Oracle Data Warehouses und Big Data im Zusammenspiel

Peter Welker, Trivadis GmbH

Immer häufiger stellt sich die Frage, wie man Daten aus der Oracle-Datenbank und solche aus Big-Data-Plattformen wie Hadoop und NoSQL-Datenbanken verschieben oder gemeinsam auswerten kann. Der Artikel erklärt zahlreiche Möglichkeiten, (nicht) immer nur aus der Oracle-Datenbank-Perspektive: Angefangen mit den Oracle Big Data Connectors über den Oracle Data Integrator, Sqoop und Oracle Heterogenous Gateway bis hin zu Big Data SQL oder Golden Gate gibt es zahlreiche Lösungen. Aber Vorsicht, die Tücke steckt – wie immer – im Detail.

Data Warehouses (DWH) sind vermutlich das prägnanteste Beispiel für Datenbanken, die Daten zahlreicher Vorsysteme in eine zentrale Datenbasis integrieren. Dafür hat sich in den letzten dreißig Jahren eine Reihe von Methoden und Techniken etabliert, die man heute in den meisten DWHs antrifft. Oft fließen die Daten zentral in eine DWH-Core-Datenbank, auf der sie – üblicherweise einmal täglich – einheitlich aufbereitet abgelegt und über lange Zeit historisch gehalten werden. Darauf setzen dann diverse, Abfrage-optimierte Data Marts auf, die jeweils anforderungsgerechte Abfrage-Schnittstellen auf diese Daten zur Verfügung stellen.

In den letzten Jahren häufen sich jedoch die Anforderungen an nichttabellarische Daten, kürzere Aufbereitungszeiten, Self-Service-Anforderungen, die DWH-Daten auch mit fremden Daten gemeinsam nutzen wollen, oder an Datenmengen und Durchsätze, die vor allem zu enormen Datenbank-Lizenz- und -Supportkosten führen, wenn man sie performant in das DWH integrieren möchte. Hier kommen neue Technologien wie Hadoop oder NoSQL-Datenbanken ins Spiel, die es gilt mit den bestehenden, relationalen DWHs zu vereinen. Die Frage ist: „Wie soll das funktionieren?" Wie in den letzten Jahren zu sehen, werden diese Technologien die relationalen und multidimensionalen Lösungen nicht einfach ersetzen können. Sie werden aber auch nicht wieder verschwinden. Vielmehr ist das Zusammenspiel der verschiedenen Welten ins Zentrum der Hersteller- und Dienstleisterbestrebungen gerückt. „Big Data Management System" heißt das Zauberwort dafür bei Oracle.

Im aktuellen „Statement of Direction" für diese Big-Data-Management-Systeme *(siehe*

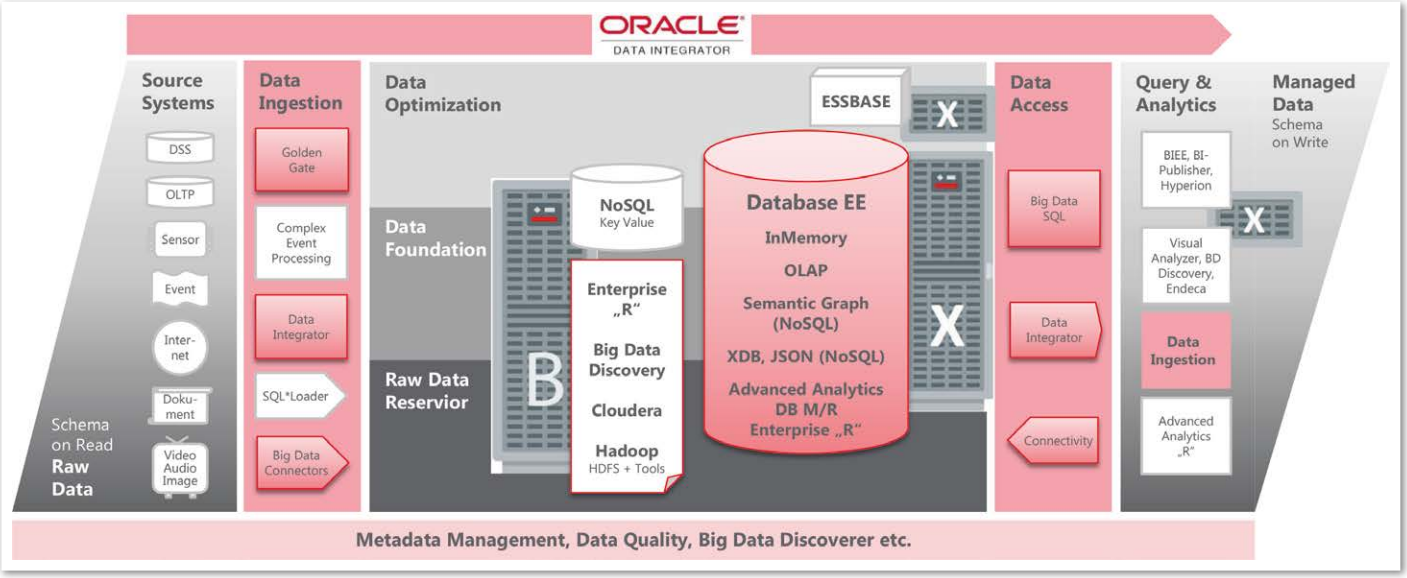

Abbildung 1: Die Oracle „Big Data Management System"-Komponenten – Stand heute – im Überblick

