*Ralf Kölling*
*Vorstand und Leiter der Regionalgruppe Bremen/ Nordniedersachsen*

*Liebe Mitglieder der Deutschen ORACLE-Anwendergruppe, liebe Leserinnen und Leser,*

als Erstes möchte ich Ihnen im Namen des neugewählten DOAG-Vorstands ein gesundes und erfolgreiches neues Jahr wünschen. Nach einer – trotz des wirtschaftlichen Umfelds – sehr gelungenen DOAG 2009 Konferenz + Ausstellung wollen wir in die Zukunft blicken und das Gewicht der Anwendergruppen gegenüber Oracle durch eine verbesserte internationale Zusammenarbeit verstärken.

Anfang des Jahres ist „die Software“ wieder einmal ins Bewusstsein der breiten Öffentlichkeit gerückt. Durch einen Fehler bei der Verarbeitung der Jahreszahl konnten Millionen Verbraucher ihre Bank- oder Kreditkarten nicht zum elektronischen Bezahlen benutzen. Dieses Beispiel beweist, welche Auswirkungen nicht ausreichende Tests haben können, bei denen kleine Fehler oder Unachtsamkeiten unentdeckt bleiben. Es unterstreicht die Notwendigkeit eines professionellen Vorgehens. Dieses zeigt sich in der Software-Entwicklung vor allem im Einsatz moderner Tools und Verfahren; Anregungen und Best Practices dazu finden Sie in der vorliegenden Ausgabe. In diesem Sinne wünsche ich Ihnen einen guten Start ins neue Jahrzehnt,

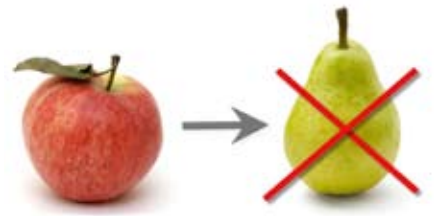
Ihr

PS: Wenn Sie dieses Heft in den Händen halten, hat die Europäische Kommission hoffentlich ihre Entscheidung zur Übernahme von Sun durch Oracle gefällt und damit eine Phase der Unsicherheit beendet. Im Laufe dieses Prozesses hat auch die EU – durch eine Anhörung zu diesem Thema – die Expertise der internationalen Usergruppen anerkannt. Die DOAG hat schon im Vorfeld auf die erwartete Entscheidung reagiert und zusammen mit sechs Java-Usergroups die Interessengemeinschaft der deutschen Java User Groups e.V. (iJUG) gegründet.

*„Für uns Entwickler ist es immer wichtig zu erfahren, wie die Kunden mit den Produkten umgehen ...“*

*Interview mit Richard Sarwal, Senior Vice President Oracle Corp. und weltweit verantwortlich für das Product Development.*

*Seite 8*



*Migration von Forms über Apex zu Oracle ADF*

*Viele Entwicklungsabteilungen beschäftigen sich damit, die Implementierungen aus bestehenden Applikationen zu migrieren oder zu modernisieren.*

*Seite 16*

iJUG

*DOAG ist Gründungsmitglied des Interessenverbund der Java User Groups e.V. (iJUG)*

*Vor dem Hintergrund der geplanten Sun-Übernahme durch Oracle hat die DOAG zusammen mit sechs Java-Usergroups aus Deutschland den iJUG-Interessenverbund gegründet.*

*Seite 61*

# Spotlight

### Dienstag, 17. November 2009

*Die Mitgliederversammlung wählt turnusgemäß einen neuen Vorstand (siehe Seite 69). Er setzt sich aus vier bestehenden und vier neuen Mitgliedern zusammen.*

### Freitag, 20. November 2009

*Die DOAG 2009 Konferenz + Ausstellung geht zu Ende. Alle DOAG-Aktiven sind überwältigt von dem großartigen Erfolg mit stabilen Teilnehmerzahlen in wirtschaftlich schwierigen Zeiten.*

### Montag, 30. November 2009

*Im Rahmen eines Treffens werden die Weichen zur Gründung des Interessenverbund der Java User Groups e.V. (iJUG) gestellt (siehe Seite 61).*

### Dienstag, 1. Dezember 2009

*Die DOAG ist mit einem eigenen Stand auf dem itSMF-Kongress vertreten. Cornel Albert aus dem Office und Vorstand Stefan Kinnen führen viele interessante Gespräche über das Thema „ITIL“ und knüpfen neue Kontakte.*

### Mittwoch, 9. Dezember 2009

*Die Expertenseminare „Oracle Streams“ und „Oracle Streams Commander“ mit Praktikum begeisterten die zwölf Teilnehmer, weil aufgrund der kleinen Gruppe ein intensiver Kurs möglich war. Im Rahmen der Abendveranstaltung ging es dann nach Berlin-Mitte zu einer Führung durch das Haus am Checkpoint Charlie, dem bekanntesten früheren Grenzübergang zwischen West und Ost. Im Anschluss daran fand ein Abendessen bei einem gemütlichen Italiener statt. Alle Teilnehmer freuen sich schon auf weitere Expertenseminare.*

### Freitag, 11. Dezember 2009

*Der neu gewählte Vorstand definiert auf seiner ersten Sitzung konkrete Meilensteine für 2010 und legt die Verantwortlichkeiten der einzelnen Vorstände fest.*

### Mittwoch, 16. Dezember 2009

*Nach mehreren vielfältigen Regionaltreffen mit spannenden Themen lässt die Regionalgruppe NRW das Jahr 2009 mit einem Adventstreffen ohne Vorträge ausklingen. Neben interessanten Gesprächen wurden zahlreiche Ideen für das Jahr 2010 gesammelt.*

### Freitag, 1. Januar 2010

*Die DOAG wünscht allen Mitgliedern und Interessenten ein erfolgreiches neues Jahr!*

### Montag, 11. Januar 2010

*Das Team der DOAG Dienstleistungen GmbH bespricht im Rahmen eines Workshops die Organisation zahlreicher Events in 2010, darunter die neue Frühjahrs-Konferenz „DOAG 2010 Applications“ und den „OAUG Connection Point 2010“ bewährte Veranstaltungen wie die „DOAG Logistik & SCM 2010“, die „ITIL & Betrieb 2010“ und insbesondere die „DOAG 2010 Konferenz + Ausstellung“.*

### Freitag, 22. Januar 2010

*Die Vorbereitungen für die diesjährige Beiratssitzung laufen auf Hochtouren. Von 18. bis 20. Februar 2010 werden alle DOAG-Aktiven bei einem gemeinsamen Treffen viele neue Vereinsaktivitäten ausarbeiten und festlegen.*

# Die DOAG in der Presse

**Die nachfolgenden Ausschnitte reflektieren die Einschätzung der Fach- und Wirtschaftspresse zu bestimmten Themen über Oracle; die Veröffentlichungen geben nicht die Meinung der DOAG wieder und sind auch nicht im Vorfeld mit der DOAG abgestimmt. Lediglich die Zitate einzelner DOAG-Vorstände geben die Meinung der DOAG wieder.**

**Computerwoche vom 24. November 2009:**

### Anwender ärgern sich über neue Service-Plattform

Anfang November hat Oracle seinen Online-Support auf eine neue Basis gehievt. Seitdem häufen sich die Klagen. Die Plattform sei zu langsam und teilweise nicht erreichbar, kritisieren viele Anwender.

„Was Oracle mit der neuen Service-Plattform My Oracle Support geliefert hat, war grauenhaft", kritisierte ein betroffener Anwender auf der Jahreskonferenz der Deutschen Oracle Anwendergruppe Mitte November in Nürnberg. Die Probleme seien eigentlich Anlass genug, die Supportgebühren zu kürzen, lautete sein von den Kollegen laut beklatschtes Fazit. Der Unmut der Oracle-Kunden ist groß. Karl-Heinz Urban, Support-Spezialist von Oracle bemühte sich auf der Anwenderkonferenz, die Vorzüge der neuen Online-Plattform zu erklären. Doch dem Manager schlug am Ende seines Vortrags eine Welle der Kritik entgegen. Im Publikum fielen Ausdrücke wie „katastrophal" und „unerträglich", als die aktuellen Probleme zur Sprache kamen.

Am 6. November hatte Oracle sein altes Support-Portal „Metalink" abgeschaltet und My Oracle Support (MOS) in Betrieb genommen. Trotz monatelanger Vorbereitung häuften sich seitdem die Klagen. Anwender berichteten, sie könnten sich auf der neuen Support-Plattform nicht einloggen. Außerdem sei die Performance des Service-Portals nicht zufriedenstellend, kritisierten die Kunden. Darüber hinaus gebe es massive technische Probleme: Schaltflächen des Portals würden nicht funktionieren, Service-Requests ließen sich nicht an Oracle übermitteln.

Ein Grund für die massiven Pannen scheint die in Oracles neuem Support-Portal integrierte Flash-Technik von Adobe zu sein. Das Servicewerkzeug verlangt den Flash-Player in der Version 9.0.115 oder höher. Aber auch das HTML-Interface, das der Hersteller optional anbietet, weil viele Unternehmen aus Sicherheitsgründen Flash-Inhalte blocken, scheint nicht fehlerfrei zu funktionieren.

„Wir haben ernste Probleme", schrieb ein Oracle-Kunde in einem offiziellen Forum des Softwareherstellers. Manchmal sei es nicht möglich, sich einzuloggen. Wenn es dann einmal geklappt habe, könnten keine Service-Anfragen abgeschickt werden. Der Prozess breche aus unerfindlichen Gründen immer wieder an unterschiedlichen Stellen ab. „Es ist ein Desaster", ergänzte ein anderer Anwender. Flash-Screens würden sich nicht vollständig aufbauen, einzelne Buttons funktionierten nicht und die ganze Seite funktioniere extrem langsam. Es sei kaum zu glauben, dass ein Unternehmen, das hochverfügbare Clusterlösungen herstelle, ein derart fehlerhaftes Online-Portal abliefere. „Das ist ein Fiasko und absolut inakzeptabel für ein Unternehmen wie Oracle", bestätigte ein ebenfalls betroffener Kunde. „Mein Unternehmen gibt jedes Jahr Millionen von Dollar allein für den Oracle-Support aus."

My Oracle Support biete den Anwendern eine Reihe von Vorteilen, versuchte Oracle-Manager Urban die neue Plattform zu verteidigen. Mit Health-Checks ihrer Systeme und Patch-Empfehlungen könnten Administratoren ihre Oracle-Umgebungen einfacher und effizienter verwalten. Der Umbau des Service-Portals sei notwendig geworden, um die im Laufe der Zeit durch zahlreiche Übernahmen unübersichtliche Support-Landschaft in einem einheitlichen System zu konsolidieren, begründete Urban die Strategie des Softwarekonzerns.

In Sachen Support war Oracle in den zurückliegenden Jahren bereits öfter mit seinen Kunden aneinander geraten. Beispielsweise hatten die Anwender kritisiert, Oracle würde seine Support-Prozesse zu stark standardisieren8 und sei damit nicht mehr in der Lage, auf die individuellen Service-Anforderungen der einzelnen Kunden einzugehen. Außerdem würden komplexe langwierigere Anfragen in Oracles weltweiter Service-Organisation rund um den Globus weitergereicht. Dabei fehle den Support-Mitarbeitern teilweise das Knowhow, auf spezifische Probleme des deutschen Markts eingehen zu können.

Grundsätzlich befürworten Anwendervertreter jedoch ein Online-basierendes Support-Portal. Beispielsweise hatten die DOAG-Verantwortlichen in den vergangenen Jahren ihre Mitglieder wiederholt dazu aufgefordert, Informationen über ihre Oracle-Landschaft im Support-System zu hinterlegen und so die damit verbundenen Möglichkeiten auch auszuschöpfen. Beispielsweise müssten Anwender nicht wiederholt

grundlegende Basics erklären, wenn ein anderer Support-Mitarbeiter ihr Problem übernimmt. Viele Anwenderunternehmen haben anscheinend Bedenken, diese Daten preiszugeben. Sie befürchten offenbar, dass Oracle Zugriff auf sensible Firmeninformationen erhalte. Diese Befürchtungen entbehrten jedoch jeder Grundlage, versuchen die DOAG-Verantwortlichen die Vorbehalte ihrer Mitglieder zu entkräften. Außerdem könnten die Kunden genau kontrollieren, welche Daten an Oracle übermittelt würden. Es könne nicht sein, dass Anwender den Oracle-Support kritisierten, sich aber gleichzeitig den angebotenen Tools verweigerten, hieß es von Seiten der Anwendervereinigung.

Auch My Oracle Support hatten die DOAG-Verantwortlichen im Vorfeld wohlwollend beurteilt. Die Plattform biete ein zeitgemäßes Layout und einige positive Neuerungen wie beispielsweise überarbeitete Suchfunktionen sowie Web-2.0-Elemente. Doch um die Akzeptanz des Support-Tools zu verbessern, müssen die Oracle-Verantwortlichen zügig die Probleme des neuen Portals beheben. Den Anwendern bleibt indes nichts anderes übrig, als zu warten. Einen anderen Weg wird es nicht geben, stellten die Oracle-Verantwortlichen klar. Zwar räumte Urban auf der DOAG-Konferenz die Probleme ein und versicherte den Kunden, man arbeite mit Hochdruck an einer Lösung. Eine zwischenzeitliche Reaktivierung des Vorgängers Metalink schloss der Oracle-Manager allerdings kategorisch aus. „Metalink ist abgeschaltet und bleibt abgeschaltet."

**databasepro online vom 23. November 2009:**

### DOAG Konferenz 2009 – Nürnberger Erfolgsmodell

Das im letzten Jahr neu eingeführte Konzept war auch für die DOAG 2009 Konferenz + Ausstellung wieder ein großer Erfolg. Mehr als 340 praxisnahe Vorträge und Keynotes deckten an drei Konferenztagen alle Produktbereiche von Oracle ab. Da die Teilnehmerzahl – rund 2.000 vermeldete der Veranstalter – in für Fachkonferenzen schwierigen Zeiten in etwa auf Vorjahresniveau gehalten werden konnte, zeigt zudem, dass Programm, Termin und Ort inzwischen etabliert sind. Und daran möchte man festhalten: Bis einschließlich 2012 hat die Deutsche Oracle-Anwendergruppe (DOAG) das Nürnberger Congress Centrum fest gebucht.

Dass die Zahl der Aussteller von rund 60 auf ca. 50 gesunken ist, erklärt man mit „natürlicher Fluktuation" und liegt natürlich auch daran, dass auch in den letzten 12 Monaten wieder ehemals eigenständige Firmen durch Zukäufe in Oracle und damit auch in deren Ausstellungsstand integriert wurden.

Der Wert des Programms lag einmal mehr auch an der Auswahl der Sprecher. So konnten neben Sessions, die direkt von Oracle-Mitarbeitern gehalten wurden, auch jede Menge Beiträge von unabhängigen Spezialisten und Beratern besucht werden. Sehr gut kam auch die Keynote von Oracle Deutschland Geschäftsführer Jürgen Kunz an, der ausführlich zu aktuellen Strategien und Marktentwicklungen Stellung nahm. Dass darin nur wenig zur bevorstehenden Sun-Einverleibung und vor allem auch zur Zukunft von MySQL gesagt werden konnte, bleibt zwar als Wermutstropfen, ist jedoch angesichts der bevorstehenden bzw. noch nicht gefällten kartellrechtlichen Entscheidung seitens der EU nachvollziehbar. Eine baldige Lösung ersehnt nicht nur das Oracle-Management, sondern die gesamte in Nürnberg anwesende Anwendergemeinde.

**Computerwoche online vom 11. November 2009:**

### Oracle integriert ERP und CRM

Mit den „Fusion Applications" will Oracle seine bestehenden ERP- und CRM-Lösungen zusammenführen und durch ein einheitliches, modular aufgebautes Produkt ersetzen. Damit unterstreicht der Hersteller den Nutzen von Service-orientieren Architekturen (SOA), die vor allem seit Beginn der Rezession in der Kritik standen.

Die Fusion Applications sind Teil eines groß angelegten Entwicklungsprojekts, das Oracle nach der Übernahme der ERP-Hersteller J.D. Edwards und Peoplesoft, sowie dem CRM-Anbieter Siebel angestoßen hatte. Ursprünglich wollte der Datenbankprimus die besagte Business-Suite schon 2008 auf den Markt bringen. Nun nimmt das Projekt endlich konkretere Formen an: Die übernommenen ERP- und CRM-Lösungen wurden in ein modulares Produkt zusammengeführt, das nächstes Jahr auf den Markt kommen soll. Laut Christian von Stengel, Senior Director Application Sales bei Oracle Deutschland, sind die neuen Business-Anwendungen auf einem guten Weg: „Die Funktionen sind alle komplett neu entwickelt und das Produkt ist bereits code-complete." Zudem würden bereits Hunderte Beta-Kunden an den Tests teilnehmen.

Nach dem Hype standen die Service-orientierten Architekturen (SOA) immer wieder in der Kritik. Vor allem in Zeiten der Krise wurden die Diskussionen über den geschäftlichen Nutzen von SOA immer lauter. Anfang des Jahres schrieb beispielsweise die renommierte Analystin Anne Thomas Manes von der Burton Group in ihrem Blog, dass die Rezession SOA den Todesstoß versetzt habe. Statt eines Heilsbringers habe sich SOA in den meisten Unternehmen zu einem „großen gescheiterten Projekt" entwickelt, urteilt sie. Nachdem Firmen Millionen in das Konzept investiert hätten, ständen IT-Systeme nicht besser da als zuvor. Die mit dem Konzept verbundenen Versprechen seien einfach nicht eingelöst worden, so die Expertin.

Dieser Ansicht können Oracle-Experten indes wenig abgewinnen. So bauen die neuen Business-Applikationen auf eine Service-orientierte Architektur auf, die die Integration von ERP-, CRM- und weiteren Systemen in komplexen Umgebungen, etwa bei weltweit agierenden Großunternehmen, deutlich vereinfachen soll. Das Kernstück der Plattform bildet dabei die „Oracle AIA" (Application Integration Architecture).

Weitere Pressestimmen zur DOAG finden Sie unter http://www.doag.org/presse/spiegel

*Dr. Dietmar Neugebauer, Vorstandsvorsitzender der DOAG, Richard Sarwal und Michael Pfautz, Leiter der Special Interest Group Database*

# „Für uns Entwickler ist es immer wichtig zu erfahren, wie die Kunden mit den Produkten umgehen …“

**Im Rahmen der DOAG 2009 Konferenz + Ausstellung sprachen die beiden Leiter der Special Interest Group Database, Christian Trieb und Michael Pfautz, sowie der DOAG-Vorstandsvorsitzende Dr. Dietmar Neugebauer mit Richard Sarwal, Senior Vice President Oracle Corp. und weltweit verantwortlich das für Product Development, über aktuelle und zukünftige Oracle-Produkte.**

*Was sind Ihre Position und Ihre Verantwortung bei Oracle?*

**Sarwal**: Ich bin Senior Vice President im Product Development und verantwortlich für das Management aller Oracle-Produkte von der Datenbank über die Middleware bis hin zu den Business Applications. Außerdem bin ich verantwortlich für die Pakete „Management“ und „Diagnostik“ der Datenbank. Hinzu kommen die Bereiche „Globalisierung“, „Built-Engineering“ und „Dokumentation“ für alle Produkte.

*Was sind aus Ihrer Sicht die besten Features der neuen Version von Enterprise Manager Grid Control?*

**Sarwal**: Die aktuelle Version 10.2.0.5 enthält zahlreiche Neuerungen beim Patching und Provisioning für die Datenbank und Middleware, außerdem haben wir das Management für die Virtualisierung, das Application-Testing sowie das Real-Application-Testing hinzugefügt.

*Wann wird die Version 11g von Enterprise Manager Grid Control verfügbar sein?*

**Sarwal**: Die Linux-Version ist im März 2010 verfügbar, die anderen Plattformen folgen sechs bis acht Wochen später.

*Warum ist der zeitliche Abstand zwischen der neuen Datenbank-Version und der entsprechenden Version von Enterprise Manager Grid Control so groß?*

**Sarwal**: In der Regel liegen hier rund drei Monate dazwischen, weil wir diese Zeit benötigen, um die entsprechenden neuen Funktionalitäten zu integrieren.

*Welcher Application-Server kommt bei der Version 11g von Enterprise Manager Grid Control zum Einsatz?*

**Sarwal**: Der Weblogic Server.

*Wird die Version 11g von Enterprise Manager Grid Control einen eigenen Scheduler enthalten?*

**Sarwal**: Wir nutzen in Enterprise Manager Grid Control Teile des Datenbank-Schedulers, wobei dieser um einige Funktionen erweitert ist.

*Wie ist der Zusammenhang zwischen dem SQL-Developer und Enterprise Manager Grid Control?*

**Sarwal**: Der SQL-Developer ist für Entwickler, während Enterprise Manager Grid Control für die Manageability zuständig ist. Es gibt hier nur ganz wenige Überlappungen.

*Gibt es Pläne für ein Tool zur Datenbank-Administration, das ohne Agent arbeitet?*

**Sarwal**: Das gibt es bereits. Bei einer Single-Database kann man „DB Control" einsetzen. Der Agent in der nächsten größeren neuen Version wird anstelle von C dann auf Java basieren, um alle Plattformen einfacher unterstützen können. Er wird dann komplett von Grid Control über OMS gemanagt und sich nur um die Applikationen kümmern, die tatsächlich vorhanden sind.

*Welche neuen Funktionalitäten sind noch geplant?*

**Sarwal**: Einige Schwerpunkte im nächste Major-Release, an denen wir gerade arbeiten, werden „Cloud Computing", „Self-Service Provisioning" und „Resource-Management" sein. Außerdem überarbeiten wir den Agent, um die Skalierbarkeit und die Hochverfügbarkeit zu verbessern. Die Liste mit allen neuen Punkten ist sehr lang.

*Wann wird die Windows-Version von 11g R2 verfügbar sein?*

**Sarwal**: Voraussichtlich im März 2010.

*Wird die Columnar-Compression zukünftig auch als Option der Enterprise Edition verfügbar sein oder bleibt dies ein Exadata-Feature?*

**Sarwal**: Soweit ich weiß, wird dies ein Exadata-Feature bleiben.

*Ab wann werden die Oracle-Produkte für Windows 7 zertifiziert sein?*

**Sarwal**: Wir haben damit bereits bei den aktuellen Releases begonnen.

*Umfasst diese Zertifizierung auch die Enterprise Edition 10.2.5?*

**Sarwal**: Die wird im April 2010 kommen.

*Wie kann ein Unternehmen Enterprise Grid Control am besten managen, wenn sowohl die Enterprise Edition als auch die Standard Edition und die Standard Edition One zum Einsatz kommen?*

**Sarwal**: Das Managen und Administrieren ist für alle Editions mit Enterprise Grid Control möglich. Die Funktionen hinsichtlich „Tuning" und „Monitoring" sind allerdings nur für die Enterprise Edition möglich. Das wird sich auch in Zukunft nicht ändern.

*Ab welcher Version ist das Patch Set Update (PSU) verfügbar?*

**Sarwal**: PSUs sind für Datenbanken ab Version 10*g* R2 verfügbar. Das Gleiche gilt auch für Fusion Middleware.

*Wann wird Oracle die Lizenzbedingungen für die Produkte unter VMware oder Xen ändern?*

Fotos: Wolfgang Taschner

**Sarwal**: Derzeit gibt es keine Pläne dafür.

*Welche Eindrücke haben Sie von der DOAG 2009 Konferenz + Ausstellung?*

**Sarwal**: Ich hatte schon viel davon gehört und bin besonders von der Qualität der technischen Vorträge beeindruckt.

*Konnten Sie hier einige Anregungen für Ihre Arbeit aufnehmen?*

**Sarwal**: Ich habe mit vielen Anwendern gesprochen und mir ihre Bedürfnisse angehört. Das ist mir sehr wichtig, da wir Entwickler etwas vom Tagesgeschäft der Anwender entfernt sind. Von daher ist es immer wichtig, zu erfahren, wie die Kunden mit unseren Produkten umgehen.

*Werden Sie im nächsten Jahr wiederkommen?*

**Sarwal**: Ja, gerne.

# Remote Debugging einer APEX-Anwendung

Carsten Czarski, ORACLE Deutschland GmbH

**Fast jeder Anwendungsentwickler kommt früher oder später in die Situation, dass die Prozesse nicht so funktionieren, wie sie sollen. Debugging kann dann bei der Fehlersuche helfen.**

Speziell bei Web-Anwendungen und damit auch im APEX-Umfeld reicht es jedoch nicht aus, den Debugging-Prozess aus einem Werkzeug heraus zu starten – Ausgangspunkt muss die normale Webseite sein, in welcher der Fehler auftritt. Man spricht hier von „Remote Debugging". Dieser Artikel zeigt, wie das mit APEX und dem Oracle SQL Developer funktioniert.

### Oracle SQL Developer

Der SQL Developer ist als SQL- und PL/SQL-Entwicklungsumgebung von Oracle ein mittlerweile recht verbreitetes Werkzeug. Es lässt sich kostenlos aus dem Oracle Technet herunterladen und bietet dem Entwickler wertvolle Hilfe beim Programmieren mit der Oracle Datenbank.

Damit das Remote Debugging mit dem SQL Developer funktioniert, müssen einige Privilegien vergeben werden. Zunächst ist sicherzustellen, dass das Parsing-Schema, in dem die Anwendung läuft, das Systemprivileg „DEBUG CONNECT SESSION" hat. Zusätzlich benötigt das Schema, unter dem die APEX-Datenbank-Verbindungen ablaufen, entweder das Systemprivileg „DEBUG ANY PROCEDURE" (für alle PL/SQL-Objekte) oder das DEBUG-Privileg am PL/SQL-Objekt, mit dem gearbeitet werden soll. Bei Verwendung des Apaches oder des neuen APEX J2EE Listeners ist dies typischerweise APEX_PUBLIC_USER, bei Verwendung des PL/SQL Embedded Gateways ANONYMOUS. Listing 1 zeigt Beispiele für die Privilegienvergabe.

```
-- DEBUG CONNECT SESSION muss
ans APEX Parsing Schema gegeben
werden
grant debug connect session to
SCOTT

-- DEBUG ANY PROCEDURE geht an
den APEX Session User
grant debug any procedure to
[APEX_PUBLIC_USER | ANONYMOUS]

-- DEBUG-Privileg kann auch
für einzelne Prozedur vergeben
werden
grant DEBUG ON SCOTT.PLSQLCODE
to [APEX_PUBLIC_USER | ANONY-
MOUS]
```

*Listing 1: Privilegienvergabe*

Zum Einrichten der Beispielapplikation erstellt man eine APEX-Anwendung mit einer leeren Seite. Dieser Seite fügt man dann einen Bericht hinzu. Als Debugging-Szenario soll der Bericht mit dynamischem SQL arbeiten – das Debugging wird sich dann auf das Generieren der SQL-Abfrage konzentrieren.

Wenn man PL/SQL-Code direkt in Application Express hinterlegt, kann kein Remote Debugging stattfinden; der Code muss dazu als PL/SQL-Objekt (Funktion, Prozedur oder Package) vorliegen. Dies ist ohnehin zu empfehlen: PL/SQL-Logik sollte stets in Packages hinterlegt werden, so dass APEX diese dann nur noch aufrufen muss. Die PL/SQL-Funktion in Listing 2 stellt demnach eine SQL-Abfrage zusammen. Ein Hinweis schon vorab: Die Funktion enthält einen kleinen Fehler, der mit dem Debugging gefunden werden soll.

```
create or replace function ge-
nerate_sql (
  p_tablename in USER_TABLES.
TABLE_NAME%TYPE
) return varchar2
is
  v_sql varchar2(32767) := ,‘;
begin
  v_sql := ,select ,;
  for col in (
    select column_name from
user_tab_columns
    where table_name = p_table-
name and data_type = ,VARCHAR2‘
  ) loop
    v_sql := v_sql || col.co-
lumn_name ||‘,. ,; -- FEHLER!!!
  end loop;
  v_sql := substr(v_sql, 1,
length(v_sql) - 2);
  v_sql := v_sql || , from , ||
p_tablename;
  return v_sql;
end;
```

*Listing 2: PL/SQL-Funktion zum Generieren einer SELECT-Anweisung*

Streng genommen öffnet diese Funktion auch ein SQL-Injection-Loch in der Anwendung. Das soll hier jedoch nicht weiter betrachtet werden. Nun fügt man der APEX-Seite eine Auswahlliste (P1_TABELLE) hinzu – damit kann man in der Anwendung eine Tabelle auswählen. Listing 3 zeigt die Wertelisten-Abfrage.

```
select TABLE_NAME d, TABLE_NAME
r from USER_TABLES order by 1
```

*Listing 3: Wertelisten-Abfrage*

Nicht vergessen darf man die Schaltfläche zum Absenden der Auswahl. Danach wird ein Bericht (SQL-Bericht, nicht interaktiv) erstellt. Als Berichtsquelle gibt man einfach nur „return generate_sql(:P1_TABELLE);" an. Hilfreich ist, noch eine Bedingung einzubauen; der Bericht soll nur gezeigt werden, falls P1_TABELLE „NOT NULL" ist. Wenn man die Anwendung dann startet, eine Tabelle auswählt und auf die Schaltfläche klickt, sollte das Ergebnis wie in Abbildung 1 aussehen.


*Abbildung 1: Die Funktion GENERATE_SQL funktioniert nicht*

### Debugging-Schritte

Nun geht es daran, die PL/SQL-Funktion GENERATE_SQL zu debuggen. Dazu startet man zuerst den SQL Developer, verbindet sich mit dem Parsing-Schema der Application-Express-Anwendung und sucht im Navigationsbaum die PL/SQL-Funktion (siehe Abbildung 2).

Mit einem Klick auf den Bleistift oben rechts lässt sich der Code editieren. Indem man auf die Zeilennummer klickt, setzt man vor einer gewünschten Zeile einen Haltepunkt (siehe Abbildung 3).

Beim Klick auf das „Compile"-Symbol mit dem schwarzen Pfeil wird die Funktion mit Debug-Informationen neu kompiliert. Anschließend geht es daran, den Remote-Debugger einzurichten. Dazu sucht man im Kontextmenü nach der Datenbank-Verbindung, ruft mit der rechten Maustaste das Kontextmenü auf und wählt „Remote Debug" aus. Im folgenden Dialog wählt man einen freien TCP/IP-Port aus und gibt den Namen beziehungsweise die IP-Adresse des Rechners, auf dem der SQL Developer läuft, an. Man merkt sich die Angaben und klickt anschließend auf OK (siehe Abbildung 4).

Daraufhin erscheint unten rechts im Run Manager der Hinweis, dass der SQL Developer einen Listener für das Remote-Debugging gestartet hat. Nun wechselt man wieder zu Application Express. Für das Debugging muss sich Application Express über das Netzwerk mit dem SQL Developer verbinden. Dazu ändert man den PL/SQL-Aufruf in der Berichtsdefinition, wie in Listing 4 dargestellt. Man trägt im Aufruf von dbms_debug_jdwp.connect_tcp den Rechnernamen und den TCP/IP-Port ein, den man im SQL Developer konfiguriert hat und behält im Hinterkopf, dass APEX sich hier zum SQL Developer verbindet. Der SQL Developer ist nun „Server" und APEX der „Client".

```
declare
  v_sql varchar2(32767);
begin
  -- 1. Hier findet die Verbindung zum Remote-Debugger statt
  dbms_debug_jdwp.connect_tcp(,192.168.2.1', 4000);

  -- 2. Ausführen der PL/SQL-Funktion im APEX-Kontext
  v_sql := generate_sql(:P1_TABLENAME);

  -- 3. Verbindung zum SQL Developer trennen
  dbms_debug_jdwp.disconnect;
  return v_sql;
end;
```

*Listing 4: PL/SQL-Berichtsquelle mit Debugging-Integration*

Jetzt speichert man den Bericht und startet die Seite. Man wird nun feststellen, dass die Seite nicht erscheint – der Browser wartet. Im SQL Developer ist jetzt allerdings der Debugger aktiv (siehe Abbildung 5).

Im Debugger kann man sich nun wie gewohnt im Einzelschrittmodus durch die PL/SQL-Funktion durcharbeiten – für den Browser sieht es so aus, als ob der Seitenaufbau nur sehr viel Zeit benötigt. Im Bereich „Daten" des SQL Developers kann man die Variablen verfolgen – hier ist besonders die Variable V_SQL von Interesse. Im Debugging kann man nun gut erkennen, dass die einzelnen Spaltennamen nicht (wie es sein soll) durch ein Komma getrennt

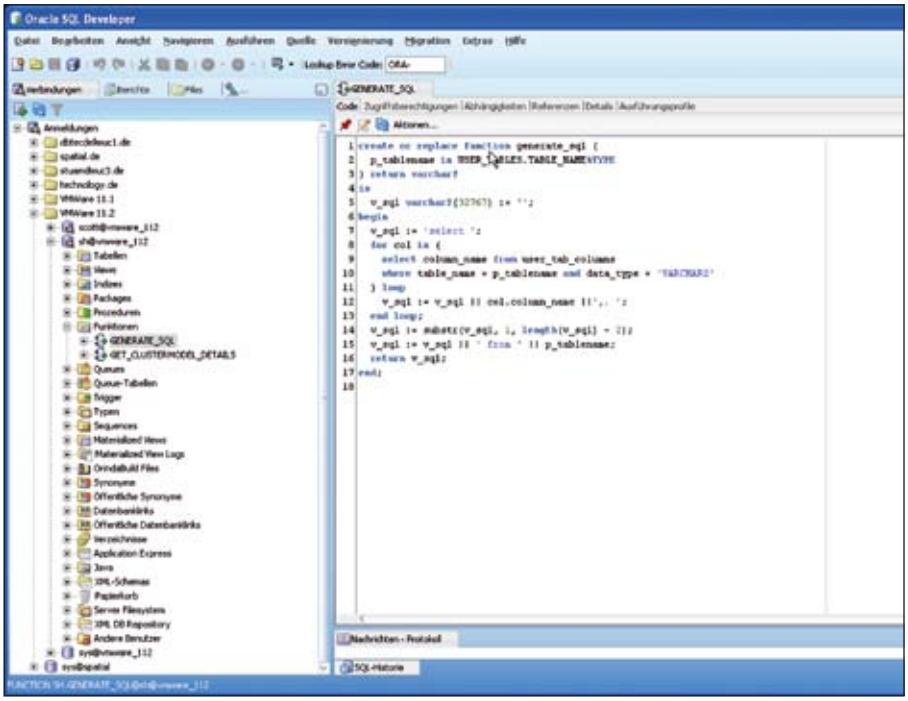


*Abbildung 2: Funktion GENERATE_SQL im SQL Developer*

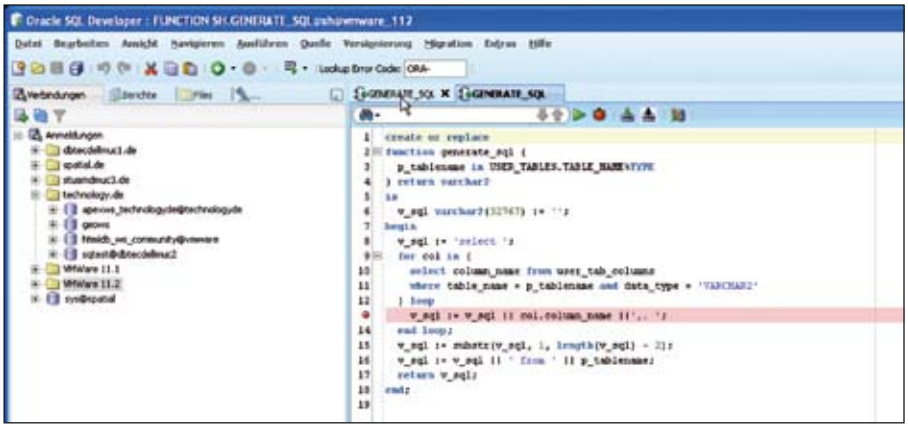


*Abbildung 3: Haltepunkt setzen*

werden, sondern vielmehr durch ein Komma und einen Punkt; das ergibt ein falsches SQL-Kommando.

Mit diesen Erkenntnissen ist es nun ein Leichtes, den Fehler im Code zu finden und zu korrigieren. Übrigens lassen sich die Inhalte der Variablen auch während des Debuggings ändern. Dazu setzt man einen Haltepunkt am Ende der Funktion und startet die APEX-Seite neu. Wiederum wartet der Browser und der Debugger ist aktiv. Im Bereich „Daten“ klickt man nun den Eintrag für die Variable V_SQL doppelt und ändert im sich dann öffnenden Fenster den Inhalt nach Belieben. Nach Abschluss des Debuggings erscheint der APEX-Bericht mit der im Debugger eingetragenen SQL-Anweisung.

## Fazit

Mit dem Remote-Debugging können APEX-Anwendungen „live“ debuggt werden; denkbar wäre sogar eine direkte Integration in die APEX-Anwendung. So könnte man in der Applikation zwei Anwendungsprozesse (siehe Listings 5 und 6) und die darin verwendeten Anwendungselemente SQLDEV_HOST sowie SQLDEV_PORT einrichten.

```
begin
  dbms_debug_jdwp.con-
nect_tcp(v(,SQLDEV_HOST‘),
v(,SQLDEV_PORT‘));
end;
```

*Listing 5: Anwendungsprozess „START_DEBUG“: Beim Laden - vor Header*

```
begin
  dbms_debug_jdwp.disconnect;
end;
```

*Listing 6: Anwendungsprozess „STOP_DEBUG“: Beim Laden – nach Footer*

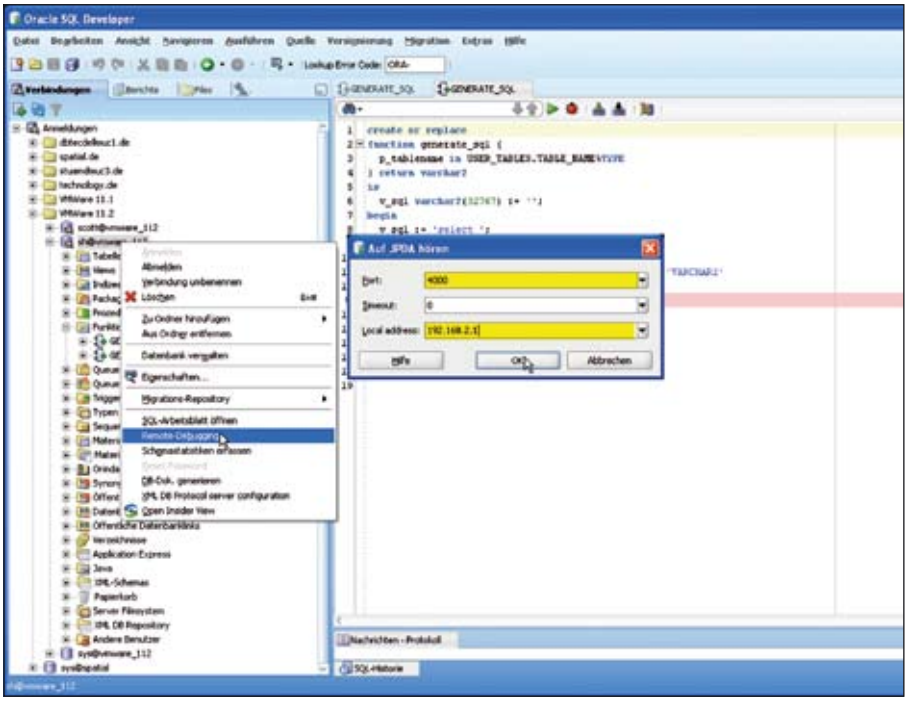


*Abbildung 4: Remote Debugging einrichten und starten*

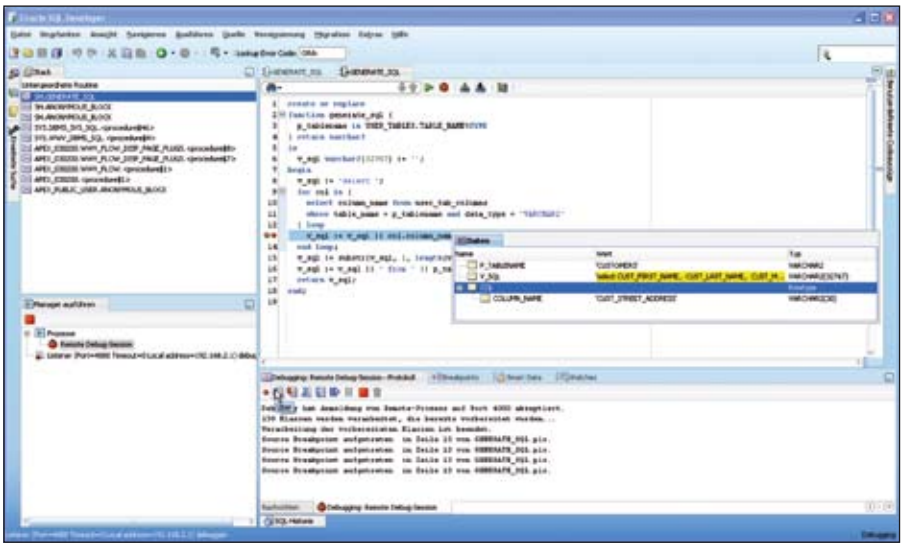


*Abbildung 5: Der Remote-Debugger ist aktiv*

Die Prozesse sollten an die Bedingung geknüpft werden, dass die beiden Elemente „NOT NULL“ sind. Setzt man nun die Elemente SQLDEV_HOST und SQLDEV_PORT per APEX-URL-Syntax, so wird der Debugger aktiv.

```
/pls/apex/f?p=&APP_ID.:&APP_
PAGE_ID.:SESSION.::::SQLDEV_
HOST,SQLDEV_
PORT:192.168.2.1,4000
```

*Listing 7: Remote-Debugging per APEX-URL aktivieren*

Integriert man dies noch in die Oberfläche, so wäre ein Endanwender (der sich mit einem Programmierer über ein Problem unterhält) in der Lage, für diesen eine Debugging-Sitzung zu starten. Der Vorteil ist, dass das Debugging im Kontext der Sitzung des Endanwenders stattfindet; die aufwendige Simulation der APEX-Umgebung ist nicht mehr nötig.

Übrigens: Das Remote-Debugging ist nicht auf Application Express beschränkt, sondern auch aus anderen Entwicklungsumgebungen heraus (Java, .NET, PHP etc.) möglich.

### Weitere Informationen

- APEX Community Tipp zum Thema: http://apex.oracle.com/url/apxdebug
- Deutschsprachige APEX Community: http://www.oracle.com/global/de/community/index.html
- Blog des Autors zu Oracle SQL und PL/SQL: http://sql-plsql-de.blogspot.com

**Kontakt:**
Carsten Czarski
carsten.czarski@oracle.com

# Forms goes Java Enterprise – aber was ist mit der PL/SQL-Fachlogik in der Oracle-Datenbank?

Oliver Zandner, Triestram & Partner GmbH

**Jeder Entwickler kennt Oracle Forms und schätzt dessen hohe Produktivität und Stabilität. Jeder Anwender kennt Web-2.0-Anwendungen wie Google Maps, die durch ihre hohe Interaktivität glänzen. Genau das erwarten Benutzer auch von ihren bestehenden Forms-Firmenanwendungen. Nur: Wie wird aus einer solchen bestehenden Anwendung eine Web-2.0-Anwendung? Und wie schützt man dabei die Investition, die in die Entwicklung der vorhandenen PL/SQL-Datenbank-Logik geflossen ist?**

Das Unternehmen des Autors entwickelt seit fast zwanzig Jahren Anwendungen auf der Basis der Oracle Datenbank sowie Oracle Forms und Reports. Dazu zählt das Standard-Produkt LISA. lims [1], ein Labor-Informations- und Managementsystem. Das, was Benutzer von solchen Informations-Systemen erwarten, ist zunehmend geprägt durch Web-2.0-Anwendungen. Insbesondere stellen Benutzer folgende Erwartungen an ihre Anwendungen:

- Abrufbarkeit im Internet via Browser, ohne dass dazu Plug-Ins installiert werden müssen
- Unterstützung von anwendungsspezifischen Funktionalitäten wie zum Beispiel die Verwendung von Tasten-Kürzeln bei einer Desktop-Anwendung wie Forms
- Bereitstellung einer ansprechenden Oberfläche, die individuell anpassbar ist [2]

Um solche Anforderungen in eine bestehende Oracle-Anwendung wie LISA. lims zu integrieren, sind folgende Varianten denkbar:

1. *Modernisierung der bestehenden Anwendung*
   Diese kann dadurch erfolgen, dass grafische Java-Komponenten wie etwa eine interaktive Straßenkarte integriert werden. Diese Variante belässt es bei der grundsätzlichen Forms-Web-Architektur – nämlich einem Java-Applet, das auf dem Client eine Laufzeitumgebung und entsprechende Bibliotheken erfordert. Nach Erfahrung des Autors wird dies in der Regel von der IT-Administration der Kunden abgelehnt. Darüber hinaus lässt sich durch eine Modernisierung nur ein Teil der oben genannten Anforderungen an die Benutzeroberfläche abdecken.

2. *Migration der bestehenden Anwendung auf eine neue Technologie-Plattform*
   Der Autor hat sich für die Migration der vorhandenen Forms-Applikation entschieden. Wichtig dabei ist, die Anforderungen genau zu klären, um anschließend die in Frage kommenden Architektur-Szenarien evaluieren zu können. Diese Schritte werden in den folgenden Abschnitten beschrieben.

## Anforderungen

Die migrierte Anwendung sollte folgende Anforderungen erfüllen:

1. Die Fach-Logik in PL/SQL in der Datenbank wiederverwenden, um die Investition zu schützen, die bislang dort hineingeflossen ist
2. Eine individualisierbare Benutzeroberfläche im beschriebenen Sinn von Web 2.0 zur Verfügung stellen
3. Auf Java als Basis-Plattform basieren
4. Sowohl als Desktop-, als auch als Internet-Anwendung verfügbar sein
5. In ihrer Internet-Version mindestens mit dem Microsoft Internet Explorer und Mozilla Firefox kompatibel sein
6. In ihrer Internet-Version mit möglichst vielen Applikations-Servern kompatibel sein

Die hier beschriebenen Anforderungen legen für das modernisierte Produkt eine Java-Enterprise-Architektur (JEE) nahe – und zwar mit einem Browser als Benutzerschnittstelle, einem Applikations-Server und der Oracle Datenbank mit dem bislang entwickelten Datenmodell und der bereits entwickelten Fach- und Querschnittslogik (wie Benutzer- und Stammdatenverwaltung, Internationalisierung etc.).

## Technologie-Evaluation

Es wurden folgende Architektur-Szenarien evaluiert, um zu prüfen, ob diese geeignet sind, das Produkt im Sinne der oben genannten Anforderungen zu modernisieren:

- JEE-Architektur auf Basis des Oracle ADF und des Oracle Application Servers
- JEE-Architektur auf Basis des JBoss Application Servers
- Eclipse RCP/RAP in Verbindung mit dem Spring Framework

Dabei haben sich folgende Aspekte als kritisch herauskristallisiert:

1. *Connection Pooling*
   In der JEE-Welt basiert die Verbindung zur Datenbank auf dem Prinzip des „Connection pooling“: Der Applikations-Server startet eine definierte Anzahl von Verbindungen mit der Datenbank. Führt der Applikations-Benutzer in der Anwendung eine Datenbank-bezogene Aktion aus, so nutzt der Applikations-Server dazu eine der Verbindungen aus dem Pool und gibt sie am Ende der Aktion wieder in diesen zurück, wo sie dem nächsten Applikations-Benutzer zu Verfügung steht. Dieses Konzept hat zwei Folgen: Erstens existiert für alle Applikations-Benutzer ein und derselbe Benutzer in der Datenbank. Zweitens ist die Verbindung zwischen Applikations-Server und Datenbank zustandslos – es gibt keinen durchgängigen Transaktions-Kontext wie in Oracle Forms. Die in der Anwendung vorhandene PL/SQL-Logik in der Datenbank benötigt jedoch genau einen solchen durchgängigen Transaktions-Kontext. Damit war das Konzept des Connection pooling nicht machbar.

