Seit der OpenWorld im Oktober 2011 ist die Oracle NoSQL DB verfügbar. Sie bietet das Speichern extrem großer Datenmengen in einer massiv verteilten (Datenbank-)Umgebung.

SQL oder NoSQL DB: Das ist die Frage!

Carsten Czarski, ORACLE Deutschland B.V. & Co. KG

Wer sich mit relationalen Datenbanken, insbesondere dem Oracle Relational Database Management System (RDBMS), beschäftigt und den Vorspann liest, stellt sich einige Fragen:

- Welche Daten speichere ich darin?
- Warum verwende ich dafür nicht das klassische RDBMS?
- Wie spielt das Ganze mit der vorhandenen Infrastruktur zusammen?

Dieser Artikel gibt einen Überblick über NoSQL-Datenbanken, stellt sie in Zusammenhang mit dem Themenkomplex „Big Data“ und geht insbesondere auf die Oracle NoSQL DB ein.

Wir werfen keine Daten mehr weg

Unter „Big Data“ wird weit mehr als die einfache Übersetzung „große Datenmengen“ verstanden. Ein besonders großes Data Warehouse ist also gerade nicht gemeint. Es geht vielmehr um die Arbeit mit Daten, die bislang in der IT kaum verwendet wurden, da sie einerseits zu wenig strukturiert sind und andererseits zu viele Daten anfallen, um alles abzuspeichern. Beispiele sind:

- Webserver-Logdateien, Trace-Dateien
- Sensordaten
- Kommentare auf der Unternehmens-Webseite oder in sozialen Netzwerken

Die Qualität dieser Daten im Rohzustand macht die Auswertung mit traditionellen Methoden nahezu unmöglich. In kurzen Zeiträumen entstehen sehr große Datenmengen, sodass beispielsweise Webserver-Logdateien meist nach einem gewissen Zeitraum entsorgt werden (müssen). Zum anderen sind die Daten nicht besonders „dicht“ – die Antwort auf eine fachliche Frage ist nur in einem kleinen, aber über den ganzen Datenbestand verstreuten Teil enthalten. Je nach Fragestellung betrifft das jedes Mal einen anderen Teil. Interessante Antworten ergeben sich meist erst bei Betrachtung eines großen Zeitraums und dem Durcharbeiten entsprechend großer Datenmengen, die man vorhalten muss.

Es ist allerdings praktisch aussichtslos, für diese Daten ein relationales Schema zu modellieren und sie ins Data Warehouse aufzunehmen – andere Konzepte sind gefragt. Abbildung 1 zeigt „Big Data“ im Zusammenspiel mit einer klassischen Infrastruktur mit OLTP- und DWH-Systemen. Letztere (in der Abbildung unten dargestellt) beschäftigen sich mit „dichten“, modellierten Daten. „Big Data“ bezeichnet den oberen Bereich: Neue Daten, mit denen man sich bislang aus genannten Gründen kaum beschäftigte, werden zunächst über einen längeren Zeitraum gespeichert. Hier kommen NoSQL-Datenbanken, wie die in diesem Artikel vorgestellte Oracle NoSQL DB, oder das quelloffene Hadoop Distributed Filesystem (HDFS) zum Einsatz.

Ein nachgelagerter Prozess verdichtet beziehungsweise aggregiert diese Daten. Da es aber hierbei um extrem große Datenmengen geht, die verarbeitet werden sollen, werden verteilte Technologien, die sehr gut horizontal skalieren können, wie Hadoop MapReduce eingesetzt. Die Ergebnisse dieser Prozesse stehen allerdings nicht für sich allein, sondern werden vielmehr gemeinsam mit den strukturierten Informationen im Data Warehouse verfügbar gemacht.

Big Data erfassen und speichern

Für eine Plattform, um die mit „Big Data“ bezeichneten Daten auch über einen längeren Zeitraum zu speichern, sind vor allem zwei Dinge erforderlich:

- Das System muss hinsichtlich der Datenstrukturen sehr flexibel sein; es muss nahezu „alles, was kommt“, aufnehmen können.