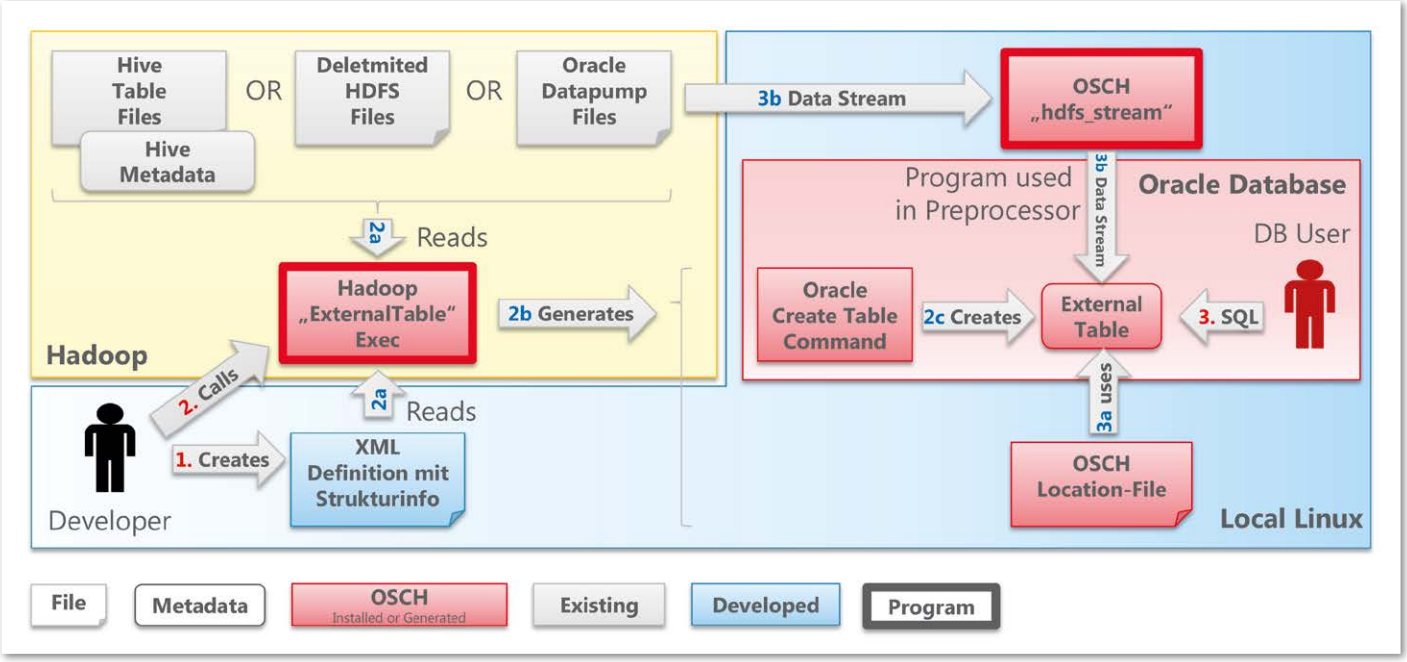

Abbildung 2: Die einzelnen Schritte und Komponenten beim Einsatz von OSCH

„http://www.oracle.com/technetwork/database/bigdata-appliance/overview/sod-bdms-2015-04-final-2516729.pdf“) erklärt Oracle eigene Absichten und Visionen für das Zusammenspiel von Big-Data-Technologien und der Datenbank. Langfristig soll alles mehr oder weniger nahtlos zusammenwachsen. Auf dem langen Weg dorthin bilden unter anderem die Big-Data-Konnektoren ODI und Golden Gate den Leim, der beide Welten zusammenhält *(siehe Abbildung 1)*.

Aktuell muss man sich noch selbst darum kümmern, welche Technologie man für welche Anforderung einsetzt. Ideen dafür liefert beispielsweise ein weiteres Oracle Whitepaper *(siehe „http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/2297765.pdf“)*.