2. *Optimistic Locking*
   Gemäß der JEE-Spezifikation werden bei Daten-Manipulationen die anzupassenden Datensätze zunächst aus der Datenbank in die Mittelschicht in sogenannte „Entity Beans“ geladen und dann durch die Applikations-Logik verändert. Erst zu einem späteren Zeitpunkt werden diese Änderungen in der Datenbank persistent gemacht. Dieses Konzept hat in diesem Fall folgende negative Auswirkungen:

   a. In der Anwendung ist die Fach-Logik in PL/SQL-Packages in der Datenbank implementiert. Diese PL/SQL-Logik dient unter anderem dazu, Daten in der Datenbank zu verändern. Da im JEE-Ansatz die Daten in der Mittelschicht verändert werden, ohne sie in der Datenbank zu sperren, kann es bei gleichzeitiger Verwendung von „DML-haltigem“ PL/SQL in der Datenbank zu Daten-Inkonsistenzen kommen.
   b. Das Gleiche gilt für alle anderen DML-Zugriffe auf die Datenbank, die von außerhalb des Application Servers erfolgen: Im Falle des Produkts existieren zahlreiche Subsysteme wie Labor-Messgeräte, die direkt an die Datenbank angebunden sind. Hier müsste der Schreib-Zugriff auf die Datenbank in Zukunft ausschließlich über den Applikations-Server erfolgen, um Daten-Inkonsistenzen zu vermeiden.

3. *Herstellerspezifische Erweiterungen des Applikations-Servers*
   Die Spezifikation der Java-Enterprise-Architektur (JEE) bezieht sich auch auf den Applikations-Server. Sun hat mit GlassFish eine Referenz-Implementation dieses Teils der Spezifikation vorgelegt. Andere Hersteller von Applikations-Servern weichen von diesem Standard ab, teilweise dadurch, dass sie spezifische Erweiterungen der Funktionalität bieten. Da die modernisierte Version der Applikation jedoch auf möglichst vielen Applikations-Servern lauffähig sein soll, ist die Nutzung solcher Hersteller-Spezifika strikt zu vermeiden.

### Schwierigkeiten auf der Ebene der Benutzeroberfläche

Hinsichtlich der Benutzeroberfläche haben sich folgende Hürden ergeben:

1. Bei den Oracle ADF Faces zeigte sich in Version 10 und 11, dass der mittels JDeveloper generierte Code nur sehr schwer zu warten beziehungsweise zu erweitern ist. Letzteres war immer dann nötig, wenn die vorgefertigten GUI-Elemente nicht den Erwartungen entsprachen, die durch Oracle Forms oder durch Web-2.0-Anwendungen geprägt sind.
2. Vorgefertigte GUI-Komponenten aus Frameworks lassen sich nicht in der Weise über die Tastatur bedienen, wie es Benutzer von Forms gewohnt sind und die Anwender des Produkts fordern.
3. Bei den verfügbaren GUI-Komponenten ist es nicht möglich, bei fehlgeschlagener Validierung den Cursor in dem entsprechenden Feld festzuhalten.

### Lösung: Eclipse RCP/RAP und Spring Framework

Folgender Ansatz wurde schließlich für die oben genannten Aspekte gewählt:

1. *Erstellung der Benutzeroberfläche mittels der Eclipse Rich Client Platform (Eclipse RCP) beziehungsweise der Eclipse Rich AJAX Platform (RAP)*
   Bei Eclipse handelt es sich um eine integrierte Entwicklungsumgebung. Diese umfasst unter anderem die Eclipse Rich Client Platform – eine Komponenten-Bibliothek zur Erstellung von Desktop-Anwendungen [3]. RCP basiert auf OSGi [4]. Die Wahl der Eclipse-Plattform bietet zwei Vorteile:

   a) OSGi ermöglicht es dem Entwickler, seine Software auf einfache Weise zu versionieren und ein Software-Produkt aus Modulen zu erstellen.
   b) Eclipse bietet eine zweigleisige Entwicklung. Der Entwickler schreibt zunächst eine Desktop-Applikation, deren GUI auf der Eclipse-SWT-Bibliothek [5] basiert. Aus dieser wird dann mit nur wenigen Modifikationen die Web-Anwendung generiert. Deren GUI basiert auf der Eclipse RAP-Bibliothek [6], die unter anderem aus einer umfangreichen AJAX-Bibliothek besteht, die der Web-Oberfläche ihre hohe Interaktivität und ihren hohen Bedienungskomfort verleiht.

2. *Verwendung des Spring Frameworks [7]*
   Das Spring Framework erlaubt es, den Benutzer-Kontext, den der Benutzer bei der Anmeldung an der Applikation erhält, an die Datenbank weiterzureichen und eine durchgängige Transaktions-Klammer zu schaffen. Auf diese Weise erfolgt eine individualisierte Anmeldung an der Datenbank. Damit lässt sich die in der Applikation vorhandene Datenbank-Logik integrieren.

3. *Verwendung des Objekt-relationalen Mappers EclipseLink [8]*
   EclipseLink ermöglicht es, bei der Manipulation von Datensätzen in der Datenbank ein pessimistisches Sperrverhalten zu implementieren – das heißt einen Datensatz für die Dauer der Bearbeitung exklusiv zu sperren und danach freizugeben – genau so, wie es in der Forms-Welt etabliert ist. Es ermöglicht, die bereits vorhandene PL/SQL-Datenbank-Logik in Transaktionen zu integrieren und die schreibende Anbindung von Subsystemen an die Datenbank unverändert zu lassen.

### Vor- und Nachteile sowie Alternativen

Welche Vor- und welche Nachteile birgt der oben beschriebene Architektur-Ansatz, mit dem wir eine „klassische" Business-Anwendung in Richtung Web 2.0 modernisiert haben?

Vorteile:

- Der Ansatz verleiht einer JEE-Anwendung das, was ihr an OLTP-Eigenschaften fehlt – nämlich das in der Forms-Welt etablierte pessimistische DML-Sperrverhalten. Hinzu kommen die Möglichkeit, „DML-haltige" PL/SQL-Datenbank-Logik in Transaktionen zu integrieren sowie die Option einer individualisierten Anmeldung an der Datenbank. Ergänzend kommt das hinzu, was eine JEE-Anwendung zu einer Web-2.0-Anwendung macht – nämlich die hohe Interaktivität der Benutzeroberfläche. Kurzum: das Beste aus zwei Welten.
- Die Integration von vorhandener PL/SQL-Datenbank-Logik bedeutet einen Schutz der bereits getätigten Investition und einen geringeren zeitlichen und finanziellen Aufwand bei der Modernisierung.
- Die gewählte Architektur ist unabhängig von Hersteller-Spezifika eines bestimmten Applikations-Servers. Dies erhöht die Flexibilität bei der Auslieferung.
- Der Single-Sourcing-Ansatz ermöglicht es, mit nur wenigen Modifikationen aus dem gleichen Quellcode zwei unterschiedliche Anwendungs-Typen zu generieren: entweder eine Java-Desktop- bzw. WebStart-Applikation oder eine Web-Applikation, die clientseitig ohne weitere Installationen bzw. Plug-ins auskommt.

Nachteile:

- Die hohe Interaktivität in der Benutzeroberfläche basiert auf der AJAX-Bibliothek des RAP-Frameworks von Eclipse – und damit auf JavaScript [9]. JavaScript ist eine standardisierte Sprache. Allerdings gibt es herstellerspezifische Unterschiede in der JavaScript-Engine, die das JavaScript im Browser ausführt. Dies kann unter Umständen zu Problemen mit der Kompatibilität führen.

Alternativen:

Hohe Interaktivität und hoher Bedienungskomfort der Benutzeroberfläche lassen sich beispielsweise auch mit folgenden Technologien realisieren:

- *Java Applets*
  Diese werden in der Java-Laufzeitumgebung des Browsers ausgeführt und erfordern damit eine entsprechende clientseitige Installation. Dies wird jedoch von einigen Kunden abgelehnt.

- *Adobe Flex [10]*
  Hiermit lassen sich Web-Anwendungen von beeindruckender Interaktivität erstellen (wie eine Mind-Mapping-Anwendung, die komplett im Browser läuft). Dazu ist ebenfalls eine Browser-Erweiterung notwendig, die allerdings im Adobe Reader ab Version 10 schon enthalten ist. Allerdings finden sich in der Praxis noch wenige industrielle Business-Anwendungen, die auf Flex basieren. Entsprechend gering ist das notwendige Know-how bei Entwicklern verbreitet. Gleiches gilt für die Rich-Client-Technologien Microsoft Silverlight [11] und JavaFX [12].

- *Microsoft ASP.Net [13]*
  Microsoft stellt mit ASP.Net ein Framework für die Entwicklung von Web-Anwendungen zu Verfügung. Diese setzen auf den Microsoft Information Server, der lizenzfrei mit dem Microsoft Server-Betriebs-System geliefert wird. Damit empfiehlt sich dieser Ansatz bei einer bereits vorhandenen Microsoft-Infrastruktur. Dieser Vorteil ist jedoch auch gleichzeitig ein Nachteil, da keine Plattform-Unabhängigkeit gegeben ist.

### Fazit

Auch in diesem Jahr waren bei der DOAG 2009 Konferenz die Vorträge zum Thema Forms-Modernisierung beziehungsweise Migration gut besucht. Es gibt zahlreiche Oracle Individual-Entwicklungen, bei denen eine solche Modernisierung beziehungsweise Migration ansteht. Der Autor freut sich darauf, seinen Ansatz mit anderen Interessierten zu diskutieren und ist gespannt zu sehen, welche Entwicklung das Thema in der Oracle Community nehmen wird.

### Weitere Informationen


[1] http://www.t-p.com/lisa/
[2] Zum Beispiel die Anpassbarkeit von Spaltenreihenfolgen und Sortierreihenfolgen; die Ein- und Ausblendbarkeit von Informationsbereichen; die Veränderbarkeit von Schriftgrößen; die Übernahme von angezeigten Informationen per Klick etwa in Microsoft Excel
[3] Gleichzeitig bildet RCP die Grundlage, mit der die Entwicklungsumgebung Eclipse geschrieben wurde, siehe http://en.wikipedia.org/wiki/Eclipse_(software)
[4] http://en.wikipedia.org/wiki/Equinox_OSGi
[5] http://en.wikipedia.org/wiki/Standard_Widget_Toolkit
[6] http://en.wikipedia.org/wiki/Rich_AJAX_Platform
[7] http://en.wikipedia.org/wiki/Spring_Framework
[8] http://en.wikipedia.org/wiki/EclipseLink
[9] http://en.wikipedia.org/wiki/JavaScript
[10] http://en.wikipedia.org/wiki/Adobe_Flex
[11] http://en.wikipedia.org/wiki/Silverlight
[12] http://en.wikipedia.org/wiki/Java_FX
[13] http://en.wikipedia.org/wiki/Asp_dot_net



**Kontakt:**
Oliver Zandner
o.zandner@t-p.com

# Migration von Forms über APEX zu Oracle ADF

Ulrich Gerkmann-Bartels und Andreas Koop, TEAM GmbH

**Viele Entwicklungsabteilungen beschäftigen sich schon länger mit der grundsätzlichen Frage, was aus bestehenden Oracle Forms-Applikationen in der Zukunft wird und ob es eine Möglichkeit gibt, die Implementierungen aus den bestehenden Applikationen zu migrieren oder zu modernisieren. Durch geschickte Kombination von verschiedenen Technologien und Produkten ist es möglich, auf einfache Art und Weise Oracle Forms-spezifische Deklarationen und Implementierungen zu extrahieren und im Anschluss auf Basis von Text-Templates Elemente einer Oracle ADF-Applikation generieren zu lassen.**

Um gleich ein grundsätzliches Missverständnis am Anfang dieses Artikels zu klären: Nach Ansicht der Autoren ist eine Technologie A mit den entsprechenden Möglichkeiten nicht auf Knopfdruck in eine Technologie B zu übersetzen, so dass die Möglichkeiten der Technologie B auch hinreichend aus genutzt werden (siehe Abbildung 1).

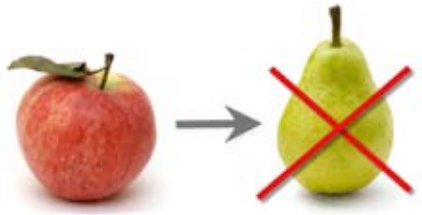


*Abbildung 1: Aus einem Apfel wird keine Birne*

Aus dieser Perspektive stellt sich jedoch die Frage, wie man die Implementierungen und Deklarationen, die man in vielen Jahren mit Oracle Forms aufgebaut hat, wiederverwenden beziehungsweise in eine neue Entwicklungsplattform überführen kann. Betrachtet man die einzelnen Forms-Dialoge als eine Beschreibungssprache für die Definition einer Benutzeroberfläche, Operationen innerhalb von Dialogen oder Daten in einem Use Case, so wäre es sicherlich von Vorteil, wenn man diese Beschreibungssprache im Dialekt „Oracle Forms“ in eine Meta-Ebene extrahieren könnte, um im Anschluss andere Dialekte wie zum Beispiel einen Oracle ADF-Java-Server-Faces-Dialog oder ein UML-Diagramm zu generieren. Dieser Artikel verfolgt dazu einen Model-Driven-Architecture-Ansatz (MDA), um nach einer Extraktion der wesentlichen Bestandteile aus Oracle Forms mit Text-Templates entsprechende andere Zielplattformen zu erreichen (siehe Abbildung 2).

### Extraktion von Oracle Forms in das APEX Metadata-Schema

Eine wesentliche Aufgabe innerhalb dieses Ansatzes ist die Extraktion der benötigten Informationen aus den Oracle Forms-Dialogen in eine Meta-Ebene, die leicht abgefragt und wiederverwendet werden kann. Diese Aufgabe hat uns Oracle mit Application Express in der Version 3.2 abgenommen (siehe DOAG *News* Q1/2009 – Modernisierung von Forms mit Application Express). Das Verfahren ist an dieser Stelle identisch mit der Migration von Oracle Forms nach Application Express. Im Einzelnen bedeutet dies:

- Konvertierung der Forms-Anwendungsdefinition in XML (Forms2XML Conversion Tool)
- Laden der XML-Dateien in das APEX Metadata-Schema

Die Vorgehensweise ist in verschiedenen Quellen durch Oracle oder im oben genannten Artikel beschrieben. Nach dem Laden der XML-Dateien liegen alle wesentlichen Informationen,

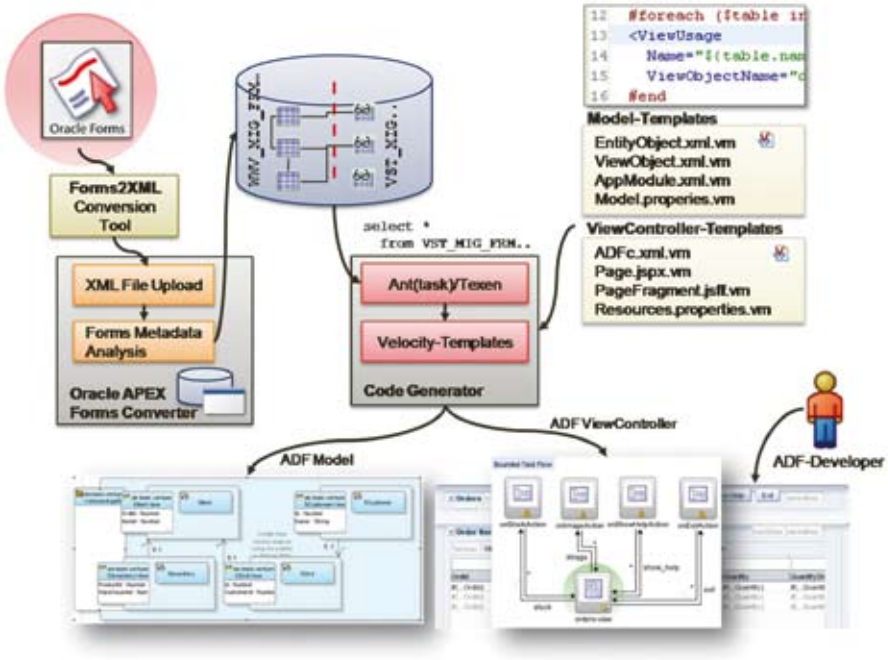



*Abbildung 2: Migration von Forms über APEX nach ADF*

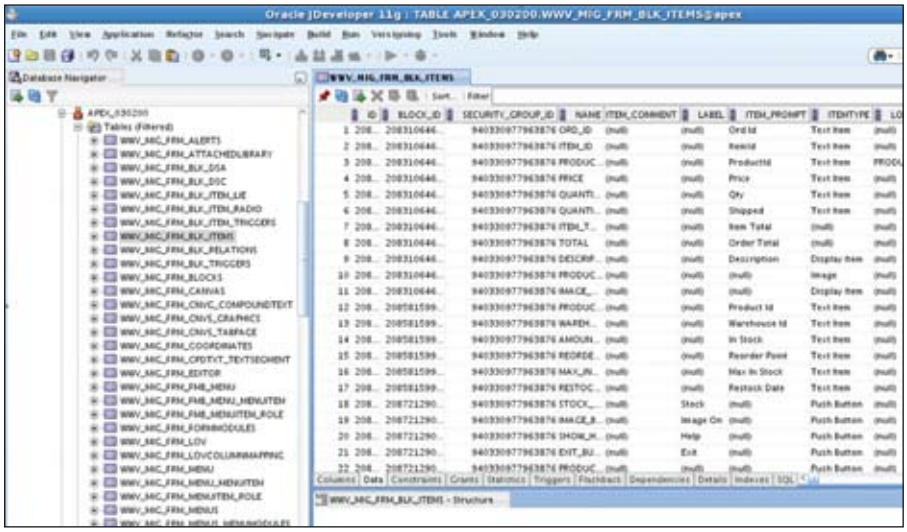


*Abbildung 3: APEX-Tabellen mit Meta-Information der Oracle Forms-Dialoge*

die in den Forms-Dialogen implementiert worden sind, in der Datenbank vor, so dass mithilfe der entsprechenden Zugriffsberechtigungen und passenden Abfragen diese Informationen abgerufen und innerhalb unseres MDA-Ansatzes verwendet werden können. Denkbar ist auch eine weitere Transformation auf eine höhere Stufe der Abstraktion (zum Beispiel ECore Meta Model). APEX wird im Weiteren nur noch als Metadata-Repository für die extrahierten Informationen aus Forms verwendet. Durch die Bereitstellung dieser Funktion innerhalb von APEX ist ein Großteil der Arbeit bereits umgesetzt.

### Verwendung des APEX Metadata-Schemas

Um die extrahierten Informationen innerhalb eines Templates verwenden zu können, ist es erforderlich, auf die entsprechenden internen Tabellen von APEX, die die Meta-Informationen beinhalten, zuzugreifen. Das entsprechende Datenmodell ist für jeden Datenbank-Entwickler nach wenigen Minuten erkennbar und verwendbar (siehe Abbildung 3).

Hier gilt es, entsprechende Abfragen zu definieren, die jene Informationen beinhalten, die im Anschluss innerhalb eines Templates verwendet werden sollen. Zum Beispiel eine Abfrage, die den Aufbau der jeweiligen Elemente eines Dialoges beinhaltet mit Elementen wie Text-Item, List-Item oder Tabs innerhalb eines Canvas.

| Name | Typ | Eigenschaften |
|---|---|---|
| $tables | Collection<Table> | |
| $tables[0-n]<br>$table | Table | name (String)<br>content.columns (Collection<Column>)<br>content.values (Collection<Row>)<br>content.datatypes (Collection<DataType)<br>... |
| <self define> | Column | name (String)<br>nameCamelCase (String)<br>datatype (DataType)<br>... |
| <self define> | Row | dynamisch: e.g. row.col1, row.col2 |
| | DataType | name (String)<br>precision, scale, charLength (Integer)<br>nullable (Boolean)<br>javaType (String)<br>javaTypeFull (String)<br>sqlType (String)<br>... |
| $generator | Generator<br>(apache-texen) | parse (...)<br>setOutputPatch (...)<br>etc |
| $strings | StringUtils<br>(apache-commons) | Hilfsfunktionen zur Manipulation von Zeichenketten |
| $files | FileUtil | Hilfsfunktionen auf Dateiobjekten |
| $properties | PropertiesUtil | Hilfsfunktionen auf Properties |

*Tabelle 1: Verfügbare Objekte im Template-Context*

### Aufbau eines Velocity Text-Templates

Die Erstellung der Templates für einzelne Komponenten der ADF-Applikation erfolgt auf Basis der Velocity Template Language. Grundlage für den im Template zu verwendenden Kontext stellen definierte Abfragen auf den Forms-Metadaten dar, die der Einfachheit halber als Views abgelegt sein sollten.

Diese Abfragen auf Views oder Tabellen werden von einem entwickelten Ant-Task (Vertumnus Sample) zu einem entsprechenden Velocity-Context aufbereitet und bieten dem Template-Entwickler eine einfache Möglichkeit, um im Template beispielsweise über die Metadaten von Tabellen, Views, Spalten oder auch über die Daten zu iterieren. Letzteres ist wichtig, weil damit Bezeichnungen von Items, Tabs und Fenstern extrahiert und im Template für Fenstertitel, Formularelemente oder Property-Dateien benutzt werden können. Tabelle 1 zeigt die Objekte, die im Kontext eines Templates zur Verfügung stehen.

Die Eigenschaften einzelner Typen lassen sich natürlich erweitern und an die Bedürfnisse der zu generierenden Artefakte einer Zielarchitektur anpassen. Mit diesem Konzept handelt es sich hier um eine sehr einfach gehaltene interne domänenspezifische Sprache (DSL) zur Transformation von Datenbankdaten und Metadaten auf Basis von Velocity.

Im nächsten Schritt wird diese (erweiterbare) Template-Sprache genutzt, um möglichst viele Bestandteile der ADF-Applikation zu generieren. Listing

1 zeigt exemplarisch den Ausschnitt eines Templates für die Generierung einer EntityObject-Definition.

### Generierung von ADF-Komponenten

Für die Generierung der gewünschten ADF-Komponenten kommen nur einige weit verbreitete Bibliotheken und Werkzeuge zum Einsatz:

- Apache Ant (Buildskript)
- Apache Velocity / Texen (Template-/Generator-Engine)
- Custom Database Ant-Task (Bereitstellung der notwendigen Objekte im Template-Context)

Zur Steuerung des Generierungsprozesses kommt ein Ant-Build-Skript zur Anwendung (siehe Listing 2). Alle Parameter sind für einen versierten Entwickler selbsterklärend.

Eine separate Ant-Task-Implementierung stellt den benötigten Velocity-Context bereit, welcher der Konfiguration entsprechend die Abfrage auf der Datenbank durchführt und anschließend das Control-Template verarbeitet.

Nachdem die Grundlagen geklärt sind, stellt sich die Frage, wie man ein Worker-Template am schnellsten anlegt. Hier hat sich der Ansatz bewährt, sich zunächst für das zu generierende ADF-Artefakt mithilfe der zahlreichen JDeveloper-Wizards ein Muster zu erzeugen. Anschließend zieht man dieses Muster als Basis für die Entwicklung des Templates heran.

In Tabelle 2 sind beispielhaft einige Transformationen von Oracle Forms-Elementen in äquivalente ADF-Artefakte dargestellt. Die Liste hat keinen Anspruch auf Vollständigkeit, sondern möchte den in diesem Artikel vorgestellten Ansatz mit konkreten Beispielen aus der Praxis untermauern. Die Transformationen können sich aufgrund von neuen ADF-Versionen oder -Entwurfsmustern auch ändern. Durch den flexiblen Generierungsprozess lassen sich solche Änderungen jedoch ohne nennenswerten Aufwand agil umsetzen.

Im Detail gibt es kaum Einschränkungen, da man immer die Möglichkeit hat, den Template-Kontext auf einfache Weise zu erweitern. Es sind nur wenige einfache Schritte erforderlich:

1. Auswahl eines zu migrierenden Bereichs der Forms-Anwendung
2. Abfrage (View) auf die Forms-Analyse daten erstellen/anpassen

**templates/adf/model/EntityObject.xml.vm**

```
...
#foreach ($column in $table.content.columns)
      #set($datatype = ${column.datatype})
      #if($datatype.name.equals("NUMBER"))
  <Attribute
    Name="${column.nameCamelCase}"
    IsNotNull="${datatype.nullable}"
    Precision="${datatype.precision}"
    Scale="${datatype.scale}"
    ColumnName="${column.name.toUpper-
Case()}"
    SQLType="${datatype.sqlType}"
    Type="${datatype.sqlType}"
    ColumnType="${datatype.name}"
    TableName="${table.name}">
    <DesignTime>
      <Attr Name="_DisplaySize"
Value="${datatype.charLength}"/>
    </DesignTime>
  </Attribute>
      #elseif($datatype.name.
equals("VARCHAR2"))
  ...
      #end
  ...
#end
</Entity>
```

*Listing 1: Template zur Generierung einer EntityObject-Definition*

**build.xml**

```
<target name="init">
  <taskdef name="generate"
           classname="...taskdef.database.DatabaseSchema-
TableExportTask"
           classpathref="team.anttask.dependencies"/>
  <tstamp/>
</target>
<!-- Generate ADF ViewController-->
<target name="gen-adf-model-objects" depends="init">
  <generate sourceconnectstring="${source.db.connectstring}"
            sourceuser="${source.db.username}"
            sourcepassword="${source.db.password}"
            controltemplate="Control.vm"
            tablenamepattern="VTS_VI_MIG_*"
            outputdirectory="${gen.model.dir}"
            templatepath="./src/templates/adf/viewcontroller"
  />
 <!—Weitere Generierungsprozesse -->
...
</target>
```

*Listing 2: Ant-Build-Skript als Generator-Workflow*

**templates/adf/model/Control.vm**

```
#set($app = "VertumnusSample")

#foreach ($table in $tables)
  #set ($outputFile = $table.nameCamelCase)
  $generator.parse("EntityObject.xml.vm", $outputFile, "table",
$table)
  ...
#end

#* iterate over all tables matching the configured pattern *#
$generator.parse("ApplicationModule.xml.vm", "${app}AppModule.
xml", "tables", $tables)
```

*Listing 3: Delegation der Generierung an Worker-Templates*

3. Generierungsworkflow (build.xml) erweitern
4. Template-Muster mit dem entsprechenden Wizard im JDeveloper generieren und als Grundlage für das Velocity-Template verwenden
5. Das jeweilige Velocity-Template anpassen
6. Generierungsergebnis verifizieren
7. Falls noch Teile der Forms-Anwendung bestehen, die per Transformationsregeln migriert werden können, wieder mit Punkt 1 fortfahren.

### Nebenprodukt: Generierung einer PL/SQL-Table-API

Für alle, die zur Zeit noch keine Migration ihrer Forms-Applikationen planen, ist dieser Ansatz trotzdem interessant, da nach dem gleichen Muster auch auf Basis des Data Dictionary der Datenbank entsprechende PL/SQL-Table-API-Packages generiert werden können. Entsprechende Templates lassen sich mit geringem Einarbeitungsaufwand implementieren. Der daraus entstandene Nutzen ist enorm hoch. Die Verwendung einer solchen Table-API als Zugriffsschicht ist auch bei der Entwicklung von APEX-Applikationen zu empfehlen. Ein entsprechendes

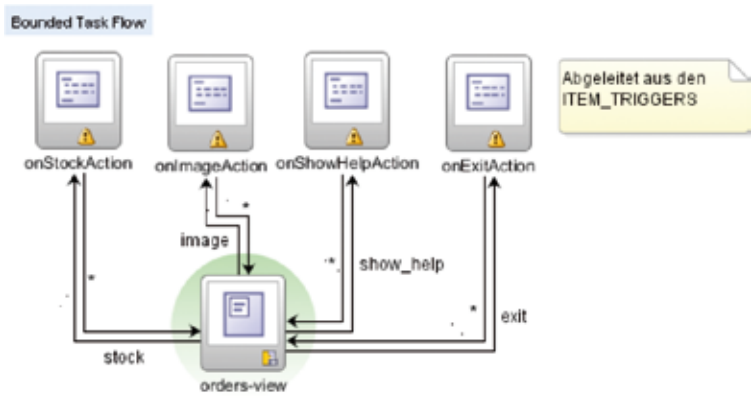
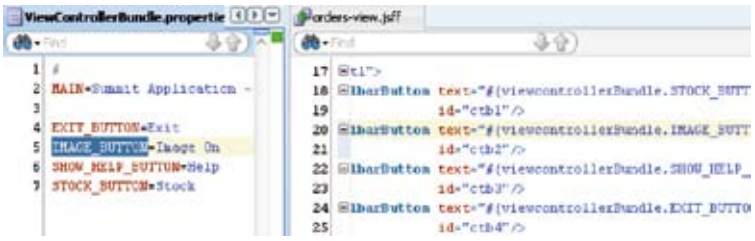
| Forms Element | Template | ADF-Artefakte |
|---|---|---|
| Item<br>Block | `TaskflowDef.xml.vm`<br>`..#foreach ($row in $table.content.values)`<br>`<method-call id="on${row.name.substring(5)}Action"></method-call>`<br>`#end..` | Method-Actions in Bounded Taskflows<br>  |
| Item<br>Window<br>.. | `ViewControllerBundle.properties.vm` | Property-Dateien<br>  |
| Window | `Page.jspx` | Java Server Page |
| Record Group<br>Lov<br>Block | `EntityObject.xml.vm`<br>`ViewObject.xml.vm`<br>`PageDef.xml.vm` | ADF Model EntityObject/ViewObject<br>Page Definitions |
| Alert | `PopupWindow.jsff.vm`<br>`PopupWindow.tf.xml.vm` | Inline PopupWindow als Taskflow |
| .. | .. | .. |

*Tabelle 2: Transformationsbeispiele von Forms-Elementen nach ADF*

Template findet sich im Vertumnus Sample.

## Fazit

Betrachtet man die verwendeten Technologien und Komponenten, so ist hier sicherlich nicht von einem Migrationstool die Rede. Aber nach Aufbau der Umgebung braucht es nur folgende Voraussetzungen, um mit den Informationen im APEX Metadata-Schema verwertbare Ergebnisse zu erzielen:

- Wissen in Oracle SQL, PL/SQL und Forms, um entsprechende Abfragen zu gestalten, die in einem Template verwendet werden können
- Erstellung von Velocity Text-Templates
- Oracle ADF XML-Dateien

Der letzte Punkt ist sicherlich der schwierigste. Diese Dateien sind aber in der Regel über Assistenten im Oracle JDeveloper als Grundlage für ein Text-Template erstellbar, zudem bietet diese Auseinandersetzung zwischen Oracle Forms und Oracle ADF eine gute Basis, um sich mit der strategischen Entwicklungsplattform Oracle ADF auseinanderzusetzen.

### Weitere Informationen

Zu den einzelnen Technologien des Ansatzes gibt es zahlreiche Informationen im Internet. Folgende Quellen sind aus unserer Sicht bedeutsam:

- Oracle Application Express for Oracle Forms Developers: http://www.oracle.com/technology/products/database/application_express/html/apex_for_forms.html
- Apache Velocity / Texen: http://velocity.apache.org/
- Apache Ant: http://ant.apache.org/
- Oracle JDeveloper / ADF: http://www.oracle.com/technology/products/jdev/index.html
- Vertumnus Sample: http://www.box.net/shared/x7iem1dfkj
- PADora – Technical Oracle Blog: http://padora.blogspot.com/2009/11/forms-migration-uber-apex-zu-adf-teil-1.html

**Kontakte:**
Ulrich Gerkmann-Bartels
ugb@team-pb.de
Andreas Koop
ak@team-pb.de

## *Wir begrüßen unsere neuen Mitglieder!*

### Firmenmitglieder

| | |
|---|---|
| Stefan Schreyer | BTC IT Services GmbH |
| Klaus Wentorp | Gebäudemanagement Schleswig-Holstein AöR (GMSH) |
| Robert Szilinski | esentri consulting GmbH |
| Wolfgang Liebich | Oberfinanzdirektion Karlsruhe |
| Jackson Basil | SKE Facility Management GmbH |
| Christian Bernhart | IMS GmbH |
| Frank Porebski | bwin Interactive Entertainment AG |
| Dirk Borchardt | gkv informatik |
| Andre Pittel | Anybet GmbH |
| Burkhard Thomas | Thales Information Systems Security |
| Barbara Erkens | AOK Systems GmbH |

### Persönlicher Mitglieder

Andreas Ströbel
Knud Loges
Heinke Voß
Matthias Popp
Stefan Seck
Marco Ramien
Andreas Drochner
Karl Glöckner
Thomas Tretter
Ralf Wörter
Joerg Otto
Ralf Kölling
Kai Brandes
Jan-Peter Timmermann
Ralph Waldenmaier
Erwin Hösch

## Oracle Newsticker

**Umsatz, Software-Erlöse aus dem Vertrieb von Neulizenzen, betriebliche Gewinnspanne und betrieblicher Cashflow steigen**

Die Oracle Corporation steigerte im zweiten Finanzquartal 2010, das zum 30. November 2009 endete, ihren GAAP-Umsatz gegenüber dem Vorjahresquartal um 4 Prozent auf 5,9 Milliarden US-Dollar. Dies entspricht einem Plus von 15 Prozent im Vergleich zum Vorjahreszeitraum. Zugleich stieg der Reingewinn nach GAAP um 12 Prozent auf 1,5 Milliarden US-Dollar.

Insgesamt stiegen die Software-Erlöse aus Neulizenzen nach GAAP um 2 Prozent auf 1,7 Milliarden US-Dollar. Die Umsätze aus Software-Lizenz-Updates und aus dem Produkt-Support stiegen um 14 Prozent auf 3,2 Milliarden US-Dollar. Die betrieblichen GAAP-Einnahmen stiegen um 10 Prozent auf 2,2 Milliarden US-Dollar.

„Wir erwirtschafteten Ergebnisse, die deutlich besser waren, als wir erwartet hatten. Unsere Non-GAAP-Marge aus dem operativen Geschäft wuchs um 280 Basispunkte auf 49 Prozent. Das ist die höchste Non-GAAP-Gewinnspanne aus dem operativen Geschäft, die wir jemals in einem zweiten Quartal verzeichneten“, sagte Jeff Epstein, Chief Financial Officer von Oracle. „Unser solides Wachstum, verbunden mit einer disziplinierten Ausgabenverwaltung, war der Schlüssel, um 8,4 Milliarden US-Dollar freien Cash-flow innerhalb der vergangenen zwölf Monate zu generieren.“

„Sowohl auf dem neuen SPARC Solaris System als auch auf der neuen Exadata Database Machine von Sun läuft Oracle Datenbank schneller als auf dem schnellsten Computer von IBM“, sagte Larry Ellison, Chief Executive Officer von Oracle. „Wir erwarten, dass Sun seine Marktanteile und Gewinnspanne rapide verbessert, sobald die Übernahme abgeschlossen ist.“

# Interaktive Reports mit automatischen Kalkulationen

Gerhild Aselmeyer, Konzepte & Anwendungsentwicklung für EDV

**Dieser Artikel beschreibt Möglichkeiten und Hindernisse für Apex-Entwickler, einem „Interactive Report" zur Laufzeit automatisch Kalkulationen/Berechnungen, Aggregationen oder andere Gestaltungselemente hinzuzufügen. Dafür muss man sich ausgiebiger mit dem Apex-Repository beschäftigen, da für notwendige Funktionen leider die entsprechende Dokumentation fehlt.**

Anforderungen dieser Art können entstehen, wenn SQL für die Anwender einer Apex-Applikation nicht zu den Qualifikationsvoraussetzungen ihres eigentlichen Aufgabengebiets zählt. In diesem Fall birgt vor allem „Compute" recht unüberwindliche Hindernisse beim Einsatz interaktiver Berichte – einer ansonsten komfortablen und gelungenen Erweiterung in Apex 3.

Nach der Installation von Apex 3.2 begeistern vor allem die interaktiven Reports. Hatte man, um ähnliche Funktionalität zur Verfügung zu stellen, zuvor noch recht aufwändig über den Datenbanksystem-Katalog Seiten, basierend auf Funktionen, die eine SQL-Query zurückgeben, aufbauen müssen, genügt jetzt als Basis ein alle Informationen umfassendes „SELECT"-Statement.

Für einen neuen Bericht sah die Autorin in einem interaktiven Report auf Grund der Anforderungen die beste Lösung. Basierend auf zwei Tabellen beziehungsweise Views für die Vereinigung von aktuellen mit historischen Daten mit „1:n"–Beziehung soll ein tabellarischer Bericht ausgegeben werden. Die Werte einer Spalte der Kind-Tabelle sind als Überschriften eigenständiger Spalten anzuzeigen, desgleichen in der jeweiligen Spalte eine Markierung, die angibt, ob die Beziehung für den aktuellen Satz existiert. Damit der Bericht von jedem Anwender auf die für ihn notwendigen Spalten eingeschränkt werden kann, bekommt er die Werte der Kind-Tabelle angezeigt und kann die ihn interessierenden als Berichtsspalten auswählen. Darüber hinaus sind auch die Werte zweier Spalten der Stammsatz-Tabelle in eigenständigen Berichtsspalten ebenso auszuwerten. Am Fuß der Tabelle erhalten alle abgeleiteten Spalten jeweils die Anzahl der in ihr markierten Sätze ausgewiesen. Zudem sollen natürlich ausgewählte Informationen des Stammsatzes als Originalspalte erscheinen. Außerdem muss der Anwender selbstverständlich die Möglichkeit haben, die Daten durch Query-Bedingungen für seine Bedürfnisse einzuschränken. Last but not least: Dem Anwender stehen ein CSV-Export und ein Drucklayout zum Erzeugen einer PDF-Datei zur Verfügung.

## Klassischer versus interaktiver Bericht

Um einen klassischen Apex-Report für diese Anforderungen zu erstellen, müssen in einer Funktion, die eine SQL-Query zurückliefert, die Vorgaben aus dafür vorgesehenen Items ausgelesen und die Query zusammengestellt werden. Auf Grund der abzuleitenden Spalten lässt sich eine recht komplexe Query erwarten, die als dynamisches Statement nicht einfach zu programmieren sein wird. Außerdem müssen die Spalten aus dem Stammsatz vorher explizit festgelegt sein, da sonst die Summation (Anzeige der Anzahl bei den abgeleiteten Spalten) nicht festgelegt werden kann – oder es müssen alle Spalten eine Summe erhalten, was nicht gewünscht ist. Die Vorgaben für die WHERE-Bedingungen auf bestimmte Spalten der Stammsatz-Tabelle einzuschränken, um den Vorgabebereich überschaubar zu halten – und das Programmieren der Query nicht noch weiter zu erschweren –, wäre kein Problem, da dieses aus anderen Berichten der Anwendung bereits bekannt ist. CSV-Export und Anzeige mit Drucklayout, um über die Druckfunktion des Browsers ein PDF zu erstellen, stehen für klassische Reports einfach zur Verfügung.

Bei einem interaktiven Report lassen sich die Anforderungen dagegen recht einfach umsetzen, indem man eine Query basierend auf Stammdaten- und Kind-Tabelle als Basisreport so zusammenstellt, dass alle Spalten der Stammdaten-Tabelle und die Werte der Überschriftenspalte aus der Kind-Tabelle als doppelpunktseparierte Liste in einer CLOB-Spalte zur Verfügung stehen. Die beiden Basis-Spalten für die Ableitungen und die CLOB-Spalte sind darin defaultmäßig ausgeblendet. Anschließend lassen sich die Markierungsspalten durch entsprechende Kalkulationen erzeugen und diese neuen Spalten mit Aggregationen versehen. Alle übrigen Anforderungen werden vom einem interaktiven Report durch entsprechende Einstellungen unter „Report Attributes" sogar mehr als erfüllt. Der Nachteil: Um die Kalkulationen für die abgeleiteten Spalten zu erstellen, benötigt der Anwender ein gewisses Maß an SQL-Kenntnissen.

Die Gegenüberstellung zeigt, dass für die gestellte Aufgabe ein „Interactive Report" die bessere Lösung bietet, wenn es gelingt, die abgeleiteten Spalten und deren Aggregationen automatisch – das heißt, programmtechnisch beim Aufruf des Berichtes – zu erstellen. Somit bestand die Hauptaufgabe darin, die Funktionen hinter den Dialogen zu finden und entsprechend den Anforderungen einzubinden.

### Die Funktionssuche

Normalerweise erwartet man diese Funktionen in der offiziellen, inzwischen recht umfangreichen API-Dokumentation, denn es klingt plausibel, dass Anforderungen auftreten, die eine automatisierte Durchführung von interaktiv zur Verfügung stehenden Aktionen erfordern. Leider sind die benötigten Funktionalitäten nicht einmal in der Liste der offiziell freigegebenen (mit PUBLIC SYNONYM und EXECUTE-Recht für PUBLIC ausgestatteten Packages, Prozeduren und Funktionen) Apex-Sourcen zu finden. Es bleibt nur die Suche im Apex-Repository, dem ehemaligen FLOWS- und jetzigen Apex-Schema in der Datenbank. Als Ansatz diente die in der Dokumentation zum Auffinden von Report-IDs erwähnte View „Apex_APPLICATION_PAGE_IR_RPT“. Über die Referenzierungen der Basis-Tabelle findet man folgende zehn Packages, die Funktionen für die interaktiven Berichte zur Verfügung stellen:

- WWV_FLOW_WORKSHEET
- globale Attribute, Initialisierung Anzeige von interaktiven Reports
- WWV_FLOW_WORKSHEET_AJAX
- WWV_FLOW_WORKSHEET_API
- Abfragen und Bearbeiten von Reports, Reportsichten und deren Spalten
- WWV_FLOW_WORKSHEET_ATTACHMENT Zeilenbezogenes Verknüpfen und Herunterladen von Dokumenten
- WWV_FLOW_WORKSHEET_DIALOGUE
- Aktionsdialoge
- WWV_FLOW_WORKSHEET_EXPR
- WWV_FLOW_WORKSHEET_FORM
- WWV_FLOW_WORKSHEET_STANDARD
- Generische APIs für interaktive Reports
- WWV_FLOW_WORKSHEET_STICKIES
- WWV_FLOW_WORKSHEET_UI

Obwohl der Kommentar (siehe Abbildung 1) im Kopf des Packages WWV_FLOW_WORKSHEET_API wenigstens auf etwas unterstützende Kurzdokumentation hoffen ließ, mussten korrekte Parameterübergabe und Verwendungsmöglichkeit mittels Versuch und Irrtum ermittelt werden. Ebenso fehlt jeder Hinweis darauf, dass im Package WWV_FLOW_WORKSHEET_STANDARD Funktionen zur Verfügung stehen, um Informationen zu Spalten und Überschriften zu erhalten.

```
6   --  Copyright (c) Oracle Corporation 2007.
    --  All Rights Reserved.
7   --
8   --    NAME
9   --      wwv_flow_worksheet_api.sql
10  --
11  --    DESCRIPTION
12  --      Public worksheet APIs.
13  --
14  --    RUNTIME DEPLOYMENT: YES
```

*Abbildung 1: Kommentar*

### Automatische Berechnungen, Gruppierungen und Hervorhebungen

Es gibt jedoch zumindest Überschriften für Funktionsgruppen und die Namen sind selbsterklärend, so dass man die für seine Anforderungen entscheidenden Prozeduren recht schnell innerhalb des Packages WWV_FLOW_WORKSHEET_API findet:

- ADD_OR_UPDATE_COMPUTATION
- Anlegen einer berechneten (abgeleiteten) Spalte
- BREAK_ON_COLUMN
- Setzen einen Umbruchs für eine Spalte
- SET_CONTROL_BREAKS
- Setzen eines Umbruchs für eine Spaltenliste, was allerdings bei Angabe der Spaltennamen in den beiden Arrays P_BREAK_ON und P_BREAK_ENABLED_LIST nicht funktionierte
- ADD_OR_UPDATE_HIGHLIGHT Hervorheben von Zeilen oder Zellen

Die Versorgung der Aufrufparameter birgt allerdings verschiedene Schwierigkeiten. Die jeweils ersten drei Parameter sind noch relativ einfach zu belegen:

- P_WORKSHEET_ID
  Entspricht der Spalte INTERACTIVE_REPORT_ID in der View Apex_APPLICATION_PAGE_IR_RPT und kann über die Funktion WWV_FLOW_WORKSHEET_API.GET_WORKSHEET_ID ausgelesen werden, wenn man in „Advanced Attributes“ den „Report Alias“ innerhalb der Applikation eindeutig belegt hat.
- P_APP_USER
  Der aktuell eingeloggte Benutzer der Anwendung und daher als „:APP_USER“ zu übergeben.
- P_REPORT_ID
  Entspricht in der View Apex_APPLICATION_PAGE_IR_RPT der Spalte REPORT_ID und ist mit der ID des entsprechenden Default-Reports für die Seite anzugeben; intern wird diese auf die ID für die Kopie der aktuellen Session umgesetzt, denn der Default-Report bleibt unverändert (siehe Abbildung 2). Setzt man hier jedoch die ID der Kopie für die aktuelle Session ein, zeigen die Funktionsaufrufe keine Wirkung oder verursachen sogar die Fehlermeldung „No Data found“ – offensichtlich erfolgt die Speicherung in der Datenbank erst später.

```
/*-----------------------------------------*/
/* Bestimmen der ID des Defaultberichtes */
/*-----------------------------------------*/
select REPORT_ID into nReportID
  from APEX_APPLICATION_PAGE_IR_RPT
 where PAGE_ID                = :APP_PAGE_ID
   and INTERACTIVE_REPORT_ID = nArbeitsblattID
   and REPORT_TYPE            = 'DEFAULT';
/*-------------------------------------------------*/
/* Betimmen der ID für die Berichtskopie der Session */
/*-------------------------------------------------*/
:P2_REPORT_ID := WWV_FLOW_WORKSHEET_STANDARD.get_report_id
                   (nArbeitsblattID, :APP_USER, nReportID);

/*-------------------------------------*/
/* Auslesen der aktuellen Spaltenliste */
/*-------------------------------------*/
tSpalten    := APEX_UTIL.STRING_TO_TABLE
               (WWV_FLOW_WORKSHEET_API.get_column_list(nArbeitsblattID,
                                                       :APP_USER, nReportID));
/*-------------------------------------------------*/
```

*Abbildung 2: Bestimmung und Nutzung der REPORT_ID*

Die Belegung der weiteren Parameter für die Prozedur ADD_OR_UPDATE_COMPUTATION lässt sich aus deren Namen ableiten. Allerdings sollte man nicht versuchen, mit einem eigenen Algorithmus Werte für P_COMPUTATION_ID zu bestimmen, um eventuell später ein Update darauf ausführen zu können, denn dann erscheint die Berechnung nicht. Immerhin lässt sich der Wert hinterher aus der Spalte COMPUTATION_ID der Tabelle WWV_FLOW_WORKSHEET_COMPUTATION lesen; hierbei muss man beachten, dass die REPORT_ID (für die Berichtskopie) immer aktuell mit der Funktion GET_REPORT_ID aus dem Package WWV_FLOW_WORKSHEET_STANDARD bestimmt wird, da sich diese auch während einer Session ändern kann. Außerdem müssen die beiden LABEL-Parameter (P_COLUMN_LABEL, P_REPORT_LABEL) identisch belegt sein.

Kalkulationen mit CASE gestalten sich – auch interaktiv – besonders schwierig, da als Operator bei WHEN ausschließlich „gleich“ (=) und unter THEN kein „Null“ akzeptiert wird (siehe Abbildung 3). Diese Einschränkung liegt offensichtlich im Code der Validierungsfunktion WWV_FLOW_WORKSHEET_EXPR.VALIDATE_COMP_EXPRESSION, denn sie tritt auch bei der interaktiven Nutzung auf, jedoch nicht bei einem entsprechenden SQL-Statement ( SELECT ..., CASE WHEN a<b THEN NULL ELSE ... END „Alias“ ... ), abgesetzt in „SQL Workshop > SQL Commands”.

Die Prozeduren zur Erzeugung von Umbrüchen und Hervorhebungen haben einen Spaltenbezug, der über den Eingangsparameter P_COLUMN zu definieren ist. Für abgeleitete Spalten ergibt sich hierbei die Schwierigkeit, die generierten Namen zu erhalten. Doch mit Hilfe der Funktionen GET_COLUMN_LIST aus WWV_FLOW_WORKSHEET_API zur Abfrage der Spaltenliste eines Reports und GET_COLUMN_LABEL aus WWV_FLOW_WORKSHEET_STANDARD zur Bestimmung der Überschrift zu einer vorgegebenen Spalte lassen diese sich aus dem Repository auslesen.