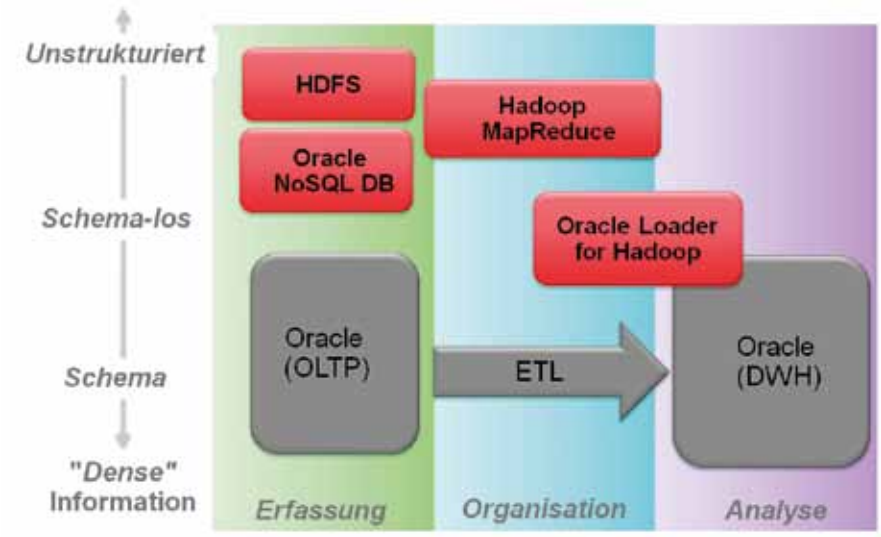

Abbildung 1: Neue Datenformen (Big Data) im Zusammenspiel mit dem Data Warehouse

- Da die Datenmengen über die Zeit sehr stark wachsen werden, muss das System horizontal sehr leistungsstark sein – sowohl hinsichtlich des Plattenplatzes als auch in Bezug auf die Prozessorleistung.

Wie gesagt, ein relationales Schema ist für Big Data normalerweise nicht angemessen, ein RDBMS ist für „dichte" Daten konzipiert, die in ein vorher durchdachtes Datenmodell passen. Ein Oracle RDBMS, das alle Sensordaten einer Produktionsanlage oder alle Webserver-Logdateien eines Internet-Unternehmens speichern soll, müsste extrem groß dimensioniert sein. Es könnte seine Fähigkeiten aber kaum ausspielen, da die Daten sich in kein Datenmodell bringen und in Rohform nicht auswerten lassen. Daher werden für Big Data nicht-schemabasierte, massiv verteilte Systeme eingesetzt.

Geht es um das reine, einmalige „Wegschreiben" von Logdateien, so kommt häufig HDFS zum Einsatz. Es ist für „Write Once – Read Many"-Szenarien optimiert und erlaubt den Betrieb eines verteilten Dateisystems (Shared Nothing) über eine nahezu beliebige Anzahl Knoten. Für die nötige Stabilität auch bei Ausfall einzelner Knoten sorgt die dreifache Replikation, jeder Block ist also im Cluster dreimal gespeichert.

Für andere Szenarien, wie das Speichern und Abrufen von Profildaten einer Internetseite, ist HDFS dagegen nicht geeignet, denn an den Profilen werden regelmäßig Änderungen gemacht und einzelne Datensätze damit aktualisiert. Dazu ist es natürlich nötig, die Profildaten unter einem Schlüssel zu speichern und direkt abrufen zu können und nicht erst eine ganze Logdatei sequenziell durchsuchen zu müssen.

