort werden die Daten in einen zweiten, nun spaltenorientierten,

nicht sortierten „Delta Store2" übertragen. Was folgt, ist eine finale Übertragung in den „Main Store". Das Wachsen der Delta-Stores führt zunehmend zu einer schlechteren Performance weiterer analytischer Abfragen. Aus diesem Grund muss intern ein Merge-Prozess stattfinden, der die Einträge in den jeweiligen Delta-Stores verarbeitet. Der Prozess vom „Delta Store2" in den „Main Store" ist dabei sehr kritisch und beeinträchtigt die Performance der SAP-HANA-Lösung *(siehe Abbildung 1)*.

Dieser notwendige Merge-Prozess sowie weitere interne Services erfordern zusätzlich etwa 50 Prozent RAM zu den im Main Store abgelegten operativen Daten. Die Delta-Stores führen zu einer zeitlich begrenzten, mehrfachen Ablage von gleichen Daten im Memory. Alle von der Datenbank mit einem „Commit" abgeschlossenen Transaktionen werden in den Redo-Log-Dateien vor Datenverlust gesichert. Zusätzlich wird alle fünf Minuten ein Savepoint eines kompletten konsistenten Image der In-Memory-Datenbank auf die vorhandenen Disks geschrieben.

SAP HANA und „Code-Push-Down"

Mit der HANA-Lösung kommt nicht nur eine neue Datenbank, sie beinhaltet zunehmend auch Teile des Applikationscodes. SAP führt hier einen Wechsel weg von einem lange propagierten Paradigma durch. Bisher wurde der Applikationscode immer auf dem SAP-Applikationsserver ausgeführt. Mit SAP-HANA wird versucht, immer mehr Anwendungscode auf der SAP-HANA-Plattform auszuführen. Dies hängt zum einen damit zusammen, dass bisherige Statements fast immer mit einem „SELECT *" erfolgten. Das hat/hatte zur Folge, dass alle Spalten einer Tabelle sofort in das RAM geladen werden/wurden. Um dieses Verhalten zu umgehen, werden Zugriffe auf die notwendigen Spalten über verschiedene Views, Attribute-, Analytical- oder Calculation-Views realisiert. Dies ist auch der Grund, aus dem Anwender ihre eigenen Skripte entsprechend anpassen müssen, um die Vorteile der HANA-Datenbank optimal nutzen zu können.

Zusätzlich wird mit SAP-ABAP-Core-Data-Services versucht, vorhandene Funktionalitäten der HANA-Datenbank wie „REPLACE", „SUBSTR" oder „LPAD" zu nutzen, oder es werden Funktionen zur Umwandlung von Einheiten, Währungen sowie Datumsformaten in der Datenbank abgelegt und dort ausgeführt. Diese Services stehen nicht nur ausschließlich für die HANA-Datenbank zur Verfügung, sondern werden ebenso bei den traditionellen Datenbanken wie Oracle zunehmend eingesetzt. Das kann zu Vorteilen in der Performance bei der Ausführung von Applikationslogik führen, muss es jedoch nicht zwangsläufig. Denn heute gibt es für sehr große SAP-Systeme eine Vielzahl von Applikationsservern, um die Last der Anfragen Tausender Anwender über viele Systeme zu verteilen. In Zukunft würde diese Last auf einem einzigen Server abgearbeitet, der zusätzlich neben dem In-Memory Computing die CPU noch stärker belastet. Die genauen Auswirkungen eines derart massiven „Code-Push-Down"-Prinzips sind heute im SAP-Umfeld überhaupt noch nicht absehbar.

Die Oracle-Database-In-Memory-Option

Bisher war die Oracle-Datenbank nicht rein Disk-basiert. Optimierungen mithilfe von Speicher-Architekturen wie Buffer-, Row-, Result-Cache etc. waren Voraussetzungen für eine optimale Performance bei mehr als 95 Prozent aller transaktionalen und analytischen Anforderungen an die Datenbank. Mit der Oracle-Datenbank-Version 12.1.0.2 steht eine neue Memory-Technologie zur Verfügung. Oracle Database In-Memory adressiert genau den Bereich von Anforderungen, in dem heute und zukünftig schnellere Analysen oder Aggregationen über große Datenmengen und wenige Spalten erforderlich sind. Anderweitige allgemeine Vorstellungen über generell deutliche Verbesserungen in der Performance durch SAP HANA oder auch durch die neue Oracle-Datenbank-In-Memory-Technologie haben nichts mit der Realität zu tun.

Die Datenbank-Version 12*c* erhält zusätzlich zu den bisherigen Elementen in der System Global Area (SGA) einen weiteren Bereich, den In-Memory Column Store. In diesem Bereich des Hauptspeichers können Tabellen vollständig, einzelne Spalten einer Tabelle oder einzelne Partitionen im Spaltenformat abgelegt werden. Allein die Bezeichnung „Database In-Memory" soll verdeutlichen: Die Oracle In-Memory-Technologie ist etwas anderes als eine „In-Memory-Datenbank". Mit dieser Option verbindet Oracle als einziger Datenbank-Hersteller beide notwendigen Technologien in einer einzigen Datenbank.

Die bisherige Praxis einer zeilenorientierten Pufferung im Hauptspeicher und deren Speicherung auf Disk oder Flash bleibt vollwertig bestehen. Dieser Ansatz hat sich über viele Jahre bis heute für transaktionale Anforderungen bestens bewährt und bleibt unverändert. Im Gegensatz zu dem vorhandenen Buffer Cache wird der Column Store nicht auf den Storage-Systemen (Disk oder Flash) gesichert, sondern liegt ausschließlich im

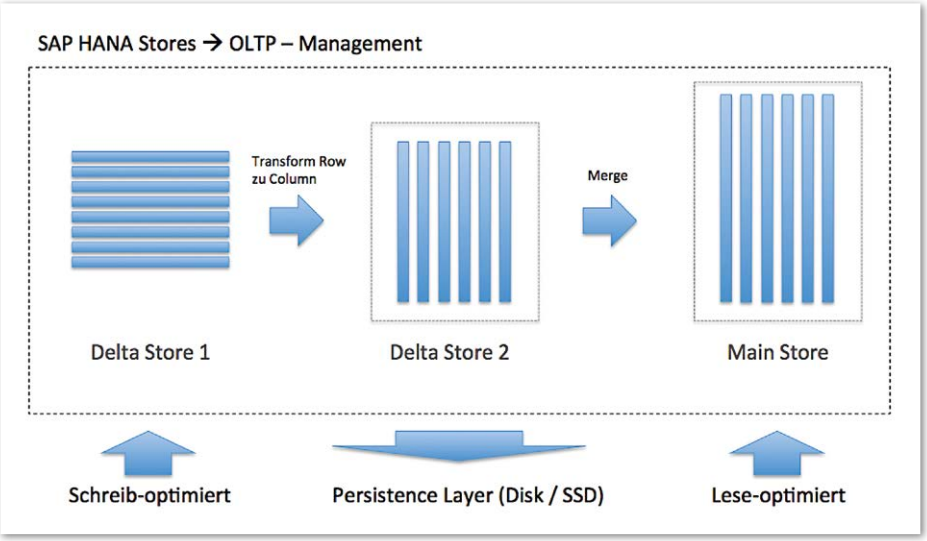


Abbildung 1: Verarbeitung der OLTP-Daten in verschiedenen Stores

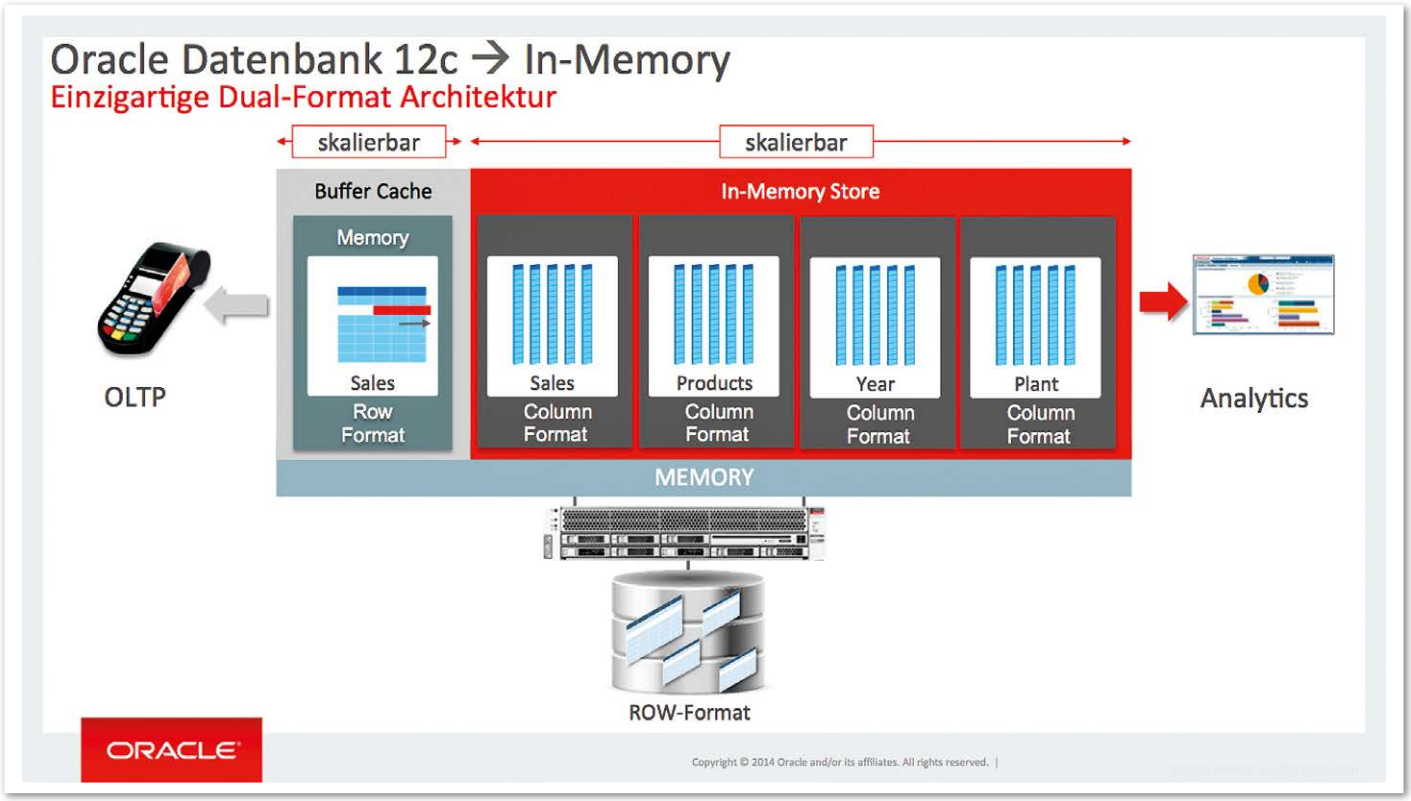


Abbildung 2: Oracle Database In-Memory

Hauptspeicher in dieser Form vor. Die Oracle-Datenbank ist somit eine hybride Datenbank, bestehend aus einem zeilenorientierten Datenbestand und einer spaltenorientierten, hoch komprimierten Repräsentation im Hauptspeicher. Damit ist die Datenbank in der Lage, alle Anforderungen hinsichtlich OLTP wie gewohnt weiterhin zu unterstützen.

Zusätzlich entstehen nun über den Column Store Verbesserungen für analytische Abfragen. Der Oracle-Datenbank-Optimizer erhält mit der Nutzung des Column Store eine weitere Möglichkeit, einen optimalen Zugriffspfad für die jeweilige Anforderung zu finden. Dies muss nicht immer der Bereich im Column Store sein; bisherige Technologien wie „UNIQUE INDEX" oder auch „BITMAP INDEX" können in der Performance für „Single Row Select"-Anforderungen durchaus schneller sein als aufwändige Joins über mehrere Spalten im Column Store.

Synchronisierungen im Buffer Cache infolge von Daten-Veränderungen werden innerhalb des Memory durch Transaktions-Journale sehr effektiv und performant ausgeführt. Beide Elemente des Hauptspeichers – sowohl Buffer Cache als auch In-Memory Store – lassen sich flexibel auf die jeweiligen Anforderungen hin dimensionieren. Dabei kann die Datenbank individuell auf ihre Hauptaufgabe eingestellt werden – entweder mehr transaktional oder mehr analytische Last. Beide Elemente des Hauptspeichers erfordern keine permanente Redundanz der Objekte *(siehe Abbildung 2)*.

Die Aktivierung dieser neuen Technologie erfolgt in zwei einfachen Schritten. Ein Parameter für die Dimensionierung des In-Memory Store in der Initialisierungsdatei der Datenbank sowie die Auswahl der Tabellen für ein Spaltenformat reichen aus. Dies bedeutet: Alle bestehenden Anwendungen können diese neue Technologie ohne Veränderung im Code auf allen traditionellen Infrastrukturen nutzen. Alle bisher nutzbaren Eigenschaften der Oracle-Datenbank wie Komprimierung, Verschlüsselung, Real Application Clusters oder Data Guard können unverändert weiter genutzt werden.

Fazit

Mit der neuen In-Memory-Technologie sind mit Oracle keine aufwändigen Migrationen, keine neuen Infrastrukturen oder Änderungen an der Applikation und den bestehenden Prozessen für Backup und Recovery notwendig. Für SAP-Anwendungen wird die Freigabe dieser Technologie Ende des ersten Halbjahres 2015 erwartet. Ein unbedingter Wechsel auf eine aus Walldorf vorgegebene neue Datenbank mit allen Risiken wird damit unnötig. Es besteht kein Grund, eine zuverlässige Oracle-Datenbank auszutauschen.

Stefan Menschel
stefan.menschel@oracle.com

Virtualisierung im Data Warehouse mit In-Memory

Alfred Schlaucher, ORACLE Deutschland B.V. & Co. KG

Warehouse-Systeme sind heute aus den Unternehmen nicht mehr wegzudenken. Ohne sie geht es nicht, das ist klar. Umso verständlicher ist es, dass man an vielen Stellen der Methodik, der Architektur und der Technologie heute kräftig um Verbesserungen ringt. Die Diskussion ist aktuell sehr lebhaft.

Fast immer beklagen sich Anwender über schlechte Antwortzeiten. Mindestens genauso intensiv bemängeln sie fehlende Flexibilität und fehlende Schnelligkeit bei der Umsetzung neuer Informationsanforderungen durch die IT-Abteilungen. Sie führen die Komplexität der Systeme und die Starrheit der Modelle ins Feld. Neue Vorgehensweisen der agilen Software-Entwicklung oder auch Modellierungspraktiken wie Data Vault werden schließlich als Allheilmittel gepriesen. Allerdings liegen die Ursachen oft in jahrelanger Fehlentwicklung und fehlender Konsequenz bei der Umsetzung wichtiger Prinzipien und Techniken. Diese Diskussion soll jedoch hier nicht weiter ausgeführt werden; dazu gibt es bereits Artikel und Veranstaltungen.

Die In-Memory-Technologie bereichert die machbaren Möglichkeiten und ist im Data Warehouse mehr als nur eine Optimierung von Abfrage-Performance. Sie kann die Architektur des Data Warehouse verändern. Verstehen kann man dies nur, wenn man sich die Ablaufprozesse in einem Data Warehouse und die Aufgaben der einzelnen Schichten vor Augen hält. Ohne die klassische Aufgabenverteilung des Inmon'schen Drei-Schicht-Modells zu verlassen, ist diese mit In-Memory effizienter realisierbar. Teile davon (Data-Mart-Schicht) lassen sich virtualisieren, wodurch die Flexibilität sowie die Schnelligkeit bei der Realisierung von Anwender-Anforderungen erhöht werden. ETL-Aufwand entfällt, Plattenplatz kann gespart werden.

Im Weg stehen allerdings einige unpassende Praktiken: In vielen Unternehmen führen die Data Marts ein umfangreiches Eigenleben und wirken wie in Beton gegossen, was eine Virtualisierung erschwert. Der später erwähnte Test des Wegwerf-Data-Mart wird fehlschlagen. Aber das soll niemanden daran hindern, über neue Konzepte nachzudenken, denn nur Visionen bringen uns weiter.

Dieser Artikel spannt einen großen Bogen und er zeigt sehr unterschiedliche Aspekte. Erst in der Gesamt-Betrachtung wird der Zusammenhang klar. Es beginnt bei den Voraussetzungen, der Kenntnis der In-Memory-Option und dem Wissen um die Abläufe in einem Data Warehouse. Weil der Artikel die Aufgabenverteilung innerhalb des Schichtenmodells berührt, wird anschließend auf die Modellierung im Data Warehouse eingegangen und gezeigt, wo und wie etwa die Hierarchisierung von Dimensions-Leveln in einem Data Warehouse entsteht. Zum Schluss wird virtualisiert. Die Technik ist schnell erklärt. Nur ist sie auch tragbar? Das überprüft ein Szenario in einer Versuchsumgebung, die Praxis demonstriert die Machbarkeit.

In-Memory im Data Warehouse

In-Memory gab es bei Oracle immer schon. Bislang wurden Datenbank-Blöcke im sogenannten „Buffer-Pool" im Cache gehalten, was wiederholtes Lesen der Daten um Faktoren beschleunigte. Seit dem Release 12 kann sogar eine komplette Datenbank im Buffer-Pool liegen, sofern er groß genug ist. Für Analysen in einem Data Warehouse oder, anders formuliert, für Abfragen auf Daten-Bereiche (im Gegensatz zum OLTP-Einzelsatz-Lesen) liefert die In-Memory-Option einen zusätzlichen Speicherbereich, den „Column-Store". „Zusätzlich" bedeutet, dass die zu lesenden Daten zweimal vorhanden sind: traditionell zeilenorientiert auf den Platten des Storage-Systems beziehungsweise im Buffer-Pool und zusätzlich als spaltenorientierte Kopie im neuen Column-Store. Data-Warehouse-Abfragen auf den neuen Column-Store erfahren gegenüber dem Lesen aus dem Buffer-Pool eine zusätzliche Performance-Steigerung, obwohl beide Lesevorgänge im Hauptspeicher stattfinden.

Zum Verständnis ist es sinnvoll, sich die Lastaufteilung im Verlauf einer Lese-Verarbeitung vorzustellen. Dabei verbringt das System etwa 30 Prozent der Zeit mit dem Zugriff auf die zu lesenden Daten, 40 Prozent werden zur Lösung von Joins und die restlichen 30 Prozent für Berechnungen, Aggregationen etc. aufgebracht. Man sieht, dass die reine In-Memory-Speicherung der Daten an sich nur einen 30-Prozent-Anteil zur Performance-Optimierung beiträgt. Die wichtigsten Effekte, die durch das schnelle In-Memory-Lesen möglich sind, sind nachfolgend kurz zusammengefasst:

- *Spaltenorientierung*
  Datensätze sind entsprechend ihren Spalten gespeichert. Domain-bezogene Abfragen, bei denen man große, zusammenhängende Mengen von Werten einzelner Tabellenspalten benötigt, müssen nur diejenigen Spalten lesen, die sie auch wirklich benötigen. Das Lesen über den satzorientierten Buffer-Pool muss immer komplette Tabellensätze anfassen und dann die Spalten extrahieren. In einem OLTP-Betrieb, in dem nur einzelne Sätze zu lesen sind, fällt das nicht ins Gewicht, aber bei Massenabfragen

im Data Warehouse schon. Deswegen ist eine spaltenorientierte Speicherung für Warehouse-Systeme optimal.

- *Dynamischer Column-Index*
  Innerhalb der Spalten sind die Daten in Teilmengen organisiert. Die Max-/Min-Werte pro Teilmenge sind bekannt. Das macht das Positionieren noch schneller.

- *Komprimierung*
  Spaltenorientiert gespeicherte Daten lassen sich höher komprimieren, wodurch die zu lesende Datenmenge um den Faktor sechs bis zwanzig sinkt. Die Verarbeitung erfolgt direkt auf den komprimierten Daten. Eine Dekomprimierung mit zusätzlichen Schreibvorgängen in Temp-Bereiche findet nicht statt.

- *Single-Instruction-Multiple-Data-Verfahren (SIMD)*
  Daten werden spaltenweise direkt in die Register der CPUs gepackt und gleichzeitig verarbeitet; die Verarbeitungsgeschwindigkeit steigt. Das Verfahren existiert schon länger, der Einsatz ergibt aber erst jetzt durch die hohe In-Memory-Lesegeschwindigkeit einen Sinn.

- *Höhere Parallelisierung*
  Durch den Wegfall des langsamen Lesens von den Festplatten kann man die Leistungsfähigkeit der CPU-Cores besser nutzen, höhere Parallelisierungsraten zeigen eher Wirkung. Bislang konnten Data-Warehouse-Umgebungen ihre CPU-Power durch Parallelisierung kaum ausreizen, da sie oft durch eine schleppende Datenversorgung des Plattensystems ausgebremst waren. Mehr Parallelisierung bringt zusätzlich Geschwindigkeit.

- *Bloom-Filter*
  Bei Joins zwischen großen und kleinen Tabellen kommen Bloom-Filter zur Anwendung. Das verhindert zusätzliche Temp-Bereiche. Die als Bloom-Filter-Objekte umgewandelten Schlüsselspalten der kleineren Tabellen werden direkt auf die größere Tabelle gemappt.

- *In-Memory-Aggregation*
  Das Verfahren parkt die sogenannten „Group-Werte" von Aggregaten in Zwischenobjekte mit Zwischensummen. Eine zusammenfassende Schlussverarbeitung auf voraggregierten Daten bringt noch mehr Performance.

Ablauf-Prozesse in einem Data Warehouse

In-Memory ist zunächst nur eine Technologie, um Datenbank-Abfragen extrem zu beschleunigen. Bei einer zweiten Betrachtung hilft In-Memory, die Abläufe in einem Data Warehouse kürzer und effizienter zu gestalten. Bevor wir die Abläufe umgestalten, sollten wir sie uns noch einmal kurz vergegenwärtigen *(siehe Abbildung 1):*

tdwi EUROPE
EUROPÄISCHE
TDWI KONFERENZ 2015
mit BARC@TDWI Track
BARC
A CXP GROUP COMPANY
22. – 24. JUNI 2015 | MOC MÜNCHEN
TDWI Konferenz 2015
Jetzt anmelden und Teilnahme sichern
Direkt zum Programm
www.TDWI-Konferenz.de

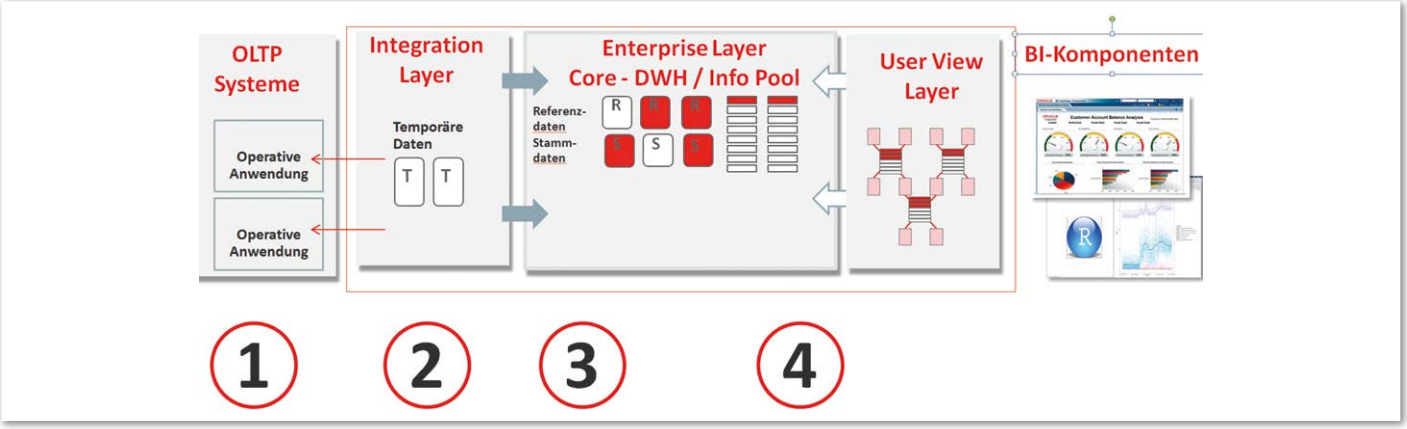


Abbildung 1: Die Abläufe in einem Data Warehouse

1. Selektion von relevanten Datenausschnitten aus den operativen Anwendungssystemen: Ein Data Warehouse hat eine Komplexitäts-reduzierende Aufgabe. Das gelingt, indem nur diejenigen Daten in das Warehouse gelangen, die die Endanwender tatsächlich benötigen. Umso weniger Ballast muss im Data Warehouse beherrscht werden. Die meisten Systeme verfügen über zu viele Daten – zu viele Tabellen und zu viele Spalten innerhalb der Tabellen. Das ist meist die Folge unkontrollierten Wachstums und fehlender Abstimmung mit den Benutzern des Systems.

2. Integration und Harmonisierung von Daten aus unterschiedlichen operativen Anwendungen: Das Data Warehouse ist eine einheitliche und einmalige Daten-Grundlage für Analysen zu möglichst allen Geschäftsprozessen im Unternehmen. Da die gleichen Daten in mehreren Geschäftsprozessen oft unterschiedliche Verwendung finden, muss ein Data Warehouse für diese Unterschiedlichkeit einen einheitlichen Verwendungsrahmen schaffen. Im Fokus dieser Harmonisierung stehen die Geschäftsobjekte als Kontext zu den Bewegungsdaten (Business Events).

3. Granularisieren: Für die weitere Verwendung der Daten ist eine kleinstmögliche granulare Form zu wählen, denn granulare Daten lassen sich leichter für die späteren Analysen kombinieren (Enterprise Layer). Dies ist nicht zwingend die dritte Normalform (3NF), aber mindestens das unterste Drill-Level aller späteren Analysen.

4. Aufbau eines endbenutzer- und analysegerechten Datenmodells. Multidimensionale Modelle stellen eine Kombination von Kennzahlen mit drillfähigen Geschäftsobjekten dar. Beispiele sind Star-Schemata mit Dimensionen und Fakten-Tabellen (User View Layer/ Data Marts).

Die Informationen von Enterprise- und User-View-Layer sind prinzipiell redundant, allerdings anders strukturiert. Das ist die Voraussetzung für eine Virtualisierung der multidimensionalen Strukturen des User-View-Layers, wie noch gezeigt wird. Dazu müssen alle Informationen, auch die der Hierarchisierung potenzieller Dimensions-Level, schon in der Data-Warehouse-Schicht erkennbar beziehungsweise ableitbar sein. Dies ist in vielen Data-Warehouse-Systemen leider oft nicht der Fall. Häufig wird in der zentralen Data-Warehouse-Schicht ein 3NF-Modell direkt aus den Vorsystemen heraus aufgebaut, ohne Endbenutzersichten zu berücksichtigen.

Modellierung im Data Warehouse – ein Muss!

Multidimensionale Modelle (wie Star-Schemata) dienen im User-View-Layer dazu, Endbenutzer in ihrer gewohnten Sprache und mit den ihnen vertrauten Objekten anzusprechen. Sie organisieren Daten in einer analysegerechten Form: Die sogenannten „Facts“ (Kennzahlen

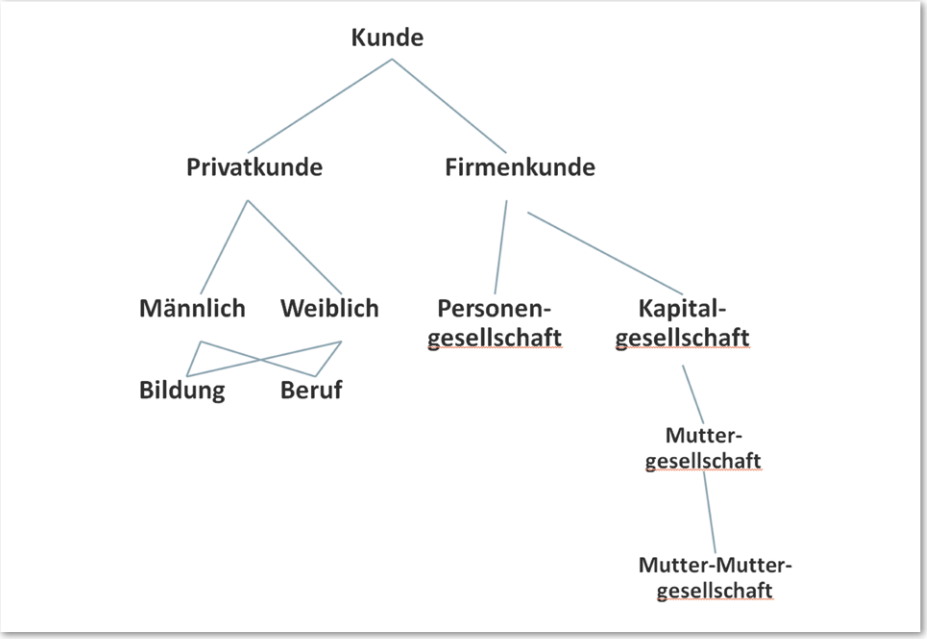


Abbildung 2: Spezialisierung/Generalisierung von Geschäftsobjekten

über das, was stattfindet beziehungsweise bewegt wird) stehen den statischen Geschäftsobjekten gegenüber.

Befragt man Anwender nach den ihnen vertrauten Geschäftsprozessen, so werden sie die Zusammenhänge aus ihrem Kontext heraus schildern. Die Geschäftsobjekte (Kunden, Produkte) werden dabei so geschildert, wie sie gerade passen; eine Strukturierung findet nicht statt. Das ist genau die Aufgabe des Modellierers von Data-Warehouse-Modellen. Er muss die Aussagen von Fachanwendern so sortieren, dass Zusammenhänge zwischen Geschäftsobjekten erkennbar werden. Geschäftsobjekte unterscheiden sich durch unterschiedliche Beschreibungsattribute, sie lassen folgende Aktionen zu:

- *Strukturieren*
  Verfügen zwei unterschiedliche Geschäftsobjekte über eine Anzahl gemeinsamer Merkmale (Attribute), so lassen sich diese Geschäftsobjekte auf einer globaleren Ebene zusammenführen (Klassifizieren). Globalere Zusammenhänge ergeben einen Hierarchie-Level.

- *Beschreiben*
  Die einem Geschäftsobjekt eigenen Ausprägungen von Eigenschaften, die nicht mit anderen Geschäftsobjekten geteilt werden, zeigen detaillierte Hintergründe zu dem Geschäftsobjekt auf. Beschreibungen kommen auf allen Hierarchie-Leveln vor.

Die Strukturierungen von Geschäftsobjekten kommen in einem für Analyse-Zwecke aufbereiteten Datenmodell als sogenannte „Drillpfade" vor. Benutzer der Modelle navigieren über festgestellte gemeinsame Merkmale hinweg entweder in eine globalere oder detailliertere Wahrnehmung der Geschäftsobjekte. In fertigen Analyse-Modellen wie Star-Schemata reden wir auch von „Aggregations-Level". Die Leistung eines solchen Datenmodells liegt also in dem Arrangement von möglichen Detaillierungs-Leveln bei der Betrachtung der Geschäftsobjekte. Es sind die von Anwendern gewünschten Sichten auf die Geschäftsobjekte, die das Modell in einer aggregierenden Art organisiert.

Ein Beispiel ist das globale Geschäftsobjekt „KUNDE", das mehrere Beschreibungsmöglichkeiten auflistet *(siehe Abbildung 2)*. Wie die Grafik zeigt, ist die Hierarchisierung von Geschäftsobjekten kein Selbstläufer. Lassen sich Kunden noch leicht von einem globalen Level aus kommend in die detailliertere Sicht „PRIVATKUNDE" und „FIRMENKUNDE" unterteilen („Spezialisierungs-Vorgang"), so ist eine weitere Spezialisierung über die beschreibenden Attribute wie „Geschlecht", „Beruf" oder „Gehalt" nur für einen Teil der Kundenarten möglich, nämlich die Privatkunden. Auf den Spezialfall „FIRMENKUNDE" passen diese Beschreibungen nicht, er kann nicht männlich oder weiblich sein.

Man führt ein Beschreibungsattribut wie „KUNDENART" ein und macht damit gleichzeitig je nach eingetragenem Wert in diesem Attribut weitere sinnvolle Beschreibungsattribute möglich. Die Verwendung von Werten in den Attributen „Geschlecht", „Beruf" oder „Gehalt" ist nur sinnvoll, wenn in „KUNDENART" ein Wert für einen Privatkunden markiert ist. Dahinter stehen zwei Geschäftsobjekte, „PRIVATKUNDE" und „FIRMENKUNDE", die über eine Art „Super-Geschäftsobjekt" anhand von gemeinsamen Merkmalen vereint wurden. Dort, wo unterschiedliche Attribute in den „Unter-Geschäftsobjekten" auftreten, existiert eine Trennlinie.

Fachanwender beschreiben Kunden nicht mit dieser Präzision. Wenn sie in einer Abteilung für Firmenkunden angestellt sind, reden sie pauschal von Kunden; arbeiten sie für die Privatkunden-Abteilung, so reden sie auch über Kunden. In einem Analyse-Modell, das für übergreifende Auswertungen genutzt werden soll, müssen Privat- und Firmenkunden zusammenhängend, aber auch differenziert auswertbar sein.

Die beschriebene Vorgehensweise ist Teil der sogenannten „Multidimensionalen Modellierung". Man strukturiert die Beschreibungsattribute von Geschäftsobjekten, indem man diejenigen Attribute zusammenfasst, die eine eindeutige Geschäftsobjektsicht ergeben. Die zu Gruppen zusammengefassten Attribute ordnet man hierarchisch an („Dimensionen") und verbindet sie über Business-Aktivitäten („Facts") miteinander. Es entsteht eine multidimensionale Sicht („Star-Schema"). Das Modell lässt sich beispielsweise durch eine ebenfalls hierarchisch strukturierte Zeit- und Orte-Dimension ergänzen. Automatisch vervielfältigt sich damit die Anzahl der Analysenabfrage-Kombinationen.

Es gibt für die Modellierung solcher Zusammenhänge zwar Methoden, aber keine festen Regeln, sondern nur ein Ziel: Das Analyse-Modell muss dem Verwendungszweck der Anwender folgen. Bevor man also solche Hierarchisierungen festlegt, sind die potenziellen Abfragen auf das Modell zu ermitteln:

- Auf welchen Leveln wird aggregiert?
- Welches sind abfragbare Merkmale?
- Wie granular werden Geschäftsobjekte betrachtet?
- Welche Geschäftsobjekte werden gemeinsam analysiert?

Für unser Thema (Virtualisierung von Strukturen beziehungsweise ganzen Modellen wie einem Star-Schema) soll diese Betrachtung genügen. Wir haben Geschäftsobjekte identifiziert und können sie über gemeinsame Merkmale zu Super-Objekten generalisieren oder durch trennende Merkmale in Sub-Objekte spezialisieren. Es entsteht eine Hierarchisierungs-Information, die Anwender als Drill-Pfade nutzen können.

Wahlfreie Kombination der Hierarchie-Level

Spannend ist die Frage, wie man mit dieser Hierarchisierungsinformation umgeht. Die zentrale Data-Warehouse-Schicht lässt sich zur Ablage der Level-Informationen nutzen. Erfolgreiche Data-Warehouse-Konzepte legen in der zentralen Warehouse-Schicht nicht einfach nur platt normalisierte Informationen der Vorsysteme ab, sondern nutzen das Normalisierungsverfahren auch dazu, um potenzielle Hierarchisierungen von normalisierten Objekten für die folgenden Star-Schemata vorzubereiten.

Man fasst „functional-dependant"-Attribute zu eigenständigen Objekten zusammen. Es ist das Aufspalten von gesammelten Informationen in einen redundanzfreien und granularen Zustand. Das muss nicht zwingend die dritte Normalform sein, aber es sind in dem zentralen Data Warehouse diejenigen Objekte („Attribut-Gruppen"), die sich für eine Verwendung im User-View-Layer hierarchisieren und zu denormalisierten Dimensionen gruppieren lassen.

Wenn es gelingt, diese Level-Objekte und die möglichen Beziehungen untereinander (hierarchisch oder parallel) in der zentralen Data-Warehouse-Schicht zu modellieren, sind die Voraussetzung für ein flexibles Analyse-Modell im nachfolgenden User-View-Layer für die Anwender geschaffen und damit die Voraussetzung für eine Virtualisierung.

Diese Level-Objekte in der Data-Warehouse-Schicht sind entsprechend ihren Hierarchisierungsregeln wahlfrei kombinierbar. Es entstehen Dimensionen mit mal mehr und mal weniger hierarchisch oder parallel organisierten Hierarchie-Leveln. Ein expliziter Modellierungsschritt im User-View-Layer (Data Marts) oder sogar später im BI-Werkzeug ist überflüssig. Dort wird nur noch aus den vorbereiteten Level-Objekten der zentralen Data-Warehouse-Schicht selektiert. In dem hier genutzten, einfachen Kundenbeispiel sind vier Dimensions-Varianten möglich *(siehe Abbildung 3):*

- Eine reine Kunden-Dimension mit einfachen Stammdaten
- Eine Privat-Kunden-Dimension
- Eine Firmen-Kunden-Dimension
- Eine Privat-Firmen-Kunden-Dimension

Dokumentation von Level-Objekten in der zentralen Data-Warehouse-Schicht

Es bleibt die Frage, wie die Level und ihre Beziehungen zueinander in der zentralen Schicht vorgehalten werden. Dazu bieten sich eine elegante und eine einfache Möglichkeit an. Die einfachste Lösung ist das „1:1"-Ablegen der möglichen Hierarchie-Level-Objekte als relationale Tabellen in der Datenbank. Die Tabellen werden mit Foreign Keys verknüpft. Zur Ableitung möglicher Dimensionen muss man entweder die Beziehungen zwischen den Tabellen kennen oder man wertet über das Datenbank-Dictionary die Foreign-Key-Beziehungen aus. Nicht elegant, aber einfach und schnell umsetzbar.

Die eleganteste Lösung sind Metadaten. Dies setzt jedoch eine wahlfreie Metadaten-Verwaltung voraus, worüber die wenigsten Data-Warehouse-Systeme verfügen. Über Metadaten lassen sich Dimensionen automatisch generieren. Mögliche Strukturen kann man als Strukturauflösungs-Baum grafisch darstellen. Letztlich könnten sich sogar Anwender ihre Dimensionen selbst zusammenklicken. Ohne das Thema hier vertiefen zu wollen, sind in *Abbildung 4* mögliche Metadaten-Typen zusammengestellt.

Wegwerf-Data-Marts

Bisher wurde über Daten-Inhalte und deren Struktur gesprochen, die flexibel zu handhaben sind. Bei der Virtualisierung von Data Marts und Star-Schemata können noch mehr Aspekte im Weg stehen. Eine Art Lackmus-Test für eine mögliche Virtualisierung existierender Data Marts liefern die Fragen „Was wäre, wenn wir einen Data Mart (die Daten eines Star-

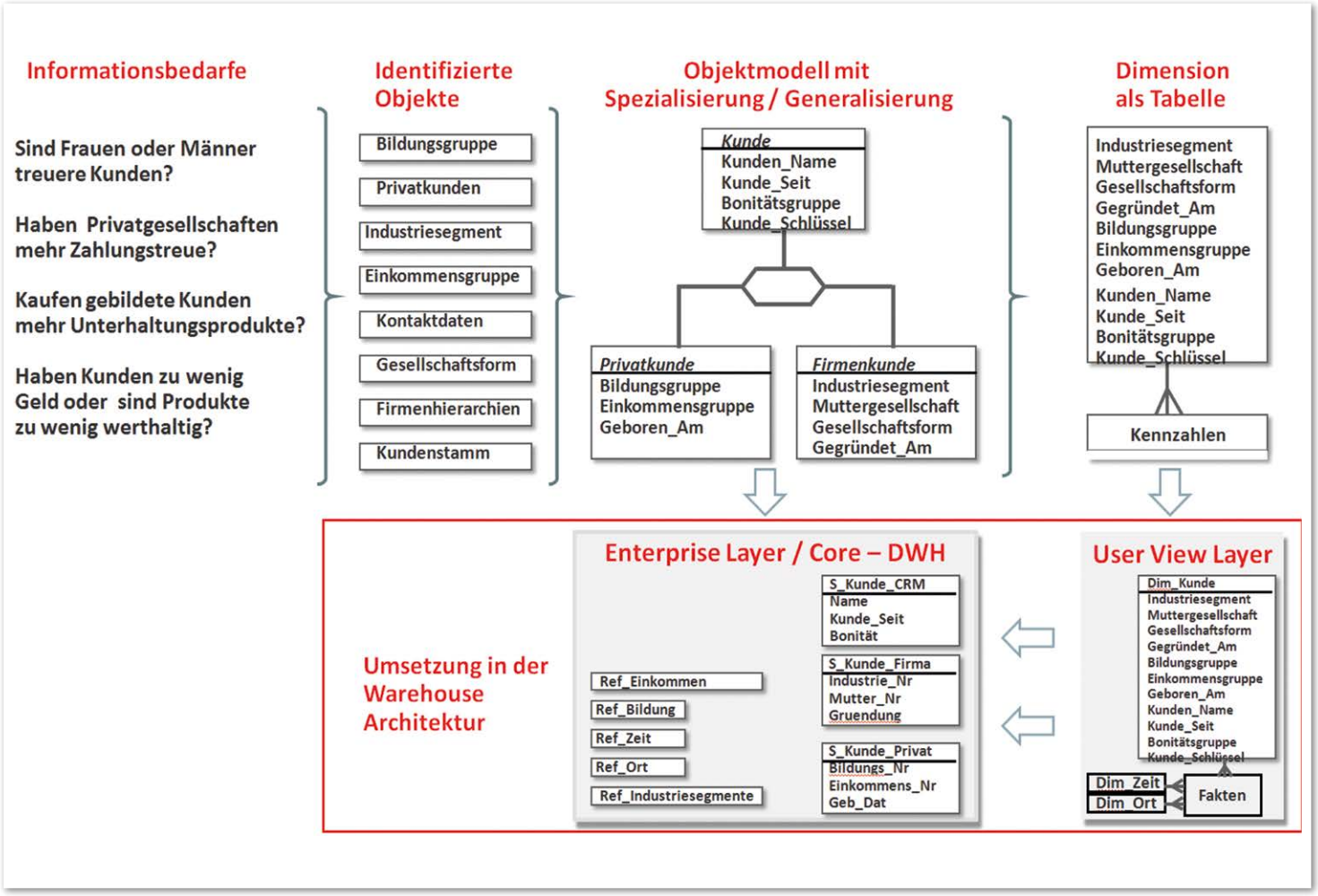


Abbildung 3: Schritte bei der Modellierung in einem Data Warehouse

Schemas) wegwerfen würden?“, „Würden Informationen verloren gehen?“ und „Können wir alle Inhalte und Kennzahlen wieder herstellen?“.

Wer die letzte Frage mit „ja“ beantwortet, hat einen solchen Wegwerf-Data-Mart. So provokativ und abfällig der Begriff auch klingt, Wegwerf-Data-Marts sind bezogen auf Flexibilität und Schnelligkeit, mit der sich geänderte Analyse-Modelle für die Anwender neu aufbauen lassen, das Beste, was passieren kann.