Dies gelingt auch für vorher im Prozess durch eine Berechnung erzeugte

```
/* Hochsetzen des Alias-Index                                          */
/* Für alle Kennungen aus der administrativen Tabelle AUSSTATTUNG_TECHNISCH */
/*    prüfen, ob die abgeleitete Markierungsspalte noch nicht existiert und */
/*            die Spalte angezeigt werden soll                        */
/*--------------------------------------------------------------------*/
nAlias := nAlias +1;
for curs3 in ( select KENNUNG from AUSSTATTUNG_TECHNISCH ) loop
   if ( instr(':'||upper(:P2_COLUMNS)||':', ':'||upper(curs3.KENNUNG)||':') = 0
        and instr(':'||:P2_TECHN_AUSSTATTUNG||':', ':'||curs3.KENNUNG||':') >0 ) then

/*--------------------------------------------------------------------------*/
/* Zusammensetzen der Kalkulationsregel:                                    */
/* CASE WHEN instr(U||':', ':'||'[Kennung]'||':')=0 THEN rtrim('-','-') ELSE 'x' END */
/* Erzeugen der abgeleiteten Spalte mit der jeweiligen Kennung als Überschrift */
/*--------------------------------------------------------------------------*/
      sComputedColStrg
      := 'CASE WHEN instr(' || chr(nAlias) || '||'':'', ''':''||''' ||
                    curs3.KENNUNG || '''||'':'')=0 THEN rtrim(''-'',''-'') ELSE ''x'' END';
      WWV_FLOW_WORKSHEET_API.add_or_update_computation
      (nArbeitsblattID, :APP_USER, nReportID, null,
       sComputedColStrg, null, curs3.KENNUNG, curs3.KENNUNG, :P2_FEHLERTEXT);
```

*Abbildung 3: Parameterbelegung beim Einfügen von Kalkulationen*

Spalten – allerdings erst nach Überwindung einiger Hindernisse, auf die später eingegangen wird.

Die Belegung der weiteren Aufrufparameter der Prozedur ADD_OR_UPDATE_HIGHLIGHT aus dem Package WWV_FLOW_WORKSHEET_API zur Hervorhebung von Zeilen oder Zellen lässt sich aus den Parameternamen ableiten. Unter Zuhilfenahme des interaktiven Dialogs kann man die Werte für einzelne Parameter auch recht einfach bestimmen. Von den Prozeduren für die weiteren Aktivitäten aus dem Aktionsmenü wurden über die bisher vorgestellten hinaus nur noch die für Aggregationen eingesetzt.

### Aggregationen schaffen besondere Probleme

Das Anlegen der Markierungsspalten abhängig von den Werten in den vorgegebenen Spalten erfolgt automatisch in einem Prozess „On Load – Before Region“. Dort soll abschließend auch die Aggregation zur Ausweisung der jeweils markierten Anzahl von Zeilen für die aus Kalkulationen gewonnenen Spalten angelegt werden. Hierfür gibt es die Prozedur WWV_FLOW_WORKSHEET_API. CREATE_AGGREGATE. Diese besitzt außer den drei Standardparametern noch drei weitere:

- P_OLD_AGGREGATION
  Wofür der Default-Wert „Null“ übernommen wurde
- P_AGGREGATE
  Wo offensichtlich die Art der Aggregation anzugeben ist
- P_COLUMN
  Für den Spaltenbezug – bekannt bereits aus den Funktionen für Umbrüche und Hervorhebungen

Allerdings ist festzustellen, dass ohne Dokumentation des bei P_AGGREGATE erwarteten Wertes (oder vielleicht einer abweichenden Belegung für P_COLUMN, was wenig wahrscheinlich ist) dieser Prozeduraufruf wirkungslos

```
/* Auslesen der aktuellen Spaltenliste                                 */
/*          der Spaltenüberschriften,                                  */
/* setzen von Gruppenumbrüchen und Hervorhebung für Spalten aus dem Defaultbericht*/
/*---------------------------------------------------------------------*/
tSpalten    := APEX_UTIL.STRING_TO_TABLE
               (WWV_FLOW_WORKSHEET_API.get_column_list(nArbeitsblattID,
                                                       :APP_USER, nReportID));
nSpaltenAnz := tSpalten.count;
for nInd in 1..nSpaltenAnz loop
   tUeberschrift(nInd) := WWV_FLOW_WORKSHEET_STANDARD.get_column_label(nArbeitsblattID,
                                                                       tSpalten(nInd));
   if (tUeberschrift(nInd) in ( 'Kategorie', 'Eigentümer', 'Standort') ) then
      WWV_FLOW_WORKSHEET_API.break_on_column
      (nArbeitsblattID, :APP_USER, nReportID, tSpalten(nInd));
   end if;

   if ( tUeberschrift(nInd) = 'Zulassung' ) then
      WWV_FLOW_WORKSHEET_API.add_or_update_highlight
      (nArbeitsblattID, :APP_USER, nReportID,
       null, 'Alte Fahrzeuge', tSpalten(nInd), '<', null, '01.01.2003',
       null, 10, 'Y', 'ROW', '#FFFF00', '#000033', null, :P2_FEHLERTEXT);
```

*Abbildung 4: Spaltennamen für Parameter P_COLUM bestimmen*

```
/*---------------------------------------*/
/* Ausgabe der Anzahl am Ende der Tabelle */
/*---------------------------------------*/
tSpalten    := APEX_UTIL.STRING_TO_TABLE
               (WWV_FLOW_WORKSHEET_API.get_column_list
                (nArbeitsblattID, :APP_USER, nReportID));
:P2_COLUMNS := NULL;
for nInd in nSpaltenAnz+1..tSpalten.count loop
   :P2_COLUMNS := :P2_COLUMNS||':'||tSpalten(nInd);
end loop;
UPDATE WWV_FLOW_WORKSHEET_RPTS
   set COUNT_COLUMNS_ON_BREAK = COUNT_COLUMNS_ON_BREAK||:P2_COLUMNS
 where id                = :P2_REPORT_ID
   and worksheet_id      = nArbeitsblattID
   and application_user  = :APP_USER
   and session_id        = :SESSION;

COMMIT;
```

*Abbildung 5: Eintragen einer COUNT-Aggregation*

bleibt. Alle Versuche mit Variationen des Schlüsselwortes „COUNT“ brachten sowohl für generierte Spalten als auch für Datenbankspalten des Default-Reports keinen Erfolg:

- Ausschließlich Groß- beziehungsweise Kleinbuchstaben
- Gewöhnliche Groß-/Kleinschreibung
- Alle vorher genannten Varianten mit hinzugefügtem Spaltennamen beziehungsweise Überschrift (wie in der Anzeige der „Report Attributes -> Default Report Settings“)
- Beide Namensangaben mit und ohne runde Klammern

Schließlich wurde interaktiv eine entsprechende Aggregation angelegt und deren Speicherung im Apex-Repository gesucht. Man findet den entsprechenden Eintrag recht schnell in der View Apex_APPLICATION_PAGE_IR_RPT; dort existiert für jede erlaubte Aggregation eine entsprechende Spalte [SQL-Aggregationsfunktionsname]_COLUMNS_ON_BREAK. Hierin sind alle zu aggregierenden Spalten als doppelpunktseparierte Liste eingetragen.

Daraufhin kam eine Radikalmethode zum Einsatz, um das gewünschte Ziel zu erreichen: ein direktes Update im Apex-Repository auf die entsprechende Basis-Tabelle der View. Hierbei muss man allerdings darauf achten, mit der REPORT_ID für die Kopie der Session zu arbeiten, da man sonst den Default-Report verändert.

Solche Lösungen sollten aber die Ausnahme bleiben und vor allem erst nach genauer Analyse des Repositorys vorgenommen werden; vor einer Übergabe in Produktion muss dabei natürlich ein besonders intensiver Test erfolgen. Dieses Vorgehen brachte endlich den gewünschten Erfolg und im interaktiven Dialog zeigen sich diese im Prozess erzeugten Aggregationen absolut fehlerfrei.

### AJAX und andere Hindernisse

Beim ersten Testen des Prozesses fällt zunächst etwas Merkwürdiges auf: Alle generierten Spalten fehlen in der Anzeige, obwohl sie im Seitenquelltext existieren. Erst durch einen Neuaufbau mindestens des Reports (über Button „Go“ oder das Ausführen einer Funktion aus dem Aktionsmenü – auch ohne Änderungen) erscheinen die Spalten. Ob dieses Verhalten auf AJAX zurückzuführen ist, kann die Autorin nicht beurteilen. Um aber beim Laden der Seite nach Abschluss aller notwendigen Prozesse einen nochmaligen Neuaufbau der Seite zu erzwingen, gibt es nur folgende Lösung (siehe Abbildung 6).

Ein direkter Aufruf der Javaskript-Funktion „pull“ aus der Klasse „ws“ des Namespace „apex.worksheet“ genügt nicht. Das Page-Item „P2_REFRESH“ wird initial mit 1 und beim Page-Processing bedingungslos mit 0 belegt; dessen Wertabfrage als Bedingung für den erzwungenen Neuaufbau unterbindet eine Endlos-Schleife. Zudem wird mit seiner Hilfe verhindert, dass der Generierungsprozess beim erzwungenen Refresh ein zweites Mal durchlaufen wird – aber dafür ein weiterer, um Aktionen für vorher automatisch generierte Spalten durchzuführen.

Dieses Vorgehen ist notwendig, da in einem Prozess generierte Spalten zwar direkt anschließend in der Spaltenliste von GET_COLUMN_LIST zur Verfügung stehen, nicht aber für die Prozeduren zur Durchführung von Aktionen (siehe Abbildung 7). Wahrscheinlich greifen diese nur mit „Update“ auf die Datenbank zu und die erstmalige Speicherung der

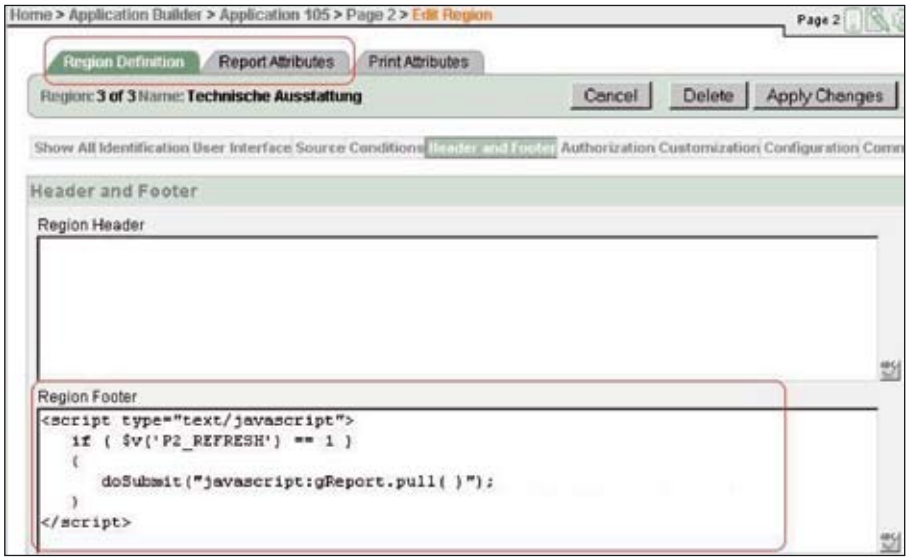



*Abbildung 6: Refresh erzwingen im Region Footer des interaktiven Berichts*

```
/*-------------------------------------*/
/* Auslesen der aktuellen Spaltenliste */
/*-------------------------------------*/
tSpalten    := APEX_UTIL.STRING_TO_TABLE
               (WWV_FLOW_WORKSHEET_API.get_column_list(:P2_ARBEITSBLATT,
                                                       :APP_USER, :P2_REPORT_ID));
/*-----------------------------------------------------*/
/* Auslesen der Spaltenüberschriften,                  */
/* setzen von Hervorhebung auf vorher berechnete Spalten*/
/*-----------------------------------------------------*/
for nInd in 1..tSpalten.count loop
   tUeberschrift(nInd) := WWV_FLOW_WORKSHEET_STANDARD.get_column_label(:P2_ARBEITSBLATT,
                                                                       tSpalten(nInd));
   if ( tUeberschrift(nInd) in ('EURO1', 'EURO2') ) then
      WWV_FLOW_WORKSHEET_API.add_or_update_highlight
      (:P2_ARBEITSBLATT, :APP_USER, :P2_REPORT_ID,
       null, 'Veraltete Abgasnorm', tSpalten(nInd), '=', null, 'x',
       null, 20, 'Y', 'ROW', '#FF0000', '#000000', null, :P2_FEHLERTEXT);
   end if;
```

*Abbildung 7: Hervorhebung in Abhängigkeit von generierten Spalten*

Session-Kopie des Reports erfolgt erst bei einem erneuten Aktionsaufruf. In diesem Prozess lässt sich daher für P_REPORT_ID die im Page-Item „P2_REPORT_ID“ gespeicherte Identifizierungsnummer der Session-Kopie des Reports verwenden; in späteren Prozessen sollte man die Identifizierungsnummer aber erneut bestimmen, da Anwenderaktionen diese verändern können.

Eindeutig dem AJAX-Konzept geschuldet ist der Umstand, dass man keine Aktion innerhalb des „Interactive Report“ mit bekannten Apex-Prozessen verknüpfen kann. Das bedeutet, dass bei einem Reset-Aufruf im Aktionsmenü keiner der „On Load“-Prozesse durchlaufen wird. Um dem Anwender bei einer solchen Aktion aber wieder seinen erweiterten Grundbericht zur Verfügung zu stel-

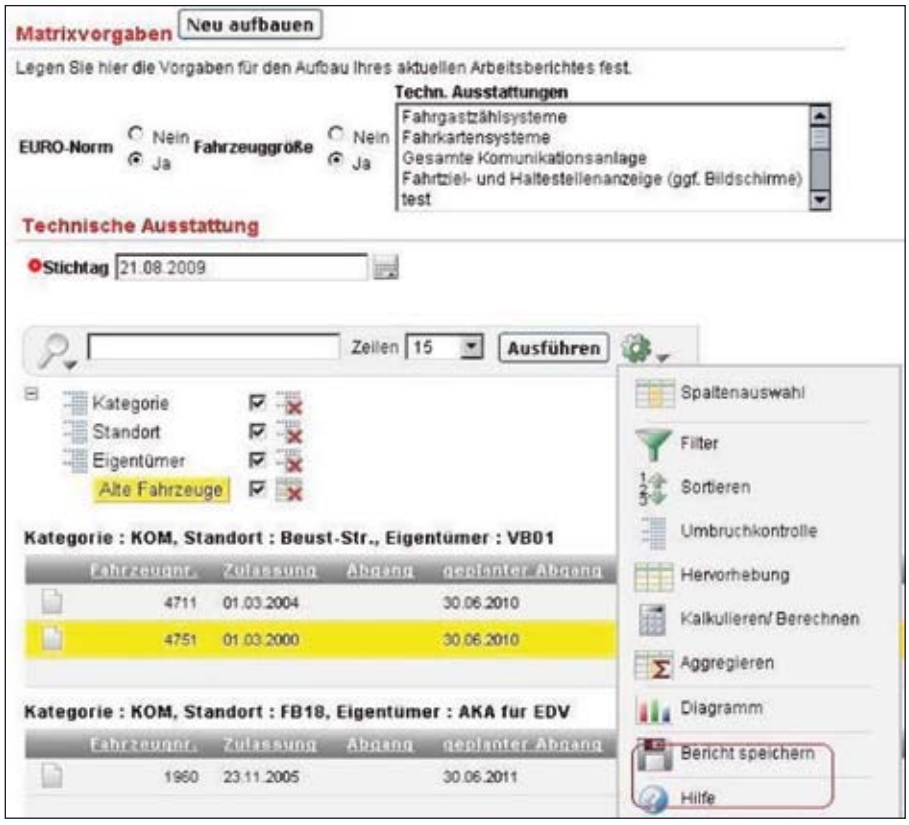



*Abbildung 8: Kein „Reset“ im Aktionsmenü, dafür der Button „Neu aufbauen“*


*Abbildung 9: Interaktiver Report mit automatisch ergänzten Kalkulationen, Aggregationen und Hervorhebungen beim Aufruf der Seite*

len, wurde diese Funktion aus dem Aktionsmenü entfernt und statt dessen ein „On Submit"-Prozess an einen Button „Neu aufbauen" gebunden, in dem das Zurücksetzen über den Aufruf von WWV_FLOW_WORKSHEET_API.RESET_REPORT_SETTINGS erfolgt und das Page Item „P2_REFRESH" zum Erzwingen des automatischen Refresh auf „1" zurückgesetzt wird (siehe Abbildung 8).

### Kleiner Makel

Nur für eine Anforderung wurde keine praktikable Lösung gefunden: dem Anwender beim Einstieg auf die Berichtsseite defaultmäßig nicht das aktuelle Arbeitsblatt, sondern beispielsweise seinen ersten gespeicherten Bericht anzuzeigen und gleichzeitig die Funktionalität aus dem Aktionsmenü zur Verfügung zu stellen, die Berichtsdaten der aktiven Reportsicht in Dateien zu speichern.

Zwar bietet Apex unter den Advanced Attributes eines interaktiven Reports die komfortable Möglichkeit, ein „Report ID Item" anzugeben, um direkt die Berichtssicht der darin angegebenen Identifizierungsnummer anzuzeigen, aber dann erhält der Anwender nur die Daten dieser Sicht in Dateien entladen – unabhängig davon, welche er gerade in der Anzeige hat. Da sich der Inhalt des angegebenen Page-Items aber aufgrund der AJAX-Implementierung nicht mit der aktuellen Sicht korrelieren lässt, gibt es wiederum nur die Möglichkeit, das Herunterladen aus dem Aktionsmenü zu entfernen und über einen zusätzlichen Button in einem selbst zu programmierenden Prozess anzubieten. Die Nachprogrammierung der Downloads mit Hilfe einer der Prozeduren Apex_UTIL.DOWNLOAD_PRINT_DOCUMENT ist allerdings etwas aufwändig.

### Weitere Versuche

Andererseits hat die Autorin versucht, ohne Angabe eines „Report ID Item" zu arbeiten. Dann erhält der Anwender beim Herunterladen über das Aktionsmenü die Daten der aktuell angezeigten Sicht in der Datei. Über den erzwungenen Refresh beim Laden der Seite wollte sie durch das Setzen der entsprechenden Identifizierungsnummer im „pull"-Aufruf eine andere Sicht als das aktuelle Worksheet anzeigen jedoch leider bisher ohne Erfolg. Da die Anzeige eines vorgegebenen gespeicherten Reports beim Aufruf der Seite aber nicht zu den Anforderungen gehörte, kann sich das Ergebnis eines interaktiven Berichtes mit automatischen Ergänzungen zur Laufzeit doch sehen lassen (siehe Abbildung 9).

### Fazit

Die „Interactive Reports" sind eine wirkliche Bereicherung für Apex. Der Entwickler kann den Anwendern eine mit wenigen Ansprechpartnern abgestimmte Grundversion eines Berichtes zur Verfügung stellen. Diese lässt sich aber von jedem einzelnen Anwender nach seinem Geschmack und seinen Erfordernissen interaktiv anpassen und in unterschiedlichen Sichten/Versionen speichern. Es entfällt damit auch die langwierige Diskussion mit der Gesamtgruppe der Anwender um die sinnvolle Spalten-Reihenfolge, die Notwendigkeit von angezeigten Spalten und Ähnliches. Auch der Einsatz des AJAX-Konzeptes bringt Vorteile – vor allem für die Performance zur Laufzeit. Allerdings bleiben (hier ist auch ein Apex-Entwickler Softwareanwender) noch eine ganze Reihe von Wünschen offen.

Warum die Aufrufparameter in den Funktionsdefinitionen nicht wenigstens im Quellcode kurz beschrieben sind und eine Liste möglicher Werte fehlt, ist nicht nachzuvollziehen, da dieses doch auch internen Entwicklern des Produktes Apex zu Gute käme. Daher stehen weitere Dokumentationen interner Funktionen auf der Wunschliste ganz oben. Außerdem sollte der Wert des „Report ID Item" entweder beim Aufruf einer anderen Berichtssicht entsprechend angepasst oder dem Entwickler eine einfache und dokumentierte Änderungsfunktion zur Verfügung gestellt werden. Es bleibt noch zu wünschen, dass bei „CASE unter WHEN" alle gültigen Operatoren akzeptiert werden sowie „Null" bei „THEN – ELSE". Im Sinn der Endanwender von Apex-Applikationen sollte auch das Layout für „XSL-FO"-Druck wie bei klassischen Reports änderbar sein. Das Standard-Layout passt leider nicht in allen Fällen.

**Kontakt:**
Gerhild Aselmeyer
g.aselmeyer@aka-edv.de

# PL/SQL-Performance-Tuning in 11g

Dr. Hildegard Asenbauer, MuniQSoft GmbH

**PL/SQL bietet eine Reihe von Features, die gezielt eingesetzt die Performance einer Applikation deutlich steigern können. Die wichtigsten davon werden in diesem Artikel vorgestellt.**

Ist von „Tuning“ die Rede, so denkt man in erster Linie an Datenbank-Tuning. Performance-Optimierung ist aber nicht nur Sache des DBA, hier ist auch der Programmierer gefragt. Es reicht in der Regel nicht aus, dass ein Programm einigermaßen fehlerfrei ist und irgendwie läuft – Performance-Engpässe durch ungünstige Programmierung kann auch der beste DBA nicht beseitigen. Je mehr Benutzer gleichzeitig mit einer PL/SQL-Applikation arbeiten, desto gravierender sind die Auswirkungen von wenig performantem Code.

Natürlich muss der fehlerfreie, den gestellten Anforderungen entsprechende Ablauf eines Programms im Vordergrund des Interesses eines Entwicklers stehen. Wenn das gewährleistet ist, sollte sich ein Entwickler aber auch zum Thema „Performance“ Gedanken machen. Manches kann man bereits durch die Beachtung einiger Grundregeln erreichen, die aus Datenbanksicht nicht auf Kosten anderer Ressourcen gehen, manchmal muss aber auch eine Abwägung getroffen werden zwischen Performance-Aspekten und Speicherverbrauch.

### Result Cache

Aus Entwicklersicht ist das herausragende neue Performance-Feature in Version 11*g* der Result Cache. Will man in früheren Versionen die wiederholte Ausführung einer Funktion vermeiden, so bleibt nur die Möglichkeit, das Ergebnis in einer Package-Variablen zu speichern. Das hat natürlich Nachteile: Der Wert ist immer nur innerhalb einer Session zugänglich, und falls die Funktion auf eine Tabelle zugreift, muss sichergestellt sein, dass sich deren Inhalt zwischenzeitlich nicht ändert; es gibt keine Möglichkeit, das Programm über eine Änderung zu informieren. Beide Nachteile werden mit dem Result Cache beseitigt.

Die Idee dahinter: Das Ergebnis einer Funktion samt zugehörigen Input-Werten wird im Hauptspeicher vorgehalten, so dass bei Aufruf mit gleichen Parameter-Werten die Funktion nicht erneut ausgeführt wird, sondern stattdessen direkt auf das Ergebnis zugegriffen werden kann. Werden in der Funktion Tabellenwerte ausgelesen, so wird das in Version 11.1 in der Signatur mit angegeben. Dadurch ist der Ergebnis-Cache automatisch invalidiert, wenn an der Tabelle Änderungen durchgeführt wurden. In Version 11.2 werden solche Abhängigkeiten schon automatisch erkannt. Der zweite große Vorteil liegt darin, dass der Result Cache Bestandteil der SGA ist; daher kann er Session-übergreifend genutzt werden, was ihn im Multi-User-Umfeld zu einem sehr nützlichen Feature macht. Gerade im Zusammenhang mit kleinen, statischen Lookup-Tabellen kann die Verwendung des Result Cache deutliche Performance-Verbesserungen bringen. Der einzige Wermutstropfen dabei ist, dass das Feature in der Oracle Standard Edition nicht zur Verfügung steht.

Der Result Cache ist denkbar einfach zu konfigurieren. Der Initialisierungsparameter RESULT_CACHE_MAX_SIZE muss auf einen Wert größer Null gesetzt sein (was Standard ist), dann ist er sowohl in SQL (über den Hint RESULT_CACHE oder die Einstellung result_cache_mode = FORCE) als auch in PL/SQL nutzbar. Daneben kann über RESULT_CACHE_MAX_RESULT gesteuert werden, wie viel Prozent des gesamten Caches maximal für ein Ergebnis verbraucht werden darf. Nachfolgend ein Beispiel zur beliebten EMP-Tabelle:

```
CREATE FUNCTION cache_it (
   p_empno  IN  NUMBER
)
   RETURN VARCHAR2
   RESULT_CACHE
   RELIES_ON(SCOTT.emp)  -- in
Version 11.2 nicht nötig
IS
   v_name   VARCHAR2 (100);
BEGIN
   SELECT ename
     INTO v_name
     FROM SCOTT.emp
    WHERE empno = p_empno;

   RETURN v_name;
END;
/
```

### Bulk Binds

Ein altbekanntes Feature zur Performance-Optimierung sind die Bulk Binds, die bereits mit Version 8*i* Einzug hielten, hier aber aufgrund ihrer großen Bedeutung zur Performance-Optimierung nicht fehlen dürfen. Seit ihrer Einführung wurden sie – genauer gesagt Bulk DML – mit jeder neuen Datenbank-Version komfortabler in der Handhabung. Bulk Binds unterteilen sich in Bulk Select und Bulk DML; beiden gemeinsam ist, dass mit Arrays gearbeitet wird und der Performance-Vorteil in der Vermeidung eines ständigen Context Switches begründet ist.

Bei Bulk Select kann prinzipiell in einem Schritt die gesamte Ergebnismenge in einem Schritt in ein Array eingelesen werden. Ist Speicherverbrauch ein Thema, so sollte in einer Schleife über die LIMIT-Klausel die Ergebnismenge häppchenweise verarbeitet werden. Eine Erhöhung von LIMIT auf einen Wert größer als 100 bringt in der Regel keine großen Verbesserungen mehr. Vergleicht man ab Version 10*g* die Geschwindigkeit einer Cursor FOR-Schleife mit Bulk Select, so wird

man jedoch bei Default-Einstellungen unter Umständen kaum Unterschiede feststellen. Der Grund dafür liegt darin, dass der in 10*g* eingeführte Initialisierungsparameter PLSQL_OPTIMIZE_LEVEL standardmäßig auf „2“ steht, was unter anderem eine implizite Umwandlung der Schleife bewirkt, sofern dies möglich ist. Bei Verwendung von explizitem OPEN / FETCH / CLOSE tritt dieser Effekt nicht auf.

Bei Bulk DML wird ein Array als Ganzes zur Verarbeitung des angegebenen DML-Befehls an die SQL-Engine übergeben. Seit Version 11*g* kann hier endlich auch ein einzelnes Feld in einem Record-Array angesprochen werden, so dass es keine Hinderungsgründe mehr gibt, dieses Feature zur Massenverarbeitung auch bei Update und Delete einzusetzen. Der größte Performance-Vorteil ist jedoch auch weiterhin bei Bulk Insert feststellbar; eine Beschleunigung um den Faktor 10 ist hier durchaus realistisch.

Auch mit Version 11.2 gab es eine Neuerung zu Bulk DML: Für Bulk Insert (und nur dafür) wurde der neue Hint „APPEND_VALUES“ eingeführt, der in seinen Auswirkungen dem APPEND-Hint eines Direct Load Inserts entspricht.

Da zu diesem Feature seit langem an genügend Stellen ausreichend Beispiele zugänglich sind (zum Beispiel: http://www.muniqsoft.de/tipps/tipps-zu-pls-ql/bulk-binds.htm), sei hier nur kurz an die allgemeine Syntax erinnert:

```
SELECT .. BULK COLLECT INTO ..
FETCH .. BULK COLLECT INTO ..
[LIMIT X]

FORALL i IN x..y [SAVE EXCEPTI-
ONS]
FORALL i IN INDICES OF v_arr
[BETWEEN x AND y] [SAVE EXCEP-
TIONS]
FORALL i IN VALUES OF v_arr
[SAVE EXCEPTIONS]
```

### Native Kompilierung

PL/SQL galt früher immer als langsam, gerade wenn es um rechenintensive Operationen ging. Aus diesem Grund hat Oracle bereits mit Version 9*i* die Möglichkeit der nativen Kompilierung eingeführt. Bis einschließlich Version 10.2 wird dafür jedoch ein externer C-Compiler benötigt, und die PL/SQL-Routinen werden als Shared Library Unit oder Dynamic Link Library (DLL) im Filesystem abgespeichert, zur Ausführung in die PGA geholt und direkt an den Oracle-Prozess gelinkt. Ein Interpreter-Schritt, wie er sonst bei der Ausführung von PL/SQL-Routinen nötig ist, entfällt damit.

In Version 11*g* wurde die native Kompilierung sehr vereinfacht und ist damit sehr komfortabel zu handhaben. Der Parameter PLSQL_OPTIMIZE_LEVEL muss dabei mindestens auf dem Wert „2“ (dem Default) stehen. Der größte Vorteil ist, dass kein C-Compiler mehr nötig ist; auch ein Zugriff auf das Filesystem ist nicht mehr erforderlich. Einziger verbleibender Parameter ist PLSQL_CODE_TYPE; dieser kann systemweit, auf Session-Ebene oder für eine bestimmte Prozedur auf NATIVE eingestellt werden; der Default ist INTERPRETED.

```
ALTER PROCEDURE my_proc COMPILE
PLSQL_CODE_TYPE = NATIVE;
```

Native Kompilierung ist jedoch kein Allheilmittel. Zugriffe auf die Datenbank können dadurch nicht beschleunigt werden, der größte Effekt ist bei komplexen oder ausführlichen Berechnungen zu erwarten. Zudem kann man nativ kompilierten Code nicht mehr debuggen.

In 11*g* wurde eigens der numerische Datentyp SIMPLE_INTEGER als Subtyp von PLS_INTEGER mit „NOT NULL“-Constraint eingeführt, dessen Verwendung nativ kompilierte Programme noch weiter beschleunigen kann. Sein Einsatz ist jedoch limitiert: Es muss sichergestellt sein, dass es nicht zu einem Überlauf kommt, da bei Überschreitung des Wertebereichs keine Exception ausgelöst wird, sondern stattdessen Wrapping stattfindet. Wer will schon als Ergebnis der Berechnung 2147483647 + 1 den Wert –2147483648 erhalten?

### Subprogramm Inlining

PL/SQL-Packages sollten aus Gründen der Wartbarkeit und der Wiederverwertbarkeit von Code möglichst modular gehalten werden. Wird dann eine Hilfsprozedur oder -funktion daraus in einer Schleife aufgerufen, kann sich ein kleiner Overhead durch den Aufruf durchaus bemerkbar machen. Mit Version 11*g* wurde zur Umgehung dieses Overheads das Feature des Subprogramms Inlining eingeführt. Der Entwickler kann weiter wie gewohnt modular programmieren, aber der Quellcode des Hilfsprogramms wird beim Kompilieren in das Hauptprogramm hineingezogen. Der Performance-Gewinn hängt jedoch stark von der Anzahl der Schleifendurchläufe ab. Bei dem unten angegebenen Beispiel mit 10.000.000 Durchläufen war er im Test deutlich messbar (ca. zwei Sekunden mit Inlining, knapp vier Sekunden ohne), während er sich bei 10.000 Durchläufen im Bereich von ein bis zwei Hundertstel Sekunden bewegte.

Wenn PLSQL_OPTIMIZE_LEVEL auf „2“ steht (Default), muss Inlining explizit über das neue Pragma INLINE angefordert werden. Bei einem Wert von „3“ dagegen, der erst ab Version 11.1 zulässig ist, entscheidet der Compiler selbst, wann Inlining Performance-Vorteile bringen könnte. In diesem Fall kann jedoch über das Pragma INLINE das automatische Einbinden explizit verhindert werden. Im Gegensatz zu anderen Pragmata wird das Pragma INLINE direkt vor dem Aufruf der Subroutine angegeben:

```
CREATE OR REPLACE PACKAGE BODY
inline_pack
AS
   FUNCTION my_function (
      p_1  IN  NUMBER
   )
      RETURN NUMBER
   AS
   BEGIN
      RETURN p_1 * p_1;
   END my_function;

   PROCEDURE doit
   IS
      v_return   NUMBER;
   BEGIN
      FOR i IN 1 .. 10000000
      LOOP
         -- Explizit Inlining
unterbinden:
         -- PRAGMA INLINE (my_
function, ‚NO‘);
```

```
         -- Explizit Inlining
anfordern:
         PRAGMA INLINE (my_
function, ‚YES‘);
         v_return := my_func-
tion (i);
      END LOOP;
   END doit;
END inline_pack;
```

### NOCOPY

Werden OUT- oder IN-OUT-Parameter an eine Prozedur übergeben, dann erfolgt die Übergabe standardmäßig BY VALUE. Für Arrays und LOBs – oder auch Objekte – bedeutet das, dass unter Umständen größere Datenmengen hin- und herkopiert werden müssen. Das kostet sowohl Speicherplatz als auch Zeit. Über den schon länger bekannten Compiler Hint NOCOPY in der Parameterliste wird eine Übergabe BY REFERENCE ermöglicht, sofern die Prozedur nicht über einen Datenbank-Link aufgerufen wird oder andere Gründe dagegen sprechen. BY REFERENCE bedeutet, dass statt des Inhalts der Variablen deren Speicheradresse übergeben wird.

Auch hier kommt die Einstellung von PLSQL_OPTIMIZE_LEVEL ins Spiel: Bei einem Wert größer als „1“ kann eine entsprechende Optimierung auch ohne Angabe von NOCOPY erfolgen. Im folgenden Beispiel dauerte die NOCOPY-Variante bei einer Collection mit ca. 150.000 Elementen sechs Hundertstel Sekunden gegenüber 28 ohne Angabe des Hints:

```
CREATE OR REPLACE PACKAGE BODY
no_copy
AS
   PROCEDURE p_nocopy (p_array
IN OUT NOCOPY no_copy.t_array)
   IS
   BEGIN
      FOR i IN 1..p_array.COUNT
      LOOP
         p_array(i).id := i;
      END LOOP;
   END p_nocopy;
   …
END no_copy;
```

Gleichzeitig konnte in v$sesstat bei einer Übergabe BY VALUE eine Verdopplung des Wertes für „session pga memory max“ beobachtet werden gegenüber einer Übergabe BY REFERENCE.

Da bei einer Übergabe BY REFERENCE direkt in die Variable geschrieben und nicht der Wert am Ende zurückgegeben wird, ist jedoch vor allem bei IN-OUT-Parametern Vorsicht geboten im Zusammenhang mit möglicherweise vorkommenden Fehlern. Tritt beispielsweise bei der Bearbeitung einer Collection-Variablen ein Laufzeitfehler auf, so kann diese im Exception-Teil nicht einfach auf den Stand zu Beginn der Prozedur zurückgesetzt werden.

### SQL in PL/SQL

Das Hauptproblem bei echten Performance-Engpässen liegt in der Regel in schlecht getunten SQL-Befehlen. In diesem Fall können die besten Tipps zu PL/SQL-Tuning nicht weiterhelfen. Da hilft nur klassisches SQL-Tuning. Daneben sollte man noch ein paar weitere Regeln beachten:

- Bind-Variablen benutzen, wo immer man kann! In 11*g* sind die Möglichkeiten dank Adaptive Cursor Sharing sogar noch weiter gefasst, da das Problem des Bind Peeking minimiert wurde.
- SQL-Befehle sollten in eigene Prozeduren beziehungsweise Funktionen ausgelagert werden, und kein Befehl sollte im Quellcode doppelt vorkommen. Das erhöht zum einen die Wartbarkeit, zum anderen wird zu häufiges Parsen aufgrund von unterschiedlicher Schreibweise verhindert.
- Keine PL/SQL-Workarounds suchen, wenn ein Problem auch mit SQL lösbar ist. Zugespitzt ausgedrückt: Ein Join ist besser als zwei ineinander geschachtelte Schleifen.

### Vermischtes

Eine Anwendung, die für Mehr-Benutzer-Betrieb vorgesehen ist, muss auch unter solchen Bedingungen getestet werden. Hier ist speziell darauf zu achten, dass sich einzelne Benutzer nicht gegenseitig ausbremsen, etwa durch unnötig lange gehaltene Sperren. Auch der Speicherbedarf darf für eine einzelne Session nicht zu hoch werden. Ein Programm dagegen, das nur einmalig gestartet wird, um beispielsweise nachts Bereinigungen oder Auswertungen durchzuführen, kann in der Regel großzügiger mit Ressourcen umgehen.

Oracle bricht bei bedingten Anweisungen die Überprüfung weiterer Zweige ab, sobald der erste Zweig gefunden ist, für den die Bedingung(en) erfüllt ist/sind. Auch bei AND- und OR-Verknüpfungen von Bedingungen wird abgebrochen, sobald das Gesamtergebnis feststeht (short-circuit evaluation). Daher ist es sinnvoll, aufwändig zu überprüfende Bedingungen ans Ende zu stellen.

### Fazit

Die Performance-Features, die PL/SQL in Version 11*g* bietet, sind sehr einfach zu implementieren. Werden sie von vorneherein so konsequent wie möglich (und sinnvoll) genutzt, erspart dies viel nachträglichen Tuning-Aufwand.

**Kontakt:**
Dr. Hildegard Asenbauer
h.asenbauer@muniqsoft.de

**Datenintegrations-Lösung Oracle GoldenGate** — **Oracle Newsticker**

Mit der Vorstellung von Oracle GoldenGate folgt Oracle seiner Strategie, eine umfassende Datenintegrations-Lösung anzubieten. Oracle GoldenGate ermöglicht schnelle Echtzeit-Datenintegration und die Transformation großer Datenmengen. Darüber hinaus stellt Oracle GoldenGate Funktionen zur Verfügung, mit denen sich zuverlässige Datenqualität erreichen lässt, Instant Business Intelligence verfügbar, höhere Online Transaction Processing Leistung erzielt und kontinuierliche Datenverfügbarkeit (24x7) für geschäftskritische Systeme möglich werden. Mit der vollständigen Integration der GoldenGate Software-Technologie in die Oracle Data-Integration-Suite-Lösungen verfügt Oracle GoldenGate nun über Echtzeit-Datenintegrations- und Replikations-Technologien für Oracle Fusion Middleware.

# Oracle und .NET – die neue ODAC-Version

Markus Kißling, ORACLE Deutschland GmbH

**Die Oracle Developer Tools for Visual Studio (ODT) und der Oracle Data Provider for .NET (ODP.NET) sind Bestandteil der Oracle Data Access Components (ODAC). In diesem Artikel werden die Neuigkeiten der Version 11.1.0.7.20 vorgestellt. Weitere allgemeine Tipps zu ODP.NET geben einen Einblick in das große Potenzial, das .Net-Anwendungen zusammen mit der Oracle Datenbank bereithalten. Da Microsoft seinen OracleClient-Provider zukünftig nicht mehr weiterentwickelt, erfolgt zum Schluss eine Zusammenstellung der wichtigsten Informationen für eine erfolgreiche Umstellung auf ODP.NET.**

Das ODAC-Paket enthält alles, was zur Anwendungsentwicklung mit Oracle unter Microsoft Windows notwendig ist. Der Oracle Instant Client ist nur ein paar MByte groß und dient als Basis für den Zugriff auf die Oracle Datenbank über Oracle Net Services und das Oracle Call Interface (OCI). Neben ODP.NET wird der Oracle Instant Client auch beim beiliegenden ODBC-Treiber, dem Oracle OLE DB Provider sowie den Oracle Objects for OLE (OO4O) verwendet. OO4O stellen eine native COM-Datenzugriffsschnittstelle dar, die zum Beispiel in zahlreichen Visual-Basic-6.0-Anwendungen auch heute noch zum Einsatz kommt. Mit den in ODAC enthaltenen Providers for ASP.NET stehen Komponenten wie Membership-Provider und Session-State-Provider zur Verfügung, um den Status von Web-Applikationen, die mit ASP.NET entwickelt wurden, persistent in Oracle Datenbanken speichern zu können. Mit den Oracle Database Extensions for .NET ist es möglich, neben PL/SQL und Java auch C#- und VB-Prozeduren innerhalb der Oracle Datenbank zu implementieren. Hierbei ist zu beachten, dass man sich in diesem Falle auf das Windows-Betriebssystem festlegt, da bei Linux und den diversen Unix-Derivaten keine Microsoft .NET-Common-Language-Runtime-Umgebung (CLR) zur Verfügung steht. Eine CLR ist mit einer Java Virtual Machine (JVM) unter Java vergleichbar. Dadurch, dass die Datenzugriffs-Schnittstellen direkt auf OCI aufsetzen, standen früher schon zentrale OCI-Spezialitäten wie die FetchSize zum Reduzieren der Anzahl der Datenbank-Roundtrips unter ODBC, OLE DB und OO4O zur Verfügung und verhelfen heute auch unter ODP.NET zu verbesserten Antwortzeiten.

Es werden Oracle Datenbanken ab Version 9*i* Rel. 2 unterstützt. Im .NET-Zeitalter bilden ODT und ODP.NET den wesentlichen Rahmen für die Anwendungsentwicklung unter Microsoft Visual Studio in Verbindung mit der Oracle Datenbank. ODAC steht im Oracle Technology Network kostenlos zum Download zur Verfügung (http://otn.oracle.com/dotnet). Zwei Versionen werden angeboten: eine auf dem Oracle Universal Installer (OUI) basierende Version, welche die Developer Tools for Visual Studio enthält, sowie eine XCopy-Deployment-Version, die die gewünschten Komponenten im Batchmodus installiert und in den Softwareverteil-Mechanismus eines Unternehmens eingebunden werden kann. Nach dem Entpacken des Xcopy-Paketes werden mit folgendem Aufruf alle enthaltenen Komponenten installiert:

```
install.bat all c:\oracle odac
```

Alternativ können auch einzelne Komponenten installiert werden, wie ODP.NET im nächsten Beispiel:

```
install.bat odp.net20 c:\oracle
odac
```

Neben den entsprechenden Komponenten werden beim Aufruf auch das Installationsverzeichnis und der Registry-Key angegeben. Dadurch ist es möglich, mehrere Oracle Home-Verzeichnisse auf dem Rechner zu verwenden. Die Verbindung zum Oracle-Server erfolgt über EZCONNECT (unter Wegfall der TNSNAMES.ORA).

Bei Verwendung von EZCONNECT ist im Connectstring der Anwendung sowie bei der Verbindung innerhalb des Server Explorers unter der Data Source nur die Angabe des Rechnernamens (Portnummer, falls abweichend von 1521) und des Service-Namens notwendig:

```
Data Source=//Host[:Port]/Ser-
viceName
```

### Entwickeln mit den Developer Tools for Visual Studio (ODT)

Die Oracle Developer Tools for Visual Studio sind ein Add-In für Visual Studio und sind aus der gemeinsamen Visual Studio Industry Partnership zusammen mit Microsoft entstanden. Mit der ersten ODT-11*g*-Version (11.1.0.6.20) fand der Visual-Studio-Entwicklern bekannte Server Explorer Einzug bei den Datenbank-Verbindungen mit der Oracle Datenbank. Der bis zur Version ODT 10*g* zum Einsatz kommende separate Oracle Explorer wurde durch die Integration in den Server Explorer abgelöst. Im Kontextmenü einer solchen Datenbank-Verbindung können sehr komfortabel Schema-Objekte wie Tabellen, Views, Packages, Stored Procedures, Functions, User Defined Types (UDT) etc. angelegt, verändert und wieder gelöscht werden (siehe Abbildung 1). Die IntelliSense-Unterstützung steht auch bei der Entwicklung von PL/SQL-Programmcode zur Verfügung. Hierbei erfolgen Vorschläge zur automatischen Vervollständigung des PL/SQL-Quellcodes, die der Programmierer übernehmen kann. Das Debugging von PL/SQL-Schema-Objekten kann isoliert oder auch fließend

aus .NET-Code (Multi-tier Debugging) erfolgen. Die Fähigkeiten der Tools können durchaus mit denen des SQL-Developers verglichen werden. Für die gemeinsame Oracle- und .NET-Anwendungsprogrammierung ist demnach nur ein Tool erforderlich.

Im Vergleich zur Vorgängerversion ODT 11.1.0.6.20 kommt mit der neuen Version ODT 11.1.0.7.20 die Unterstützung von Users, Roles und Advanced Queues hinzu. Des Weiteren können Automatic Workload Repository (AWR) Snapshots und Automatic Database Diagnostic Monitor (ADDM) Tasks innerhalb von Visual Studio zum Performance-Tuning gestartet und verwendet werden.

### Kompletter Development-Lifecycle mit einer Sourcecode-Versionsverwaltung

„User Designer" legt neue Benutzer an. Dazu ist eine Verbindung mit sysdba-Rechten notwendig. Entsprechende Rollen wie „Connect" und „Resource" werden mit dem Role Designer zugewiesen. Über den Import Table Wizard ist es möglich, Daten aus Excel, Access und weiteren Datenquellen wie dem SQL-Server einzulesen. Dabei werden die Oracle-Tabellen angelegt und mit den Daten gefüllt. Für kleinere Winforms und ASP.NET-Web-Anwendungen kann man auch die in Visual Studio integrierten Wizards verwenden, um mit ein paar Mausklicks einfache Anwendungen automatisch generieren zu lassen. Mit dem Microsoft Query Builder lassen sich die Data Sets aus mehreren Tabellen automatisch erstellen, die die Basis für eine derartige Anwendung bilden. Besonders interessant ist die Verwendung eines Oracle Database-Projekts, das innerhalb einer Visual Studio Solution (Projektmappe) neben dem Applikationsprojekt gemeinsam innerhalb einer Sourcecode-Versionsverwaltung abgespeichert werden kann. Auf diese Weise werden der Code der Anwendung und jener der Datenbank-Logik synchron gehalten. Die Datenbank-Skripte lassen sich dabei versionsabhängig aus einem Oracle Database-Projekt generieren, das innerhalb des Sourcecode-Versionsverwaltungssystems abgespeichert vorliegt. In Projekten wird sehr häufig die Kombination aus Subversion, TortoiseSVN (freier Windows-Client für Subversion) und VisualSVN (preiswertes Visual Studio Plugin, das Subversion und TortoiseSVN integriert) verwendet. Alternativ können auch andere Versionsverwaltungssysteme wie Microsoft SourceSafe 2005 zum Einsatz kommen.

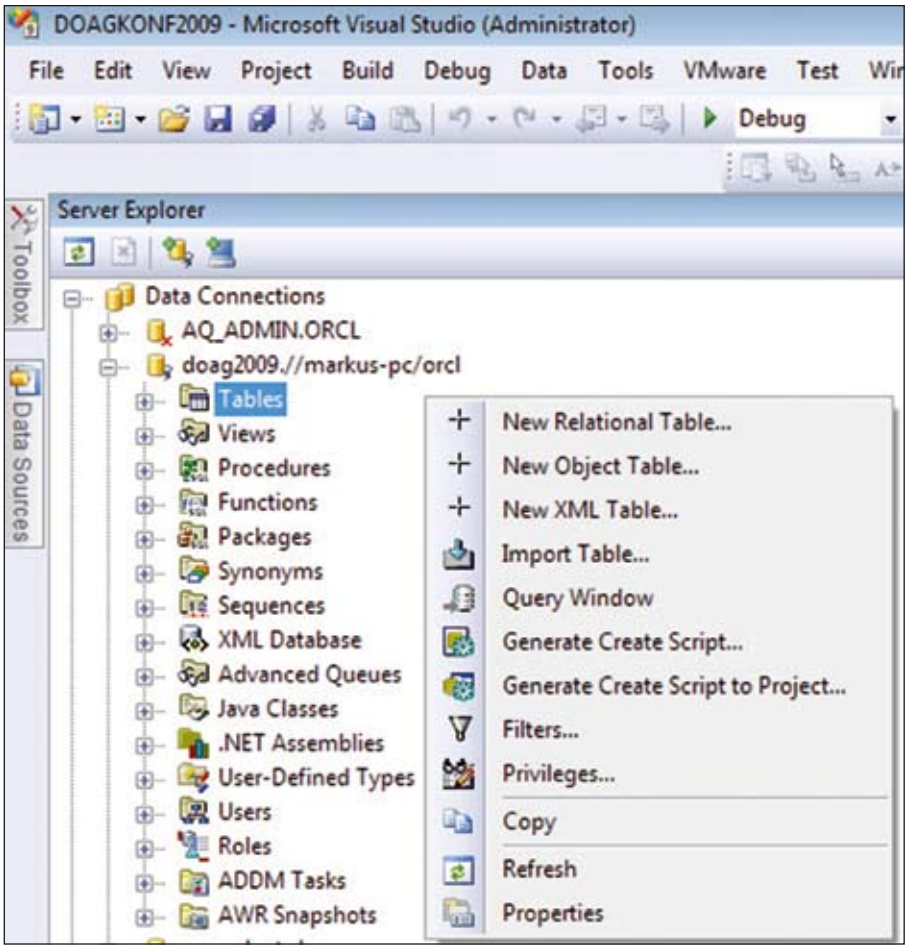



*Abbildung 1: Eine Verbindung (mit EZCONNECT) innerhalb des Server Explorers mit den Oracle Schema-Objekten*

### Performance-Tuning schon während der Anwendungsentwicklung

Der SQL Tuning Analyzer, der direkt als Button im Query Window (Abfragefenster) integriert ist, macht Vorschläge zum Tunen eines SQL-Statements. Auf diese Weise wird zum Beispiel die Anlage eines fehlenden Indexes vorgeschlagen, den man direkt anlegen kann. Im Übrigen lassen sich aus dem Query Window auch die Ausführungspläne eines SQL-Statements anzeigen.