NoSQL-Datenbanken vs. RDBMS

Für die genannten Fälle kommt nur eine NoSQL-Datenbank infrage. Sie unterscheidet sich vom bekannten RDBMS vor allem durch zwei Merkmale:

- NoSQL-Datenbanken werden als massiv verteilte Systeme ausgelegt.
- NoSQL-Datenbanken haben bei Weitem nicht den Funktionsumfang eines RDBMS. Sie konzentrieren sich auf einfache Datenstrukturen wie Key-Value-, Dokument- oder Graph-Strukturen.

Da die Verteilung über mehrere Rechnerknoten oberste Priorität hat, spielt das „CAP-Theorem" eine wichtige Rolle. Es beschäftigt sich mit drei Anforderungen, die an verteilte Datenbanksysteme gestellt werden können (siehe Abbildung 2):

- *Datenkonsistenz (C)*
  Das System stellt sicher, dass stets konsistente Daten ausgeliefert werden.
- *Verfügbarkeit (A)*
  Das System ist stets verfügbar. „Verfügbar" ist hier definiert als „Antwortet innerhalb einer vorgegebenen Zeit".

ORACLE PARTNER

Technology for success

Wir unterstützen Ihren Erfolg mit der Konzeption und dem Aufbau Ihrer Datenbankanwendungen, sowie deren technischem Support.

Wir von der Krug & Partner GmbH bestehen aus einem hoch motivierten Team von Oraclespezialisten. Regelmäßige Schulungen halten unsere Fachkräfte immer auf dem aktuellsten Stand der Technik.

- **Datenbanken**
  Datenbank-Installation & -Konfiguration
  Administration & Wartung
  Backup/Recovery
  Health-Check
- **Application Server Systeme**
  JDeveloper Forms & Reports
  PL/SQL (CMSDK, OEM, Disco...) u.v.m.
- **Projekte**
  Gewerblicher Rechtsschutz (IP)
  Automotive Medien- & Energiewirtschaft
- **Lizenzierung**
  Beratung & Analyse des Lizenzstatus
  Lizenzierung

for-success

Krug & Partner GmbH ○ Treitschkestr. 3 ○ D-69115 Heidelberg
Telefon: +49 (0) 62 21/60 79 0 ○ Telefax: +49 (0) 62 21/60 79 60
E-Mail: info@krug-und-partner.de
www.krug-und-partner.de

KRUG & PARTNER

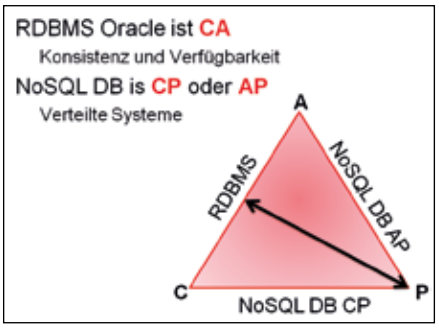

Abbildung 2: CAP-Theorem

- *Partitionstoleranz (P)*
  Das System arbeitet als Ganzes auch weiter, wenn einzelne Teile ausfallen.

Von diesen drei Anforderungen kann ein Datenbanksystem maximal zwei gleichzeitig erfüllen:

- Konsistent und verfügbar sein (CA)
- Partitionstolerant und verfügbar sein (AP)
- Partitionstolerant und konsistent sein (CP)

Das klassische RDBMS ist ein „CA"-System, denn die jederzeitige Datenkonsistenz hat höchste Priorität. Da es kein verteiltes System ist, spielt das „P" nur eine untergeordnete Rolle.