In vielen bestehenden Data-Warehouse-Systemen sind Data Marts für die Ewigkeit gebaut; zu viele Informationen und Spezialmodellierungen kommen darin vor. Zudem verstehen Techniker das zentrale Data Warehouse oft nur als dominierte

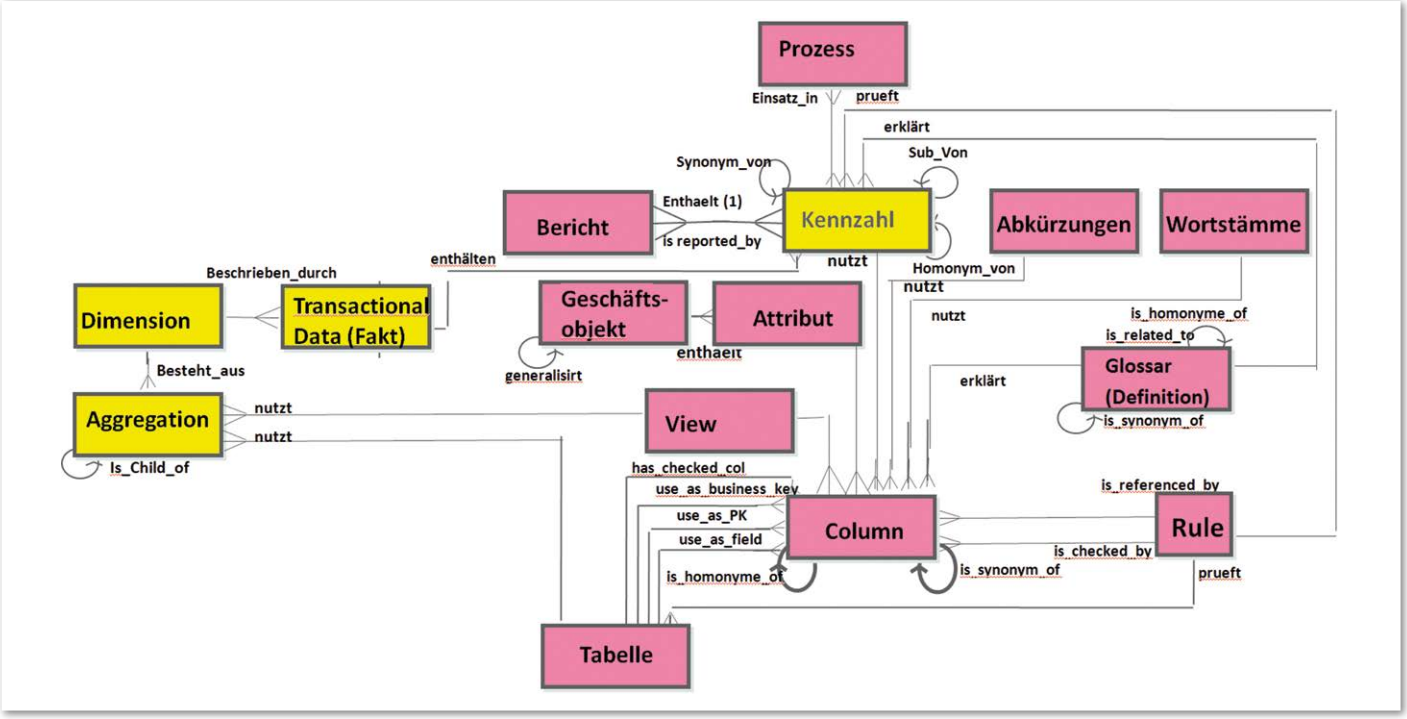


Abbildung 4: Die dargestellten Metadaten-Typen lassen erahnen, welche erweiterten Möglichkeiten die Lösung im Sinne einer Data Governance bereithält

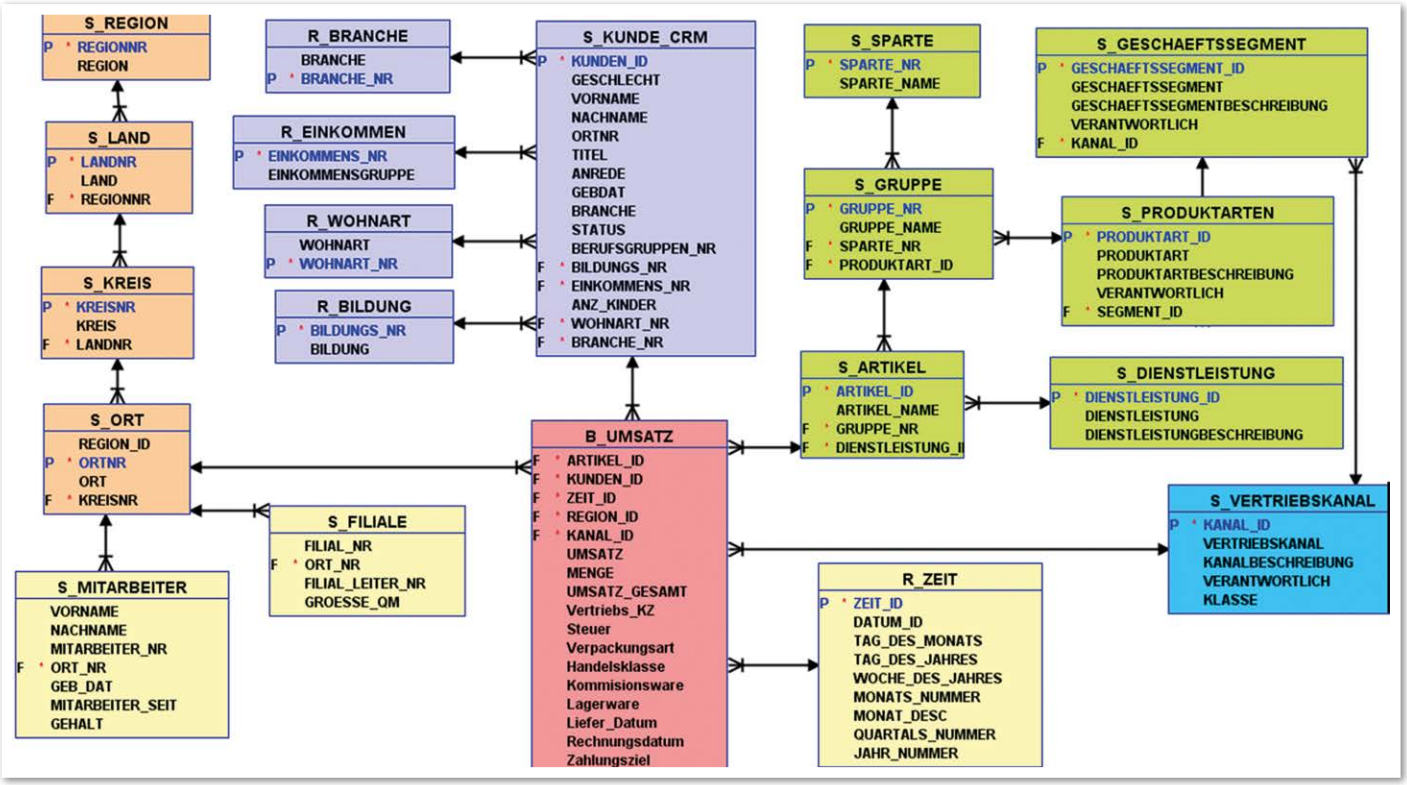


Abbildung 5: Datenmodell aus der zentralen Data-Warehouse-Schicht (ER-Modeler im SQL Developer)

Datenablage, die eher den Vorsystemen gleicht. Die sogenannte „Fachlichkeit", also das Berechnen von Kennzahlen oder spezielle Berichtsvorbereitungen, entsteht erst in den Data Marts. Das Ergebnis dieses Verständnisses sind in Beton gegossene Data-Mart-Komplexe, die das Image des unflexiblen Data Warehouse in den Diskussionen prägen.

Zurück zu den Wegwerf-Data Marts: Data-Warehouse-Architekturen, in denen diese möglich sind, benötigen folgende Eigenschaften:

- Historisierungs-Attribute und Stammdaten-Versionierungen (im Sinne von SCD) sollten bereits in der zentralen Warehouse-Schicht stattfinden
- Die Bildung der künstlichen Data-Warehouse-Schlüssel erfolgt beim Übergang vom Integration-Layer („Stage") in die Warehouse-Schicht
- Referenzdaten sind nur in der Data-Warehouse-Schicht vorhanden
- Endbenutzer haben auch Zugriff auf die zentrale Data-Warehouse-Schicht (ist erst für die Virtualisierung erforderlich)
- Besondere Kennzahlen-Berechnungen müssen als Vorschrift bereits in der Data-Warehouse-Schicht vorhanden sein (etwa als ETL-Strecke oder Materialized-View-Definition oder einfach nur als Metadaten-Objekt)
- Level-Objekte (wie beschrieben) müssen erkennbar sein, um Drill-Pfade in Dimensionen aufbauen zu können
- Die Referenzierung zwischen Stamm- und Referenzdaten auf der einen und den Bewegungsdaten auf der anderen Seite muss bereits in der Data-Warehouse-Schicht angelegt sein; das ist sowieso der Fall, wenn künstliche Schlüssel im Data Warehouse aufgebaut sind
- Orphan-Management bezüglich der in den Bewegungsdaten benutzten Stammdaten muss bereits in der Data-Warehouse-Schicht stattfinden

Tabellen	Anzahl_Sätze	Größe (MB)
F_UMSATZ_BREIT_PAR	564517400	114493.75
S_SPARTE	2	.04
S_GRUPPE	7	.04
S_ARTIKEL	10000	.46
S_REGION	5	.04
S_LAND	11	.04
S_KREIS	257	.04
S_ORT	7202	.28
S_KUNDE_CRM	415584	34.21
S_MITARBEITER	2999	.16
S_FILIALE	199	.04
S_DIENSTLEISTUNG	6	.04
S_GESCHAEFTSSEGMENT	3	.04
S_PRODUKTARTEN	3	.04
R_BILDUNG	10	.04
R_BRANCHE	23	.04
R_EINKOMMEN	7	.04
R_WOHNART	20	.04

Tabelle 1

Virtualisieren

Nachdem die architektonischen Voraussetzungen klar sind, ist der eigentliche Virtualisierungsschritt nur noch eine technische Umsetzung. Man ersetzt die ETL-Strecken, die man zum Aufbau von Star-Schemata benötigt, durch den In-Memory-Column-Store-Ladevorgang. Die Joins, mit denen die modellierten Level-Objekte aus der Warehouse-Schicht zu Dimensionen kombiniert wurden, finden erst zum Ablaufzeitpunkt der Benutzerabfrage statt. Sie sind nach diesem Konzept in Views verpackt und nicht mehr in den ETL-Strecken. Das folgende Szenario *(siehe Tabelle 1)* demonstriert die Machbarkeit und Vorgehensweise sehr deutlich. Es verfügt über eine Data-Warehouse-Schicht (eigenes Datenbank-Schema) mit überwiegend normalisierten Tabellen.

Die Verteilung der Datenmenge auf die einzelnen Tabellen ist dabei typisch für ein Data Warehouse – es gibt einzelne sehr große Tabellen. In diesem Fall ist das „F_UMSATZ_BREIT_PAR" mit 560 Millionen Sätzen und etwa 114 GB Datenvolumen. Die Tabelle umfasst Bewegungsdaten aus fünf Jahren und ist nach Monaten partitioniert (60 Partitionen). Die Masse der Tabellen ist wesentlich kleiner. Die größten sind „Kunden" mit 400.000, „Artikel" mit 10.000, „Mitarbeiter" mit etwa 3.000 und „Orte" mit rund 7.000 Sätzen. Dazu gruppieren sich viele ganz kleine Tabellen *(siehe Abbildung 5)*.

Die Datenverteilung auf die Tabellen begünstigt das Vorhaben der Virtualisierung. Bei wenigen großen und vielen kleinen Tabellen wird das Bloom-Filter-Verfahren häufiger zur Anwendung kommen. Die Chance, eine sehr schnelle Join-Verarbeitung zu erhalten, wird also bei dieser für Data-Warehouse-Systeme typischen Mengenverteilung höher sein als in einem System mit einer homogeneren Mengenverteilung auf die Tabellen.

Das Modell ist nicht völlig normalisiert. Vor allem rund um die Artikel-Tabelle finden sich Tabellen, die eher der oben beschriebenen, modellierten Beschreibungsmöglichkeit und Sichtweise folgen als den Regeln der Normalisierung. Man kann Artikel einfach nach Gruppen und Sparten klassifizieren, aber auch nach Produktarten und Segmenten. Artikel lassen sich letztlich sogar über Vertriebskanäle grup-

	Anzahl Records	GB-Gesamt auf Platte	GB-In-Memory	Compression Factor
Nur genutzte Partitionen In-Memory geladen (12 Partitionen / 1 Jahr)	123047941	11,83	0,91	~13
Alle Partitionen In-Memory geladen (60 Partitionen / 5 Jahre)	564985121	117,65	9,4 GB	12,52

Tabelle 2: Die Mengenverteilung der Testdaten. In der ersten Zeile sind nur diejenigen Partitionen geladen, die auch tatsächlich genutzt werden, in der zweiten Zeile kommen alle Partitionen in dem Column-Store.

pieren. Entscheidend sind die Geschäftsregeln, die das Modell abbilden muss, mit all den Generalisierungs-/Spezialisierungsarbeiten im Rahmen der Informationsbedarfs-Analyse. Die Tabellen entsprechen den Hierarchie-Level-Objekten, die potenziell in Dimensionen nutzbar sind.

Die meisten Tabellen sind durch Foreign-Key-Beziehungen verknüpft. Sie sind mit beliebigen Datenmodellierungswerkzeugen grafisch darstellbar, wie in diesem Beispiel mit dem ER-Modeler des SQL Developer. Praktisch ist die Möglichkeit, ständig zwischen Datenbank- und Modell-Sicht zu wechseln, um Unterschiede herauszufinden, die man dann beseitigen kann.

Das Modell entspricht schon jetzt potenziellen Drill-Pfaden. Die Tabellen oder deren Teile kann man bequem über Views anfassen. Für das Szenario wurde dazu in einem zweiten Datenbank-Schema ein Star-Schema als Views nachempfunden. In diesem liegen nur eine Zeit-Tabelle und die Views als eine Art „Pointer" auf das zentrale Data-Warehouse-Schema. Über die Views kombiniert man jetzt (wahlfrei) die Hierarchie-Level-Objekte zu virtuell dargestellten Dimensionen.

Der Dimensions-Schlüssel ist der Schlüssel des untersten Levels. Die Joins zwischen den Hierarchie-Level-Objekten werden zum Aufrufzeitpunkt erst gebildet. Auch die Fakten-Tabelle kann über eine View-Definition repräsentiert werden. Das ist allerdings nur nötig, wenn der Name einer Fakten-Tabelle im Star-Schema ein anderer ist als der Name der Bewegungsdaten-Tabelle in der Data-Warehouse-Schicht *(siehe Abbildung 6)*.

Jetzt kommt In-Memory ins Spiel. Das, was aufwändig klingt – potenziell viele Joins zum Aufrufzeitpunkt aufzulösen –, findet in diesem Szenario mit In-Memory statt. Alle für Auswertezwecke gedachten Objekte der Data-Warehouse-Schicht sind mit „In-Memory" markiert („ALTER TABLE tablename INMEMORY;") und befinden sich im In-Memory-Column-Store, auch die große Bewegungsdaten-Tabelle mit 560 Millionen Zeilen und 114 GB Datenvolumen.

Für die großen Bewegungsdaten-Tabellen werden zwei Fälle durchgespielt. Im ersten Fall lädt das Szenario nur die Jahre 2015 und 2014 in den Column-Store, also die Daten, die am häufigsten genutzt werden. Das spart in dem Szenario etwa 90 Prozent des Column-Store-Hauptspeichers und ist an sich bereits ein Vorteil. Wie sich in dem Szenario herausstellte, verkürzt sich auch die Zeit, die man benötigt, um die Daten von der Platte in den Column-Store zu laden. Das muss man in der Praxis sicherlich mitberücksichtigen.

In einem zweiten Fall lädt man die Tabelle komplett. Der Platz ist da und man kann einen Versuch starten, um das System auszureizen. In der Praxis ist das allerdings nicht nötig *(siehe Tabelle 2)*.

Die Hardware für das Szenario

Bezüglich Hardware denkt man jetzt sicher an einen mittleren bis großen In-Memory-Rechner. Ein Ziel dieses Szenarios ist es auch, die Folgen des In-Memory-Einsatzes hinsichtlich der Hardware herauszustellen und mit manchen Hardware-Vorurteilen aufzuräumen. Als die ersten Beta-Versionen der neuen In-Memory-Datenbank-Option verfügbar waren, hat man gleich nach sehr großen Demo-Maschinen für dieses Feature Ausschau gehalten, bis dann die ersten Demos auf handelsüblichen Laptops unter Windows liefen.

Für dieses Szenario wurde bewusst ein einfacher Desktop-Rechner mit einer Intel 4-Core-CPU, 32 GB Hauptspeicher und vier direkt eingebauten 500 GB Festplatten konfiguriert. Die Anschaffungskosten lagen bei etwa 800 Euro (November 2014). Es soll hier jedoch nicht dafür plädiert wer-

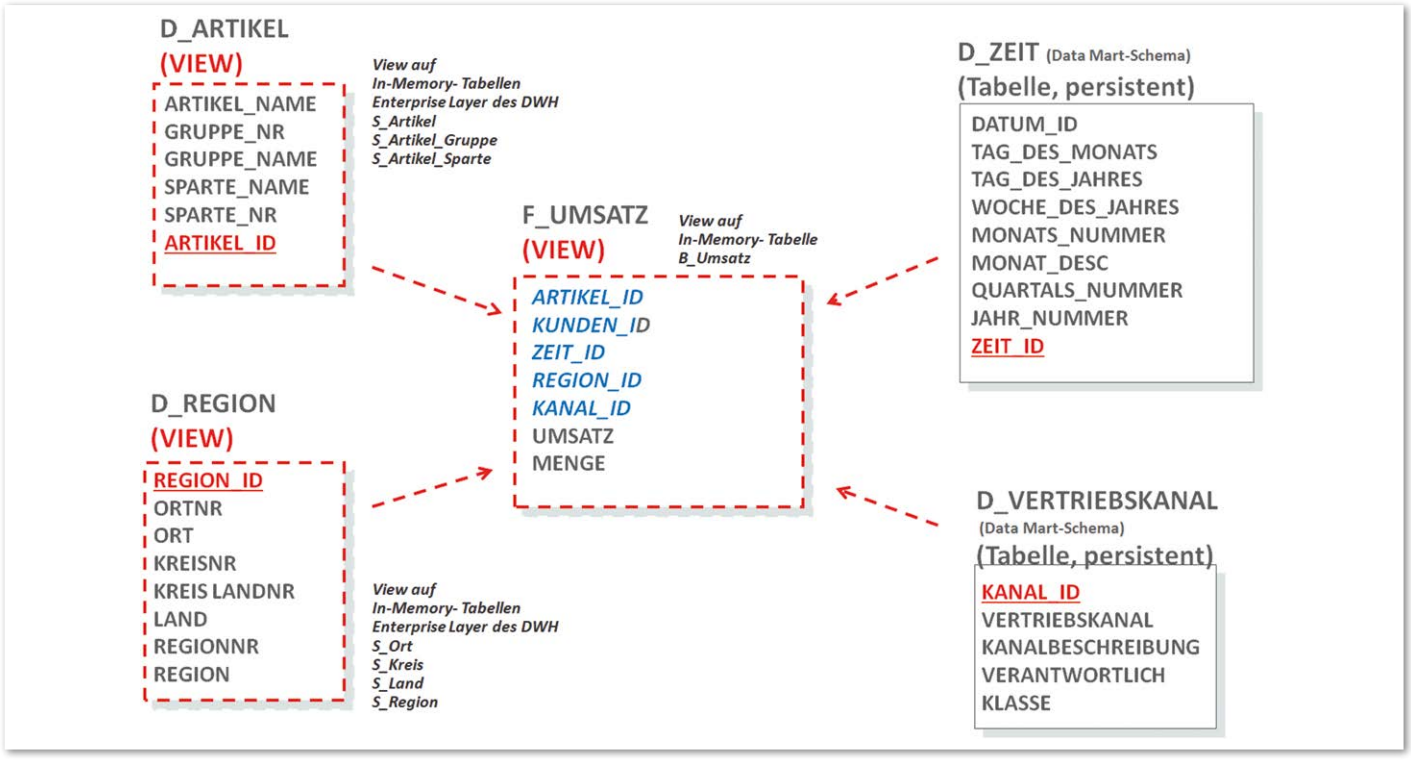


Abbildung 6: Teil-virtuelles Star-Schema, bei dem Dimensionen und Fakten („D_ARTIKEL", „D_REGIONEN" und „F_UMSATZ") als Views auf das zentrale Data-Warehouse-Schema realisiert sind

den, solche günstigen Rechner für sein Data Warehouse zu wählen. Hier wurde ein Extrem durchgespielt. In der Realität sollte ein solches System wesentlich mehr Hauspeicher aufnehmen können und das Plattensystem aus wesentlich mehr physikalischen Platten bestehen, mit denen beispielsweise ASM betreibbar ist. Die Anzahl der CPU-Cores sollte erheblich höher sein, um höhere Benutzerzahlen und Parallelisierung zu erlauben.

Die meisten Warehouse-Umgebungen leiden unter der nicht ausreichenden Datenmenge, die das Plattensystem der Warehouse-Datenbank bereitstellt. Oft eine Folge von nicht Data-Warehouse-adäquatem Betrieb im Zusammenspiel mit einem zentralen SAN. Auch wenn Server über ausreichend Memory und CPU verfügen, bleibt das oft ohne Nutzen für die Anwender, wenn das Plattensystem bremst. Hier wird In-Memory helfen.

Das Demo-System liefert ungefähr 240 MB Daten pro Sekunde. Für die Datenmenge des Szenarios und vor allem, wenn die Anzahl paralleler Prozesse steigt, ist das viel zu wenig. Das Szenario wird aber genau diese Restriktion mithilfe von In-Memory aufheben und damit zeigen, mit welch geringem Aufwand bereits gute Effekte durch In-Memory erzielbar sind. Es sind auch die kleineren Systeme, die von In-Memory massiv profitieren.

```
SQL> SELECT * FROM
  (SELECT a.Artikel_Name    Artikel,
          r.Land     Bundesland,
          z.Jahr_nummer  Jahr,
          sum(U.umsatz)      Wert,
          sum(U.Menge)   Menge,
          round(sum(U.umsatz) / sum(U.Menge),2) Ums_pro_Art,
     RANK() OVER (ORDER BY sum(U.umsatz) DESC ) Rangfolge
  from   F_umsatz_breit_Par U,
         D_Artikel A ,
         D_Zeit z,
         d_region r
  WHERE
         U.artikel_id =    a.artikel_id   and
         U.REGION_ID =     R.REGION_ID    AND
         U.zeit_id      = z.zeit_id       AND
         U.zeit_id  > to_date(,01.11.2013‘,‘DD.MM.YYYY‘) AND
         U.zeit_id  < to_date(,01.12.2013‘,‘DD.MM.YYYY‘)
  group by
         a.artikel_name,r.Land,z.Jahr_nummer)
  WHERE rownum < 11;

ARTIKEL                          BUNDESLAND
-------------------------------- ----------------------
Kupferrohr 18mm                  Bayern
SchraubenschluesselSet           Schleswig Holstein
Wanne 180                        Bayern
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Wanne Kurz                       Schleswig Holstein
10 rows selected.
Elapsed: 00:00:00.65
```

Listing 1

Die Hauptspeicher-Belegungssituation in dem Rechner sieht wie folgt aus: Dem Datenbank-System sind 28 GB Hauptspeicher zugewiesen. Davon entfallen unter anderem 13 GB auf den Buffer-Pool (zeilenorientierte Blöcke im Hauptspeicher) und 14 GB auf den Column-Store.

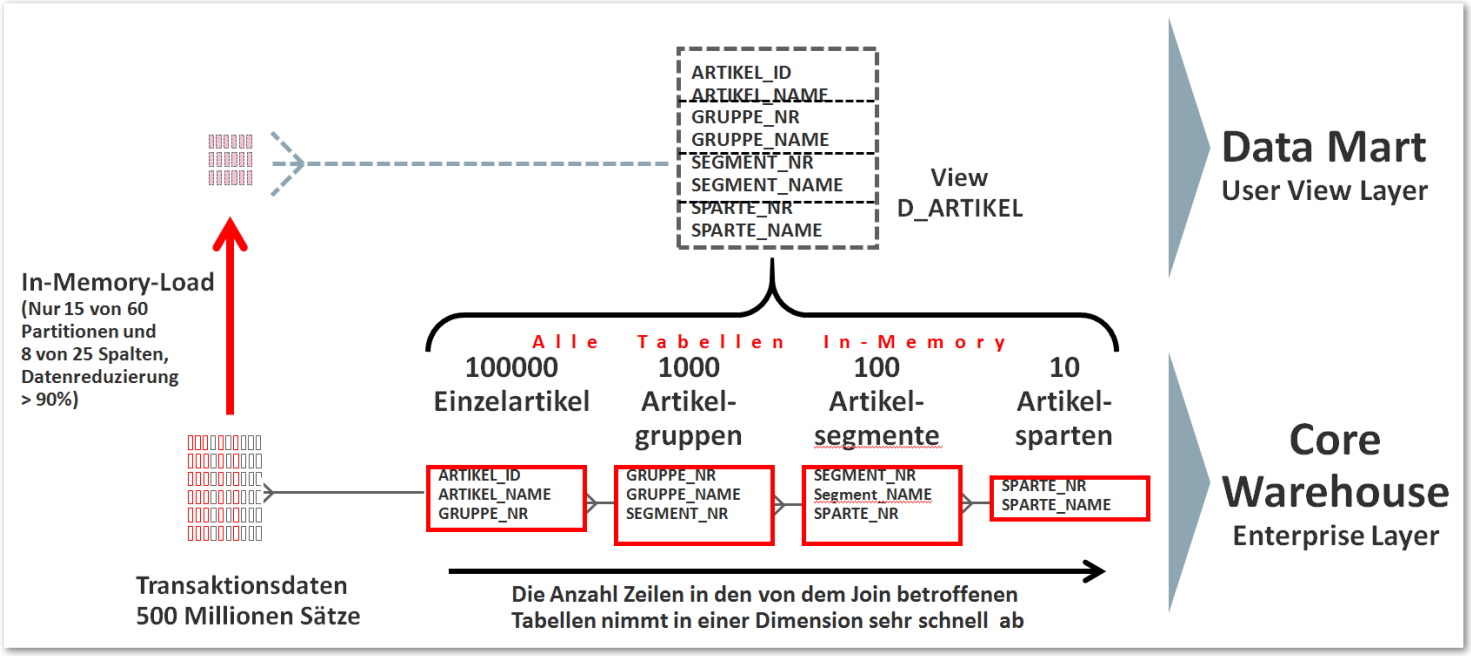


Abbildung 7: Joins über die gedachten Hierarchie-Level einer virtualisierten Dimension werden kleiner mit zunehmender Aggregationshöhe, was dem Virtualisierungskonzept hilft

```
-------------------------------------------------------------------------------------------------------------------
 Id   |             Operation                    | Name              |  Rows | Bytes |TempSpc| Cost (%CPU)|
-------------------------------------------------------------------------------------------------------------------
    0 |             SELECT STATEMENT             |                   |    10 |  1190 |       | 2554   (1)|
~~~~~~~~~~~

   14 |             PX RECEIVE                   |                   |     7 |    21 |       |     2   (0)|
   15 |             PX SEND BROADCAST            | :TQ10005          |     7 |    21 |       |     2   (0)|
   16 |             PX BLOCK ITERATOR            |                   |     7 |    21 |       |     2   (0)|
   17 |             TABLE ACCESS INMEMORY FULL   | S_GRUPPE          |     7 |    21 |       |     2   (0)|
*  18 |             HASH JOIN                    |                   | 4109K |   423M|       |   106   (8)|
   19 |             PX RECEIVE                   |                   | 10000 |   361K|       |     2   (0)|
   20 |             PX SEND BROADCAST            | :TQ10006          | 10000 |   361K|       |     2   (0)|
   21 |             PX BLOCK ITERATOR            |                   | 10000 |   361K|       |     2   (0)|
   22 |             TABLE ACCESS INMEMORY FULL   | S_ARTIKEL         | 10000 |   361K|       |     2   (0)|
*  23 |             HASH JOIN                    |                   | 4109K |   278M|       |   103   (7)|
   24 |             PART JOIN FILTER CREATE      | :BF0000           |    30 |   360 |       |     2   (0)|
*  25 |             TABLE ACCESS FULL            | D_ZEIT            |    30 |   360 |       |     2   (0)|
*  26 |             HASH JOIN                    |                   | 4246K |   238M|       |   100   (6)|
   27 |             PX RECEIVE                   |                   |  7202 |   253K|       |     8   (0)|
~~~~~~~~~~~~~~~~~~~~~

   39 |             STATISTICS COLLECTOR         |                   |       |       |       |            |
   40 |             PX BLOCK ITERATOR            |                   |     5 |    15 |       |     2   (0)|
   41 |             TABLE ACCESS INMEMORY FULL   | S_REGION          |     5 |    15 |       |     2   (0)|
   42 |             PX RECEIVE                   |                   |    11 |   198 |       |     2   (0)|
   43 |             PX SEND HYBRID HASH          | :TQ10001          |    11 |   198 |       |     2   (0)|
   44 |             PX BLOCK ITERATOR            |                   |    11 |   198 |       |     2   (0)|
   45 |             TABLE ACCESS INMEMORY FULL   | S_LAND            |    11 |   198 |       |     2   (0)|
   46 |             PX RECEIVE                   |                   |   257 |  1799 |       |     2   (0)|
   47 |             PX SEND HYBRID HASH          | :TQ10003          |   257 |  1799 |       |     2   (0)|
   48 |             PX BLOCK ITERATOR            |                   |   257 |  1799 |       |     2   (0)|
   49 |             TABLE ACCESS INMEMORY FULL   | S_KREIS           |   257 |  1799 |       |     2   (0)|
   50 |             PX BLOCK ITERATOR            |                   |  7202 | 57616 |       |     2   (0)|
   51 |             TABLE ACCESS INMEMORY FULL   | S_ORT             |  7202 | 57616 |       |     2   (0)|
   52 |             PX BLOCK ITERATOR            |                   | 4246K |   93M |       |    92   (7)|
*  53 |             TABLE ACCESS INMEMORY FULL   | F_UMSATZ_BREIT_PAR| 4246K |   93M |       |    92   (7)|
-------------------------------------------------------------------------------------------------------------------
```

Listing 2

Die Testabfragen

Nachfolgend ist eine Beispiel-Abfrage mittlerer Komplexität mit mehreren Joins, Funktionen, Aggregationen und einer analytischen Funktion formuliert. Sie wird aus dem separaten Data-Mart-Schema heraus aufgerufen. Die Tabellen „D_ARTIKEL, D_REGION“ und „F_UMSATZ_BREIT_PAR“ sind Views in dem Schema *(siehe Listing 1)*.

Wie man erkennt, läuft das Ganze in 0,65 Sekunden durch. Hinweis: Werte unter einer Sekunde sind in diesem System nicht mehr korrekt messbar, da die unterschiedlichen Faktoren zusätzlich beeinflusst sind. Der Plan (verkürzt dargestellt) zeigt, dass alle zugehörigen Stammdaten-Tabellen, aus denen sich die Views bedienen, angezogen werden *(siehe Listing 2)*.

Als Vergleich dazu wird die gleiche Abfrage auf echte, nicht virtualisierte Dimensionen abgesetzt. Aber auch diese Abfrage liegt in einem Bereich von unter einer Sekunde (00.78). Wie oben bereits erwähnt, sind echte Vergleiche bei Abfragezeiten von unter einer Sekunde kaum machbar.

Wie realistisch ist dieses Szenario? In weiteren Tests ist die Zahl der Artikel von 10.000 auf beispielsweise 100.000 (Anzahl Sätze) erhöht worden. Damit gehen die Antwortzeiten in einen Bereich zwischen einer und zwei Sekunden. Eine Veränderung wird also spürbar, wenn die beteiligten Join-Tabellen wachsen. Man „joint“ immerhin eine 500-Millionen-Tabelle nach 100.000 Sätzen. Zusätzliche Datenmengen kann man bequem durch Parallelisierung ausgleichen.

Dieser Umstand hat allerdings kaum Auswirkungen auf das Virtualisierungskonzept. Denn meist werden direkt benachbarte Tabellen, also das unterste (granulare) Level einer Dimension, und die Fakten-Tabelle von dem aufwändigeren Join betroffen sein. Alle weiteren Joins einer simulierten Dimension ziehen wesentlich kleinere Tabellen heran, denn je weiter man bei einer Dimension in Richtung höherer Aggregation geht, desto weniger Sätze sind in einem Join betroffen, wenn man Level-Objekte erst zum Abfragezeitpunkt miteinander verknüpft *(siehe Abbildung 7)*.

Gravierender schlägt die Parallelisierung durch. Die große Bewegungsdaten-Tabelle „F_UMSATZ_BREIT_PAR“ lässt sich auf dem Testsystem optimal mit „parallel 8“ lesen.

Die Maschine besitzt vier echte Cores; pro Core lassen sich zwei Verarbeitungsprozesse gut betreiben, also in Summe „8“. Eine noch stärkere Parallelisierung macht sich nicht mehr bemerkbar, weil die Antwortzeiten bereits weit unter einer Sekunde liegen.

Nimmt man die Parallelisierung zum Beispiel zurück auf „4“, so werden die Zeiten schon etwas länger (von ungefähr 0,4 - 0,5 sec. auf 0,7 - 0,8 sec.). Ganz ohne Parallelisierung liegt der Wert bei ungefähr zwei Sekunden (alle Werte sind für den Testlauf mit 10000 Artikeln im Join mit der 500 Millionen-Tabelle). Diese Werte erreicht man in beiden Fällen: sowohl im virtualisierten Star-Schema des Data Mart als auch im persistierten Tabellenwerk der Data-Warehouse-Schicht. In der Praxis empfehlen sich Rechner mit vielen Cores, damit die Parallelisierung gut ausgereizt werden kann (siehe *Abbildung 8*, die Sekundenwerte können jedoch nicht als absolute Werte genommen werden, da es noch eine Reihe von Dritt-Faktoren gibt, die das Ergebnis mit beeinflussen).

Man hätte bei dieser Vorgehensweise auch das zehnfache Datenvolumen in dem Szenario verarbeiten können. Im Fall 1 hat man nur die letzten 14 Partitionen der großen Bewegungsdaten-Tabelle in den Column-Store geladen, was den 14 GB großen In-Memory-Column-Store bei einer Kompressionsrate von 13 - 14 nur zu 7 Prozent belegt. Man hätte also noch zehn weitere 110-GB-Tabellen in das Szenario aufnehmen können. Entsprechende Abfragen auf diese weiteren zehn Tabellen wären auch unter einer Sekunde Antwortzeit geblieben. Warum sollte es auch länger dauern, wenn Mechanik und Aufwand gleich bleiben. In-Memory skaliert durch Adress-Positionierung besser als das Plattensystem, in dem größere physikalische Strecken zurückzulegen sind, wenn die Datenmenge steigt.

Mit diesem Gedankenspiel erahnt man, welches Potenzial in dieser Technologie steckt. In dem Mini-Szenario bewegen wir Mengen, die im Jahr 2000 unter dem Stichwort „Terabyte Data Warehouse“ noch einen Ruf als dicke Brocken in der IT-Abteilung hatten.

Das Virtualisieren des Data Mart führt auch zu einem erheblichen Plattenplatz-Spareffekt. Bei der Oracle-In-Memory-Option kann man normalerweise nicht von „Sparen beim Plattenplatz“ sprechen,

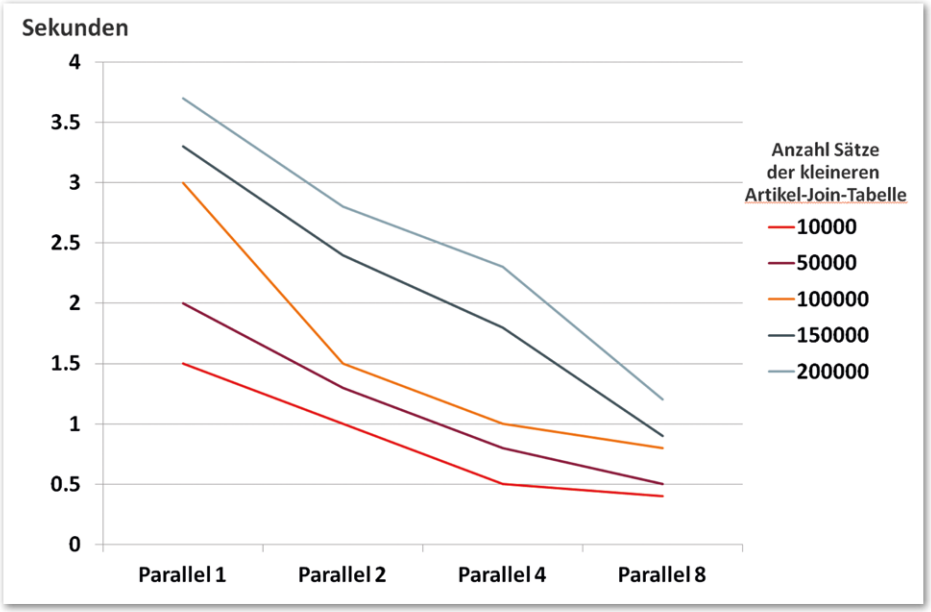


Abbildung 8:. Beispiel-Abfrage mit Join zwischen einer 550-Millionen-Sätze-Fakten-Tabelle und wachsender Artikel-Tabelle

weil die Daten, auch wenn sie im Hauptspeicher vorgehalten werden, im Plattensystem persistiert sind. Der Spareffekt bei dieser Virtualisierungslösung entsteht durch den Wegfall der Redundanz, die im Architektur-Ansatz von Inmon zwischen der Kern-Warehouse-Schicht und den Data Marts besteht. Bei konsequenter Anwendung spart man den Plattenplatz für die Data Marts. Hier existieren oft die großen Tabellen und meistens viele Indizes. Geschätzt sind das etwa zwanzig bis fünfzig Prozent des gesamten Plattenplatzes für das Warehouse. Das sind bei einem 10-TB-Warehouse schon 2 bis 5 TB. Allein damit lassen sich bereits die Anschaffungskosten für die In-Memory-Option amortisieren.

Fazit

Die Virtualisierung von multidimensionalen Strukturen in einem Data Mart wie dem Star-Schema ist technisch gut machbar. Die Vorteile liegen in der verkürzten Data-Warehouse-Architektur, während alle Funktionen der jeweiligen Schichten erhalten bleiben, also vor allem die auf Geschäftsobjekte fokussierte, multidimensionale Sicht für die Anwender. Es gibt vier wichtige Vorteile:

- ETL-Aufwand für die letzte Schicht fällt weg.
- Die letzte Schicht (Data Marts) kann wesentlich schneller umgebaut und variiert werden. Auf Änderungswünsche der Anwender kann man schneller reagieren.
- Die Vorgehensweise zwingt die Modellierer des Systems, die benötigten Geschäftsobjekte bereits in der Data-Warehouse-Schicht entsprechend dem tatsächlichen Anwenderbedarf zu modellieren, was zu mehr Ordnung und Übersichtlichkeit im Gesamtsystem führt. Diese Vorgehensweise widerspricht der oft vorgefundenen Methode, wonach erst die Quellsysteme „1:1“ gelesen und deren Daten dann stur in die dritte Normalform überführt werden.
- Der Wegfall einiger zum Teil sehr großer Tabellen und Indizes im Data-Mart-Bereich spart spürbar Plattenplatz.

In-Memory ist eben mehr als einfach nur Performance-Gewinn, es ist auch ein Hilfsmittel zur Architektur-Optimierung.

Alfred Schlaucher
alfred.schlaucher@oracle.com

In-Memory-Computing mit der Oracle-Datenbank 11g R2

Matthias Weiss, ORACLE Deutschland B.V. & Co. KG

In-Memory-Verarbeitung ist in aller Munde. Unternehmen erhoffen sich Verarbeitungs- und Prozess-Beschleunigung beziehungsweise sehen dies als Voraussetzung für Industrie 4.0 und/oder Internet of Things.

Dieser Artikel zeigt In-Memory-Funktionen für die Oracle-Datenbank 11*g* R2, die aus dem regulären Support läuft, aber bei vielen Kunden noch im Einsatz ist. Sie sind schnell und einfach umzusetzen, um Performance-Gewinne zu erzielen beziehungsweise neue Geschäftsfelder zu erschließen. Außerdem werden eventuell notwendige und sinnvolle Hardware-Erweiterungen und -Ergänzungen kurz erläutert. Abschließend kommen die drei großen Erweiterungen für In-Memory-Verarbeitung der Oracle-Datenbank 12*c* (ab 12.1.0.2) kurz zur Sprache, um einen Planungshorizont zu bieten. Nachfolgend die zur Verfügung stehenden In-Memory-Funktionen:

- DB-Cache
- Row-Caches
- Statement-Caches
- Pinnen von ganzen Tabellen
- Oracle-Text-Index im Memory
- Query-Result-Cache
- In-Memory Parallel Query (RAC)
- Cache Fusion (RAC)
- Cache Hierarchie
- Smart-Flash-Cache

Um die Optimierungsmöglichkeiten für In-Memory-Verarbeitung besser einordnen zu können, muss man die Hauptspeicher-Struktur der Oracle-Datenbank verstehen. Die rot markierten Bereiche in *Abbildung 1* sind für diesen Beitrag essenziell wichtig.

Grundsätzlich sollte die Hauptspeicher-Ausstattung ausreichend sein. Die Varianz ist allerdings von Kunde zu Kunde beziehungsweise von Anwendung zu Anwendung so groß, dass es keine eindeutige Empfehlung geben kann. Allerdings sind viele Datenbank-Server, insbesondere im Mittelstand, mit einem zu kleinen Hauptspeicher ausgestattet. Ein Ausbau mit 32, 64 oder 128 GB reicht in den meisten Fällen nicht aus; sinnvolle In-Memory-Verarbeitung beginnt bei mindestens 256 GB. Bestimmte Rechner-Architekturen lassen jedoch bei Zwei-Socket-Servern nur einen maximalen Ausbau von 768 GB zu, was nicht immer ausreicht, sodass hier nach Alternativen geschaut werden muss (siehe Kapitel „Smart-Cache-Funktionalität"). Die *Abbildungen 2 und 3* verdeutlichen, wie sich Hardware-Konfigurationen durch die Nutzung von In-Memory-Techniken ändern und was beim Beschaffungsprozess neuer Server für In-Memory-Computing berücksichtigt werden muss.

Bei einer SAN-I/O-Leistung von 700 MB/s und einer durchschnittlichen Core-Leistung von 200 MB/s sind acht Cores für die optimale Verarbeitungsgeschwindigkeit notwendig, unter der Annahme, dass rund die Hälfte der Core-Kapazität für Spitzenbelastung, wie bei vielen Kunden gewünscht, als Reserve vorgehalten werden.

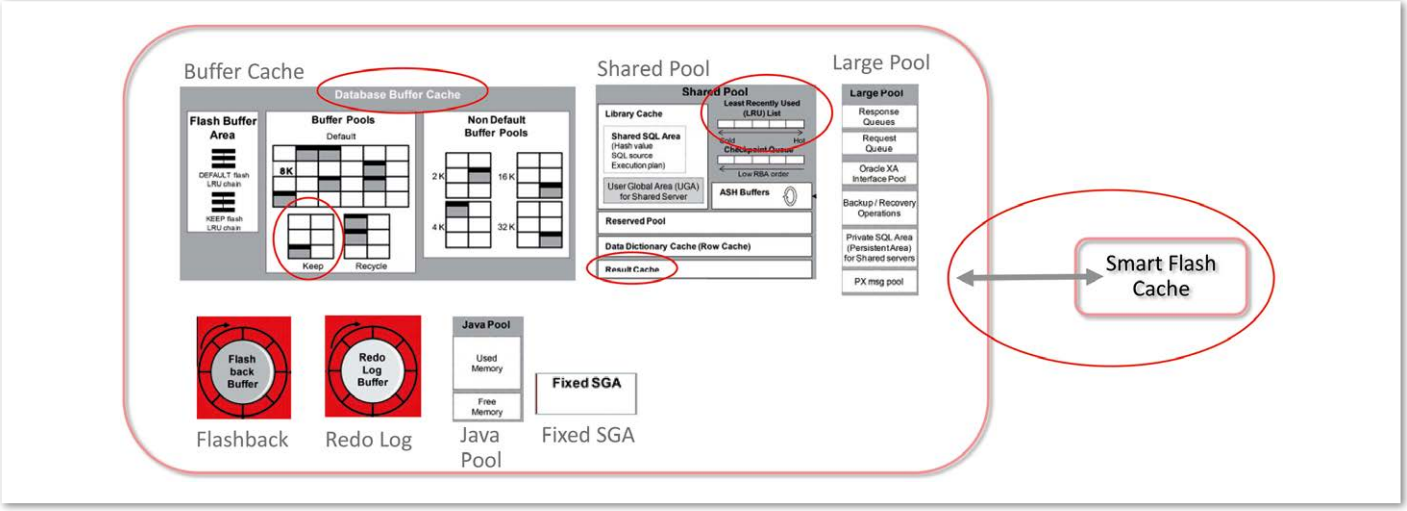


Abbildung 1: Oracle-Memory-Struktur (SGA)

Bei gleicher SAN-I/O-Leistung sind beim In-Memory-Computing deutlich mehr Cores für eine optimale Verarbeitungsgeschwindigkeit notwendig. Die Daten-Bereitstellung („Memory I/O“) liegt jetzt bei mindestens 5 GB/s. Unter der gleichen Prämisse für die Cores wie bei einer klassischen Verarbeitung (200 MB/s; 50 Prozent Auslastung) liegt die optimale Core-Anzahl jetzt bei 50.