Mit dem Oracle Performance Analyzer lässt sich die Anwendung aus Visual Studio heraus tunen. Dazu werden AWR Reports (Begin Snapshot und End Snapshot) angestoßen, die wiederum über ADDM Tasks ausgewertet werden. Die Vorschläge zum Performance-Tuning lassen sich direkt implementieren. Zudem ist es möglich, auch rückblickend festzustellen, ob eine Anwendung nach Veränderungen an der Datenbanklogik tatsächlich langsamer geworden ist, wenn die Endbenutzer plötzlich über eine schlechtere Performance klagen. Die ADDM Tasks sind im Server Explorer aufgelistet und können direkt eingesehen werden. Innerhalb der Performance-Analyse stehen die „Flaschenhälse" unter Findings, beispielsweise TOP-SQL by DB Time, CPU Usage und Virtual Memory Paging.

Die Oracle Developer Tools lassen sich unter den Visual-Studio-Einstellungen konfigurieren (|Tools|Options). Für die Kompilierung von PL/SQL-Code lässt sich neben der interpre-

tierten auch die native Kompilierung einstellen. Auf diese Weise kann eine weitere Performance-Verbesserung erzielt werden.

### Weitere nützliche Helfer

Im Server Explorer können die Schema-Objekte, die angezeigt werden sollen, nun gefiltert werden. Über ein spezielles Paging-Verfahren wird die Performance verbessert, wenn viele Objekte in der Datenbank vorhanden sind. Das Umbenennen von Schema-Objekten im Server Explorer ist möglich. Datenbank-Skripte können nun über mehrere Verbindungen hinweg generiert und mehrere Funktionen und Stored Procedures können auf einmal kompiliert werden. Die Unterstützung von Oracle Streams Advanced Queuing (AQ) ist ebenfalls neu. Die Queues und Queue Tables lassen sich komfortabel anlegen, verändern und administrieren.

Beim Kompilieren der .NET-Anwendung auf einem Windows 64-Bit-System ist es notwendig, die Compiler-Einstellungen unter den Projekt-Eigenschaften auf x86 umzustellen, da der neue Provider derzeit nur für 32-Bit zur Verfügung steht. Zudem bereitet das Debugging unter 64-Bit mit x64-Compilereinstellungen oft Probleme.

### Oracle Data Provider (ODP.NET)

Oracle implementiert mit ODP.NET einen ADO.NET-konformen Provider, der einen nativen Zugriff auf die Oracle Datenbank ermöglicht. Dabei werden neben ADO.NET generischen Standards (vorgegeben durch abstrakte Klassen und Interfaces) auch die Advanced Features der Oracle Datenbank unterstützt. Dazu gehören neben der Oracle RAC- und Data-Guard-Unterstützung auch die speziellen Datentypen, wie Secure-Files, XML-Type und die User Defined Types (UDT), sowie die Performance- und Security-Features. Der Oracle Server kann auf einem beliebigen Betriebssystem laufen, während für eine .NET-Client-Umgebung ein Windows 32-Bit- oder 64-Bit-Betriebssystem Verwendung findet. Unterstützt sind das .NET Framework 1.0 und höher; derzeit aktuell ist die Version 3.5 SP1. Wichtig ist auch, dass der DB-Client und die Server-Version nicht übereinstimmen müssen. So kann problemlos mit der aktuellen ODP.NET-Version 11.1.0.7.20 (Basis ist der Oracle Instant Client in der Version 11.1) auf eine Datenbank-Version 9*i* R2 oder 10*g* zugegriffen werden.

### Oracle Streams Advanced Queuing (AQ)

Die auf Standards basierende Enterprise-Messaging-Infrastruktur für die asynchrone Nachrichtenübertragung zwischen Applikationen steht nun auch einer .NET-Umgebung durchgängig zur Verfügung. AQ unterstützt Point-to-Point und Publish/Subscribe Queues. Es stellt eine Alternative bzw. Ergänzung zu Middleware-Produkten wie Microsoft Message Queuing (MSMQ) und zu .NET-Programmier-Modellen wie den Windows Communication Foundation (WCF) oder Windows Workflow Foundation (WF) dar. AQ als Basistechnologie ist ideal für Kommunikation in verteilten Umgebungen. Es stehen neben der für ODP.NET auch APIs für PL/SQL, OCI/OCCI, Visual Basic (Oracle Objects for OLE OO4O) und Oracle Java Message Service (OJMS) zur Verfügung. Listing 1 zeigt in C#, wie eine Queue verwendet und eine Nachricht in eine Queue eingestellt wird.

```
OracleConnection con = new
OracleConnection(constr);
OracleAQQueue queue = new
OracleAQQueue(„scott.test_q“,
con);
con.Open();
queue.MessageType = OracleAQ-
MessageType.Raw;
OracleAQMessage enqMsg = new
OracleAQMessage();
byte[] bytePayload = { 0, 1, 2,
3, 4, 5, 6, 7, 8, 9 };
enqMsg.Payload = bytePayload;
queue.EnqueueOptions.Visibili-
ty = OracleAQVisibilityMode.
OnCommit;
queue.Enqueue(enqMsg);
Console.WriteLine(“Produzent
hat folgenden Nachrichteninhalt
eingestellt: “
+ ByteArrayToString(enqMsg.Pay-
load as byte[]));
```

*Listing 1*

Weitere Funktionen sind:

- *Promotable Local Transaction Support*
  Die Database 11*g* R1 oder höher kann als Resource Manager unter Verwendung der „system.transactions“ agieren, und eine lokale Transaktion wird zu einer verteilten promotet, sobald eine weitere Verbindung zu einer anderen Datenressource hinzukommt.

- *High Availability Event Notification und Callback*
  Über einen Event Handler kann direkt auf Status-Änderungen einer Oracle-Datenbank reagiert werden, dies betrifft insbesondere RAC- und Data-Guard-Umgebungen. Das automatische Bereinigen der ungültigen Verbindungen innerhalb eines ODP.NET-Connection-Pools über die HA-Events funktionierte schon mit älteren ODP.NET-Versionen.

- *Oracle Database-Klasse*
  Mit ODP.NET ist es jetzt möglich, eine Oracle Datenbank hoch- beziehungsweise herunterzufahren.

- *Oracle Permission-Klasse*
  Code Access Security (CAS) kann mit der Oracle Permission-Klasse hergestellt werden. Dadurch ist sichergestellt, dass nur berechtigte .NET-Assemblies auf Oracle zugreifen.

- *Sonstige Verbesserungen*
  ODP.NET kommt jetzt mit weniger Speicherbedarf aus und die Daten werden schneller aus dem Oracle Server übertragen. An dieser Stelle wurde unter anderem der Oracle DataAdapter überarbeitet. Die „StatementCacheSize“ passt sich dynamisch den Gegebenheiten an.

### Tipps beim Umgang mit ODP.NET

Aus Oracle User-Defined Types (UDT) lassen sich die .NET Custom Types generieren. Auf diese Art und Weise können Oracle UDTs nahtlos innerhalb einer .NET-Applikation verwendet werden. Dies kann eine interessante Alternative zu O/R-Mappern darstellen.

Die Oracle XML Database und Oracle Spatial sind ebenfalls transparent in .NET-Anwendungen verwendbar. Um mandantenfähige Anwendungen unter .NET erstellen zu können, bietet sich Virtual Private Database (VPD) an, die Bestandteil der Enterprise Edition ist.

Mit der Oracle Database Change Notification können .NET-Clients, die mit Data Sets (unter ADO.NET verbindungslose In-Memory-Datencaches) arbeiten, laufend benachrichtigt werden, sobald sich ihr Result Set in der Datenbank geändert hat. Diese laufende Benachrichtigung ist ein Alleinstellungsmerkmal der Oracle Datenbank.

Der Client Result Cache (seit Oracle Database 11*g* Rel. 1 verfügbar) kann transparent mit ODP.NET genutzt werden.

**Multiple Active Result Sets**

Bei der Massendatenverarbeitung hat sich die Verwendung der PL/SQL Associative Arrays bewährt, die auch unter ODP.NET genutzt werden können. Listing 2 zeigt, wie man innerhalb eines anonymen PL/SQL-Blocks die Bulk-Operation FORALL-Anweisung verwendet, die INSERT-Anweisungen als Ganzes an den Oracle-Server übergibt. Drei Parameter in Form von PL/SQL Associative Arrays wurden zuvor mit drei C#-Arrays mit jeweils 100.000 Einträgen befüllt (Bulk Binding). Zum Vergleich ist in Listing 2 auch eine Loop-Anweisung enthalten, um die Dauer der Einzel-Inserts mit der FOR ALL-Anweisung vergleichen zu können. Während die FORALL-Anweisung unter einer Sekunde benötigt, sind bei der Schleife fünf Sekunden notwendig. Über diese anonymen PL/SQL-Blöcke innerhalb des Command-Textes des Command-Objektes können auch mehrere SQL-Anweisungen zusammenfasst und innerhalb einer Transaktion mit einem DB-Roundtrip an Oracle übergeben werden. Dieses Konstrukt nennt sich „Multiple Active Result Sets“ (MARS), siehe Listing 2.

```
//Größe für Arrays festlegen, Arrays anlegen und befüllen
int MAXSIZE = 100000;
Int32[] persid = new Int32[MAXSIZE];
string[] persnana = new string[MAXSIZE];
string[] persvona = new string[MAXSIZE];
for (int i = 0; i < MAXSIZE; i++)
{
   persid[i] = i;
   persnana[i] = „NACHNAME“ + i;
   persvona[i] = „VORNAME“ + i;
}
//Arrays für Bulk Binds erstellen
OracleParameter pidParameter = new OracleParameter(„p_person_id“,
OracleDbType.Int32);
pidParameter.Direction = ParameterDirection.Input;
pidParameter.CollectionType = OracleCollectionType.PLSQLAssociative-
Array;
pidParameter.Size = MAXSIZE;
//Speicherstruktur für DB
for (int i = 0; i < MAXSIZE; i++)
{
   pidArrayBindSize[i] = 100;
}
pidParameter.ArrayBindSize = pidArrayBindSize;
//eigentliches Bind
pidParameter.Value = persid;
command.Parameters.Add(pidParameter);
[…]
//Arrays werden als Parameter übergeben. FORALL schreibt Daten
//mit einem DB Roundtrip in Oracle
OracleCommand command = new OracleCommand(„begin forall i in 1..“ +
MAXSIZE + „ insert into personentemp(id, nachname, vorname) values
(:p_person_id(i), :p_nachname(i), :p_vorname(i)); end;“, connection);
//klassische Schleife benötigt um Faktoren länger
OracleCommand command = new OracleCommand(„begin for i in 1.. „ +
MAXSIZE + „ loop insert into personentemp(id, nachname, vorname)
values (:p_person_id(i), :p_nachname(i), :p_vorname(i)); end loop;
end;“, connection);
```

*Listing 2*

Die Fetch- und Rowsize-Eigenschaft des OracleDataReader-Objekts beziehungsweise OracleCommand-Objekts reduziert zusätzlich die Anzahl der DB-Roundtrips beim Abfragen von größeren Datenmengen, was sich nochmals spürbar auf die Anwortzeiten auswirkt. Die FetchSize kann man wie folgt dynamisch anpassen:

```
dr.FetchSize = cmd.RowSize *
numRows;
```

Getestet werden kann dies mit einer Testtabelle, bestehend aus einer ID-Spalte vom Daten-Typ „Number“ und einer Daten-Spalte von Daten-Typ „Varchar2“. Sie enthält 100.000 Datensätzen und wird über den DataReader sequentiell ausgelesen. Durch die Angabe von „numRows = 10.000“, multipliziert mit einer RowSize von „120“, ergibt die daraus resultierende FetchSize lediglich zehn Datenbank-Roundtrips. Das Testprogramm wurde in nur 0.05 Sekunden durchlaufen. Bei der Angabe von „numRows = 1“ dauert der Programmdurchlauf zehn Sekunden, da entsprechend viele Roundtrips notwendig sind. Die Größe der RowSize-Eigenschaft wird übrigens über das Command-Objekt automatisch aus den Metadaten errechnet.

ODP.NET kann „Input REF-Cursor“ als Parameter wieder an Oracle übergeben (ParameterDirection.Input), und zwar aus „Output REF-Cursorn“, die zuvor vom Oracle-Server abgefragt wurden (ParameterDirection.Output). So können (Massen-) Datenverarbeitungen komplett auf dem Oracle-Server ablaufen, sodass das Netzwerk entlastet und die Daten wirklich nur dann übertragen werden, wenn sie der Client tatsächlich benötigt. Zusammen mit dem ODP.NET Connection Pool

(auch transparent mit RAC- und Data-Guard-Umgebungen) und dem Statement Caching können große Anwendungsumgebungen auch unter .NET performant realisiert werden.

Beim Einsatz der ADO.NET Datasets sollte auch jeden Fall darauf geachtet werden, lieber direkt Arrays zu verwenden, wenn mit größeren Datenmengen gearbeitet wird. Die Frage des richtigen Einsatzes eines O/R-Mappers wie etwa dem ADO.NET-Entity-Framework hängt ebenfalls von der zugrundeliegenden Datenmenge ab. Auch hier sollten auf jeden Fall die oben beschriebenen Szenarien beachtet und bei Bedarf vorgezogen werden.

### Microsoft stellt die Entwicklung ihres Oracle Clients ein

Microsoft hat angekündigt, ihren Oracle Provider (System.Data.OracleClient) zukünftig nicht mehr weiterzuentwickeln. Bei der Umstellung auf ODP.NET muss im Visual-Studio-Projekt zuerst eine Referenz auf das Assembly Oracle.DataAccess.dll erstellt werden. Desweiteren müssen die Namespaces „Oracle.DataAccess.Client“ und „OracleDataAccess.Types“ im Sourcecode hinzugefügt sein. Beim Verhalten hinsichtlich des Parameter-Bindings benutzt ODP.NET standardmäßig „Bind by Position“, während der Microsoft OracleClient standardmäßig „Bind by Name“ verwendet. Um dieses Verhalten zu erreichen, muss die BindByName-Eigenschaft des OracleCommand-Objektes auf „true“ gesetzt werden. Die umgestellte Anwendung sollte auf jeden Fall getestet werden.

## Fazit

Mit den neuen Features von ODP.NET stehen .NET-Projekten weitere interessante Möglichkeiten zu Verfügung. Auch die Oracle-Kennern bereits seit Jahren bekannten Fetch- und Bulk-Operationen bringen unter ODP.NET einen enormen Leistungsschub. Die neuen Oracle Developer Tools for Visual Studio ermöglichen es, den kompletten Development Lifecycle aus Visual Studio abzudecken, ohne auf andere Tools ausweichen zu müssen. Obwohl Microsoft die Entwicklung des eigenen Oracle Clients zukünftig einstellen möchte, bleibt festzuhalten, dass trotz eines überschaubaren Umstellungsaufwands der Mehrwert überwiegt. Mit ODP.NET kann die gesamte Leistungsfähigkeit der Oracle Datenbank genutzt werden, während der Oracle Client von Microsoft nur die generischen ADO.NET-Funktionalitäten implementiert hat.

### Weitere Informationen

Oracle stellt auf ihrer „Microsoft Community Page“ unter http://www.oracle.com/global/de/community/platform/index.html Webcasts zur Integration mit Microsoft bereit. Auf diese Art lassen sich die hier vorgestellten Technologien und Features bei Bedarf im Detail nachvollziehen.

**Kontakt:**
Markus Kißling
markus.kissling@oracle.com

# Oracle-Provider im .NET-Framework

Alexander Schmidt, SAB Ingenieurgesellschaft mbH

**Die Zahl der .NET-Applikationen steigt und damit auch die Menge an Projekten, die auf Oracle Datenbanken zugreifen müssen. Dieser Artikel gibt einen Überblick über den Stand der Entwicklungen auf .NET-Seite, beleuchtet einige potenzielle Risiken und bringt ein wenig mehr Klarheit in den .NET-Dschungel.**

Es vergeht keine Woche, in der nicht irgendwelche Neuerungen im nahezu unübersichtlichen Bereich der .NET-Entwicklung herauskommen. Neben Microsoft sind Hunderte von Dritt-Anbietern im Markt versammelt, so dass eine nutzbringende Gesamtübersicht nicht existiert. Es gibt aber einen gewissen Status quo sowie eine Technologie-Roadmap, die recht gut aufzeigen, was den .NET-Entwickler und damit auch den Oracle Administrator in nächster Zeit erwartet.

### Zugriffstechnologien

.NET-Anwendungen (im .NET-Sprachgebrauch „Assemblies“ genannt) benötigen immer sogenannte „Provider“, um auf Datenbanken zugreifen zu können. Solche, in der Microsoft-Nomenklatur „ADO.NET-Provider“ (ADO = active data objects) genannte Bibliotheken, bieten immer einen Basissatz an Funktionen an. Dazu gehören beispielsweise das Verbinden zu Datenbanken über Connection-Objekte, das Abfragen über Commands und das Durchlaufen von Ergebnissen über DataReader.

Für viele Datenbanken (SQL Server, Oracle, MySQL etc.) existieren bereits Provider, die die jeweiligen Eigenheiten der Backend-Systeme kennen und berücksichtigen. Bis zur Version 3.0 des .NET-Frameworks bot sogar Microsoft einen eigenen Oracle-Provider an. Nicht weiter überraschend war sicherlich, dass dieser nicht gerade durch Performance-Wunder auf sich aufmerksam machte. Das überließ man in Redmond dann doch dem SQL-Server-Provider und rückte damit aus Haussicht die Prioritäten gerade. Microsoft hat nun angekündigt, den Oracle-Provider nicht weiter entwickeln zu wollen.

Die entstandene Lücke für Oracle-Entwickler wurde aber relativ schnell geschlossen. Derzeit werden drei ernstzunehmende Oracle-.NET-Provider angeboten: Oracle ODP.NET, DataDirect Connect for ADO.NET sowie DevArt dotConnect for Oracle. Zwischen diesen Bibliotheken besteht eine für .NET-Entwickler entscheidende Abgrenzung. Nur der ODP.NET-Provider erfordert einen installierten Oracle Client auf der Maschine, auf der die .NET-Assembly ausgeführt werden soll. Da dies aus Verteilungs-Gesichtspunkten einem GAU gleichkommt, wird dieser Provider meist nur zu Testzwecken beziehungsweise in Intranet-Lösungen genutzt. Da er noch dazu nicht zu den schnellsten Zugriffs-Möglichkeiten zählt (Oracle will gerade Sun übernehmen, den Erfinder von Java), bleibt der Provider in diesem Artikel außen vor.

### Die Funktionsweise

Aus Sicht eines .NET-Entwicklers sieht der Verbindungsaufbau mit einer Oracle-Datenbank ungefähr so aus:

```
OracleConnection cnnThis = new
OracleConnection();
cnnThis.ConnectionString = “”;
cnnThis.Open();
…
cnnThis.Close();
cnnThis.Dispose();
```

Dieses stark vereinfachte Listing zeigt nur den prinzipiellen Aufbau. Über die Parameter im ConnectionString lassen sich Einstellungen zum Beispiel für das Pooling oder den Direkt-Zugriff ohne Oracle-Client konfigurieren. Je nach verwendetem Provider werden bei Anwendung der Open()-Methode die Aufrufe an die Oracle Datenbank geleitet.

Wie bei vielen anderen Programmier-Umgebungen auch ist es hier ganz entscheidend, richtig mit den angebotenen .NET-Objekten umzugehen. Aus den Praxis-Erfahrungen haben sich bei dieser Zugriffsmethode vor allem drei Problemfelder herauskristallisiert:

- Vergessene Close() und Dispose()
- Zu lange offen gelassene Verbindungen
- Auslassung parametrisierter Abfragen

### Problemfeld 1: Close() und Dispose()

Das explizite Schließen einer geöffneten Verbindung wird auch in anderen Programmier-Umfeldern gern vergessen. Viel mehr ins Gewicht fällt jedoch das unscheinbare Dispose(). Diese Methode veranlasst den .NET-Garbage-Collector (ein Objekt-Aufräumdienst, den es auch bei Java gibt), ein Objekt als „Bereit zum Entsorgen“ zu markieren. Die Ressourcen des Objekts werden dann so schnell wie möglich freigegeben. Unterbleibt das Dispose(), kann es gerade bei parallelisierten Anwendungen dazu führen, dass das Connection-Pooling unterlaufen wird beziehungsweise die Last der geöffneten Session steigt. Ein typischer Fehler in diesem Bereich ist „ORA-01000: maximum opened cursors exceeded.“

### Problemfeld 2: Offen gelassene Verbindungen

ADO.NET wird von vielen alten Microsoft-Hasen noch immer so genutzt wie früher ADO. In der alten Welt war es durchaus üblich, Eingabeformulare direkt an die Datenbank zu binden. Tippte in diesem Szenario jemand etwas in ein Textfeld, so wurde es durch

eine Dauerverbindung im Hintergrund direkt in die Datenbank geschrieben. Dieses Szenario führte lange Zeit dazu, dass ADO-Anwendungen als träge und lastintensiv galten. Besonders die Administratoren waren nicht immer begeistert, wenn sie vom Einsatz von ADO-Anwendungen erfuhren. ADO.NET räumt damit auf und baut Verbindungen sofort nach der Nutzung ab. Natürlich kann ein Programmierer auch dieses Modell unterlaufen, indem er beispielsweise interne Routinen zum Nachbilden von Verbindungs-Pools schreibt (durchaus nicht so selten, wie man glaubt). Wichtig ist hier die Nutzung der angebotenen Pooling-Technologien der Provider.

### Problemfeld 3: Auslassung parametrisierter Abfragen

Ein oft gesehenes Konstrukt kann in .NET wie folgt aussehen:

```
public int
GetCustomerFirstname(int intCu-
stomerID)
{
String strReturn = string.
Empty;
string strSQL = string.
Format(@"SELECT CUST_FIRSTNAME
  FROM OE.CUSTOMERS
  WHERE CUSTOMER_ID =
      {0}",
intCustomerID);
OracleConnection cnnThis = new
OracleConnection();
cnnThis.ConnectionString = "";
cnnThis.Open();
OracleCommand cmdThis = new
OracleCommand();
cmdThis.Connection = cnnThis;
cmdThis.CommandText = strSQL;
intResult = cmdThis.ExecuteSca-
lar().ToString();
cnnThis.Close();
cnnThis.Dispose();
return strReturn;
}
```

Die Funktion nutzt ein Oracle-Command-Objekt, um eine Abfrage gegen die Datenbank laufen zu lassen. Da nur ein Feld in einem Rowset der Länge „1" erwartet wird, nutzt die Funktion die ExecuteScalar()-Methode des Command-Objekts, die auf solche Abfragen hin optimiert ist. Aus Sicht des .NET-Entwicklers ist damit alles gut.

Für den Oracle Administrator stellen solche Konstrukte einen Albtraum dar, da sie den Optimizer komplett außer Gefecht setzen. Jede Abfrage kommt mit einem neuen SQL-String daher, wird daher im Cache nicht wiedergefunden und es werden keine gespeicherten Ablaufpläne genutzt (oder zumindest eher selten). Dabei ist die Lösung aus .NET-Sicht denkbar einfach:

```
public int
GetCustomerFirstname(int intCu-
stomerID)
{
String strReturn = string.
Empty;
string strSQL = @"SELECT CUST_
FIRSTNAME
   FROM OE.CUSTOMERS
   WHERE CUSTOMER_ID = :0";
OracleConnection cnnThis = new
OracleConnection();
cnnThis.ConnectionString = "";
cnnThis.Open();
OracleCommand cmdThis = new
OracleCommand();
cmdThis.Connection = cnnThis;
cmdThis.CommandText = strSQL;
OracleParameter parThis = new
OracleParameter(":0",
intCustomerID);
cmdThis.Parameters.
Add(parThis);
cmdThis.Prepare();
intResult = cmdThis.ExecuteSca-
lar().ToString();
cnnThis.Close();
cnnThis.Dispose();
return strReturn;
}
```

Die Änderungen betreffen zum einen den SQL-String, dem die Customer-ID nicht direkt übergeben wird. Dem OracleCommand wird stattdessen ein OracleParameter mit dem Wert der ID übergeben. Wichtig ist dann noch der Aufruf von Prepare() auf dem Command-Objekt, da dieser die eventuelle Nutzung des Optimizers sicherstellt.

Anhand der Beispiele im Internet, in einigen Büchern und vor allem aus eigenen Erfahrungen in Kundenprojekten ist festzustellen, dass derzeit kaum parametrisierte Abfragen zum Einsatz kommen. Dies ist einer der derzeit entscheidenden „Flaschenhälse" beim .NET-Zugriff auf die Oracle Datenbanken.

### Performance

Um die aktuelle Marktsituation möglichst objektiv zu betrachten, wurde ein kleines Benchmark-Programm in .NET entwickelt. Dabei wurden insgesamt drei Tests (bezeichnet mit A, B und C) mit den drei eingangs erwähnten Providern durchgeführt. Test A fragt dabei 1.000-mal alle Spalten der Tabelle HR.EMPLOYEES mit einem DataReader ab und durchläuft diesen bei jeder Abfrage komplett. Test B führt Test A 5.000-mal durch. Test C legt 200-mal eine Tabelle im Schema HR an und löscht sie dann wieder.

Der dotConnect-Provider von DevArt wurde jeweils im Oracle-Client- und im Direkt-Modus getestet. Da der DataDirect-Provider niemals einen Oracle-Client nutzt (selbst wenn einer installiert ist und eine Verbindung über TNSNames konfiguriert wurde), reichte hier ein Test aus. Die Ergebnisse wurden in Laufzeit-Sekunden gemessen. Keiner der vorhandenen Performance-Tuning-Maßnahmen bei den jeweiligen Providern wurde genutzt.

DotConnect verweist in diesem nicht repräsentativen einfachen Test die beiden anderen Provider auf die Plätze und erledigt die beiden Aufgaben A und B um 60 Prozent schneller als das vergleichbare dotConnect im Direkt-Modus. ODP.NET ist mit dotConnect vergleichbar und im B-Test sogar etwas schneller.

Im C-Test, der das praxisfernste Beispiel simuliert, stürzte dotConnect im Direktmodus reproduzierbar ab. Insgesamt vermittelte DataDirect den Eindruck, ein sehr leistungsorientiertes Produkt zu sein. Bei dotConnect als zweitem kommerziellen Tool stechen vor allem die ausgefeilten AddOns ab der Professional-Variante (Kosten: 300 Dollar) zum Visual Studio ins Auge. Für 100 Dollar weniger bekommt man bereits den DataDirect-Provider, wobei sich dieses Produkt vom Lieferumfang her eher mit der 150 Dollar teuren Standard-Edition von dotConnect vergleichen lassen kann.

Die Ergebnisse liefern dabei natürlich nur einen Abriss und sind nicht für alle Bereiche repräsentativ. DataDirect merkt man an, dass es das technisch

ausgereifteste Protokoll hat. Herstellerangaben zufolge wird dieses in enger Zusammenarbeit mit Oracle erarbeitet und besteht zu 100 Prozent aus „managed code“.

### Die Zukunft des Datenzugriffs

Der Einsatz objektorientierter Programmiersprachen wie jener des .NET-Frameworks (C#, VB.NET etc.) führte zu einer Vielzahl neuer Frage- und Problemstellungen für Entwickler. Das augenscheinlichste Problem ist das des Arbeitsaufwands. Neben der gesamten Geschäftslogik und dem GUI-Design obliegt es dem Entwickler, die Daten einer relationalen Datenbank in sein Objektmodell zu transferieren. Dazu sind Objekte nötig, die Datenverbindungen aufbauen, solche, die Daten abfragen und andere, die die flachen Tabellen und ihre Beziehungen in Objekthierarchien übersetzen.

Auch hier gibt es mittlerweise ausgereifte Tools, die sogenannten „objektrelationalen Mapper“ (ORM). Bekanntester Vertreter dieser Zunft ist „NHibernate“, ein Open-Source-NET-Ableger des bei Java-Entwicklern sehr beliebten Hibernates. Letztlich kann man sich die ORM als zusätzliche Schicht zwischen der Anwendungsdomäne und dem RDBMS vorstellen, die jegliche Datenzugriffe aus Sicht des Entwicklers transparent macht.

Die Entwicklungen in diesem Bereich münden inzwischen in einem Microsoft-Produkt namens „ADO Entity Framework“, das mit Erscheinen des Visual Studio 2010 das präferierte Zugriffs-System auf relationale Datenbanken darstellen wird. Auf der PDC 2009 – der Microsoft Entwickler-Hausmesse – bescherten die Redmonder der anwesenden Entwicklergemeinde größere Sicherheit mit der Aussage, das Entity Framework sei die präferierte Datenzugriffs-Technologie des Konzerns.

Technisch gesehen startet der Entwickler einen Assistenten in Visual Studio, der es ihm ermöglicht, Objekte und deren Relationen zueinander aus einer bestehenden Datenbank abzuleiten. Aus Oracle-Sicht gesehen fragt der Assistent das Data Dictionary ab und erstellt eine Anzahl von Klassen (für jede Tabelle eine). Fremdschlüssel werden automatisch zu Auflistungen innerhalb einer Klasse. Am Beispiel der Tabelle CUSTOMERS aus dem OE-Schema sieht das Ergebnis des Assistenten aus, wie in Abbildung 1 gezeigt.

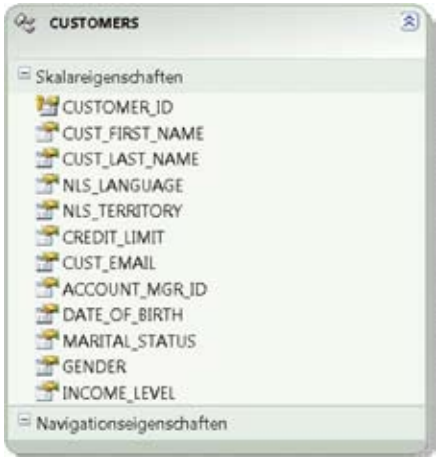



*Abbildung 1: Ergebnis des Entity-Framework-Assistenten*

Auch der umgekehrte Weg wird nun möglich. Das heißt, der Entwickler gestaltet sein Objektmodell zunächst in Visual Studio und lässt den Assistenten dann Tabellen und Relationen auf der Oracle-Datenbank generieren. Das entstandene Objektmodell kann der Entwickler verändern oder erweitern und bei Bedarf mit der „echten“ Datenbank synchronisieren.

Prinzipiell sind diese Ansätze kritisch zu betrachten. Auf der einen Seite ersparen sie dem Entwickler eine Menge Zeit. Diese Ersparnis wird allerdings durch ein erhöhtes Risiko an unsauberen Datenbank-Designs und ineffizienten Zugriffen erkauft. Dass die Technologie noch nicht ganz ausgereift ist, zeigt auch die Tatsache, dass der Assistent mit speziellen Konstrukten wie „Nested Tables“ nicht viel anzufangen weiß. In Abbildung 1 fehlt die Spalte „CUST_ADDRESS“ komplett.

Viel entscheidender ist aber die Tatsache, dass das Verantwortungsbewusstsein der Entwickler gegenüber den Datenbank-Systemen schrumpft. Die Hersteller fördern die immer weiter steigende Transparenz der eingesetzten Backends, und somit entfernen sich die Entwickler immer mehr von den bisherigen Paradigmen. Im Unternehmenseinsatz heißt dies, mehr als bisher darauf zu achten, eine saubere und kritische Diskussion zwischen Entwicklern und DB-Administratoren zu gewährleisten und Richtlinien zu schaffen. Denn nicht .NET als Infrastruktur macht den Code schlecht, sondern Unwissenheit und Bequemlichkeit.

**Kontakt:**
Alexander Schmidt
schmidt@sab-team.de

# Apex – Kreativität durch die (Daten)-Bank

Thomas Zielbauer, MAN Nutzfahrzeuge

**Durch Kombination von unterschiedlichen, ineinander greifenden Web-Technologien kann Apex wesentlich dynamischer Daten aufbereiten, als es mit den Standard-Editor-Methoden der Fall zu sein scheint.**

Wie die meisten wissen, ist Oracle Apex eine Sammlung von Möglichkeiten, die helfen, die von internen oder externen Kunden geforderten Daten darzustellen. Man erstellt mit Apex Web-Anwendungen, in denen man den Kunden Daten präsentiert. Es handelt sich in der Tat um eine Präsentation, wie die Entwicklung des Internets in den letzten Jahren belegt. Es ist für den Endanwender zunehmend wichtiger, in welcher Form ihm Daten verfügbar gemacht werden – seien es unterschiedliche Formate (Excel-Export, PDF etc.) oder ein standardisiertes Design der Anwendung (corporate identity). Der Artikel stellt keine Musterlösungen oder endgültige Ergebnisse vor – er gibt lediglich einen Einblick, was mit Apex, Verständnis für Web-Technologien und Kreativität alles möglich ist.

## Der Ausgangspunkt

Basis für alle hier dargestellten Szenarien ist ein und dieselbe Datentabelle. In dieser simpel gehaltenen Tabelle findet sich allerdings eine Situation wieder, die man bestimmt aus einem aktuellen oder vergangenen Projekt kennt (siehe Abbildung 1).

| ID | Kategorie | Produkt | Preis |
|---|---|---|---|
| 1 | Computer | Grafikkarte X | 299 |
| 2 | Computer | Grafikkarte X | 250 |
| 3 | Handy | Handy Modell X | 350 |
| 4 | Handy | Modell Y | 250 |
| n | ... | ... | ... |

*Tabelle 1: Produkttabelle*

Diese Tabelle ist bewusst einfach gehalten. Es ist natürlich möglich, die Kategorie in einer separaten Tabelle auszulagern und in dieser, unserer Produkttabelle, anstatt der kompletten Kategoriebezeichnung (z.B. Computer) beispielsweise lediglich eine Kategorie-ID zu speichern. Deutlicher wird das Ganze jedoch in der dargestellten Form.

## Unser Ziel

Ein Kunde möchte nun über eine Apex-Anwendung auf die Daten in der dargestellten Produkttabelle zugreifen. In welcher Funktionalität er das möchte, sei hier jetzt nicht beachtet. Es soll lediglich dem Kunden eine Möglichkeit gegeben werden, die Daten übersichtlich einzusehen.

Mit den vorhandenen Apex-Kenntnissen legen wir eine neue Anwendung an und erzeugen darin eine leere Seite. Auf dieser wird per Region-Erstellungsassistent eine SQL-Berichtsregion angefertigt.

Die zuvor angelegte Tabelle wird in der Berichtsregion mit dem folgenden Beispiel ungefiltert, aber sortiert abgefragt:

```
SELECT ID, Kategorie, Produkt,
Preis
FROM Demo_Artikel
ORDER BY Kategorie, Produkt
```

Abbildung 2 zeigt den Bericht in der Ansicht für den Anwender, abhängig von dem der Apex-Anwendung zugeordneten Theme.

Durch das im Apex-Berichtsentwurf festgelegte Gruppenwechselformat bekommt der Bericht eine gewisse Übersicht, da die Kategorien visuell ein wenig voneinander Abstand bekommen.

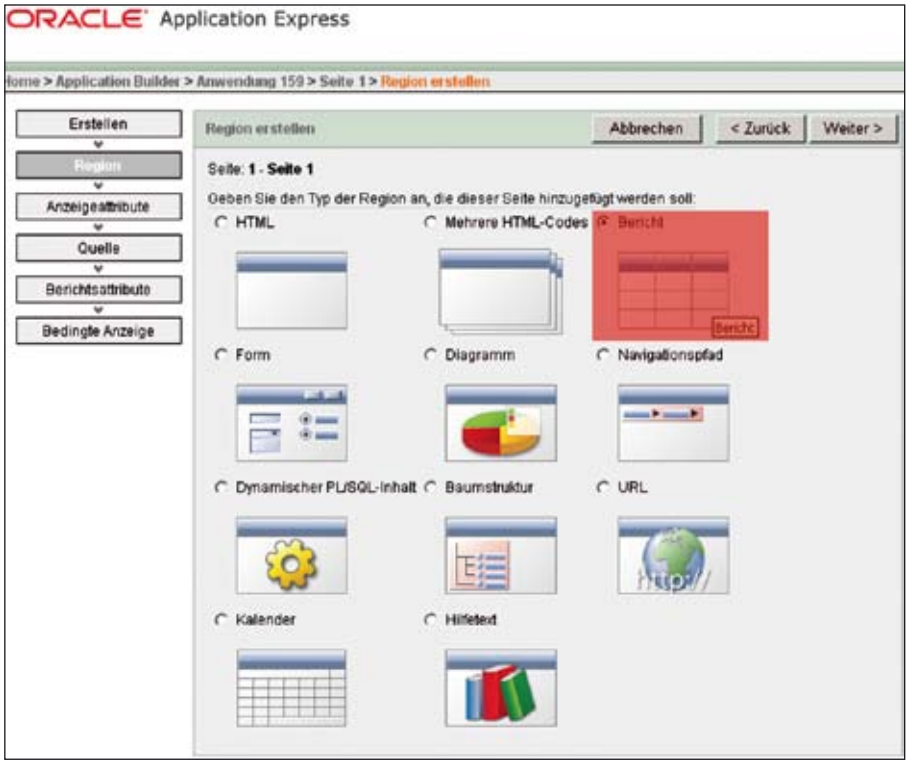



*Abbildung 1: Den gewünschten Bericht erzeugen*

Info

Dieser Bericht wurde mit dem Berichtsassistenten erstellt.

Standardbericht Artikel

| KATEGORIE | PRODUKT | PREIS |
|---|---|---|
| Computer | Grafikkarte X | 299 |
| | Grafikkarte Y | 250 |
| | LCD Moniter X | 199 |
| Handy | Handy Modell X | 350 |
| | Modell Y | 250 |
| Haushalt | Mikrowelle C | 180 |
| | Pfannenset | 300 |
| | Toaster X | 35 |

1 - 8

*Abbildung 2: Der Apex-Standardbericht*

Bei größeren Datenmengen kann aber auch diese Darstellung unangenehm für den Anwender werden.

### Schritt 1: CSS und Apex kombinieren

Die im Webdesign standardisierte Formatsprache Cascading Stylesheets (CSS) wurde im Laufe der Jahre immer mächtiger – Standards wurden definiert. Davon profitiert auch unsere Webanwendung unter Apex. Simpel ausgedrückt gestaltet CSS unsere HTML-Seite nicht nur ansprechender, sie verschlankt auch den Seitencode und trennt das Design vom Seiteninhalt. Ändern oder gar Austauschen des Designs ist problemlos möglich. Allerdings würde es an dieser Stelle den Rahmen sprengen, das Thema „CSS“ ausführlich zu beschreiben.

Eine Möglichkeit, die wir hier aufgreifen, ist die farbliche Unterscheidung unserer Kategorien (Computer, Handy etc.) voneinander. Apex bietet an dieser Stelle einen interessanten, aber nicht offensichtlich erkennbaren Vorteil: Nahezu an jeder Stelle in Apex kann HTML-Code eingefügt werden. Wir werden zuerst, der Einfachheit halber, direkt CSS-Klassen im Seitentemplate anlegen. Für Entwickler, die noch nicht sehr viele Kenntnisse in Apex sammeln konnten, ist der entsprechende Bereich im Seitenentwurf markiert (siehe Abbildung 3).

Zur direkten Integration von CSS-Code ist der Bereich „HTML-Header“ einer Seite wichtig. Diesen finden wir entweder direkt auf der Seite (unter Seitenwiedergabe –> Seite –> HTML Header) oder über das Seitentemplate. Wir gehen an dieser Stelle direkt in das Seitentemplate. Der Unterschied ist einfach: Im Template bekommt jede Seite, die das veränderte Template verwendet, automatisch den CSS-Code mitgeliefert (Vererbung). Verwenden wir stattdessen den HTML-Header der aktuellen Seite, wird der dort eingefügte Code lediglich per Ersetzungszeichenfolge #HEAD# in das Template temporär für genau diese eine Seite integriert. Auf einer anderen Seite, die nach Erstellung zunächst keinen Code im HTML-Header der Seite hat, wären unsere Style-Formate nicht vorhanden. Wir gehen mit dem Gedanken an Globalisierung der Änderungen direkt in das Seitentemplate.

Im Entwurf des Templates (zum Beispiel One Level Tabs) steht unter dem

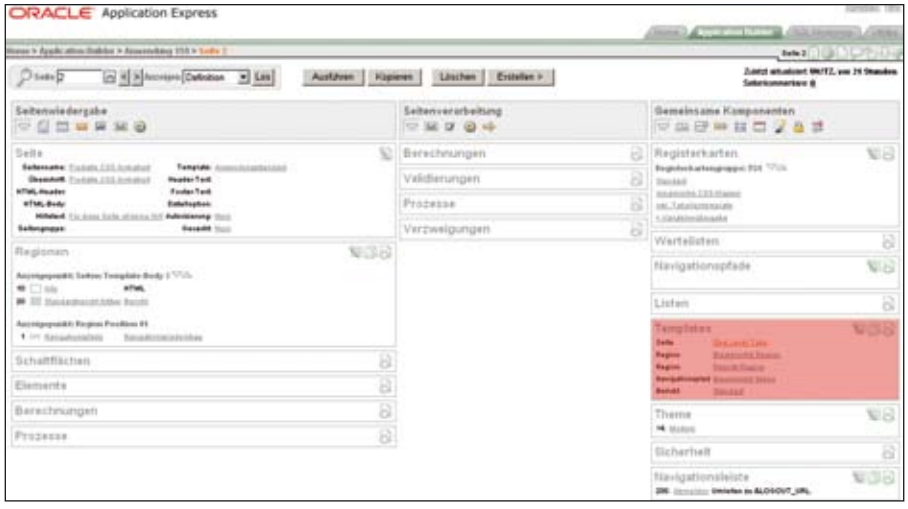



*Abbildung 3: Die Seitenentwurfs-Ansicht unter APEX*


*Abbildung 4: Das Seitentemplate bildet den optischen Rahmen der Anwendung*

Punkt „Definition“ bereits der HTML-Code des Headers. Dieser enthält, wie für einen normalen HTML-Head üblich, den öffnenden HTML-Tag <head> sowie den abschließenden HTML-Tag </head>. Zwischen diesen Tags wird der CSS-Code eingefügt (siehe Listing 1). Im Code fällt auf, dass gleich ein Table-Tag, den wir später benötigen, mit der Klasse „meineTabelle“ formatiert wurde.

```
<style type="text/css">
<!--
div.Computer, div.Handy, div.
Haushalt
{
   position: relative;
   width: 100%;
   height: 100%;
   margin: 0;
   padding: 3px;
   padding-left: 5px;
   padding-right: 5px;
   height: 100%;
   background-color:#d7ec98;
   color: #3e6900;
}
div.Handy
{
   color: rgb(0,42,78);
   background-color:
rgb(120,170,210);
}
div.Haushalt
{
   color: rgb(81,62,9);
   background-color:
rgb(250,220,130);
}

table.meineTabelle
{
   margin: 0px;
   padding: 0px;
   border-collapse: collapse;
   border: 1px solid #757575;
   font-size: 13px;
}
table.meineTabelle th
{
```

```
    border: 1px solid #757575;
    background-color: #333333;
    color: #FFFFFF;
    padding: 5px;
    letter-spacing: 1px;
}
table.meineTabelle td
{
    border: 1px solid #757575;
    margin: 0px;
    padding: 0px;
}
-->
</style>
```

*Listing 1*

Nachdem die Änderungen des Seitentemplates gespeichert sind, können wir die angelegten Stile bereits verwenden. Wie Sie sehen können, haben wir diese nun mit den Klassen angelegt. Die Klassen „Computer“, „Handy“ und „Haushalt“ in den  Tags entsprechen exakt der Bezeichnung unserer Kategorien.

Wie schon erwähnt, ist es in Apex möglich, nahezu überall HTML-Code zu platzieren. Wir werden jetzt damit unsere Daten einpacken.

Apex arbeitet zumeist mit Ersetzungszeichenfolgen. Die Spaltennamen sind dies beispielsweise innerhalb der Region eines SQL-Berichts. Wir verwenden folgenden Bericht:

```
SELECT ID, Kategorie, Produkt,
Preis
FROM Demo_Artikel
ORDER BY Kategorie, Produkt
```

Damit haben wir durch das SQL-Statement in der Berichtsregion die Ersetzungszeichenfolgen #ID#, #KATEGORIE#, #PRODUKT# und #PREIS# verfügbar gemacht. Im Entwurf des Berichts (zu finden im Seitenentwurf) werden wir diese Zeichenfolgen verwenden.


*Abbildung 5: Hier steht der Berichtsentwurf*

Abbildung 6 zeigt den Berichtsentwurf, welcher eine Auflistung aller


*Abbildung 6: Der geöffnete Berichtsentwurf*

vom SQL-Statement zurückgegebenen Spalten enthält. Diese Spalten beinhalten über das in der Abbildung 6 hervorgehobene rote Symbol weitere Anpassungsmöglichkeiten. Diese werden wir, beginnend mit der Kategorie und abschließend mit dem Preis, der Reihe nach editieren.

In der Entwurfsansicht der jeweiligen Spalte findet man im zweiten Abschnitt (Spaltenformatierung) die Möglichkeit, einen „HTML-Ausdruck“ zu editieren. Dieser Bereich ist zu Beginn leer, da Apex als Standard lediglich den Spalteninhalt übergibt. In dieses leere Feld fügen wir nun den folgenden HTML-Code ein:

```
<div class="#KATEGORIE#">#KATEG
ORIE#</div>
```

In Worten formuliert, haben wir hier einen leeren Container, der eine Klasse verwendet, die dem Inhalt der Spalte „Kategorie“ entspricht. Der Container enthält als sichtbaren Inhalt in diesem Fall auch die „Kategorie“. In der Spalte „Produkt“ würde das Ganze wie folgt aussehen:

```
<div
class="#KATEGORIE#">#PRODUKT#</
div>
```

Die Klasse ist noch immer gleich dem Inhalt der Spalte „Kategorie“ – aber der Inhalt des HTML-Tags entspricht dem Inhalt der Spalte „Produkt“. Die Klasse setzen wir für alle Spalten immer auf „Kategorie“. Nur der Inhalt entspricht dem eigentlichen Inhalt der Spalte, welche wir gerade bearbeiten. Betrachten wir doch einmal, was Apex beim Rendern der Seite aus dem Code macht. Als Beispiel nehmen wir das Produkt „Grafikkarte X“:

```
<div
class="Computer">Grafikkarte
X</div>
```

Apex füllt die Ersetzungszeichenfolgen mit den Daten auf. Nicht ganz zufällig gibt es seit kurzem die CSS-Klasse „Computer“ für ein -Element. Diese wird jetzt automatisch immer dann herangezogen, wenn der Wert der Spalte „Kategorie“ = „Computer“ ist. Abbildung 7 zeigt das Ergebnis.

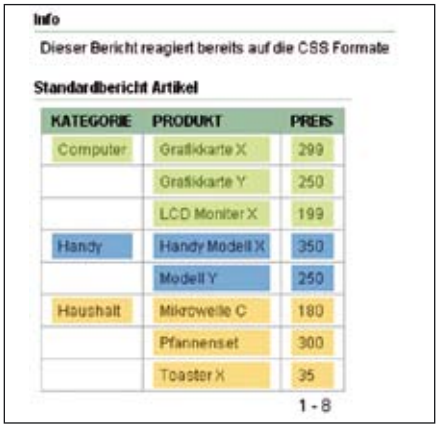



*Abbildung 7: Vorläufiges Ergebnis von Schritt 1*

### Schritt 2 – Apex-Darstellung ausbessern

Sobald man eines der vordefinierten Apex-Themes verwendet, bekommt man auch dessen gesamte Formatstruktur zur Verfügung gestellt. Wie das Ergebnis von Schritt 1 zeigt, kann das manchmal störend sein; in diesem Fall stört der Innenabstand der Tabellenzellen. Schritt 2 zeigt, dass das Thema „Layout“ nicht immer undurchsichtig bleiben muss.