NoSQL-Datenbanken legen ihren Fokus immer auf die Partitionstoleranz (P). Ob ein NoSQL-System nun ein „CP"- oder ein „AP"-System ist, hängt von der verwendeten Datenbank selbst oder bei der Oracle NoSQL DB vom Entwickler ab – denn dieser kann bei seinen Aufrufen entsprechende Parameter setzen. Die Partitionstoleranz genießt höchste Priorität, alles andere wird diesem Ziel untergeordnet; so auch der Funktionsumfang, der sich auf einfache Operationen wie „GET", „PUT" und „DELETE" beschränkt und nicht im Entferntesten mit dem eines RDBMS vergleichbar ist:

- Es gibt keine Abfragesprache und keinen Optimizer
- Es gibt keine Funktionsbibliothek, keine Stored Functions oder Procedures
- Es gibt keine Nutzerverwaltung, kein Rechte- und Rollenkonzept

Diese Liste lässt sich weiter fortführen. Im Kern gilt, dass eine NoSQL-Datenbank sich auf die grundlegenden Funktionen zum Speichern und Abrufen der Daten konzentriert und diese in einer verteilten Umgebung bereitstellt.

Key-Value-Speicherung bei der Oracle NoSQL DB

Die Oracle NoSQL Datenbank ist ein kommerzieller Vertreter der Gattung „NoSQL-Datenbank". Neben der frei verfügbaren Community-Edition kann diese auch unter kommerzieller Lizenz mit entsprechendem Supportvertrag eingesetzt werden.

Die Oracle NoSQL DB ist eine Key-Value-Datenbank. Das bedeutet, dass die Daten unter einem Schlüssel (Key) abgespeichert und mit diesem (und nur mit diesem) wieder abgerufen werden können. Die NoSQL-Datenbank verwendet keine Schemata und Datenstrukturen. Die als „Value" gespeicherten Daten sind ein Byte-Array, das nur die Anwendung interpretieren kann. Dies ist ein sehr wichtiger Unterschied: Eine NoSQL-Datenbank ist eng an die Anwendung gebunden und ohne Anwendung sind die Daten nicht les- oder interpretierbar.

Auch in einem RDBMS könnte man sich eine Key-Value-Datenhaltung vorstellen – eine Tabelle mit zwei Spalten: „KEY" vom Typ „NUMBER" oder „VARCHAR2" mit Index und Primary-Key-Constraint sowie „VALUE" vom Typ „BLOB". In einem solchen Setup

---

Die System Change Number definiert den Zeitstrahl jeder Oracle-Datenbank. Sie wird mit jeder abgeschlossenen Transaktion inkrementiert und ist so groß definiert, dass sie in absehbarer Zeit nicht an ihr Maximum kommen sollte.

Das SCN Headroom Problem – ist meine Oracle Datenbank in Gefahr?

Die System Change Number (SCN) erlaubt 281 Billionen Werte. Oracle sieht die aktuelle SCN als unkritisch an, wenn sie den Wert nicht überschreitet, der der Anzahl Sekunden seit 1988 multipliziert mit 16384 darstellt. Dieser Wert wird als maximale SCN angesehen. Diese wird etwa für die nächsten 500 Jahre nicht überlaufen. Die Differenz zwischen dem aktuellen Wert und der maximalen SCN wird als SCN Headroom bezeichnet.

Wo liegt nun die Gefahr? Sie ist in dem Verhalten begründet, dass Datenbanken sich auf die größte beteiligte SCN gleichziehen, wenn eine verteilte Transaktion abgeschlossen wird. Der hier beteiligte Code hat wohl mehrere Bugs, sodass die SCN springen kann. Somit besteht die Gefahr, dass der SCN Headroom kleiner wird beziehungsweise ganz verschwindet. Da eine Oracle-Datenbank jede SCN ablehnt, die die maximale SCN überschreitet, könnten hierdurch größere Probleme entstehen.

Alle Bugs im Zusammenhang mit dem SCN Headroom sind im Critical Patch Update für Januar 2012 behoben. Bei der Planung zur Patch-Installation ist zu beachten, dass nicht nur die wichtigsten Produktions-Datenbanken zu beachten sind, sondern auch alle anderen Datenbanken, die mit diesen verteilte Transaktionen haben.