In diesem Artikel werden unter anderem die Konnektor-Komponenten in einem Überblick mit Fokus auf die Funktionsweise erklärt. Für mehr Informationen sei auf einschlägige Artikel und Dokumentationen verwiesen. Insbesondere empfehlenswert sind die Hands-on-Übungen zu den „Oracle Big Data Lite“ VMs, die kostenfrei auf der Oracle Website verfügbar sind. Um einen nahtlosen Übergang zu erleichtern, basieren die Beispiele aus diesem Artikel auf den Bit-Data-Lite-Übungen.

Apache Sqoop

Eingangs sei kurz Sqoop erwähnt. Die Open-Source-Lösung ist das traditionelle Hadoop-Werkzeug, um Daten von und nach Hadoop zu bewegen, und darum essenzieller Teil von Hadoop-Distributionen. Es basiert in den neueren Versionen auf einer Client-Server-Architektur. Man kann sich Sqoop wie ein einfaches Daten-Integrationswerkzeug vorstellen. Es nutzt beliebige JDBC-Treiber und direkten Dateizugriff, ist parallelisierbar und via Kommandozeile scriptbar. Es kann in viele ETL-Tools eingebunden werden und wird folgerichtig auch von ODI genutzt, um Daten in die Hadoop-Plattform zu importieren. Behalten wir dies im Hinterkopf und beschäftigen wir uns nun ausführlicher mit den Oracle-eigenen Lösungen.

Oracle Big Data Connectors

Die lizenzpflichtigen Konnektoren von Oracle bestehen aus diversen Werkzeugen, mit denen Stand heute Daten aus der Big-Data-Welt in der Oracle-Datenbank zugänglich gemacht werden. Für den umgekehrten Weg – also Daten aus der Datenbank beispielsweise in Hadoop einzubinden – hat Oracle bisher lediglich Lösungen angekündigt (Oracle Table Access for Hadoop (OTA4H), *siehe „https://www.doag.org/konferenz/konferenzplaner/konferenzplaner_details.php?id=473721&locS=0&vid=504667“*). Die Big Data Connectors bestehen aus vier separaten Werkzeugen:

- Oracle SQL Connector for HDFS (OSCH)
- Oracle Loader for Hadoop (OLH)
- Oracle XQuery for Hadoop (OXH)
- Oracle R Advanced Analytics for Hadoop (ORH)

Oracle SQL Connector for HDFS

Der Oracle SQL Connector for HDFS (OSCH) erlaubt aus der Datenbank heraus mit der bekannten External-Table-Technik (ET, inklusive Präprozessor) direkte Read-only-Zugriffe auf das Hadoop File System (HDFS). Die unterstützten Formate sind „Oracle Data Pump“, „Delimited Text“ und „Delimited Text aus Apache-Hive-Tabellen“. Letzteres ist etwas erklärungsbedürftig: Apache Hive ist vereinfacht gesagt eine SQL-Schnittstelle auf Dateien in HDFS. Da-

```
hadoop jar \                                          -- hadoop Befehl
   $OSCH_HOME/jlib/orahdfs.jar \                      -- OSCH Bibliothek
   oracle.hadoop.exttab.ExternalTable \               -- OSCH Java Programm
   -conf /home/oracle/movies/moviefact_hdfs.xml \     -- XML mit ET Definition
   -createTable                                       -- Erzeugt Oracle ET Code
```