*Abbildung 8: Die rot markierten Abstände sollten entfernt werden*

Zunächst erzeugen wir uns ein neues Template für einen Bericht oder ändern einfach das verwendete aus unserem Standard-Theme. In diesem Fall zerstören wir das Theme dadurch nicht – in einer anderen Anwendung ist alles wieder dem Standard entsprechend. Im Seitenentwurf benötigen wir jetzt unter „Gemeinsame Komponenten" den Bereich „Templates". Dort aufgelistet ist das für unseren Bericht verwendete Template „Standard". Ein Klick darauf öffnet den Template-Editor. Dort ändern wir den Code folgender Punkte:

Vor Zeilen:

```
<table #REPORT_ATTRIBUTES#
id="report_#REGION_STATIC_
ID#">#TOP_PAGINATION#
<tr><td><table
class="meineTabelle">
```

Spaltenüberschriften > Spaltenüberschrifts-Template:

```
<th #ALIGNMENT# id="#COLUMN_
HEADER_NAME#">#COLUMN_HEADER#</
th>
```

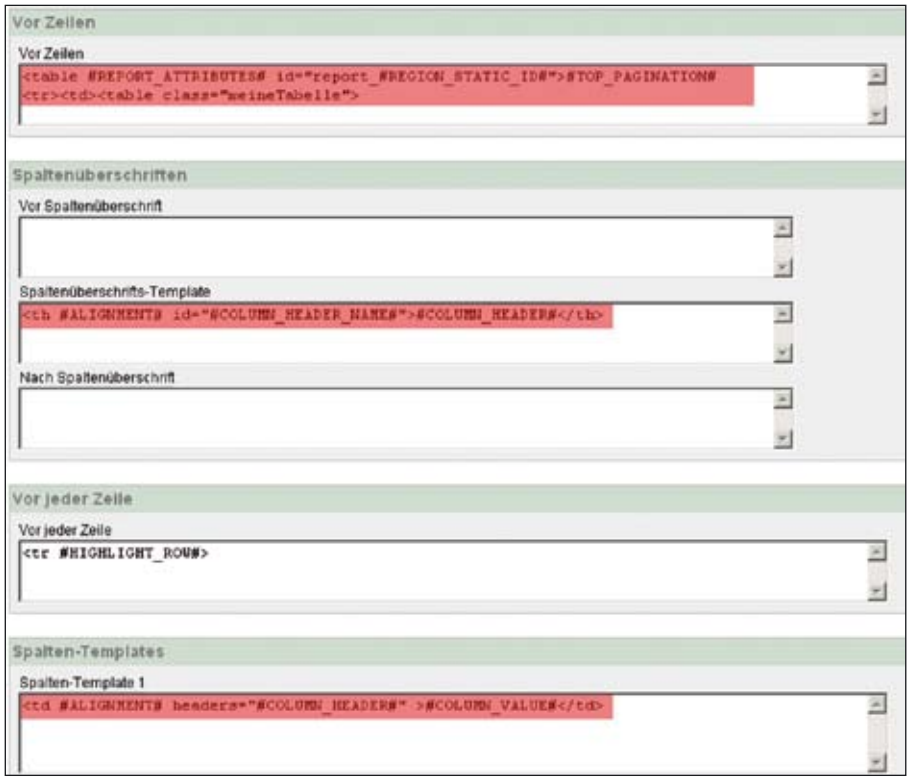



*Abbildung 9: Das Template wird geändert*

Spalten-Templates > Spalten-Template 1:

```
<td #ALIGNMENT#
headers="#COLUMN_HEADER#" >#CO-
LUMN_VALUE#</td>
```

Nachdem die Änderungen gespeichert sind, können wir uns das Ergebnis bereits ansehen. Die zu Anfang bereits mit in unseren CSS-Code integrierte „meineTabelle"-Klasse wird nun angesprochen.


*Abbildung 10: Das angepasste Template*

**Fazit**

Wie man anhand dieses einfachen Beispiels sehen kann, bietet Apex viele nicht offensichtliche Möglichkeiten. Es gilt, diese herauszufinden und zu publizieren. Als nächsten Schritt zu einer moderneren Webanwendung sollte man das Thema „Javascript" angehen.

**Kontakt:**
Thomas Zielbauer
altasheth@gmx.de
thomas.zielbauer@man.eu

**Oracle Newsticker**

**Oracle erweitert Plattform für Governance, Risk and Compliance (GRC)**

Mit der Erweiterung seiner Governance, Risk and Compliance Application Suite bietet Oracle eine vollständige, integrierte End-to-end-Lösung für GRC-Anforderungen. Neu vorgestellt wurden Oracle Enterprise Governance, Risk and Compliance Manager (Oracle Enterprise GRC Manager) und die neue Version von Oracle Enterprise Governance, Risk and Compliance Controls (Oracle Enterprise GRC Controls). Diese Erweiterungen der GRC-Plattform ermöglichen einen geschlossenen Ansatz um die Einhaltung von Gesetzen und Richtlinien sowie Risikomanagement und Kontrollautomation zu gewährleisten. Beide Produkte sind eng miteinander verzahnt, so dass Unternehmen Risiken erkennen, bewerten sowie priorisieren können und damit das bestmögliche Risikomanagement durch eine Kombination aus manuellen und automatischen Kontrollen bestimmen. Dieser integrierte Ansatz gibt Organisationen und Unternehmen ein Verständnis davon, welche Risiken kritisch sind.

# Installation der Oracle Fusion Middleware 11g R1 – Portal, Forms, Reports und Discoverer

Bernd Vierschilling, OPITZ CONSULTING Gummersbach GmbH

**Mit Fusion Middleware 11*g* R1 hat Oracle ein umfangreiches Softwarepaket auf den Markt gebracht, aus dem sich einzelne Komponenten modular installieren lassen. Dieser Artikel beschreibt die Erfahrung mit der Installation von Fusion Middleware Portal, Forms, Reports und Discoverer entlang der Roadmap und verweist auf Besonderheiten, die während des Prozederes auftreten können.**

Dieser Beitrag sollte nicht als Installationsanleitung verstanden werden. Er bietet jedoch nützliche Hinweise für die optimale Vorbereitung und damit auch reibungslose Durchführung der Installation.

### Die Softwareauswahl

Die Zusammenstellung der Software aus dem OTN stellte schon eine gewisse Herausforderung dar, weil das von Oracle geschnürte Paket mit Portal, Forms, Reports und Discoverer nicht alle notwendigen Module enthält. Auf diese zusätzlich notwendigen Komponenten sollte eigentlich „Additional Software Requirements“ verweisen – was aber nicht vollständig gelingt. Von einigen wenigen Modulen erfährt man daher leider erst aus der Installationsanleitung der einzelnen Produkte. Des Weiteren zeigte sich, dass die Komponente SSO noch nicht auf 11*g* gehoben wurde. Hier musste mit etwas Aufwand noch eine 10er Version integriert werden.

### Voraussetzungen

Für unsere Beispiel-Installation standen zwei Linux-RH5.3-Server zur Verfügung. Der eine sollte die Middleware hosten, der zweite die notwendige Repository-Datenbank in der Version 11.1.0.7 beherbergen. Bei dieser Version war darauf zu achten, dass sie mit dem Characterset AL32UTF8 und der Knowledge Base der Examples-CD (vormals Companion) installiert wurde. Diese Knowledge Base wird im Zusammenhang mit der Portal-Installation verlangt. Die gesamte Installation erfolgte unter Nutzung des Typs „typical“ und mit den vom Installer vorgeschlagenen Benutzern und Ports.

### RCU

Das RCU bereitet die Datenbank auf die Aufnahme der Repositories vor. Es zeigt sich quasi mandantenfähig, da während der Installation ein sogenanntes „Präfix“ verlangt wird. Dieses bildet dann zusammen mit dem Modul den Schema-Namen. Es können also beliebig viele Repository-Besitzer mit wechselnden Präfixen generiert werden. Jedoch sollte man unbedingt darauf achten, zu Beginn bereits alle Komponenten zu installieren, die in Zukunft benötigt werden. Vergisst man nämlich zum Beispiel den Discoverer, so kann der unter dem schon installierten Präfix nicht mehr nachinstalliert werden. In dem Fall erscheint dann die Meldung, dass dieses bereits existiere. Hier besteht auf Seiten des Herstellers noch Nachbesserungsbedarf. Sind die gewünschten Komponenten ausgewählt, erfolgt die Installation auf der Datenbank. Installiert werden sowohl die Schemata als auch die zugehörigen Tablespaces.

### WebLogic Server

Die Installation des Weblogic Servers verläuft genau so, wie man sich das vorstellt. Nach dem Start des Installers ist das Home-Verzeichnis zu hinterlegen, und es erfolgt die Installation der Software.

### Oracle Identity Management

Mit der Installation dieser Komponente wird unter anderem das OID installiert, in das nachfolgend dann das SSO implementiert werden muss. Bei diesem Vorgang folgte der Autor den Anweisungen des Installation Guides. Mithilfe der vorgegebenen Pfade, Benutzer und Ports verlief auch diese Installation ohne weitere Probleme.

### SSO-Installation

Hier war zum ersten Mal der Punkt erreicht, an dem der Autor während des Prozesses eine Verbindung zu „My Oracle Support“ herstellen musste. Zum einen liefen nicht alle Skripte so, wie in der Dokumentation beschrieben, und zum anderen wurden Patches benötigt, um auf einen Stand zu kommen, auf dem die zuvor installierten Versionen funktionieren konnten. Oracle bedient sich hierzu eines kleinen Perl-Skripts, das vor, während und nach der SSO–Installation gestartet werden sollte: „inspre11.pl“.

Es wird vor der SSO-Installation mit „-op1“ auf der Console des Middle-Tiers wie folgt aufgerufen:

```
prompt:$ORACLE_HOME/perl/bin/
perl \
> $ORACLE_HOME/ldap/bin/insp-
re11.pl vsllsetg049.opitz-con-
sulting.int 3131 -ssl \
```

```
> oid1 10.1.203.52:1521:DB1
<pwdoid> <pwddb> -op1
prompt:Use RepCA to load SSO
and other schemas against DB
before running -op2
```

Das OID der Version 11 war jetzt für die Aufnahme des SSO mit der Version 10 vorbereitet. Die Ausgabe teilte daraufhin mit, dass nun der RepCA (und zwar die Version 10.1.4.3.0) gestartet werden müsste. Der RepCA installierte das Repository für das SSO und wir registrierten es am OID. Dieser Teil der Installation endete mit dem Restart des OID und dem abermaligen Aufruf des Perl-Skripts mit „-op2“:

```
prompt:$ORACLE_HOME/perl/bin/
perl \
> $ORACLE_HOME/ldap/bin/in-
spre11.pl vsllsetg049.opitz-
consulting.int 3131 -ssl \
> oid1 DB1.world <pwdoid>
<pwddb> -op2
```

An der Stelle sei darauf hingewiesen, dass das Skript beim zweiten Aufruf einen Net-Alias für die Datenbank erwartet. Es wird jedoch zur Auflösung des Alias‘ nicht etwa die TNS_ADMIN-Variable genutzt, sondern im $ORACLE_INSTANCE/config wird nach der Konfigurationsdatei tnsnames.ora gesucht.

Nun folgte die eigentliche Installation des SSO, bei der wir uns eng an die Dokumentation hielten. Aus der Dokumentation erfuhren wir im weiteren Verlauf von einem Patch 5649850, den es herunterzuladen und – vor Ausführung des root.sh – zu installieren galt, was bei Oracle-Installationen eigentlich nichts Ungewöhnliches ist. Doch in diesem Fall meldete das Programm opatch, dass es sich hier um ein unbekanntes Betriebssystem handele. Erst nach einem kleinen Eingriff in die Konfigurationsdatei des Patches wurde opatch erfolgreich ausgeführt. Mit der anschließenden Installation des Patches 7215628 bekamen wir die SSO-Version auf die finale 10er Version 10.1.4.3.

Zu guter Letzt war jetzt noch einmal das Perlscript zu starten, diesmal mit dem Parameter „-op3“. Leider bekamen wir auch hier noch einen Fehler, der aber mittlerweile unter der Metalink-Note 943666.1 dokumentiert ist: Es wird auf das Skript oidiptch.sql verwiesen, das sich im Verzeichnis $ORACLE_HOME/ldap/admin befindet. Wenn man sich als ODS-User an der Datenbank anmeldet, wird das Skript ausgeführt. Es korrigiert Unschärfen, die durch die SSO-Installation entstehen, und sorgt dafür, dass das Perl-Skript mit dem Parameter „-op3“ nun fehlerlos läuft.

Nach einem Restart aller bisher eingebrachten Komponenten war die größte Hürde genommen und das System stand jetzt bereit für den letzten Schritt der Installation.

### Oracle Portal, Forms, Reports und Discoverer 11*g*

Diese Komponenten ließen sich jetzt alle problemlos installieren. Im weiteren Verlauf wurden die vorgenommenen Konfigurationen abgefragt. Auch Software und Repositories wurden im Anschluss schnell und erfolgreich installiert.

## Fazit

Wünschenswert wäre die Ergänzung der Roadmap mit den benötigten Patches. Sind aber erst einmal alle notwendigen Software-Komponenten zusammengetragen, verläuft die Installation, von dem einen oder anderen Schönheitsfehler abgesehen, recht reibungslos.

**Kontakt:**
Bernd Vierschilling
bernd.vierschilling@
opitz-consulting.com

# JasperReports aus der Datenbank: Konsolidierung mit Open Source

Wolfgang Stähle, Trivadis GmbH

**Es war ein Konsolidierungsprojekt: Bei einem Unternehmen im Automotive-Umfeld sollte ein System mit dedizierter Hardware und eigener Datenbank (SQL-Server), eigenem Java-Backbone-Prozess für das Datenmanagement, eigener Reports-Engine und einer WEB-GUI in eine bestehende Oracle-BEA-Landschaft überführt werden. Vorgaben waren: möglichst geringe Kosten, insbesondere keine zusätzlichen Lizenzkosten, hohe Performance-Anforderungen und die Rahmenbedingung, kein Oracle Reports zu verwenden. Die zu erstellenden Reports sollten sowohl interaktiv als auch durch die Datenbank im Batch ausgeführt werden.**

Anforderungen dieser Art sind ja nicht neu. Die Gesamtkosten für den Betrieb solcher Systeme mit Lizenzen, Wartung und Anwendungssupport sind beachtlich. Die Tendenz, solche Systeme zu konsolidieren, ist allgegenwärtig. Im Projekt sollte der ROI nach einem Jahr erreicht sein. Die bestehende Oracle-Instanz, die einen Teil der Basisdaten für das Altsystem hält, soll zentrale und vollständige Datenhaltung sein. Diese Datenbank musste also um weitere Schnittstellen ergänzt werden. Als Application Server sollte ein bestehender BEA-Weblogic-Server zum Einsatz kommen. Die Anwendung generiert zur Dokumentation von Produktionsschritten PDF-Dokumente, die – um gesetzlichen Anforderungen gerecht zu werden – im Nachgang zur Verfilmung auf Microfiche weitergegeben werden. Die Reports mussten also gegenüber dem Altsystem „pixelgleich“ sein.

## Anforderungen an Reports

Die Anzahl zu generierender Reports ist zwar relativ gering (pro Tag bis zu 500 im Batch und bis zu 100 interaktiv), diese müssen aber interaktiv gestartet nach einer Sekunde am Bildschirm erscheinen. Ein Report selbst hat folgende Eigenschaften: Er besteht aus bis zu acht unterschiedlichen Bereichen, wobei sieben davon durch verschiedene Queries aus der Datenbank erstellt werden und ein fertiges PDF-Dokument eingebettet wurde. Die Reports sind zum Teil mehrspaltig, enthalten Gruppierungen und Barcodes, das Logo des Kunden als Bild und einen dynamischen Titel. Im Durchschnitt enthält ein Report dreizehn Seiten. Für die Steuerung der Inhalte, ob der Kopf angezeigt werden soll und einige direkt angezeigte Textbausteine sind Parameter eingebaut. Alle Ausgaben sind in das Format PDF zu rendern.

## Entscheidung für JasperReports

Insbesondere der Funktionsumfang und die positiven Erfahrungen aus anderen Projekten führten zur Entscheidung, das Open-Source-Produkt „JasperReports“ als Rendering Engine einzusetzen. Die komfortable Möglichkeit, Reports mit iReport grafisch zu erstellen, sowie die Server-Variante mit ihren OLAP-Funktionalitäten könnten zudem die im Projektumfeld eingesetzten Tools „Oracle Reports“ und „Oracle Discoverer“ beerben.

Im Vergleich zu PL/PDF und Apache FOP mit Stylesheets erschien der Aufwand zur Erstellung der Reports deutlich geringer, die Alternative BI Publisher lizenztechnisch zu wenig attraktiv.

Die Oberfläche wurde als Rich-Client Java Anwendung realisiert: Dies ließ erweiterte Möglichkeiten gegenüber der alten JSP-Anwendung zu.

## Aufrufe

Die Reports sollten folgendermaßen gestartet werden können:

- Über die Anwendung
- Aus der Datenbank über Jobs generiert und wieder in die Datenbank zurückgespeichert
- In einer Drittanwendung generiert und visualisiert

Ein Aufruf aus der Anwendung führt dazu, dass der Report ausgeführt wird, ein PDF erzeugt, dieses in der Datenbank speichert und dem Anwender anzeigt. Aus der Datenbank heraus wird eine URL erstellt, und hiermit die Java-Komponente auf dem BEA-Applications-Umfeld-Server gestartet, die dann das PDF erzeugt und ebenfalls zurückspeichert. In die Drittanwendung ist ein ActiveX-Browser-Control eingebettet worden, das dann den URL-Aufruf via http durchführt und das erstellte PDF anzeigen kann.

## Realisierung

Generell wird darauf hingewiesen, dass sich die Anschaffung und Lektüre der „Ultimate Guides“, also der Dokumentation von JasperReports und iReport lohnt, die bei Jaspersoft für jeweils 50 Dollar zu haben sind. Darin sind einige Beschreibungen und Hinweise zu finden, die bei der Definition und der Verhaltensweise von bestimmten Eigenschaften sehr hilfreich sind.

Der Hauptreport enthält sechs Subreports, die jeweils eigene Queries ausführen. Die Steuerung der Inhalte, ob ein Subreport im Gesamtreport erscheint, wird über Parameter gesteuert, die in den „print when expressions“ evaluiert werden (siehe Listing 1).

```
new Boolean($P{INHALT}.
equals(„KOMPLETT“) ||
            $P{INHALT}.
equals(„PM_SPERRMELDUNGEN“)||
            $P{INHALT}.
equals(„PM_BEREICHSFREIGABEN“)||
            $P{INHALT}.
equals(„PM_REP_CHECK_BZP“)||
```

```
            $P{INHALT}.
equals(„PM_MUSTEREINBAUTEN“)||
            $P{INHALT}.
equals(„PM_WERKERANWEISUN-
GEN“)||
            $P{INHALT}.
equals(„PM_AKTIONSPUNKTE“))
```

*Listing 1: „Print when Expression“ des Subreports*

In einem Subreport wurde aus Komplexitäts- und Performance-Gründen eine „pipelined table function“ benutzt. Listing 2 zeigt die Übergabe von zwei Parametern in der Query.

```
SELECT
...
FROM
  TABLE(GET_EWBK_PDF_
SIRELNA($P{pProdNr})) P
WHERE
  p.ERFASST <= $P{pZeitstempel}
AND
...
```

*Listing 2: Parameterübergabe an „pipelined table function“*

Da auf dem Application Server keine Fonts und auch kein Fontserver installiert waren, musste der entsprechende Font mit ausgeliefert werden. Dazu wurde das TTF-File in die Anwendung aufgenommen. Damit es geladen und im Report benutzt werden kann, sind folgende Properties zu definieren (siehe Listing 3).

```
# PDF exporter font settings:
net.sf.jasperreports.
export.pdf.font.
ArialTrueTypeFont=Arial.ttf
```

*Listing 3:default.jasperreports.properties*

Der Reportaufruf wurde in der Java-Komponente gekapselt und auf den BEA Application Server deployed. Aus der Datenbank erfolgte der Generierungsprozess mithilfe des UTL_HTTP–Packages.

### Tücken

Bei der Realisierung der Reports mit dem iReport-Tool zeigten sich einige Stärken und Schwächen. Während sich das Authoring für mit Reporting-Tools-affinen Produkten intuitiv bewerkstelligen lässt, hat das Tool (in der eingesetzten Version 3.5.2) einige bemerkenswerte Eigenschaften. Beispielsweise wurden im gespeicherten Report-XML (JRXML) beim Verlassen des Tools oder beim Schließen des Reports einige Eigenschaften sporadisch zurückgesetzt (Ausrichtungen, visuelle Attribute etc.). Dies änderte sich in der später aktualisierten Version 3.6.0 etwas zum Besseren, führte allerdings nach einigen schmerzlichen Erfahrungen dazu, dass nach der Weiterentwicklung jeweils die neue Version mit der alten verglichen werden musste, um ungewollte Änderungen zu identifizieren. Generell sieht man sich immer wieder einigen „Meldungen“ ausgesetzt, wie Abbildung 1 zeigt.


*Abbildung 1: Warnhinweis; in Version 3.6.0 ohne Auswirkung*

### Zusatzanforderung: Verschicken des PDF per sFTP

Während der Realisierung des Projekts kam eine zusätzliche Anforderung hinzu, nämlich das generierte PDF-Dokument per sFTP auf einen Server in der DMZ zu kopieren – scheinbar kein großes Problem, zumal es in der Datenbank schon zwei Implementationen des FTP-Protokolls gab, eine über geladene Java-Klassen, die andere über die Verwendung von UTL_TCP. Allerdings war der Ansatz, sFTP aus der Datenbank heraus zu betreiben, nicht stabil zu realisieren: Das Ergebnis waren regelmäßige ORA-600-Fehlermeldungen (auf einer V10.2.0.2), obwohl in der Test-Instanz mit derselben Version der Datenbank und des Betriebssystems keine Komplikationen auftraten. Die Entscheidung, diese Anforderung über die Applications-Umfeld-Server-

Anwendung zu realisieren, musste getroffen werden.

### Super-GAU: Ein Wort fehlt

Am Tag der Abnahme kam dann von den Testern die Hiobsbotschaft: Ein Text würde abgeschnitten. Es fehlte genau das letzte ganze Wort in mehreren Textfeldern! Die Analyse mithilfe des Jaspersoft-Supports ergab, dass unterschiedliche physische Fonts verwendet wurden, obwohl sowohl in der Textfeld-Eigenschaft, als auch für den PDF-Font derselbe Name referenziert wurde. Abhilfe schaffte eine Font-Extension. Ein jasperreports-fonts-3.6.0.jar musste in der Applikation ausgeliefert werden (siehe Listing 4). Darin stehen die notwendigen TTF-Files, die im fonts.xml dem jeweiligen Font-Namen zugeordnet sind (siehe Listing 5).

```
net.sf.jasperreports.extensi-
on.registry.factory.fonts=net.
sf.jasperreports.extensions.
SpringExtensionsRegistryFactory
net.sf.jasperreports.ex-
tension.fonts.spring.beans.
resource=net/sf/jasperreports/
fonts/fonts.xml
```

*Listing 4: jasperreports_extensions.properties*

```
<beans xmlns="http://www.
springframework.org/schema/be-
ans"
      xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.
springframework.org/schema/be-
ans
      http://www.springframe-
work.org/schema/beans/spring-
beans-2.0.xsd">

      <bean id="Arial"
class="net.sf.jasperreports.
engine.fonts.SimpleFontFamily">
            <property
name="name" value="Arial"/>
            <property
name="normal" value="net/sf/
jasperreports/fonts/arial/
arial.ttf"/>
            <property
name="bold" value="net/sf/jas-
perreports/fonts/arial/arialbd.
ttf"/>
            <property
name="italic" value="net/sf/
jasperreports/fonts/arial/aria-
li.ttf"/>
            <property
name="boldItalic" value="net/
sf/jasperreports/fonts/arial/
arialbi.ttf"/>
            <proper-
ty name="pdfEncoding"
value="Identity-H"/>
            <proper-
ty name="pdfEmbedded"
value="true"/>
      </bean>
</beans>
```

*Listing 5: Fonts.xml*

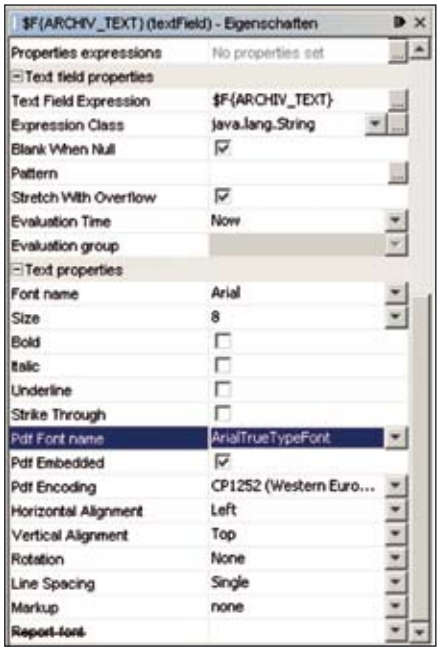



*Abbildung 2:Text-Properies*

Notwendig wurde dieses wenig durchsichtige Vorgehen deshalb, weil laut Jaspersoft der im „Font Name“ des Textfelds definierte Font zur Ermittlung der Metrics verwendet wird, zur Darstellung im PDF-Format aber der Font aus „PDF Font Name“. Nachdem sichergestellt war, dass dasselbe physische Fontfile verwendet wird, wurden auch alle Texte komplett angezeigt.

### Fazit

Die Implementierung von JasperReports als Java-Webkomponente erwies sich als äußerst performante Lösung. Das Authoring gelingt auch bei komplexen Anforderungen zielgerichtet und schnell. Leider gibt es einige wenig erbauliche Schwächen im iReport und die Transparenz bezüglich der Font-Thematik ist verbesserungswürdig. Eventuell werden viele Jasper-Anwender dieses Verhalten erst noch bemerken. Dass bei Jasper vieles im Fluss ist und die Probleme erkannt wurden, spiegelt sich im iReport-Versionszyklus wieder: 3.6.1 wurde am 28. Oktober 2009 und 3.6.2.am 24. November 2009 released. Es wurde ein neues „TrueType font Management“ eingebaut sowie jeweils ein Wizard für den Import von TrueType-Fonts und einer für die Erstellung von JasperReports-Font-Extensions. Trotz allem überzeugt JasperReports und präsentiert sich als vollwertige Alternative zu den bestehenden professionellen Reporting-Lösungen, was insbesondere in Hinsicht auf den Funktionsumfang gilt. Support kann bei Jaspersoft entweder für einen bestimmte Anzahl Incidents (Open Source Incident Packs) oder als „Professional Support“ mit unterschiedlichen Reaktionszeiten bezogen werden.

**Kontakt:**
Wolfgang Stähle
wolfgang.staehle@trivadis.com

# Hochwertigen PL/SQL-Code entwickeln

Thomas Klughardt, Quest Software GmbH

**Auf Dauer macht die Pflege des Codes ein Vielfaches der Kosten der Entwicklung aus, jede unsaubere Implementierung rächt sich. Deshalb sind die Anforderungen an die Entwickler hoch: Der von ihnen programmierte Code sollte wartbar, performant sowie robust und damit gut erweiterbar sein. Es mag verlockend sein, bestimmte Funktionalitäten „quick and dirty" zu implementieren – doch auch dieser Code gehört zur Anwendung und muss gepflegt werden. Wie die Y2K-Krise gezeigt hat, ist Code oft viel länger in Betrieb, als man je angenommen hätte.**

Am Anfang jedes Projekts werden klare Regeln definiert, wie der zu entwickelnde Code auszusehen hat, wie vorgegangen werden soll und welche Techniken zum Einsatz kommen. Mithilfe einer Top-Down-Ansicht legt man fest, wie die Anwendung letztendlich aussehen soll. Ziel ist es, Namenskonventionen einzuhalten, Best Practices umzusetzen und, obwohl die Anforderungen an die Anwendung scheinbar feststehen, das Design möglichst flexibel zu halten, um auf Änderungswünsche reagieren zu können. Denn erfahrungsgemäß ändern sich die Vorgaben noch mehrfach während des Projekts. Die Anwendung sollte also so geschrieben sein, dass sie an alle Eventualitäten angepasst werden kann, klar strukturiert und leicht zu warten ist.

## Theorie und Praxis

Soweit die Theorie, doch es gibt zwei störende Faktoren in der Programmier-Idylle: den Anwender, der auch Auftraggeber ist, sowie unvorhergesehene Ereignisse. Änderungswünsche der Anwender überfordern oft jedes noch so flexible Design. Unvorhersehbare und damit nicht geplante Verzögerungen verbrauchen die sorgsam gesetzten Puffer im Zeitplan. Plötzlich ist die Zeit zu knapp, um sich an die eigenen Konventionen zu halten oder ausgiebig zu testen. Manche Funktionalitäten müssen nun nicht-konform implementiert werden, weil es zu aufwändig wäre, das gesamte Design zu ändern. Die Funktionalität wurde dann zwar auf den ersten Blick umgesetzt und der Anwender ist vorerst zufrieden. Jedoch entstehen so Schwachstellen im Code der Anwendung, die in Zukunft immer besonders beachtet werden müssen. Gerade für neue Entwickler, die in das Team integriert werden sollen, ist es schwierig, alle diese Besonderheiten und Widersprüche zu den Konventionen zu kennen.

In der agilen Software-Entwicklung gibt es Vorgehensmodelle wie „Scrum", die versuchen, solche Störfaktoren aufzufangen. Im Endeffekt ist dies aber nur ein formalisierter Weg, möglichst frühzeitig Kompromisse zu schließen, in denen die Anwender auf einen Teil der Funktionalität verzichten und die Entwickler deshalb den Zeitplan einhalten können.

Eine andere vielversprechende Lösung ist es, möglichst viele der Schritte, die der Entwickler machen muss, zu automatisieren. Ob eine Funktionalität umgesetzt wurde, kann vom Anwender leicht festgestellt werden. Wie sie umgesetzt wurde, ist für den Nutzer aber zunächst einmal nicht ersichtlich. Langfristig kann hier zwar die Fehlerhäufigkeit Aufschluss geben, aber bei der Lieferung kann der Nutzer lediglich feststellen, ob die Anwendung funktioniert oder nicht. Gerade deshalb werden Code Reviews, also das Überprüfen der Einhaltung der Best Practices und Konventionen, und das Testen nur rudimentär ausgeführt oder fallen sogar ganz weg.

## Automatisiertes Testen

Die Vorteile des automatisierten Testens liegen auf der Hand. Beim manuellen Testen führt man das Feature von Hand aus und validiert das Ergebnis. So sieht der Entwickler, ob der Testfall in diesem Moment erfolgreich war. Danach aber geht die gerade gemachte Arbeit verloren: Will man den gleichen Test noch einmal laufen lassen, nachdem beispielsweise Änderungen am Code vorgenommen wurden, muss man erneut aufwändig manuell vorgehen. Beim automatisierten Testen bleibt der Fortschritt erhalten. Sind die Testfälle einmal definiert, kann man die Tests wieder und wieder nutzen. Dadurch ist es möglich, bei jeder Änderung alle Testfälle durchzuführen und so Regressionstests zu machen. Regressionen im Code lassen sich so vermeiden, das bedeutet, dass alle Funktionen, die in Version 1 eingebaut wurden, auch in Version 2 noch immer fehlerfrei funktionieren. Im Gegensatz dazu ist es bei der manuellen Testmethode nicht möglich, jeweils alle Testfälle durchzuspielen. Die Folge: Nebenwirkungen von Code-Modifikationen bleiben oft unentdeckt. Robust ist ein Code jedoch nur dann, wenn Änderungen durchgeführt werden können, ohne durch Seiteneffekte andere Programmteile und Funktionalitäten zu stören.

Automatisiertes Testen ermöglicht eine weitere extreme Technik, das „testgetriebene Entwickeln" (Test Driven Development). Hier werden als Teil der Spezifikation schon vor der Implementierung die Testfälle auf Basis der Anwendungsfälle definiert. So stehen die Testfälle dem Entwickler schon vor Programmierbeginn zur Verfügung und er sieht immer, welche Funktionalitäten noch fehlen. Zusätzlich kann damit eine weitere häufige Fehlerquelle vermieden werden: Mancher Entwickler neigt dazu, aus der eigenen Implementierung unterbewusst Schlüsse auf ein bestimmtes Vorgehen der Nutzer zu ziehen. Die Praxis zeigt

jedoch immer wieder – der Nutzer verhält sich anders, als der Entwickler annimmt.

### Mindestens das Zehnfache an Test-Code

Es gibt eine Vielzahl von Lösungen, mit denen Tests automatisiert werden können. Vor allem Java-Entwicklern dürfte das Framework „JUnit" ein Begriff sein. Im PL/SQL-Bereich findet das Framework „utPLSQL" breite Akzeptanz. Ursprünglich auch von Steven Feuerstein entwickelt, wird die Lösung nun auf „Sourceforge" gepflegt. Der „Code Tester for Oracle" von Quest Software hat gegenüber diesen Tools einen Vorteil: Die Testfälle und erwarteten Ergebnisse lassen sich deklarativ definieren. Das bedeutet, dass man die Tests beschreiben kann, ohne tatsächlich Code zu schreiben.

Code Tester enthält einen Code-Generator, der den Test-Code erzeugt. Das ist eine große Arbeitserleichterung, denn pro Zeile Anwendungscode rechnet man mit mindestens zehn, oft aber auch mit über 20 Zeilen Test-Code, die geschrieben werden müssen, um eine gute Abdeckung der Anwendungsfälle zu erreichen. Es ist oft schwierig, beim Auftraggeber die Mittel zu erhalten, 1000 Zeilen Code schreiben zu dürfen, um 50 Zeilen Anwendungscode zu testen. Bedenkt man jedoch, dass die Tests nach ihrer Erstellung immer und immer wieder ausgeführt werden können, relativiert sich der tatsächliche Aufwand.

Im Code Tester Frontend lassen sich die Testfälle, Vorbedingungen und Eingabeparameter als konkrete oder dynamisch ermittelte Werte oder Mengen, zu erwartende Ergebnisse und Nachbedingungen beschreiben. In den meisten Fällen sollte das ausreichen. Zusätzlich gibt es die so genannten „Customization Zones", in denen PL/SQL-Code geschrieben werden kann. Es kann alles getestet werden, was man mit PL/SQL-Code ausdrücken kann.

Der Test-Code selbst wird als PL/SQL-Code in der Datenbank hinterlegt. Über eine entsprechende API können die Testfälle direkt über PL/SQL-Prozeduraufrufe ausgeführt werden: Mithilfe des Schedulers laufen die Testfälle regelmäßig ab und die Ergebnisse werden in Berichten zusammengestellt. So erfährt der Entwickler, wie es um die Funktionalität der Anwendung steht, und ob durch die letzten Änderungen Fehler aufgetreten sind. Kommen Build-Tools wie zum Beispiel „Apache Ant" zum Einsatz, kann man die Testfälle auch als Teil des Build-Prozesses ausführen. In der Java-Welt sind sogenannte „Nightly Builds" durchaus üblich, bei denen die Anwendung in kurzen Zyklen immer wieder neu zusammengebaut wird, um die neuen Funktionalitäten entsprechend schnell verfügbar zu haben. Hier ist es möglich, die neue Version nur dann zusammenzubauen, wenn alle Testfälle erfolgreich waren.

Automatisiertes Testen bietet eine Menge Möglichkeiten, wenn es darum geht, korrekten Code zu schreiben. Ob man extreme Techniken wie „testgetriebenes Entwickeln" einsetzen will oder ob man für alle Funktionalitäten Testfälle definiert, hängt von vielen Faktoren ab. Je höher die Abdeckung durch automatisierte Testfälle aber ist, desto mehr Arbeit spart sich der Entwickler später beim Testen und, nicht weniger wichtig, beim Debugging. In der agilen Software-Entwicklung verschieben sich dadurch möglicherweise sogar die Prioritäten: Geht Code durch ein fehlerhaftes Backup verloren, ist das nicht so schlimm, Hauptsache die Testfälle bleiben erhalten.

### Code Reviews

Neben der semantischen Korrektheit, also dem richtigen Funktionieren des Codes, ist für die Lesbarkeit, Übersichtlichkeit, Performance und Erweiterbarkeit des Codes die Einhaltung der Programmier-Richtlinien wichtig. So gibt es zum einen die selbst auferlegten Namenskonventionen. Zum anderen existieren allgemeine Best Practices, die aussagen, wie PL/SQL-Code geschrieben sein und worauf besser verzichtet werden sollte. Allein die Definition solcher Konventionen genügt nicht, es bedarf auch der Prüfung, ob sie eingehalten werden.

Diesem Zweck dienen Code Reviews, bei denen sich idealerweise das gesamte Team, oder zumindest die Teamleiter, zusammensetzen und den geschriebenen Code analysieren. Code Reviews kosten Zeit, aber genau wie beim automatisierten Testen wer-

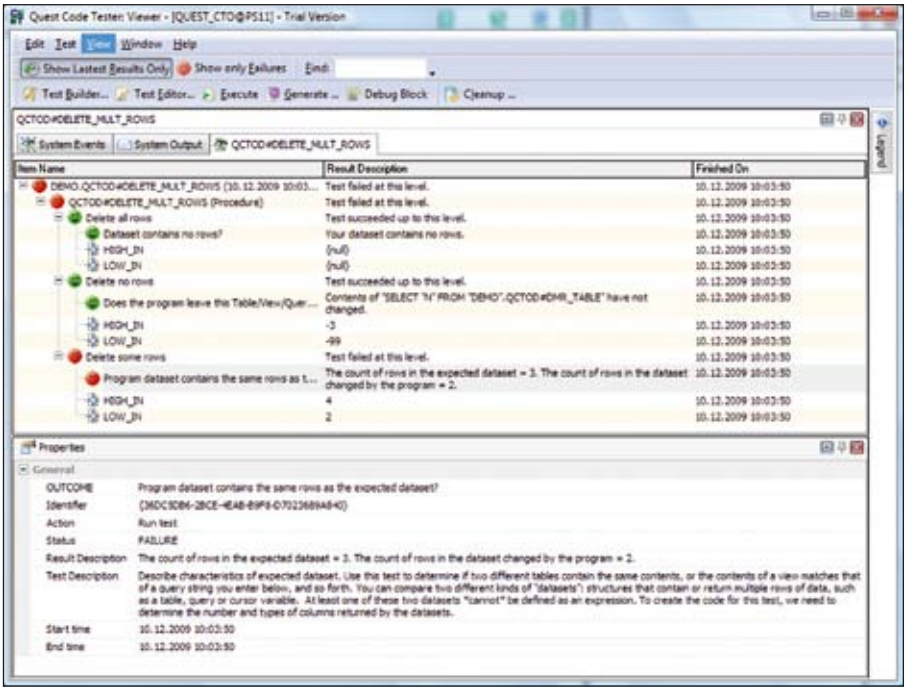



*Abbildung 1: Einblick in den Code Tester*

den dadurch viele Probleme vermieden, die zu einem späteren Zeitpunkt auftreten würden. Es lohnt sich, funktionierenden Code noch einmal zu kontrollieren. Regeln und Empfehlungen finden sich zahlreich in der Fachliteratur, wie in dem Buch „PL/SQL Development Best Practices“ von Steven Feuerstein. Zusätzlich gibt es viele Komplexitätsmetriken, die Auskunft darüber geben, wie verständlich und wartbar der Code ist.

Moderne IDEs unterstützen den Entwickler dabei, Best Practices und Konventionen einzuhalten. In „Toad“ zum Beispiel gibt es den Code „Xpert“, der statisch PL/SQL-Code analysiert und prüft, ob die PL/SQL Best Practices eingehalten werden. Zusätzlich werden Metriken, wie zum Beispiel der „Halstead Volume“ oder „Mc Cabe's Index“ ermittelt, die Aufschluss darüber geben, wie komplex beziehungsweise wie verständlich der Code für andere Entwickler ist. Hierüber lassen sich die Schwachstellen im Code aufspüren, die sonst langfristig Probleme und hohe Kosten verursachen würden.

So ist es mit diesen IDEs möglich, automatisierte Code Reviews durchzuführen. Zu bestimmten Zeiten werden das komplette Projekt gescannt und Berichte über die Code-Qualität und die Einhaltung der PL/SQL Best Practices generiert. Dadurch lassen sich auch aus einem bestehenden großen Projekt die Module herausfinden, die am dringendsten überarbeitet werden sollten. Diese Analysen werden automatisiert durchgeführt, es ist also nicht nötig, wertvolle Entwicklungszeit für Code Reviews zu verwenden. Ganz überflüssig werden die Code Reviews dadurch aber nicht, die Einhaltung der eigenen Konventionen bedarf auch der Überprüfung.

## Fazit

Auch wenn die Zeit knapp ist, sollte ein angemessener Teil der Entwicklungszeit auf die Qualitätssicherung entfallen. Eine solide Codebasis, die sich gut pflegen lässt, ist wichtig, damit das Projekt oder die Anwendung keine unvorhersehbaren Aufwände nach sich ziehen. Dazu gehören natürlich immer realistische Rahmenbedingungen wie ausreichend Zeit und die entsprechenden Prozesse. Kurzfristig betrachtet dauert die Implementierung zwar länger, langfristig gewinnt man jedoch ein Vielfaches der hier eingesetzten Zeit zurück. Zudem unterstützen verschiedene Tools die Anwendungsentwicklung, um ohne allzu viel zusätzlichen Zeitaufwand guten Code zu schreiben.

**Kontakt:**
Thomas Klughardt
thomas.klughardt@quest.com

# Erfahrungen bei der Migration von Oracle SOA Suite 10*g* auf 11*g*

Roland Könn und Danilo Schmiedel, OPITZ CONSULTING Berlin GmbH

**Nach dem lang ersehnten Release der Oracle Fusion Middleware 11*g* wurden nun die ersten Migrationsvorhaben in die Tat umgesetzt. Im Rahmen der Erweiterung bestehender BPEL-Prozesslandschaften konnten die Autoren bereits einige Projekterfahrungen auf diesem Gebiet sammeln.**

Neben den technischen Herausforderungen, die insbesondere dem neuen WebLogic Application Server sowie dem Konzept der Service Component Architecture (SCA) zugrunde liegen, sind bei der Migration auch die Beantwortung lizenzrechtlicher Fragen und ein Upgrade der darunterliegenden Datenbank in Betracht zu ziehen. Wie aus den bisherigen Erfahrungen hervorgeht, bildet die sorgfältige Analyse der bestehenden Architektur und Prozesse einen wesentlichen Grundstein für den Erfolg entsprechender Projekte.

Obwohl mit dem im Oracle JDeveloper integrierten Migration Wizard beziehungsweise dem kommandozeilenbasierten Oracle SOA Suite Upgrade Tool eine (teil-) automatisierte Migration vorhandener BPEL-Prozesse angeboten wird, können dabei nicht zwangsläufig neue Konzepte wie das Complex Event Processing (CEP), der Mediator und die durch SCA hinzugewonnene Abstraktionsschicht berücksichtigt werden. Um von diesen Features profitieren zu können, ist zwar manueller Konzeptionierungsaufwand erforderlich, der jedoch einen entscheidenden Beitrag zur Optimierung bestehender Prozesse leisten kann.

Die Grundvoraussetzung für die Ablösung einer komplexen serviceorientierten Architektur (SOA) und deren Überführung auf eine neue Umgebung bildet eine klare Migrationsstrategie, die im Vorfeld von den Projektbeteiligten festzulegen ist. In diesem Kontext hat sich bisher ein der Abbildung 1 entsprechendes Vorgehen bewährt. Angefangen bei der Analyse der vorhandenen Installationen sollte zu Beginn geklärt werden, ob die zur Verfügung stehenden Hardware- und Software-Ressourcen für die gestiegenen Anforderungen in 11*g* ausreichen. Darüber hinaus sind Fragestellungen hinsichtlich des Betriebs im Cluster sowie des Einsatzes von Features wie etwa dem Business-to-Business (B2B) Gateway oder dem Business Activity Monitoring (BAM) zu diskutieren.

Ein zentraler Punkt bei der Planung der Migration zur Fusion Middleware 11*g* besteht darin, sich zunächst mit den Neuerungen im Vergleich zur Version 10*g* vertraut zu machen. Wie Abbildung 2 zeigt, bringt die Ablösung des Oracle Containers for Java EE (OC4J) zu Gunsten des WebLogic Servers viele neue Aspekte mit sich. Neben einer Reihe neuer Komponenten gibt es auch Begrifflichkeiten, die zwar bereits in der alten Architektur vorhanden waren, dort jedoch eine andere Bedeutung besaßen.

Erfahrungsgemäß führt dies in der Praxis durchaus zu Problemen in der Kommunikation. Ein diesbezügliches Beispiel ist der Begriff der „Domain". Während in 10*g* häufig auf die „Faustregel" zurückgegriffen wurde, für jedes Projekt eine eigene Domain im OC4J-SOA-Container anzulegen, so stellt sie in 11*g* eine Gruppe von logisch zusammengehörigen WebLogic-Server-(WLS)-Instanzen dar, die sich eine gemeinsame Konfiguration teilen. Abstrahiert man diese neue Begrifflichkeit auf die frühere Architektur in 10*g*, so ist dies im Groben mit der Oracle Application Server Cluster Topology vergleichbar.

Innerhalb einer 11*g*-Domain existieren zwei Arten von WLS-Instanzen: Admin Server und Managed Server. Es gibt immer genau einen Admin Server pro Domain, der als zentraler Controller fungiert. Darauf werden die WebLogic Admin Console sowie der Enterprise Manager zur Administration bereitgestellt. Die zentrale Aufgabe des Admin Servers ist die Verteilung von Konfigurationen an die zugehörigen Managed Server. Diese repräsentieren die eigentlichen „Arbeitstiere" im Domain-Kontext. Jeder Managed Server besitzt eine eigene Read-only-Kopie der jeweiligen Konfiguration, die bei jedem Neustart mit dem Admin Server der Domain synchronisiert wird. Managed Server können im Cluster betrieben werden


*Abbildung 1: Vorgehen zur Migration der Architektur und Prozesse*

und sind im Wesentlichen für den Betrieb der Business-Applikationen und -Services zuständig. Im Vergleich zur Version 10*g* entsprechen sie den OC4J-Groups.

### Untersuchung vorhandener BPEL-Prozesse

Nachdem die bestehende Architektur analysiert und Neuerungen in 11*g* beschrieben wurden, sollte man die Applikationen und Prozesse untersuchen, die in der alten Umgebung deployt wurden. In diesem Artikel befassen wir uns speziell mit der Analyse ausführbarer BPEL-Prozesse.

Im Vorfeld eines Migrationsprojekts ist es notwendig, sich auf eine Strategie zu einigen, die der Orientierung über den gesamten Projektverlauf hinweg dient. Dabei sind verschiedene Szenarien in Betracht zu ziehen, die sich in erster Linie nach den zeitlichen Vorstellungen der Projektverantwortlichen und den zur Verfügung stehenden Ressourcen richten. Wichtige Fragen lauten hierbei: „Migriere ich zunächst meinen Hauptprozess oder mein ESB-Projekt und behalte die referenzierten Komponenten für eine Übergangsphase auf dem alten System bei?“ oder „Beginne ich bei meinen Child-Prozessen beziehungsweise -Services und arbeite mich sukzessive nach oben?“ Die Berücksichtigung des Laufzeitverhaltens bildet ebenfalls einen wichtigen Aspekt, da die Migration lange laufender Prozesse im Gegensatz zu einfachen Datentransformationen oder Systemabfragen zusätzliche Fragestellungen aufwirft.