Um eine Vorstellung davon zu bekommen, ob eine Datenbank ein SCN-Headroom-Problem hat, gibt es von Oracle Support ein Skript (scnhealthcheck.sql), das als Patch 13498243 ausgeliefert wird. Als Resultat gibt das Skript aus, ob der SCN Headroom in Ordnung ist, oder ob eine oder mehrere Maßnahmen empfohlen werden. Diese können aus der Installation des aktuellen CPUs und dem Setzen von Parametern bestehen. Neben dem Skript sind auf Oracle Support weiterer Informationen zu bekommen, beispielsweise im Dokument 1393363.1.

Dierk Lenz
dierk.lenz@hl-services.de

ist der Zugriff allein über den Schlüssel möglich; in den BLOB selbst kann man nicht hineinsehen.

Damit ist einleuchtend, warum es keine Abfragesprache und keine umfassende Funktionsbibliothek gibt: Dies würde voraussetzen, dass die Datenbank, auch unabhängig von der Anwendung, über die Daten Bescheid weiß.

Eine Key-Value-Datenbank bietet drei grundlegende Zugriffsmöglichkeiten an: „Get“, „Put“ und „Delete“, die vom Anwendungsentwickler per Programmierschnittstelle (API) angesprochen werden. Wie einfach der Umgang mit der NoSQL-Datenbank für den Anwendungsentwickler ist, zeigt sich an einem Codebeispiel, das ein Key-Value-Paar speichert (siehe Listing 1).

```
import oracle.kv.*;
public class NoSQLDB_Store {
 public static void main(String args[]) {
   KVStore store= KVStoreFactory.getStore (
     new KVStoreConfig(
       „workshopstore“,
       new String[] {„sccloud031:10100“, „sccloud031:10200“}
     )
   );
   store.put(
     Key.createKey(„DiesIstEinKey“),
     Value.createValue(new String(„Dies ist ein Wert“).getBytes())
   );
   store.close();
 }
}
```

Listing 1: Java-Code zum Abspeichern eines Key-Value-Paares in der Oracle NoSQL DB

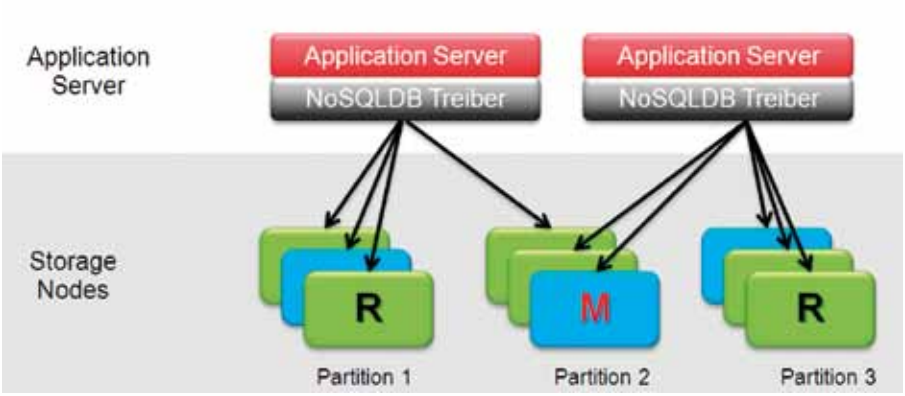

Abbildung 3: Replikationskonzept der Oracle NoSQL DB

Verfügbarkeit durch Replikation

Die Oracle NoSQL Datenbank ist ein verteiltes System. Damit das System auch bei Ausfall eines Knotens verfügbar bleibt (Partitionstoleranz), müssen die Daten eines jeden Knotens mindestens einmal repliziert sein (siehe Abbildung 3).

Beim Aufsetzen der NoSQL-Datenbank wird festgelegt, auf wie viele Partitionen die Daten verteilt werden sollen. Ein Hash-Algorithmus bestimmt für jedes Key-Value-Paar anhand des Keys die zugehörige Partition. Dieser Hash-Algorithmus bleibt für den gesamten Lebenszyklus der NoSQL-Datenbank gleich. Neben der Partitionszahl wird beim Einrichten auch der Replikationsfaktor bestimmt. „Faktor drei“ bedeutet zum Beispiel: ein Master und zwei Replikate.