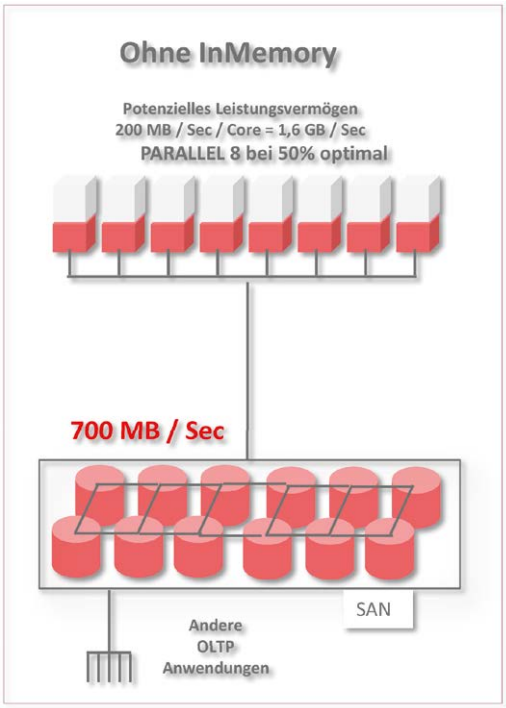


Abbildung 2: Ohne In-Memory

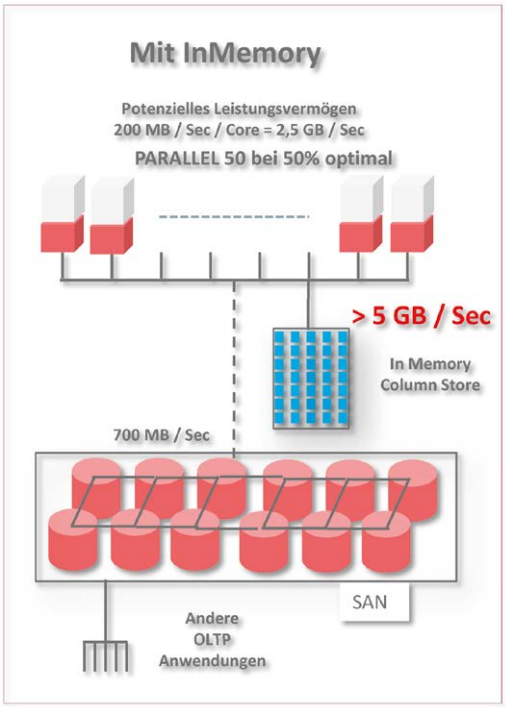


Abbildung 3: Mit In-Memory

Beim In-Memory-Computing verschiebt sich die Last vom Storage-I/O in den CPU-Bereich. Dies erfordert in vielen Fällen eine Änderung der Hardware-Konfiguration. Die Verarbeitungsgeschwindigkeit steigt dramatisch an und es lassen sich dadurch viele Geschäftsvorteile erzielen beziehungsweise bestimmte Geschäftsmodelle bis hin zur Echtzeitverarbeitung realisieren. Je größer der Datenbank-Cache, desto mehr Daten können im Hauptspeicher für die schnellstmögliche Verarbeitung in der Oracle-Datenbank liegen und desto individueller lassen sich die einzelnen Bereiche an die besonderen Anforderungen der Anwendungen und Benutzerbedürfnisse anpassen.

Daten im Buffer-Cache behalten

Die Idee beim Pinnen von Objekten ist, dass Objekte im Pool nicht verdrängt werden. Einem Pool werden nur so viele Objekte zugewiesen, bis er voll ist. Alle Segment-Typen wie LOB, INDEX etc. werden unterstützt, aber auch nicht relationale Daten wie JSON, Spatial oder XML. Dies bedeutet, dass die entsprechende Klausel „ALTER TABLE <table_name> CACHE;“ für das Pinnen von Objekten nicht den Cache benutzt, sondern die Blöcke auf den MRU-Punkt („most recently used“) der LRU-Liste („least recently used“) legt. Diese Objekte müssen aber erstmal von der Platte in den Hauptspeicher geladen werden, damit sie dann auch dort verbleiben. Dazu ist etwa Vorarbeit nötig. Der Pool ist um die entsprechende Größe des Objekts zu erweitern. Als Daumenwert gilt, dass der Pool um das eineinhalb- bis zweifache der Größe des Objekts zu vergrößern ist, also für eine Zwei-GB-Tabelle auf drei bis vier GB.

Der Buffer-Cache wird in zwei Bereiche („KEEP“ und „RECYCLE POOLS“) aufgeteilt, die vom Algorithmus her gleich funktionieren (Init-Parameter „DB_KEEP_CACHE_SIZE“ und „DB_RECYCLE_CACHE_SIZE“). Einem dieser Bereiche sind die Objekte (Tabellen, Partitionen, etc.) zugeordnet, die im Cache gehalten werden sollen. In den Keep-Pool gehören Daten, auf die häufig zugegriffen wird.

Die Bezeichnung der Pools als „Keep“, „Recycle“ und „Default“ könnte zu einem einfacheren Verständnis auch einfach „Pool1“, „Pool2“ und „Pool3“ lauten. Da Daten jedoch erst in den Cache kommen, nachdem sie erstmalig abgefragt wurden, kann bei Bedarf eine Prozedur ausgeführt werden, um eine Query zu erzwingen. Oracle legt keine Tabellen-Buffer in den Pool, bis eine Query auf diese Tabelle ausgeführt wird.

Die Vorgehensweise: Zuerst einmal ist die Pool-Größe zu definieren, hier beispielhaft der Keep-Pool über „DB_KEEP_CACHE_SIZE“; anschließend sind die Objekte per Storage-Klausel dem KEEP-Pool zuzuweisen *(siehe Listing 1)*.

Letztendlich ist je nach Bedarf ein „Pre-Load“ der Objekte in den KEEP-Pool (etwa Trigger bei „On startup database“) zu implementieren. Dazu wird ein Objekt-Scan über die Tabelle, den Index oder das Lob-Segment ausgelöst. Dies ist prinzipiell eine „SELECT“-Anweisung *(siehe Listing 2)*.

Es ist zu beachten, dass der Buffer-Cache optimiert werden muss. Dabei ist der Parameter „_small_table_threshhold“ auf größer oder gleich „benötigte Blöcke“ zu setzen. Weitere Informationen dazu am Ende des Artikels.

Einmal rechnen, mehrfach nutzen

In der Oracle-Datenbank gibt es einen Cache, der speziell für Ergebnisse aus SQL-Abfragen

```
ALTER TABLE ... STORAGE (buffer_pool keep)
ALTER INDEX ... STORAGE (buffer_pool keep)
ALTER TABLE ... MODIFY LOB (lobcol) (STORAGE (buffer_pool keep))
```

Listing 1

```
SELECT /*+ FULL(TAB) */ SUM(numeric_column), min(txt_column) FROM
tabelle TAB;
SELECT /*+ FULL(TAB) */ dbms_lob.getlength(lob_column) FROM tabelle
TAB;
```

Listing 2

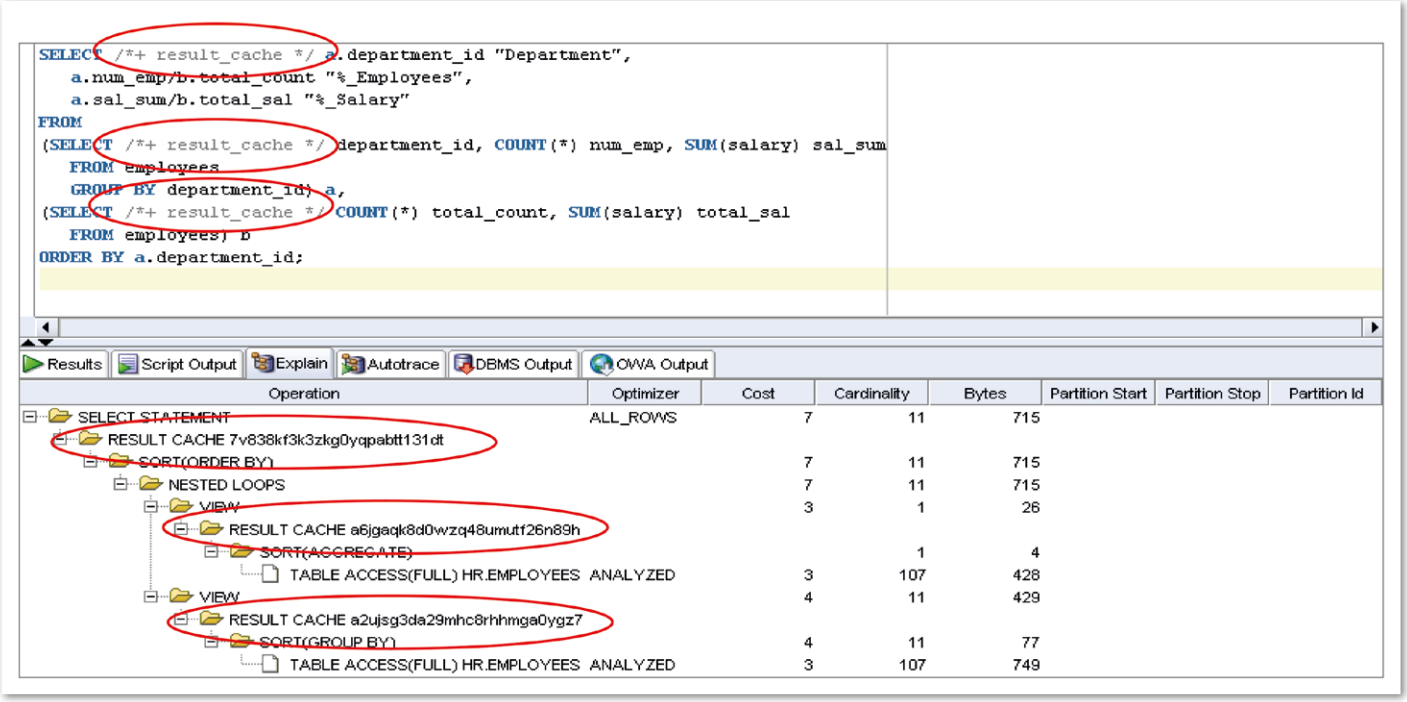


Abbildung 4: Result Cache – Query Hints

oder PL/SQL-Funktionen in der Shared Global Area (SGA) reserviert ist, der Server Result Cache. Dort sind Ergebnisse einer Datenbank-Abfrage oder ein „Query-Block" zur wiederholten Nutzung gespeichert. Der SQL Query Result Cache enthält die Ergebnisse von SQL-Queries und der PL/SQL Function Result Cache die Werte, die von einer PL/SQL Funktion stammen. Die Datenbank verwaltet den Cache völlig selbstständig und stellt dabei sicher, dass niemals veraltete Ergebnisse ausgegeben werden.

Der Oracle Call Interface (OCI) Client Result Cache ist ein Memory-Bereich innerhalb des Client-Prozesses, der SQL-Query-Ergebnisse der OCI-Applikationen speichert. Es wird empfohlen, Client-Result-Caching nur für Abfragen von „read-only"- oder „read-mostly"-Tabellen zu nutzen.

Der Entwickler muss nur entscheiden, ob man den Result Cache nutzen möchte oder nicht, und der DBA kümmert sich um einige Parameter, um alles optimal auf die Anwendungen abgestimmt bereitzustellen.

Um eine Nutzung zu ermöglichen, sind Parameter zu definieren. Dabei ist zu beachten, dass die Defaults eher zu klein und auf jeden Fall anzupassen sind. Über „ALTER SESSION", „ALTER SYSTEM" oder per Hint wird der Result Cache eingeschaltet und nutzbar. Der DBA legt durch die Parameter den individuellen Rahmen für den Result Cache fest *(siehe Tabelle 1)*. Dabei gilt:

- „RESULT_CACHE_MAX_RESULT" legt den prozentualen Anteil am Result Cache für die einzelnen Ergebnisse fest
- „RESULT_MAX_SIZE" definiert die Gesamtgröße des reservierten Bereichs im Shared-Pool in Bytes
- „RESULT_CACHE_MODE" bietet den Automatismus oder die manuelle Steuerung. Bei „MANUAL" ist explizit ein Hint erforderlich. „FORCE" ist nicht zu empfehlen, da ansonsten alle SQL-Queries und PL/SQL-Ergebnisse im Cache landen
- „RESULT_CACHE_REMOTE_EXPIRATION" legt die Anzahl der Minuten für die Validität eines Result-Set fest

RESULT_CACHE_MAX_RESULT	5 (%)
RESULT_CACHE_MAX_SIZE	abh. von O/S
RESULT_CACHE_MODE	MANUAL/FORCE
RESULT_CACHE_REMOTE_EXPIRATION	0 (min)

Tabelle 1

```
SELECT /*+ RESULT_CACHE*/ * FROM tabelle
ALTER TABLE tabelle RESULT_CACHE FORCE
ALTER SESSION SET RESULT_CACHE_MODE=FORCE
```

Listing 3

Lang laufende und rechenintensive SQL-Abfragen beziehungsweise PL/SQL-Funktionen müssen identifiziert werden. Vorhersehbare SQL-Abfragen oder gleichbleibende deterministische Ergebnismengen werden erkannt und die DML-Aktivitäten auf den zugrunde liegenden Tabellen beobachtet, damit nur Tabellen mit geringen Änderungen für die Result-Cache-Funktion genutzt werden.

Den Result Cache optimal nutzen

Die Nutzung des Result Cache *(siehe Abbildung 4)* ist durch drei Faktoren bestimmt: auf SQL-Query-Ebene durch den Hint „RESULT_CACHE", bei der Einstellung „RESULT_CACHE" in der Tabellen-Definition und durch den Session-Parameter „RESULT_CACHE_MODE" *(siehe Listing 3)*. *Listing 4* zeigt eine Beispiel-Query ohne Hin-

weise. Das Monitoring des Result Cache ist sehr einfach und der Report bietet einen einfachen und schnellen Überblick *(siehe Listing 5)*.

In-Memory Parallel Query

Die Antwortzeit eines Full-Table-Scans lässt sich durch die Nutzung mehrerer paralleler Execution-Server deutlich verbessern. Falls der Server viel Speicher besitzt, kann die Datenbank Informationen in der SGA cachen, anstatt „Direct Reads" in der PGA auszuführen. Parallel Query kann sowohl individuell und manuell eingestellt als auch automatisch genutzt werden, und das sogar über Rechner-Grenzen hinaus in einem RAC-Verbund. Zur Verfügung stehende Ressourcen werden dadurch optimal genutzt. Bei der automatischen Nutzung („Automatic Parallel Degree Policy") legt Oracle alles selbstständig fest, sowohl ob überhaupt eine Parallelisierung durchgeführt wird, als auch, welcher Parallelisierungsgrad genutzt wird *(siehe Listing 6)*.

Jedes SQL-Statement wird optimiert und durchläuft einen Parallelisierungs-Prozess beim Parsen. Wenn die Parallelisierung genutzt wird, kommen folgende Schritte zum Tragen: Die User-Session oder der Shadow-Prozess übernehmen die Rolle des Koordinators, auch „Query Koordinator" genannt. Dann beschafft dieser die notwendige und verfügbare Anzahl von parallelen Servern, damit das SQL-Statement seine Abfolge von Operationen (ein Full Table Scan, um einen Join auf „non indexed" Spalten durchzuführen, eine „ORDER BY"-Clause etc.) verarbeiten kann. Jeder Parallel-Execution-Server versucht, seine Operation parallel und – soweit möglich – unabhängig von anderen auszuführen. Wenn die Parallel-Server die Abarbeitung des Statements beenden, übernimmt der Query-Koordinator alle Arbeit, die nicht parallel erledigt werden kann, um letztendlich das Ergebnis an den Benutzer zu geben. Der SQL-Tuning-Advisor hilft bei der Analyse und Optimierung enorm. Bei diesen beiden Nutzergruppen zeigen sich deutliche Leistungsgewinne *(siehe Abbildung 5)*:

- *Allgemeiner Anwender*
  - Kleinere Datenmenge
  - Shared Pool ausreichend
  - Direct Path Read
- *Führungskraft*
  - Große Datenmenge
  - Shared Pool zu klein

Dieses Beispiel zeigt, welche Performance-Gewinne zu erzielen sind, aber dass auch andere Komponenten wie ein zu kleiner Shared Pool Auswirkungen haben. Es kommt immer auf eine abgestimmte Kombination der unterschiedlichen In-Memory-Funktionen an.

Smart Flash Cache

Wie bereits erwähnt, sind die durchschnittlichen Server mit zu kleinem Hauptspeicher für In-Memory-Computing ausgestattet. Natürlich ist eine Aufrüstung möglich, allerdings stößt man bei weitverbreiteten Servern relativ schnell an die Hardware-Begrenzungen von 768 GB pro Server bei Zwei-Socket-Maschinen. Dies ist in vielen Anwendungsfällen nicht ausreichend, insbesondere in Hinblick auf die neuen In-Me-

```
SELECT
   a.department_id           "Department",
   a.num_emp/b.total_count   "%_Employees",
   a.sal_sum/b.total_sal     "%_Salary"
FROM (
   SELECT department_id,
          COUNT(*)       num_emp,
          SUM(salary)    sal_sum
   FROM employees
   GROUP BY department_id
) a, (
   SELECT
       COUNT(*)       total_count,
       SUM(salary)    total_sal
    FROM employees
) b
ORDER BY a.department_id;
```

Listing 4

```
SQL> set serveroutput on
SQL> execute dbms_result_cache.memory_report()
R e s u l t   C a c h e   M e m o r y   R e p o r t
[Parameters]
Block Size          = 1K bytes
Maximum Cache Size  = 6M bytes (6K blocks)
Maximum Result Size  = 307K bytes (307 blocks)
[Memory]
Total Memory = 151840 bytes [0.022% of the Shared Pool]
... Fixed Memory = 5296 bytes [0.001% of the Shared Pool]
... Dynamic Memory = 146544 bytes [0.021% of the Shared Pool]
....... Overhead = 113776 bytes
....... Cache Memory = 32K bytes (32 blocks)
........... Unused Memory = 27 blocks
........... Used Memory = 5 blocks
............... Dependencies = 1 blocks (1 count)
............... Results = 4 blocks
................... SQL  = 4 blocks (3 count)
```

Listing 5

```
Manuell
ALTER TABLES sales PARALLEL 8;
ALTER TABLE customers PARALLEL 4;

Automatisch
PARALLEL_DEGREE_POLICY is set to AUTO
```

Listing 6

mory-Funktionen der Datenbank 12*c* und die Ziele von In-Memory-Computing –die Realisierung von Big Data, Echtzeit-Analysen und vielem mehr.

Der Hauptspeicher (Level 1) ist leicht durch PCI-Flashkarten (Level 2) aufzurüsten. Diese Server-Aufrüstung ist nicht mit der Flash-Erweiterung des Storage-Systems zu verwechseln, womit andere Ziele verfolgt und die besten Vorteile erst durch die Nutzung intelligenter Software erzielt werden, die auf die Datenbank abgestimmt ist, wie es bei einer Exadata der Fall ist. Folgendes ist für die Nutzung der Smart-Flash-Cache-Funktionalität nötig:

- *Hardware-Voraussetzung*
  - Flashspeicher (PCI) im Server
- *Software-Anforderungen*
  - Datenbank EE (11*g* R2 od. 12*c*)
  - Betriebssystem
    - Oracle Linux
    - Solaris

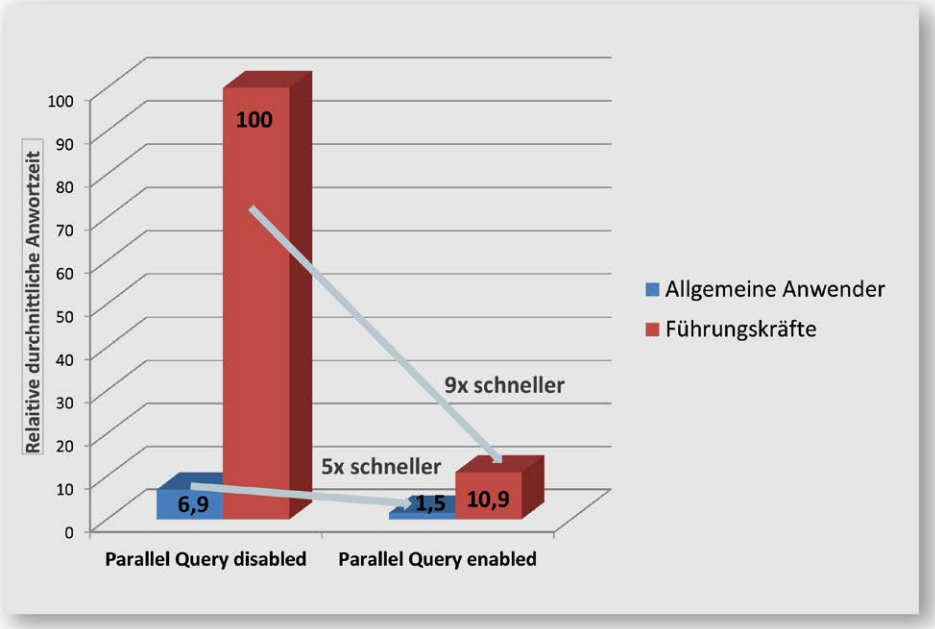


Abbildung 5: In-Memory Parallel Query

Der Smart-Flash-Cache ist eine Erweiterung des Datenbank-Buffer-Cache. Die PCI-Flashkarten sind günstiger als Memory-Komponenten, allerdings auch etwas langsamer, aber immer noch deutlich schneller als eine Platte. Außerdem bietet diese Second-Level-Cache-Erweiterung den Vorteil, dass sie eine weitaus größere Kapazität hat als der normale maximale Hauptspeicher-Ausbau der Standard-Server. Bei Lese-Operationen lässt sich ein Performance-Gewinn von bis zu Faktor „100“ gegenüber einer Disk erzielen. Dies kommt gerade den erwähnten Anwendungsfällen wie Analyse, Big Data etc. zugute. Durch diese Erweiterung und Nutzung als Buffer-Cache-Erweiterung steigt letztendlich der I/O-Durchsatz des Gesamtsystems deutlich an. Diese Konfiguration kommt in der neuen Oracle Database Appliance (ODA X5) zum Einsatz *(siehe Abbildung 6)*.

Dieses Verhalten der Datenbank bringt einen klaren Performance-Vorteil, was vielfach dokumentiert ist und durch das folgende Schaubild verdeutlicht wird. Es ist aber auch erkennbar, dass es Grenzwerte gibt, ab denen ein größerer Smart-Flash-Cache keinen Leistungsschub mehr bringt, sondern der interne Verwaltungsaufwand der Datenbank die Leistungsverbesserung übersteigt. Dazu gibt es keine allgemeingültigen Aussagen, sondern dies

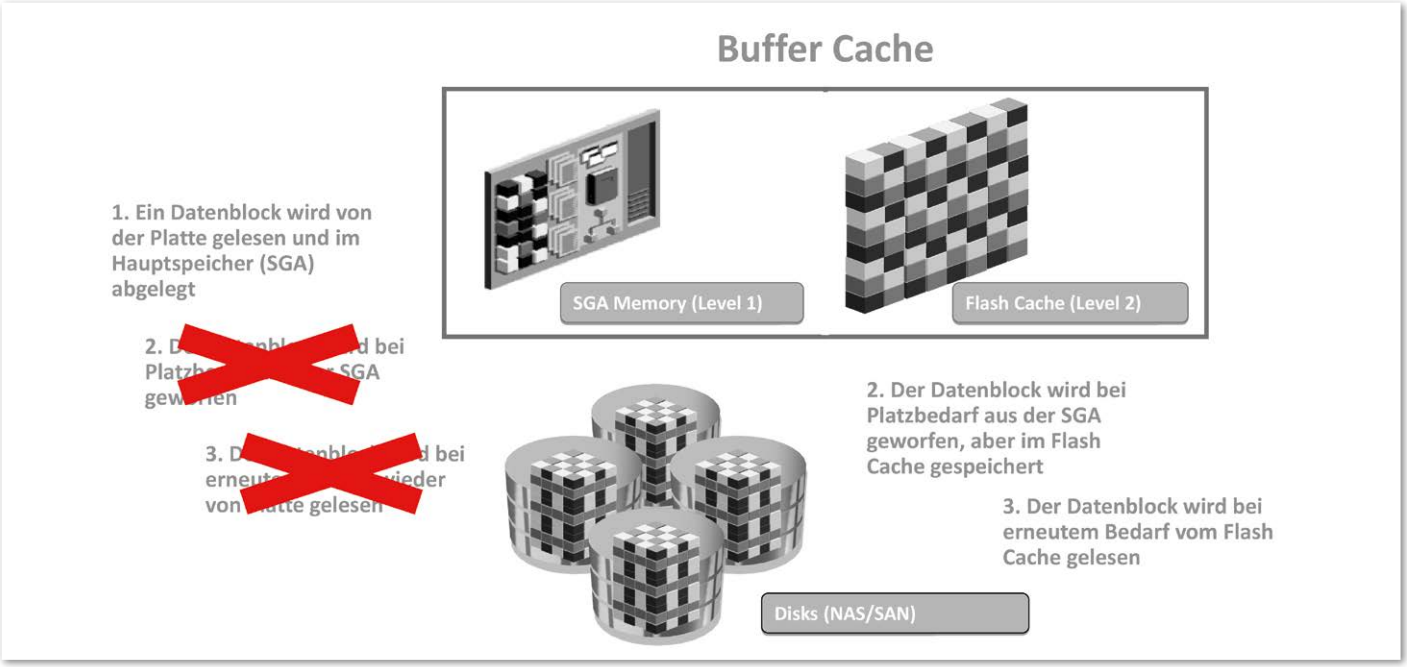


Abbildung 6: Beim Oracle-Datenbank-Verhalten mit Smart Flash Cache entfallen die Punkte 2 und 3

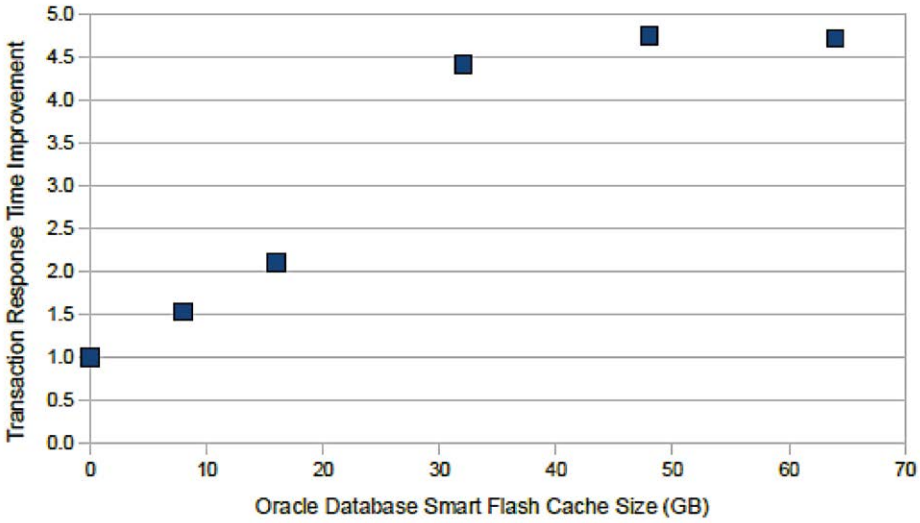


Figure 3: Response Time Improvement with Increasing Database Smart Flash Cache Size

Abbildung 7: Smart Flash Cache – Performance-Verbesserung (siehe „http://de.slideshare.net/ylouis83/11g-r2-flashcachetips“)

ist Projekt- bzw. Anwendungs-spezifisch zu klären.

Die Datenbank läuft auf Oracle Linux oder Solaris. Der Performance-Flaschenhals liegt beim Buffer-Cache; die Buffer-Pool-Advisory-Sektion des Automatic-Workload-Repository- Reports (AWR) oder der STATSPACK-Report zeigen, dass die Verdoppelung der Größe des Buffer-Cache von Vorteil wäre. Bei den Top-Wait-Events erscheint „db_file_sequential_read” und es sind freie CPU-Ressourcen vorhanden. Dazu sind die Init-Parameter „db_flash_cache_file = {OS-Pfad zur Flash Disk}“ und „db_flash_cache_size = {Größe der Flash Disk}“ einzustellen sowie die Strategien zum Pre-Loading von Objekten zu beachten. Die Storage-Klausel „FLASH_CACHE { KEEP | NONE }” wird auf Tabellen-Ebene gesetzt. Der zusätzliche SGA-Bedarf für Metadaten-Verwaltung (pro Datenbank-Block 100 Byte; auf RAC-Systemen 200 Byte) ist beim Memory-Sizing zu berücksichtigen. Wichtig zu wissen ist, dass im RAC die Flash-Disks exklusiv von einer Instanz genutzt werden.

Fazit

Die Nutzung der vorgestellten In-Memory-Funktionen für die Oracle Datenbank 11*g* R2 und das erste 12*c*-Release bringen enorme Leistungsgewinne. Der höchste Mehrwert liegt in der Kombination der Technologien und ist Datenbank-, Kunden- beziehungsweise Anwendungs-spezifisch. Ergänzt und deutlich erweitert werden diese Möglichkeiten durch neue, im Ausblick dargestellte Funktionen.

Ausblick

Mit der Oracle-Datenbank 12*c* (ab 12.1.0.2) sind die In-Memory-Funktionen der Datenbank in drei wesentlichen Punkten erweitert worden. Zwei Funktionen sind mehr oder minder unbemerkt kostenfrei im Basisumfang der Oracle-Datenbank Enterprise Edition enthalten:

- *Full DB in Memory*
  Die ganze Datenbank im Hauptspeicher zu halten, ist eine gute Möglichkeit, um I/O-Engpässe zu vermeiden. Durch die Nutzung einer Kombination von Level-1- und Level-2-Hauptspeicher für die Oracle-Datenbank (Smart Flash Cache) lassen sich eventuelle Hardware-Restriktionen einfach umgehen, etwa Zwei-Sockel-Server mit einem maximalen Memory-Ausbau von 768 GB. Oracle bietet aber auch Alternativen, bei denen Server mit bis zu 32 TB Memory ausgestattet sein können.

- *Automatic Big Table Caching*
  Datenbanken mit großen Tabellen können enorm vom Performance-Gewinn durch „Automatic Big Table Caching“ profitieren.

Besondere Beachtung findet natürlich die In-Memory-Datenbank-Option für die Version 12*c*, die eine spaltenorientierte Datenhaltung und alle von Kunden geforderten Grundfunktionen wie Sicherheit, Verfügbarkeit, Persistenz, sicheres Backup und Recovery bietet. Innerhalb eines RAC ist sogar eine Parallelisierung beziehungsweise Duplizierung des Column-Stores über Rechnergrenzen hinaus möglich. Es lassen sich Performance-Steigerungen erzielen, wie sie früher undenkbar waren. Dadurch werden die Verarbeitung und Analyse von Massendaten in nahezu Echtzeit möglich, ohne dass eine Voraggregation der Daten notwendig ist.

Ideal ist natürlich die Kombination mit den vorgestellten 11*g*-R2-In-Memory-Funktionen. Oracle hat darauf geachtet, dass die Inbetriebnahme der In-Memory-Datenbank-Option für die 12*c* ohne Migration und Neu-Implementierung nicht nur möglich, sondern so einfach wie ein Schalter zu nutzen ist – also einfach einschalten und ausprobieren.

Weitere Informationen

1. Memory Tuning: https://docs.oracle.com/database/121/TGDBA/part3.htm#sthref644
2. Smart Flash Cache: http://docs.oracle.com/cd/E11882_01/server.112/e25494/memory.htm#ADMIN13391
3. Parallel Query: http://docs.oracle.com/cd/E11882_01/server.112/e25523/parallel002.htm
4. DBACommunity Tipp: https://apex.oracle.com/pls/apex/GERMAN_COMMUNITIES.SHOW_RESOURCE_BY_FNAME?P_TIPP_ID=362&P_FILE_NAME=index.html
5. MOS Note 787373.1

Matthias Weiss
matthias.weiss@oracle.com

Maximum Availability Architecture trifft auf Oracle Multitenant

Ludovico Caldara, Trivadis AG

Dieser Artikel zeigt die Funktionsweise der Pluggable Databases in einer Maximum Availability Architecture sowie deren Aufbau und Services.

Oracle hat mit der Datenbank 12*c* die Multitenant-Option herausgebracht. Sie galt von Anfang an als „die" Lösung für die Konsolidierung und den Aufbau privater Cloud-Lösungen. Die Konsolidierung vieler Pluggable Databases in eigenständige Instanzen ist jedoch nur beschränkt möglich. Dies betrifft vor allem die Skalierbarkeit des Servers: Sobald die Oracle-CDB-Instanz die gesamten Host-Ressourcen in Anspruch nimmt, muss die Konsolidierung neu erstellter Container Databases (CDB) auf verschiedenen Servern erfolgen *(siehe Abbildung 1)*.

Diese Art der Konsolidierung kann bald zu einem Überladen der CDB führen, sodass Kunden ihre Pluggable Databases (PDB) in unterschiedliche CDBs verschieben müssen, was zu einem steigenden Ressourcen-Verbrauch führt *(siehe Abbildung 2)*.

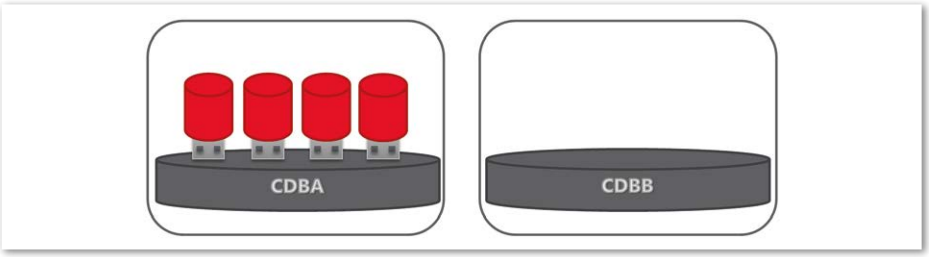


Abbildung 1: Konsolidierung neu erstellter CDBs auf verschiedenen Servern

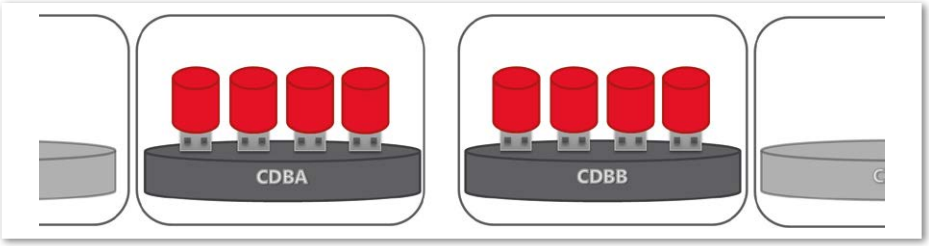


Abbildung 2: Steigender Ressourcen-Verbrauch

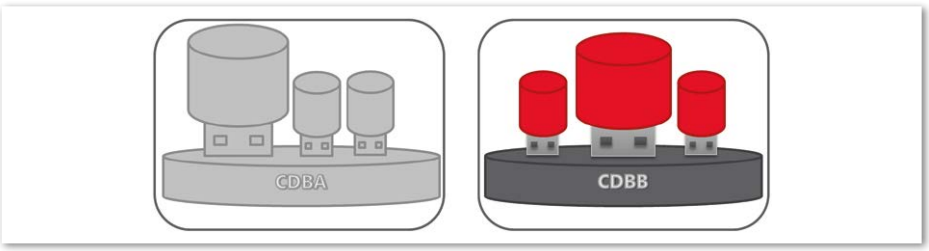


Abbildung 3: Unterschiedliche Wartungsfenster

Zudem entstehen Probleme bei der Verfügbarkeit: Sobald Kunden statische Parameter ändern oder die Soft- beziehungsweise Hardware warten oder patchen möchten, können die vielen auf einer CDB konsolidierten PDBs die Planung einer Ausfallzeit erschweren, weil die verschiedenen Anwendungen unterschiedliche Wartungsfenster benötigen *(siehe Abbildung 3)*.

Kurz gesagt, die Multitenant-Implementierung auf eigenständigen Instanzen löst das Silo-Paradigma nicht auf; sie verlagert lediglich das Problem von der Datenbank- auf CDB-Ebene.

Oracle Real Application Clusters kann als Backend-Lösung den Unterschied ausmachen, da sie die Verfügbarkeit und Skalierbarkeit bietet. Um die Verfügbarkeit sicherzustellen, läuft die CDB auf mehreren Instanzen; der Ausfall einer Instanz wirkt sich nicht auf die Anwendungen aus *(siehe Abbildung 4)*. Hinsichtlich der Skalierbarkeit ist es möglich, neue Instanzen hinzuzufügen, um weitere PDBs zu aktivieren, sobald neue Ressourcen erforderlich sind *(siehe Abbildung 5)*.

Die vielen PDBs konkurrieren nicht um die Ressourcen, weil jede PDB gezielt auf nur eine Teilmenge der Instanzen geöffnet werden kann, was den Ressourcen-Verbrauch auf alle Instanzen verteilt. Welche Instanzen auf welchem Server geöffnet sind, hängt von den Services ab, die auf Cluster-Ebene definiert sind.

Der Umgang mit PDBs und Services

Die PDBs werden nach dem Erstellen standardmäßig bereitgestellt. Im folgenden Beispiel hat eine PDB mit dem Namen „MAAZ" auf beiden Instanzen im Zwei-Knoten-RAC „CDB" den Status „MOUNT" *(siehe Listing 1)*. Wenn aber ein Service für

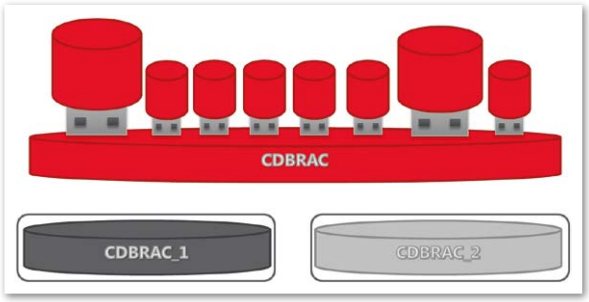


Abbildung 4: Der Ausfall einer Instanz wirkt sich nicht auf die Anwendungen aus

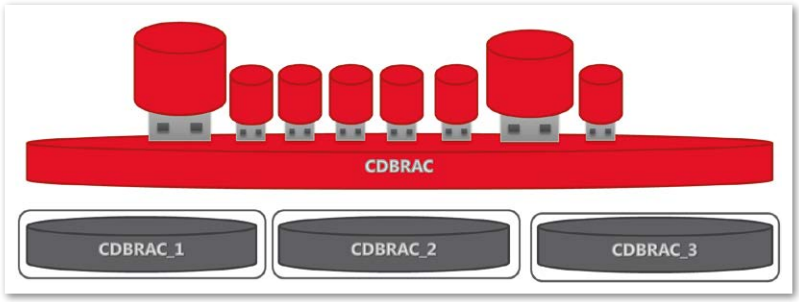


Abbildung 5: Neue Instanzen hinzufügen

die PDBs existiert, öffnet sich beim Start des Service die PDB *(siehe Listing 2)*.

Es ist zu sehen, dass die „Srvctl"-Befehle einen neuen Schalter „-Pdb" haben, der angibt, welcher PDB der neu erstellte Service zugewiesen wird. In diesem Fall wurde der Service als „Singleton" definiert: Er läuft nur in einer Instanz, deshalb wird die PDB nur für diese Instanz geöffnet *(siehe Listing 3)*.

Auf der anderen Seite schließt das Beenden des Service die PDB nicht: Alle Sitzungen bleiben verbunden, auch wenn der Dienst beendet wird. DBAs müssen die Pluggable Database manuell beenden, wenn sie die Instanz bereinigen möchten. Es ist möglich, PDBs selektiv für eine bestimmte Anzahl von Instanzen zu öffnen, indem man verschiedene Services mit unterschiedlichen Größen öffnet. Dies ermöglicht eine vollständige Trennung der Ressourcen sowie eine deutlich höhere Skalierbarkeit, gepaart mit Multitenant, da jede Instanz über einen eigenen Redo-Thread, Undo-Tablespace und vor allem einen eigenen Buffer-Cache verfügt: Der Buffer-Cache einer Instanz wird nur mit Blöcken gefüllt, die zu den PDBs dieser Instanz gehören *(siehe Abbildung 6)*.

Diese Trennung der Ressourcen zeigt, dass RAC die wichtigen Technologie-Voraussetzungen schafft, die nicht nur eine höhere Verfügbarkeit der Multitenant-Architektur bieten, sondern auch die Skalierbarkeit steigern: Bei großen Ressourcen-Anforderungen ist es möglich, dem Cluster einen oder mehrere Knoten hinzuzufügen und die Dienste/PDBs auf die neu erstellten Instanzen auszugleichen.

Die wichtigste Einschränkung ist die maximale Anzahl von Diensten: Es können nicht mehr als 512 Services auf einem einzigen CDB erstellt werden; danach ist eine neue CDB erforderlich. Die Anzahl der Services ist ein wichtiger Faktor für eine erfolgreiche Konsolidierungsstrategie in großen Umgebungen.

Ein weiterer wichtiger Punkt ist, dass, wenn ein Knoten abstürzt, bevor die Sessions umschalten können, die PDB zu öffnen ist: Wenn der Service „Singleton" ist und keine weiteren Instanzen die PDB geöffnet haben, müssen die Sessions warten, bis eine neue Instanz bereitsteht. Das Öffnen kritischer PDBs auf mehr als einer Instanz kann dieses Problem lösen.

RAC, Data Guard und Multitenant

Oracle Multitenant bringt in einer Oracle-Data-Guard-Umgebung einen wesentlichen Vorteil: Da die Redo-Threads über alle PDBs verteilt sind, ist nur ein Standby-CDB erforderlich, in der alle PDBs repliziert werden. Damit ist auch nur eine Data-Guard-Konfiguration notwendig, was zu einer einfachen Wartung und zu einem enorm reduzierten administrativen Aufwand führt *(siehe Abbildung 7)*.

Der Nebeneffekt dieser Architektur ist, dass die Granularität der Switchover- und Failover-Operationen auf CDB-Ebene erfolgt, was bedeutet, dass alle PDBs in einem CDB die gleiche Rolle wie die CDB innehaben; sie stehen entweder alle auf „primary" oder alle auf „standby". Aus Sicht der Konsolidierung ist es wichtig auszuwählen, welche PDBs auf welche CDBs konsolidiert werden müssen, sodass später eine gewisse Flexibilität für die kritischsten PDBs erreicht wird.

```
SYS@CDBATL_2> select INST_ID, CON_ID, name, OPEN_MODE
2  from gv$pdbs where con_id!=2 order by name, inst_id;

INST_ID          CON_ID                    NAME       OPEN_MODE
----------    ---------- ------------------------      ----------
1                     3                    MAAZ       MOUNTED
2                     3                    MAAZ       MOUNTED
```

Listing 1

```
$ srvctl add service -db CDBATL -service maazapp -serverpool CDBPOOL
-cardinality
singleton -role primary -failovertype select  -failovermethod basic
-policy
automatic -failoverdelay 2 -failoverretry 180 -pdb maaz
$ srvctl start service -db CDBATL -service maazapp -instance CDBATL_1
```

Listing 2

```
INST_ID          CON_ID                    NAME       OPEN_MODE
----------    ---------- ------------------------      ----------
1                     3                    MAAZ       READ WRITE
2                     3                    MAAZ       MOUNTED
```

Listing 3

Eine PDB erstellen und in einer MAA-Umgebung klonen

Bei der Erstellung einer neuen PDB aus „PDB$SEED“ („Standard“-Einstellung) werden die Datendateien der PDB$SEED-Datenbank kopiert und der neu erstellten PDB zugewiesen. Auf der Stand-by-Seite kopiert die Data-Guard-Technologie die PDB$SEED-Datendateien ebenfalls von der lokalen (Stand-by-)Seed-Datenbank, um den Erstellungsprozess transparent zu machen und dem Recovery-Prozess weiterhin das Wiederherstellen neuer Datendateien zu ermöglichen *(siehe Abbildung 8)*.