Ein weiterer Punkt, der beachtet werden sollte, betrifft die Integration externer Partner beziehungsweise Systeme und Technologien. Neben einer Untersuchung der Komplexität und der auszutauschenden Daten gehört dazu auch eine Prüfung der verwendeten Adapter, die beispielsweise für die Datenbank-, File-, FTP- oder JMS-Anbindung verantwortlich sind. Hilfsmittel für ein systematisches Vorgehen stellt etwa der Deployment-Deskriptor des BPEL-Prozesses dar, der die Verweise zu allen beteiligten Partnerlinks enthält. Es ist wichtig zu wissen, dass es dem Prozess-Implementierer nun nicht mehr gestattet ist, sogenannte „Non-Managed Connections“ zu verwenden. Dabei handelt es sich im Wesentlichen um Verbindungsdaten beispielsweise zu einer Datenbank, die direkt im Code beziehungsweise im referenzierten JCA-File platziert werden konnten. Zur Erhöhung der Wartbarkeit des Codes wurde in 11*g* auf diese Möglichkeit verzichtet, das heißt, diese Angaben sind nun zwingend als JNDI-Informationen im WebLogic Server zu hinterlegen.

Gerade bei Prozessen mit langen Durchlaufzeiten oder einem enormen Aufkommen an Instanzen waren in der Vergangenheit häufig Workarounds notwendig, um derartige Anwendungsfälle überhaupt in BPEL implementieren zu können. Ähnliche Erfahrungen mussten wir in einigen Projekten auch bei der Verwendung komplexer Datenbankabfragen machen. Tatsächlich sollten diese Lösungen nur als Übergangs- oder Hilfslösungen dienen und beim Aufbau einer neuen Architektur nicht eins zu eins mit übernommen werden. Im Rahmen einer Migration empfiehlt es sich daher, das aktuelle Design der Prozesse und Services aktiv zu hinterfragen und das Optimierungspotenzial auszuschöpfen.

### Installation der 11*g*-Komponenten

Auch bei der Installation hat sich im Vergleich zur Version 10*g* einiges verändert. Das neue Tool zur Erstellung der Datenbank-Schemata ist das sogenannte „Repository Creation Utility“ (RCU), das die Skriptsammlung des Integration Repository Creation Assistant (IRCA) ablöst. Das RCU bietet neben einer grafischen Oberfläche auch Customizing-Funktionen für das Repository an. Als neues Feature ist dabei besonders hervorzuheben, dass nun mithilfe von Präfixen mehrere Schemata einer Komponente (zum Beispiel SOA Infrastructure) in einer Datenbank-Instanz angelegt werden können. Dies hat den Vorteil, nicht mehr an einen festen Schemata-Namen wie etwa „ORABPEL“ gebunden zu sein. Ebenfalls freie Wahl besitzt man bei der Auswahl der Tablespaces. Als Datenbank-Release werden die Versionen 10.2.0.4 beziehungsweise 11.1.0.7 (oder höher) gefordert.

Ein wichtiger Punkt bei der Installation der WebLogic-Server-Software besteht in der Auswahl der Java Virtual Machine (JVM). Hier stehen zwei Optionen zur Auswahl: die Sun JVM oder Oracle JRockit. Oracle empfiehlt an dieser Stelle die Sun JVM für Entwicklungs-Umgebungen und JRockit für den produktiven Betrieb. Die Erfahrungen zeigen allerdings, dass auch die Entwicklung auf der geplanten produktiven JVM durchaus sinnvoll ist. Ein Wechsel der JVM ist auch nach der Installation möglich.

Bei der Installation der Oracle SOA Suite sollte möglichst von der Option Gebrauch gemacht werden, die reine Software-Installation von der Konfiguration zu trennen. Darüber hinaus wird empfohlen, den Installationsvorgang mit der Erzeugung eines Backups abzuschließen, um auf etwaige Probleme in der Konfigurationsphase reagieren zu können.

Der Konfigurationsassistent stellt eine große Vielfalt an Einstellmöglichkeiten bereit, die unter anderem das Setup der Managed Server und JMS File Stores erlauben. Ferner kann im Zuge der Konfiguration eine neue Domain erstellt oder eine bereits bestehende Domain erweitert werden. Erfahrungsgemäß sollte man für die Durchführung der Konfigurationsphase ein großzügiges Zeitkontingent einplanen und auch bei scheinbar „eingefrorenem“ Status nicht sofort den „Abbrechen“-Button des Assistenten betätigen.

### Überführung der Prozesse

Ein weiterer interessanter Ansatz aus Sicht der Service-Entwicklung ist die sogenannte „Service Component Architecture“ (SCA), die im Kern ein Modell zur Entwicklung serviceorientierter Anwendungen beschreibt. Aus der fachlichen Perspektive stellt SCA einen Container dar, der verschiedene SOA-Komponenten enthält (siehe Abbildung 2). Die Abbildung vermittelt den Anwendern einen groben Überblick

bezüglich der vorhandenen Servicekomponenten und deren Beziehungen untereinander. Bei den Komponenten kann es sich beispielsweise um BPEL-Prozesse, Human Tasks, Business Rules oder den Mediator (eine Komponente für das Routing, Filtering und die Transformation von Nachrichten) handeln. Die einzelnen Komponenten und der SCA-Container bieten Schnittstellen für den Aufruf als Service an. Darüber hinaus können sie selbst externe Systeme auf Basis verschiedenster Technologien aufrufen (Reference).

Aus dem funktionellen Blickwinkel repräsentieren SCA Composites abstrakte Objekte, die verschiedene Business-Anforderungen enthalten. Diese Anforderungen werden als gekapselte SOA-Komponenten implementiert, die nach außen nicht sichtbar sind. Aufgrund der Betrachtung mehrerer Servicebausteine als Business Application kann sowohl das Deployment als auch das spätere Monitoring vereinfacht werden.

Mithilfe des integrierten Migration Wizards bietet der Oracle JDeveloper 11*g* die Möglichkeit, BPEL-Prozesse aus früheren Versionen in ein 11*g*-konformes Format zu überführen. Dies geschieht, indem die ursprünglichen Source-Files in der neuen Produktversion geöffnet werden. Es ist sicher nachvollziehbar, dass es sich bei diesem Vorgang maximal um eine teilautomatisierende Migration handeln kann, da Informationen wie Data Sources oder Connection Pools manuell angelegt werden müssen. Beinhalten die früheren BPEL-Prozesse eingebetteten Java Code, so sind auch hier Nachbesserungen einzukalkulieren, weil im Zuge der SCA-Einführung Methodennamen angepasst wurden.

Darüber hinaus sind fundierte Kenntnisse erforderlich. Man sollte wissen, welche APIs sich im Vergleich zur früheren Version unterscheiden und welche Maßnahmen sich daraus ergeben. Dazu gehören unter anderem Modifikationen an Anwendungen, die OracleAS Web Services, Oracle Web Services Proxy und Oracle Web Services SOAP API nutzen. Diese müssen dahingehend verändert werden, dass sie zukünftig die standardbasierte API (JAX-RPC, JAX-WS) des WebLogic Servers verwenden.

Zusätzlich gilt die Einschränkung, dass der Migration-Wizard einen BPEL-Prozess in einen Composite mit genau einem BPEL-Prozess überführt. Mehrere BPEL-Prozesse können nur mithilfe des Oracle SOA Suite Command-Line-Upgrade-Tools einem gemeinsamen Composite zugeordnet werden. Allerdings sind auch hier im Nachgang die oben erwähnten manuellen Anpassungen vorzunehmen.

## Fazit

Mit der SOA Suite stellt Oracle eine Infrastruktur-Lösung bereit, die auf der Grundlage von Web-Service-Technologien die wesentlichen Anforderungen für die technische Implementierung einer SOA unterstützt. Auf diese Weise lassen sich Services erstellen, verwalten und zu modularen Anwendungen und technischen Geschäftsprozessen zusammenstellen. Die einzelnen Komponenten der Produktsuite unterstützen nahezu alle SOA-relevanten Aspekte. Wurden in den früheren Versionen noch die Unübersichtlichkeit der BPEL Console bei steigender Anzahl von Prozess-Instanzen, das schwierige Clustering, der mangelnde Linux-Support für das Business Activity Monitoring und die wenig komfortable Konfiguration als unbequeme Stolpersteine bemängelt, so haben sich diese Punkte mit dem Launch der Fusion Middleware 11*g* entscheidend verbessert. Die übersichtliche Menüführung der Admin Console sowie des Fusion Middleware Enterprise Managers, die flexiblere Installation und die umfassenderen Einstellmöglichkeiten sorgen ebenfalls für gesteigerte Akzeptanz bei Administratoren und Anwendern.

Der erfolgreiche Umstieg setzt jedoch eine klare Migrationsstrategie voraus. Wer seine bestehende Infrastruktur mit allen vorhandenen Stärken und Schwächen kennt, kann optimal von den zahlreichen Verbesserungen in 11*g* profitieren. Den Grundstein hierfür bilden die Auseinandersetzung mit den vorhandenen Prozessen, Services und Applikationen sowie das Verständnis für die neue Architektur und deren Konzepte.

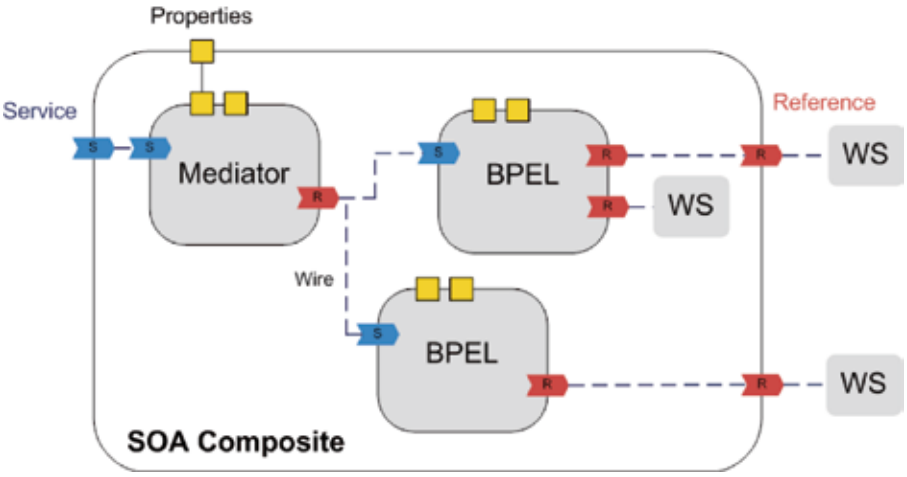



*Abbildung 2: Aufbau eines SCA Composites, Quelle siehe [1]*

### Weitere Informationen

[1] P. Shepherd: „Oracle SCA – The Power of the Composite“, August 2009

**Kontakte:**
Roland Könn
roland.könn@opitz-consulting.com
Danilo Schmiedel
danilo.schmiedel
@opitz-consulting.com

# Conditional Compilation

Dr. Christoph Burandt, ORACLE Deutschland GmbH

**Conditional Compilation ist eine Methode, mit der man unter anderem festlegen kann, welche PL/SQL-Features in einer PL/SQL-Anwendung für ein spezifisches Datenbank-Release verwendet werden. Der Artikel stellt zunächst anhand eines Beispiels das Konzept von Conditional Compilation vor. Im Anschluss daran erfolgt ein Überblick über die gesamte Funktionalität.**

Obwohl ein Unternehmen mittelfristig einen Wechsel der Datenbank-Version plant, entstehen oft noch Anwendungen für die alte Fassung. Im Rahmen der Migration auf die neue Version sind die neuen Features in diese Anwendungen mit entsprechendem Programmieraufwand einzupflegen.

Als Lösung für dieses Problem bietet Oracle seit der Datenbank 10*g* R2 das Feature „Conditional Compilation" an. Hierbei handelt es sich nicht um einen separaten Precompiler, sondern um eine Erweiterung innerhalb des PL/SQL-Compilers [1]. Bevor die eigentliche Kompilierung stattfindet, wird mit einer speziellen „$if"-Bedingung beispielsweise in Bezug zur Datenbank-Version der Code ausgewählt, der anschließend kompiliert wird.

Dazu ein Beispiel: Es gibt in der Datenbank-Version 11*g* für ganze Zahlen den neuen Datentyp „SIMPLE_INTEGER", der zu deutlich kürzeren Laufzeiten führen kann als der bisherige Datentyp „PLS_INTEGER". Unter Verwendung von Conditional Compilation kann man in PL/SQL -Routinen mit der neuen Auswahlanweisung „$if" eine Bedingung beziehungsweise Auswahl definieren, die in Abhängigkeit von der Datenbank-Version den entsprechenden Datentyp auswählt. Hierbei entspricht die Auswahlanweisung „$if" in ihrer Logik der normalen „If"- Anweisung. Beim Kompilierungsvorgang auf der jeweiligen Datenbank-Version wird nur der Programmblock kompiliert, dessen „$if"-Bedingung „true" ergibt. In der folgenden Prozedur ist so eine Auswahlanweisung enthalten. Das Auswahltoken „$if" leitet den Bedingungsblock ein. Die Überprüfung der Version wird mit dem Package DBMS_DB_VERSION (siehe Punkt 5) realisiert. Das Beispiel zeigt den entsprechenden Deklarationsteil der Prozedur P. Der Body spielt hier keine Rolle und ist als Null- Anweisung definiert.

```
CREATE OR REPLACE PROCEDURE P
IS
 $IF DBMS_DB_VERSION.VERSION <
11 $THEN
      v_zahl   pls_integer;
 $ELSE
      v_zahl   simple_integer;
 $END
BEGIN
   null;
END;
```

Der tatsächlich kompilierte Text wird im folgenden SQL*PLUS-Skript mit dem Package DBMS_PREPROCESSOR(siehe Punkt 6) sichtbar gemacht. Hier wurde die Prozedur in der Datenbank-Version 10*g* R2 kompiliert.

```
SQL> call DBMS_PREPROCESSOR.
PRINT_POST_PROCESSED_SOURCE
('PROCEDURE','SH','P');
```

Nachfolgend ist die Ausgabe des nachverarbeiteten Codes dargestellt. Die Variable „v_zahl" wird der Version entsprechend mit dem 10*g*-Datentyp „pls_integer" deklariert.

```
PROCEDURE P
IS
v_zahl   pls_integer;
Begin
   null;
end;
```

### Gesamtkonzept

Conditional Compilation besteht aus folgenden Komponenten:

1. Auswahlanweisung
2. Abfrageanweisungen
3. Parameter „PLSQL_CCFLAGS"
4. Fehleranweisungen
5. Package „DBMS_DB_VERSION"
6. Package „DBMS_PREPROCESSOR"

### Auswahlanweisungen

Die „$if"-Abfrageanweisung entspricht prinzipiell der „if – then – else"-Logik in PL/SQL; die Syntax gleicht sich, abgesehen vom „$"-Zeichen und dem END.

```
$IF <Boolean-Ausdruck> $THEN
$ELSIF <Boolean-Ausdruck> $THEN
Code
$ELSE Code
$END
```

### Abfrageanweisungen

Die Abfrageanweisungen werden allgemein mit zwei „$"-Token eingeleitet. Man unterscheidet zwischen vordefinierten und benutzerdefinierten Abfrageanweisungen. Vordefinierte Abfrageanweisungen beziehen sich auf Parameter wie zum Beispiel den Compilerparameter „PLSQL_CODE_TYPE". Mit „$$PLSQL_CODE_TYPE" lässt sich der aktuell gesetzte Wert abfragen:

```
SQL> exec dbms_output.put_
line($$PLSQL_CODE_TYPE)
INTERPRETED
```

Weitere abfragbare Parameter sind:

- PLSQL_CCFLAGS
- PLSQL_CODE_TYPE
- PLSQL_OPTIMIZE_LEVEL
- PLSQL_WARNINGS
- PLSQL_DEBUG
- PLSQL_LINE
- PLSQL_UNIT
- NLS_LENGTH_SEMANTICS

Benutzerdefinierte Abfragen beziehen sich auf die in dem Parameter „PLSQL_CCFLAGS“ definierten Namen / Wertepaare. Dazu ein Beispiel: Von der Prozedur „P“ wird je nach gesetzten Werten für „Flag1“ und „Flag2“ der entsprechende Code kompiliert. Der Parameter „PLSQL_CCFLAGS“ hat folgende Namen / Wertepaare PLSQL_CCFLAGS = ‚Flag1: true, Flag2: false‘:

```
SQL> CREATE OR REPLACE PROCE-
DURE P
  2  IS
  3  BEGIN
  4
  5   $IF    $$Flag1 $THEN
  6      dbms_output.put_
     line(‚Programmteil_1‘);
  7   $ELSIF $$Flag2 $then
  8        dbms_output.put_
    line(‚Programmteil_2‘);;
  9   $END
 10
 11  END;
 12  /
```

Hier wird Programmteil_1 für die Kompilierung ausgewählt. Es ist naheliegend und auch von Oracle empfohlen, diese Methode fürs Debugging zu verwenden. Sicher gibt es bei den meisten grafischen Entwickler-Tools eine Debugging-Funktionalität. Mit Conditional Compilation kann man aber in Testphasen gezielt einige wichtige Information mit dem altbekannten Package dbms_output ausgeben, ohne die Aufrufe wie früher ein- und wieder auszukommentieren.

Dazu ein Beispiel: Die Prozedur „update_emp“ wird mit Conditional Compilation fürs Debugging konfiguriert. Diese Prozedur erhöht das Gehalt von Angestellten. Das ursprüngliche Gehalt wird durch das „select“-Statement aus der Tabelle „employees“ in die Variable „v_old_sal“ eingelesen. Die „$if“-Anweisung gibt im Debug-Modus mit der Package-Prozedur „dbms_output.put_line“ eine Info-Zeile aus:

```
SQL> alter session set PLSQL_
CCFLAGS= ‚debug:true‘;
```

Anschließend wird die Prozedur UPDATE_EMP kompiliert:

```
SQL> create or replace procedu-
re update_emp(p_emp_id number,
p_new_sal
      number)
  2  is
  3   v_old_sal number;
  4  begin
  5   select salary into v_old_
      sal
  6   from employees
  7   where employee_id = p_
      emp_id;
  8
  9   $if  $$debug = true $then
 10     dbms_output.put_
    line(‚Altes Gehalt: ‚||v_
    old_sal||‘, ‚||
 11                      ‚Neues
     Gehalt: ‚||p_new_sal);
 12   $end
 13
 14   update employees
 15   set salary = p_new_sal
 16   where employee_id = p_
      emp_id;
 17
 18  end;
 19  /
```

Beim Aufruf der Prozedur „update_emp“ werden die Werte der Variablen ausgegeben:

```
SQL> exec update_emp(101,6000)
Altes Gehalt: 5500, Neues Ge-
halt: 6000
```

Nur wenn in Zeile 9 der Wert für den Namen „debug“ den Wert „true“ hat, werden die Zeilen 9 –12 kompiliert. Sobald der Wert von „debug“ auf „false“ gesetzt ist, wird der Debug-Programmteil komplett ausgeblendet und somit für den Produktionsbetrieb verborgen. Dies kann allein durch folgenden Kompilierungs-Befehl ausgeführt werden:

```
SQL> alter procedure update_
emp compile PLSQL_CCFLAGS=
‚debug:false‘;
```

Abfrageanweisungen wie zum Beispiel „$$debug“ werden in folgender Reihenfolge aufgelöst:

1. Der Parameter „PLSQL_CCFLAGS“ wird bei der Kompilierung von rechts nach links ausgewertet. Die Groß-/Kleinschreibung wird dabei nicht beachtet.
2. Die vordefinierten Abfrageanweisungen werden durchsucht.
3. Findet sich allerdings nicht der gesuchte Name, dann wird die Warnung „PLW6003“ ausgegeben. Dies wäre auch bei nicht gesetztem Parameter „PLSQL_CCFLAGS“ der Fall:

```
SQL> show err
Fehler bei PROCEDURE UPDATE_
EMP:
LINE/COL ERROR
-------- ----------------------
9/7      PLW-06003: Unbekannte
Inquiry-Anweisung ‘$$DEBUG’
```

### Parameter „PLSQL_CCFLAGS“

Der Parameter „PLSQL_CCFLAGS“ besteht aus Namen/Wertepaaren. Die Werte können entweder PL/SQL-Literale, PLS_INTEGER-Literale oder Null-Literale (Default) sein. Gemeinsam mit den anderen Kompilierungs-Parametern wird der Parameter „PLSQL_CCFLAGS“ zusammen mit den jeweiligen PL/SQL-Objekten beim Kompilieren abgespeichert. In der View „ALL_PLSQL_OBJECT_SETTINGS“ können diese für jedes Objekt abgefragt werden. Es gibt verschiedene Arten des Kompilierens:

- alter procedure update_emp compile;
  Hier werden alle Session-bezogenen Kompilierungs-Parameterwerte hinzugezogen. Das bedeutet, vorhergehende Parameterwerte werden überschrieben.
- alter procedure update_emp compile reuse settings;
  Der Zusatz „reuse settings“ kompiliert die Prozedur mit den vom letzten Kompilierungsvorgang abgespeicherten Parameterwerten. Hat man es mit abhängigen Prozeduren, Funktionen und Packages zu tun, kompiliert Oracle im Falle einer Invalidierung beim Aufruf automatisch. Hierbei werden auch die abgespeicherten Parameterwerte berücksichtigt.

### Fehleranweisungen

Treffen keine der Bedingungen vom „$if“-Konstrukt zu, besteht die Möglichkeit, einen benutzerdefinierten

Kompilierungsfehler auszulösen. Dafür gibt es die Fehleranweisung „$error“. Sie hat folgende Syntax:

```
$ERROR varchar2_statischer_Aus-
druck $END
```

Dabei muss der statische Ausdruck bei gleicher Umgebung immer den gleichen String ergeben. Zum Beispiel: Der Parameter „PLSQL_CCFLAGS“ wird mit dem Namen „Trace_Level“ und dem zugehörigen Wert „3“ belegt. In der danach erstellten Prozedur ist keine der Bedingungen erfüllt und so wird im „else“-Zweig ein Kompilierungsfehler ausgelöst.

```
SQL> ALTER SESSION SET Plsql_
CCFlags = , Trace_Level:3 , ;

SQL> CREATE PROCEDURE P IS
  2  BEGIN
  3    $IF       $$Trace_Level
       = 0 $THEN ...;
  4    $ELSIF $$Trace_Level = 1
       $THEN ...;
  5    $ELSIF $$Trace_Level = 2
       $THEN ...;
  6    $else $error ,Bad:
       ,||$$Trace_Level $END
  7    $END
  8  END P;
  9  /

Warnung: Prozedur wurde mit
Kompilierungsfehlern erstellt.

SQL>
SQL> show errors
Fehler bei PROCEDURE P:

LINE/COL ERROR
-------- -----------------
6/9      PLS-00179: $ERROR:
Bad: 3
SQL>
```

Innerhalb von Conditional-Compilation-Anweisungen sind nur statische Ausdrücke zugelassen, die bei der Kompilierung vollständig ausgewertet werden können. Sie müssen immer die gleichen Werte liefern, da diese auch bei der automatischen Rekompilierung genauso ausgewertet werden.

### Package „DBMS_DB_VERSION“

Statische Ausdrücke lassen sich auch mit den Konstanten aus dem Package „DBMS_DB_VERSION“ erstellen. Dieses enthält eine Reihe Boole'scher Konstanten und sieht wie folgt aus:

```
  CREATE OR REPLACE PACKAGE
„SYS“.“DBMS_DB_VERSION“ is
  version constant pls_integer
:= 11; -- RDBMS version number
  release constant pls_integer
:= 1;  -- RDBMS release number

  ver_le_9_1   constant boolean
:= FALSE;
  ver_le_9_2   constant boolean
:= FALSE;
  ver_le_9     constant boolean
:= FALSE;
  ver_le_10_1  constant boolean
:= FALSE;
  ver_le_10_2  constant boolean
:= FALSE;
  ver_le_10    constant boolean
:= FALSE;
  ver_le_11_1  constant boolean
:= TRUE;
  ver_le_11    constant boolean
:= TRUE;

end dbms_db_version;
```

„Ver“ steht für Version, „le“ für lower and equal , die erste Zahl gibt die Datenbankversion an, die zweite die Release-Nummer. Zum Beispiel repräsentiert die Konstante „ver_le_10_2“ die Versionen „10 und darunter“ und die Release-Nummer „kleiner gleich 2“. Im Quellcode findet sich im Kommentar eine typische Anwendungssyntax, die für sich spricht:

```
       $if dbms_db_version.
ver_le_10 $then
           version 10 and ea-
lier code
         $elsif dbms_db_versi-
on.ver_le_11 $then
           version 11 code
         $else
           version 12 and later
code
         $end
```

### Package „DBMS_PREPROCESSOR“

Dieses Package ermöglicht die Ausgabe des durch Conditional Compilation ausgewählten Programm-Codes. Es enthält zwei überladene Routinen, die Prozeduren „PRINT_POST_PROCESSED_SOURCE“ sowie die Funktionen „GET_ POST_PROCESSED_SOURCE”. Diese ruft mit den Parametern (OBJECT_TYPE, SCHEMA_NAME, OBJECT_NAME) intern das Package „dbms_output“ auf. Deshalb muss auch im SQL-Developer oder in SQL*PLUS die Umgebungsvariable „serveroutput“ auf den Wert „on“ gesetzt sein.

Das Package „DBMS_PREPROCESSOR“ bietet die Möglichkeit, auch Quellcode über 32 KByte zu verarbeiten. Dazu verwenden die überladenen Unterprogramme „Index-by“-Tabellen, die den Programmtext zeilenweise aufnehmen können. Weitere Information dazu in [2].

## Fazit

Conditional Compilation ist ein einfach zu verwendendes Feature, das insbesondere bei Release-Wechseln oder auch in heterogenen Datenbank-Umgebungen mit Versionen von 9.2 aufwärts den Entwicklern die Arbeit sehr erleichtern kann. Es verwundert ein wenig, dass Conditional Compilation in der Datenbank-Version 10*g* R2 eingeführt wird, aber auch in niedrigeren Versionen – wenn auch mit Einschränkungen – verwendet werden kann. Dies manifestiert sich auch in den Konstanten des Packages „DBMS_DB_VERSION“.

Oracle ermöglicht Conditional Compilation für die Versionen 10.1 durch das Patchset 10.1.0.4 [1] und für 9.2 durch das Patchset 9.2.0.6 [1]. Allerdings wird der Parameter „PLSQL_CCFLAGS“ erst in Version 10*g* R2 eingeführt und steht in früheren Versionen nicht zur Verfügung. Weitere Informationen zur Nutzung von Conditional Compilation in den Versionen 9.2 und 10.1 sind in [1] zu finden.

### Weiterführende Literatur

[1] PL/SQL conditional compilation, An Oracle White Paper, October 2005
[2] Oracle Database PL/SQL Packages and Types Reference 11g Release 1 (11.1)

**Kontakt:**
Dr. Christoph Burandt
christoph.burandt@oracle.com

# Grid Infrastructure 11*g* R2 – eine Grid-Infrastruktur für alle Fälle

Markus Michalewicz, Oracle Corp., Redwood Shores, USA

**Mit der neuesten Version der Oracle Datenbank hat sich auch eine maßgebliche Änderung im Oracle Real-Applications-Umfeld-Clusters-Stack (RAC) ergeben: die Oracle Grid Infrastructure 11*g* R2 wurde eingeführt.**

Die Oracle Grid Infrastructure vereint die beiden vormals selbständigen Produkte Automatic Storage Management (ASM) und Clusterware. Diese Umbenennung und Bündelung macht die Verwendung beider Produkte, die in dem neuesten Release signifikant erweitert wurden, auch außerhalb eines dedizierten Oracle RAC-Stacks interessant. Dieser Artikel zeigt, weshalb die neue Grid Infrastructure 11*g* R2 eine Grid-Infrastruktur für alle Fälle ist, wie sie im Verhältnis zu der neuen Option „RAC One Node" steht und wie sie in den einzelnen Anwendungsfällen eingesetzt werden kann.

## Die Grid Infrastructure

Mit der Bündelung von Clusterware und ASM und unter Verwendung des neuen Namens „Oracle Grid Infrastructure" sowie einem gemeinsamen Oracle Home – dem Oracle Grid Home – adressiert Oracle einen neuen Anwenderkreis: die System- und Storage-Administratoren. Während beide Gruppen bereits mit vorherigen Versionen von ASM und Clusterware angesprochen wurden, ergaben sich meist nur unzureichende Anwendungsfälle außerhalb des Oracle RAC-Stacks, den traditionell die Datenbank-Administratoren verwalten. Das soll sich mit der neuen Grid-Lösung ändern.

Maßgebliche Erweiterungen im ASM-Bereich, wie zum Beispiel die ASM Dynamic Volumes, das damit einhergehende ASM Cluster File System (ACFS) sowie die ACFS-basierte Snapshot-Technologie machen ASM zu einem vollwertigen Speicherverwaltungs- und Volume-Management-Werkzeug, das auch außerhalb von dedizierten Datenbank-Systemen eingesetzt werden kann. Neue und erweiterte Verwaltungswerkzeuge, wie beispielsweise eine vollständige Enterprise-Manager-Integration sowie ein vollwertiges ASMCMD-Kommandozeilen-Interface vervollständigen ASM als eigenständige Speicherverwaltungs-Lösung.

Während ASM die Oracle Datenbank seit seiner Einführung sowohl im Single-Instance- als auch im Cluster-Modus unterstützt hat, war Oracle Clusterware bisher eher auf den Cluster, genauer gesagt auf den Einsatz mit RAC, spezialisiert. Mit der Kombination beider Produkte in der Grid Infrastructure hat sich auch diese eher einseitige Verwendung erweitert, wenn auch mit demselben Ziel – der Bereitstellung von Hochverfügbarkeits-Umgebungen.

Die Grid Infrastructure versteht sich als Infrastruktur, auf deren Basis jede Oracle Datenbank-Umgebung aufgesetzt werden sollte. Egal, ob eine Single Instance-Datenbank oder eine RAC-Umgebung, die Grid Infrastructure ist in jedem Fall die Hochverfügbarkeits-Lösung, um die Oracle Datenbank sowie Applikationen jeder Art noch verfügbarer zu machen. Aus diesem Grunde sieht die Installation der Grid Infrastructure auch zwei grundsätzlich verschiedene Installationsvarianten vor: „Install and Configure Oracle Grid Infrastructure for a Cluster" sowie „Install and Configure Oracle Grid Infrastructure for a Standalone Server" (siehe Abbildung 1).

Obwohl für beide Installationsvarianten dieselbe Software verwendet wird, ist die Umgebung, die jede Variante adressiert, komplett verschieden. Die „Oracle Grid Infrastructure for a Standalone Server" sollte man ausschließlich auf dedizierten Systemen verwenden. Ein typischer Anwendungsfall ist eine reine Single-Instance-Umgebung auf einer dedizierten Server-Hardware. Diese Installationsvariante sollte in keinem Fall mit einem 1-Knoten-Cluster-System verwechselt werden.

Die Standalone-Server-Konfiguration, welche die Restart-Funktionalität bereitstellt, die in Folge noch näher erläutert wird, kann derzeit lediglich durch Neuinstallation in eine Cluster-Konfiguration überführt werden. Die Standalone-Server-Konfigurationbildet

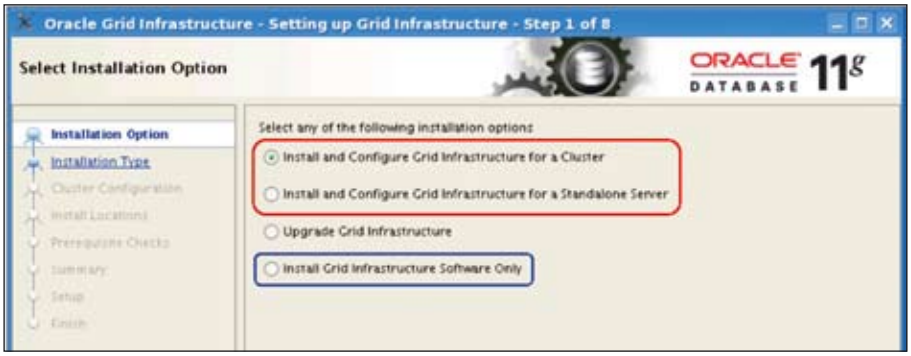



*Abbildung 1: Oracle Grid Infrastructure - Installationsvarianten*

die Grundlage für jede Single-Instance-Datenbank, die auf ASM basieren soll, da ASM 11g R2 nunmehr nur noch aus dem Oracle Grid Infrastructure Home betrieben werden kann.

Für eine RAC-Umgebung, eine Failover-Cluster-Umgebung oder die neue RAC-One-Node-Konfiguration muss die Cluster-Option bei der Installation ausgewählt werden. Diese sieht vor, dass zwei oder mehr Maschinen in einem Cluster-Verbund verwendet werden. Selbst wenn zunächst nur ein sogenannter „1-Knoten-RAC-Cluster" zum späteren Ausbau aufgebaut werden soll, sollte diese Option verwendet werden, da sie – anders als die Standalone-Variante – eine Cluster-Konfiguration vorsieht.

Zur vollständigen Erläuterung von Abbildung 1 noch eine Bemerkung zu den beiden weiteren Installationsvarianten: Während „Upgrade Grid Infrastructure" von der Idee her selbsterklärend ist (Upgrade von Clusterware und / oder ASM zu Grid Infrastructure 11*g* R2), stellt die „Install Grid Infrastructure Software Only"-Option ein weiteres Novum dar.

Die „Software Only"-Option erlaubt erstmalig seit Einführung von Clusterware die Trennung von Installation und Konfiguration. Mit vorherigen Versionen (Clusterware 10*g* R1 und R2 sowie 11*g* R1) war der Benutzer gezwungen, alle Informationen für die Konfiguration bereits bei der Installation der Software anzugeben, da die Konfiguration integraler Bestandteil der Installation war. Gerade für Massen-Deployments ist dies ein nicht unerheblicher Nachteil, der nur durch Rückgriff auf ein Cloning-basiertes Deployment umgangen werden konnte. Die neue Grid Infrastructure löst dieses Problem.

### Architektur-Übersicht

Die Verwendung von Oracle Grid Infrastructure in Cluster-Umgebungen sowie auch auf dedizierten Servern war mit ein Grund dafür, die Clusterware-Architektur den Änderungen entsprechend anzupassen. Augenfällig ist in diesem Zusammenhang die sehr viel stärkere Trennung der einzelnen Schichten innerhalb von Clusterware (siehe Abbildung 2).

Erst die Trennung der Basisschicht, maßgeblich durch den „Oracle High Availability Service Daemon" (OHASD) und den „Oracle Cluster Synchronization Service Daemon" (CSSD) repräsentiert, von der Cluster-Schicht, maßgeblich bestimmt durch den „Cluster Ready Services Daemon" (CRSD), ermöglicht die Unterstützung der beiden unterschiedlichen Konfigurationsvarianten (Cluster- und Standalone-Umgebungen).

OHASD ersetzt alle Daemons, die in vorherigen Versionen aus der Initialisierung („init") heraus gestartet wurden. Er ist verantwortlich für den Start aller weiteren Agents und die Grundlage für eine Standalone-Server-Konfiguration der Oracle Grid Infrastructur.

Clusterware 11*g* R2 setzt verstärkt auf die Verwendung von speicherresistenten Agent-Prozessen, die es der Clusterware erlauben, im Cluster verwaltete Applikationen effizienter zu überwachen. Damit einhergehend kann ein Ausfall der Applikation zeitnaher erkannt und in Folge die vollständige Funktionalität im Cluster schneller wiederhergestellt werden.

Der CSSD ist ein weiterer Daemon, der in jeder der beiden Konfigurationsvarianten (Standalone Server und Cluster) verwendet wird. Als maßgeblicher Fencing Daemon kommt der CSSD auch in dedizierten Umgebungen immer dann zum Einsatz, wenn eine Datenbank auf Oracle ASM basiert. In diesem Fall wurde der CSSD auch in vorherigen Versionen bereits in einer speziellen, lokalen Konfiguration betrieben. Eine Idee, die auch in der neuesten Clusterware-Version aufgegriffen wird, wenngleich auch in erweiterter Form.

In einer Standalone-Umgebung stellen diese beiden Daemons neben den Agenten die entscheidenden Prozesse dar und bilden damit die bereits angesprochene Basisschicht. Im Cluster bilden sie die Grundlage für eine darauf aufsetzende Cluster-Schicht, die im wesentlichen aus dem bereits in vorherigen Versionen verwendeten Cluster Ready Services Daemon (CRSD) besteht sowie den weiteren, in Abbildung 2 dargestellten Prozessen. Die Cluster-

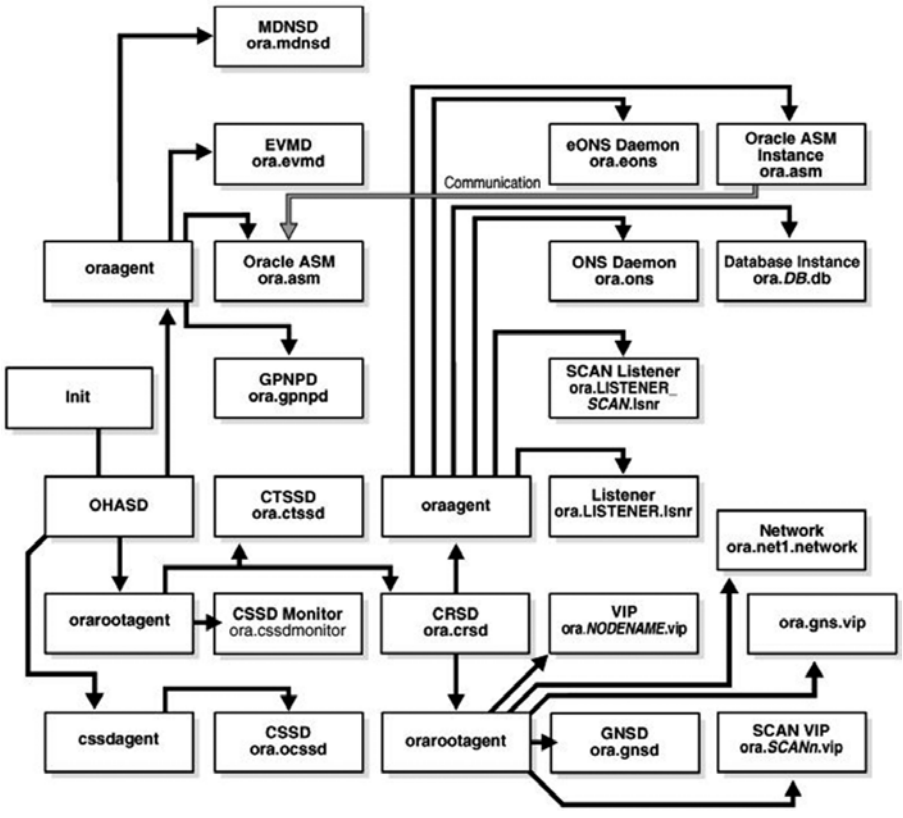



*Abbildung 2: Oracle Clusterware Prozesse und Daemon*

Schicht wird in einer Standalone-Umgebung nicht mitinstalliert.

### Grid-Infrastruktur für alle Fälle

Durch die dargestellte Unterstützung sowohl von Standalone-Server-Umgebungen als auch von Cluster-Umgebungen kann man die Grid Infrastructure 11*g* R2 als generelle Grid-Infrastruktur-Lösung ansehen. Es stellt sich jedoch die Frage, unter welchen Umständen welche Konfiguration eingesetzt werden soll. Zur Beantwortung dieser Frage betrachten wir die wichtigsten Anwendungsfälle näher:

1. Standalone Server und Oracle Restart
2. Cluster und Oracle RAC One Node
3. Cluster und Oracle Real Application Clusters

Ferner wird aufgezeigt, wie sich Failover-Datenbanken von der neuen RAC-One-Node-Option unterscheiden und warum Oracle eine Restart-Funktion für Standalone-Umgebungen anbietet, obgleich im Cluster die RAC-One-Node-Option favorisiert wird.

### Anwendungsfall 1: Standalone Server und Oracle Restart

Die Standalone-Server-Konfiguration sollte man, wie bereits erwähnt, immer dann einsetzen, wenn eine Single-Instance-Datenbank auf einem dedizierten System zum Einsatz gebracht werden soll. Dies ist der maßgebliche Anwendungsfall für diese Konfigurationsvariante.

Mit der Standalone-Server-Variante konfiguriert Oracle die sogenannte „Oracle Restart-Funktionalität". Oracle Restart (englisch für „Neustart / Neuanlauf") hat genau ein Ziel: die Überwachung und den gegebenenfalls erforderlichen Neustart von Oracle Datenbank- oder ASM-Instanzen auf demselben (lokalen) Server. Oracle Restart ermöglicht kein Failover.

Oracle Restart verlässt sich bei der Überwachung und dem eventuell notwendigen Neustart auf die bereits erwähnten und von dem OHASD verwalteten Agenten. Die erforderlichen Informationen, die benötigt werden, um etwa festzulegen, mit welchem Intervall Oracle die Instanzen überprüft und gegebenenfalls neu startet, sind in einer lokalen „Registry" hinterlegt, die in einer Cluster-Umgebung als Oracle Local Registry (OLR) bezeichnet wird. Das Kommandozeilen-basierte Werkzeug SRVCTL, das zusammen mit der Oracle Grid Infrastructure ausgeliefert wird, erkennt die Standalone-Umgebung selbständig und stellt entsprechende Operationen zur Verfügung.

### Anwendungsfall 2: Cluster und Oracle RAC One Node

Kann ein System nicht als dedizierte Standalone-Umgebung klassifiziert werden, weil es zum Beispiel als Grundlage eines später auszubauenden Clusters eingesetzt werden soll oder bereits in einem Cluster integriert ist, sollte die Grid Infrastructure in der Cluster-Variante installiert sein. Dies gilt insbesondere dann, wenn ein Failover-Cluster oder die Verwendung der neuen Oracle RAC-One-Node-Option angestrebt wird.

Diese neue Option zur Enterprise Edition erlaubt im Prinzip die Verwendung eines Ein-Knoten-RACs zu einem reduzierten Lizenzpreis im Vergleich zu der bewährten RAC-Option. Da es sich technisch um eine RAC-Konfiguration handelt, stellt die RAC-One-Node-Option, auch als „virtuelle Single-Instance-Datenbank" bezeichnet, weitere Funktionalitäten zur Verfügung, die weit über die generelle Single-Instance-Funktionalität hinausgehen:

- Live Migration einer Instanz zwischen zwei Servern im Grid („Omotion")
- „Rolling Patching" für Oracle Single-Instance-Datenbanken
- Ein eingebautes Datenbank-Failover im Falle eines Instanz- oder Server-Ausfalls
- Online Upgrade zu einer vollständigen Oracle RAC-Umgebung („pay as you grow")

Die neue RAC-One-Node-Option ist sowohl für bisherige Single-Instance-Kunden interessant, die ihre Datenbank zur besseren Absicherung vor Ausfällen in einem Failover-Cluster betreiben, als auch für Kunden, die ihre Datenbanken derzeit in virtuellen Umgebungen betreiben und einen erhöhten Bedarf an Flexibilität haben. Ferner kann RAC One Node unter Verwendung von „Rolling Patching" für Single-Instance-Datenbanken zur Reduzierung von ge-

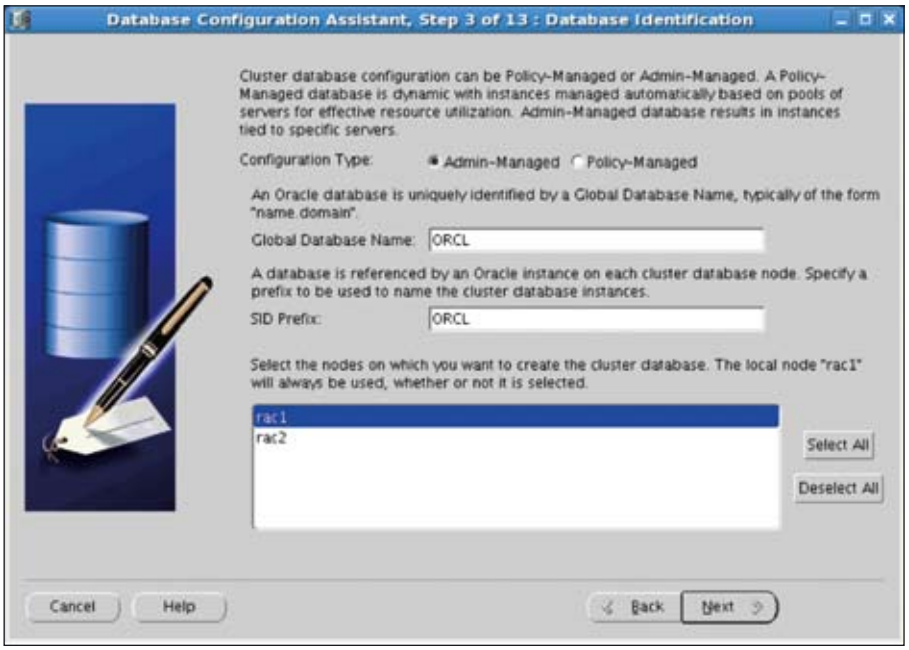



*Abbildung 3: Erstellung einer RAC-One-Node-Datenbank mit dem DBCA*

planten Ausfallzeiten eingesetzt werden. Für Systeme, für die langfristig von einem Bedarfsanstieg ausgegangen werden kann, stellt RAC One Node die ideale Basis für einen späteren Ausbau im Sinne eines Clusters-Systems im Real-Applications-Umfeld dar („pay as you grow"), sollte der initiale Bedarf lediglich eine einzelne Datenbank-Instanz rechtfertigen. Weitere Informationen zu RAC One Node sind unter http://otn.oracle.com/rac zu finden.

### Installation, Konfiguration und Administration

Da es sich bei RAC One Node im Prinzip um einen 1-Knoten-RAC handelt, ist die Mindestvoraussetzung für ein RAC-One-Node-System ein Cluster-System mit zumindest zwei vernetzten Servern. Die Grid Infrastructure muss in der Cluster-Variante installiert sein und das System die Mindestanforderungen für ein RAC-System erfüllen, wie sie in der jeweiligen Dokumentation angegeben sind. Sind diese Voraussetzungen erfüllt, kann man die Datenbank-Software installieren. Für einen RAC One Node gelten dieselben Voraussetzungen wie für einen vollwertigen RAC-Cluster und es kommen dieselben Werkzeuge (Oracle Universal Installer) zum Einsatz.

Alle Server im Cluster sollten die Oracle Datenbank-Software installiert haben, falls eine Migration oder ein Failover zu allen Servern im Cluster gewünscht ist. Soll die RAC-One-Node-Datenbank nur auf bestimmten Servern im Cluster zum Einsatz kommen, kann man die übrigen Server von der Installation der Datenbank-Software ausnehmen. Generell ist von dieser Beschränkung jedoch abzuraten und es wird in Folge davon ausgegangen, dass die erforderliche Datenbank-Software auf allen Knoten im Cluster zur Verfügung steht. Gleiches kann übrigens erreicht werden, wenn eine Installation auf einem verteilten Cluster-Dateisystem (z.B. dem ACFS) vorgenommen wird.

Einige Besonderheiten ergeben sich bei der Erstellung der Datenbank mit dem Database Configuration Assistant (kurz DBCA). So muss für eine RAC-One-Node-Datenbank eine Administrator-verwaltete Datenbank (im DBCA als „Admin-Managed" bezeichnet, siehe Abbildung 3) angelegt werden, da RAC One Node in der ersten Version lediglich Administrator-verwaltete Datenbanken unterstützt. Für eine Übersicht über die beiden neuen Datenbank-Verwaltungsoptionen in der Database 11*g* R2 sei auf die offizielle Dokumentation verwiesen.

```
[RAC]> Omotion

RAC One Node databases on this cluster:

 #  Database      Server          Fix Required
=== ========      ===========     ==================
[1] ORCL          rac2            N

Enter number of the database to migrate [1]: 1

Specify maximum time in minutes for migration to complete (max 30) [30]: 30

Available Target Server(s) :
 #      Server       Available
===     ==========   =========
[1]     rac1         Y

Enter number of the target node [1]: 1

Omotion Started...
Starting target instance on rac1...
Migrating sessions...
Stopping source instance on rac2...
Omotion Completed...

=== Current Status ===
Database orcl is running on node rac1
```

*Abbildung 4: Omotion-Beispiel*

Des Weiteren darf für eine RAC-One-Node-Datenbank nur ein Knoten im DBCA zur Erstellung der Datenbank ausgewählt werden. Nun ist es eine Besonderheit im DBCA, dass der Server, auf dem der DBCA gestartet wurde (lokaler Knoten), bereits vorselektiert ist und nicht von der Selektion ausgenommen werden kann. Dies sollte man gegebenenfalls beim Start des DBCAs berücksichtigen. Im Endeffekt kann eine RAC-One-Node-Datenbank nach der Installation auf einem beliebigen Knoten zum Einsatz kommen. Notfalls muss die entsprechende Instanz nach dem Start mittels Omotion migriert werden.