Die Anwendung kommuniziert mit den Servern der Oracle NoSQL DB per „NoSQL DB Treiber“. Wird nun ein Key-Value-Paar gespeichert, also die Java-Methode „put()“ aufgerufen, so bestimmt der Treiber zunächst die Partition. Aus dieser lässt sich der Master ableiten, zu dem das Key-Value-Paar gesendet wird. Die Administrationsprozesse sorgen dann dafür, dass es zusätzlich zu den Replikaten transportiert und dort abgelegt wird.

Die NoSQL DB verwendet eine Master-Slave-Replikation: Schreibzugriffe erfolgen also immer und grundsätzlich auf dem Master. Lesende Zugriffe können folgerichtig sowohl vom Master als auch von einem der Replikate bedient werden.

Fällt ein Master aus, so sorgen die Administrationsprozesse dafür, dass ein neuer Master gewählt wird. Aus diesem Grund ist ein Replikationsfaktor von mindestens drei erforderlich, denn nur dann sind wenigstens zwei Replikate vorhanden, von denen einer der neue Master werden kann. Nachdem der ausgefallene Knoten von einem Administrator wiederhergestellt wurde, arbeitet er als Replikat weiter.

Stets konsistent oder kurze Antwortzeiten?

Wie gesagt, genießt die Partitionstoleranz bei der Oracle NoSQL DB höchste Priorität. Ob das System nun „CP“ (Fokus auf Datenkonsistenz) oder „AP“ (Fokus auf Verfügbarkeit) ist, hängt von den vom Entwickler verwendeten API-Aufrufen ab.

Beim Schreiben eines Key-Value-Paares kann der Entwickler ein „Durability“-Objekt übergeben. Damit wird festgelegt, ob die Anwendung warten soll, bis sowohl der Master als auch alle Replikate den Schreibvorgang bestätigt haben, ob die einfache Mehrheit der Replikate reicht oder ob sie weiterlaufen soll, ohne auf eine Bestätigung zu warten.

Analog dazu kann beim Lesevorgang mit der Methode „get()“ ein „Consistency“-Objekt übergeben werden. Legt der Entwickler darin fest, dass er nur die aktuellsten Daten sehen möchte, so wird vom Master gelesen. Alternativ kann der Entwickler auch eine Zeitspanne angeben, welche die Daten maximal zurückliegen dürfen. Wenn ein Replikat die Bedin-

```
// Erzwinge Lesen vom Master => liefert stets jüngste Version  store-
Conf.setConsistency(Consistency.ABSOLUTE);

// Lese auch von Replica, akzeptiere alle Ergebnisse, egal wie alt
storeConf.setConsistency(Consistency.NONE_REQUIRED);

// Akzeptiere nur Ergebnisse, die nicht älter als vorgegeben sind
storeConf.setConsistency(
 new Consistency.Time(
  5,    // Erlaubter Rückstand
  java.util.concurrent.TimeUnit.SECOND,
  10,   // Timeout
  java.util.concurrent.TimeUnit.SECOND
 )
);
```

Listing 2: Die Einstellungen zur Lese- und Schreibkonsistenz werden allein vom Entwickler festgelegt

gung erfüllen kann, also hinreichend „up to date“ ist, bedient diese die Anfrage. Schließlich kann der Entwickler auch ohne Einschränkungen jede Antwort akzeptieren. Der NoSQL-DB-Treiber wählt dann irgendein Replikat aus, ohne darauf zu achten, wie weit dieses im Replikationsprozess zurückliegt.