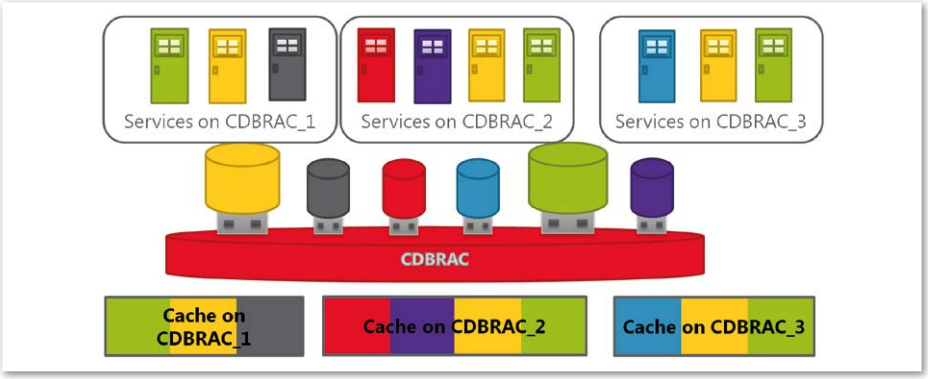


Abbildung 6: Der Buffer-Cache

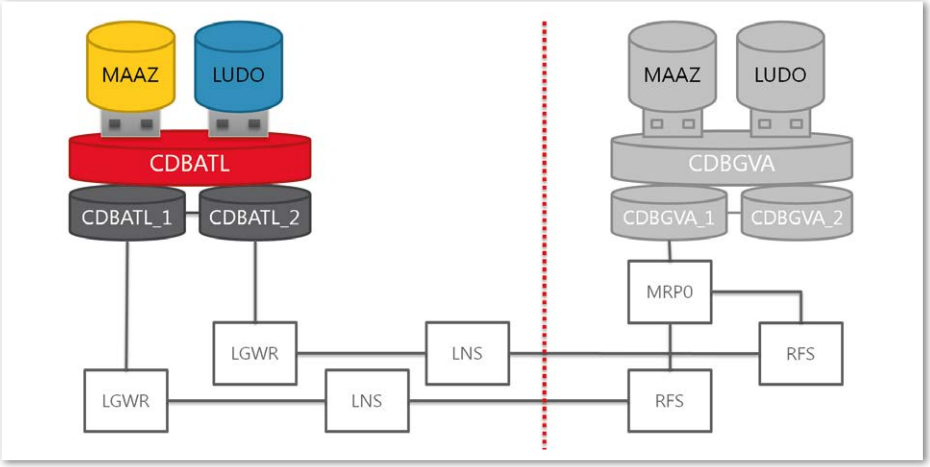


Abbildung 7: Nur ein Stand-by-CDB

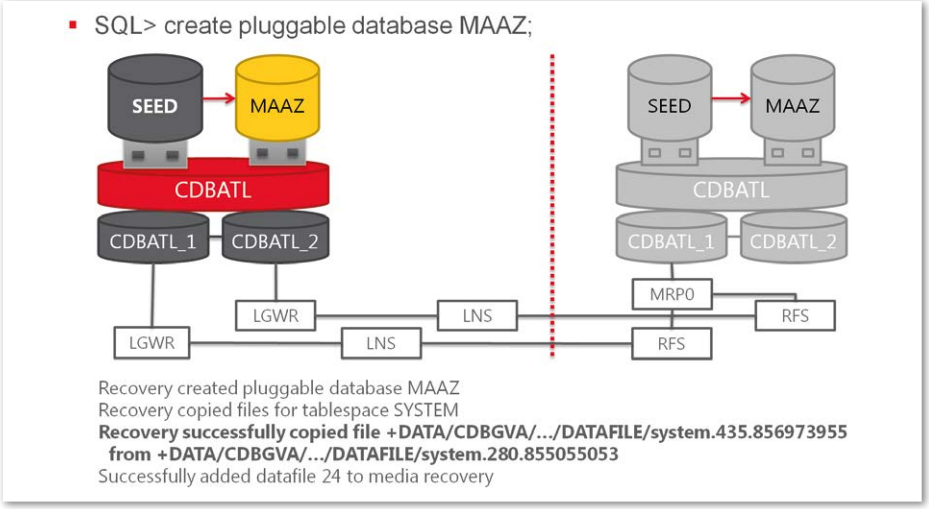


Abbildung 8: Eine neue PDB erstellen

Sobald die PDB durch das Klonen einer bestehenden PDB („create pluggable database A from B;“) erstellt ist, wird die Quelldatenbank nur dann auf beiden Seiten kopiert („primary“ und „stand-by“), wenn die Stand-by-Datenbank schreibgeschützt geöffnet ist, was eine Oracle-Active-Data-Guard-Lizenz erfordert *(siehe Abbildung 9)*.

Wenn Active Data Guard nicht lizenziert ist, sagt die Dokumentation, dass es notwendig ist, die Datendateien vor dem Klonen in die Stand-by-Datenbank zu kopieren *(siehe „http://docs.oracle.com/database/121/SBY-DB/create_ps.htm#SBYDB5260“)*: „If you plan to create a PDB as a clone from a different PDB, then copy the data files that belong to the source PDB over to the standby database. This step is not necessary in an Active Data Guard environment because the data files are copied automatically when the PDB is created on the standby database.“

In einer RAC-Umgebung werden die Datendateien jedoch von ASM verwaltet und es gibt keine Möglichkeit, diese im gleichen Pfad zu kopieren, der in den Controlfiles angegeben ist. Deshalb ist nach dem Klonen eine manuelle Wiederherstellung der neuen PDB auf der Stand-by-CDB erforderlich *(siehe Abbildung 10)*.

Oracle hat dazu eine MOS-Note veröffentlicht: „Making Use of the STANDBYS=NONE Feature with Oracle Multitenant“ (Doc ID 1916648.1). Sie beschreibt den besten Weg für eine PBD-Erstellung in einer Data-Guard-Umgebung ab Release 12.1.0.2.

Der Umgang mit PDBs und Services

In einer Data-Guard-Umgebung hat sich die Möglichkeit, neue Services zu erstellen, nicht geändert. Wenn man einen Read-only-Service auf der Stand-by-Datenbank (Active Data Guard) erstellt, ist nur die entsprechende Rolle („physical_standby“) anzugeben. Der einzige Unterschied besteht darin, die richtige PDB mit dem „-pdb“-Schalter anzugeben *(siehe Listing 4)*. Man sollte nochmal im Auge behalten, dass es ein Gesamtlimit von 512 Services in einem CDB gibt. In einer Active-Data-Guard- und Multitenant-Umgebung gibt es für jede PDB:

- Einen Standard-Service mit dem Namen der PDB
- Einen Read-Write-Service (mehr als empfohlen)
- Einen Read-only-Service für die Stand-by-Datenbank

Die maximale Anzahl von PDBs pro CDB sind dann 512/3 =170 PDBs, also weit unter der Grenze von 252 PDBs pro CDB. Je mehr Services, die man pro PDB erstellt, desto weniger PDBs kann man in der gleichen CDB konsolidieren. Die „connection description" hat sich gegenüber dem herkömmlichen MAA- Connection-String nicht geändert, außer dass der „SERVICE_NAME" auf den in der ausgewählten PDB zugewiesenen Service zeigen muss *(siehe Listing 5)*. Aus Sicht der Konsolidierung ist ein hochverfügbares Oracle Internet Directory (OID) zur Auflösung der Namen dringend angeraten.

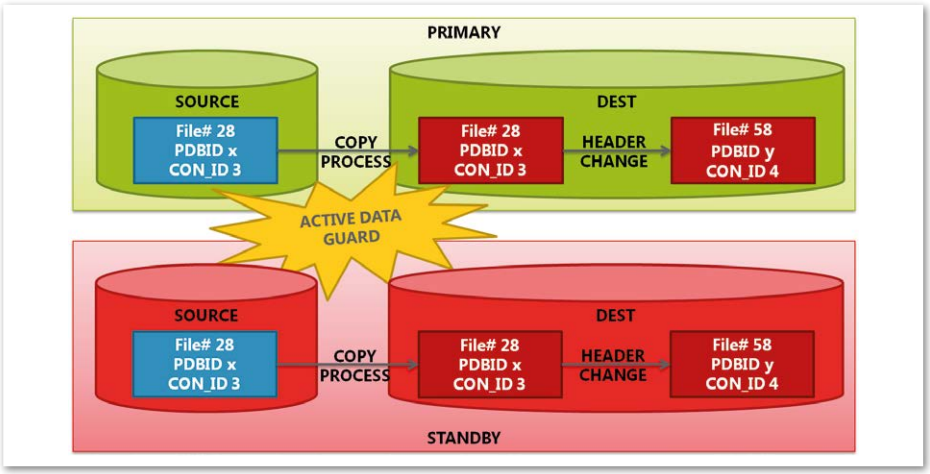


Abbildung 9: Oracle-Active-Data-Guard-Lizenz erforderlich

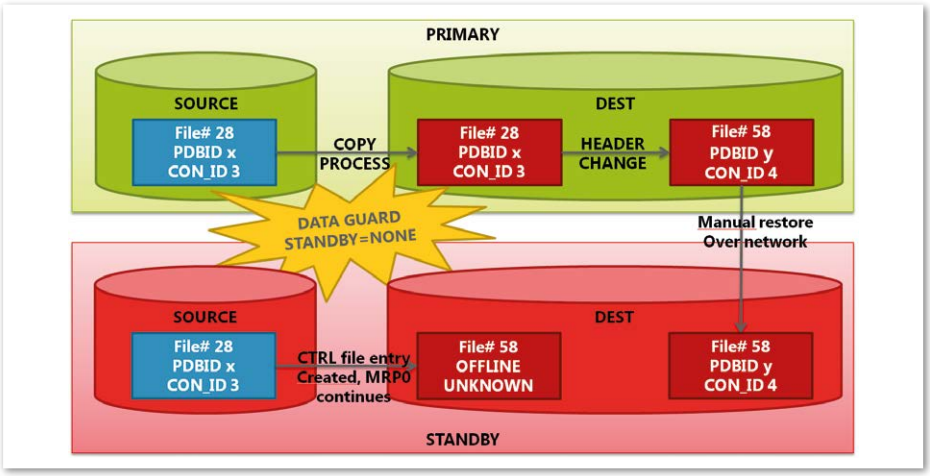


Abbildung 10: Manuelle Wiederherstellung der neuen PDB

```
$ srvctl add service -db db_unique_name-service ro_service_name \
-serverpool server_pool -cardinality uniform -role physical_standby \
-failovertype select -failovermethod basic -policy automatic \
-failoverdelay 2 -failoverretry 180 -pdb pluggable_database
```

Listing 4

```
LUDOAPP = (DESCRIPTION_LIST= (LOAD_BALANCE=off) (FAILOVER=on) (DESCRIP-
TION = (CONNECT_TIMEOUT=5)
(TRANSPORT_CONNECT_TIMEOUT=3) (RETRY_COUNT=3) (ADDRESS_LIST=(LOAD_
BALANCE=on)
(ADDRESS = (PROTOCOL = TCP)(HOST = raca-scan)(PORT = 1521))) (CONNECT_
DATA = (SERVICE_NAME = LUDOAPP))    )
(DESCRIPTION =    (CONNECT_TIMEOUT=5) (TRANSPORT_CONNECT_TIMEOUT=3)
                  (RETRY_COUNT=3) (ADDRESS_LIST=
                  (LOAD_BALANCE=on)
(ADDRESS = (PROTOCOL = TCP)(HOST = racb-scan)(PORT = 1521))) (CONNECT_
DATA = (SERVICE_NAME = LUDOAPP)) )  )
```

Listing 5

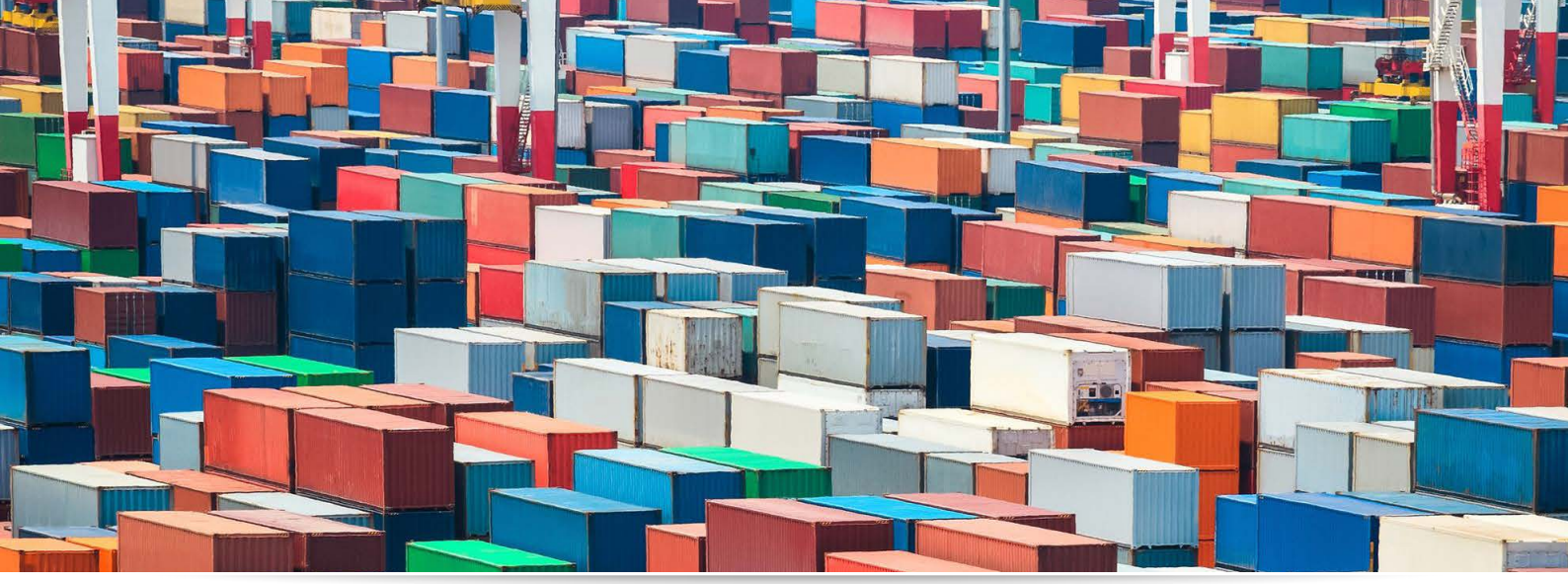
Fazit

Active Data Guard ist eine zunehmend attraktive Option geworden und trotz der Beschränkungen beim PDB-Klonen auf nicht aktiven Stand-bys kein Nice-to-have-Produkt. Active Data Guard ist die beste Lösung für eine Data-Guard- und Multitenant-Beschränkung. Die Konsolidierung mit Multitenant erfordert im Idealfall alle drei Optionen (Multitenant, Real Application Cluster, Active Data Guard), die für Kunden trotz der vielfältigen Vorteile nicht leicht erschwinglich sind.

Aus Sicht der Konsolidierung ist die MAA- und Multitenant-Architektur die beste Wahl in Richtung einer Cloud-Infrastruktur, denn sie erfüllt alle Anforderungen in Bezug auf Konsolidierungsdichte, Skalierbarkeit, Verfügbarkeit und einfache Administration; deshalb würde der Autor nicht zögern, sie zu empfehlen. Zudem hat Oracle kürzlich die Freigabe der Nicht-CDB-Architektur angekündigt: Ein Weg, der Kunden die neue Architektur zur sicheren Wahl macht.

Hinweis: Ins Deutsche übersetzt von global syntax Language Management Services. Die englische Fassung des Artikels kann unter „*www.doag.org/go/News/201503/Caldara*" heruntergeladen werden.

Ludovico Caldara
Ludovico.Caldara@trivadis.com



Daten-Virtualisierung in Oracle 12c mit Docker-Containern

Franck Pachot, dbi-services

In diesem Jahr dreht sich alles um die Daten-Virtualisierung. Die Zeiten, in denen der Datenbank-Speicher für jede Preproduction, jeden Test, jede Entwicklung etc. vervielfältigt wurde, sind vorbei. Die Bereitstellung muss flexibler sein und es gibt bereits entsprechende Tools dafür. Für Anwendungs-Umgebungen ist Docker ein mögliches Tool. Es stellt Daten-Container über das Copy-on-Write-Dateisystem und Linux-Container bereit. Warum sollte man es also nicht auch für Datenbanken nutzen?

Achtung: Die beschriebene Vorgehensweise ist für die Produktion nicht geeignet. Es geht hier nicht um eine ausgereifte Lösung, sondern um einige Prototypen, die zeigen, was machbar ist, und praktische Erfahrungen liefern. Von daher freut sich der Autor über Feedback („*franck.pachot@dbi-services.com*").

Der Artikel beschreibt nicht, was Docker ist. Dafür gibt es unter *„www.docker.com"* eine eigene Website. Hier geht es um eine Oracle-Installation in einem Docker-Container anhand eines Beispiels. Dazu wird eine Stand-by-Datenbank erstellt, die sich laufend aus einer Quell-Datenbank aktualisiert. Darüber hinaus lassen sich neue virtuelle Datenbanken sehr einfach erstellen: Ein neuer Docker-Container stellt den Speicher für Daten bereit, die nicht aktualisiert werden.

Docker und Oracle installieren

Man kann Docker entweder unter Linux installieren oder die „boot2docker"-VirtualBox-VM verwenden. Hier werden „boot2docker" und VirtualBox auf einem Windows-Laptop installiert. Dadurch entsteht eine VirtualBox VM; über das Verzeichnis „/users" ist das Windows-Verzeichnis von Docker aus im Zugriff. Dazu werden die Oracle-Software-Installationsdateien gestartet. Hinweis: Die entsprechenden Befehle und Listings stehen in der englischsprachigen Version des Artikels unter *„www.doag.org/go/News/201503/Pachot"*.

Mit Docker beginnt man nie ganz von vorn. Man startet in einem Container, der aus dem Repository („Pull") heruntergeladen wird. Im öffentlichen Verzeichnis fehlt allerdings das Image mit dem bereits installierten Oracle. Grund dafür sind die Lizenz-Bedingungen, die eine VM mit bereits installierter Software nicht erlauben. Es gibt allerdings eine Ausnahme: Oracle XE. In der Tat enthält das Verzeichnis Docker-Images mit Oracle XE.

Hier kommt jedoch eine vollständige Oracle-Datenbank 12c mit Data Guard Stand-by zum Einsatz, die selbst zu installieren ist. Die Software, die von der Oracle-Website *(siehe „http://www.oracle.com/technetwork/database/enterprise-edition/downloads")* heruntergeladen wird, enthält die beiden Zip-Dateien „linuxamd64_12102_database_1of2.zip" und „linuxamd64_12102_database_2of2.zip". Diese werden in einem Verzeichnis abgelegt, das mit dem Docker-Container verbunden ist. Weil hier „boot2docker" zum Einsatz kommt, wird das Verzeichnis „/users" in der Boot2docker-VM verwendet. Dann startet man das „breed85/Oracle-12c"-Image mit dem Tag „12101" und verbindet es unter „/users" mit der Boot-2docker-VM. Wenn das „breed85/Oracle-12c"-Image fehlt, wird es heruntergeladen. In diesem Fall muss man dem Aufruf sowohl „unzip" als auch das „preinstall"-Package hinzufügen.

Im Yum-Cache wird etwas Platz geschaffen und das „/oracle"-Verzeichnis erstellt, in dem alles liegt, auf das Oracle zugreifen

möchte. Die Installation erfolgt im Silent-Modus, wofür eine Response-Datei zu erstellen ist. Anschließend wird der Oracle-User erzeugt. Man ignoriert eine Menge System-Voraussetzungen, weil die Konfiguration ja nicht im Support ist. Die Installation ist nach etwa fünf Minuten abgeschlossen und man kann die „root.sh" ausführen.

Da es sich um einen Test handelt, ist der Platz in der Docker-VM nicht besonders groß, sodass man die nicht mehr benötigten entpackten Dateien löscht. Das neue Image wird abgespeichert und der Container entfernt.

Die Datenbank duplizieren

Ziel ist es, die Virtualisierung einer vorhandenen Datenbank zu simulieren. Dazu wird eine vorhandene Datenbank mit RMAN dupliziert und als physische Stand-by-Datenbank eingerichtet. Die einfachste Quell-Datenbank ist mit „dbca" und allen Standard-Optionen schnell erstellt. Um die Sache einfach zu halten, ist es keine Container-Datenbank. Sie heißt „DEMO111" und ist auf „192.168.78.111" gehostet. Das „SYS"-Passwort lautet „Oracle"; „dg_brokerstart" ist auf „true" eingestellt, weil die Stand-by-Datenbank mit Data Guard erstellt werden soll. Die Docker-VM hat die IP-Adresse „192.168.59.104". Wer die Listings kopiert und einfügt, muss diese Werte ändern.

Zunächst wird eine Shell in einem Container in dem Image gestartet, das zuvor erstellt wurde. Der Host-Name lautet „DEMO111_SBY". Das Verzeichnis „/users" ist jetzt nicht mehr notwendig; ab sofort wird der Port „1521" auf „9000" umgeleitet. Damit ist der gestartete Listener auf Port „1521" im Container („listening to //DEMO111_SBY:1521 into the container") extern mit der Adresse „//192.168.59.104:9000" verbunden.

Jetzt wird die Umgebung eingerichtet. Der Name der Instanz ist „DEMO111", die erstellte Passwort-Datei identisch mit der Primär-Datenbank. Der Listener erhält einen statischen Service und wird gestartet. Die Verzeichnisse entstehen per OMF; für die Instanz ist eine „init.ora" erforderlich. Nachdem daraus „spfile" erzeugt und die Instanz gestartet wird, lässt sich die Netzwerk-Verbindung testen.

Wenn alles korrekt läuft, kann man jetzt die Quelle mit dem Ziel verbinden und eine RMAN-Duplizierung starten. Es ist zwar empfohlen, den Verbindungs-String in einer „tnsnames.ora" zu deklarieren, mit dem „ezconnect"-String geht es allerdings schneller, solange es weniger als 64 Zeichen sind.

Die Stand-by-Datenbank konfigurieren und aktualisieren

Über den Data-Guard-Broker ist die Stand-by-Datenbank schnell aufgesetzt. Es ist zu bedenken, dass es nur darum geht, eine synchronisierte Stand-by-Datenbank zu haben. Es wird nie in den Docker-Container gewechselt. Um die Konfiguration behalten zu können, werden die interaktive Shell beendet, das Image abgespeichert und anschließend der Container entfernt. Das neue Image hat jetzt zusätzlich 2,13 GB für die Datenbank.

Um die Stand-by-Datenbank laufend zu aktualisieren, wird ein Container ausgeführt, der permanent läuft. Er kann im Hintergrund sein, zum Testen befindet er sich jedoch in einem Fenster. Der Listener, die Instanz sowie Docker werden in einem anderen Fenster gestartet.

Die virtuelle Datenbank bereitstellen und öffnen

Die erstellen Images sind aufeinander aufgebaut und der „refresh"-Container läuft. Damit gibt es jetzt einen zusätzlichen virtuellen Datenbank-Server, der über 10,2 GB (System + Oracle-Software + Datendateien + archivierte Protokolle) verfügt, aber nur 1,79 GB Speicherplatz benötigt – genau die Größe des „vdb1"-Layers.

Um die neue virtuelle Datenbank zu verwenden, wird einfach das Image in einem neuen Container ausgeführt und Port „1521" an einen anderen Port umgeleitet. Die „rm"-Option entfernt den Container, wenn die Shell endet. Grund dafür ist, dass er nur für eine kurze Zeit genutzt wird.

Der Zugang zur virtuellen Datenbank erfolgt über den umgeleiteten Port „9001", der Verbindungs-String lautet demnach „//192.169.59.104:9001/DEMO111_SBY". Jetzt wird der Listener gestartet. Dabei ist zu beachten, dass die „listener.ora", die für einen anderen Host-Namen konfiguriert wurde, entfernt ist. Nach dem Start der Instanz wird die Stand-by-Datenbank als „primary" aktiviert, damit man sie mit „read/write" als sogenannten „Failover" öffnen kann.

Es kann eine gute Idee sein, den Container nach dem Start zu isolieren, um sicherzugehen, dass dieser neue „Primary" nicht versucht, mit der tatsächlichen „Primary" und der Stand-by-Datenbank zu kommunizieren. Wir könnten auch den Datenbank-Namen ändern, aber Container bieten bereits die notwendige Isolation. Letztendlich soll nur die Umleitung von Port „1521" nach Port „9001" erfolgen.

Als Ergebnis laufen jetzt zwei Container und man kann so viele virtuelle Datenbanken starten, wie man will. Jede wird als Docker-Container verwaltet. Dort laufen alle Instanzen und alles liegt in Containern isoliert auf demselben Server. Jetzt lässt sich die „vdb1" mit jedem beliebigen Client verbinden, der Zugriff auf die Docker VM über das Netz hat.

Fazit

Der Artikel zeigt die Konzepte der Daten-Virtualisierung, die man hier anhand eines Beispiels anschaulich erfahren kann. Die zugrunde liegenden Technologien (Copy-on-Write-Dateisystem, Virtualisierung, Isolierung) sind vorhanden, man kann sie mit einfachen Mitteln auf dem Laptop einsetzen. Das Beispiel ist so aufgebaut, dass man die Befehle *(siehe „www.doag.org/go/News/201503/Pachot")* direkt kopieren und einfügen kann, lediglich die IP-Adressen sind zu ersetzen.

Für die Produktion wird man natürlich robuste und unterstützte Software einsetzen; viele Hersteller bieten auch gute Lösungen für Daten-Virtualisierung, einfache Snapshots und Clones an. Diese Do-it-yourself-Übung schafft die Grundlagen für die Auswahl und die Herausforderungen im Umgang mit kommerziellen Lösungen.

Hinweis: Ins Deutsche übersetzt von global syntax Language Management Services. Die englische Fassung des Artikels kann unter *„www.doag.org/go/News/201503/Pachot"* heruntergeladen werden.

Franck Pachot
franck.pachot@dbi-services.com

Oracle Coherence: In-Memory-Computing für Einsteiger

Gabriele Jäger und Michael Fuhr, ORACLE Deutschland B.V. & Co. KG

Trotz Einsatz aktueller Hard- und Software-Komponenten, optimaler Datenstrukturen und Algorithmen, Tuning-Maßnahmen etc. zeigt eine Anwendung nach einiger Zeit nicht mehr die erwartete Performance. Häufig liegen dann die Ursachen in mangelnder Kapazität und zu hoher Latenzzeit.

Die speicherresidente Data-Grid-Lösung „Coherence" schafft Abhilfe bei Performance-Problemen und ermöglicht eine vorhersagbare Skalierung von Anwendungen für einen schnellen und zuverlässigen Zugriff auf häufig benötigte Daten. Hinzu kommen Datenanalysen in Echtzeit, Berechnungen im In-Memory Grid sowie eine parallele Transaktions- und Ereignisverarbeitung. Der Artikel zeigt neben den Grundprinzipien der Oracle In-Memory Data-Grid-Lösung den Mehrwert durch Anwendungsbeispiele wie RDBMS- und Mainframe-Offload, Anwendungen mit Zustands-Informationen in Web-Anwendungen, Http-Session-Management und Objekt-Interoperabilität (.NET, Java) auf.

Das In-Memory Data-Grid

Eine Anwendung wird entwickelt, getestet, produktiv geschaltet und läuft − die Anwender sind zufrieden. Aber über die Zeit wird die Anwendung langsamer und die Anwender beschweren sich über zu lange Antwortzeiten. In der Regel werden dann erst einmal ein Upgrade auf aktuellste Software-Komponenten, der Einsatz der aktuellsten Funktionalitäten, optimaler Datenstrukturen und Algorithmen, die Aktualisierung der Hardware und natürlich Tuning-Maßnahmen empfohlen. Diese Maßnahmen helfen meist nur kurzfristig, langfristig jedoch verschlechtert sich die Performance dieser Anwendung immer weiter. Es stellt sich die Frage nach dem wirklichen Problem.

Cameron Purdy, ehemaliger Gründer und CEO der Tangosol Inc., hat sich bereits im Jahr 2000 mit diesem Thema beschäftigt und mit Kollegen eine sehr erfolgreiche und gute Software-Lösung – das heutige Coherence – für diese Aufgabenstellung auf den Markt gebracht. Sieben Jahre nach Firmengründung wurde Tangosol von Oracle aufgekauft. Cameron Purdy ist mittlerweile Senior Vice President of Development der Oracle Corporation und verantwortet den Bereich „Cloud Application Foundation" mit dem WebLogic Server, Coherence und vielem mehr. Er beantwortet die Frage wie folgt: „Trust me, you have a capacity problem, not a performance problem." Wir haben also kein Performance-, sondern ein Kapazitätsproblem − ähnlich wie bei einer Warteschlange an einem Schalter. Der erste Kunde der Warteschlange fühlt eine gute Performance, da er sofort bedient wird, wohingegen der letzte Kunde der Warteschlange eine schlechte Performance erfährt, denn er muss warten, bis alle vor ihm bedient worden sind.

Eine kürzere Wartezeit lässt sich in diesem Fall durch den zeitgleichen parallelen Betrieb mehrerer Schalter erreichen. Diese sogenannte „horizontale Skalierung" ist auch die Lösung unseres Problems. Die Gerätekosten sind dank Commodity-Hardware abschätzbar, die Relation „Kosten zu Performance" ist nahezu linear und die physikalischen Limitierungen sind weitaus höher als bei einer vertikalen Skalierung.

Die System-Architektur einer horizontalen Skalierung benötigt etwas Aufmerksamkeit und sollte im Vorfeld betrachtet werden. Damit kommen wir zu unserer ersten Frage zurück: „Was ist ein In-Memory Data-Grid?" Im Internet gibt es mittlerweile viele Definitionen des Begriffs. So gibt der Java-Community-Weblog-Service eine Definition (siehe Abbildung 1).

Ein Data-Grid ist demnach ein System von mehreren Servern (Knoten), das Informationen in Form von Anwendungsobjekten auf diesen Knoten verwaltet und Operationen (wie Berechnungen) mit diesen Informationen ausführt. Ein In-Memory Data-Grid ist ein Data-Grid, das Informationen In-Memory speichert, um eine sehr hohe Performance zu erreichen.

Die Informationen werden in Anwendungsobjekten (Domain Objects) gehalten.

Ein In-Memory Data-Grid ist ein **[In-Memory] Datenmanagment-System zum Verwalten von Anwendungsobjekten**, welche **verteilt** genutzt werden können und mit denen Operationen durchgeführt werden können.

Es zeichnet sich aus durch
- eine geringe **Latenz** bei Zugriffen
- einen hohen **Durchsatz**
- vorhersehbare **Skalierbarkeit**
- hohe **Verfügbarkeit** und
- ein **robustes** Verhalten.

Abbildung 1: Definition eines In-Memory Data-Grid

Kopien der Objekte sind redundant auf verschiedenen Knoten des Systems verteilt. Diese Eigenschaft bildet die Grundlage für die Elastizität des Systems und die Hochverfügbarkeit der Information im Falle eines Knotenausfalls. Die Struktur der Anwendungsobjekte wird mittels objektorientierter Sprachen wie Java (POJOs), C++ oder C# definiert. Es existieren Schnittstellen (APIs) auf Basis dieser Sprachen, um „Create"-, „Read"-, „Update"- und „Delete"-Operationen (CRUD), Abfragen und Berechnungen durchzuführen.

Ein In-Memory Data-Grid speichert jegliche Informationen im Hauptspeicher und führt dort auch alle Operationen aus. Dies ist die Grundlage für hoch performantes, verteiltes Rechnen, da der Zugriff auf persistent gehaltene Strukturen vermieden wird (keine Objekterzeugung aus relationalen Strukturen, kein File-I/O etc.). Anwendungsobjekte sind auf den verschiedenen Knoten des Systems gemeinsam nutzbar. Andere Systeme (wie Middleware-Anwendungen) können auf diese Objekte transparent zugreifen – das auf den In-Memory Data-Grid zugreifende System muss also nicht wissen, auf welchem Knoten sich die Objekte befinden. Die Objekte können Informationen aus anderen Systemen (RDBMS, Filesystem etc.) widerspiegeln und, falls notwendig, in diese Systeme geschrieben oder von dort gelesen werden. Das Schreiben kann sowohl synchron als auch asynchron erfolgen.

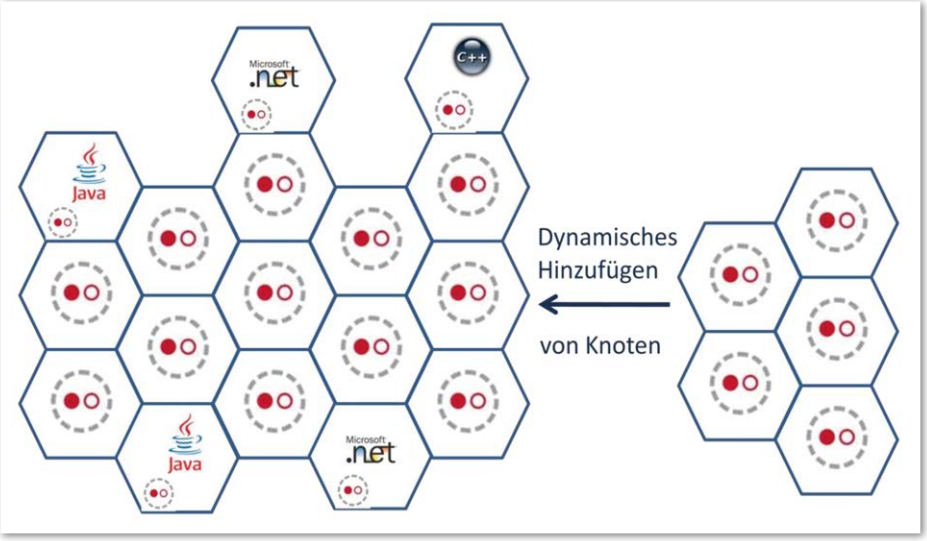


Abbildung 2: Verschiedene Coherence-Knoten formen ein Cluster, die Client-Knoten (Java, .NET, C++) sind selbst Teil des Clusters

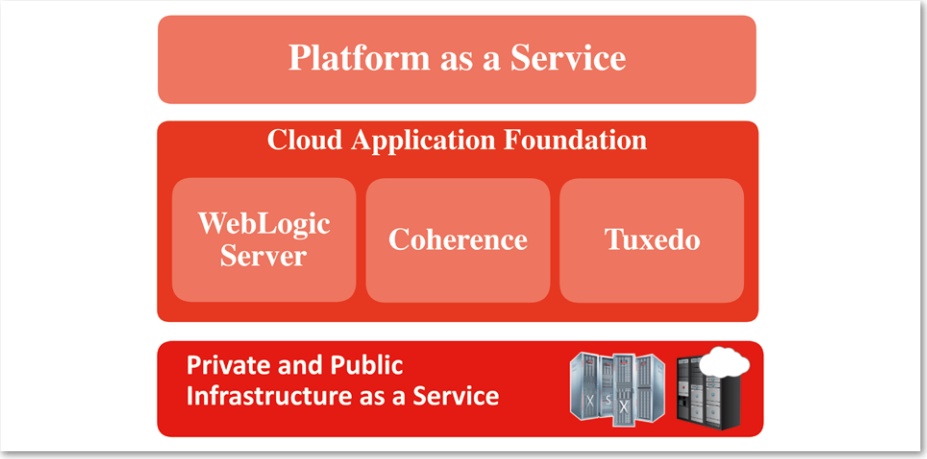


Abbildung 3: Coherence als Bestandteil der Oracle Cloud Application Foundation

Oracle Coherence

Ein Coherence-Knoten ist nichts anderes als ein Java-Prozess (JVM-Prozess), der mit den geforderten Coherence-Java-Bibliotheken und gewissen Coherence-Konfigurations-Informationen gestartet wurde. Lädt man Coherence vom Oracle Technology Network (OTN) herunter, erhält man ein 86 MB großes „.zip"-File, das im Wesentlichen eine Handvoll Java-Bibliotheken enthält. Diese haben keine Abhängigkeiten zu Drittpartei-Bibliotheken; es wird nur eine JDK-Installation vorausgesetzt.

Ein Coherence Cluster ist eine Menge von solchen Java-Prozessen (auf einem Rechner beziehungsweise verschiedenen physikalischen Rechnern) mit ähnlicher Konfiguration, die miteinander kommunizieren. Ein Client, geschrieben in Java, C++ oder auch .NET, kann sich mit dem Cluster verbinden, um per API gewünschte CRUD-Operationen, Abfragen und Berechnungen auf den Storage-Knoten (Knoten, auf denen sich die Anwendungsobjekte befinden) ausführen zu lassen (siehe Abbildung 2).

Ist zur Laufzeit zusätzliche Kapazität notwendig, werden einfach neue Storage-Knoten gestartet und dem Cluster hinzugefügt. Nach Spitzenzeiten können Knoten den Cluster auch wieder verlassen. In beiden Fällen sind die Anwendungsobjekte automatisch im Cluster rebalanciert – sie werden also auf den im Cluster befindlichen Knoten gleichmäßig neu aufgeteilt. Coherence sieht verschiedene Cache-Topologien vor, so zum Beispiel Local Cache, replizierter Cache, partitionierter Cache und Near Cache.

Es können verschiedene Backing-Map-Schemata zum eigentlichen Ablegen der Objekte im Cache zum Einsatz kommen. Je nach Anwendungsfall wird das gewünschte Szenario flexibel umgesetzt. Die Konfiguration erfolgt durch XML und/oder Java-System-Eigenschaften. Es gilt das Prinzip „Configuration by Default"; nur Verhalten, das vom Default abweicht, muss per Konfiguration überschrieben werden. Für das Anlegen und Manipulieren der XML-Konfigurationsdateien besteht grafische Unterstützung im Oracle Enterprise Pack for Eclipse ab Version 11.1.1.6.

Zudem kann Coherence out of the box für das Management von Http-Session-Zuständen in Application Servern mittels Coherence*Web oder für das Ersetzen der Standard-Caching-Mechanismen in verschiedenen Persistenz-Frameworks (Hibernate L2 Cache, EclipseLink JPA Shared Cache) genutzt werden. Der Zugriff auf einen Coherence Cache kann per API mit

Java, .NET oder auch C++ erfolgen. Das API umfasst unter anderem:

- Zugriffssteuerung auf Cache
- Unterstützung von speziellen, sehr effizienten Serialisierungsmechanismen für die Domain-Objekte
- Abfragen durch Filter und Value Extractors
- Erstellen von Indizes
- Benutzen von Aggregatoren
- Benutzen von Entry Processors zur effizienten parallelen Verarbeitung im Cache-Verbund
- Definition von Agenten und Benutzen des WorkManager-API
- Definition und Abgreifen von Data-Grid-Events
- Implementieren von CacheLoader und CacheStore

Oracle Coherence lässt sich mittels JMX administrieren und überwachen. Es gibt eine Integration in den Oracle Enterprise Manager, zudem stehen Monitoring-Werkzeuge anderer Hersteller zur Verfügung. Coherence bietet ferner eine komplette Integration im WebLogic- und GlassFish-Server, falls Coherence in Verbindung mit einem dieser Java-Applikationsserver laufen soll.

Coherence setzt auf eine komplett geclusterte Architektur auf. Analog zu einem Telefonkonferenz-Modell ist Coherence Consensus ein Vertrag zwischen einer Menge von Prozessen über die Mitgliedschaft innerhalb des Verbunds zu einer bestimmten Zeit. Dies ermöglicht transparentes, dynamisches und automatisches Failover/Failback von Services und Daten, zuverlässiges Partitionieren von Daten und Services sowie In-Memory-Funktionalitäten.

Die Software benutzt das Tangosol Coherence Management Protocol (TCMP), ein spezielles proprietäres Peer-to-Peer Unicast-basiertes Protokoll. Es arbeitet asynchron, die Kommunikation ist also nie blockiert, auch nicht wenn viele Threads eines Servers gleichzeitig kommunizieren, und gewährleistet damit eine echte Skalierbarkeit. Darüber hinaus bedeutet die Asynchronität auch, dass die Netzwerk-Latenz den Cluster Throughput nicht beeinträchtigt, obwohl sie die Geschwindigkeit von bestimmten Operationen beeinflussen kann. TCMP ist ein geclustertes, IP-basiertes Protokoll für Server Discovery, Cluster Management, Service Provisioning sowie Datenübertragung und kann optional Multicast unterstützen.

Coherence wird vor allem eingesetzt als Puffer für stark frequentierte Daten und zur Reduzierung der Latenz von Backend-Systemen, dadurch werden In-Memory Datenabfragen fünf bis zwanzig Mal schneller beantwortet als aus dem Backend. Es ist einer der wichtigsten Bestandteile der Oracle Cloud Foundation, einem Platform-as-a-Service-Angebot von Oracle, kann aber auch standalone betrieben werden (siehe Abbildung 3).

Grid Computing kombiniert Server-Virtualisierung und Clustering über den gesamten Stack und liefert damit eine dynamische, verteilte Infrastruktur als Basis für Cloud Computing. Cloud Computing teilt damit viele Eigenschaften und technologische Anforderungen mit Grid Computing.

avato information technology consulting

cloud@avato-consulting.com
exadata@avato-consulting.com
www.avato-consulting.com

Abbildung 4: RDBMS-Offload

Abbildung 5: Mainframe-Offload

Abbildung 6: Zustands-Informationen in Web-Anwendungen

Der Oracle-Schwerpunkt bei Grid-Computing-Funktionalitäten, wie dynamisches Resource Provisioning, dynamisches Resource Scheduling und hochautomatisiertes Management von Clustern und virtuellen Maschinen, passt exakt zu den Anforderungen aus dem Cloud Computing. Nachfolgend wird auf einige am häufigsten anzutreffende Coherence-Anwendungsfälle und ihren Nutzen eingegangen.

RDBMS-Offload

Wiederholtes Lesen von sich nicht verändernden Daten (wie Preislisten, Fahrplänen etc.) aus Datenbanken ist ineffizient, da massive Redundanz durch die Datenbank-Zugriffe auf dieselben Inhalte erzeugt wird und es somit zu korrespondierenden Objekten in allen JVMs führt, in denen Abfragen gestartet wurden *(siehe Abbildung 4)*.

Ebenso latenzträchtig ist die Zwischenspeicherung von großvolumigen, transient verwendeten Daten mit gewisser Lebensdauer in der Datenbank, bei der sich die Datenbank-Zugriffe als Flaschenhals erweisen. Auch die Bereitstellung transienter Daten (Daten und Berechnungen) über die Anwendungsgrenzen hinaus durch Zwischenspeicherung in der Datenbank führt in der Regel zu Engpässen.

Diese Engpässe werden durch einmaliges, hochfrequentes Laden der sich nicht verändernden Daten im Coherence Grid vermieden. Die Verwaltung transienter Daten findet nur im In-Memory Data-Grid statt. Eine Policy steuert die Lebensdauer von Objekten. Wenn diese Zeitlimits erreicht sind, werden die entsprechenden Objekte aus Coherence entfernt und entweder eliminiert oder in den Langzeitspeicher für eventuell später noch durchgeführte Langzeit-Aktionen verlagert. Coherence ermöglicht demzufolge die schnelle Daten-Analyse auf Basis transienter Daten. Der Einsatz von Coherence ermöglicht Einsparpotenziale durch Ressourcen-Schonung, deutlich höhere Performance und mehr Skalierbarkeit.

Mainframe-Offload

Der Zugriff und Operationen auf Mainframes (MOPS, million operations per second) sind teuer. Wiederholtes Lesen von sich nicht verändernden Daten (wie Kundenstammdaten) in Anwendungen, die Nutzung von transienten Daten über Anwendungsgrenzen hinaus (Daten und Berechnungen) und die Zwischenspeicherung von großvolumigen, transient verwendeten Daten mit gewisser Lebensdauer im Mainframe erzeugen viele MOPS und somit Kosten. Zudem dauert die Ergebnisbereitstellung bei einem synchronen Zugriff auf einen Mainframe zu lange *(siehe Abbildung 5)*.

Die Lösung dieser Latenzprobleme besteht im Herauslesen von benötigten Daten aus dem Mainframe und aus der einmaligen Ablage in Coherence: Die Kalkulation findet nun im Java-Adressraum statt, anschließend kann das Ergebnis zurückgeschrieben werden. Der Zugriff auf bestimmte Anwendungsobjekte erfolgt über den Coherence Grid. Zugriffszeiten lassen sich somit optimieren. Die Skalierbarkeit erfolgt durch Hinzunahme von kostengünstiger Commodity-Hardware. Somit können transiente Daten in Coherence über Anwendungsgrenzen hinaus bereitgestellt werden. Diese Ressourcen-Schonung führt zu deutlichen Mainframe-Kosteneinsparungen und schneller Ergebnisbereitstellung bei synchronen Zugriffen.