Listing 1

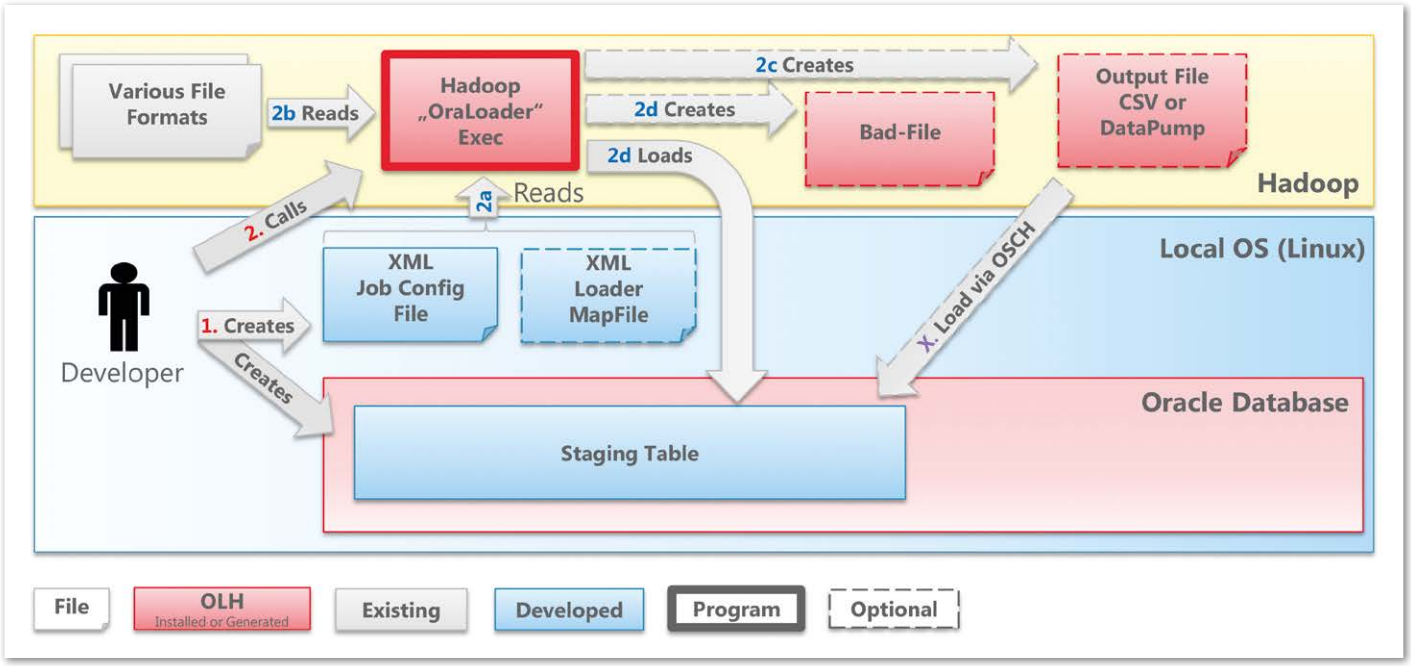

Abbildung 3: Die einzelnen Schritte und Komponenten beim Einsatz von OLH im Online-Modus

bei verwendet Hive ein eigenes Data Dictionary, den sogenannten „HCatalog". Die Konnektoren können dort hinterlegte Metadaten für das Mapping auf die zugehörigen HDFS-Dateien nutzen. Wichtig: Die SQL-Funktionalität von Hive selbst spielt dann für den Zugriff auf die Daten keine Rolle. Das geschieht direkt über HDFS und Oracle-eigene Komponenten. Leider ist die Verwendung von OSCH nicht ganz so einfach und bedarf einiger Vorbereitung *(siehe Abbildung 2)*.

Im ersten Schritt erzeugt man eine XML-Datei mit Orts- und – soweit nötig – Struktur-Informationen. Diese übergibt man dann in einem zweiten Schritt mittels Hadoop-Client-Aufruf (ein Hadoop Client muss auf dem DB Server installiert sein) an ein Hadoop-MapReduce-Programm, das Oracle mit OSCH mitliefert *(siehe Listing 1)*.

Das Programm erzeugt lediglich einen passenden External-Table-Befehl (auf Wunsch gleich mit Ausführung gegen die Datenbank) und ein sogenanntes „Location File", das dem OSCH-External-Table-Präprozessor den Ort der zu lesenden Datei(en) auf dem Hadoop-Filesystem mitteilt. Damit hat man nun eine External Table in der Oracle-Datenbank erstellt, die über einen generischen Streaming-Präprozessor auf eine – oder mehrere gleichformatige – Dateien in HDFS zugreift. Dieser Ansatz unterstützt – genau wie bei einer herkömmlichen External Table vom Typ „ORACLE_LOADER" – paralleles Lesen via SQL auf mehrere Dateien, Formatmasken, Datenkonvertierung etc.

Genauso herkömmlich ist allerdings auch der Zugriff auf die Daten: Es werden alle in die Oracle-Datenbank gezogen. Push-down von Filtern aus „WHERE"-Klauseln oder andere Mechanismen, um den Datenstrom zwischen Hadoop und der Datenbank einzuschränken, gibt es nicht. Es ist also keine gute Idee, auf diese Art in Terabyte-großen Hadoop-Dateien x-mal in der Stunde nach wenigen Datensätzen zu suchen. Gut geeignet ist es aber, um eine batchorientierte Leseschnittstelle auf kleinere oder mittelgroße Hadoop-basierte Daten zu implementieren.

```
hadoop jar \-- hadoop Befehl
   $OLH_HOME/jlib/oraloader.jar \          -- Tool aus folgender Bibliothek
   oracle.hadoop.loader.OraLoader \        -- Name des Tools (Java Programm)
   -conf /home/oracle/movies/jobconfig.xml  -- XML mit Loaderdefinition
```

Listing 2

Oracle Loader for Hadoop (OLH)

Dieses Werkzeug nutzt einen etwas anderen Weg, um Daten aus einem Hadoop-Cluster in die Oracle-Datenbank zu importieren. Im Mittelpunkt steht dabei vor allem der reine Durchsatz. OLH dürfte derzeit die schnellste Möglichkeit sein, um Daten aus HDFS in eine Tabelle in der Oracle-Datenbank zu laden. Unterstützt werden die bereits in OSCH genannten Datenformate sowie das AVRO-Format. Zudem gibt es eine (eingeschränkte) Möglichkeit, Daten aus der Oracle-NoSQL-Datenbank zu importieren und beliebige weitere Formate mittels Java-Programmierung (Erweiterung der „MapReduce InputFormat"-Klasse) einzubinden *(siehe Abbildung 3)*.