Nachdem eine Datenbank entsprechend der RAC-One-Node-Spezifikationen erstellt wurde, muss die neu erstellte Datenbank noch als RAC-One-Node-Datenbank initialisiert werden. Dies geschieht mittels des raconeinit.sh.Skripts, das als Patch zur Verfügung stehen wird. Zum Zeitpunkt der Erstellung dieses Artikels waren die RAC-One-Node-Skripte noch nicht offiziell verfügbar.

Generell gilt für die Administration der ersten Version von RAC One Node, dass sie durchgehend skript- beziehungsweise kommandozeilenbasiert ist. Für die nicht allzu ferne Zukunft ist jedoch geplant, die Verwaltung der RAC-One-Node-Datenbanken in das bereits erwähnte SRVCTL-Kommandozeilen-Werkzeug zu integrieren. Auch eine Integration in den Oracle Enterprise Manager sowie eine bessere Unterstützung durch den DBCA sind geplant.

Sobald eine Datenbank als Oracle RAC-One-Node-Datenbank initialisiert wurde, profitiert sie von den zusätzlichen Funktionalitäten der RAC-One-Node-Option. Das bedeutet zum Beispiel, dass sie automatisch neu gestartet wird, sollte die Instanz abfallen. Kann die Instanz auf dem lokalen Server nicht neu gestartet werden (aus welchen Gründen auch immer), wird sie einem Failover unterzogen und es wird versucht, sie auf einem anderen Server zu starten.

### Omotion oder Live Migration einer Instanz

Die maßgebliche Innovation von RAC One Node ist zweifelsohne die Mög-

lichkeit, Instanzen nahezu transparent für die Anwendung von einem Server zu einem anderen im Grid zu verschieben oder zu migrieren. Diese Aktion, im RAC-One-Node-Jargon „Omotion“ oder „Live Migration“ genannt, wird ebenfalls mittels Skript initialisiert. Abbildung 4 zeigt eine typische Omotion-Sequenz.

Zu beachten ist, dass die RAC-One-Node-Datenbank während der Omotion-Phase mit zwei Datenbank-Instanzen und damit als vollwertige RAC-Datenbank betrieben wird. Im Cluster verwaltete Services stellen in dieser Zeit sicher, dass nur eine Datenbank-Instanz für neue Verbindungsanfragen zur Verfügung steht. Die zu migrierende Datenbank-Instanz befindet sich nach der Omotion-Initialisierung im „shutdown transactional“-Status und wird nach Ablauf der im Skript angegebenen Wartezeit (siehe Abbildung 4, „maximum time in minutes for migration“), maximal jedoch nach 30 Minuten, mit einem „shutdown abort“ heruntergefahren, sollten zu dieser Zeit noch Transaktionen offen sein.

### Anwendungsfall 3: Cluster und Oracle Real Application Clusters

Dieser Anwendungsfall stellt die klassische Oracle RAC- und Clusterware-Konfiguration dar. Diese Variante sieht vor, dass die Grid Infrastructure in der Cluster-Konfiguration installiert wurde und folgt auch sonst den bereits bekannten RAC-Vorgaben. Entsprechend stellt sie alle bereits bekannten RAC-Vorteile zur Verfügung, die in der Datenbank 11g R2 hinsichtlich einer dynamischeren und flexibleren Grid-Verwaltung noch erweitert wurden. Für weitere Informationen in Bezug auf die generellen Neuerungen in der Datenbank 11g R2 mit der RAC-Option und der Grid Infrastructure 11g R2 stehen diverse Whitepaper unter http://otn.oracle.com/rac und http://otn.oracle.com/clusterware bereit.

### Abgrenzung 1: Oracle Restart und RAC One Node

Oracle Restart und RAC One Node unterscheiden sich grundlegend. Wie bereits beschrieben, erfordert die Restart-Funktionalität eine Installation der Grid Infrastructure in der Standalone-Variante, die derzeit nur mittels einer Neuinstallation in eine Cluster-Konfiguration überführt werden kann. Ferner stellt Oracle Restart lediglich eine Überwachungs- und Neustart-Funktionalität zur Verfügung. RAC One Node unterstützt überdies eine Failover-Funktionalität sowie Omotion im Cluster oder im Grid.

### Abgrenzung 2: RAC One Node und Failover-Datenbanken

Selbst unter der Annahme, dass die Failover-Datenbanken auf Basis von Oracle Clusterware aufgesetzt und verwaltet werden, unterscheiden sich Failover-Datenbank und RAC-One-Node-Datenbanken grundlegend.

Ein wichtiger Unterschied ist, dass eine RAC-One-Node-Datenbank aus einem Oracle Datenbank-Home heraus betrieben wird, das mit der RAC-Option gelinkt wurde. Dies sollte für eine Failover-Datenbank nicht der Fall sein. Von diesem Unterschied leiten sich diverse andere Merkmale ab, die eine RAC-One-Node-Datenbank sehr viel leistungsfähiger und flexibler erscheinen lassen als eine klassische Failover-Datenbank.

Für Clusterware-verwaltete Failover-Datenbanken gilt zudem ein beschränkter Support seitens Oracle. Im Prinzip handelt es sich dabei um eine Skript-basierte Lösung, für die Oracle die Skripte selbst nicht vollständig unterstützt, wohl aber die Integration bestehender Skripte in die Clusterware-Umgebung. Weitere Informationen zu diesem Thema stehen in der Metalink Note 790189.1. Eine RAC-One-Node-Datenbank stellt im Unterschied dazu eine vollständig supportete Failover-Funktionalität für eine Oracle Datenbank dar.

Zudem unterstützt die RAC-One-Node-Datenbank Omotion, Rolling Patching für Oracle Single-Instance-Datenbanken sowie ein RAC Online Upgrade. Diese Funktionen bieten klassische Failover-Datenbanken nicht.

### Fazit

Die neue Grid Infrastructure 11*g* R2 adressiert erstmals direkt Anwendungsfälle außerhalb der klassischen Oracle RAC-Umgebung. Automatic Storage Management wurde zu einer universellen Speicherverwaltungslösung erweitert und eine generelle Unterstützung für Standalone Server sowie Cluster-Umgebungen durch die Oracle Clusterware eingeführt. Beide Maßnahmen machen die Grid Infrastructure zur universellen Basis für hochverfügbare Oracle Datenbanken. RAC One Node erschließt zudem diverse Anwendungsfälle im Cluster, für die eine vollständige RAC-Lösung bisher als überdimensioniert oder als zu unflexibel angesehen wurde.

**Kontakt:**
Markus Michalewicz
markus.michalewicz@oracle.com

# DOAG ist Gründungsmitglied des Interessenverbund der Java User Groups e.V. (iJUG)

Vor dem Hintergrund der geplanten Sun-Übernahme durch Oracle hat die DOAG zusammen mit sechs Java-Usergroups aus Deutschland den iJUG-Interessenverbund der Java User Groups e.V. gegründet. Ziel des Vereins ist die umfassende Vertretung der gemeinsamen Interessen der Java-Anwendergruppen sowie der Java-Anwender im deutschsprachigen Raum, insbesondere gegenüber Entwicklern, Herstellern, Vertriebsunternehmen sowie der Öffentlichkeit. Bei Wahrung der Eigenständigkeit der Mitglieder sollen insbesondere eine gemeinsame Öffentlichkeitsarbeit stattfinden und der Informations- und Erfahrungsaustausch sowie Netzwerkbildung gefördert werden.

Der Vorstand des iJUG setzt sich zusammen aus:

- Fried Saacke (DOAG Deutsche ORACLE-Anwendergruppe e.V.)
- Oliver Szymanski (Java User Group Erlangen-Nürnberg)
- Frank Schwichtenberg (Java User Group Deutschland e.V.)

„Die Java-Community war schon immer sehr eng mit der Oracle-Technologie verknüpft“, erklärt Dr. Dietmar Neugebauer, Vorstandsvorsitzender der DOAG. „Deshalb ist die Gründung einer Interessenvertretung der Java-Gemeinde ein sinnvoller und notwendiger Schritt, gerade im Hinblick auf eine gemeinsame Vertretung gegenüber dem Hersteller.“

Die sieben Gründungsmitglieder des iJUG sind:

- Java User Group Deutschland e.V.
- Java User Group München
- Java User Group Erlangen-Nürnberg
- Java User Group Stuttgart e.V. (JUGS)
- Java User Group Saxony, Dresden
- Java User Group Köln
- DOAG Deutsche ORACLE-Anwendergruppe e.V.

Ziel der Gründungsmitglieder ist es, alle Java-Usergroups unter einem Dach zu vereinen. So können sich alle interessierten Java-Usergroups in Deutschland, Österreich und der Schweiz, die sich für den Verbund interessieren und ihm beitreten möchten, gerne beim iJUG melden.

Eine Mitgliederversammlung wird die mittelfristigen Ziele und Meilensteine sowie erste gemeinsame Aktivitäten festlegen.

**Kontakt**
Fried Saacke
sun@doag.org

# visualDependencies for Databases – Visualisierung der Abhängigkeiten von Datenbank-Objekten

Prof. Dr. Heide Faeskorn-Woyke, Andre Kasper und Jan Philipp, FH Köln sowie Dr. Andreas Behrend, Uni Bonn

**Ein bedeutender Teil der heutzutage zu speichernden Daten ist nicht in semistrukturierten Systemen wie Web-Anwendungen, Katalogen, Portalen, Blogs etc., sondern nach wie vor in relationalen Datenbanken gespeichert. Erstaunlicherweise existiert nach Recherche der Autoren bisher kein visuelles Werkzeug, das verschiedenartige Beziehungen zwischen relationalen Tabellen, Views und Datenbank-Triggern visualisiert. Ein Open-Source-Projekt schafft Abhilfe.**

Es gibt zwar eine Reihe von visuellen Werkzeugen für Datenbank-Systeme oder Anwendungen, die einige grafische Zusatzfunktionen anbieten. Nahezu alle beschränken sich auf die Darstellung der Fremdschlüsselbeziehungen und zielen daher auf die referentielle Integrität. Für die Entwicklung eines Datenbank-Modells und dessen Verständnis ist eine solche Darstellung hilfreich. Hingegen verschafft es dem Datenbank-Administrator oder einem interessierten Anwender nicht die gewünschte Transparenz, um die komplexen, wechselseitigen Abhängigkeiten der Views und Tabellen nachvollziehen zu können.

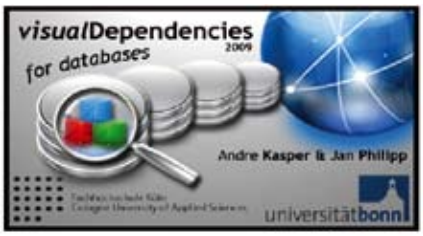



*Abbildung 1: visualDependencies*

„visualDependencies for Databases“ wurde im Rahmen einer Diplomarbeit von Andre Kasper und Jan Philipp an der FH Köln in Zusammenarbeit mit der Universität Bonn entwickelt und stellt die Abhängigkeiten von Objekten einer Datenbank transparent für den Anwender dar. Das Open-Source-Projekt steht unter http://drop.io/visualDependencies zum Download für Windows, Mac OS X und Linux bereit. Es richtet sich an Datenbank-Entwickler und -Administratoren, die sich einen visuellen Überblick über die in ihrer Datenbank vorhandenen verschiedenartigen Beziehungen machen möchten.

Datenänderungen können nicht vorgenommen werden. Die Tabellen, Trigger und Views werden dazu zunächst analysiert und anschließend mittels Graphen visualisiert. Die Anwendung stellt dabei vier Funktionalitäten zur Verfügung: Verbindungsübersicht, View-Hierarchie, Trigger und Fremdschlüsselbeziehungen. Die Verbindungsübersicht zeigt alle angelegten Datenbank-Verbindungen an. Weiterhin werden die Views, die jeweils Abhängigkeiten zu anderen Tabellen und Views besitzen, in einem Graphen dargestellt. Die vorhandenen Trigger eines Datenbank-Schemas mit ihren Abhängigkeiten werden ebenfalls gesondert dargestellt. Als vierte Funktionalität werden die Tabellenabhängigkeiten und Fremdschlüssel in einem einfachen ERD veranschaulicht.

visualDependencies unterstützt zunächst nur Oracle 9*i*/10*g*/11*g*-Datenbanken, ist aber grundsätzlich herstellerunabhängig implementiert und besitzt schon eine experimentelle MySQL-Schnittstelle. Der OR-Wrapper „Hibernate“ mit einer HSQL-Datenbank speichert die bereits analysierten Ver-


*Abbildung 2: Verbindungsübersicht*

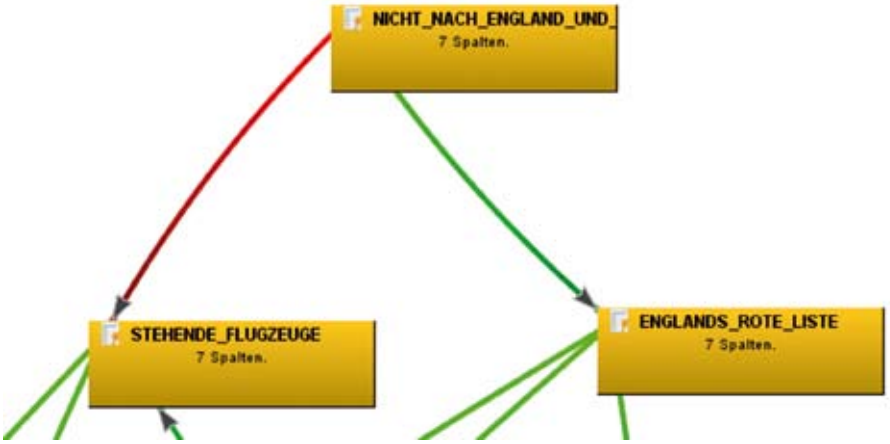



*Abbildung 3: Abhängigkeiten von Views in Datenbanken*

bindungsobjekte. Die Benutzerschnittstelle ist mit Java und Eclipse realisiert.

### Die Verbindungsübersicht

Beim Starten von visualDependencies wird die Verbindungsübersicht direkt geöffnet und zeigt dem Benutzer einen Willkommensbildschirm, der eine Übersicht über die Funktionalitäten der Anwendung gewährt. Die Verbindungsübersicht besteht aus zwei verschiedenen Bereichen. Im linken werden die bereits angelegten Datenbank-Verbindungen sowie schon in der Datenbank abgelegte Objekte (Tabellen, Views und Trigger eines Datenbank-Schemas) angezeigt.

Es können neue Verbindungen erstellt werden, aber keine sonstigen Datenbank-Objekte. Der rechte Bereich enthält eine Beschreibung der Grundfunktionalitäten der Anwendung mit einem kurzen Hilfetext.

### Die View-Hierarchie

Zwischengespeicherte Sichten für oft angeforderte Anfragen dienen in einem DBMS zur Optimierung der Abfrage-Performance und des Datenaufkommens. Diese Sichten erreichen oft einen hohen Grad an Komplexität, da sie auch in einer View-Hierarchie oder sogar rekursiv aufeinander aufbauen können. Um die Visualisierung von Views geht es in der zweiten grafischen Ansicht, in der Views und Tabellen angezeigt werden, die Abhängigkeiten zu anderen Tabellen oder Views in der Datenbank besitzen. Dabei werden diese Abhängigkeiten weiter differenziert. Positive Abhängigkeiten sind solche, in denen der WHERE-Bedingungsteil der SELECT-Anfrage eine Enthaltungsmenge durch Operatoren wie „=“ oder „LIKE“ spezifiziert. Neutrale SELECT-Anweisungen ohne WHERE-Bedingungen sind immer positiv. Negative Abhängigkeiten sind solche, in welchen der WHERE-Bedingungsteil explizit Daten ausschließt. Hier wird die äußere SELECT-Anweisung mittels eines NOT EXISTS mit einem Subselect oder einem MINUS verbunden. Die Farben der Kanten im Graphen sind farblich durch den Kontext ihrer Abhängigkeit bestimmt. Während grüne Kanten positive Abhängigkeiten repräsentieren, sind negative Abhängigkeiten durch rote Kanten hervorgehoben. Die Kanten objekte zeigen in einer kleinen Box die genauen Details der Abhängigkeit, wenn der Mauszeiger über sie gehalten wird.


*Abbildung 4: Abhängigkeiten zwischen Triggern*

### Datenbank-Triggersicht

Um der steigenden Komplexität der Datenbanken gerecht zu werden, helfen oft automatische, interne Aufträge (Trigger) im Datenbank-System. Die Trigger definieren ein Stück Programmcode und führen ihn zu einem bestimmten Zeitpunkt und unter einer bestimmten Bedingung aus. Der Zweck eines Triggers ist es, die Datenintegrität aufrechtzuerhalten oder in Verknüpfung mit materialisierten Views für konsistente Daten zu sorgen. Allerdings ist es nicht so ohne Weiteres ersichtlich, ob die Ausführung eines Triggers erfolgreich sein kann, wenn dadurch weitere Trigger angestoßen werden. Erst zur Laufzeit, also nach dem statischen Analysieren und gegebenenfalls auch Parsen des Triggers, kann ein Fehler auftreten. Damit stößt man jedoch schnell an die Grenzen des Machbaren. Dies ist auch ein Grund, warum beispielsweise Oracle trotz der beschriebenen Problematik anstandslos das Anlegen von Triggern akzeptiert, was zu rekursiven Abläufen führen oder das Mutating-Table-Problem auslösen kann.

In der dritten Ansicht werden die vorhandenen Trigger der Datenbank analysiert, die auf den Tabellen liegen. Mit dieser Funktionalität wird sowohl eine Übersicht der zu einer Tabelle verfügbaren Trigger erzeugt als auch die Beziehungen der Trigger untereinander visualisiert. Dabei werden durch die Abhängigkeiten die impliziten, potenziellen Aufrufe anderer Trigger sichtbar gemacht. Die Informationen über zyklische, rekursive Abfolgen oder eines mög-

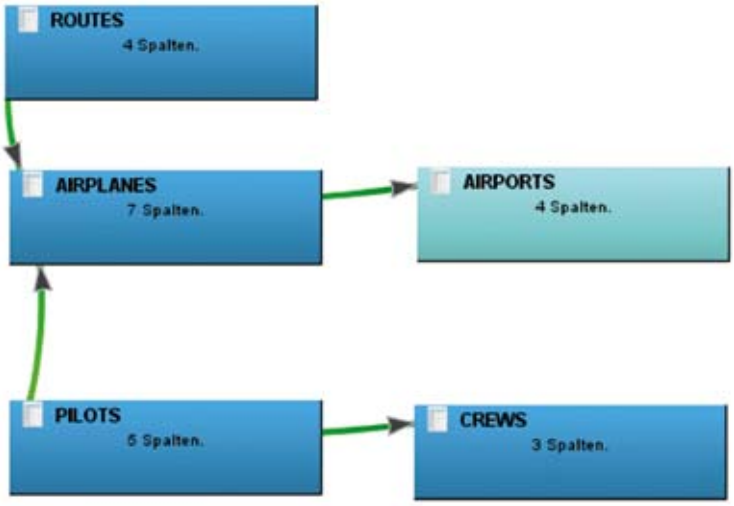



*Abbildung 5: Fremdschlüsselbeziehungen in Datenbanken*

lichen Triggers mit Mutating-Table-Problem werden im Graphen dargestellt.

Die Trigger-Objekte und die Verbindungen (Kanten) werden in dieser Darstellung grün angezeigt, solange es zu diesem Objekt keine besonderen Auffälligkeiten gibt. Ein Trigger, der ein Mutating-Table-Problem hervorrufen kann, wird in der Ansicht orange dargestellt, und es gibt einen Hinweistext in diesem Objekt. Weiterhin wird eine rote Kante von diesem Trigger auf sich selbst angezeigt. Die zweite Auffälligkeit, die gesondert dargestellt wird, ist die Rekursion unter den Triggern. Die Farbe der Triggerobjekte ist rot, ebenso wie die rekursiven Kanten, die diese Trigger verbinden.

Das Entity-Relationship-Diagramm in Form von Fremdschlüsselbeziehungen ist die vierte und letzte Funktionalität von visualDependencies. Im Gegensatz zu anderen Produkten beschränkt sich die Anzeige jedoch auf die lose Darstellung der Abhängigkeiten ohne weitere Details wie Kardinalität oder Art der Assoziation. Einzig und allein die Richtung der Abhängigkeit wird dargestellt, weshalb die Visualisierung auch durch einen Graphen realisiert wird.

### Ausblick

Mit diesen Funktionalitäten wurde ein hilfreiches Werkzeug bei der visuellen Betrachtung und Untersuchung von Datenbank-Schemata erstellt. Es gibt noch zahlreiche weitere Ideen, um die Darstellung zu verbessern oder weitere Funktionalitäten aufzunehmen. So wäre eine andere visuelle Darstellungsmöglichkeit, eventuell auch dreidimensional, denkbar, die die Tabellen mehr in das Zentrum der Betrachtung stellt. Dabei würden alle drei Ansichten der Anwendung – Views, Trigger und das ERD mit Fremdschlüsseln – vereinigt und bei Bedarf im Graphen jeweils zu den Tabellenobjekten auf unterschiedliche Weise angezeigt. Für eine geeignete Darstellung der Knoten und Kanten eines Graphen sind weitere Alternativlayouts möglich. Insbesondere die Problematik der sich überschneidenden Kanten kann durch effizientere Algorithmen gelöst werden.

Bei der Visualisierung der Abhängigkeiten in einer Datenbank gibt es weitere, für diese Anwendung interessante Objekte: Integritätsbedingungen, Funktionen, Prozeduren und Typen. Ähnlich wie Trigger enthalten Funktionen und Prozeduren SQL-Code, der sowohl vordefinierte als auch dynamisch zusammengesetzte SQL-Anweisungen enthalten kann. Damit ergeben sich weitere direkte oder indirekte Abhängigkeiten auf andere Objekte in der Datenbank. Auch die Aufschlüsselung einer objektrelationalen Typhierarchie kann in Erwägung gezogen werden.

**Kontakte:**
andre.kasper@fh-koeln.de
jan@philipp-online.de
behrend@cs.uni-bonn.de
heide.faeskorn-woyke@fh-koeln.de

## Vorschau auf die nächste Ausgabe

Das Schwerpunkt-Thema der Ausgabe Q2/2010 lautet:

**„Sun und Java“**

Gesucht sind Beiträge zu folgenden Themen:

- Oracle-Produkte für Java: WebLogic, JDeveloper, ADF 11*g*
- Open-Source-Projekte für Java: Spring, Eclipse etc.
- JavaFX
- Rapid Application Development mit Java
- Java Virtual Machines (Suns JVM, Oracle JRockit, OpenSource-Varianten)
- Java Community Process sowie aktuelle JSRs
- Sun Server und die “Sun Oracle Database Machine”

Darüber hinaus finden Sie wie in jeder Ausgabe:

- Fachartikel aus den Bereichen „Datenbank“ und „Entwicklung“
- Best Practice im Umgang mit den Oracle-Produkten
- Aktuelle Informationen über Veranstaltungen der DOAG
- Neues von Oracle

**Redaktionsschluss ist am 5. März 2010**

Die Ausgabe Q2/2010 erscheint am 7. Mai 2010

# Oracle Real Application Cluster – Best Practices

Rainier Kaczmarczyk und Andrew Lacy, OPITZ CONSULTING München GmbH

**Dieser Artikel richtet sich an Datenbank-Administratoren, die mit der RAC-Technologie vertraut sind. Aus eigenen Erfahrungen insbesondere beim Setup des RAC haben die Autoren hilfreiche Tipps und Tricks zusammengestellt. Diese sollten das Setup erleichtern und für einen stabilen Betrieb des Systems sorgen.**

### Besonderheiten von Windows

- Ab Windows Server 2003 ist das Auto-Mounting nicht mehr per Default eingeschaltet. Hierzu findet sich zwar ein Hinweis in den Installationsanweisungen, dieser Umstand wird aber gerne übersehen, obwohl das Tool „diskpart“ Auto-Mounting mit einem einzigen Befehl (automount enable) aktiviert. Die Installation der Clusterware schlägt fehl, weil der Installer die Platten für die Votingdisk und OCR nicht findet. Ein aufwändiges manuelles Aufräumen gemäß Metalink-Note 341214.1 ist die Folge.
- Auch wenn Windows selbst keinen Unterschied zwischen groß- und kleingeschriebenen Hostnamen macht, schlägt die Installation der Clusterware bei der Verwendung von Großbuchstaben fehl. Obwohl alle notwendigen Prozesse laufen, meldet das Werkzeug „Cluster-verify“ (cluvfy) Fehler. Großbuchstaben in Hostnamen sind daher absolut zu vermeiden!
- Die Bindungsreihenfolge der Netzwerkkarten ist unbedingt zu beachten! Die Netzwerkkarte des öffentlichen (public) Netzwerks muss die erste sein. Die Einstellung hierzu ist zu finden unter: Systemsteuerung –> Netzwerkverbindungen –> Erweiterte Einstellungen


*Abbildung 1: Screenshot zu den erweiterten Einstellungen für die Netzwerkkarte*

### Besonderheit von HP-UX

- Das Betriebssystem HP-UX hat beim ASYNC-Interface per Default keinen Timeout gesetzt. ASM hat umgekehrt die Fähigkeit, einzelne Platten aus einer Diskgruppe herauszunehmen, wenn diese nicht mehr antwortet. Dazu muss aber der Timeout auf einen sinnvollen Wert gesetzt sein, weil andernfalls der I/O auf die Platten via ASM einfach hängen bleibt.

### Besonderheit von Linux

- Die OCR und die Voting Disk sollten bei Verwendung eines Linux-Systems nicht auf Raw Devices installiert werden, da diese in Zukunft nicht mehr unterstützt werden. Stattdessen sollte man ab Oracle 10g O_DIRECT als Filesystem-Option verwenden. Faktisch wird das Raw Device durch ein Block Device ersetzt.

### Betriebssystemunabhängige Tipps

- Weitgehend unbekannt ist, dass die Version von Clusterware, ASM und Datenbank unterschiedlich sein kann. Dies ist besonders für Neuinstallationen interessant, bei denen der Hersteller einer Anwendung beispielsweise noch RDBMS 10.2.0.4 vorschreibt. Beispielsweise lässt sich der zukünftige Upgrade-Aufwand

für Clusterware und ASM entscheidend reduzieren, indem man gleich beide mit der derzeit höchsten verfügbaren Version (11.1.0.7) installiert. Die zertifizierten Kombinationen sind in der Metalink-Note 337737.1 zu finden.

- Die Verwendung der asmlib wird nicht empfohlen, da deren Vorteile nicht den Nachteil einer zusätzlichen Kernelabhängigkeit in einem Linux-System aufwiegen.
- In 10g war die Mindestgröße für die Voting Disk 20 MB, die der OCR 100 MB. In 11*g* stieg dieser Wert bereits auf je 262 MB. Aufgrund dieses eindeutigen Trends wird für die Upgrade-Fähigkeit auf neuere Versionen eine Mindestgröße von je 1 GB für die Block Devices empfohlen.
- Seit 11*g* bietet Oracle den Rolling Upgrade der ASM-Instanzen an. Um diesen zu ermöglichen, müssen drei ORACLE_HOME-Verzeichnisse verwendet werden, eines für ASM (Automatic Storage Management), eines für OCR (Oracle Cluster Registry) und eines für die eigentliche Datenbank. Sinnvoll ist hier nur die Verwendung von non-shared Oracle Homes, das heißt, jeder Rechner hat ein eigenes Oracle Home. Leider endet die Rolling-Upgrade-Fähigkeit von Oracle bei ASM. Ein unterbrechungsfreier Upgrade der eigentlichen Datenbank wird derzeit nicht unterstützt. Das wäre einer der größten Wünsche der Autoren für die Version 12*g*.
- Wird auf einem RAC-System als Filesystem ASM verwendet und mehr als eine Datenbank betrieben, darf man nicht vergessen, den Wert für den Initialisierungsparameter „processes“ entsprechend der Formel 25+15*(Anzahl der Datenbanken) für die ASM-Instanzen zu setzen. Wird dies nicht geändert, kann ASM clusterweit nicht mehr korrekt funktionieren.
- Das größte und meist auch ungelöste Problem einer RAC-Installation ist die Voting Disk. Per Definition kann nur eine ungerade Anzahl von Voting Disks festgelegt werden. Verwendet man nur eine (dies ist bei der Mehrheit der Installationen der Fall!), wird die Voting Disk praktisch zur Achillesferse des RAC Clusters. Verwendet man drei Voting Disks, stellt sich die Frage, wo die dritte Disk liegen soll. Da hier sinnvollerweise ein extra Plattensystem verwendet werden soll, wird aus Kostengründen gerne auf diese zweite Variante verzichtet. Um diesen einen Single Point of Failure auszuschließen, sollte auf ein relativ kostengünstiges zusätzliches Laufwerk (z.B. iSCSI) nicht verzichtet werden.
- Ein wenig beachtetes Problem ist die Überlastung von speziellen Zwei-Knoten-Installationen: Sind beide Knoten im Normalbetrieb so hoch belastet, dass der verbleibende Knoten im Fehlerfall überlastet wird, führt dies zu massiven Problemen. Hintergrund ist die Tatsache, dass einige Prozesse (speziell bei Clusterware) durch die Überlast in ein Timeout laufen und dadurch einen Reboot des verbleibenden Systems auslösen. Daher ist eine solche Überlast des Systems unbedingt zu vermeiden!

### Fazit

Wie man sieht, sind es oft nur Kleinigkeiten, die zu massiven Problemen bei Setup und Betrieb eines RAC-Systems führen können. Die dargestellten Hinweise können helfen, diese Schwierigkeiten zu vermeiden.

**Kontakte:**
Rainier Kaczmarczyk
rainier.kaczmarczyk
@opitz-consulting.com
Andrew Lacy
andrew.lacy@opitz-consulting.com

## Start der eigenständigen SIG Java

Die DOAG startet im April 2010 die neue Special Interest Group Java. Diese wird in Zukunft als eigenständige Veranstaltung neben der SIG Development geführt. Die SIG Development wird dann den Themen aus dem Bereich der Entwicklung im Oracle-Umfeld mehr Raum bieten (PL/SQL, APEX, die eher datenbanknahe Programmierung etc.), während die SIG Java das Thema „Java“ in seiner vollen Breite behandelt. Damit wird der stetig gewachsenen Bedeutung dieses Bereichs in der Oracle-Welt Rechnung getragen – einem Trend, der schon viele Jahre anhält und durch die geplante Akquisition von Sun durch Oracle noch einmal deutlich verstärkt wird.

Die Ausgründung der neuen SIG Java erfolgt auf einer gemeinsamen Veranstaltung von SIG Development und SIG Java am 29. April 2010 in Nürnberg – sicherlich nicht die letze gemeinsame Veranstaltung, denn die Themen lassen sich nicht vollständig trennen. Da die DOAG mit den deutschen Java User Groups kooperiert (siehe Seite 61), werden die Veranstaltungen der SIG Java nach Möglichkeit gemeinsam mit den jeweiligen lokalen Gruppen geplant. Kooperationspartner für die Gründungsveranstaltung wird entsprechend die Java User Group Nürnberg sein.

Thema der ersten Veranstaltung sind die Möglichkeiten des „Rapid Application Development“ mit Java. Dabei wird es vorrangig um Oracles Application Development Framework (ADF) gehen, aber auch andere Werkzeuge und Frameworks sollen betrachtet werden. Das Treffen wird zurzeit noch geplant – der aktuelle Planungsstand mit Agenda ist unter www.doag.org/termine einzusehen.

Ihre Fragen, Themenvorschläge und Vortragsangebote etc. richten Sie bitte an sig-java@doag.org.

**Kontakt**:
Andreas Badelt
sig-java@doag.org

# Datenflohmarkt – oder Gedanken zum Datenschutz in der IT

Volker Ricke, POINT. Consulting GmbH

**Gedanken über Datensicherheit drehen sich häufig um vergleichsweise komplexe Lösungen wie Verschlüsselung, Label Security oder Auditing. Dabei fängt der Datenschutz bereits viel früher an.**

„Die Diskette mit den gestohlenen Kundendaten konnte von der Polizei sichergestellt werden" – so ähnlich verbreitete der Tagesschau-Sprecher die vermeintlich gute Nachricht. Das Schmunzeln über den Begriff „Diskette" wich schnell einem unguten Gefühl. Denn erneut waren sensible Kundendaten abhanden gekommen und in dunkle Kanäle versickert. Die Hälfte der Kundendaten eines deutschen Mobilfunkanbieters, echte Testdaten eines Finanzdienstleisters, Kreditkartendaten von in Spanien getätigten Umsätzen – nur drei prominente Beispiele aus den letzten zwei Jahren. Sind sensible Daten zur unkontrollierbaren Handelsware geworden? Was können wir dagegen tun?

## Fluch der Vernetzung

Die Wirtschaftswelt wurde in den letzten Jahren immer mehr vernetzt, Prozesse optimiert und neu auf der ganzen Welt verteilt, zum Teil aus den Unternehmen ausgelagert. Auch die IT-Prozesse folgen der Vorgehensweise und stellen die benötigten Daten und Funktionen zur Verfügung. Hat aber jetzt noch jemand den Überblick, welche Daten wohin geflossen sind?

Diesen Überblick zu gewinnen kann bereits schwer sein, wenn die Daten im eigenen Unternehmen bleiben – verschiedene operative Systeme werden möglicherweise über Schnittstellen versorgt, Daten in ein Data Warehouse übertragen, Reports erstellt, Excel-Exporte in Fachabteilungen weiterverarbeitet. Vielleicht hat auch dort jemand schnell eine Access-Datenbank-Anwendung erstellt, weil die IT-Abteilung kurzfristig keine Kapazität hatte, eine Anforderung umzusetzen? Auch Testsysteme werden vielleicht regelmäßig mit Echtdaten versorgt, die nicht ausreichend anonymisiert werden. Dem Außendienst wird der Kundenstamm auf Notebooks repliziert, das eine oder andere Notebook gerät vielleicht mal abhanden. Externe Vertriebs- und Servicepartner wurden beauftragt und Schnittstellen zu ihnen aufgebaut. Werbeagenturen erhalten zur Durchführung von Marketing-Kampagnen die benötigten Kundendaten.

Den Gesamt-Überblick über diese Vielfalt von Prozessen und zum Teil redundanten Daten kann niemand ernsthaft haben. Wer mag da noch die Hand dafür ins Feuer legen, dass die beteiligten Menschen und Systeme gleichermaßen verlässlich sind?

## Schritte zum Überblick über die IT-Landschaft

Am Anfang sollte eine Bestandsaufnahme stehen. Folgenden Fragen helfen dabei:

- Welche IT-Systeme sind im Einsatz – Standard-Produkte ebenso wie Eigenentwicklungen?
- Welche „Parallel-IT" mit eigenentwickelten Spreadsheets, Serienbriefen, Makros und Desktop-Datenbanken gibt es in den Fachabteilungen?
- Welche Schnittstellen gibt es zwischen den Systemen? Dazu gehören zum Beispiel auch Excel-Exporte aus Fachanwendungen.
- Wie sicher sind die Übertragungswege beim Datenaustausch zwischen Systemen über Netzwerk oder Datenträger?
- Welche externen Partnersysteme sind eingebunden (B2B) und welche Informationen werden mit ihnen ausgetauscht?
- Wie werden die Daten in externen Partnersystemen geschützt?
- Wer hat Zugang zu den Anwendungen und Daten?
- Wie schützenswert oder vertraulich sind die jeweiligen Daten?
- Welche Prozesse gibt es, um Benutzer und Privilegien einzurichten und zu löschen? Hat wirklich jeder Benutzer nur die Rechte, die er für seine Aufgabe benötigt? In vielen Unternehmen haben Auszubildende nach zwei Jahren die meisten Privilegien, weil Rechte erteilt, aber nach dem Wechsel der Abteilung nicht mehr entzogen wurden!
- Welche Prozesse gibt es, um Daten in den angeschlossenen Systemen aktuell zu halten? Werden insbesondere Löschungen sauber übertragen?
- Welche Produktivdaten aus gut geschützten operativen Systemen werden in weniger gut geschützte Auswertungs- und Testsysteme übertragen? Erfolgt dabei eine Anonymisierung?
- Welche Kopien von Daten sind zum Beispiel auf Notebooks oder in E-Mails gespeichert?
- Wie sind die Datensicherungen geschützt?
- Wie ist das Netzwerk geschützt? Welche Zugänge gibt es aus dem Internet?
- Wie abgesichert sind die Arbeitsplatzrechner (USB-Anschlüsse, Internet-Zugang)?
- etc.

Die Bestandsaufnahme deckt meist bereits die ersten Schwachpunkte auf.

### Menschliche Fehler und schwarze Schafe

Natürlich dürfen wir nicht vergessen, dass – zum Glück – immer noch Menschen an den Systemen sitzen. Wer arbeitet, macht auch mal Fehler. Eine Aufgabe der IT-Systeme sollte allerdings sein, vor gravierenden Fehlern zu schützen. Eine umfangreichende Schulung hilft ebenfalls, Schaden zu vermeiden. Der Autor erinnert sich an die Erzählung eines Bekannten, der per E-Mail ein in Word geschriebenes Angebot eines Lieferanten erhalten hatte – und mit „Änderungen nachverfolgen" die bisherigen Adressaten ähnlicher Angebote und die ihnen eingeräumten Konditionen einblenden konnte, weil der Absender nicht wusste, was Word alles in seinem Dokument mitprotokolliert hatte!

Ein angemessenes Rechte-Konzept kann Anwender ebenfalls davor schützen, ungewollt Schaden anzurichten. Dass ein schwarzes Schaf unter den Mitarbeitern Daten abzieht, kann es zwar nicht völlig verhindern, dafür aber die möglichen Auswirkungen verringern. Der Datendieb ist heute in vielen Fällen nicht der geheimnisvolle Hacker von außen, sondern jemand, der legalen Zugriff auf die Systeme hat.

### Sicher mit Oracle-Optionen?

Oracle bietet eine ganze Anzahl von Sicherheits-Optionen für die Enterprise Edition ihrer Datenbank an – Begriffe wie Verschlüsselung, Label Security, Single-Sign-On oder Data-Masking-Pack fallen in diesem Zusammenhang. In vielen Gesprächen mit Oracle-Anwendern ist dem Autor allerdings aufgefallen, dass bereits mit einfachen Mitteln damit begonnen werden kann, die Sicherheit erheblich zu erhöhen.

Wenn man den Standardfall betrachtet, dass sich eine Applikation mit einem normalen Datenbank-User an der Datenbank anmeldet, sollte man folgende Punkte überprüfen:

- Die Datenbank-Objekte sollten in einem Schema angelegt sein, in dem niemand direkt als Benutzer arbeitet.
- Für die unterschiedlichen Zugriffsprofile sollten Datenbank-Rollen definiert sein, an die die erforderlichen Objekt-Privilegien erteilt werden – in einem einfachen Fall Rollen für „Admin" , „Sachbearbeiter" und „Nur-Lese-Zugriff".
- Die Datenbank-User für den Zugriff mit der Applikation sollten die minimal erforderlichen System-Privilegien besitzen (zum Beispiel CREATE / ALTER SESSION).
- Objekt-Privilegien sollte man den Datenbank-Usern nur über die oben erwähnten Rollen zugeweisen. Kennen die Anwender die Login-Daten, so verhindern zur Laufzeit freigeschalteten Rollen, dass ungefiltert mit anderen Werkzeugen – zum Beispiel Excel – auf die Daten zugegriffen wird.
- Datenbank-Packages sollten gegebenenfalls in Packages für Nur-Lese- und Schreib-Zugriffe getrennt werden, um auch einem reinen Auswertungs-User die Verwendung von Single Row Functions oder Table Functions im SQL zu ermöglichen.
- Zum Protokollieren von Schreib-Zugriffen sollten Audit-Spalten vorhanden sein, die über Datenbank-Trigger gefüllt werden.
- Wenn kein Single-Sign-On zum Einsatz kommt, sollten Prozesse zum Sperren inaktiver Benutzer vorhanden sein.

Es lohnt sich, auch die folgenden Punkte zu überprüfen:

- Sind vertrauliche Daten unerwünscht über Database-Links lesbar?
- Wurden Privilegien an „Public" erteilt?
- Welche Regeln existieren für die Vergabe und Änderung von Datenbank-Passwörtern? Wer kennt die Passwörter von Batch-Usern? Lassen sich diese problemlos ändern oder sind sie fest in Programmen verdrahtet?
- Gibt es entsperrte Datenbank-User mit Standard-Passwörtern? Nicht nur „Scott/Tiger" ist bekannt ...
- Wer hat Betriebssystem-Zugriff auf den Datenbank-Server?
- Wie gut sind Verzeichnisse für den Datenaustausch geschützt?

Nachdem die grundlegenden Punkte abgehakt sind, wird die Verwendung weitergehender Funktionalitäten und Optionen der Oracle Datenbank in Betracht gezogen.

### Anmerkungen zu einigen Oracle-Sicherheitsfunktionen

Oracle bietet sehr mächtige Funktionen, um die Datensicherheit weiter zu erhöhen. Dazu zählen:

- Die „Virtual Private Database" (VPD) ist eine kostengünstige Möglichkeit, auch nachträglich Zugriffsbeschränkungen zu integrieren. Das Unternehmen des Autors hat sie genutzt, um eine bestehende Applikation mandantenfähig zu machen. Die größte Änderung in der Applikation bestand darin, einen Datenbank-Kontext für den Mandanten zu setzen. Die relevanten Tabellen wurden mit einer Mandanten-Spalte versehen und über eine Policy Function geschützt, die den Kontext des aktuellen Benutzers auswertet und mit dem Mandanten der jeweiligen Zeile abgleicht. Einmal aufgesetzt, sind die Zugriffe für die Applikationen transparent – das „Select" wird implizit um eine „Where"-Klausel erweitert.
- „Label Security" ist eine mächtige Lösung, wenn man Datenzugriffe im Unternehmen über Hierarchien, Vertraulichkeitsstufen und Unternehmensbereiche abbilden kann. Das Produkt setzt auf der Virtual Private Database auf und erweitert sie um eine Zugriffsschicht mit einem Label aus Vertraulichkeitsstufe (Level), hierarchischer Gruppenstruktur (Group) und Unternehmensbereich (Compartment). Es kann Lese- und Schreibzugriffe unterschiedlich behandeln und gleicht auf Zeilenebene das Label der Daten mit dem aktiven Label des Benutzers ab. Zusätzlich lassen sich auch die aus der VPD bekannten Policy-Funktionen

verwenden. Schwächen zeigt die Label Security, wenn es Grauzonen gibt – etwa Übergangsphasen oder Vertretungsphasen, in denen hierarchieübergreifend Daten verwendet werden sollen.

- Das Data-Masking-Pack enthält viele interessante Funktionen, um aus Produktivdaten Testdaten zu erstellen, ohne die Aussagekraft der Daten zu verwässern. Werte können so ersetzt werden, dass optisch (Namen) und wertmäßig (beispielsweise Datums- und Zahlenbereiche oder formal richtige Kreditkartennummern) plausible Daten herauskommen.

**Der Beauftragte für den Datenschutz – mit einem Bein im Gefängnis?**

Über die Verarbeitung personenbezogener Daten (also Informationen über natürliche Personen) muss wenigstens einer im Unternehmen den Überblick haben, nämlich der Beauftragte für den Datenschutz nach § 4f Bundesdatenschutzgesetz (BDSG). Er ist zur Auskunft darüber verpflichtet, welche Daten über einen Betroffenen gespeichert sind, und trägt die Verantwortung dafür, dass personengebundene Daten nach Aufforderungen des Betroffenen berichtigt, gesperrt oder gelöscht werden. Erhält der Betroffene dennoch im Rahmen einer späteren Werbekampagne unerwünschte Werbung, ist das nicht nur peinlich – Ordnungswidrigkeiten und Strafvorschriften sind ebenfalls im Bundesdatenschutzgesetz geregelt.

## Fazit

Das Datenschutz-Thema ist sehr vielschichtig und darf nicht unterschätzt werden. Mit Betriebssystem- und Datenbank-Bordmitteln kann man aber bereits eine gesunde Basis legen, um später unliebsame Überraschungen zu vermeiden.

**Kontakt:**
Volker Ricke
volker.ricke@point-gmbh.com

*Michael Paege, Stefan Kinnen, Ralf Kölling, Dr. Dietmar Neugebauer, Franz Hüll, Christian Trieb, Fried Saacke und Michael Pfautz (von links)*

# Neuer DOAG-Vorstand und dessen Ziele für 2010

Die Mitglieder der DOAG haben auf ihrer Versammlung am 17. November 2009 turnusgemäß einen neuen Vorstand für die nächsten zwei Jahre gewählt:

- Dr. Dietmar Neugebauer (Vorsitzender)
- Stefan Kinnen (stellv. Vorsitzender)
- Christian Trieb (stellv. Vorsitzender)
- Franz Hüll
- Ralf Kölling
- Michael Paege
- Michael Pfautz
- Fried Saacke

Im Rahmen der Mitgliederversammlung am 17. November 2009 hat der neue und alte Vorstandsvorsitzende Dr. Dietmar Neugebauer den Mitgliedern die Ziele der DOAG für das kommende Jahr vorgestellt. Ein Schwerpunkt ist die geplante Sun-Integration durch Oracle. Die DOAG ist an der Gründung eines Interessenverbunds beteiligt (siehe Seite 61). Auch eine SIG Java ist geplant (siehe Seite 66).

Ein zweiter Schwerpunkt ist die Motivation der Mitglieder zu erweiterter Aktivität auf der DOAG-Plattform, insbesondere die Verstärkung der Beiratsarbeit sowie der Ausbau des Networkings in den Regionalgruppen. Hinzu kommen grenzübergreifende Aktivitäten in Deutschland, Österreich und der Schweiz für den Bereich „Applications“. Dazu findet am 24. und 25. März 2010 in Mainz erstmals die DOAG 2010 Applications statt. Hinsichtlich der Interessenvertretung gegenüber dem Hersteller soll der konstruktiv-kritische Dialog mit Oracle auf die EMEA-Ebene ausgeweitet werden, um entsprechende Resonanz zu erzielen.

Fazit: Dr. Dietmar Neugebauer ist zuversichtlich, mit dem neuen Vorstand die genannten Ziele im nächsten Jahr umsetzen zu können.

Weitere Informationen unter www.doag.org/doagev/adressen/vorstand/

# Der BI Publisher in einer klassischen Druck-Umgebung

Heinz-Michael Anders, InfoSys GmbH

**Dieser Artikel ist ein Erfahrungsbericht über den Einsatz und die Integration des BI Publishers in ein klassisches Druck-Umfeld. Dabei kommen die Integrationsmöglichkeiten in PL/SQL-Datenbankprozeduren ebenso zur Sprache wie die Einschränkungen, die man bei höheren Lastanforderungen berücksichtigen muss.**

Liest man Präsentationen und Einsatz-Szenarien über den BI Publisher, findet man viel über dessen Integrationsmöglichkeiten in verschiedene moderne Umgebungen wie BI Presentation Services, BI Discoverer oder Oracle Forms. Es gibt allerdings noch eine Vielzahl von existierenden Anwendungen, die standalone oder in älteren System-Umgebungen ablaufen. Wenn hier der BI Publisher zum Einsatz kommen soll, ist es hilfreich, ein Vorgehensmodell zu entwickeln, das die Vorteile des Publishers, wie die Separation von Daten und Layout, zur Verfügung stellt, ohne die heute übliche System-Infrastruktur vorauszusetzen.

Dazu ein konkreter Fall: In einem großen Distributionszentrum eines internationalen Konsumgüterherstellers sind verschiedene Anwendungssysteme im Einsatz. Neben Forms und Reports Version 4.5 und 6*i* kommen auch C- und Java-Programme zum Einsatz. Für die Datenhaltung gibt es verschiedene Oracle Datenbanken der Version 10.2.0.3. Täglich werden mehr als 50 000 verschiedene Dokumente, Label und Etiketten auf Papier erzeugt. Neben den üblichen Office-Druckern sind etliche proprietäre Etikettendrucker im Einsatz, die man über spezielle Protokolle ansteuert.