Zustands-Informationen in Web-Anwendungen

Bei der „Zustandslose"-Programmierung ist der Zustand innerhalb von Request-/Response-Paaren in der Datenbank gespeichert. Der Zustand muss also immer wieder geschrieben und geladen werden. Bei Web-Anwendungen mit Megabyte-großen Zustands-Informationen oder großem Volu-

men statischer Information (wie Benutzerprofile) stellt dann das Speichern in der Datenbank einen Engpass dar. Darüber hinaus stellt sich die Frage, was mit der Anwendung geschieht, wenn die Datenbank nicht verfügbar ist. In solch einem Fall ist die Anwendung dann auch nicht verfügbar *(siehe Abbildung 6)*.

Die Daten und der Application Server sind in der JVM kollokiert. Dies bedeutet, dass viele beziehungsweise große Anwendungsdaten zu Garbage-Collection-Herausforderungen in JVMs führen, was Performance-Einbußen durch große Heaps zur Folge hat.

Betrachtet man das Rolling Upgrade von Produktionsservern (Infrastruktursoftware, Anwendungssoftware), also die Frage, was mit Daten geschieht, die nur In-Memory gehalten und von Anwendungen auf Application Servern verwendet werden (wie Benutzerprofile, die kollokiert mit der entsprechenden Anwendung im Application Server in einer JVM vorliegen). Auch hier gilt, dass das Herunterfahren eines oder mehrerer Application Server nicht ohne Weiteres möglich ist, da Daten quasi verloren gehen können.

Die Lösung für die beschriebenen Probleme besteht in der Einführung einer separaten Datenschicht durch ein Coherence Cluster. Cache Loader und Cache Store werden zum Laden und Speichern von zu persistenzierenden Zustands-Informationen (zum Beispiel Benutzerprofile) eingesetzt. Damit können Lastspitzen ausgeglichen, Service-Level-Agreements wieder eingehalten, die Applikationsschicht-Kapazität erhöht und Backend-Last stark reduziert werden. Dieser Anwendungsfall erzielt neben einer Ressourcen-Schonung auch deutlich bessere Ergebnisse im Hinblick auf Skalierbarkeit, Verfügbarkeit und Zuverlässigkeit.

Http-Session Management

Werden Http-Session-Zustände von verschiedenen Web-Anwendungen gleichzeitig genutzt, dann führen viele beziehungsweise große Http-Sessions zu Garbage-Collection-Herausforderungen. Für die Ablage dieser Information in einer Datenbank gilt analog zum vorhergehenden Beispiel: Ist die Datenbank nicht verfügbar, dann ist die Anwendung auch nicht verfügbar. Zudem wird die Datenbank bei Skalierbarkeitsproblemen schnell zum Flaschenhals und vorhandene In-Memory-Replikationsmechanismen von Anwendungsservern sind zum Teil zu ineffizient. Auch das Rolling Upgrade der Produktionsserver

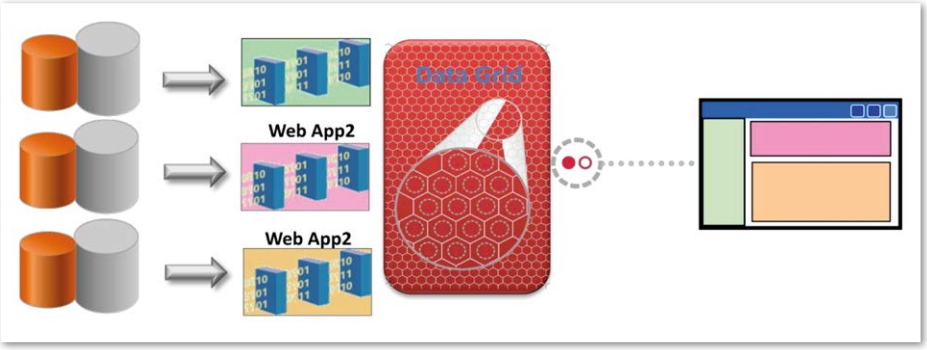


Abbildung 7: Http-Session-Management (unter anderem über Anwendungsgrenzen hinaus)

Abbildung 8: Objektoperabilität (Java, C++, .Net)

gestaltet sich schwierig, wie bereits zuvor beschrieben *(siehe Abbildung 7)*.

Eine separate Datenschicht durch Coherence Cluster mit Coherence*Web als Out-of-the-Box-Lösung für verschiedene Anwendungsserver löst hierbei den Engpass. Coherence*Web ist ein Http-Session-Management-Modul für das Managen/Verwalten des Session State beziehungsweise des Http-Session-Zustands in geclusterten Umgebungen. Coherence*Web unterstützt den WebLogic Server, GlassFish und alle andere Mainstream Application Server.

Bei Performance-Problemen ist es die ideale Lösung für sehr umfangreiche, webbasierte Umgebungen mit einer Composite Application (zusammengesetzte Anwendung), die von unterschiedlichen darunterliegenden Inhalten abhängt. Diese Lösung führt zu einer massiven Verbesserung der Gesamt-Performance und Skalierbarkeit der Infrastruktur, zu höherer Site-Performance durch Session Sharing und zu guter Benutzererfahrung durch stabile Sessions.

Objekt-Operabilität

In diesem Anwendungsfall geht es um die gemeinsame Benutzung von Objekten aus verschiedenen Programmiersprachen heraus. Coherence agiert hierbei als eine Art „Objektmediator" und ersetzt an dieser Stelle langsamere, Webservice-basierte Serialisierungs-/Deserialisierungs- und Standardserialisierungs-Mechanismen. Dieser Ansatz erzielt sehr hohe Verfügbarkeit und bietet eine schnelle, zuverlässige und hochperformante Bereitstellung von High-End-Produktions-Umgebungen *(siehe Abbildung 8)*.

Gründe für Oracle Coherence

Mit Coherence können Unternehmen die Skalierung erfolgskritischer Anwendungen voraussehen, sodass ein schneller Zugriff auf häufig benötigte Daten gegeben ist. Anwendungen lassen sich mithilfe von Coherence linear und dynamisch skalieren, was eine bessere Vorhersehbarkeit der Kosten und eine bessere Ressourcen-Nutzung ermöglicht.

Zudem ist eine signifikante Steigerung der Transaktionsraten bei Verkürzung der Antwortzeiten von Anwendungen zu verzeichnen. Coherence sorgt selbst bei einem Serverausfall für eine kontinuierliche Datenverfügbarkeit und Transaktionsintegrität. Organisationen kommen ebenfalls die extremen Transaction-Processing-Fähigkeiten zugute: Durch die Verteilung der Daten, kombiniert mit der Leistungsstärke des Grid Com-

puting, ergibt sich eine schnelle und zuverlässige High-End-Transaktionsverarbeitung.

Fazit

Zusammenfassend lässt sich sagen:

- Coherence ist eine elegante und praktikable Lösung für Caching
- Coherence ist keine klassische persistierende Datenbank, sondern ein Zwischenspeicher (Cache)
- Es gibt umfangreiche Möglichkeiten, an Coherence anzudocken
- Coherence entlastet das Backend und beschleunigt die Zugriffe
- Dies führt zu weniger Wartezeit durch höhere Kapazitäten

Gabriele Jäger
gabriele.jaeger@oracle.com

Michael Fuhr
michael.fuhr@oracle.com

OWB – Wie finde ich den richtigen Nachfolger?

Michael Klose, CGI Deutschland Ltd. & Co. KG

Der Warehouse Builder wird von Oracle nicht weiterentwickelt und der Standard Support endet in naher Zukunft. Gerade Kunden, die die Datenbank-Version 12*c* nicht installieren wollen, sind gezwungen, sich nach Alternativen umzusehen.

Oracle stellt mit dem Data Integrator zwar einen würdigen Nachfolger bereit; dieser ist allerdings mit zusätzlichen Lizenzkosten verbunden. Der Artikel zeigt verschiedene Möglichkeiten des Umstiegs auf ein anderes ETL-Tool auf. Sie werden anhand von Praxiserfahrungen aus einer ETL-Toolauswahl dargestellt. Darüber hinaus werden die Vor- und Nachteile der verschiedenen Tools sowie grundsätzliche Veränderungen in der Entwicklungstätigkeit und Architektur aufgezeigt. Die Schwerpunkte liegen bei Oracle Data Integrator, Informatica Powercenter, Talend, PL/SQL und einer Empfehlung für die Vorgehensweise bei der Tool-Auswahl.

Grundsätzliches

Bei den ETL-Tools gibt es durch die große Anzahl von Entwicklungswerkzeugen gerade aus Architektur-Sicht unterschiedliche Ansätze der Hersteller. Grundsätzlich unterscheidet man zwischen ETL (Extract – Transform – Load) und ELT (Extract – Load – Transform). Beim ETL-Prozess werden die Daten aus den Quellsystemen beispielsweise als Flat Files extrahiert, im Anschluss in einer ETL-Engine transformiert, um letztendlich final in die Zieldatenbank (Data Warehouse) geladen zu werden. Im Gegensatz dazu laden ELT-basierte Werkzeuge die Daten zuerst in die Zieldatenbank, um dort dann die Transformationen durchzuführen. Ein weiterer Unterschied liegt in der Programmiersprache des generierten Codes der Transformationsprozesse.

OWB – Stärken und Schwächen

Gerade für die in der Oracle-Programmierung kompetenten Entwickler stellt die Generierung von PL/SQL-Code eine klare Stärke dar. Der Code ist komplett nachvollziehbar und für die jeweilige Datenbank-Version optimiert. Nahezu alle Datenbank-Features können verwendet werden und die Ausführung erfolgt direkt auf der Datenbank (Push-Down). Die hohe Integration in die Oracle-Produktpalette und die einfache Installation innerhalb der Datenbank bedingen keine zusätzlichen Backup-Aufwände. Das vollständige, teilweise sogar Produkt-übergreifende Metadaten-Repository (OBI) ermöglicht detaillierte Data-Lineage-Analysen und unterstützt das Deployment auf andere Umgebungen. Der Funktionsumfang im ETL-Bereich umfasst alle wichtigen Komponenten wie Mappings, Workflows, Fehlerprotokollierung etc. Eine nicht zu vernachlässigende Stärke ist auch, dass der Oracle Warehouse

Builder in der Standardvariante ein Bestandteil der Datenbank-Lizenz war.

Leider bestehen neben den vielen Vorteilen auch gravierende funktionale Mängel. Gerade in sehr heterogenen System-Landschaften geriet der OWB schnell an seine Grenzen. Für den Datenabzug aus Nicht-Oracle-Datenbanken konnten noch Gateways oder Heterogeneous Services eingesetzt werden, allerdings war als Ziel-Datenbank nur Oracle sinnvoll einsetzbar. Ebenso kann nur ELT-basiert gearbeitet werden, was immer eine Integration der Daten in die Oracle-Datenbank bedingte. Sicherlich traf das den Großteil der Anwendungsfälle für ein Data Warehouse, allerdings werden ETL-Tools heutzutage vielschichtiger eingesetzt.

Mankos gibt es auch im Bereich des generierten PL/SQL-Codes. Dieser war zwar lesbar, es ist jedoch dringend davon abzuraten, den Code nach der Generierung zu verändern, da diese Veränderungen bei der nächsten Generierung verloren wären.

Zusätzlich zur lizenzfreien Variante konnte man sich weitere Funktionalitäten innerhalb einer Enterprise Version lizenzieren. Funktionen wie Lade-Reihenfolge für Tabellen, Schleifen im Workflow, Variablen-Übergabe oder das automatisierte Befüllen von Slowly Changing Dimensions sind extra zu lizenzieren oder mit mehr oder weniger aufwändigen Workarounds zu lösen. Das Thema „Versionierung" war jahrelang ein Enhancement Request bei Oracle, wurde jedoch nie realisiert.

ETL-Tools und wichtige Funktionalitäten im Überblick

Für eine Tool-Auswahl wird auf jeden Fall die Heranziehung von Analysten-Bewertungen (Gartner, Forrester, Barc) als Basis empfohlen, da eine Betrachtung aller marktrelevanten ETL-Tools einen sehr hohen Aufwand bedeutet. Will man die Auswahl von Beginn an stark einschränken, kommen sicherlich der Oracle Data Integrator als Produkt-Nachfolger, Informatica Powercenter als Marktführer und möglicherweise Talend als Open-Source ETL-Tool infrage.

Bezüglich der Anforderungen an moderne ETL-Tools müssen diese die klassischen Funktionalitäten enthalten. Eine große Funktionsbibliothek, Workflow-Steuerung, Wiederverwendbarkeit von Teilkomponenten, integrierte Versionsverwaltung, Deployment-Unterstützung, Metadaten-gestützte Entwicklung, Protokollierung und natürlich eine intuitive Bedienung sind selbstverständliche Bestandteile.

Im Zeitalter von Big Data und Self-Service-BI reichen diese Funktionen allein nicht aus. Hinzu kommen Anforderungen an die Integration unstrukturierter Daten, (Near-)Real-Time-Fähigkeiten, Anbindung von heterogenen Umgebungen, Change-Data-Capture-Mechanismen auf den Quellsystemen, Generierung von Mappings und Workflows, Skalierbarkeit und vor allem, vom Gesetzgeber teilweise sogar vorgeschrieben, die komplette Nachvollziehbarkeit von Datenveränderungen.

Oracle Data Integrator

Der Oracle Data Integrator (ODI) wird von Oracle als Nachfolger des Warehouse Builder (OWB) positioniert. In der aktuellen Version ist ein Migrationstool mitgeliefert, um je nach Komplexität der Mappings 70 bis 80 Prozent automatisiert zu migrieren. Der ODI ist ein ELT-Werkzeug und verwendet die Datenbank als ETL-Engine. ODI-Neueinsteiger bezeichnen den ODI-Agent gerne als „ETL-Engine". Dabei handelt es sich um einen Java-Prozess, der für die Ausführung der generierten Statements auf den Datenbanken zuständig ist, selbst aber keine Transformationen durchführt.

Eine weitere Kern-Komponente sind die Knowledge-Module, die für die verschiedenen Datenbanken entsprechende SQL-Statements generieren, um die Transformations- und Ladeprozesse abzubilden. Eine eigene ETL-Engine besitzt ODI hingegen nicht und kann somit nur die Funktionsbibliotheken der verwendeten Datenbanken nutzen. Der Benutzer kann die Knowledge-Module an die entsprechenden Anforderungen anpassen. Durch die Generierung von SQL-Statements bleibt der Code lesbar und kann beispielsweise für Tuning-Zwecke gut nachvollzogen werden.

Weitere wichtige Bestandteile sind Workflow und Load Plans. Diese Komponenten realisieren die Lade- und Ablaufsteuerung. Sie sind im Vergleich zum OWB wesentlich mächtiger. Dies zeigt sich vor allem im Bereich der Wiederaufsetzbarkeit von Ladeprozessen nach Abbrüchen.

Auch in der Versionierung und Paketierung für das Deployment gibt es nennenswerte Verbesserungen. Im ODI besteht die Möglichkeit, jede einzelne Komponente zu versionieren und in Deployment-Pakete zusammenzufassen. Der Funktionsumfang von Versionierungswerkzeugen wie Subversion wird damit zwar nicht erreicht, allerdings ist ein sinnvolles Arbeiten mit Versionierung möglich.

Im Gegensatz zum OWB lässt sich eine Vielzahl von Datenbanken als Quell- und Zielsystem anbinden. Die Verbindung zur Datenbank wird über den ODI-Agent und entsprechende JDBC-Treiber hergestellt. Für nahezu alle namhaften Datenbank-Hersteller sind umfangreiche Best-Practice-Knowledge-Module enthalten. Dies hat zur Folge, dass der ODI auch in sehr heterogenen System-Landschaften im Gegensatz zum OWB sehr gut eingesetzt werden kann.

Informatica Powercenter

Informatica Powercenter (IPC) gehört zur Kategorie der ETL-Tools und ist sicher eines der mächtigsten und umfangreichsten Werkzeuge im ETL-Bereich, was Analysten regelmäßig bestätigen. Der Einsatz einer eigenen ETL-Engine führt zu einer Unabhängigkeit gegenüber der Funktionsbibliothek der Datenbank. Allerdings benötigt diese Engine einen (im Normalfall) eigenständigen Server zur Ausführung der Transformationsprozesse. Dadurch ist es möglich, Daten aus verschiedenen Quellen zu extrahieren, in die ETL-Engine zu laden, Transformationen, Joins und Aggregationen durchzuführen sowie die Daten final in die Zieldatenbank zu schreiben. In der klassischen Datawarehouse-Layer-Architektur (Stage/EDWH/Data Marts) ist es in diesem Fall allerdings auch notwendig, die Daten beim Transport durch die verschiedenen Layer aus dem Warehouse zu extrahieren und nach den Transformationen wieder in dieses zu laden (etwa EDWH nach Data Mart).

Informatica bietet für Powercenter zwar eine sogenannte „Push-Down-Funktion" an, um möglichst viele Tätigkeiten auf die Datenbank auszulagern, allerdings kann dort nicht alles ausgeführt werden. Diese Funktionalität war in der Vergangenheit mit zusätzlichen Kosten verbunden und wurde deswegen vom Kunden häufig nicht eingesetzt. Um beispielsweise bei Delta-Verarbeitung die Performance beim Laden zwischen den DWH-Layern zu verbessern, werden häufig die „Source Qualifier" (ein Informatica-Objekt, welches das SELECT-Statement enthält) manuell

mit Joins und Filtern angepasst. Dies führt zu einem Verlust der Data-Lineage-Funktionalität in diesem Bereich, ist aber häufig die einzige Optimierungsmöglichkeit.

IPC generiert keinen lesbaren Code, der optimiert und visualisiert werden kann, und stellt somit eine „Black Box" dar. Die Oberfläche selbst erinnert stark an den OWB und aus der Erfahrung zeigt sich, dass Mitarbeiter mit OWB-Kenntnissen sehr schnell beginnen können, Mappings in IPC zu entwickeln.

Zur Ausführung und Prozess-Steuerung ist eine Workflow-Komponente verfügbar. Ohne ein externes Workflow/Scheduling-Tool wie CTRL-M ist die Vorgehensweise gerade beim Verschachteln von Workflows eher schlecht gelöst. Die kostenpflichtige Option „Team Based Development" ermöglicht die Versionierung sowie ein Check-In/Out ähnlich Subversion. IPC besitzt viele (lizenzpflichtige) Konnektoren zu Datenbanken, die bei den großen Datenbank-Herstellern sogar ein Change Data Capture für den Datenabzug unterstützen.

Talend Enterprise Data Integration

Talend bietet abhängig von den Anforderungen verschiedene Ausbaustufen seiner Data Integration Suite an. Um das Leistungsspektrum des OWB abzubilden, wird hier die „Enterprise Data Integration"-Variante betrachtet. Talend gehört ebenso wie Informatica zur Kategorie der ETL-Tools. Als ETL-Engine dient hier letztendlich die Java-Runtime-Engine.

Mappings werden wie bei den vorherigen Tools grafisch entwickelt und ausführbarer lesbarer Java-Code generiert. Die Anbindung von Quell- und Zielsystemen erfolgt über JDBC-Treiber, die fast alle Datenbank-Herstellern anbieten. Dadurch eignet sich Talend für sehr heterogene Quell- und Zielsysteme.

Die Daten müssen zur Durchführung von Transformationen aus dem Data Warehouse extrahiert und in der Java-Runtime-Engine verarbeitet werden. Hierfür ist ein zusätzlicher ETL-Server empfehlenswert, der vor allem in den Bereichen „CPU" und „Memory" nicht zu klein ausgestattet sein sollte. Der Push-Down von Datenbank-Abfragen ist möglich, allerdings wird dies nicht für alle Datenbanken angeboten.

Die mitgelieferte Funktionsbibliothek, die out oft he box in einer grafischen Entwicklung verwendet werden kann, ist nicht sehr umfangreich. Dies hat zur Folge, dass die Transformationen in Java programmiert werden müssen. Hier steht wiederum der komplette Funktionsumfang von Java zur Verfügung, sodass auch weitere Libraries eingebunden werden können. Da es sich um reinen Java-Code handelt, bettet sich Talend optimal in Versionierungssoftware wie Subversion ein.

Oracle PL/SQL

Sicherlich nimmt Oracle PL/SQL für den ETL-Prozess bei einer Tool-Betrachtung eine Nebenrolle ein, nichtsdestotrotz haben viele Unternehmen mit Oracle-Datenbanken diese Variante erfolgreich im Einsatz. Im Vergleich zu ETL-Tools wie Informatica oder Talend sind PL/SQL und SQL wesentlich näher am Ergebnis der Mapping-Generierung des OWB. Die großen Vorteile der Tools sind grafische Entwicklung, Metadaten-Repository, automatische Protokollierung und Data Lineage. Auf den ersten Blick stellen diese nicht zu vernachlässigende Komponenten dar, allerdings können einige auf andere Weise in einem Oracle-Datenbank- und im PL/SQL-Umfeld alternativ abgebildet werden.

Die grafische Entwicklungsoberfläche ist sicherlich nicht durch PL/SQL zu ersetzen. Eine Variante, um SQL-Statements zu generieren, stellt beispielsweise der Query Builder dar. Allerdings werden nur wenige Entwickler diese Alternative einsetzen wollen.

Beim Metadaten-Repository und der Data Lineage ist die Diskrepanz lange nicht so groß. Letztendlich stellt das Data Dictionary der Datenbank eine Art „Metadaten Repository" dar. Auf Datenbankobjekt-Ebene können hier viele Informationen hinterlegt werden. Für die Transformations-Metadaten bietet es sich an, ein eigenes Mini-Repository anzulegen. Ein solches steht in der Oracle-DWH-Community kostenfrei zur Verfügung. Damit lässt sich eine Data-Lineage-Analyse auf einfachem Weg aufsetzen; andernfalls bietet die Datenbank selbst Möglichkeiten, die Abhängigkeiten zwischen verschiedenen Objekten auszuwerten.

Ein wichtiger Bestandteil jedes ETL-Prozesses ist die Protokollierung der Laufzeit-Informationen. In diesem Bereich existieren fertige Logging-Frameworks, die teilweise kostenfrei sind. Alternativ dazu können eigene Frameworks in PL/SQL entwickelt und auf einfachem Wege eingebunden werden.

Eine weitere wichtige Komponente stellt die Ablaufsteuerung dar. Gerade in Anbetracht der Parallelisierung von Ladeprozessen sind die Möglichkeiten von PL/SQL doch sehr begrenzt. Für diesen Zweck kann der Oracle Enterprise Manager verwendet werden. Dort lassen sich sowohl Datenbank- als auch Betriebssystem-Jobs definieren. Eine Parallelisierung der Prozesse und die Abbildung von Abhängigkeiten unter den Prozessen ist möglich.

Viele OWB-Entwickler verfügen über PL/SQL-Kenntnisse und somit wäre ein einfacher Umstieg − was die Entwicklung betrifft − möglich. Bei bestehenden OWB-Implementierungen könnten die „Insert As Select"-Statements über OMB+ als Script generiert werden und somit die Basis für die Migration darstellen. Selbst Row-Based-Mappings können überführt werden. Voraussetzung dafür ist allerdings, dass man die OWB-Komponenten (Logging, Initialisierung etc.) vorher entfernt.

Die ETL-Tool-Auswahl

Der wichtigste Punkt, um das richtige ETL-Tool beziehungsweise einen OWB-Nachfolger auszuwählen, ist eine genaue und möglichst detaillierte Anforderungserfassung. Diese sollte die häufig genutzten OWB-Funktionalitäten identifizieren, vermisste sowie benötigte Funktionen enthalten sowie um zusätzliche, möglicherweise bereits bekannte Funktionalitäten für die Zukunft (Big Data) ergänzt werden.

Wer wählt das Tool aus? Diese Frage ist sehr wichtig, da Entwickler, Architekten und auch der Betrieb häufig unterschiedliche Betrachtungen, gerade im Bereich der Gewichtung von Funktionalitäten, haben. Sobald das Team zusammengestellt und die Anforderungen erfasst sind, erfolgt die Erstellung des Anforderungskatalogs, der dann an die verschiedenen Anbieter zur Beantwortung versendet werden kann. Wichtig sind hier die Benennung von K.O.-Kriterien (wie Betriebssystem-Unterstützung) sowie die Festlegung der Gewichtung der einzelnen Anforderungen nach ihrer Wichtigkeit. Dies sollte unbedingt vor der Auswertung erfolgen, um den Prozess zu beschleunigen.

Liegen die Antworten der möglichen Hersteller aus der Long List vor, kann über den Katalog die Reduzierung der möglichen Kandidaten erfolgen. Im weiteren Verlauf lädt man wenige Kandidaten für Präsentationstermine (Short List) ein. Um eine Vergleichbarkeit zu gewährleisten, sollte den Herstellern ein fester Fahrplan für die Präsentation der angeforderten

Features zugeteilt werden. Anhand der Ergebnisse reduziert sich die Anzahl der Hersteller weiter, sodass die finalen Kandidaten zu einem intensiven, mehrtägigen „Proof of Technology"-Workshop eingeladen werden können. In jedem dieser Schritte gilt die Prämisse: „Keine Angst, Kandidaten auszusortieren."

Im „Proof of Technology"-Workshop werden final die gestellten Anforderungen nochmals live geprüft. Es empfiehlt sich, während der gesamten Dauer Spezialisten aus dem eigenen Unternehmen den Workshop begleiten zu lassen. Dies stellt sicher, dass auch im Nachgang bereits das Wissen darüber existiert, wie die gestellten Anforderungen umgesetzt werden können.

Fazit

Der Artikel zeigt anhand von Beispielen die verschiedenen Möglichkeiten des Umstiegs auf ein anderes ETL-Tool. Da jedes Tool seine Vor- und Nachteile hat, ist es umso wichtiger, die Anforderungen, individuell auf das eigene Umfeld betrachtet, festzulegen und in einem Tool-Auswahlprozess zu bewerten.

Michael Klose
michael.klose@cgi.com

OWB ohne OWB: Wie rette ich meine ETL-Sourcen nach 12c R2?

Sven Bosinger, its-people GmbH

Oracle hat angekündigt, den Oracle Warehouse Builder (OWB) ab der Datenbank-Version 12*c* R2 nicht mehr zu unterstützen. Dies hat zur Folge, dass alle ETL-Prozesse, die mit dem OWB erstellt wurden, in ein neues Werkzeug migriert werden müssen. Oracle empfiehlt hier den Oracle Data Integrator (ODI). Doch ist eine Migration immer notwendig?

Der Artikel stellt eine Möglichkeit vor, die mit dem OWB erzeugten ETLs ohne den OWB und damit potenziell auch unter 12*c* R2 weiter zu nutzen. Dazu wurde ein ETL-Workframe geschaffen, der die fehlenden OWB-Runtime-Komponenten ersetzt und so eine Lauffähigkeit der Sourcen ohne Installation des OWB ermöglicht.

Motivation

Im Januar 2010 und später noch einmal konkretisiert im Oktober 2013 hat Oracle folgendes Statement of Direction veröffentlicht: „No major enhancements are planned for Oracle Warehouse Builder beyond the OWB 11.2 release. OWB 11.2 continues to be available and supported by Oracle, and patches and bug fixes will continue to be offered at regular intervals. Oracle will continue to support OWB 11.2 for the full lifetime of Oracle Database 11*g* in accordance with Oracle's Lifetime Support Policies for Oracle Database releases. Future database releases beyond Oracle Database 12*c* Release 1 will not be certified with OWB 11.2." Daraus ergeben sich für die Nutzer des OWB folgende Konsequenzen:

- Wer OWB einsetzt, muss sich zeitnah Gedanken über eine Migrationsstrategie machen.
- Die Datenbank 12*c* R2 ist für Ende 2015 / Anfang 2016 angekündigt. Wer diese einsetzen möchte, muss zuvor alle OWB-Mappings migrieren.
- Der Premium-Support für 12*c* R1 endet im Juli 2018, der Extended Support im Juli 2021. Ein Betrieb über den Juli 2018 hinaus zieht zusätzliche Support-Kosten nach sich. Nach dem Juli 2021 ist ein Support in der Regel nicht mehr möglich.
- Das von Oracle präferierte Nachfolgeprodukt des OWB ist der Oracle Data Integrator (ODI). Eine Migration dorthin ist aufwändig und zieht Kosten nach sich. Der ODI deckt nicht alle Funktionalitäten des OWB ab. Es ist davon auszugehen, dass bei einer durchschnittlichen DWH-Datenbewirtschaftung maximal 90 Prozent der ETL-Sourcen „1:1" migriert werden können. Für die restlichen 10 Prozent ist eine zum Teil aufwändige Neuentwicklung notwendig (Erfahrungswerte).
- Der ODI ist nicht wie der OWB (Standard Edition) in den Lizenzkosten der Datenbank enthalten, sondern muss zusätzlich lizenziert werden.

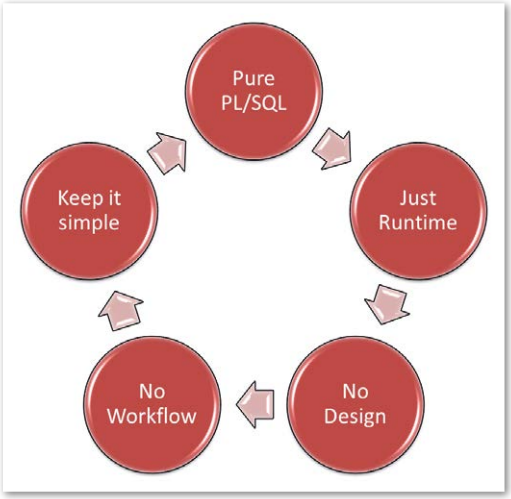


Abbildung 1: Design-Prinzipien des ETL-Workframes

- Da eine „1:1"-Migration der ETL-Sourcen hin zum ODI nicht immer möglich ist, muss für die Änderungen eine gegebenenfalls aufwändige Testphase eingeplant werden.

Die genannten Punkte bedeuten in Summe, dass heutige OWB-Nutzer massiven Handlungsdruck haben und damit einhergehende Migrationskosten für die nächsten Jahre planen müssen. Betrachtet man diese durch Oracle gesetzten Rahmenbedingungen, lohnt es sich, in diesem Zusammenhang etwas genauer hinzuschauen, worum es sich beim OWB technologisch gesehen eigentlich handelt.

OWB ist ein ETL-Werkzeug, das ausschließlich Tabellen in einer Oracle-Datenbank befüllen kann. Es besteht aus zwei Modulen – der Design- und der Runtime-Komponente. In der Design-Komponente wird mithilfe einer grafischen Oberfläche ein Mapping modelliert, also ein Datentransfer zwischen einer Quell- und einer Ziel-Tabelle. Um einen lauffähigen Code zu erhalten, muss man das Mapping nach abgeschlossener Design-Phase generieren. Das Ergebnis ist ein PL/SQL-Package, das in der Lage ist, den Datentransfer durchzuführen. Es wird dann über die Runtime-Komponente aufgerufen. Diese stellt zudem ein Repository zur Verfügung, in dem die Laufzeitinformationen der Packages gespeichert sind.

Es gibt keinerlei Anzeichen dafür, dass PL/SQL über 12*c* R2 hinaus nicht mehr zur Verfügung steht. Somit müsste theoretisch ein einmal generiertes Package auch weiterhin in einer Umgebung, die PL/SQL unterstützt, lauffähig sein. Das Runtime-Repository, das OWB bereitstellt, müsste durch eine Eigenentwicklung substituierbar sein. Der im Weiteren vorgestellte ETL-Workframe basiert auf diesen Annahmen und stellt eine technische Umsetzung dieser Ideen dar. Es handelt sich hierbei nicht um ein Produkt, sondern um einen Consulting-Softframe.

Der ETL-Workframe

Der durch die Firma its-people entwickelte ETL-Workframe stellt eine Möglichkeit dar, mit dem OWB generierte Mappings in einer OWB-freien Umgebung lauffähig zu halten. Mit diesem Ansatz werden folgende Ziele verfolgt:

- Der zeitliche Druck, schon bei der Migration auf Oracle Database 12*c* R2 alle ETLs, die mit dem OWB erzeugt worden sind, auf ein neues Produkt umgestellt zu haben, soll entschärft werden.
- Fehlende Funktionen (wie Premap- und Postmap-Trigger) des neuen ETL-Werkzeugs (etwa ODI) können durch partiellen Einsatz der alten OWB-Mappings ausgeglichen werden.
- Die Migrationskosten sinken, indem komplett vorhandene ETL-Strecken (Mappings) in das neue zu beschaffende ETL-Werkzeug integriert werden, sofern dieses in der Lage ist, PL/SQL-Blöcke zu starten.
- Aufwändige Tests der „1:1" übernommenen ETL-Sourcen entfallen, da diese beim Einsatz des ETL-Workframes nicht verändert und somit nicht erneut getestet werden müssen.

Der ETL-Workfarme selbst ist eine reine PL/SQL-Entwicklung. Somit ist eine prinzipielle Unabhängigkeit gegenüber dezidierten Datenbank-Releases gegeben. Der ETL-Workframe und die durch ihn gekapselten Mappings können also auch jenseits von 12*c* R1 betrieben werden *(siehe Abbildung 1)*.

Der ETL-Workframe deckt lediglich die Runtime-Komponente des OWB ab, da es hier nicht um die Entwicklung eines neuen ETL-Werkzeugs geht, sondern nur um die Lauffähigkeit der generierten Mappings. Ebenso werden hierbei Workflow-Themen nicht betrachtet.

Anwendungsfall

Zur Verdeutlichung der Funktionsweise des ETL-Workframes wird im Folgenden auf ein einfaches Beispiel anhand eines OWB-Mappings mit Namen „MAP_KUNDEN" Bezug genommen *(siehe Abbildung 2)*. Die Inhalte der Stage-Tabelle „STG_KUNDEN" werden in die Tabelle „DWH_KUNDE" integriert, dabei ermittelt eine PL/SQL-Funktion eine Bonitäts-Kennzahl und formatiert über eine Expression diverse Spalten um. Durch einen Premap-Trigger wird in einer Protokoll-Tabelle ein Eintrag gesetzt. Zusätzlich erzeugt eine Sequenz ein Surrogate-Key.

Betrachtet man den PL/SQL-Code des generierten Mappings, kann man leicht die Schnittstellen zum OWB-Runtime-Repository erkennen. Durch das Mapping werden drei durch den OWB bereitgestellte Packages aufgerufen:

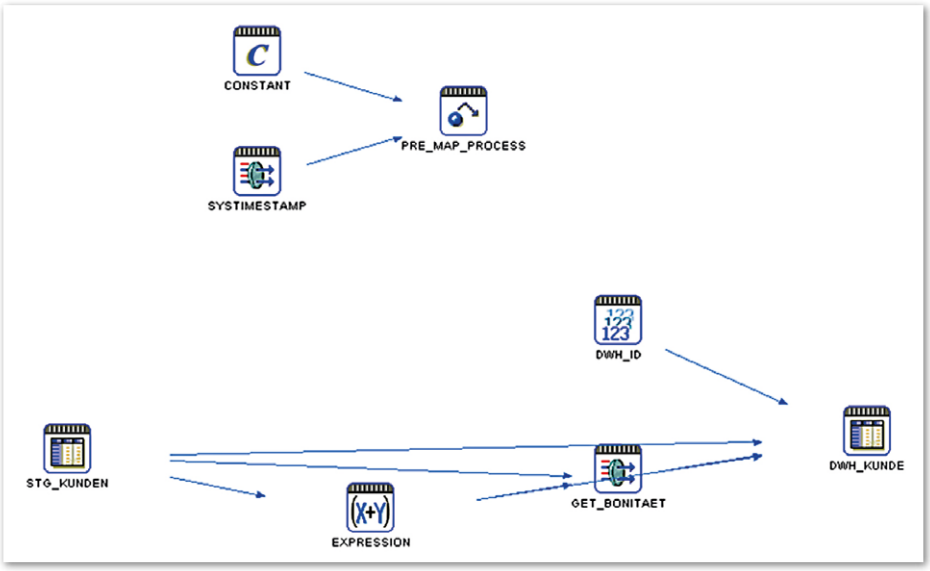


Abbildung 2: Einfaches OWB-Mapping

- WB_RT_MAPAUDIT
- WB_RT_MAPAUDIT_UTIL
- WB_RT_MAPAUDIT_UTIL_INVOKER

Alle drei Packages befinden sich im Schema „OWBSYS“. Da eine Veränderung des Source-Codes der Mappings nicht vorgenommen werden soll, ist es notwendig, in der OWB-freien Datenbank ein Schema „OWBSYS“ anzulegen und dort die drei genannten Packages durch eine Eigenentwicklung nachzubilden. Dazu gilt es zu betrachten, welche Funktionalitäten diese bereitstellen.

Im Package „WB_RT_MAPAUDIT“ sind alle Protokollierungen hinsichtlich des Runtime-Repository gebündelt. Insbesondere werden folgende Prozeduren/Funktionen benötigt:

- AUDITD_BEGIN
- AUDIT_BEGIN
- REGISTER_AUDIT_DETAIL_TYPE
- ERROR
- AUDITD_END
- AUDIT_FAIL
- ERROR_SOURCE
- REGISTER_FEEDBACK
- REGISTER_FEEDBACK_BULK
- AUDITD_PROGRESS
- LOG_FILE

Die Runtime-Repository-Tabellen des OWB sind ein einfaches Datenmodell, das über eine View-Schicht angesprochen wird. Der Aufbau kann der OWB-Dokumentation entnommen werden. Baut man das Package „WB_RT_MAPAUDIT“ nach, indem man seine eigenen Runtime-Tabellen entwirft und die entsprechenden OWBSYS-Calls auf das eigene Repository umleitet, so kann man einfach eine eigene Protokollierung implementieren. Es ist dabei nicht zwingend notwendig, sich an den Aufbau des OWB-Repository zu halten. Lediglich die Package-Calls aus dem Mapping müssen über entsprechende Wrapper-Prozeduren/-Funktionen weiterhin unterstützt werden.

Im Package „WB_RT_MAPAUDIT_UTIL“ sind alle Hilfsfunktionen hinsichtlich des Mapping-Ablaufs gebündelt. Insbesondere werden folgende Prozeduren/Funktionen benötigt:

- SUPPORTSDESIGNCLIENT
- RESOLVE_NAME
- VALIDATE_RUNTIME_PARAMETER
- PREMAP
- POSTMAP
- REGISTER_SYS_PARAM
- SHOW_RUN_RESULTS
- SET_SCHEMA_WORKSPACE

Die Funktionalität, die sich hinter den einzelnen Prozeduren/Funktionen versteckt, ist aus ihrem Namen schon ersichtlich. So prüft beispielsweise die Funktion „SUPPORTSDESIGNCLIENT“, ob die zu den Mappings passende Runtime-Umgebung

installiert wurde. Diese Prozeduren/Funktionen müssen durch eine Eigenentwicklung entsprechend bereitgestellt sein.

Das Package „WB_RT_MAPAUDIT_UTIL_INVOKER“ kapselt Aufrufe der durch die Datenbank selbst bereitgestellten Funktionen. Besonders herauszuheben ist dabei die Prozedur „GATHER_TABLE_STATS“, die in der Regel nach einem erfolgreichen Ablauf des Mappings für neue Statistiken sorgt. Die Kapselung der durch den OWB verwendeten Funktionen ist ebenfalls durch ein eigenes Package vorzunehmen.

Der wachsende ETL-Workframe

Ersetzt man die drei beschriebenen Packages durch Eigenentwicklungen und stellt eigene Protokoll-Tabellen zur Verfügung, ist man in der Lage, das Mapping „MAP_KUNDEN“ aus dem obigen Anwendungsfall zu starten und zu überwachen. Damit ist eine Migrationsfähigkeit dieses einfachen Mappings in eine OWB-freie Umgebung gegeben. Auch komplexere Mappings basieren hauptsächlich auf den Funktionen der drei Packages. In Einzelfällen können natürlich noch weitere von OWB bereitgestellte Funktionen notwendig sein, die dann entsprechend nachgebildet werden müssen.

Der ETL-Workframe ist so gestaltet, das man ihn jederzeit erweitern kann. Die im vorliegenden Fall aufgeführten OWB-Objekte sind nicht alle, die potenziell von einem Mapping genutzt werden können. Je nach Funktionalität eines Mappings sind weitere möglich. Daher muss im Einzelfall eine Erweiterung erfolgen. So kann der ETL-Workframe im Laufe der Zeit um weitere Funktionen ergänzt werden *(siehe Abbildung 3)*.

Migration

Hat man durch Nachbildung der Runtime-Komponente dafür gesorgt, dass Mappings auch in einer OWB-freien Umgebung lauffähig sind, kann eine Migration der einzelnen Mappings erfolgen. Um ein einzelnes Mapping in eine OWB-freie Umgebung zu migrieren, sind folgende Schritte notwendig:

- Das zu migrierende Mapping wird hinsichtlich noch nicht vorhandener OWB-Calls untersucht
- Sind zusätzliche Funktionen erforderlich, müssen sie erstellt und in die Funktionsbibliothek im Schema „OWBSYS“ aufgenommen werden

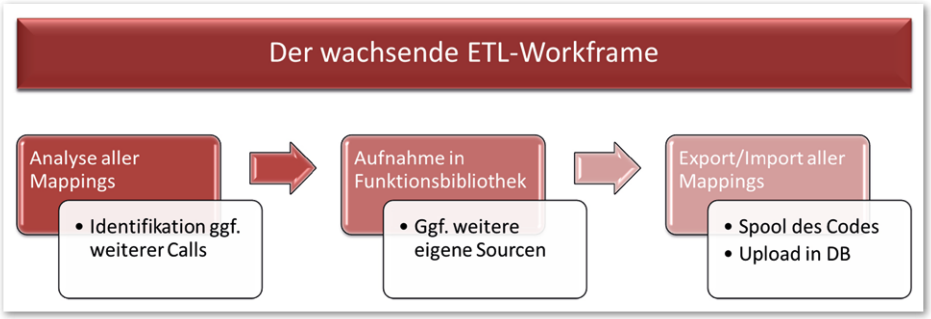


Abbildung 3: Wachsender ETL-Workframe

- Der Source-Code des Mappings wird aus der OWB-Umgebung exportiert; ein einfache Spool der Sourcen reicht aus, um diese in die OWB-freie Ziel-Datenbank zu importieren

Mit diesen drei Schritten kann nahezu jedes Mapping in eine OWB-freie Umgebung migriert und dort lauffähig gehalten werden. Die künstliche Grenze, die durch die Oracle-Datenbank 12*c* R2 gezogen wird, lässt sich somit zumindest hinsichtlich der Runtime-Komponente auflösen.

Fazit und Abgrenzung

Ziel des ETL-Workframes ist es, den Migrationsdruck hinsichtlich der Datenbewirtschaftung mit dem OWB zu mindern. Dies geschieht dadurch, dass die aus dem OWB heraus generierten Mappings auch über die Version 12*c* R1 hinaus lauffähig sind. Dabei ist allerdings zu beachten, dass der ETL-Workframe nur die Runtime-Komponente des OWB und nicht die Design-Komponente abbildet. Er ist weder ein vollwertiger ETL-Werkzeug-Ersatz, noch kann er unabhängig von einer zuvor stattgefundenen OWB-Entwicklung gesehen werden. Er leistet den Betrieb eines oder mehrerer generierter Mappings in einer OWB-freien Umgebung. Deshalb ist eine regelmäßige Datenbewirtschaftung durchzuführen. Dies eröffnet heutigen OWB-Nutzern, die vor der Migrationsentscheidung stehen, verschiedene Perspektiven.

Szenario 1: Man setzt zukünftig den ODI oder ein beliebiges anderes ETL-Werkzeug ein. Alle vorhandenen Mappings, die sich einfach „1:1“ migrieren lassen, werden migriert. Alle anderen werden durch den ETL-Workframe gekapselt und aus dem neuen ETL-Werkzeug heraus aufgerufen. Nachteil dieser Strategie ist, dass zusätzlich zu den Migrationskosten noch Lizenzgebühren für das ETL-Werkzeug anfallen.

Szenario 2: Man entscheidet sich für eine Open-Source-Strategie und setzt auch im ETL-Bereich darauf. Dann kann man wie in Szenario 1 verfahren und spart die Lizenzgebühren für das ETL-Werkzeug.

Szenario 3: Man betreibt ohne Support die Design-Komponente des OWB 11.2 auf einer Datenbank 12*c* R1. Dort erfolgen Wartung und Weiterentwicklung. Für den produktiven Einsatz der Mappings im eigentlichen DWH wird der ETL-Workframe eingesetzt. Dies kann natürlich nur ein Übergangsszenario sein, da auch der Extended Support Lizenzgebühren kostet und Juli 2021 für 12*c* R1 endet.