Der Einsatz ist auch hier mit ein wenig Arbeit verbunden. Zunächst erstellt man ein Job-Config-XML mit allen nötigen Loader-Definitionen wie Dateinamen, Parallelisierungsgrad, Name der Zieltabelle, Infos zur JDBC-Connection zur Datenbank etc. Wenn die Zieltabelle für den Ladevorgang der Quelldatei entspricht und weitere Bedingungen wie ein einheitliches Datumsformat und Spaltennamen passen, genügt das, um einen Online-Mode-Ladevorgang auszuführen. Wenn es komplizierter wird, muss man noch eine weitere XML-Datei mit Mapping-Informationen erstellen. Im nächsten Schritt wird der Loader gestartet und mit den XML-Dateien versorgt *(siehe Listing 2)*.

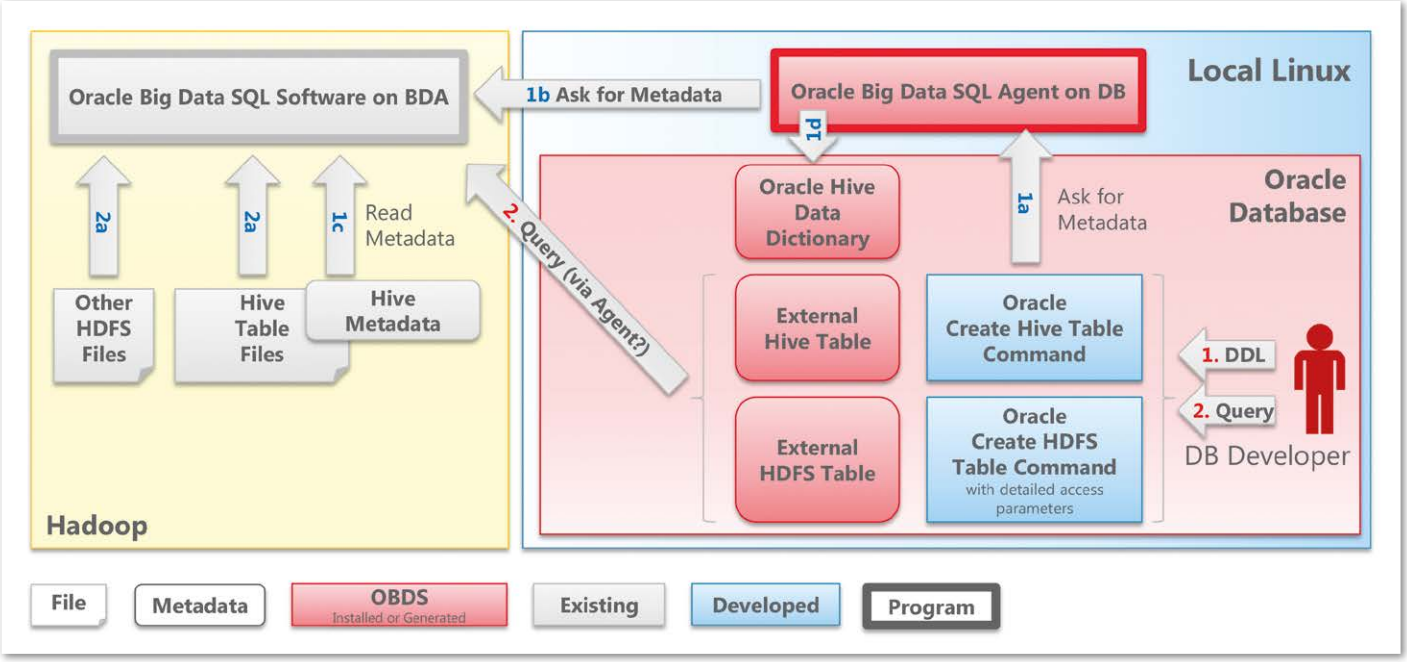

Abbildung 4: Die einzelnen Schritte und Komponenten beim Einsatz von Big Data SQL

Was nun passiert, ist einem klassischen „SQL*Loader"-Vorgang recht ähnlich: Das „OraLoader"-MapReduce-Programm liest die Dateien, konvertiert sie in ein passendes Ladeformat und lädt sie in eine Tabelle in der Datenbank. Bei Bedarf wird höchste Performance durch Parallelisierung dadurch erreicht, dass jeweils eine Datei in eine Partition einer Zieltabelle via Direct Insert (nur OCI-Treiber) geladen wird. Das dafür vorab nötige Sortieren und Gruppieren von Daten auf Hadoop erledigt OLH dabei selbstständig. Genau wie beim SQL*Loader können auch hier Bad Files geschrieben werden. In einem weiteren Modus, dem sogenannten „Offline-Mode", können die für das Laden vorbereiteten Daten auch in Dateien im Datapump-Format geschrieben und dann später via OSCH geladen werden.