## Zielsetzung

Im Frühjahr 2009 wurde die Entscheidung getroffen, für all diese Ausdrucke ein zentrales System (Druckserver) einzuführen, das sämtliche Drucker mit Druckaufträgen versorgt. Ein wichtiges Ziel aus Sicht des Kunden war dabei die Trennung der Datenselektion vom Layout, um Änderungen eigenständig ohne Programmänderungen durchführen zu können. Aus den eingesetzten Anwendungen sollen danach keine Drucke mehr direkt an einen Drucker erfolgen. Die vorhandenen 2.5-Reports werden abgelöst. C-Programme, die über Pro*C Daten aus der Datenbank lesen und im PCL- oder ASCII-Format direkt zum Drucker senden, werden durch Java-Programme ersetzt. Alle Programm-Module selektieren nur noch die Daten und stellen sie über eine Schnittstelle dem Druckserver zur Verfügung. Das bisherige Layout der Druckausgaben soll dabei möglichst erhalten werden.

Zusätzlich müssen bestimmte Dokumente einem Archivsystem zugeführt werden, inklusive Verschlagwortung und Indizierung. Log- und Status-Informationen über die angefertigten Drucke sollen ebenfalls erstellt werden und grundsätzlich vom Anwendungssystem über die Schnittstelle abrufbar sein. Die Verarbeitung der Aufträge muss zeitnah erfolgen, in der Regel beginnt der Ausdruck innerhalb weniger Sekunden nach dem Anlegen des Auftrags.

## Realisierung

Vom Unternehmen des Autors fiel die Entscheidung – aufgrund der hohen Anforderungen an den Durchsatz und die kurze Reaktionszeit –, die Sammlung und Aufbereitung der Druckaufträge in den einzelnen Datenbanken unter PL/SQL durchzuführen. Für das Layout wird der BI Publisher Version 10.1.3.4 auf einem Tomcat-Webserver verwendet. Beim Anwendungsdesign wurde besonderes Augenmerk darauf

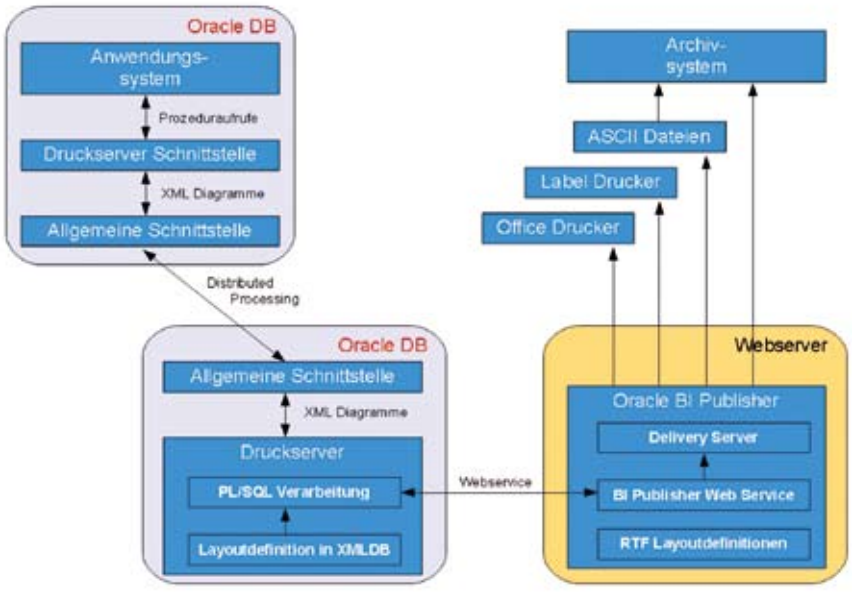



*Abbildung 1: Schematische Systemkopplung*

gelegt, die Verarbeitung möglichst innerhalb der Datenbank zu realisieren und den BI Publisher nur für die reine Layout-Aufbereitung und die Druckeransteuerung zu verwenden.

Die Programmierung des zentralen Druckservers ist in PL/SQL realisiert, das System läuft in einem eigenen Schema auf einer Datenbank-Instanz. Allen Anwendungen, die über dieses System drucken wollen, wird eine Schnittstelle zum Druckserver auf der lokalen Datenbankinstanz der jeweiligen Anwendung bereitgestellt. Diese Daten werden als XML-Diagramme an eine nachfolgende Schnittstelle übertragen, die asynchron die Daten zwischen Datenbankinstanzen austauscht. Das Anwendungssystem benötigt keinerlei Informationen über Ort und Name der Datenbank des Druckservers. Das Routing dieser allgemeinen Schnittstelle wird über Tabellen administriert und ist dadurch von der jeweiligen Anwendung völlig unabhängig.

### Schnittstelle zum Anwendungssystem

Die Schnittstelle zum Druckserver besteht aus einem PL/SQL-Package und Tabellen, die lokal in einem dedizierten Schema installiert sind. Ziel war es, diese Schnittstelle einerseits möglichst einfach zu gestalten, um den Umstellungsaufwand für die Programme zu minimieren, andererseits aber alle geforderten Struktur-Informationen (zum Beispiel Gruppenwechsel), die für das Layout relevant sind, zu integrieren. Alle Daten eines Druck-Aufrufs werden zu einem XML-Diagramm zusammengefasst und als XML transportiert. Die Funktionen und Prozeduren der Schnittstelle bereiten aus den Daten intern das XML auf; das Anwendungssystem benötigt keine Information über den XML- Aufbau, um die Schnittstelle zu nutzen. Der BI Publisher stellt eine Vielzahl von Optionen bereit, die die Bearbeitung eines Aufrufs (Request) steuern (EmailDeliveryOption, FaxDeliveryOption etc.). All diese Optionen lassen sich in der Schnittstelle spezifizieren. Durch die Wahl geeigneter Default-Einstellungen sind nur wenige Optionen für einen Aufruf explizit zu setzen (siehe Listing 1).

```
<prn application="LVS" id="3160" type="epl_label">
  <request>
    <cronExpression/>
    <deliveryRequest>
      <emailOption>
        <emailBCC/>
        <emailBody>Das Artikel-Label wurde gedruckt</emailBody>
        <emailCC/>
        <emailFrom>bipublisher@infosys-gmbh.de</emailFrom>
        <emailReplyTo/>
        <emailSubject>Artikel-Label erstellt</emailSubject>
        <emailTo>oracle@infosys-gmbh.de</emailTo>
      </emailOption>
    </deliveryRequest>
    <endDate/>
    <jobCalendar>GREGORIAN</jobCalendar>
    <useUTF8Option>true</useUTF8Option>
    ...
  </request>
  ...
</prn>
```

*Listing 1: XML-Fragment eines Requests*

Die Inhaltsdaten eines Reports oder Etiketts werden ebenfalls als XML abgelegt. Die Struktur dieses XML-Fragments wird stark von der Verwendung dieser Daten im BI Publisher bestimmt. Das Layout eines Reports ist in einem RTF-File festgelegt, das über Microsoft Word erstellt wird. Der Zugriff auf die XML-Daten ist dabei besonders einfach und anwenderfreundlich, wenn die einzelnen Tags im XML individuelle Namen besitzen. Dies hat leider zur Folge, dass nicht ohne Weiteres eine XML-Schema-Definition für diese XML erzeugt werden kann, da die Element-Namen beliebig wählbar sind. Um die Verarbeitungszeit möglichst gering zu halten, hat man sich entschieden, das XML nur als CLOB in der Datenbank zu speichern. Die Größe des XMLs eines üblichen Dokuments ist moderat, so dass die fehlende Option des strukturierten Speicherns nicht relevant ist (siehe Listing 2).

Zusätzlich zu den Dokumenten, die über die Layout-Informationen aus RTF-Files im Publisher aufbereitet werden, generiert das System auch Etiketten und Label. Über die Schnittstelle kommen auch hier nur die Nutzdaten zum Versand. Diese werden allerdings im Druckserver auf der Datenbank mit den Layout-Definitionen zusammengeführt, bevor sie im Publisher als reiner ASCII-Output ausgegeben werden. Die Layout-Definitionen für diese Etiketten und Label sind in speziellen, druckerspezifischen Formaten (ZPL oder EPL) als ASCII-File auf der Datenbank abgelegt. Für die Speicherung dieser Dateien kommt das Oracle Standardpackage XMLDB zum Einsatz.

Die Listings 3 bis 5 zeigen exemplarisch die Label-Daten, Layout-Definitionen und den daraus generierten ASCII-Output für ein Etikett.

### Druckserver / BI Publisher

Die hohen Anforderungen an den Durchsatz von Ausdrucken werden durch eine möglichst hohe Parallelisierung der Prozesse erfüllt, sowohl in der Datenbank als auch im Publisher. Die eingehenden Aufträge laufen in einer Eingangs-Queue auf. Über den DBMS_SCHEDULER werden eigenständige Sitzungen auf der Datenbank angelegt und gestartet, die die Aufbereitung der Daten für einen einzelnen Request durchführen und die Kommunikation mit dem BI Publisher abwickeln. Durch diese Parallelisierung wurde eine sehr geringe Reaktionszeit des Gesamtsystems erreicht.

Die Kommunikation mit dem BI Publisher, der auf einem Tomcat-Webserver unter Windows läuft, erfolgt über die Webservice-Schnittstelle des Publishers, die direkt aus der Datenbank heraus

```
<reports>
<reports>
 <report id="0" layout="LS">
  <List_Kopf gid="0">
   <Kopf id="0">
    <List_Kopftext_allgemein gid="1">
     <Kopftext_allgemein id="0">
      <F_AUFNR type="number">400404040</F_AUFNR>
      <F_TERMIN_TAG type="date">2009-12-03</F_TERMIN_TAG>
      <F_LDT type="datetime">2009-12-03T09:20:06.846263000</F_LDT>
      <F_LOGO_ZEILE_1>InfoSys GmbH</F_LOGO_ZEILE_1>
      <F_LOGO_ZEILE_2>Holsteiner Str. 15</F_LOGO_ZEILE_2>
      <F_LOGO_ZEILE_3>24768 Rendsburg</F_LOGO_ZEILE_3>
      <F_LOGO_ZEILE_4>Tel: +49 4331 58010</F_LOGO_ZEILE_4>
     ...
     </Kopftext_Position>
    </List_Kopftext_Position>
   </Kopf>
  </List_Kopf>
  ...
  <List_Positionen gid="4">
   <Positionen id="0">
    <List_Positionsdaten gid="5">
     <Positionsdaten id="0">
      <F_ART_NR>47123760</F_ART_NR>
      <F_AUFPOS>0001</F_AUFPOS>
      <F_ART_NR_BEZ>Leiterplatte, r 2 &lt;LED&gt;</F_ART_NR_BEZ>
      <F_KUNDEN_ARTIKELNR>040506</F_KUNDEN_ARTIKELNR>
      <F_CHARGE>Charge 5</F_CHARGE>
      <F_MHD>31.12.2010</F_MHD>
      <F_EAN_DISPATCH>1234567896337</F_EAN_DISPATCH>
      ...
     </Positionsdaten>
    </List_Positionsdaten>
   </Positionen>
  </List_Positionen>
  ...
  <List_Spediteurkopf gid="9">
   <Spediteurkopf>
   ...
   </Spediteurkopf>
  </List_Spediteurkopf>
  ...
 </report>
</reports>
```

*Listing 2: Beispiel XML-Fragment von Reportdaten*

```
  <reports>
    <report id="0" layout="Artikellabel">
      <T_ARTNO>Art-No:</T_ARTNO>
      <F_ART_NR>1234567</F_ART_NR>
      <F_BEZ_TEXT_D>Leiterplatte, r 2 &lt;LED&gt;</F_BEZ_
TEXT_D>
      <F_IST_MENGE_PICK>14</F_IST_MENGE_PICK>
      <T_PCS>Pcs:</T_PCS>
      <F_EAN_NR>1234567890123</F_EAN_NR>
    </report>
  </reports>
```

*Listing 3: Beispiel XML-Label-Daten*

```
N
Q200,1
R000,0
q320
O
S2
D7
ZT
JF
A010,085,0,1,1,1,N,"T_ARTNO"
A079,080,0,3,1,1,N,"F_ART_NR"
A224,085,0,1,1,1,N,"T_PCS"
A260,080,0,3,1,1,N,"F_IST_MEN-
GE_PICK"
A010,115,0,3,1,1,N,"F_BEZ_
TEXT_D"
B010,000,0,E30,2,3,60,B,"F_EAN_
NR"
GG235,000,"PCX1"
P1
```

*Listing 4: Beispiel Label-Layout-Definition*

```
N
Q200,1
R000,0
q320
O
S2
D7
ZT
JF
A010,085,0,1,1,1,N,"Art-No:"
A079,080,0,3,1,1,N,"1234567"
A224,085,0,1,1,1,N,"Pcs:"
A260,080,0,3,1,1,N,"14"
A010,115,0,3,1,1,N,"Leiterplatt
e, r 2 <LED>"
B010,000,0,E30,2,3,60,B,"123456
7890123"
GG235,000,"PCX1"
P1
```

*Listing 5: Output-Datei für den Drucker*

angesprochen wird. Das Repository des Publishers kann wahlweise im Filesystem des Webservers oder unter XMLDB in einer Oracle Datenbank angelegt sein. Im Vorfeld durchgeführte Lasttests haben ergeben, dass es zu erheblichen Performance-Einbrüchen kommt, wenn man das Repository unter XMLDB hält. Der Prozessor des Datenbank-Rechners wird erheblich belastet und die Publisher-Prozesse warten auf die Bearbeitungen in der Datenbank. Der Grund sind unter anderem die Output-Files, die in PDF- oder ASCII-Format vorliegen und ebenfalls im Repository abgelegt sind. Deshalb wurde das Filesystem für das Repository gewählt.

### Verarbeitung eines Requests

Die Verarbeitung eines Requests wird durch dedizierte Jobs in der Datenbank durchgeführt, die der DBMS_SCHEDU-

LER automatisch startet. Die Maximalzahl der Jobs, die parallel Requests abarbeiten können, ist in den Administrations-Tabellen festgelegt.

Jeder dieser Jobs scannt die Input-Queue, wählt den nächsten Request aus, markiert diesen als „in Bearbeitung“ und startet die weitere Durchführung. Jeder Job prüft nach der Ausführung erneut die Input-Queue; sobald keine Aufträge mehr vorhanden sind, beendet sich der Job.

Für Reports vom Typ „Dokument“ werden lediglich die in dem Request spezifizierten Optionen ausgelesen und der Report über den Webservice aktiviert. Falls es sich nicht um Dokumente, sondern um Label handelt, werden die im Request mitgelieferten Daten mit einem über XMLDB in der Datenbank gespeicherten Layout verknüpft und verarbeitet. Erst danach wird der Auftrag an den Publisher gesandt.

Falls man den Ausdruck eines Dokuments archivieren möchte, wird die LocalDelivery-Option des Publishers genutzt, um das Ausgabe-PDF im Filesystem abzulegen. In diesem Fall wird auch ein weiterer Report im Publisher gestartet, der das Indexfile erzeugt.

### RTF-Templates für Dokumente

Der Ausdruck von Dokumenten erfolgt grundsätzlich im PDF-Format auf Basis von RTF-Templates. Es werden für einige Reports auch Barcodes im Code128 erzeugt. Die Versorgung mit Daten erfolgt für jeden Report über eine standardisierte SQL-Query. Grundsätzlich werden nur zwei Aufrufparameter – eine ID und der Name des Anwendungssystems – verwendet, alle anderen Daten transportiert man über das XML (siehe Abbildung 2).

### RTF-Templates für Label und Etiketten

Alle Label und Etiketten werden im ASCII-Format auf der Basis eines einzigen eText-Templates verarbeitet, das die Daten lediglich zeilenweise ausgibt. Die Datenquelle ist auch in diesem Fall die standardisierte SQL-Query (siehe Abbildung 3).

### Anbindung Archiv-Server

Zusätzlich zu dem eigentlichen Ausdruck des Dokuments kann man das erzeugte PDF-File in einem Archiv-Server ablegen. Neben dem File ist dafür eine weitere ASCII-Indexdatei zu erstellen, auch hierfür stehen ein eText-Template und ein Report zur Verfügung. Diese Dateien werden im Filesystem abgelegt (LocalDestination) und von dort durch einfache Copy-Befehle des Betriebssystems weiter transportiert. Der Aufruf dieses zusätzlichen Reports und der Inhalt dieser Indexdatei werden in der Datenbank von den durch DBMS_SCHEDULER erzeugten Jobs generiert. Der BI Publisher kommt nur für die Bearbeitung des Requests zum Einsatz.

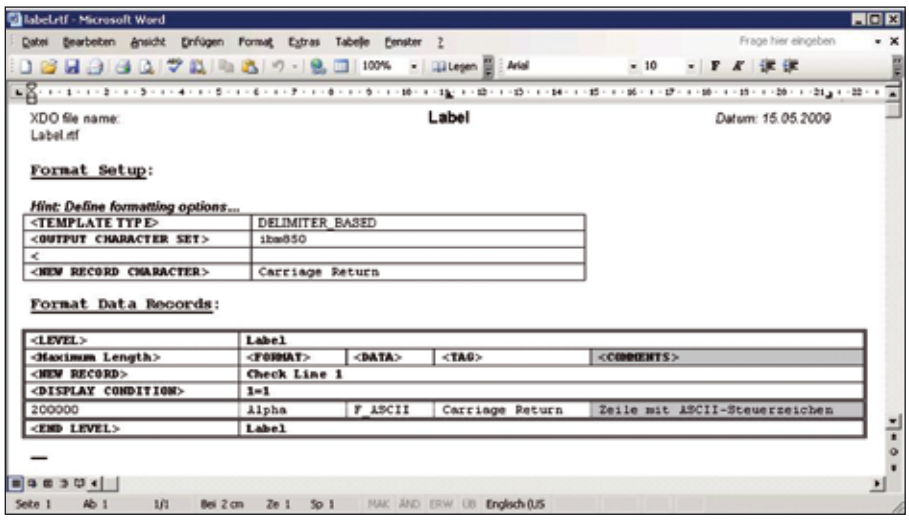



*Abbildung 3: eText-Template*

Für jeden einzelnen Auftrag wird vom Druckserver ein Log erstellt. In den Administrations-Tabellen kann man den Informationsumfang der Logdaten spezifizieren. Der aktuelle Bearbeitungsstand jedes Requests steht in Form von Status-Informationen zur Verfügung. Über die allgemeine Schnittstelle werden diese Daten zurück zum auftraggebenden System transferiert und sind dort lokal abrufbar.

## Fazit

Das Projekt wurde innerhalb von vier Monaten realisiert und läuft seit September 2009 produktiv. Derzeit erzeugt das System täglich rund 3000 Dokumente und 7000 Label. Die Latenzzeit vom Abschicken des Requests in der Anwendung bis zum Start des Ausdrucks am Drucker liegt unter fünf Sekunden, bei Labeln unter drei Sekunden. Die Reaktionszeiten und der Durchsatz sind besser als in der abgelösten Umgebung.

Der eingeschlagene Weg, den BI Publisher aus der Oracle Datenbank heraus anzusteuern, hat sich bewährt. Der Webserver, auf dem der Publisher installiert ist, braucht nicht im lokalen Netz unternehmensweit sichtbar zu sein, es genügt, lediglich den Zugriff vom Datenbank-Rechner zu ermöglichen. Dadurch sind Sicherheitsaspekte in Umgebungen, die nicht auf eine Integration von Webservern ausgerichtet sind, leichter zu berücksichtigen.

Die Performance ist sehr gut, wenn durch die Einbettung des Publishers in der beschriebenen Art und Weise seine Leistungsfähigkeit genutzt werden kann. Anfängliche Befürchtungen wegen des Aufwands für die Adaption der vorhandenen Reports haben sich als unbegründet erwiesen. Hier konnten durch die Trennung von Daten und Layout für die einfachen Reportlayouts verstärkt die Fachabteilungen einbezogen werden. Dadurch lässt sich der BI Publisher auch in einer klassischen Druckumgebung wirtschaftlich implementieren sowie stabil und performant betreiben.

**Kontakt:**
Heinz-Michael Anders
hma@infosys-gmbh.de

## Tipps und Tricks aus Gerds Fundgrube

# Heute: Datenblöcke auf Basis von Updatable Views

Gerd Volberg, OPITZ CONSULTING GmbH

**Typischerweise werden Oracle Forms-Masken auf Basis von Views und Tabellen erzeugt. Wenn Tabellen benutzt werden, dann meistens, um Daten zu erfassen und zu ändern. Views werden häufig genutzt, um komplexe Datenstrukturen in tabellarischer Form darzustellen und beliebige Sortierungen und Filter anzubieten.**

Warum nutzt man nicht die Vorteile beider Techniken? Obwohl das so einfach klingt, wird es in der Praxis nur selten gemacht, weil man dabei ein paar Dinge beachten muss. Nehmen wir als Beispiel eine View namens EMP_V, die auf der EMP- und DEPT-Tabelle des Users Scott aufbaut:

```
CREATE OR REPLACE FORCE VIEW
EMP_V
(empno, ename, sal, deptno,
dname, loc) AS
SELECT emp.empno, emp.ename,
emp.sal, emp.deptno, dept.dna-
me, dept.loc
  FROM EMP, DEPT
 WHERE EMP.deptno = DEPT.dept-
no;
```

Diese View arbeitet auf zwei Tabellen, wobei DEPT als Lookup-Tabelle genutzt wird und die "key-preserved"-Spalten aus EMP für DML-Statements genutzt werden können. Daraus folgt:

- Alle Spalten der Tabelle EMP finden sich in der View wieder und sind änderbar
- Alle Zeilen der Tabelle EMP sind änderbar
- DNAME und LOC sind nicht änderbare Spalten (reine Lookup-Spalten)

Nutzt man diese View nun in einem Forms-Block, dann müssen die beiden Felder DNAME und LOC in den Attribut-Eigenschaften auf „Nur Abfrage = Ja" gesetzt werden, damit die Felder durch eine Query weiterhin gefüllt werden, jedoch von allen DML-Statements, die Forms generiert, ausgenommen sind.

| EMPNO | ENAME | SAL | DEPTNO | DNAME | LOC |
|---|---|---|---|---|---|
| 7369 | SMITH | 800 | 20 | RESEARCH | DALLAS |
| 7499 | ALLEN | 1600 | 30 | SALES | CHICAGO |
| 7521 | WARD | 1250 | 30 | SALES | CHICAGO |
| 7566 | JONES | 2975 | 20 | RESEARCH | DALLAS |
| 7654 | MARTIN | 1250 | 30 | SALES | CHICAGO |
| 7698 | BLAKE | 2850 | 30 | SALES | CHICAGO |
| 7782 | CLARK | 2450 | 10 | ACCOUNTING | NEW YORK |
| 7788 | SCOTT | 3000 | 20 | RESEARCH | DALLAS |
| 7839 | KING | 5000 | 10 | ACCOUNTING | NEW YORK |
| 7844 | TURNER | 1500 | 30 | SALES | CHICAGO |
| 7876 | ADAMS | 1100 | 20 | RESEARCH | DALLAS |
| 7900 | JAMES | 950 | 30 | SALES | CHICAGO |
| 7902 | FORD | 3000 | 20 | RESEARCH | DALLAS |
| 7934 | MILLER | 1300 | 10 | ACCOUNTING | NEW YORK |

*Abbildung 1: Daten der View EMP_V*

Diese Eigenschaft wird somit bei allen Lookup-Spalten immer auf „Ja" gesetzt. Danach ist der Forms-Block dann in der Lage, alle DML-Statements korrekt gegen die View abzusetzen.

**Kontakt:**
Gerd Volberg
talk2gerd.blogspot.com

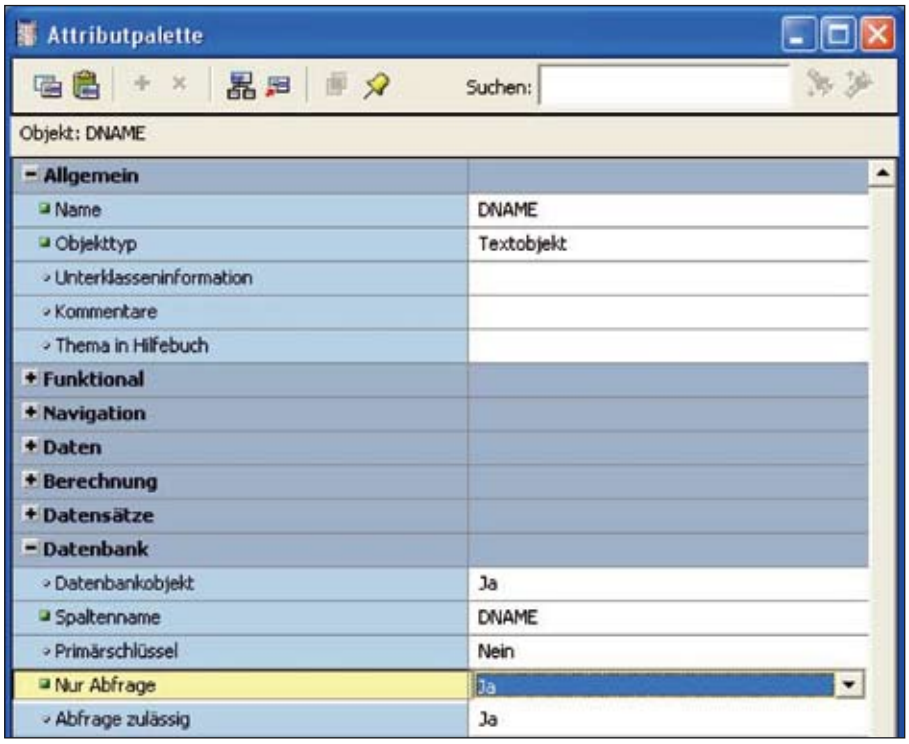



*Abbildung 2: Attribute des Feldes DNAME*

ALTRAN CIS

ALLOUT Security

CENTRIC

IBM

ORACLE

pdg PRIMUS DELPHI GROUP

pdv TAS

PROMATIS

# DOAG 2010 Applications Konferenz + Ausstellung

## DOAG verbindet: E-Business Suite, Agile, Siebel, JD Edwards
## Die Applications-Community wächst zusammen

Infos unter apps.doag2010.org

QUEST SOFTWARE Smart Systems Management

Riverland Your Oracle Solution Experts

Sun microsystems The Network is the Computer™

24. – 25. März 2010 in Mainz

2010 DOAG APPLICATIONS

# DOAG-Regionaltreffen in München

**Franz Hüll und Andreas Ströbel, Leiter der Regio München, führten Tagebuch.**

## Oktober 2009

Am 22. Oktober 2009 haben wir uns wieder einmal in den Räumen der Hauptverwaltung von Oracle in München getroffen. Zuerst warfen wir einen Blick in den prall gefüllten Einkaufskorb von Larry Ellison. Weit über 50 Zukäufe haben sich mittlerweile darin angesammelt und es werden immer noch mehr. Vier Mitarbeiter von Oracle (Wolfgang Meidenbauer, Maik Sandmann, Christian Patrascu und Ute Müller) stellten jeweils einige der Akquisitionen vor, erklärten, worum es sich bei den einzelnen Produkten handelt und was aus diesen in der Oracle-Welt geworden ist beziehungsweise noch werden soll.

Die Schwerpunkte waren in der Reihenfolge der Vortragenden:

- Content Management and Collaboration
- System Application Management (SAM)
- Identity- & Accessmanagement (I&AM)
- Enterprise Performance Management System (EPM)

Nach dieser Vielzahl von Informationen brauchten alle erst einmal eine kurze Pause.

Anschließend trat Dr. Günter Unbescheid von Database Consult GmbH ans Pult. Er berichtete über erste Erfahrungen und neue Features der Version 11g R2 und begann mit einem kurzen Überblick über Neuerungen und Verbesserungen. Für den weiteren Vortrag hatte er als Schwerpunkt „Edition-Based Redefinition“ gewählt – ein völlig neues Feature, das das Online-Upgrade von Datenbank-Applikationen erlauben soll, oder wie Dr. Unbescheid es ausdrückte: „Aktualisieren von Code-Objekten online“. Mit „Code-Objekten“ sind gemeint: Synonyme, Views, Functions, Procedures, Packages, Libraries und Trigger.

Das neue Feature ist für den Software-Entwickler sicher sehr interessant, aber wie so oft muss man wissen, was man tut. Es ist ein bisschen wie die „Operation am offenen Herzen“: Es funktioniert, ist komplex, aber man sollte genau planen und keine Fehler machen.

Beim anschließenden Buffet (Dank an Oracle) wurde dann in entspannter Atmosphäre das Ganze nochmal heftig diskutiert.

## Dezember 2009

Wieder zurück in der Fachhochschule München, ging es am 9. Dezember 2009 um die Themen „Sicheres Testdatenmanagement – Oracle Datamasking Best Practices” und “Oracle 11*g* New Security Features”. Michael Schellin von Trivadis befasste sich mit „Data Masking“. Häufig ist es so, dass in Test-Umgebungen Kopien von Produktiv-Datenbanken zum Einsatz kommen. Im Gegensatz zu den Produktiv-Datenbanken sind aber Test-Datenbanken meist leichter zugänglich und unterliegen leider nicht immer den gleichen Sicherheitsanforderungen. Um dem Datenmissbrauch hier vorzubeugen, ist die Anonymisierung von Testdaten ein probates, aber auch nicht leicht zu handhabendes Hilfsmittel. In diese Kerbe schlägt Oracle mit „Data Masking“. Auch wenn auf einer der Folien die berühmte „Eierlegende Wollmilchsau“ abgebildet war, zeigte Herr Schelling die Möglichkeiten, aber auch die Grenzen von Data Masking auf, wobei die positiven Aspekte deutlich überwiegen.

Der zweite Vortrag des Abends wurde von Sven Vetter, ebenfalls von Trivadis, gehalten. Neue Security-Features in 11*g* R2 standen auf der Agenda. Mit „Tablespace Encryption“ besteht jetzt eine Verschlüsselung der Daten auf Platte, die gegenüber der bisher verfügbaren Methode „Transparent Data Encryption (TDE)“ erhebliche Vorteile bietet. Damit wird der Diebstahl von Daten durch Kopieren von Datenbankfiles erheblich erschwert. Des Weiteren sprach Herr Vetter über folgende Features:

- Local Auto-Open Wallet
- „Secure by Default»
- Besserer Passwortschutz
- Ausführen von Scripts per External Tables

Der Referent zeigte im Detail, teilweise mit Code-Beispielen unterlegt, was im Einzelnen möglich ist. Dabei beschränkte er sich nicht auf explizite Security-Features, sondern ging auch auf Risiken ein, die durch andere Features (Beispiel Scripts) entstehen können.

Beide Vorträge gaben den Zuhörern einen sehr guten und informativen Überblick über die angesprochenen Themenbereiche.

Allen Referenten herzlichen Dank für die Zeit, die sie aufgewändet haben, um ihren Vortrag vorzubereiten und auf dem DOAG-Regionaltreffen zu präsentieren.

Wenn Sie Themenwünsche haben, die wir bei einem der nächsten Regio-Treffen behandeln sollten, dann schicken Sie uns bitte eine Mail an regio-muenchen@doag.org. Wir nehmen Ihre Anregungen gerne auf.

Die Vorträge der Regionaltreffen stehen auf dem DOAG-Server für Mitglieder und Teilnehmer zum Download bereit. Sie sind aber auch über die Referenten verfügbar.

**Kontakte:**
Franz Hüll
Andreas Ströbel
regio-muenchen@doag.org

# Oracle veröffentlicht Januar-CPU 2010

Franz Hüll, DOAG Competence Center Security

**Oracle hat am 12. Januar 2010 das Critical Patch Update (CPU) Januar 2010 veröffentlicht (Rev.1). Gegenüber dem letzten Quartal hat sich die Summe der behobenen Sicherheitslücken von 38 auf 24 verringert. Betroffen sind die Datenbank-Versionen 9.2.0.8 bis 11.1.0.7.**

Wiederum sind diesmal Lücken korrigiert, deren Risiko mit der höchsten Stufe „10" gekennzeichnet ist (CVE-2010-0071). Oracle schreibt, dass diese Lücke „remote" und ohne Authentifizierung ausgenutzt werden kann. Eingeschränkt wird diese Aussage durch den Hinweis „only for Windows", andere Plattformen werden bei „7.5" eingestuft. Eine weitere Lücke ist mit ebenfalls sehr hohen neun Punkten bewertet. Hier ist lediglich das Create-Session-Privileg erforderlich um diese auszunutzen.

In der BEA Product Suite wird auch eine Lücke mit der Einstufung „10" korrigiert. Hier weist Oracle darauf hin, dass diese hohe Einstufung nicht durch Oracle selber vorgenommen wurde, sondern von Sun MicroSystems stammt. Und wie immer: „Due to the threat posed by a successful attack, Oracle strongly recommends that customers apply fixes as soon as possible."

| Produkt | Anzahl Sicherheitslücken | Base Score (10=höchste Risikostufe) | |
|---|---|---|---|
| | | Min | Max |
| Oracle Database | 9 | 1.0 | 10.0 |
| Oracle Secure Backup | 1 | 10.0 | |
| Oracle Application Server | 3 | 4.3 | 5.0 |
| Oracle E-Business Suite | 3 | 4.3 | 6.4 |
| Oracle PeopleSoft Enterprise and JD Edwards EnterpriseOne | 1 | 4.9 | |
| BEA Product Suite | 5 | 4.3 | 10.0 |
| Oracle Primavera Product Suite | 2 | 4.0 | 4.0 |

In den Produkten Oracle Beehive, Oracle Collaboration Suite, Secure Enterprise Search und Oracle Enterprise Manager sind zwar direkt keine Sicherheitslücken behoben worden, Oracle empfiehlt aber wegen bestehender Abhängigkeiten von Produkten, die mit dem Januar CPU korrigiert werden, den CPU hier auch zu implementieren.

Die nächsten CPUs sind geplant für:

- 13. April 2010
- 13. Juli 2010
- 12. Oktober 2010
- 18. Januar 2011

### Weitere Informationen

Oracle Critical Patch Update Advisory – Januar 2010: http://www.oracle.com/technology/deploy/security/critical-patch-updates/cpujan2010.html

**Kontakt:**
Franz Hüll
securityfragen@doag.org

**Impressum**

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

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

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

**Chefin von Dienst (CvD)**:
Carmen Al-Youssef,
office@doag.org

**Gestaltung und Satz:**
Claudia Wagner,
DOAG Dienstleistungen GmbH

**Anzeigen**:
Carmen Al Youssef, office@doag.org
DOAG Dienstleistungen GmbH

Mediadaten und Preise finden Sie unter:
www.doag.org/publikationen/

**Druck**:
adame Advertising and Media GmbH Berlin
www.adame.de



# DOAG 2009 Konferenz + Ausstellung: Erfolg auf hohem Niveau

Die Deutsche ORACLE-Anwenderkonferenz ist seit 22 Jahren die beste Plattform zum Erfahrungsaustausch der Oracle-Anwender. Das im letzten Jahr neu eingeführte Konzept war auch für die DOAG 2009 Konferenz + Ausstellung wieder ein großer Erfolg. Mehr als 340 praxisnahe Vorträge und Keynotes von renommierten Referenten deckten an drei Konferenztagen alle Produktbereiche von Oracle mit entsprechenden Themen ab.

Wie bereits im letzten Jahr gab es einen kompletten Vortragsstream in englischer Sprache sowie die Simultanübersetzung ausgewählter deutschsprachiger Vorträge. Damit war die DOAG 2009 Konferenz +

Fotos: Katrin Heim


Ausstellung wieder die größte Veranstaltung zum Erfahrungsaustausch und für das Networking der Anwender aller Oracle-Produkte in Zentraleuropa.

„Wir vermitteln die besten und neuesten Informationen zu Technik, Applikation und Strategie, die die Anwender in Ihren Projekten zu Ihrem Vorteil umsetzen können“, so der DOAG-Vorstandsvorsitzende Dr. Dietmar Neugebauer. „Mit rund 2000 Teilnehmern konnten wir das hohe Niveau des Vorjahres sogar noch leicht übertreffen.“

Ein Highlight der DOAG 2009 Konferenz + Ausstellung war natürlich die neue Oracle Datenbank 11*g* R2. Andrew Mendelsohn, Senior Vice President und weltweit verantwortlich für die Oracle Server Technologies, reiste eigens aus den US-Headquarters an und eröffnete mit seiner Keynote eine Reihe von Vorträgen zu diesem aktuellen Thema.

Im Rahmen der Q&A-Session wurden in diesem Jahr mit dem deutschen Top-Management von Oracle das Thema „Chancen mit Oracle in der Wirtschaftskrise“ diskutiert. Dabei kamen auch aktuelle Brennpunkte wie „Oracle und Sun“ sowie „My Oracle Support“ zur Sprache.

„Neu in diesem Jahr war die DOAG Unconference, bei der Ablauf und Inhalt einer Session ausschließlich von den Teilnehmern bestimmt werden“, erläutert Fried Saacke, Vorstand und Geschäftsführer der DOAG.


„Auch der kiloschwere Tagungsband hat ausgedient. Stattdessen konnten sich die Teilnehmer in diesem Jahr ihren Vortragsband individuell zusammenstellen und per „Print on Demand“ ausdrucken lassen.“

Auf der eigenständigen DOAG 2009 Ausstellung stellten Oracle und Oracle-Partner aus dem deutschsprachigen Raum, den europäischen Nachbarländern sowie am europäischen Markt interessierte internationale Unternehmen ihre Produkte und Lösungen im Oracle-Umfeld vor.

Bei der traditionellen Abendveranstaltung am Mittwoch stand das Networking im Vordergrund. Unter jeweils einem eigenen Motto waren in verschiedenen Räumen internationale Themen-Buffets aufgebaut. Live-Bands sorgten für gute Stimmung; ruhige Zonen ermöglichten persönliche Gespräche. Sportlich ambitionierte Teilnehmer vergnügten sich derweil blendend beim Kickerturnier unter Beteiligung der Tischfußball-Weltmeisterin Lilly Andres und Vize-Europameister Johannes Kirsch.

Nach dem Motto „Nach der Konferenz ist vor der Konferenz ...“ arbeitet das DOAG-Team bereits jetzt schon an der Vorbereitung: Die DOAG 2010 Konferenz + Ausstellung findet von 16. bis 19. November 2010 wieder im CongressCenter Nürnberg Ost statt.

# „Oracle-Sun-Übernahme ein Sieg für Open Source …“

Martin Schindler, silicon.de

**Unter der Leitung von Marten Mickos wurde 2008 MySQL für eine Milliarde Dollar von Sun Microsystems gekauft. Seit 2001 war Mickos auf dem Chef-Posten der wohl am weitesten verbreiteten Open-Source-Datenbank, die auch manchem kommerziellen Anbietern in die Quere kam. Manche Experten sind gar der Ansicht, dass ohne MySQL Microsoft, Oracle oder IBM niemals Express-Versionen ihrer großen Datenbanken angeboten hätten.**

„Ich war von dem Moment an, in dem ich zu MySQL kam, von der Idee von Open Source begeistert“, erklärte Mickos. Er habe auch sofort das Gefühl gehabt, dass das eine „große Sache werden könnte“. Es wurde eine große Sache.

Beim Verkauf von MySQL an Sun brachte das Management Trinksprüche aus, es herrschte Feierlaune. Und jetzt steht MySQL wieder vor einem Verkauf. Diesmal aber klingen irgendwie keine Gläser. Das mag vielleicht auch damit zusammenhängen, dass sich Sun-Mitarbeiter und damit auch MySQL-Mitarbeiter nicht zu der formal noch nicht abgeschlossenen Übernahme äußern dürfen. Nichtsdestotrotz macht sich Mickos, derzeit als Mitarbeiter des Investors Benchmark Capital, wieder für eine Übernahme stark. Unter den ehemaligen MySQL-Mitarbeitern ist er damit nicht unumstritten.

Mitte Oktober drängte Michael ‹Monty› Widenius, Gründer und Entwickler der ersten Stunde von MySQL, in einem Blog Oracle, MySQL doch lieber zu verkaufen. MySQL brauche ein Zuhause, in dem es keine Interessenskonflikte gebe. Bislang ist MySQL die einzige Sun-Geschäftseinheit, die von der EU-Kommission in ihrer Begründung für das Einleiten einer eingehenden Untersuchung des Fusions vorhabens thematisiert wurde, erklärte Widenius weiter. Es sei sinnvoller, MySQL in geeignete Hände abzugeben, als Sun weiter unter der Verzögerung der Übernahme leiden zu lassen.

Der Vorkämpfer für freie Software Richard Stallman, warnte in einem Schreiben an die EU vor den negativen Folgen durch die Übernahme für MySQL. Stallman argumentiert, dass MySQL sich immer weiter entwickle und immer mehr zu einer Alternative und damit zu einer echten Konkurrenz für Oracle Datenbanken heranreife. Für ihn ist schon jetzt klar, dass Oracle den quelloffenen Konkurrenten beschneiden werde.

Aber das will Mickos nicht gelten lassen. Auch er wandte sich in einem Schreiben an die EU. Oracle habe seiner Meinung nach noch bessere Gründe als Sun, an der aktuellen Ausprägung von MySQL festzuhalten. Außerdem sei die Kontrolle Oracles über MySQL durch die große Zahl der Nutzer der quelloffenen Version beschränkt. So werde MySQL meist zusammen mit großen Web-Servern eingesetzt, eine Aufgabe, für die Oracles Backend nicht ausgelegt sei. Weil Sun durch die Unsicherheit starke Umsatzeinbußen hinnehmen müsse, schade jede weitere Blockade durch die Kartellbehörden dem freien Markt mehr, als diese Überprüfung nutzen könnte.

Jetzt erklärt Mickos, dass Oracle von der Übernahme „enorm profitieren“ könne. Doch nicht nur Oracle könnte profitieren, auch für die gesamte Open-Source-Bewegung wäre diese Übernahme in Mickos Augen ein Segen. Schließlich habe Sun nicht nur MySQL im Portfolio, sondern auch eine ganze Reihe anderer quelloffener Projekte wie Java, Glassfish, ZFS, OpenSolaris, OpenOffice. Und Sun hat zudem eine quelloffene Prozessor-Architektur.

„Oracles Bereitschaft, Milliardensummen für Open-Source-Technologien zu bezahlen, sollte als ein positives Zeichen gewertet werden. Es ist keine Bedrohung, es ist eine der wichtigsten Siege für Open Source aller Zeiten.“ Es sei für ihn ein Zeichen für einen sehr positiven Wandel, dass Oracle es als strategisch wichtig erachtet, den weltgrößten Produzenten einer quelloffenen Datenbank zu übernehmen.

Schon vor einigen Jahren war Mickos überzeugt, dass eines Tages kein Anbieter mehr ohne Open-Source-Technologien auskommen wird. Und mit der immer stärker werdenden Konkurrenz durch Google könnte selbst Microsoft damit beginnen, vermehrt auf Open Source zu setzen. Warum also sollte Oracle dann eine Open-Source-Datenbank behindern?

„Ich habe die Überzeugung, dass mit der Zeit auch Microsoft umkehren und einer der größten Freunde von Open Source werden wird. Ich kann zwar nicht wirklich erklären, warum ich das glaube, aber ich bin überzeugt, dass es unvermeidbar ist“, prognostiziert Mickos. Denn schon jetzt gebe es viele Parallelen.

„Man möchte glauben, dass diese beiden Lager ja total gegensätzlich sind, aber das einzige was Microsoft von Open Source unterscheidet, ist die Tatsache, dass Microsoft bislang geschlossenen Code für den besseren Weg gehalten hat. Aber auf der anderen Seite ist Microsoft sehr entwicklerfreundlich – sie haben gute Tools und sie lieben es, Communities zu bilden. Microsoft hat viele Aspekte, in denen es der Open-Source-Welt sehr ähnlich ist.“

Sobald Microsoft die Herangehensweise bei der Lizenzierung und bei Software-Patenten ändere, könnte Open Source tatsächlich ein integraler Bestandteil der Microsoft-Strategie für die Bindung von Entwicklern und die Bildung von Communities werden: „Das wird noch einige Zeit in Anspruch nehmen und ich kann mich auch täuschen, aber irgendwie fühle ich, dass genau das passieren wird“, so Mickos.

# Terminkalender Februar bis April 2010

## Februar

Montag, 08.02.2010
**Regionaltreffen Osnabrück/ Bielefeld/Münster**
*Andreas Kother, Klaus Günther, Stefan Kinnen*
regio-osnabrück@doag.org

Dienstag, 09.02.2010
**DOAG Berliner Expertenseminare**
Advanced Oracle Objekt Monitoring
*DOAG Geschäftsstelle*
office@doag.org

Dienstag, 09.02.2010
**Regionaltreffen Hamburg/Nord**
Themen: PL/SQL und Performance, Oracle Support
*Stefan Thielebein, Michael Paege*
regio-nord@doag.org

Mittwoch, 10.02.2010
**DOAG Berliner Expertenseminare**
Advanced Oracle Datenbank Monitoring
*DOAG Geschäftsstelle*
office@doag.org

Dienstag, 23.02.2010
**DOAG Logistik & SCM 2010**
*DOAG Geschäftsstelle*
office@doag.org

Donnerstag, 25.02.2010
**Regionaltreffen Südbayern/München**
*Franz Hüll, Andreas Ströbel*
regio-muenchen@doag.org

## März

Montag, 01.03.2010
**DOAG Berliner Expertenseminare**
Fiscal Requirements, Financial Standards for German Oracle E-Business Suite R12 Implementations
*DOAG Geschäftsstelle*
office@doag.org

Dienstag, 02.03.2010
**DOAG Berliner Expertenseminare**
Governance, Risk & Compliance for Oracle Applications Users
*DOAG Geschäftsstelle*
office@doag.org

Dienstag, 02.03.2010
**Regionaltreffen Jena/Thüringen**
*Jörg Hildebrandt*
regio-thueringen@doag.org

Mittwoch, 03.03.2010
**Regionaltreffen Berlin/Brandenburg**
*Michel Keemers*
regio-berlin@doag.org

Dienstag, 09.03.2010
**SIG Security & Identity Management**
*Frank Stöcker*
sig-security@doag.org

Donnerstag, 11.03.2010
**Regionaltreffen NRW**
Themenfeld Development: Nutzung von Webservices & n.N.
*Stefan Kinnen*
regio-nrw@doag.org

Dienstag, 16.03.2010
**Regionaltreffen Nürnberg/Franken**
*Jürgen Häffner*
regio-nuernberg@doag.org

Donnerstag, 18.03.2010
**Regionaltreffen Stuttgart**
*Jens-Uwe Petersen*
regio-stuttgart@doag.org

Dienstag, 23.03.2010
**SIG BPM**
*Sebastian Graf*
sig-bpm@doag.org

Mittwoch, 24.03.2010 – Donnerstag, 25.03.2010
**DOAG 2010 Applications**
*DOAG Geschäftsstelle*
office@doag.org

Donnerstag, 25.03.2010
**Regionaltreffen Südbayern/München**
*Franz Hüll, Andreas Ströbel*
regio-muenchen@doag.org

Donnerstag, 25.03.2010
**Regionaltreffen Trier/ Saarland/Luxemburg**
*Bernd Tuba, Holger Fuchs*
regio-trier@doag.org

## April

Mittwoch, 14.04.2010
**SIG Middleware**
*Perry Pakull*
sig-middleware@doag.org

Donnerstag, 15.04.2010
**Regionaltreffen Rhein-Neckar**
Oracle Database 11g R1/R2
*Kai F. Christianus*
regio-rhein-neckar@doag.org

Mittwoch, 21.04.2010
**Großes Frühjahrstreffen der Regionalgruppe NRW**
mehrere Themen mit viel Networking
*Dierk Lenz*
regio-nrw@doag.org

Mittwoch, 21.04.2010
**Regionaltreffen Südbayern/München**
*Franz Hüll, Andreas Ströbel*
regio-muenchen@doag.org

Montag, 26.04.2010
**DOAG Berliner Expertenseminare**
Security
*DOAG Geschäftsstelle*
office@doag.org

Dienstag, 27.04.2010
**DOAG Berliner Expertenseminare**
Security
*DOAG Geschäftsstelle*
office@doag.org

Dienstag, 27.04.2010
**Regionaltreffen Hamburg/Nord**
*Stefan Thielebein, Michael Paege*
regio-nord@doag.org

Donnerstag, 29.04.2010
**SIG Development**
Rapid Development mit Java (Start der SIG Java)
*Andreas Badelt*
sig-development@doag.org

Weitere, aktuelle Informationen finden Sie unter www.doag.org/termine