Sowohl die „Durability“ beim Schreiben als auch die „Consistency“ beim Lesen können individuell für jeden Aufruf, aber auch per Default für die ganze Anwendung gesetzt werden: Wie sich die Oracle NoSQL Datenbank verhält, liegt also allein in den Händen des Anwendungsentwicklers (siehe Listing 2).

Es stellt sich nun die Frage, wie die in einer Oracle NoSQL DB gespeicherten Daten hinsichtlich bestimmter Kriterien ausgewertet werden können. Um Daten abrufen zu können, muss man schließlich den Schlüssel (Key) kennen. Ist gefordert, alle „Datensätze“ mit einem bestimmten Merkmal auszugeben, so kann die NoSQL-Datenbank selbst wenig tun. Denn wie schon gesagt: Sie kennt die Struktur der Daten gar nicht. Vielmehr muss der gesamte Datenbestand durchgearbeitet werden. Da wir davon ausgehen müssen, dass die zu verarbeitenden Datenmengen sehr groß sind, sollte dieser Vorgang ebenfalls massiv parallel und verteilt erfolgen. Für solche Fälle ist das Hadoop MapReduce Framework geeignet. Eine ausführliche Betrachtung folgt in der nächsten Ausgabe.

Fazit

Das neue Thema „Big Data“ beschäftigt sich mit der Auswertung und Analyse von Daten, die bislang wegen ihrer fehlenden Struktur, der großen Datenmengen und der geringen Qualität der Rohdaten keine Berücksichtigung fanden. Wichtig hierfür ist eine Plattform, die diese Daten auch über einen längeren Zeitraum hinweg speichern kann und entsprechend horizontal skalierbar ist. Für komplett unstrukturierte Daten wie Logdateien wird vielfach das zum Hadoop Framework gehörende HDFS eingesetzt, für schwach strukturierte Daten wie Profildaten, die wahlfreien Zugriff per Schlüssel erfordern (Key-Value-Paare), kommen NoSQL-Datenbanken zum Einsatz. Mit der Oracle NoSQL Datenbank liegt ein kommerzieller Vertreter dieser Gattung vor – sie erlaubt das Speichern schwach strukturierter Daten in einer verteilten, horizontal sehr gut skalierbaren Umgebung und damit das Aufnehmen auch sehr großer Datenmengen. Gemeinsam mit Hadoop MapReduce, dem HDFS und den Oracle Connectors for Hadoop bildet die Oracle NoSQL DB eine Plattform für Big Data, die vor allem den Anspruch hat, Big Data ins Zusammenspiel mit der vorhandenen IT-Infrastruktur, dem vorhandenen Data Warehouse und den vorhandenen BI-Systemen zu bringen.

Weitere Informationen

[1] Oracle NoSQL Datenbank im OTN: http://www.oracle.com/technetwork/products/nosqldb/overview/index.html

[2] Whitepaper „Big data Overview”: http://www.oracle.com/technetwork/server-storage/engineered-systems/bigdata-appliance/overview/wp-bigdatawithoracle-1453236.pdf

[3] Apache Hadoop (Map Reduce und HDFS): http://hadoop.apache.org

Carsten Czarski
carsten.czarski@oracle.com
http://twitter.com/cczarski
http://sql-plsql-de.blogspot.com

Unsere Inserenten

ARETO Consulting GmbH www.areto-consulting.de	S. 15
Hunkler GmbH & Co. KG www.hunkler.de	S. 3
KeepTool GmbH www.keeptool.com	S. 17
Krug & Partner GmbH www.krug-und-partner.de	S. 35
Libellle AG www.libelle.com	S. 21
MuniQsoft GmbH www.muniqsoft.de	S. 23
OPITZ CONSULTING GmbH www.opitz-consulting.com	U 2
Oracle Deutschland B.V. & Co. KG www.oracle.com	U 3
ProLicense GmbH www.prolicense.com	S. 9
Trivadis GmbH www.trivadis.com	U 4
WIN-Verlag GmbH & Co. KG www.win-verlag.de	S. 49