Oracle Big Data SQL

Betrachten wir die Möglichkeiten, die man beispielsweise mit Datenbank-Links hat, wirkt der Oracle Big Data Connector (OSCH) ungewöhnlich sperrig. Einerseits ist er auf registrierte, einzelne Tabellen beschränkt – klar, bei Zugriffen über External Tables. Insbesondere bietet er aber keinerlei Optimierung für verteilte Abfragen. Die Hadoop-basierten Dateien werden bei jeder Abfrage immer vollständig auf die Oracle-Datenbank gestreamt und erst dort in die übliche Abfrage-Optimierung eingebunden. Drittens ist es mit OSCH ziemlich aufwändig, die External Tables zu erstellen.

Die bessere Oracle-Lösung dafür lautet „Big Data SQL". Mit diesem Mechanismus können – ähnlich wie in den Exadata Cell Nodes – Smart Scans innerhalb von Hadoop ausgeführt werden. Damit sind auch die Übergabe von Filtern nach Hadoop und die Einschränkung der zurückgegebenen Spalten (Projektion) oder auch effiziente Joins via Bloom-Filter möglich. Oracle benötigt dafür allerdings eigene Komponenten auf dem Hadoop-Cluster und einen speziellen SQL-Agent, der für die Datenbank beispielsweise den Austausch der Hadoop-Metadaten sicherstellt. Dazu kommt der Vorteil, dass das Erstellen von External Tables mit diesem Werkzeug nicht komplizierter sein muss als das Erzeugen normaler External Tables auf normale Dateien im direkten Zugriff der Datenbank.

Klingt im Ansatz also sehr gut, hat aber – neben den Lizenzkosten – ein besonders unangenehmes Manko: Big Data SQL ist aktuell nur auf der Kombination Exadata/Big Data Appliance verfügbar. In der Hoffnung auf baldige Verfügbarkeit des Features auch für anderen Konstellationen – das Statement-of-Direction mag man in diese Richtung interpretieren können – betrachten wir die Lösung im Folgenden dennoch etwas genauer *(siehe Abbildung 4)*.

Nachdem Big Data SQL auf DB-Seite konfiguriert wurde – und damit die Hadoop-Locations in Oracle-Directories gemappt sind – ist es recht einfach, HDFS-basierte Dateien als External Tables mit nur einem Befehl einzubinden. Das Beispiel in *Listing 3* ermöglicht das Lesen einer Log-Datei mit genau einer Spalte im JSON-Format).

Wichtig dabei sind zwei neue External-Table-Typen namens „ORACLE_HDFS" und „ORACLE_HIVE". Erstere für den direkten Zugriff auf alle möglichen Dateien in HDFS, die zweite für den Zugriff auf

```
CREATE TABLE movielog
   (click VARCHAR2(4000))           -- Für ganze Applog Zeile
   ORGANIZATION EXTERNAL            -- External Table
   (TYPE ORACLE_HDFS                -- Neuer Typ ORACLE_HDFS
    DEFAULT DIRECTORY DEFAULT_DIR
    LOCATION ('/user/oracle/dat/')  -- Dateien auf Hadoop Cluster
   );
```

Listing 3

```
CREATE TABLE movieapp_log (                -- Spaltenangeben
      custid  INTEGER,       movieid       INTEGER ,         genreid INTEGER ,
      time    VARCHAR2(20),  recommended   VARCHAR2(4),      activity NUMBER,
      rating  INTEGER,       price         NUMBER )          ORGANIZATION EXTERNAL
   (TYPE ORACLE_HIVE                       -- Neuer Typ ORACLE_HIVE
ACCESS PARAMETERS (                        -- Spalten wie in Hive
                                           -- oder freies Mapping möglich
      com.oracle.bigdata.tablename: log_db.applog
      com.oracle.bigdata.colmap: {"col":"CUSTID", field":"cust_id"}
          ... )
   );
```

Listing 4

Hive-Tables direkt via HCatalog-Metadaten. *Listing 4* zeigt ein Beispiel dafür.

Nun sind deutlich intelligentere Zugriffe auf Hadoop-basierte Daten möglich und auch einer Einbindung in verteilte Datenbank-Abfragen steht nichts mehr im Wege – außer natürlich den üblichen, schon bei Database-Links bekannten Distributed-Query-Problemen.