Szenario 4: Man verzichtet auf ein ETL-Werkzeug. Jegliche Wartung und Weiterentwicklung wird durch PL/SQL-Programmierung vorgenommen. Die vorhandenen Mappings werden durch den ETL-Workframe lauffähig gehalten. Was Lizenzkosten betrifft, ist das die günstigste Variante. Allerdings verzichtet man langfristig auf alle Vorteile, die ein ETL-Werkzeug bietet.

Welche dieser Szenarien im jeweiligen Fall zum Einsatz kommen, ist im Einzelfall zu klären. Allerdings eröffnet der ETL-Workframe Strategien jenseits der Komplett-Migration in ein anderes ETL-Werkzeug.

Sven Bosinger
sven.bosinger@its-people.de

Mit „Smart Replication“ Daten effizient, robust und flexibel verteilen

Carsten Kaftan, Sanacorp Pharmahandel GmbH

Erfreulicherweise kann der Autor alle Ankündigungen bezüglich Advanced Replication und Streams sowie die Preislisten von GoldenGate mit Gelassenheit studieren – er setzt seit dem Jahr 2008 für Replikationen die Eigenentwicklung „Smart Replication“ ein. Dieser Artikel ist die aktualisierte und überarbeitete Version eines Vortrags auf der DOAG-Jahreskonferenz; er richtet sich an Leser mit Interesse an einer schlichten und allgemein einsetzbaren Lösung für Replikationsaufgaben.

Wegen erheblicher Probleme mit der Flexibilität (Multi-Master-Architektur bei Advanced Replication) und hohen Ressourcen-Bedarfs (CPU-Verbrauch bei Streams) sowie der Tatsache, dass die angepriesenen und beworbenen Eigenschaften und Möglichkeiten zum Teil gar nicht den Bedürfnissen entsprachen, hat das Unternehmen des Autors bereits vor vielen Jahren den Schritt zur Eigenentwicklung „Smart Replication“ unternommen. Herausgekommen ist ein vollständig in PL/SQL geschriebenes Werkzeug, das sich auf die stabile und anpassungsfähige Weitergabe von Datenänderungen konzentriert. Elementare Verfahren erfassen diese und übertragen sie in die gewünschten Ziele. Die üblichen Replikations-Szenarien − sehr wenige Änderungen im großen zeitlichen Abstand, Massendaten-Änderungen, mehrere Quellen kombiniert mit mehreren Senken, Gültigkeitszeitraum-gesteuertes Herausziehen von aktuellen Daten aus Stammtabellen etc. − sind out oft he box abgedeckt.

Prinzip

Eine Smart Replication bezieht sich auf eine Quell-Tabelle beziehungsweise einen Quell-View und gegebenenfalls auf mehrere gleichartige Ziel-Tabellen beziehungsweise Views. Es lassen sich beliebig viele voneinander unabhängige Smart Replications einrichten, die mit einfachen Methoden wie Zwischen- und temporären Tabellen, Triggern und Jobs die Daten aus der Quelle heraus auf die gewünschten Ziele verteilen – innerhalb einer oder zwischen verschiedenen Datenbanken. *Abbildung 1* zeigt den strukturellen Aufbau einer Smart Replication.

Zunächst müssen die Quelldaten so, wie im Ziel benötigt, bereitgestellt werden; im einfachsten Fall sind Ziel und Quelle Tabellen von gleicher Struktur und mit gleichem Primärschlüssel. In anderen Fällen kann etwa eine passende View auf die Quell-Tabellen eingerichtet werden. Die Konfiguration der Smart Replication legt Quell- und Ziel-Datenobjekte, zugehörige Schlüsselfelder (normalerweise den Primärschlüssel) und Übertragungsparameter fest (Quelle, Ziel und Transfer). Bei Änderung eines Quelldatensatzes wird der zugehörige Schlüssel in die „Schlüsselwerte der Änderungen“-Tabelle eingefügt.

Die Erfassung der Datenänderungen erfolgt am einfachsten über Trigger (Auslöser) auf die Quelltabelle beziehungsweise auf die Basis-Tabellen zum Quellview; es ist aber auch möglich, die zu übertragenden Schlüssel gezielt in die Smart-Replication-Tabelle einzutragen. Beim Eintrag werden neben dem Schlüssel auch der vorgesehene Übertragungszeitraum und die Priorität festgelegt.

Ein Aufräumjob soll gewährleisten, dass sich keine überflüssigen Einträge ansammeln: So werden bei Bedarf Einträge zusammengefasst, etwa nach Mehrfach-Änderungen eines Datensatzes beziehungsweise Überschneidungen in den vorgesehenen Übertragungsfenstern. Gemäß der jeweiligen Konfiguration werden auch sogenannte „Ewigkeits-Einträge“ entfernt, deren Übertragungsfenster sehr weit in der Zukunft liegen. Diese können bei Historisierungen entstehen, falls etwa ein Datensatz mit Gültigkeit bis zum 31.12.9999 („für immer“) geändert wird und die zugehörige Smart Replication so konfiguriert ist, dass sie für „gültig von“- und „gültig bis“-Daten Aktualisierungen vormerkt.

Der Übertragungsjob wird immer bei Änderungen in der „Schlüsselwerte der Änderungen“-Tabelle aktualisiert und plant sich für den Beginn des nächsten Übertragungsfensters sowie danach solange in den vorgegebenen Zeitabständen ein, bis alle anliegenden Änderungen in der konfigurierten Blockgröße abgearbeitet sind. Danach plant sich der Übertragungsjob für den Beginn des nächstfolgenden Übertragungsfensters beziehungsweise, falls keine weiteren Datenänderungen vorliegen, für „nie“ ein. Die von diesem Job aufgerufene Datenübertragung läuft in folgenden Schritten ab:

- Zusammenstellen des nächsten Datensatzblocks nach Priorität und vorgesehenem Übertragungsfenster mit der konfigurierten Anzahl an Einträgen und Übertragen in die Temporary-Table im Ziel
- Löschen der Einträge mit Schlüsseln aus der Temporary-Table aus der Ziel-Tabelle
- Kopieren der Einträge mit Schlüsseln aus der Temporary-Table von der Quell- in die Ziel-Tabelle

Architektur

Es sind Quell-, Ziel- und gegebenenfalls Administrations-Datenbanken beteiligt, die problemlos auch identisch sein können. Überall muss ein „Admin“-User mit Hilfscode, auf den Quell- und Ziel-Datenbanken außerdem ein „Replic“-User eingerichtet sein. Bei optionaler Verwendung einer Administrations-Datenbank (enthält lediglich ein zusätzliches „Admin“-Package) können zentral zusätzliche nützliche Funktionen zum Einsatz kommen.

Das Verfahren Smart Replication ist in dem Package „DBA_REPLIC“ codiert. Dort finden sich auch die Prozeduren zur Einrichtung, Administration und Entfernung von Smart Replications. Bei dem Einrichten der ersten Smart Replication werden im „Replic“-Schema einige nötige Hilfsstrukturen (Views, Functions, Jobs etc.) erstellt, die beim Entfernen der letzten Smart Replication auch wieder gelöscht werden. Datenbankübergreifende Funktionen, etwa zur Administration, sind im Package „ADM_REPLIC“ auf der Administrations-Datenbank.

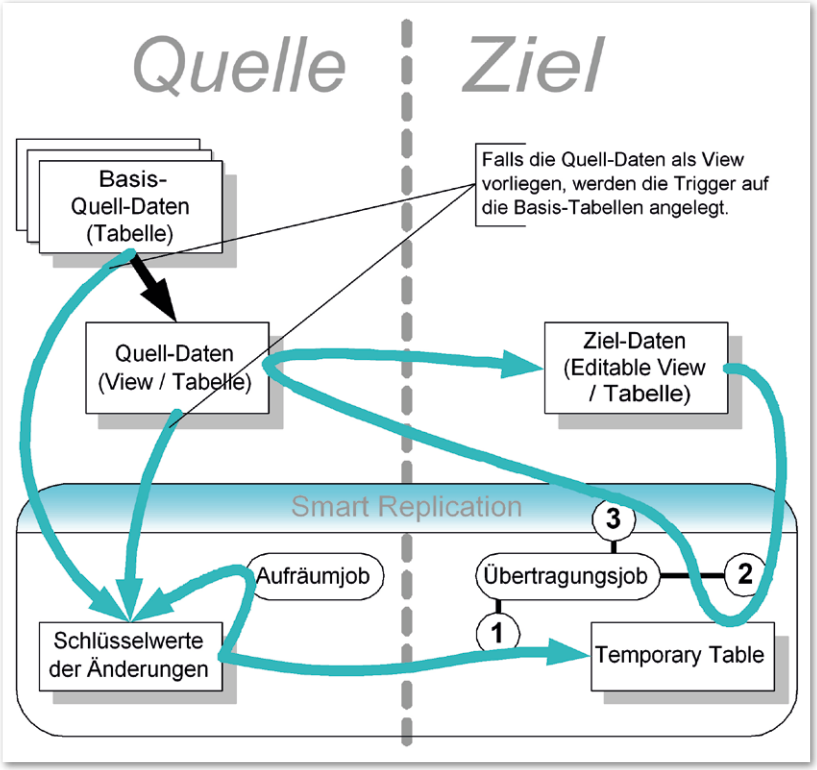


Abbildung 1: Schema-Diagramm der Smart Replication

Installation

Man geht entsprechend der Dokumentation *(siehe „http://www.doag.org/go/doagnews/smart_replication“)* vor und setzt, unter anderem damit die Compiler-Direktiven korrekt arbeiten, in den Skripten „SR_install_....sql“ die „Install“-Konstanten passend *(siehe „http://www.doag.org/go/doagnews/sr“ und Tabelle 1)*.

Die Diagnose-Funktion prüft, ob alles grundsätzlich funktioniert, beispielsweise bei Verwendung nur einer Datenbank mit dem Aufruf „clobResult := ADM_REPLIC.Diagnose(‘«SCHEMA»’, ‘’, ‘’, ‘«SCHEMA»’, ‘’, ‘’);“. Die Diagnose erstellt vorübergehend einige Test-Tabellen und Views mit mehreren Replikationen, es werden verschiedene Aufrufe getestet und eventuelle Probleme im Ergebnis-Report hervorgehoben.

<<<<<<< EDIT <<<<<<<	Weist auf Code-Stellen hin, die gegebenenfalls bearbeitet werden sollten
INSTALL_MINIMAL	Auf „true“ setzen – das „Admin“-Framework ist hier eingeschränkt
INSTALL_USER_ADMIN	Name des Administrations-Users
INSTALL_USER_REPLIC	Name des Replikations-Users
INSTALL_GRANTS_ROLE	Am einfachsten „public“; sonst eine kommagetrennte Liste derjenigen Benutzer, die Replikationen benutzen werden
INSTALL_DIRECTORY_LOG	Directory für Logdateien (sollte häufiger aufgeräumt werden; die Logdateien können groß werden)

Tabelle 1

Einrichtung einer Smart Replication

Beispielhaft wird jetzt die Einrichtung einer Smart Replication skizziert; Quell- und Ziel-Tabelle seien in zwei verschiedenen Datenbanken vorhanden, gleich geformt und benannt und mit Primärschlüsseln versehen; außerdem existieren die nötigen Datenbank-Links:

- *Einrichtung der Senke (Aufruf in der Zieldatenbank)*
  ADMIN.DBA_REPLIC.SmartTargetAdd(‘&&SCHEMA’,’&&TABELLE’);
- *Einrichtung der Quelle (dieser und die weiteren Aufrufe in der Quelldatenbank)*
  ADMIN.DBA_REPLIC.SmartSourceAdd(‘&&SCHEMA’,’&&TABELLE’)
- *Einrichtung des Transfers*
  ADMIN.DBA_REPLIC.SmartTransferAdd (‘&&SCHEMA’,’&&TABELLE’, ‘&&QUELLDATENBANK’,’&&ZIELDATENBANK’);
- *Erstellung und Ausführung des Skripts für die Auslöser-Trigger*
  sResult := ADMIN.DBA_REPLIC.SmartScriptTrigger(‘&&SCHEMA’,’&&TABELLE’);
- *execute immediate sResult;*

Das Erstellen des Skripts legt den Trigger noch nicht an, dazu muss das generierte Skript in einem weiteren Schritt ausgeführt werden. Dies gibt gegebenenfalls die Möglichkeit zur Kontrolle und Anpassung. Das Anlegen dieser Trigger ist der entscheidende Punkt bei der Einrichtung; in diesem Schritt werden sozusagen die Datenstrukturen mit der Replikation verbunden und ab diesem Moment die Änderungen repliziert.

Die Initialisierung der Senken bei bereits vorhandenen Daten kann auf viele verschiedene Weisen durchgeführt werden, ein passendes Skript lässt sich beispielsweise auf der Administrations-Datenbank generieren *(siehe Listing 1)*. Das

```
select TEXT_CONTENT from table(ADMIN.ADM_REPLIC.
SmartScript('&&ZIELDATENBANK', '&&SCHEMA','&&TABELLE'));
```

Listing 1

```
select TEXT_CONTENT from table(ADMIN.DBA_REPLIC.SmartReport('%'));
commit;
```

Listing 2

```
select    SR_OWNER, SR_OBJECT,
          SR_OVERDUE, SR_CURRENT,
          trunc(case when SR_SCHEDULED_OVERDUE-SR_SCHEDULED_FIRST >= .9/24
                     and SR_SCHEDULED_FIRST < SR_DATEREF
                then 100*(SR_DATEREF-SR_SCHEDULED_FIRST)/
                       (SR_SCHEDULED_OVERDUE-SR_SCHEDULED_FIRST)
                     else NULL end) PROZENT,
          case when nvl(SR_JOB_NEXTDATE,to_date('9999','yyyy')) >
                     greatest(SR_SCHEDULED_FIRST, SR_DATEREF + 5/1440)
              then 'JOB_NEXTDATE > SCHEDULED_FIRST'
              else 'ok'
              end NEXTDATE,
          SR_SCHEDULED_FIRST, SR_SCHEDULED_OVERDUE,
          SR_JOB_NEXTDATE, SR_JOB_FAILURES, SR_JOB_BROKEN
   from table (REPLIC.SmartStatusFct('%', '%', null, 1))
   where nvl(SR_SCHEDULED_FIRST, SYSDATE) < SYSDATE - 5/1440
       or nvl(SR_JOB_FAILURES, 0) > 0
       or nvl(SR_JOB_BROKEN, 'N') != 'N'
       or nvl(SR_JOB_NEXTDATE,to_date('9999','yyyy')) >
       greatest(SR_SCHEDULED_FIRST, SR_DATEREF + 5/1440)
order by 1, 2;
```

Listing 3

```
with info as (select OBJEKT, ZEIT, SEKUNDEN, ZEILEN from (
   select VON ZEIT, (BIS-VON)*24*60*60 SEKUNDEN, OBJEKT, ZEILEN from (
       select OBJEKT, ZEIT VON,
           lead(Zeit, 1) over (partition by OBJEKT
                               order by ZEIT, ISTSTART desc, ISTENDE) BIS,
           ZEILEN, IstStart, IstEnde, TEXT from (
       select ZEIT, OBJEKT, TEXT,
              case when TEXT like '%will now be transferred%'
                  then 1 else 0 end IstStart,
              case when TEXT like '%arget(s) completed.%'
                  then 1 else 0 end IstEnde,
              case when TEXT like '%will now be transferred%'
                  then to_number(substr(TEXT, 1, instr(TEXT, ' ')))
              else NULL end ZEILEN from (
          select to_date(substr(TEXT, 1, 20), 'dd.mm.yyyy, hh24:mi:ss') ZEIT,
              substr(TEXT, 24, instr(TEXT, ', SmartExecute')-24) OBJEKT,
              substr(TEXT, instr(TEXT, ', SmartExecute: ')+
                     length(', SmartExecute: ')) TEXT
                  from REPLIC.ALERT_REPLIC_LOG
              where TEXT like '%SmartExecute%'))
          where IstStart = 1 or IstEnde = 1 )
       where IstStart = 1
       order by OBJEKT, VON))
select OBJEKT, trunc(ZEIT, 'HH'), sum(Zeilen) ZEILEN,
       sum(Sekunden) SEKUNDEN, count(1) AUFRUFE
   from info
group by OBJEKT, trunc(ZEIT, 'HH')
order by 1, 2;
```

Listing 4

erhaltene Skript wird dann auf der Ziel-Datenbank ausgeführt und überträgt die Daten aus der Quelle.

Reporting und Monitoring

Abfragen auf „SmartReport", „SmartStatusFct" und „ALERT_REPLIC_LOG" geben die Informationen zu Status und Konfiguration der Smart Replications. Eine Auswahl dazu ist die umfassende Status- und Konfigurations-Abfrage (Aufruf in der Quell- oder in der Zieldatenbank, *siehe Listing 2)*, die schnelle Abfrage auf eventuelle Probleme (Aufruf in der Quelldatenbank, *siehe Listing 3)* sowie die Analyse der Logdatei nach „Übertragungsvolumen pro Stunde" (Aufruf in der Quelldatenbank, *siehe Listing 4)*.

Gezieltes Unterbrechen

Smart Replicaton erlaubt das Unterbrechen von Daten-Übertragungen in ein oder mehrere Senken (etwa bei Verbindungsproblemen) und das anschließende kontrollierte Nachfahren. Das Stoppen der Replikation in eine Senke (die anderen Transfers bleiben aktiv) erfolgt mit „ADMIN.DBA_REPLIC.SmartTransferSetStatus('«SCHEMA»', '«TABELLE»', '«ZIELDB_2»', '%', '%', 0);". Zum Starten aller Replikationen und Nachfahren der angefallenen Änderungen, soweit und wohin nötig, gibt man „ADMIN.DBA_REPLIC.SmartTransferSetStatus('%', '%', '%', '%', '%', 1);" ein.

Reparatur

Grundsätzlich können alle Installations- und Einrichtungs-Prozeduren (auch die Deinstallations- und Entfernungs-Prozeduren) beliebig oft aufgerufen und wiederholt werden; der jeweils erreichte Status wird überprüft und nur die eventuell noch ausstehenden Aktivitäten durchgeführt. Falls etwa während der Einrichtung einer Smart Replication ein Fehler aufgetreten ist, wird nach Behebung der Fehlerursache der verwendete Aufruf einfach wiederholt. Falls beispielsweise Smart-Replication-Objekte versehentlich entfernt wurden, kann zur Reparatur die Prozedur SmartSetup verwendet werden „ADMIN.DBA_REPLIC.SmartSetup;". Fehlende Installationsobjekte werden dann wieder eingerichtet.

Eigenschaften

Die Schlüssel-Qualitäten von Smart Replication sind:

- Massenänderungen werden zügig abgearbeitet und führen nicht zu Ressourcen-Engpässen.
- Die Übertragung von Änderungen lässt sich priorisieren – etwa bei Online-Anwendungen gegenüber Batch-Abläufen – und zeitlich steuern.
- Einfache Konfiguration: Weitgehend automatische Einrichtung, jederzeitige Anpassung der Einstellungen, unkomplizierte Änderung von Datenstrukturen, problemlose Erweiterung etwa um neue Zieldatenbanken, einfaches Kopieren, Reparieren etc.
- Sichere und flexible Administration: Durchgängige Kontrolle über den aktuellen Status und direkter Zugriff auf die Smart-Replication-Komponenten. Im Bedarfsfall sind eine gezielte Drosselung der Übertragungsraten, die Unterbrechung und Umleitung von Datenübertragungen etc. möglich. Im Status „InReorganisation" können auf Session-Ebene Datenänderungen ohne Folge-Replikation durchgeführt werden.

Wesentliche Unterschiede zu Streams beziehungsweise Advanced Replication sind:

- Die zu übertragenden Inhalte stehen nicht von vornherein fest, sondern nur die den Datensatz identifizierenden Schlüsselwerte. Die zum Übertragungszeitpunkt – nicht dem Änderungszeitpunkt – in der Quelle konkret vorhandenen Werte werden im Ziel eingefügt.
- Hohe Fehlertoleranz: Beispielsweise gibt es keinen Feldinhalts-Vergleich zwischen alten und neuen Einträgen, der gegebenenfalls zu einer Fehlermeldung führt – der Inhalt zum Zeitpunkt der letzten Datensatz-Übertragung liegt vorn. Ein Kollisionshandling ist deshalb nicht ohne Weiteres möglich; bei Bedarf lässt sich der Übertragungsschlüssel geeignet erweitern, etwa unter Einbeziehung des letzten Datensatz-Update-Zeitpunkts, um diese Thematik abzudecken.
- Keine Transaktionstreue und keine Einzelsatzverarbeitung: Smart Replication überträgt in unabhängigen Transaktionen mehrere, eventuell geeignet zusammengefasste Änderungen auf einmal.

Weitere Merkmale sind:

- Abweichend vom Default (der für bi- und multidirektionale Replikationen geeignet ist) kann konfiguriert werden, dass von Smart Replication übertragene Änderungen in einer anschließenden Smart Replication weiter übertragen werden; damit sind Ketten möglich.
- Mit der Verwendung von (Editable) Views lassen sich zu denselben Tabellen völlig verschiedene Smart Replications einrichten.
- Änderungen werden in den vorgesehenen Übertragungsfenstern sofort übertragen, es gibt kein Warten auf den Ablauf eines regelmäßigen Refresh-Intervalls oder Ähnliches.

Fazit

Smart Replication ist ein individuell entwickeltes Projekt; der Autor hat zwar die Installation allgemeingültig gestaltet und beschrieben, aber es handelt sich nicht um ein Produkt. Vor einer Verwendung sind Prüfungs-, Anpassungs- und Testaufwände fällig. Trotzdem ist der Autor davon überzeugt, dass in Fällen, in denen über den Ersatz der mittelfristig obsoleten Oracle-Techniken durch selbstgeschriebenen Code nachgedacht wird, die Adaption eines seit Jahren produktiven Verfahrens einigen Entwicklungsaufwand sparen kann.

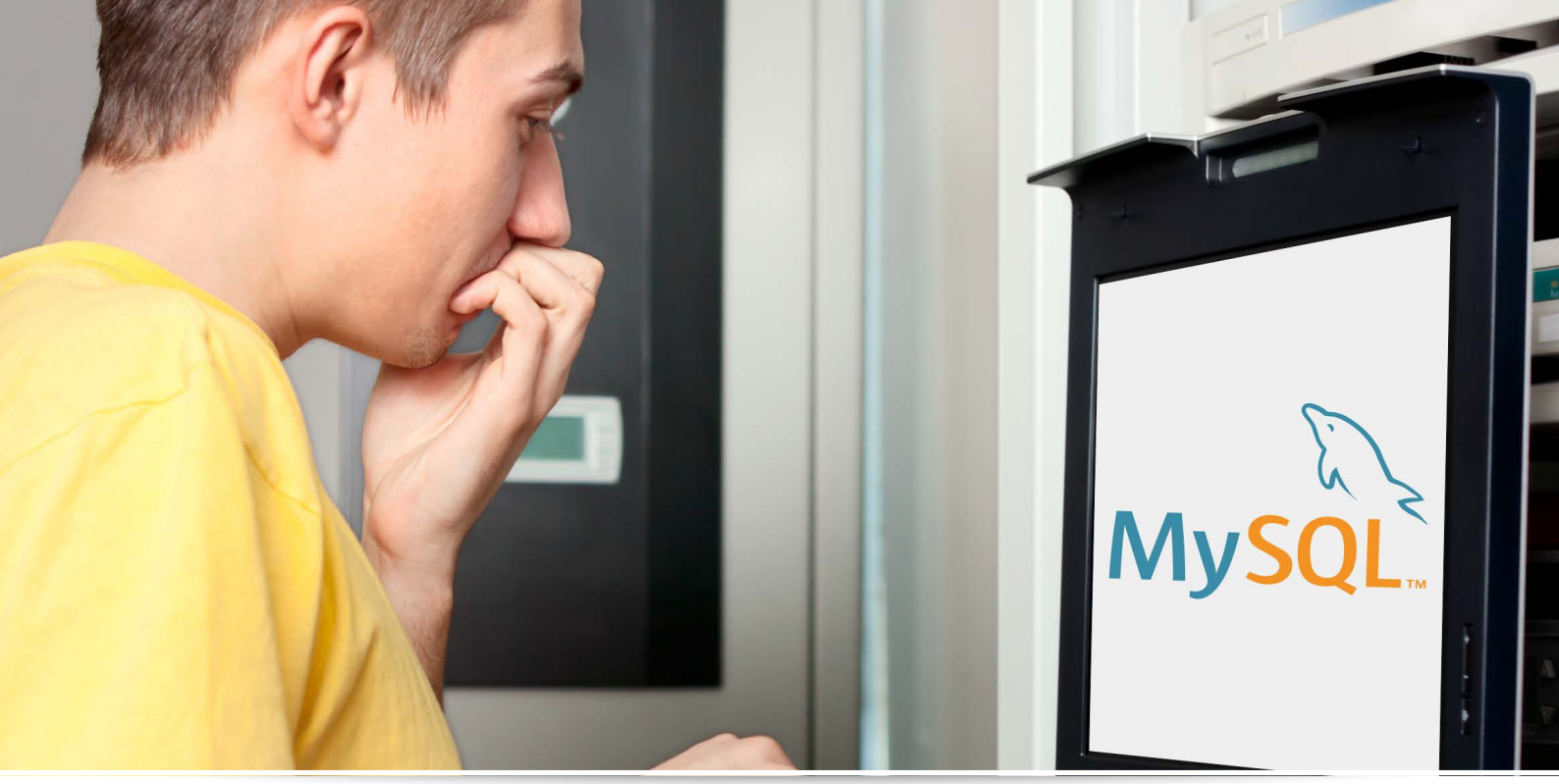
Carsten Kaftan
c.kaftan.sanacorp@myway.de

Delegiertenversammlung: DOAG soll offener und internationaler werden

Die Gewinnung von neuen Mitgliedern und Referenten wurde auf der Delegiertenversammlung stark diskutiert. Als eine Maßnahme der Mitgliederakquise sollen digitale Inhalte auf den Webseiten der DOAG zukünftig für jeden frei im Internet verfügbar sein. Die neue, offenere Ausrichtung der DOAG sowie weitere Maßnahmen und Details werden nachfolgend im Vorstand besprochen.

Besonderen Wert legten die Delegierten auch auf die Analyse und Verbesserung des Prozesses zur Gewinnung von Referenten für alle Veranstaltungen der DOAG. Hier wurde eine entsprechende Arbeitsgruppe mit fünf Delegierten eingerichtet, die zusammen mit je zwei Vertretern des Vorstands und der Geschäftsstelle den gesamten Prozess zur Referentengewinnung analysieren und dem Vorstand bis Ende des Jahres entsprechende Verbesserungsvorschläge präsentieren soll.

Weiterhin stimmten die Delegierten einstimmig der Internationalisierungsstrategie des Vorstands zu. Insbesondere die Jahreskonferenz der DOAG, die DOAG 2015 Konferenz + Ausstellung, soll internationaler werden. Auch eine noch intensivere Vernetzung mit den Usergroups aus den USA und aus Europa ist geplant.



MySQL für Oracle-DBAs

Oli Sennhauser, FromDual GmbH

In den letzten Monaten wurde der Autor vermehrt von Verantwortlichen für das Datenbank-Team kontaktiert, die MySQL-Datenbanken quasi aufs Auge gedrückt bekommen haben. Ihre Mitarbeiter sind ausgebildete und erfahrene Oracle-DBAs, die zukünftig auch MySQL-Datenbanken betreiben sollen. Sie kennen ihre Datenbanken zwar aus dem Effeff, sind bei MySQL-Datenbanken dann aber doch etwas ratlos. Der Artikel zeigt, wie man MySQL-Datenbanken professionell und hochverfügbar installiert und betreibt.

Oft wird MySQL durch die Applikation getrieben als Datenbank-Backend eingesetzt. Diese Applikationen entwickeln sich anfangs meist außerhalb des Fokus der IT-Strategen, werden dann plötzlich wichtig oder sogar geschäftskritisch. Das IT-Management kommt zu der Einsicht, dass diese Datenbank jetzt in professionelle, betriebliche Hände gehört. Was liegt da näher, als bestehende DBAs mit dieser Aufgabe zu betreuen. Es ist ja eine Datenbank und das können die DBAs ja. Eh man sich versieht, ist man verantwortlicher DBA von MySQL-Datenbanken.

Die betroffenen DBAs brauchen sich keine Sorgen bezüglich der Zuverlässigkeit von MySQL zu machen. MySQL ist eine transaktionsfähige Datenbank. Commit und Rollback im Fehlerfall sind mit MySQL genauso möglich wie bei einer Oracle-Datenbank. Auch bezüglich Hochverfügbarkeit und Leistungsfähigkeit kann MySQL mit anderen Produkten locker mithalten, sodass Datenbanken im Terabyte-Bereich mit einer Hauptspeichernutzung von 512 GB sowie Dutzenden von CPU-Cores im Einsatz sind.

Installation von MySQL

MySQL ist eine Open-Source-Datenbank und im Open-Source-Umfeld gibt es meist mehrere Wege, die zum Ziel führen. So auch bei der Installation von MySQL. Zuerst stellt sich die Frage: Community Edition oder Enterprise Edition? Technisch gibt es zwischen diesen beiden Varianten keinen Unterschied. Somit ist es dem persönlichen Geschmack jedes DBAs oder der Firmenpolitik überlassen, was eingesetzt wird. Die Community Edition wird auf jeden Fall breiter eingesetzt und Support für MySQL kann für beide Varianten von diversen Anbietern bezogen werden. Beide MySQL-Editionen können unter MySQL-Downloads *[1]* heruntergeladen werden. Hat man die Edition gewählt, besteht die nächste Entscheidung darin, auf welche Art und Weise MySQL installiert werden soll. Es stehen verschiedene Möglichkeiten zur Auswahl.

Am häufigsten werden RPM- oder Debian-Pakete eingesetzt, die mit der Distribution mitgeliefert werden. Diese Methode ist recht einfach und die Distribution stellt sicher, dass das Datenbank-Paket sauber in die Distribution integriert ist und mit allen Applikationen, die eine MySQL-Datenbank benötigen, reibungslos funktioniert. Der größte Nachteil bei der Wahl dieser Pakete ist, dass die Versionen im Stand oft etwas hinterherhinken und die allerneu-

este Version in der aktuellen Distribution noch nicht zur Verfügung steht.

Eine weitere Möglichkeit besteht darin, RPM- oder Debian-Pakete zu verwenden, die der Software-Hersteller oder Drittanbieter bereitstellen. Der Vorteil dieser Methode liegt darin, dass immer Pakete der neusten MySQL-Version vorhanden sind. Zudem kann man mit Hersteller-Paketen zu Testzwecken auch auf ein Beta-Release zugreifen. Der Nachteil bei den Hersteller-Paketen ist, dass sich diese nicht immer ganz reibungslos in die Distribution integrieren und gegebenenfalls zu Konflikten oder nicht sauber aufzulösenden Abhängigkeiten mit den übrigen Distributionspaketen führen können. Soll die MySQL-Datenbank aber ausschließlich als Backend für eine selbst geschriebene Applikation dienen, ist dies kein Problem. Zudem bieten verständlicherweise nicht alle Support-Anbieter für alle Drittanbieter-Pakete Support an.

In manchen Fällen, etwa bei Multi-Instanz-Setups mit unterschiedlichen MySQL-Versionen oder wenn ganz kurze Downtimes bei Upgrades gefordert werden, kann die Installation von MySQL über generische binäre „Tar-Balls“ („.tar.gz“) erfolgen. Bei dieser Installationsart ist die Interaktion mit der Distribution minimal, aber leider auch die Integration. Oft muss manuell das eine oder andere nachkonfiguriert werden. Diese Variante der MySQL-Installation ist eher für Spezialfälle gedacht.

Für die ganz ausgekochten DBAs besteht schließlich noch die Möglichkeit, MySQL aus den Quellen selber zu kompilieren – was bei einer Oracle-Datenbank unvorstellbar ist. Diese Variante der MySQL-Installation wird heute aber nur noch in seltenen Fällen gebraucht, zum Beispiel wenn man selber MySQL-Bugs fixen oder spezielle Patches von Drittanbietern in MySQL integrieren möchte. Nichtsdestotrotz lohnt es sich aus Verständnisgründen, diese Variante einmal durchzuprobieren. Die MySQL-Installation ist für die folgenden drei wichtigsten Linux-Distributionen möglich:

- *CentOS/RedHat/RHEL*
  shell> yum install mysql-server
- *Ubuntu/Debian*
  shell> apt-get install mysql-server
- *OpenSuSE/SLES*
  shell> zypper install mysql-community-server

Starten und Stoppen der MySQL-Instanz

Das Starten und Stoppen der MySQL-Instanz erfolgt üblicherweise durch den Betriebssystem-eigenen Start-/Stopp-Mechanismus („init“, „systemd“, „upstart“). Die Distribution sorgt dafür, dass beim Installieren des Pakets die Datenbank richtig in den Start-/Stopp-Mechanismus eingehängt wird.

Ebenfalls stellt die Distribution sicher, dass entweder beim Installieren des MySQL-Servers oder aber beim ersten Start der Instanz eine Datenbank auf der Default-Lokation („/var/lib/mysql“) angelegt wird. Diese Lokation kann natürlich nachträglich durch Verändern der MySQL-Konfiguration an jede beliebige Stelle des Filesystems verlegt werden. Leider heißt der Service, unter dem MySQL läuft, je nach Distribution leicht unterschiedlich:

- *CentOS/RedHat/RHEL*
  mysqld
- *Ubuntu/Debian*
  mysql
- *OpenSuSE/SLES*
  mysql

Der Befehl für das Starten und Stoppen des MySQL-Service lautet wie folgt *(siehe Listing 1)* oder etwas altbacken *(siehe Listing 2)*.

Diese Befehle bewirken, dass eine MySQL-Instanz gestartet oder gestoppt wird. Ob diese auch wirklich läuft, lässt sich mit „shell> service mysql status“ oder „shell> /etc/init.d/mysql status“ beziehungsweise mit Betriebssystem-Befehlen prüfen *(siehe Listing 3)*.

Hierbei wird man feststellen, dass es zwei MySQL-Prozesse gibt, den eigentlichen „mysqld“- und den „mysqld_safe“-Prozess. Letzterer wird auch als „Angel-Prozess“ bezeichnet. Er dient dazu, den „mysqld“-Prozess wieder zu starten, sollte sich dieser mal unerwarteterweise beenden. Der eigentliche Datenbank-Prozess ist jedoch der „mysqld“-Prozess.

Ob das Starten und Stoppen der MySQL-Instanz fehlerfrei funktioniert, kann im sogenannten „MySQL Error Log“ überprüft werden. Das MySQL Error Log entspricht in etwa dem „Oracle Alert Log“. Dieses liegt je nach gewählter Installationsmethode und Konfiguration an unterschiedlichen Orten und hat unterschiedliche Namen:

- *CentOS*
  „/var/lib/mysql/<hostname>.err“
- *Ubuntu*
  „/var/log/mysql/error.log“ und/oder „/var/log/syslog“
- *OpenSuSE*
  „/var/lib/mysql/<hostname>.err“

Datenbank-Clients

Wie auch bei Oracle gibt es bei MySQL ein Kommandozeilen-Interface (CLI) namens „mysql“. Mit diesem kann man sich gegen die MySQL-Datenbank verbinden. Eine MySQL-Instanz ist immer eindeutig durch die beiden Optionen „--hostname“ und „--port“ charakterisiert. Ein Instanz- oder Datenbank-Name, wie ihn Oracle kennt, ist bei MySQL unbekannt: „shell> mysql --host=127.0.0.1 --port=3306 --user=root --password“. Wurde MySQL frisch installiert und noch nicht gehärtet, hat der User „root“ noch kein Passwort.

```
shell> service mysql start
shell> service mysql stop
```

Listing 1

```
shell> /etc/init.d/mysql start
shell> /etc/init.d/mysql stop
```

Listing 2

```
shell> ps -ef | grep mysqld
UID        PID    PPID    C STIME    TTY      TIME CMD
mysql     1354    1       0 Jun11    ?        00:00:00 /bin/sh bin/mysqld_safe --datadir=/var/lib/mysql
--basedir=usr
mysql     1580    1354    0 Jun11    ?        00:00:43 /usr/bin/mysqld --basedir=/usr --datadir=/var/lib/mysql
--log-error=/var/lib/mysql/error.log --pid-file=/var/lib/mysql/mysql.pid --socket=/tmp/mysql.sock --port=3306
```

Listing 3

Bei der „--host“-Option gilt es zu beachten: „localhost“ ist nicht wie bei Unix-Systemen üblich ein Synonym für „127.0.0.1“, sondern zeigt dem MySQL-Client an, dass er sich nicht über TCP (Netzwerk), sondern über einen lokalen Unix-Socket verbinden soll.

Zudem muss beachtet werden, dass die MySQL-Standard-Pakete der Ubuntu- und Debian-Distributionen per Default den Zugriff auf MySQL von Remote unterbinden. Dies geschieht mit der Variable „bind_address“ in der MySQL-Konfigurations-Datei „/etc/mysql/my.cnf“. Ein Auskommentieren dieser Zeile, gefolgt von einem MySQL-Neustart, erlaubt den Zugriff auch von Remote. Wer lieber mit einem grafischen Benutzer-Interface (GUI) arbeitet, kann sich entweder der MySQL-Workbench *[2]* oder „phpMyAdmin“ *[3]* bedienen.

Schemata

Ist die Verbindung auf die Datenbank erst einmal hergestellt, kann man sich mit den Befehlen „mysql> SHOW SCHEMAS;“ oder „mysql> SHOW DATABASES;“ anzeigen lassen, welche Schemata die MySQL-Datenbank kennt. Ein Schema in MySQL entspricht in etwa einem Schema in Oracle. Der Unterschied zu Oracle besteht hauptsächlich darin, dass ein Schema nicht einem spezifischen User gehört, sondern der Instanz selbst. In MySQL können einem User nur spezifische Rechte auf bestimmte Objekte übertragen werden. Ein User ist aber nie Besitzer eines Objekts. Vier Schemata werden in MySQL typischerweise beim Erstellen der Datenbank angelegt:

- Das Schema „mysql“ enthält hauptsächlich alle Informationen über Benutzer und deren Privilegien.
- Das Schema „test“ ist, wie der Name schon sagt, für Testzwecke da. In diesem Schema kann man beliebig Tabellen anlegen und wieder löschen.
- Das „INFORMATION_SCHEMA“ („I_S“) bietet ein standardisiertes Interface für SQL-Abfragen auf Meta-Informationen der MySQL-Instanz. Hier gibt es zum Beispiel eine Tabelle namens „TABLES“, die Informationen zu allen Tabellen innerhalb der Instanz beinhaltet.
- Das „PERFORMANCE_SCHEMA“ („P_S“) beinhaltet zahlreiche Informationen und Messwerte, die für MySQL-Performance-Messungen und -Auswertungen zu Rate gezogen werden können.

„I_S“ und „P_S“ entsprechen in etwa den Oracle-„v$“-Tabellen.

Tabellen anlegen

Mit dem Befehl „mysql> use test“ wechselt man in den Kontext eines Schemas. Alle Operationen, die nicht explizit mit einem Schema-Namen spezifiziert werden, erfolgen nun auf den Objekten des gewählten Schemas. Der Befehl „mysql> SHOW TABLES;“ gibt einen Überblick darüber, welche Tabellen im entsprechenden Schema vorhanden sind. Jetzt kann man, wie von Oracle gewohnt, eine Tabelle anlegen *(siehe Listing 4)*. Neu dürfte für den Oracle-DBA das Schlüsselwort „ENGINE“ sein.

MySQL Storage Engines

Ein Konzept, das MySQL zugrunde liegt, sind die sogenannten „Pluggable Storage Engines“. Das bedeutet, dass MySQL eigentlich mit verschiedenen Datenbank-Containern als Backend umgehen kann, sofern diese die entsprechenden Interfaces aufweisen. Hypothetisch wäre es sogar möglich, dass die MySQL-Instanz über ein natives Interface auf eine Tabelle einer Oracle Datenbank zugreift. Mit „mysql> SHOW ENGINES;“ kann man sehen, welche Storage Engines MySQL zurzeit gerade kennt *(siehe Listing 5)*.

Es gibt eine Storage Engine namens „InnoDB“, die in der Lage ist, Daten in transaktionalen Tabellen zu speichern („InnoDB“ ist heute „default“) sowie eine Storage Engine names „MyISAM“, die Daten in nicht-transaktionalen Tabellen speichert. Die CSV Storage Engine speichert Daten zum Beispiel in CSV-Dateien (Comma-Separated-Values) ab, die dann von anderen Anwendungen wie zum Beispiel Excel gelesen werden können.

So gibt es für verschiedene Anwendungszwecke unterschiedliche Storage Engines. Das bietet die Flexibilität, eine Storage Engine für die spezifischen Bedürfnisse zu wählen, hat aber auch den Nachteil, dass man sich falsch entscheiden kann, wenn man deren Eigenschaften nicht genau kennt. Für die meisten Anwendungsfälle ist jedoch die voreingestellte InnoDB Storage Engine die richtige Wahl. Die schematische Darstellung des „mysqld“-Prozesses in *Abbildung 1* veranschaulicht die Idee der Storage Engines.

Mit „mysql> SHOW TABLE STATUS LIKE ‚test‘\G“ oder „mysql> SHOW CREATE TABLE test\G“ kann angezeigt werden, welche Storage Engine von der Tabelle genutzt

Sparen Sie Zeit, Geld und Nerven.

Effizient und preiswert: DBConcepts.

Wir unterstützen Sie remote beim Betrieb von Oracle Datenbanken.

SLA ab 10hx5 bis 24hx7 inklusive
- proaktiver Überwachung
- rascher Reaktionszeit
- periodische Health Checks
- Backup und Recovery Tests

DBCONCEPTS

Die Oracle Experten

www.dbconcepts.at
Tel.: +43 1 890 89 990
office@dbconcepts.at

ORACLE Platinum Partner

```
mysql> CREATE TABLE test (
   id INT UNSIGNED NOT NULL
, data VARCHAR(255)
, ts TIMESTAMP
) ENGINE = InnoDB;
```

Listing 4

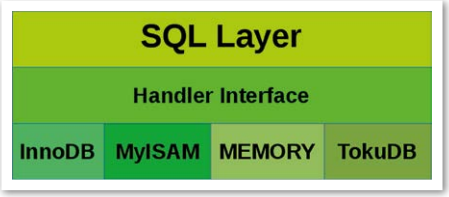


Abbildung 1: MySQL-Architekur

```
+--------------------+---------+--------------+------+
| Engine             | Support | Transactions | XA   |
+--------------------+---------+--------------+------+
| MyISAM             | YES     | NO           | NO   |
| CSV                | YES     | NO           | NO   |
| MRG_MYISAM         | YES     | NO           | NO   |
| BLACKHOLE          | YES     | NO           | NO   |
| MEMORY             | YES     | NO           | NO   |
| PERFORMANCE_SCHEMA | YES     | NO           | NO   |
| ARCHIVE            | YES     | NO           | NO   |
| InnoDB             | DEFAULT | YES          | YES  |
| FEDERATED          | NO      | NULL         | NULL |
+--------------------+---------+--------------+------+
```

Listing 5

wird. Storage Engines werden jeder Tabelle zugewiesen. Es besteht so die Möglichkeit, Abfragen beziehungsweise Joins über zwei Tabellen mit unterschiedlichen Storage Engines auszuführen.

MySQL-User

Um herauszufinden, welche User in der MySQL-Instanz existieren, hilft die Tabelle „user" im Schema „mysql". Entweder mit „mysql> use mysql;" und „mysql> SELECT user, host, password FROM user;" oder mit „mysql> SELECT user, host, password FROM mysql.user;" kann bestimmt werden, welche User aktuell in der Instanz sind *(siehe Listing 6)*.

Hierbei stellt man fest, dass es zum Beispiel den User „root" mehr als einmal gibt. Hierzu muss man wissen, dass in MySQL ein User immer aus „Username" und „Host" besteht. Die User „'root'@'localhost'" und „'root'@'127.0.0.1'" sind somit unterschiedliche User. Und wie wir aus obigen Erläuterungen erraten können, darf sich der erste User „root" nur über den lokalen Unix-Socket („localhost"), der zweite User „root" („127.0.0.1") nur über den lokalen TCP-Port 3306 und der dritte User „root" nur vom Host „centos-tmpl", der über einen DNS-Lookup aufgelöst wird, verbinden.