Oracle Database Gateway

Was bisher auffällt, ist die praktisch völlige Absenz von Hadoop-eigenen Abfrage-Mechanismen und Tools, außer HCatalog für die Metadaten. Bei allen betrachteten Tools nutzt Oracle fast ausschließlich eigene Programme für den Zugriff auf die Daten. Damit sind ein sauberes Typen-Mapping und eine zentrale Kontrolle über die Kompatibilität möglich. Eine andere Möglichkeit – und die ist interessanterweise sogar weitgehend kostenfrei, wenn man von den Lizenzen für ODBC-Treiber absieht – ist der Einsatz des „Generic ODBC Gateway". Früher auch bekannt als „Heterogenous Services", bietet Oracle schon lange die Möglichkeit, via Datenbank-Links auch auf andere Datenbanken als auf die von Oracle zuzugreifen. Die darunter versammelten Lösungen sind fast alle kostenpflichtig – bis auf die Variante für „GENERIC ODBC", bei der beliebige ODBC-Treiber für den Zugriff auf beliebige Datenquellen eingesetzt werden können (sogar auf Microsoft Excel).

Um es klar zu sagen: Man wird sich einige Limitierungen bei Kompatibilität und Performance einhandeln. Dennoch ist dieser Ansatz – gerade für erste Schritte, Evaluationen oder bei nur vereinzelten, selektiven Zugriffen auf die Hadoop-Welt – durchaus einen Blick wert. Folgende Schritte sind auf dem Datenbankserver (und/oder einem extra Gateway Server) auszuführen:

1. Passenden OBCD-Treiber installieren und konfigurieren
2. ODBC-Gateway installieren
3. „listener.ora" und „tnsnames.ora" anpassen
4. In „ORACLE_HOME\hs\admin" eine neue Datei erzeugen
5. Public-Database-Link erzeugen

Gehen wir von einem „HIVE ODBC"-Treiber aus. In diesem Fall wird die SQL-Engine von Hive selbst in die Abfragen eingebunden. Man hat also nach der Einrichtung des Datenbank-Links freien Zugriff auf alle Hive-Tabellen, auf die man als Hive-User Zugriff hat – nicht anders als bei normalen Datenbank-Links – und nutzt die Rechenleistung des Hadoop-Clusters via Hive. Sogar die Metadaten kann man direkt über das Oracle Data Dictionary nutzen. Befehle wie die in *Listing 5* funktionieren also oft problemlos, während die Last der Zugriffe und der Ort der Metadaten außerhalb der Oracle-Datenbank verortet sind.

Nicht jede Hadoop-Datenquelle oder jeder ODBC-Treiber funktioniert allerdings gleich gut. Mit der In-Memory-SQL-Lösung für Hadoop „Impala" von Cloudera und dem dazugehörigen ODBC-Treiber sind beispielsweise heute nur wenige Metadaten verfügbar, während mit Apache Hive und den dafür verfügbare Treibern aktuell schon sehr viele Metadaten-Informationen ermittelt werden können. Hier heißt es: Gut evaluieren.

Oracle Data Integrator

Für den ODI gibt es eine Big-Data-Option (separate Lizenz), die für Mappings nativen Code für die Hadoop-Werkzeuge „Pig" (eine Scripting Engine, die ihrerseits wieder MapReduce-Code generiert), „Spark" (eine modernere und schnellere Map-Reduce-Alternative) und „Oozie" (das zentrale Workflow-Tool in der Hadoop-Welt) generiert.

Darüber hinaus erlaubt es die Option, entweder den traditionellen ODI-Agent oder Apache Oozie als Orchestrierungs-Engine für ODI einzusetzen. Sie bringt einen schnellen WebLogic Hive JDBC Driver und zahlreiche Direct Load KMs (LKM = Loading Knowledge Modules, kombiniert mit anderen KMs in einem Mapping einsetzbar) etwa für Sqoop mit.

Nutzt man die Oracle Big Data Connectors, erhält man damit auch die Lizenz für Oracle Data Integrator Application Adapter for Hadoop, mit dem die Konnektoren über entsprechende Knowledge-Module in ODI eingebunden werden können. Gleiches gilt für Big Data SQL. Auch hier ist eine Integration in ODI verfügbar – wenn man die entsprechenden Systeme und Lizenzen hat.

Oracle Golden Gate

Oracle Golden Gate for Big Data unterstützt Golden-Gate-Replikation von allen bekannten Golden-Gate-Quellen (Oracle,

```
SELECT COUNT(*) FROM mytab@hive WHERE cust_cat = 95;
SELECT table_name FROM all_tables@hive;
```

Listing 5

DB2, SQL Server, MySQL, Informix, Sybase ASE, SQL/MX, JMS etc.) in die Big-Data-Welt – allerdings nicht in die Gegenrichtung. Mit Change-Data beliefert werden Apache Flume, HDFS direkt, Hive und die Hadoop-eigene NoSQL-Datenbank HBase. Über die ebenfalls in der Lizenz enthaltene „Golden Gate for Java"-Lösung ist auch die Integration der Ziele Oracle-NoSQL-Datenbank, Apache Kafka, Apache Storm und Apache Spark möglich.