Der User „root" in MySQL hat typischerweise alle Privilegien und ist somit Superuser der MySQL-Instanz. Dies entspricht in etwa dem User „SYS" in Oracle. Welche Rechte ein User hat, lässt sich mit dem Befehl „mysql> SHOW GRANTS FOR 'root'@'localhost';" herausfinden.

Allgemein gilt es als sicherheitstechnisch bedenklich, dem User „root" Zugriff von Remote zu gewähren. Darüber hinaus sind auch der User „test" sowie das Schema „test" auf Produktiv-Umgebungen fragwürdig. Um eine MySQL-Instanz für den produktiven Einsatz zu härten, gibt es daher das Skript „shell> mysql_secure_installation", das einige sicherheitsrelevante Einstellungen für MySQL vornimmt.

```
+------+-------------+----------+
| user | host        | password |
+------+-------------+----------+
| root | localhost   |          |
| root | centos-tmpl |          |
| root | 127.0.0.1   |          |
|      | localhost   |          |
|      | centos-tmpl |          |
+------+-------------+----------+
```

Listing 6

MySQL-Konfiguration

Die Konfiguration von MySQL erfolgt über die Datei „/etc/my.cnf" (CentOS, SuSE) oder „/etc/mysql/my.cnf" (Ubuntu, Debian). Unter Windows liegt diese unter „C:\Program Files\MySQL\MySQL Server 5.6" und heißt „my.ini". Darin sind alle MySQL-spezifischen Variablen angegeben, um die Instanz zu konfigurieren und zu tunen.

Die Datei „my.cnf" ist unterteilt in verschiedene Abschnitte, die mit eckigen Klammern gekennzeichnet sind. Es gibt einen Abschnitt für den eigentlichen Datenbank-Prozess („mysqld"), für den „mysqld_safe"-Prozess („mysqld_safe"), für alle lokalen MySQL-Client-Prozesse („client") etc. Je nachdem, was man konfigurieren möchte, muss man beachten, dass die entsprechenden Parameter in der richtigen Sektion landen. Sonst gibt es entweder Fehler oder eine Konfiguration greift nicht.

Einige Variablen in MySQL sind Session-spezifisch, andere gelten global. Einige Variablen lassen sich dynamisch im laufenden Betrieb ändern, andere erfordern einen Instanz-Neustart, um aktiviert zu werden. Welche Variablen wie aktiviert werden, steht in der MySQL-Dokumentation *[4]*. Der Befehl „mysql> SHOW GLOBAL VARIABLES LIKE '%xyz%';" gibt an, welche Werte für MySQL-Variablen aktuell gesetzt sind.

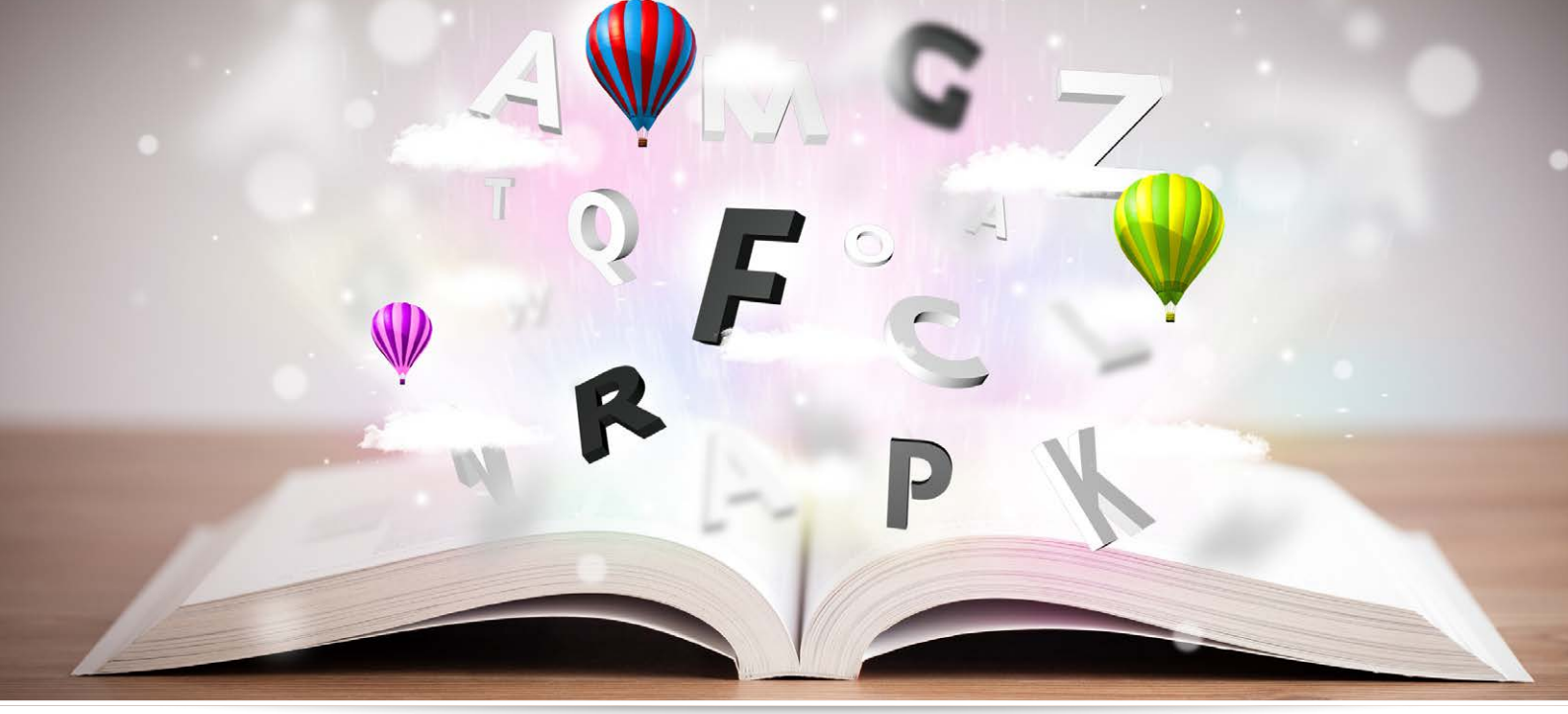
MySQL-Dokumentation

Neben einer ganzen Reihe von Büchern zum Thema MySQL bietet MySQL auch eine Online-Dokumentation, die recht gut ist und täglich aktualisiert wird *[5]*.

Links

[1] MySQL Downloads: http://dev.mysql.com/downloads
[2] MySQL Workbench: http://dev.mysql.com/downloads/workbench
[3] phpMyAdmin: http://www.phpmyadmin.net
[4] Server Option and Variable Reference: http://dev.mysql.com/doc/refman/5.6/en/mysqld-option-tables.html
[5] MySQL Dokumentation: http://dev.mysql.com/doc

Oli Sennhauser
oli.sennhauser@fromdual.com



Character Sets – die Welt ist nicht genug

Martin Hoermann, Ordix AG

Die Auswahl des Character Sets beim Anlegen einer Datenbank entscheidet über die zur Verfügung stehenden Zeichen der Datenbank. Die richtige Verwendung und die Änderung des Zeichensatzes, etwa auf einen Unicode-Zeichensatz, stellen den Administrator und Anwendungsentwickler vor verschiedene Herausforderungen. Sollten Anwender im Geheimdienst arbeiten, müssen natürlich Nachrichten aus allen Sprachen dieser Welt gespeichert werden können. Der Artikel beleuchtet die Grundlagen von Character Sets, deren Migration sowie die verschiedenen Ausprägungen der unter Oracle verfügbaren Unicode-Zeichensätze.

Der Character Set einer Datenbank legt prinzipiell die zur Verfügung stehenden Zeichen einer Oracle Datenbank-Anwendung fest. In Westeuropa waren früher die Datenbank-Zeichensätze „WE8ISO8859P1", „WE8ISO8859P15" und „WE8MSWIN1252" sehr weit verbreitet. Diese Zeichensätze haben höchstens 256 Zeichen, sodass jedes Zeichen mit einem Byte kodiert werden kann *(siehe Abbildung 1)*.

Diese sogenannten Single-Byte-Zeichensätze haben den Nachteil, dass nicht definierte Zeichen – etwa die Buchstaben des griechischen Alphabets – damit nicht definiert gespeichert werden können. Es ist zwar prinzipiell möglich, die Bytes anders als definiert zu interpretieren, jedoch führt dies zu einer sehr unübersichtlichen Datenspeicherung. Davon ist dringend abzuraten.

Um das Problem des eingeschränkten Zeichensatzes zu beheben, wurde mit Unicode ein sehr mächtiger Standard definiert, der die Zeichen der meisten lebenden und vieler ausgestorbener Sprachen definiert. Die am meisten verwendeten Kodierungen der Datenbank sind „UTF8" (veraltet), „AL32UTF8" und „AL16UTF16". Mit dem Unicode-Standard 6.1 stehen 110.181 Zeichen in der Version 12*c* zur Verfügung. Die Unicode-Zeichensätze sind Multibyte-Zeichensätze, die je nach Kodierung und Zeichen zwischen einem und vier Byte Speicherplatz benötigen.

Vom Client zur Datenbank und zurück

Dieses Kapitel beschreibt den Weg eines Zeichens von der Tastatur bis hin zur Datenbank und von der Datenbank wieder zurück zum Bildschirm des Anwenders: Wenn eine Datei auf einem Client erzeugt wird, so hat diese Datei eine bestimmte Kodierung (engl. „Encoding") oder auch einen bestimmten Zeichensatz. Idealerweise sollte die „NLS_LANG"-Variable genau diesen Zeichensatz benennen, beispielsweise „NLS_LANG=. WE8MSWIN1252", wenn es um eine auf Windows mit deutscher Spracheinstellung und mit einem Standard-Editor erzeugte Datei geht *(siehe Abbildung 2)*.

Unterscheidet sich der Zeichensatz in der Definition von „NLS_LANG" und dem Datenbank-Zeichensatz, so wird über einen festgelegten Algorithmus eine Konvertierung durchgeführt. Existiert dasselbe Zeichen in beiden Zeichensätzen, wird es entsprechend auf die Kodierung des Ziels – also der Datenbank – konvertiert. Existiert das Zeichen nicht, gibt es jedoch ein äquivalentes Zeichen, so wird das Zeichen ersetzt. So könnte beispielsweise aus einem „ä" ein „a" werden.

Gibt es weder das genaue noch ein äquivalentes Zeichen, so wird ein Ersatzzeichen („Replacement Character") verwendet. Äqui-

valente Zeichen und Ersatzzeichen können nicht wieder zurückgewandelt werden. Idealerweise sollten die Zeichensätze zwischen Client und Datenbank also entweder identisch sein oder die Datenbank sollte eine Obermenge („Superset“) von Zeichen zur Verfügung stellen. Dies ist dann in der Regel ein Unicode-Zeichensatz. Sind der Client-Zeichensatz und der Datenbank-Zeichensatz identisch, gehen weder Zeichen verloren noch werden Zeichen ersetzt. Dies ist der Idealzustand einer jeden Datenbank-Anwendung.

Wird nun ein Zeichen aus der Datenbank zurückgelesen, passiert prinzipiell genau das zuvor Beschriebene, nur in umgekehrter Richtung. Unterscheiden sich also „NLS_LANG“ des Client und der Zeichensatz der Datenbank, können Zeichen entweder in ein äquivalentes Zeichen oder ein Ersatzzeichen konvertiert werden. Alle Daten, die verlustfrei vom Client in die Datenbank gelangt sind, kommen bei gleicher Konfiguration auch verlustfrei wieder zurück.

Ob die Zeichen allerdings korrekt dargestellt werden können, hängt vom Zeichensatz des Frontend und dem zur Verfügung stehenden Font zur Darstellung ab. In einer DOS-Box („Codepage 850“) gelingt es meist nicht, trotz korrekter Einstellung die deutschen Umlaute darzustellen. Schreibt man hingegen aus einer DOS-Box eine Spool-Datei und öffnet diese wiederum mit einem Standard-Editor („Codepage 1252“), so sind die Zeichen dort sichtbar.

Code	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	nicht belegt															
1…																
2…	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5…	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6…	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7…	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8…	nicht belegt															
9…																
A…	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B…	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C…	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D…	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E…	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F…	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Abbildung 1: ISO-8859-1, Quelle: http://de.wikipedia.org/wiki/ISO_8859-1

Invalid Data & Conversion Errors

Bei der oben beschriebenen Verbindung kann es zu zwei Arten von Problemen kommen, die Oracle als „Invalid Data“ beziehungsweise „Pass Through“-Konfiguration und „Conversion Errors“ bezeichnet. Beide Phänomene wurden bereits zuvor erklärt und werden nun an einem Beispiel veranschaulicht.

Ist die Kodierung einer Datei beispielsweise in „WE8ISO8859P7“ und enthält Zeichen des griechischen Alphabets und sind die „NLS_LANG“-Variable sowie der Datenbank-Zeichensatz auf WE8ISO8859P1 eingestellt, werden die Zeichen an die entsprechenden Positionen „0xA0..0xFF“ geschrieben. Werden die Daten mit derselben Konfiguration ausgelesen und als „P7“-kodiert interpretiert, sind die griechischen Zeichen wieder sichtbar.

Ohne dieses Wissen sind die Daten jedoch bedeutungslos, da es sich im „P1“-Zeichensatz um Sonderzeichen verschiedener Sprachen handelt. Wird die Datenbank nun beispielsweise in einen anderen Zeichensatz konvertiert, werden natürlich semantisch die „P1“-Daten konvertiert, was ein Auslesen praktisch unmöglich macht. Oracle spricht hier von „Invalid Data“. *Abbildung 3* zeigt, wie eine solche „Pass Through“-Konfiguration mittels DMU (siehe unten) behoben werden kann.

Ist die Kodierung einer Datei beispielsweise „WE8ISO8859P7“ und enthält Zeichen des griechischen Alphabets und sind die „NLS_LANG“-Variable auf „P7“ und der Datenbank-Zeichensatz auf „P1“ eingestellt, so konvertiert die Datenbank die Zeichen. Dabei werden bis auf das kleine Beta alle Zeichen durch den Replacement-Character des „P1“ – das umgekehrte Fragezeichen – ersetzt. Das kleine Beta wird zum deutschen „ß“. Die ursprünglichen Daten sind damit nicht mehr reproduzierbar. Oracle spricht von „Conversion Errors“.

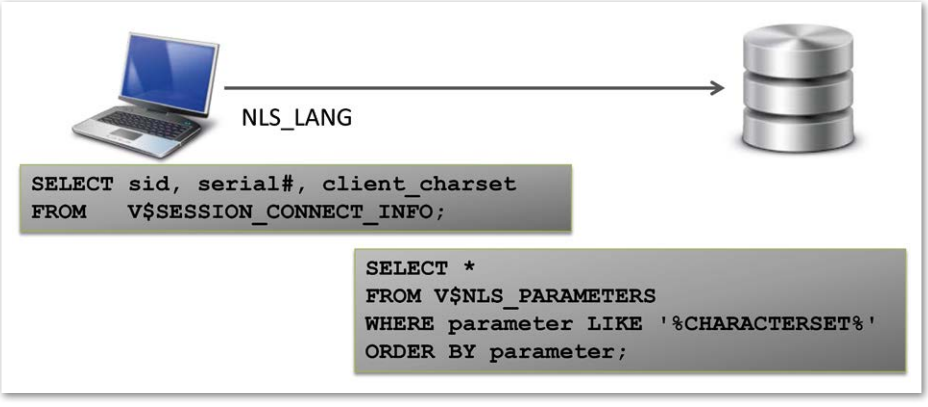


Abbildung 2: Informationen zum Character Set des Client und der Datenbank

Encoding I

Bei Single-Byte-Datensätzen ist die Kodierung relativ einfach. Jedem Buchstaben wird ein Wert zwischen „0x00“ und „0xFF“ zugewiesen. Die Zuweisung erfolgt über Character-Set-Tabellen, die im Internet, etwa bei Wikipedia, einfach zu recherchieren sind.

Bei Unicode-Zeichensätzen gestaltet sich die Kodierung etwas schwieriger. Jedes Unicode-Zeichen erhält eine Nummer.

Diese Nummer wird in der Regel Hex-kodiert mit führenden „U+“ geschrieben. So ist ein „Ä“ beispielsweise ein „U+00C4“. Im Internet stehen umfangreiche Unicode-Tabellen zur Verfügung.

Jetzt gibt es für verschiedene Anwendungszwecke unterschiedliche Kodierungen. Für den westeuropäischen Raum eignet sich die Kodierung „AL32UTF8“. Dabei werden die 128 ASCII-Zeichen identisch wie in der ASCII-Tabelle kodiert und kommen daher mit 7 Bit plus einem führenden „0“-Bit aus. Alle anderen Zeichen werden mit zwei bis vier Byte kodiert. Hilfreich kann der Algorithmus zum Auslesen sein, um etwa festzustellen, ob ein „Ä“ tatsächlich ein „Ä“ ist.

Mithilfe der Dump-Funktion auf ein einzelnes Zeichen ergibt sich beispielsweise „SELECT dump(zuklaerendeszeichen) FROM tabelle“ und „Typ=1 Len=2 195, 132“. Im ersten Schritt bietet es sich an, die Zahlen in Binärschreibweise als „11000011 10000100“ aufzustellen. Die führende „110“ des ersten Byte beschreibt, dass es sich um einen Zwei-Byte-Zeichen handelt. Die führende „10“ des zweiten Byte besagt, dass es ein Folgebyte ist. Alle anderen Bits werden nun zusammengefasst, in Hex-Code umgewandelt und aus einer Unicode-Tabelle ausgelesen: „00011000100 = 19610 = U+00C4 = Ä“. Bei dem zu klärenden Zeichen handelt es sich also tatsächlich um ein „Ä“.

Encoding II

Wie bereits erläutert, sind vielfältige Komponenten auf dem Weg von der Tatstatur bis zur Datenbank beteiligt. Der Datenbank-Administrator legt beim Anlegen der Oracle-Datenbank in der Regel nach Vorgabe des Applikationsverantwortlichen den Zeichensatz der Datenbank fest. Dieser ist im Nachhinein nur mit einigem Aufwand zu ändern.

Auf Seite des Client sind die Oracle-NLS-Einstellungen und die Kodierung der Anwendung relevant. Im Falle einer Datei kann diese von verschiedenen Werkzeugen gelesen und manipuliert werden. SQL*Developer, TOAD und Ultra-Compare verfügen zwar über Einstellungen für die Default-Kodierung, interpretieren aber häufig zur Laufzeit anhand des Daten-Inhalts die Kodierung der Datei um. Gelangt zum Beispiel ein Schmierzeichen in eine Datei, so wird von vielen Tools eine Unicode-Kodierung angenommen. Dies kann fatale Auswirkungen bei der Arbeit mit den Dateien haben. Für die Arbeit mit einer Anwendung gilt prinzipiell dasselbe.

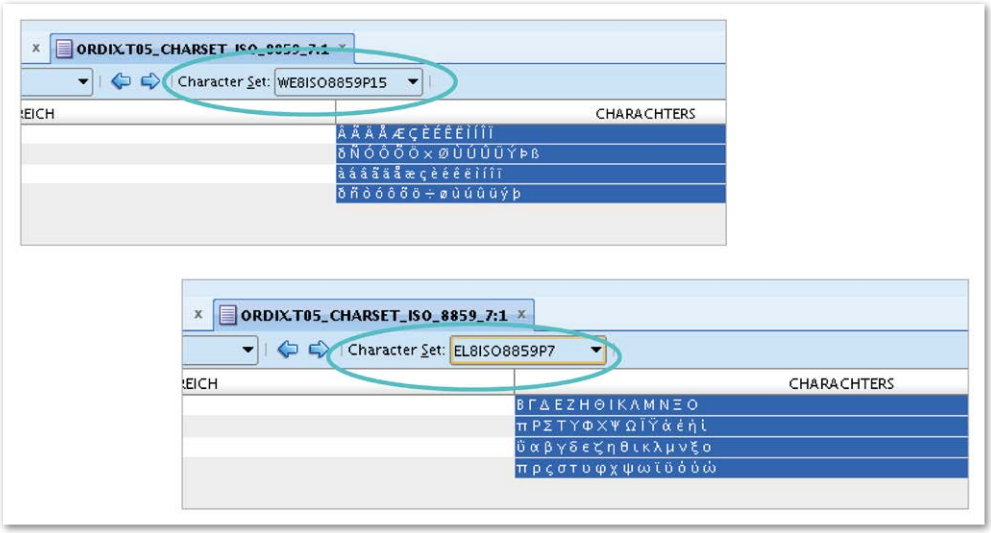


Abbildung 3: Behebung einer „Pass Through“-Konfiguration mittels DMU

Unicode in der Datenbank

Die Datenbank verfügt über zwei Zeichensätze („v$nls_parameters“). Der Datenbank-Zeichensatz („NLS_CHARACTERSET“) legt die Kodierung für die Spalten mit den Datentypen „VARCHAR2“, „CHAR“ und „CLOB“ fest. Der NLS-Datenbank-Zeichensatz („NLS_NCHAR_CHARACTERSET“) bestimmt die Kodierung für die Spalten mit den Datentypen „NVARCHAR2“, „NCHAR“ und „NCLOB“.

Bis zu der Oracle-Datenbank-Version 8 lautete der Name des „UTF-8“-Zeichensatzes „UTF8“, ab der Version 9 heißt er „AL32UTF8“. Der Präfix „AL2 steht dabei für „All Languages“. Abhängig von der Oracle-Version wird mit diesem Zeichensatz eine jeweils aktuelle Unicode-Spezifikation unterstützt, beispielsweise in Version 11 die Spezifikation 5.0 und in der Version 12*c* R1 die Spezifikation 6.1 *(siehe Abbildung 4)*.

In früheren Versionen war es noch möglich, sowohl für den Datenbank-Zeichensatz als auch für den NLS-Datenbank-Zeichensatz einheitlich „UTF8“ zu verwenden. Bei Migrationen stehen beim Datenbank-Zeichensatz „AL32UTF8“ für den NLS-Datenbank-Zeichensatz ausschließlich die Zeichensätze „AL16UTF16“ und „UTF8“ zur Verfügung. Der Zeichensatz „AL16UTF16“ ist unter anderem optimiert auf chinesische, koreanische und japanische Zeichen, von denen die meisten nur zwei Byte benötigen. In „AL32UTF8“ benötigen diese Zeichen in der Regel drei Byte.

Bei der Migration des Zeichensatzes kann es aufgrund der unterschiedlichen Kodierung natürlich zu einem Datenverlust kommen, da die Kodierungen für ein und dasselbe Zeichen unterschiedlich viele Bytes benötigen. Insbesondere die Einstellung von „NLS_LENGTH_SEMANTIC“ auf Bytes statt Character verursacht hier potenziell Probleme.

Aber auch mit der Einstellung auf Character kann das Problem entstehen, wenn die maximale Größe der Datentypen „CHAR“ und „VARCHAR2“ bei der Konvertierung 4.000 Bytes überschreitet. Unter Oracle 12*c* lässt sich diese Grenze auf 32.767 Byte erweitern. Hierzu ist der Initialisierungsparameter „MAX_STRING_SIZE“ auf „EXTENDED“ zu setzen. Diese Einstellung ist unumkehrbar.

Abbildung 4: Das wunderschöne Zeichen „U+2A6A5“ besteht aus 64 Grundstrichen und stellt vier Mal das Wort „Drachen“ dar

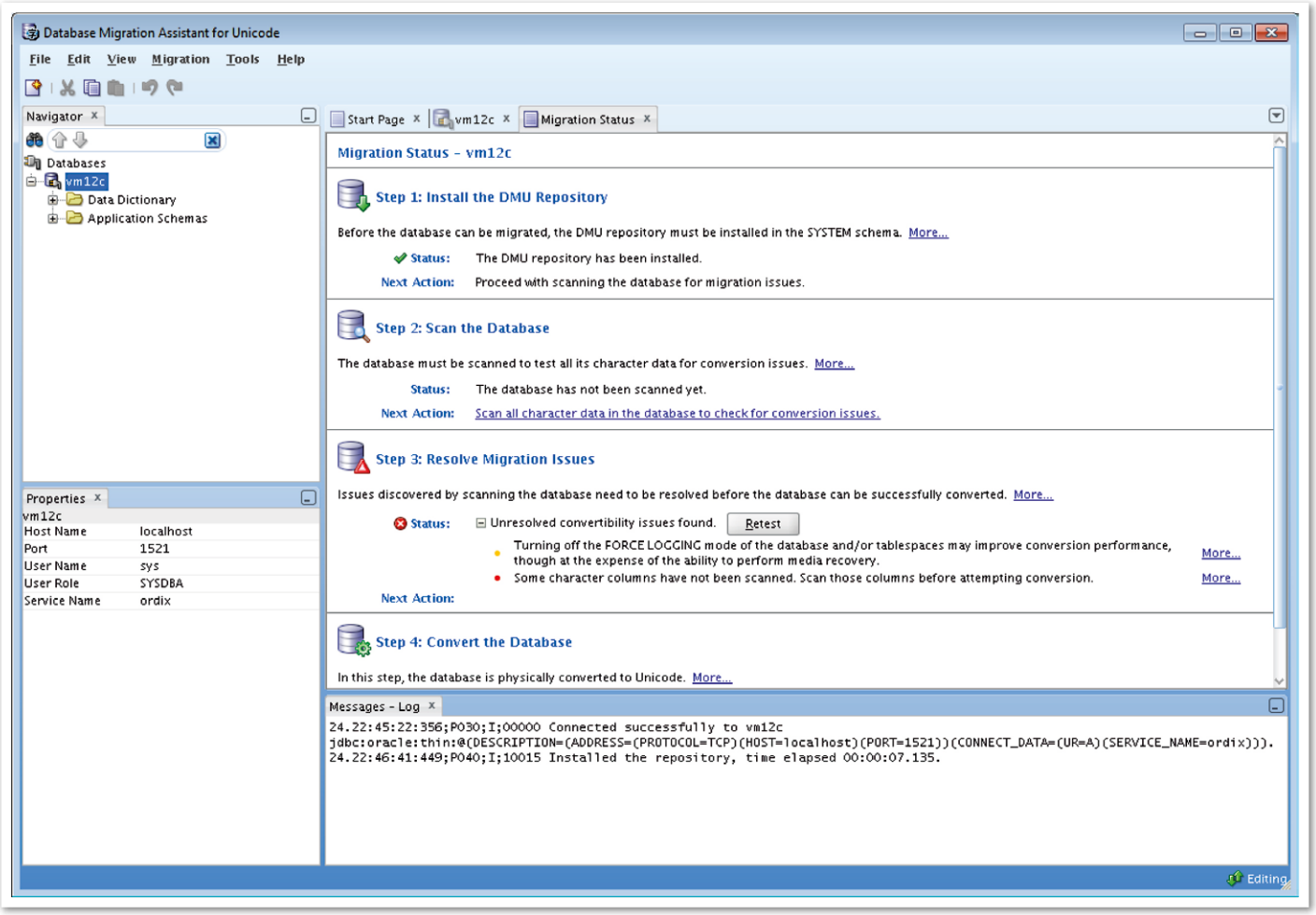


Abbildung 5: Startmaske des DMU

Character-Set-Migration

Oracle bietet verschiedene Möglichkeiten, den Zeichensatz einer Datenbank zu migrieren oder die Migration zu prüfen: „Export/Import“, „CSSCAN“ und „DMU“. Das Database Migration Utility (DMU) steht seit der Version 12.1 zur Verfügung und soll die Arbeit bei der Migration des Zeichensatzes vereinfachen *(siehe Abbildung 5)*.

Bei der Migration auf einen anderen Zeichensatz besteht aufgrund der Kodierung immer die Möglichkeit des Datenverlusts („Data Truncation“), weil Character-Felder eine für die neue Kodierung nicht ausreichende Länge haben (siehe oben). Zudem kann es zu Konvertierungsfehlern kommen, wenn ein Zeichen des ursprünglichen Zeichensatzes im neuen Zeichensatz nicht zur Verfügung steht („Replacement Errors“). In diesem Fall werden die Zeichen durch ein sogenanntes „Replacement Character“ ersetzt (siehe oben). Sowohl „DMU“ als auch der Vorgänger „CLSCAN“ prüfen die Daten vor der Migration auf potenzielle Fehler.

Sortierung

Der Datenbank-Zeichensatz legt die zur Verfügung stehenden Zeichen fest. Dies ist jedoch nur die Basis der Globalisierung einer Datenbank-Anwendung. Im nächsten Schritt muss für die Anwendung eine Sortierung festgelegt werden. Dies geschieht in der Regel über den Parameter „NLS_SORT“, der wiederum abhängig vom Parameter „NLS_LANGUAGE“ ist. Für die gängigen Sprachen stehen die dort typisch verwendeten Sortiermöglichkeiten zur Verfügung. Diese sind für den deutschen Sprachraum beispielsweise „GERMAN“, „XGERMAN“, „GERMAN_DIN“ und „XGERMAN_DIN“. Ab der Version 12 stellt Oracle eine Implementierung des Unicode Collation Algorithm (UCA) zur Verfügung. Über dieses Verfahren können auch mehrsprachige Sortierungen eingestellt und sogar selbst implementiert werden.

Fazit

Dieser Artikel zeigt die zahlreichen Facetten des Datenbank-Zeichensatzes. Mit einem Unicode-Zeichensatz stehen zwar alle gängigen Zeichen dieser Welt zur Verfügung, nichtsdestotrotz kann es bei der Anwendung zahlreiche Hürden geben. Vielleicht kann ja die NSA bei der Konvertierung kyrillischer Geheimbotschaften helfen.

Martin Hoermann
mh@ordix.de

Oracle 12c Automatic Performance Diagnostics

Angelika Gallwitz, Gallwitz-IT

Die Datenbank-Version 12c bietet maßgebliche Neuerungen in der Performance-Analyse und -Diagnose durch den „REAL TIME ADDM". Der „Automatic Database Diagnostic Monitor" (ADDM) kann in unterschiedlichen Connection-Modi gefahren werden. In Abhängigkeit davon werden unterschiedliche Diagnose-Ergebnisse erzielt.

Der Artikel zeigt die automatischen Features der Performance-Diagnose und des Tunings in der Oracle-Datenbank 12*c* und wie der ADDM ein Set vordefinierter Kriterien durchläuft, nach denen die aktuelle Performance der Datenbank analysiert wird. Nachdem ein Problem identifiziert und analysiert worden ist, hilft der „Real Time ADDM" mit dem „Emergency Monitoring Feature", die Ursachen und Lösungen zu finden, wenn die Datenbank hängt.

Die Oracle-Datenbank 12*c* hat viele automatische Performance-Diagnose-Empfehlungen, unter anderem den „Automatischen Datenbank Diagnose Monitor" (ADDM), den Real-Time ADDM und den „Vergleichszeitraum Compare Period ADDM". Der ADDM-Problem-Report zeigt bei der Analyse festgestellte Probleme der letzten Stunde an und macht Vorschläge zur Behebung. Der Spot ADDM ist ein neues Advisory, das automatisch angestoßen („angetriggert") wird, wenn die Datenbank anfängt, Performance-Probleme zu bekommen, und der versucht, die Ursache des Problems zu identifizieren. Das Ergebnis wird in das Automatic Workload Repository (AWR) gespeichert.

In 12*c* sind Daten aus dem Real-Time SQL Monitoring, Datenbank-Operations-Monitoring und dem Real-Time ADDM Report ebenfalls im AWR abgespeichert. Dadurch hat der Administrator die Möglichkeit, die Performance der Ausführung der Datenbank-Abfragen nachträglich zu überprüfen. Mit dem Real-Time SQL-Monitoring kann die Performance einer speziellen SQL-Abfrage über die Zeit untersucht werden.

Der ADDM analysiert Performance-Probleme direkt nach dem Auftreten des Events. Das spart Zeit und Ressourcen, indem der ADDM das Problem reproduziert. Der ADDM sollte deshalb von DBAs erst benutzt werden, wenn Performance-Probleme auftreten. Der SQL-Monitor-, ASH- und AWR-Report zeigen jetzt Statistiken von „In-Memory"-Operationen. ADDM und AWR sind standardmäßig eingeschaltet und werden über Datenbank-Parameter gesteuert *(siehe Listing 1)*.

System-Statistiken müssen vorhanden sein. Der ADDM benötigt für die I/O-Statistiken beispielsweise „average read time", die hardwarespezifischen Antwortzeiten für das Lesen eines Datenbankblocks. Normale Werte liegen zwischen 5.000 und 20.000 Mikrosekunden. Mit einem Script werden die System-Statistiken gesammelt und die Antwortzeit eines Lesevorgangs auf acht Mikrosekunden gesetzt *(siehe Listing 2)*.

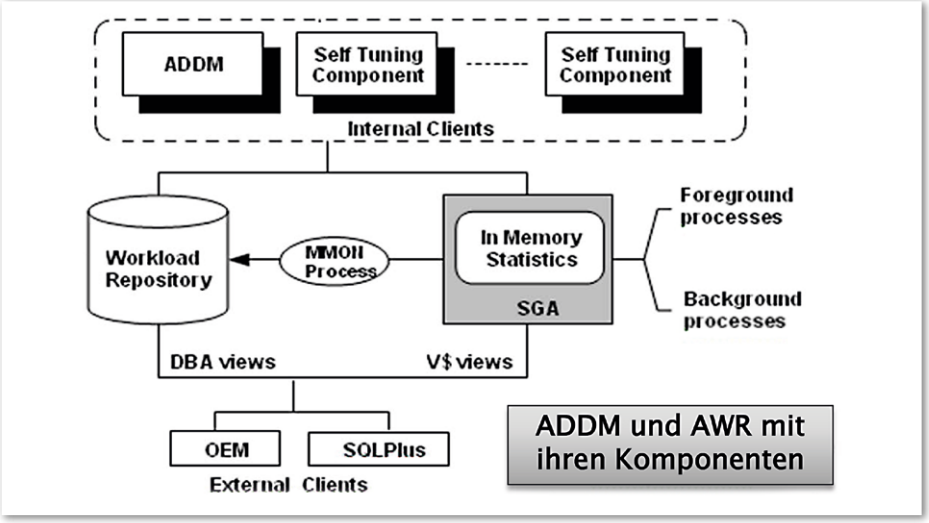


Abbildung 1: Komponenten-Überblick AWR und ADDM

```
timed_statistics = true
statistics_level =
[typical|all Control_manage-
ment_pack_access=diagnostic/
diagnostic+tuning
```

Listing 1

```
SQL>dbms_stats.gather_system_
stats(gathering_modIe=>'start');
SQL>dbms_stats.gather_system_
stats(gather_mode=>'stop');
SQL>Execute dbms_advisor.set_de-
fault_task_parameter(
'ADDM', 'DBIO_EXPECTED', 8000)
```

Listing 2

Feature	Speicher-ort	Sichtbar in Root	Sichtbar in PDB	PDB-Daten werden entfernt wenn „unplugged“
ASH	Root	Ja	Nur PDB-Daten	nein
ADDM-Daten	Root	ja	nein	nein
AWR-Daten	Root	ja	Nur PDB-Daten	nein
Optimizer-Statistikdaten	Root/PDB	ja	Auf PDB-Daten eingeschränkt	ja
Segment Advisordaten	Root/PDB	ja	Auf PDB-Daten eingeschränkt	ja

Tabelle 1: Wo AWR-, ASH- und ADDM-Performance-Daten in einer Container-Datenbank gespeichert sind und wer sie sehen kann, Auszug aus MOS.ID 1586256.1

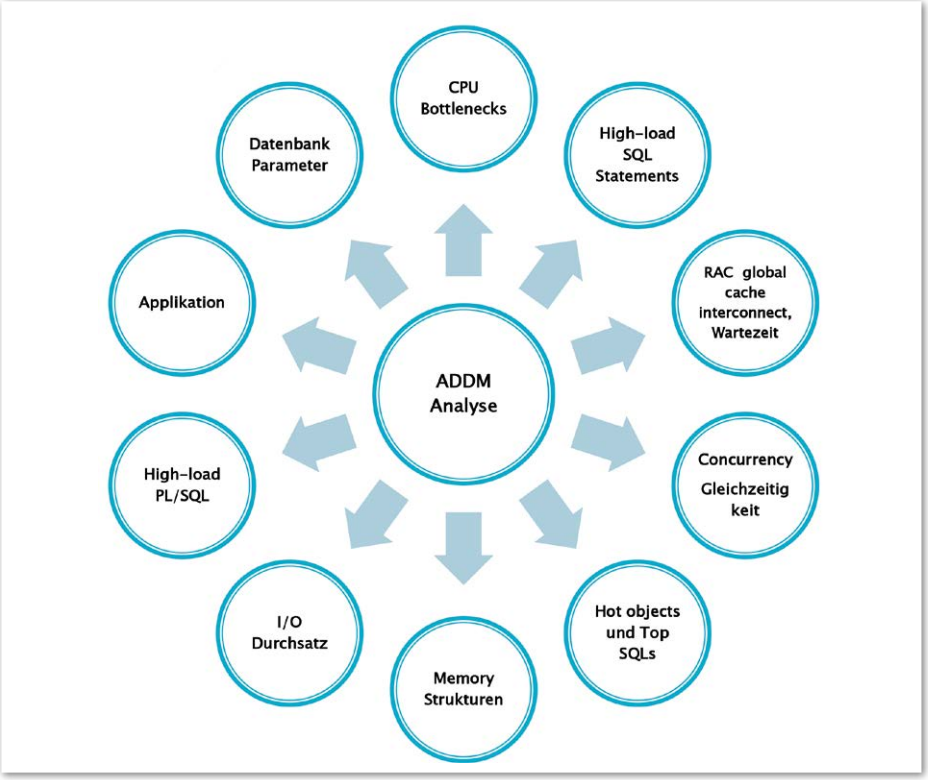


Abbildung 2: Problemtypen der ADDM-Analyse

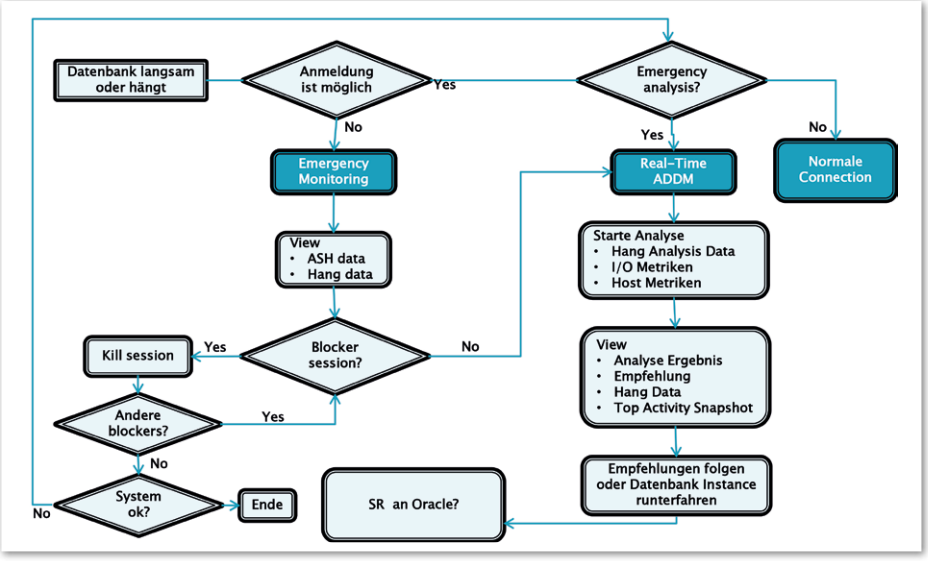


Abbildung 3: ADDM-Trigger-Auslöser in Anlehnung an die Oracle-Dokumentation

Per Default erfolgen die allgemeinen Datenbank-Statistiken als Snapshot in einem Intervall von sechzig Minuten. Diese werden im AWR des Users „SYS“ im Tablespace „SYSAUX“ gespeichert und acht Tage lang vorgehalten. ADDM ermittelt und analysiert diese Daten unmittelbar nach jedem „AWR“-Snapshot. Dabei werden die Top-25 „High Load“ SQL-Statements ebenfalls abgespeichert. Nach acht Tagen löschen die „Maintenance Tasks“, die schon bei der Installation der Datenbank aktiviert werden, automatisch die Snapshot-Statistiken im AWR. Die aktuelle Speicherplatzbelegung der Statistikdaten im „SYSAUX“-Tablespace wird mit der SQL-Prozedur „SQL>$ORACLE_HOME/rdbms/admin/awrinfo.sql.” nach den einzelnen Komponenten aufgeschlüsselt und als Report abgespeichert *(siehe Listing 3)*.

Die Informationen zu den Top 25 Sessions werden über das Diagnose-Werkzeug „Active Session History“ (ASH) im Sekundenbereich ermittelt. Statistiken und Reports zu den Statistiken können manuell zu jeder Zeit erstellt werden (siehe *Tabelle 1* und *Abbildung 1)*.

Der Prozess „Manageability Monitor“ (MMON) sammelt Snapshots und speichert sie im „SYSAUX“-Tablespace des Users „SYS“ ab. Die hier gesammelten Daten sind über DBA-Views ersichtlich. Die Komponente „Automatic Database Diagnostic Monitor“ (ADDM) interpretiert diese Daten zyklisch und macht Vorschläge, die über Advisory-Views oder über GRID Control (OEM) ausgewertet werden können.

Der „Manageability Monitor Lite Process“ (MMNL) unterstützt den „MMON“-Prozess, wenn der „ASH- Buffer“ voll ist oder ein AWR-Snapshot gezogen wird, indem er die Daten aus dem „ASH-Buffer“ in die AWR-Tabellen schreibt. Die Default-Parameter des AWR-Snapshot-Intervalls lassen sich von einer Stunde auf zehn Minuten reduzieren. Dadurch läuft auch die ADDM-Analyse jeweils anschließend an.

Real-Time ADDM

Der Real-Time ADDM analysiert die Performance ähnlich wie der reguläre ADDM, der nach jedem AWR-Snapshot angestoßen wird und die AWR-

Snapshots untersucht. Er liefert Diagnose-Ergebnisse und macht Vorschläge, um die Performance zu verbessern. Der Real-Time ADDM benutzt nicht die AWR-Snapshots, sondern die aktuellen Aktivitäten aus den SGA-Daten.

Der Anmelde-Modus an die Datenbank mit „SYSDBA"-Privileg ist abhängig von dem Status der Instanz. Es wird automatisch der „Diagnostic Modus" gewählt, wenn die Anmeldung an der Instanz nicht mehr möglich ist. Der „normale Modus" kommt zum Einsatz, wenn eine Anmeldung an der Instanz keine Probleme bereitet. Danach beginnt die Analyse mit der Sammlung der aktuellen Performance-Daten aller Datenbank-Instanzen aus der SGA. Die Daten werden für die Systeme beziehungsweise Bereiche untersucht, die blockiert oder lahmgelegt worden sind, weil sie starke Zugriffskonflikte auf lokale oder globale Ressourcen aufweisen.

Bereiche mit ungewöhnlich hoher Datenbank-Aktivität werden aufgezeigt und Analyse-Ergebnisse geliefert. Nachdem die Analyse beendet ist, lassen sich die „Real-Time ADDM Findings" abfragen.

In 12*c* werden die Pluggable Datenbanken (PDBs) vom ADDM mit den Analyse-Ergebnissen einzeln aufgeführt. Der Real-Time ADDM findet heraus, dass eine Session beendet („killed") werden muss, weil ein „Logon Trigger" eine Tabelle schließen will, die bereits geschlossen ist. Real-Time ADDM identifiziert Lang-Läufer-Abfragen, die fast 100 Prozent der Ressourcen benötigten. Dafür wird die Tabelle „v$longops" ausgelesen. In *Abbildung 2* sind weitere ADDM-Kriterien aufgeführt.

Der Oracle Enterprise Manager zeigt auf der zentralen Advisory-Seite die ADDM Analyse Tasks. Es kann eine neue Analyse gestartet sowie ein aktueller oder historischer Bericht mit den Analyse-Ergebnissen und Empfehlungen angesehen werden. ADDM-Reports werden mit „SQL> @?/rdbms/admin/addmrpt.sql" gestartet. *Listing 4* zeigt das Ansehen.