Fazit

Die Zusammenarbeit mit Big-Data-Plattformen steht im Zentrum der heutigen Oracle-Big-Data-Aktivitäten. Die entsprechenden Oracle Big Data Connectors (OLH, OSCH, OXH etc.) sind dafür schon länger verfügbar, allerdings nur für bestimmte Anforderungen geeignet und auch nicht unbedingt einfach zu benutzen. Oracle Big Data SQL vereinfacht und beschleunigt die Konnektivität deutlich, ist äußerst vielversprechend – aber lizenzmäßig extrem limitiert und daher leider für die meisten Kunden heute irrelevant.

Nicht von Oracle propagiert, aber dennoch interessant ist Oracle Generic ODBC Database Gateway als kostengünstige und einfache Option für Big-Data-Plattform-Verbindungen ohne besondere Ansprüche an Durchsatz und Kompatibilität.

ODI bietet in den neuen Versionen ebenfalls zahlreiche KMs rund um Hadoop & Co. Mit der Big-Data-Option ist sogar die Nutzung von Hadoop als Daten-Integrationsplattform greifbar geworden, weil Mappings auch direkt auf Hadoop-Distributionen ausführbaren Code generieren, der früher der relationalen Welt vorbehalten war. Auch Golden Gate mit Hadoop & Co. als Replikationsziel erweitert sein Repertoire in Richtung Big-Data-Technologien.

Aus Sicht eines Data-Warehouse-Entwicklers offenbaren sich mit der Annäherung beider Welten neue Möglichkeiten für daten- und kostengerechtes Processing, Data Lifecycle Management (etwa Archive auf Hadoop), Self Service BI und vieles mehr. Nicht zuletzt blicken wir gespannt in die Zukunft und warten auf Lösungsansätze rund um die Oracle-Vision eines Big-Data-Management-Systems.

Peter Welker
peter.welker@trivadis.com

Ein großer Schritt für die Oracle-Lizenzierung in virtuellen Umgebungen

In einem gemeinsamen Gespräch der DOAG, SOUG (Swiss Oracle User Group) und AOUG (Austrian Oracle User Group) am 18. November 2015 auf der DOAG 2015 Konferenz + Ausstellung mit Andrew Mendelsohn, Executive Vice President Database Server Technologies bei Oracle, wurde deutlich, dass es im Bereich der lange diskutierten Lizenzierungsproblematik hoffungsvolle Fortschritte gibt.

Bisher ist es je nach eingesetzter Hypervisor-Version vorgeschrieben, den Cluster, alle von der Management-Konsole erreichbaren Host oder die gesamte virtuelle Infrastruktur zu lizenzieren. Bei VMware wäre das bis einschließlich Version 5.0 der Cluster, ab Version 5.1 das vCenter und ab Version 6.0 die gesamte virtuelle Infrastruktur. Laut Mendelsohn gibt es Anzeichen dafür, dass die Segmentierung der Hypervisor- oder VM-Umgebung mit VLAN-Technologie zunehmend akzeptiert wird.

Eine Segmentierung der virtuellen Netze in logische Gruppen hat den Vorteil der Trennung der Umgebungen bei gleichzeitiger Beibehaltung von Datenkommunikation. Das VLAN kann somit als Begrenzung des lizenzpflichtigen Bereichs angenommen werden, so dass nur die Host-Server innerhalb des VLANs lizenziert werden müssen, da es mit den derzeit bekannten Hypervisor-Technologien nicht möglich ist, virtuelle Maschinen im Betrieb über die Grenze des VLANs hinaus zu verschieben.

Die VLAN-Lösung gilt sinngemäß für alle Hypervisor-Anbieter, die das Verschieben virtueller Maschinen im Betrieb über Clustergrenzen hinaus möglich machen. Derzeit muss dies auf dem Wege eines Einzel-Approvals durch Oracle genehmigt werden. Dies ist noch nicht die von den Usergroups und von vielen Kunden und Partnern erwartete allgemeingültige Lösung.

Die DOAG, SOUG und AOUG begrüßen diesen wichtigen Schritt aber ausdrücklich und ermutigen ihre Mitglieder, von dieser Möglichkeit Gebrauch zu machen. Allerdings raten sie dazu, das Verhalten der virtuellen Maschinen mit System Logs zu dokumentieren und zu archivieren. Die Usergroups wünschen sich jedoch noch immer eine einheitliche und allgemeingültige Regelung.