Ziel der ADDM-Analyse ist es, bei gleichen Ressourcen von CPU und Hardware die Datenbank-Zeit zu reduzieren, um mehr User-Anfragen in der gleichen Zeit unterstützen zu können und den

```
**************************************************
(1a)  SYSAUX usage - Schema breakdown  (dba_segments)
**************************************************
Total SYSAUX size        878.3 MB   ( 43% of
2,048.0 MB MAX with AUTOEXTEND ON )
SchemaSYS     occupies   605.6 MB   (69.0% )
SchemaSYSMAN  occupies    78.2 MB   ( 8.9% )
SchemaMDSYS   occupies    66.8 MB   ( 7.6% )
SchemaXDB     occupies    60.8 MB   ( 6.9% )
SchemaWMSYS   occupies    29.6 MB   ( 3.4% )
SchemaSYSTEM  occupies    14.8 MB   ( 1.7% )
SchemaORDDATA occupies    13.6 MB   ( 1.5% )
SchemaCTXSYS  occupies     3.6 MB   ( 0.4% )
SchemaEXFSYS  occupies     3.6 MB   ( 0.4% )
SchemaDBSNMP  occupies     1.3 MB   ( 0.1% )
SchemaORDSYS  occupies     0.4 MB   ( 0.0% )
```

Listing 3

```
SQL> DBMS_ADDM.GET_REPORT function:
SQL> execute DBMS_ADDM.GET_REPORT ( task_name IN VARCHAR2
     RETURN   CLOB);
SQL> SET LONG 1000000 PAGESIZE 0;
SQL> SELECT DBMS_ADDM.GET_REPORT(:tname) FROM DUAL;
SQL> Select DBMS_ADDM.REAL_TIME_ADDM_REPORT() from dual;
REAL_TIME_ADDM_REPORT
--------------------------------------------------------
<report db_version="12.1.0.1.0" inst_count="1" cpu_cores="4"
hyperthread="N" con
```

Listing 4

```
DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE
start_snap_id => 270, end_snap_id => 280,
baseline_name => 'peak baseline', dbid => 3310949047, expiration =>
30);
```

Listing 5

```
ADDM-Bericht für Task 'TASK_1015'
Analysezeitraum
---------------------------------------------------------------
AWR-Snapshot-Bereichvon 980 bis 989.
Zeitraum beginnt um 05.11.14 07:00:43
Zeitraum endet um 05.11.14 15:00:52
Ergebnis 1: Hard Parse aufgrund von literaler Auslastung
Auswirkung ist ,2aktive Sessions, 26,89\% der Gesamtaktivitat.
---------------------------------------------------------------
SQL-Anweisungen wurden wegen der Verwendung von Literalen
nicht gemeinsam benutzt.
Empfehlung 1: Anwendungsanalyse
  Geschätzter Vorteil betragt,2 aktive Sessions, 26,89\% der
  Gesamtaktivitat.
Aktion
  Überprüfen Sie die Anwendungslogik auf die mögliche
  Benutzung von Bindevariablen anstelle von Literalen.
Aktion
  Alternativ können Sie den Parameter "cursor_sharing" auf
  "force" festlegen.
Begründung
  Es wurde ermittelt, dass mindestens 49 SQL-Anweisungen mit
  FORCE MATCHING SIGNATURE 18143935563635607236
```

Listing 6: Auszug eines ADDM-Reports

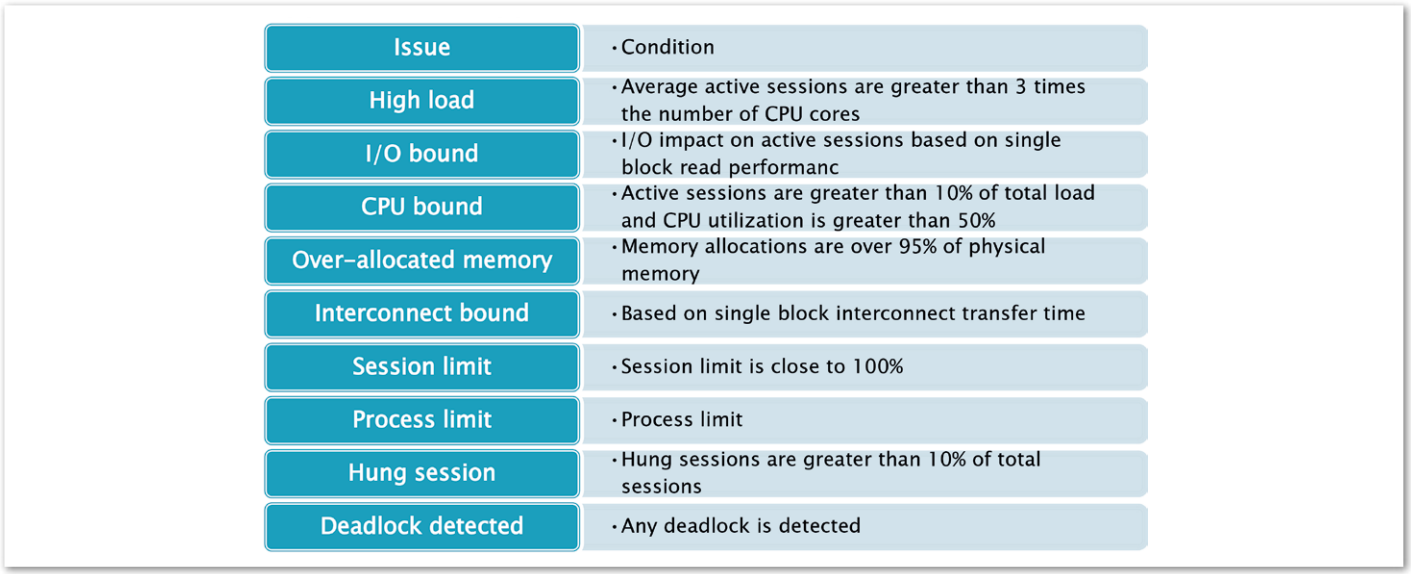


Abbildung 4: Emergency-ADDM-Ablauf in Anlehnung an die Oracle-Dokumentation

Durchsatz der Datenbank zu erhöhen. *Abbildung 2* zeigt, welche Bereiche vom ADDM untersucht werden.

Dabei werden „top down" die Hauptkonsumenten der Ressourcen analysiert. Bei dem ersten gefundenen Symptom wird nach der Ursache geforscht. ADDM benutzt dabei die „DB Time", um Performance-Engpässe zu finden. In den oben angezeigten Bereichen *(siehe Abbildung 2)* sind in einem ADDM-Report die System-Ressourcen angezeigt, die signifikante Zeit verbraucht haben. Danach wird absteigend nach der Zeit sortiert. Die DB-Time-Statistik wird nach der „Time-Model-Statistik" bemessen. Darunter versteht man die Gesamtzeit aller User-Prozesse, die in ihrer Datenbank-Abfrage entweder aktiv arbeiten oder aktiv warten; sie ist ein Indikator für den „Total Instance Workload". Die folgenden Datenbank-Views werden zusätzlich ausgewertet:

- v$sys_time_model
- v$sess_time_model
- v$sysstat
- v$sesstat
- v$active_session_history

Die Statistik der aktiven Sessions wird einmal pro Sekunde gesammelt. Der Durchschnitt aller aktiven Sessions in der Datenbank zum Zeitpunkt „X" wird berechnet mit „Average Active Sessions=Total DB-Time / Wall Clock Elapsed Time". Die Datenbank-Zeit „DB Time" beinhaltet nur Vordergrund-Sessions inklusive CPU-, I/O- und Wartezeit. Die „Idle Wait Time" wird dabei ausgeschlossen. Datenbank-Views mit ADDM-Information sind:

- DBA_ADVISOR_FINDINGS
- DBA_ADVISOR_FINDING_NAMES
- DBA_ADVISOR_RECOMMENDATIONS
- DBA_ADDM_FINDINGS
- DBA_ADDM_FDG_BREAKDOWN
- DBA_ADDM_INSTANCES
- DBA_ADDM_TASKS
- DBA_ADDM_SYSTEM_DIRECTIVES
- DBA_ADDM_TASK_DIRECTIVES

Real-Time ADDM Connection Modi

In Abhängigkeit vom Datenbank-Status benutzt ADDM zwei unterschiedliche Connection Modi im Enterprise Manager. Die normale JDBC-Connection führt umfangreiche Performance-Analysen durch. Wenn die Datenbank steht und eine normale Anmeldung nicht möglich ist, wird der Diagnose-Anmelde-Modus mit einer Latch-losen Verbindung durchgeführt. Dabei werden extreme „Hang-Situationen" analysiert ohne exklusiven Zugriff auf geschützte Ressourcen und Strukturen, wenn keine JDBC-Anmeldung mehr möglich ist.

Vor der Version 12*c* mussten die Datenbank oder der Rechner durchgestartet werden. Jetzt schlägt ADDM Lösungen ohne Neustart der Datenbank vor und unterstützt bei der Ursachenforschung. Die Analyse-Informationen stammen in Echtzeit aus der SGA und beinhalten blockierende Sessions, Deadlocks, Hangs, Shared Pool Connections, Objekt Locks, Top-Aktivitäten und weitere Ausnahmesituationen.

Real-Time ADDM-Trigger oder Spot ADDM

Ab der Version 12*c* ermittelt der Real-Time ADDM proaktiv Performance-Probleme. Der „Manageability Monitor" (MMON) sammelt automatisch alle drei Sekunden Performance-Statistiken ohne Lock und Latches aus der SGA und benutzt dabei die „In-Memory Data". Es erfolgt eine Diagnose über Performance-Peaks in der Datenbank.

Der MMON-Slave-Prozess kreiert den ADDM-Report und speichert ihn in das AWR-Repository ab. Die Metadaten des Reports stehen in den „DBA_HIST_REPORT"- und „ADDM"-Views. Der MMON-Prozess prüft die Statistiken und triggert eine Real-Time ADDM-Analyse, wenn Vorfälle oder Probleme auftreten *(siehe Abbildung 3)*.

Real-Time ADDM Trigger-Kontrolle

Damit der Automatic Trigger nicht zu viele System-Ressourcen verbraucht, prüft der Real-Time ADDM das Zeitdelta zwischen zwei Real-Time ADDM-Reports. Ist

das Intervall kleiner als fünf Minuten, so wird verhindert, dass ein neuer Report entsteht. In einer RAC-Umgebung arbeitet die „Oracle RAC Kontrolle“. Der Real-Time ADDM-Report läuft nur auf einer Instanz, dabei generiert der „MMON Slave Process“ ein Lock. Es findet eine Prüfung daraufhin statt, ob der ADDM-Report schon auf einer anderen Instanz läuft. Ist diese negativ, wird der Report generiert.

Repeated Trigger

Repeated Trigger sind automatische Trigger für Performance-Probleme, die angestoßen werden, wenn die Anzahl der aufgetretenen Probleme 100 Prozent oder mehr im Vergleich zu dem vorherigen Report innerhalb der letzten 45 Minuten beträgt. Zum Beispiel gibt es aktuell TOP 8 aktive SQL-Sessions. Innerhalb der nächsten 45 Minuten sind 16 Top aktive Sessions aufgetreten oder es gibt ein neues Problem, das nicht in den letzten 45 Minuten aufgetreten ist. Dann wird ein neuer Report generiert.

Compare ADDM

Zur Vorbereitung der Performance-Analyse sollte man sogenannte „Baseline AWR Snapshots“ erstellen, die das zu untersuchende System bei normalem Zustand oder in Augenblicken guter Performance widerspiegeln. Baselines können mit dem Oracle Enterprise Manager oder mit dem AWR Baseline Package erstellt werden. Oracle empfiehlt, in Zeiten guter Performance eine Baseline als Referenz zu ziehen, damit diese in Zeiten schlechter Performance herangezogen werden können *(siehe Listing 5)*.

Der Vergleichs-ADDM wird berechnet, indem man die genannte Baseline-Periode mit dem gewählten Zeitraum vergleicht. Es lassen sich auch zwei beliebige Zeiträume miteinander vergleichen. Dabei wird folgende Vorgehensweise priorisiert:

Es wird nach System-Änderungen im Bereich „Datenbank-Konfiguration“ sowie nach Workload- und SQL-Änderungen gesucht, die das Performance-Problem hervorgerufen haben können.

Eine ADDM-Analyse wird für die beiden Zeiträume gestartet und der Unterschied zwischen den beiden Perioden gemessen. Schließlich werden die Ergebnisse nach vordefinierten internen Regeln verglichen. Eine Vergrößerung des Datenbank-Parameters „SGA_TARGET” kann den I/O-Durchsatz erhöhen und somit den „WORKLOAD”. Es werden „init.ora”-Parameter-Änderungen herausgefunden und Lösungen vorgeschlagen. Vorschläge für Änderungen in der Hard- und Software sind das Ergebnis *(siehe Listing 6)*.

Fazit

Folgende Regeln sind zu beachten:

- Kenne dein System mit normaler Performance
- Mache ein Konzept für Performance-Messung und -Tuning
- Habe normale AWR-Snapshots als Referenz
- Habe normale ADDM-Empfehlungen als Referenz
- Emergency ADDM statt Reboot

Weitere Informationen

1. Angelika_Gallwitz-Oracle_12c_Automatic_Performance_Diagnostics-Praesentation.pdf
2. 2012-K-DB-Angelika_Gallwitz- Oracle_Statistik_Reports_Leitfaden_zur_Auswertung_-Praesentation.pdf

Angelika Gallwitz
angelika@Gallwitz-IT.de

apps associates
extreme expertise

Business Intelligence
Managed Services
Custom Development
E-Business Suite

Wir sind dabei
APEX connect
by DOAG

Apps Associates GmbH

Flughafenring 11
D-44319 Dortmund

Tel.: +49 231 22 22 79-0

www.appsassociates.com

ORACLE Platinum Partner

Metadata Service Respository – sehen, lernen, verstehen

Carsten Wiesbaum, esentri AG

Früher oder später wird jeder, der sich im Umfeld der Oracle Fusion Middleware bewegt, einen Kontaktpunkt mit dem Metadata Service Repository haben – bewusst oder unbewusst. Oft lernt man dabei nur eine Facette dieses Bausteins der Oracle Fusion Middleware kennen, etwa beim ADF UI Customizing oder beim Deployment von SOA Composites. Dabei ist das Metadata Service Repository die Grundlage für eine Vielzahl von Funktionen im gesamten Oracle-Fusion-Middleware-Stack.

Der Artikel zeigt, was das Metadata Service Repository (MDS) genau ist, wo es innerhalb der Fusion Middleware eingesetzt wird und wie man selbst von der Funktionalität profitieren und sie im eigenen Projektalltag und Systemdesign einbinden kann.

In der heutigen Software-Entwicklung spielen Metadaten eine große Rolle. Nahezu jede Anwendung verwendet Metadaten in der einen oder anderen Form. Als Metadaten bezeichnet man Daten über Daten, im Bereich der Enterprise-Anwendungen auch Daten über Anwendungen. Große Teile und ganze Enterprise-Anwendungen werden nicht mehr in einer Programmiersprache entwickelt, sondern über Metadaten konfiguriert. Diese Metadaten ermöglichen erst das Verhalten bestehender Komponenten beziehungsweise deren Funktion.

Prominente Beispiele für die Verwendung von Metadaten in der heutigen IT sind zum Beispiel JSF-Seiten, WSDL-Dokumente oder auch XML-Schemata (XSDs). Wenn man die Bedeutsamkeit von Metadaten für heutige Enterprise-Anwendungen betrachtet, wird schnell klar, dass ein zentraler Ansatz zur Verwaltung und Verwendung von Metadaten notwendig ist. Innerhalb des Oracle-Fusion-Middleware-Stacks übernimmt das MDS genau diese Funktion.

Metadata Service Repository als Bindeglied der Oracle Fusion Middleware

Das MDS stellt eine zentrale Komponente der Oracle Fusion Middleware dar. Seine Aufgabe ist die zentrale Verwaltung der Konfigurationsdateien von Komponenten sowie Metadaten zu Applikationen. Die im MDS abgelegten Daten werden durch den kompletten Oracle-Fusion-Middleware-Stack verwendet. So verwalten SOA Suite und Oracle Identity Manager ihre Konfigurationen über das MDS. Auch die SCA-Anwendungen der SOA Suite sind im MDS-Repository abgelegt. So finden sich hier BPEL-Prozess, Business Rules und DVMs.

Darüber hinaus basieren viele Funktionen des Application Development Framework (ADF) auf der Nutzung des MDS-Repository. Gerade die Funktionen rund um Customizing und Personalization verwenden das Repository exzessiv. Während der Design-Time werden die ursprünglichen Metadaten, die die Funktion der ADF-Anwendung beschreiben, in Oracle JDeveloper erzeugt. Die generierten Metadaten reichen durch alle Schichten des ADF-MVC-Modells von Business Components bis hin zu den JSF-Seiten.

Beim Customizing einer Anwendung generiert der entsprechende Entwickler weitere Metadaten. Diese werden je nach Bedarf in der laufenden Anwendung verwendet, um das Aussehen und Verhalten der Applikation an bestimmte Benutzer oder Benutzergruppe anzupassen. Zudem kann dem Benutzer die Möglichkeit eingeräumt werden, über „Personalization“ weitere Metadaten zur Laufzeit zu generieren, die wiederum im MDS abgelegt sind.

Auch viele Funktionen der Fusion Applications und Oracle WebCenter bauen auf den Funktionen des MDS-Repository auf. Sie verwenden unter anderem ADF und SOA Suite als Basis-Technologie und bieten fortgeschrittene Customization, Personalization sowie „Design Time at Run Time (DT@RT)“-Funktionen an *(siehe Abbildung 1)*. Diese Beispiele zeigen die starke Integration des MDS-Repository in der Fusion Middleware.

Technische Grundlagen

Im Grunde besteht das MDS-Repository aus drei Komponenten – einem Datenspeicher, einer Runtime Engine und einigen MBeans zur Interaktion mit dem MDS-Repository über das WebLogic Sever Scripting Tool (WLST) oder Enterprise Manager Fusion Middleware Control (EM). Der Datenspeicher kann sowohl Datei- als auch Datenbank-basiert genutzt werden *(siehe Abbildung 2)*. Für eine Entwicklungsumgebung reicht die Speicherung der Metadaten auf Dateisystem-Ebene völlig aus. Oracle JDeveloper besitzt standardmäßig eine entsprechende Verzeichnis-Struktur, die während der Entwicklung verwendet werden kann.

Für den produktiven Einsatz sollte die Datenbank-basierte Variante zum Einsatz kommen. Sie bietet eine bessere Performance sowie die Versionierung der abgelegten Artefakte und verwendet Datenbank-Transaktionen, um die Integrität der abgelegten Metadaten zu gewährleisten. Auch für den Cluster-Betrieb bietet diese Variante Vorteile, so können mehrere Ins-

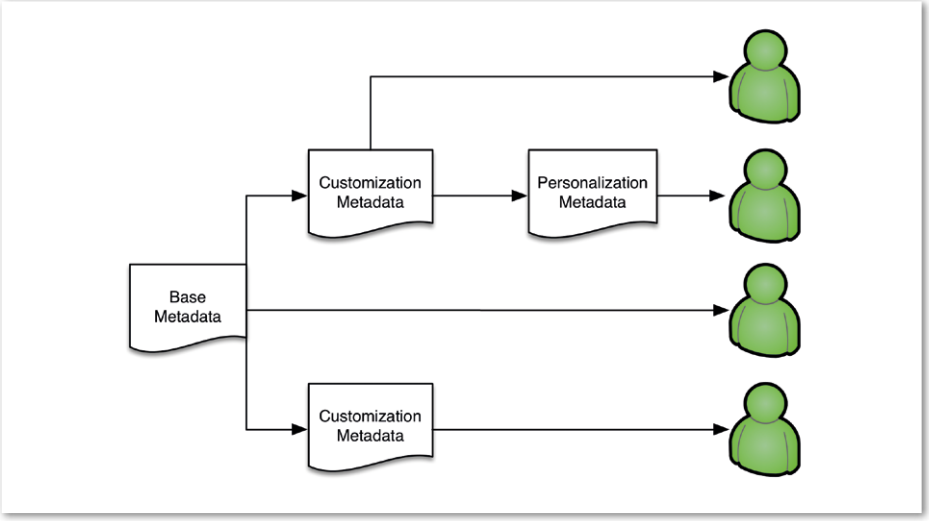


Abbildung 1: ADF Customization und Personalization

tanzen einer ADF-Anwendung das gleiche MDS-Repository verwenden.

Wenn in einer Anwendung Customization-Änderungen durchgeführt werden, werden diese durch Polling auch auf anderen Instanzen der Anwendung sichtbar. Allerdings muss bei der Nutzung der Datenbank-basierten Variante natürlich auch zusätzlicher Aufwand betrieben werden. Es muss beispielsweise geklärt sein, wie viele Repositories notwendig sind und welche Purging-Strategie für die gewählte Architektur genutzt wird.

Innerhalb des MDS-Repository sind die Metadaten in einer Verzeichnis-Struktur abgelegt. Jede abgelegte Datei ist einem bestimmten Pfad zuzuordnen. Zusätzlich kann das MDS-Repository noch in mehrere Partitionen aufgeteilt sein. Partitionen ermöglichen die logische Trennung von Metadaten entsprechend ihrem Nutzen. Im Hinblick auf ADF-Anwendungen gilt es etwa als Best Practice, für jede Anwendung eine eigene Partition anzulegen. Durch diese Strategie sind die Metadaten einzelner Anwendungen voneinander getrennt.

Die zweite Komponente des MDS-Repository ist die Runtime Engine. Diese bietet dem Fusion-Middleware-Stack und dessen Anwendungen die Interaktion mit den Metadaten innerhalb des MDS-Repository. Sie kontrolliert den Zugriff auf darin enthaltene Metadaten und ermöglicht das Importieren von Metadata-Archiv-Dateien (MAR). Außerdem beinhaltet die MDS-Runtime-Engine einen integrierten Cache, der den Zugriff auf häufig angefragte Metadaten optimiert.

Bei der letzten Komponente handelt es sich um MDS-spezifische MBeans, die zur Verwaltung des MDS verwendet werden können. Sie ermöglichen das Entleeren des MDS-Runtime-Engine-Cache, den Im- und Export von anwendungsspezifischen Metadaten sowie die Konfiguration einiger MDS-Repository-Funktionen *(siehe Abbildung 3)*.

Das MDS-Repository in der Service-Entwicklung

Bisher wurde aufgezeigt, wie tief das MDS-Repository im Fusion-Middleware-Stack verankert ist. Gerade im Bereich der SOA Suite besteht jedoch auch die Möglichkeit, es aktiv zu nutzen und in seine Service-Architektur einzubinden. Zunächst einmal kann das MDS-Repository als zentrales

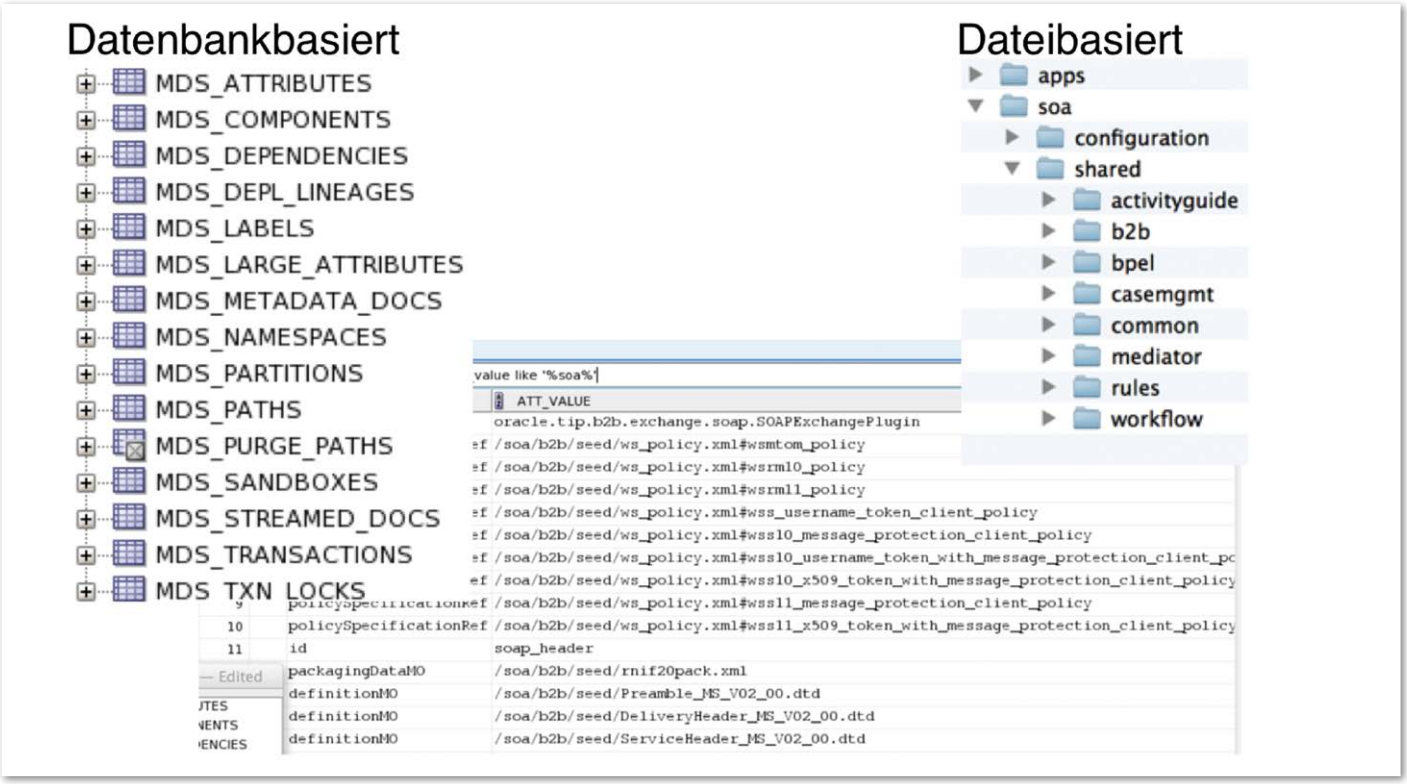


Abbildung 2: Datenbank- und Datei-basierte Variante des MDS-Repository

Repository für die Serviceschnittstellen-Definitionen zum Einsatz kommen *(siehe Abbildung 4)*.

Hier sind WSDL-, XSD- und XSLT-Dateien in einem zusätzlichen Projekt abgelegt. Dieses wird dann in das MDS-Repository eingerichtet. Die Dateien sind anschließend unterhalb des MDS-Repository im „apps"-Ordner zu finden. SCAs und deren Komponenten können diese Artefakte dann über „oramds:/apps/Verzeichnis_1/Verzeichnis_2/fooService.wsdl" referenzieren.

Eine weitere Möglichkeit, das MDS-Repository zu nutzen, ist die Verwendung von Domain-Value-Maps (DVMs) und Cross References (XRef). Innerhalb der Service-Entwicklung werden diese Dateien verwendet, um ein Mapping zwischen verschiedenen Werten zu ermöglichen. Sowohl DVMs als auch XRef -Dateien werden beim Deployment eines SCA im MDS-Repository abgelegt. Die darin enthaltenen Werte lassen sich dann über Xpath-Funktionen oder innerhalb von Java-Code abfragen.

Ein Anwendungsbeispiel

Für MDS-Zugriff auf eine DVM mit Java wird das MDS-Repository verwendet, um konfigurierbare Werte im MDS in Form von DVMs abzulegen. Dies ermöglicht es, die Werte innerhalb einer Anwendung auch während der Laufzeit zu ändern, und das ohne ein erneutes Deployment der Anwendung auf dem Server. Als konkretes Beispiel dient hier die Ablage und Konfiguration von Fehler-Informationen. Ziel ist es, Informationen wie Fehlercode, Fehlername und Fehlerbeschreibung als DVM im MDS abzulegen und aus Java darauf zuzugreifen. Um dies zu erreichen, wurde die DVM aus *Abbildung 5* im MDS abgelegt. *Abbildung 6* zeigt die Ablage-Struktur innerhalb des MDS.

Aus einer anderen Komponente werden die Fehler-IDs in der ersten Spalte geliefert. Um dem Benutzer der Anwendung eine möglichst aussagekräftige Fehlermeldung anzuzeigen, werden die übrigen Werte mit der ID aus der DVM abgefragt. *Listing 1* zeigt, wie Java-Werte von der DVM „Errors.dvm" innerhalb des MDS abgefragt werden.

Im Code werden die relevanten Werte aus der DVM abgefragt und in einer Fehlermeldung dem Anwender ange-

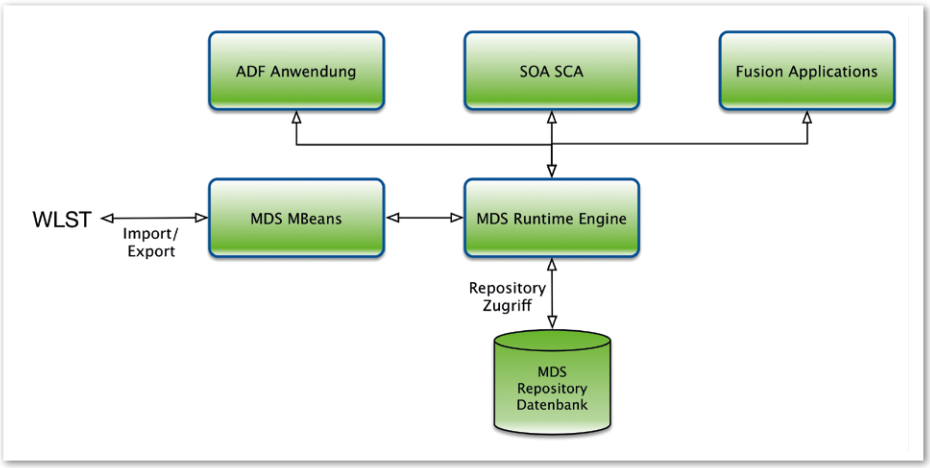


Abbildung 3: MDS Repository Runtime Architecture

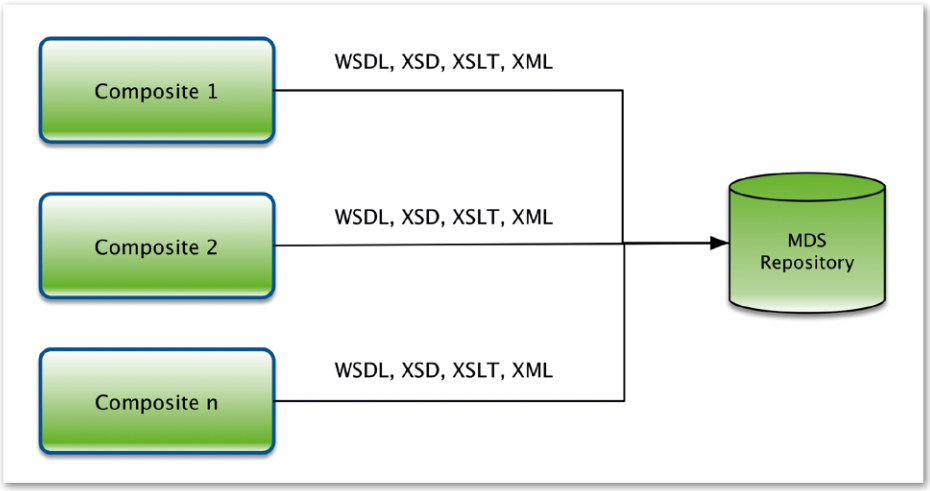


Abbildung 4: Verwendung des MDS-Repository in der Service-Entwicklung

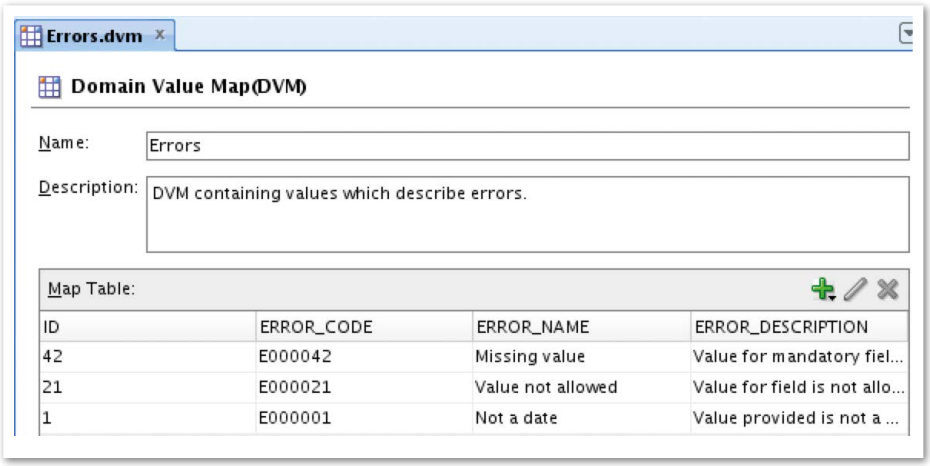


Abbildung 5: DVM mit Fehler-Informationen

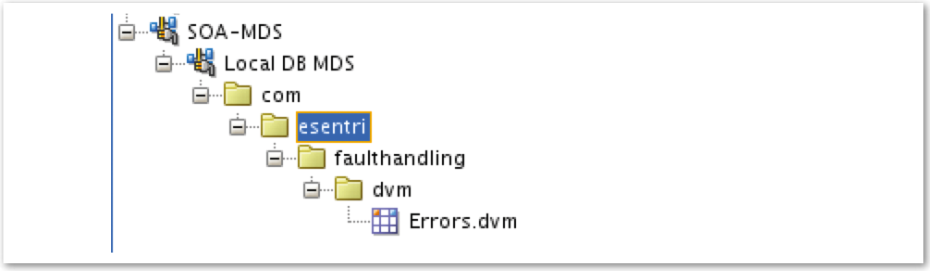


Abbildung 6: Im MDS-Repository abgelegte DVM

```
DVMManager dvmManager = DVMManagerFactory.getDVMManager();

String errorCode;
String errorName;
String errorDescription;

try {
   errorCode = dvmManager.lookupValue(
               "oramds:/com/esentri/faulthandling/dvm/Errors.dvm"),
               "ID", "42",
               "ERROR_CODE", null);

   errorName = dvmManager.lookupValue(
               "oramds:/com/esentri/faulthandling/dvm/Errors.dvm"),
               "ID", "42",
               "ERROR_NAME", null);

   errorDescription = dvmManager.lookupValue(
               "oramds:/com/esentri/faulthandling/dvm/Errors.dvm"),
               "ID", "42",
               "ERROR_DESCRIPTION", null);
} catch (DVMException e) {
   // Handle Exception
}
```

Listing 1

zeigt. Sollte sich ein bestimmter Wert ändern, ist dieser lediglich in der DVM innerhalb des MDS-Repository zu ändern. Beim nächsten Zugriff auf die DVM verwendet die Anwendung den neuen Wert.

Fazit

Das MDS-Repository wird oftmals als eine einfache Komponente innerhalb des Fusion-Middleware-Stacks dargestellt. Wie beschrieben, stellt sie jedoch einen zentralen Bestandteil für die Funktionsweise und Interaktion vieler Oracle-Produkte dar, sowohl während der Entwicklung als auch zur Laufzeit. Daher ist bei der Entwicklung von Enterprise-Applikationen auf Basis von Oracle-Produkten ein grundlegendes Verständnis der MDS-Repository-Architektur und der damit verbundenen Prinzipien von großem Wert.

Carsten Wiesbaum
carsten.wiesbaum@esentri.com

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

Alexander Alers
Henriette Cebulla
Klaus Debus
Christina Eicher
Richard Haberkorn
André Pittel
Per Reinholdt
Gerd Schoen
Hermann von Kobylinski
Thorsten Ahlbrecht
Artur Czudnochowski
Martin Götz
Houssem Eddine Hariz
Christian Höcherl

Firmenmitglieder DOAG

Kundan Lamsal, Cleverbridge AG
Friedrich Mayerhofer, GRZ IT Center GmbH
Klaus Rieck, S&H EDV-Beratung GmbH
Erwin Rossgoderer, ISE GmbH
Martin Waiblinger, TransnetBW GmbH

Neumitglieder SOUG

Richard Koller, CSS Versicherung
Laszlo Hadhazy, Accarda AG
Christina Pavalache, Diso AG
Karl-Heinz Sütterlin, Microsoft Schweiz GmbH
Jacques Kostic, Trivadis AG
Thomas Rieder, Atos AG
Dr. Stefan Meyer, Prolicense Schweiz GmbH
Tobias Schaller, Zürich Kantonalbank
Marco Kurmann, bbi software ag

08.06.2015
Regionaltreffen NRW
Stefan Kinnen, Andreas Stephan
regio-nrw@doag.org

09.06. - 10.06.2015
APEXConnect 2015 | Düsseldorf
DOAG Geschäftsstelle
office@doag.org

09.06.2015
Regionaltreffen Hamburg/Nord
Jan-Peter Timmermann
regio-nord@doag.org

09.06. - 11.06.2015
DOAG 2015 Business Solutions Konferenz | Darmstadt
DOAG Geschäftsstelle
office@doag.org

10.06.2015
Regionaltreffen NRW (MySQL)
Stefan Kinnen, Andreas Stephan
regio-nrw@doag.org

11.06.2015
Regionaltreffen Dresden/Sachsen
Helmut Marten
regio-sachsen@doag.org

11.06.2015
SOUG SIG
SOUG Geschäftsstelle
sekretariat@soug.ch

12.06.2015
DOAG Webinar: ORACLE ODA 12c News
Johannes Ahrends, Christian Trieb
sig-database@doag.org

15.06.2015
Regionaltreffen NRW
Stefan Kinnen, Andreas Stephan
regio-nrw@doag.org

15.06.2015
Regionaltreffen Osnabrück/Bielefeld/Münster
Andreas Kother, Klaus Günther
regio-osnabrueck@doag.org

16.06.2015
DOAG 2015 Datenbank | Düsseldorf
DOAG Geschäftsstelle
office@doag.org

18.06.2015
Regionaltreffen Rhein-Main
Thomas Tretter
regio-rhein-main@doag.org

18.06.2015
Regionaltreffen Nürnberg/Franken – APEX für Einsteiger
André Sept, Martin Klier
regio-franken@doag.org

22.06.2015
Regionaltreffen Halle/Leipzig
Matthias Reimann
regio-halle@doag.org

23.06.2015
Regionaltreffen München/Südbayern
Franz Hüll, Andreas Ströbel
regio-muenchen@doag.org

23.06.2015 – 24.06.2015
Berliner Expertenseminar mit Boris Gloger und Stephan LaRocca
Cornel Albert
expertenseminare@doag.org

23.06.2015
Regionaltreffen Bremen
Ralf Kölling
regio-bremen@doag.org

25.06.2015
Regionaltreffen Rhein-Neckar
Frank Stöcker
regio-rhein-neckar@doag.org

26.06.2015
Regionaltreffen Würzburg
Oliver Pyka
regio-wuerzburg@doag.org

30.06.2015
DOAG Oracle & SAP Day
Jörg Hildebrandt
sig-sap@doag.org

09.07.2015
Regionaltreffen Nürnberg/Franken
André Sept, Martin Klier
regio-franken@doag.org

09.07.2015
Regionaltreffen Trier/Saar/Luxemburg
Bernd Tuba, Holger Fuchs
regio-trier@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

Titelfoto: © lightwise / 123RF.com
Foto S. 39: © chuyu / 123RF.com
Foto S. 57: © hoboton / fotolia.com
Foto S. 61: © ra2 studio / fotolia.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

Stellenausschreibung

DOAG Dienstleistungen GmbH

Erfahrungs- und Wissensaustausch zu Oracle Produkten sind unsere Ziele. Um den Austausch zwischen Oracle-Anwendern zu fördern, hat die DOAG eine attraktive, vielschichtige Informationsplattform aufgebaut, die ihren Platz sowohl bei Veranstaltungen, als auch in Print- und Online-Medien einnimmt. Inzwischen begleitet die DOAG mehr als 100 Fachveranstaltungen pro Jahr mit bis zu 2000 Teilnehmern, die sowohl produktbezogene als auch branchen- bzw. prozessspezifische Themen vermitteln und die komplette Oracle-Produktpalette abdecken. Das geballte Know-how der Community gibt die DOAG in diversen Publikationen, Fachbüchern und Fachzeitschriften sowie Online-Medien weiter. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. ist in Deutschland die einzige Interessenvertretung der Anwender von Oracle-Produkten. Die DOAG Dienstleistungen GmbH führt die Geschäftsstelle des Vereins.

Zur Verstärkung unseres Teams suchen wir in Vollzeit oder Teilzeit eine/n

Leiter/in Vertrieb (m/w)

Sie sind in einem kleinen Team verantwortlich für den Verkauf von Anzeigen in unseren Medien und der Ausstellungsplätzen auf unseren Veranstaltungen sowie die Gewinnung von Sponsoren. Daneben unterstützen Sie uns auf unseren größeren Veranstaltungen in ganz Deutschland vor Ort mit der Organisation der Ausstellungen. Sie verfügen über eine fundierte betriebswirtschaftliche Ausbildung oder vergleichbare Qualifikationen und haben bereits Berufserfahrung im Vertrieb in der Softwareindustrie.

Sie bringen ausgeprägte Teamfähigkeit und Kommunikationskompetenz mit, arbeiten mit hoher Kundenorientierung und sind mobil für Dienstreisen in Deutschland.

Ihre Qualifikation:

- Fundierte Betriebswirtschaftliche Ausbildung oder vergleichbare Qualifikation
- mindestens 5 Jahre Berufserfahrung im Vertrieb
- sehr gute Kenntnisse des IT Marktes und der Oracle Produkte
- Genauigkeit und Zuverlässigkeit
- eine strukturierte und selbstständige Arbeitsweise
- Flexibilität, Teamfähigkeit und Stressresistenz

Entwickler/in für den Bereich Internet und Backend auch Teilzeit möglich

Sie sind in einem kleinen Team verantwortlich für den Betrieb und die Weiterentwicklung unserer IT-Systeme. Daneben unterstützen Sie uns auf unseren größeren Veranstaltungen in ganz Deutschland vor Ort. Sie verfügen über einen Hochschulabschluss der Informatik oder vergleichbare Qualifikationen und haben Berufserfahrung in vergleichbaren Positionen. Sie haben gute Kenntnisse im Bereich Softwaremanagement. Die aktuellen Trends bei Neuen Medien haben Sie im Blick. Zusätzlich bringen Sie vertiefte Fachkenntnisse in Webtechnologie sowie Webplattformen mit. Ihre Arbeitsweise ist geprägt von einem strukturierten und methodischen Vorgehen. Sie verfügen über Vermittlungskompetenz zwischen den Fachanforderungen und der EDV-technischen Umsetzung. Sie bringen ausgeprägte Teamfähigkeit und Kommunikationskompetenz mit und arbeiten mit hoher Kundenorientierung und sind mobil für Dienstreisen in Deutschland.

Ihre Qualifikation:

- Abgeschlossenes Studium oder vergleichbare Abschlüsse bzw. Berufserfahrung
- Sehr gute Kenntnisse PHP, SQL, HTML, CSS, JavaScript, jQuery, Bootstrap
- Erfahrungen mit dem CMS Typo 3 wünschenswert
- Gutes gestalterisches Gespür, stilsicheres Arbeiten und hoher Qualitätsanspruch an die eigenen Arbeitsergebnisse
- Genauigkeit und Zuverlässigkeit
- Eine strukturierte und selbstständige Arbeitsweise
- Flexibilität, Teamfähigkeit und Stressresistenz
- Bereitschaft, sich in administrativen Themen einzuarbeiten

Wir bieten Ihnen einen vielseitigen, zukunftssicheren Arbeitsplatz in einem kreativen jungen Team, eine attraktive Vergütung mit einem fixen und einem variablen Anteil und einen modernen, ansprechenden Arbeitsplatz in der Hauptstadt Berlin.

Ihre Bewerbung senden Sie bitte per E-Mail unter Angabe Ihres möglichen Eintrittstermins und Ihrer Gehaltsvorstellungen an bewerbung@doag.org. Für Rückfragen steht Ihnen der IT-Verantwortliche Herr Jürgen Pittorf (technik@doag.org) sowie der Geschäftsführer Herr Fried Saacke gerne zur Verfügung.

DOAG Dienstleistungen GmbH Tempelhofer Weg 64, 12347 Berlin | +49 (0) 30-6090218-22 | bewerbung@doag.org | www.doag.org

Wir sind jetzt auch in Dänemark für Sie da.

Mit unserer neuen Niederlassung in Kopenhagen ist Trivadis jetzt an 14 Standorten in Dänemark, Deutschland, Österreich und in der Schweiz immer direkt in Ihrer Nähe. So kommen Sie noch schneller zu ganzheitlichen und zukunftsweisenden IT-Lösungen mit Oracle- und Microsoft-Technologien. Von der Pflege und dem Support für Ihre Software- und BI-Lösungen über den hochverfügbaren Betrieb Ihrer IT-Infrastruktur bis hin zum Outsourcing und innovativen Cloud-Services – wir sind Ihr kompetenter Partner. www.trivadis.com | info@trivadis.com

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

trivadis
makes IT easier.