Neu: Oracle NoSQL DB 3.0

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

Seit April 2014 steht die neue Version 3.0 zum Download bereit. Neue Funktionen wie ein „Tabellen-API“ lassen aufhorchen: Tabellen in einer NoSQL DB? Dieser Artikel gibt einen Überblick über die neuen Funktionen – neben dem Tabellen-API gibt es neue Security-Funktionen und Erweiterungen im Cluster-Management.

Die neue Version führt eine Tabellen-Schnittstelle in der NoSQL DB ein. Das bedeutet aber nicht, dass aus der NoSQL- nun eine SQL-Datenbank wird, vielmehr bekommt der Entwickler eine Schnittstelle zur Definition von Tabellenstrukturen – also zum Lesen und Schreiben von Daten in Form von Zeilen und Spalten. Alle Vorgänge werden dabei auf die Key-Value-Technologie abgebildet; die ab Version 2 unterstützten „Avro Schemata“ können ebenfalls zur Definition von Tabellen verwendet werden. *Abbildung 1* zeigt die Zusammenhänge zwischen Tabellenmodell, AVRO-Schemata und Key-Value-Paaren.

Die Definition von Tabellen erfolgt mithilfe des Command Line Interface (CLI) der NoSQL DB, mit der (wie in den vorherigen Versionen) die NoSQL-Datenbank initial aufgesetzt wird. Wie dieser Setup im Detail aussieht, soll hier nicht vertieft werden, die Dokumentation (siehe „Weitere Informationen“) gibt hierzu Auskunft. *Listing 1* zeigt, wie eine Tabelle mit dem CLI erstellt wird.

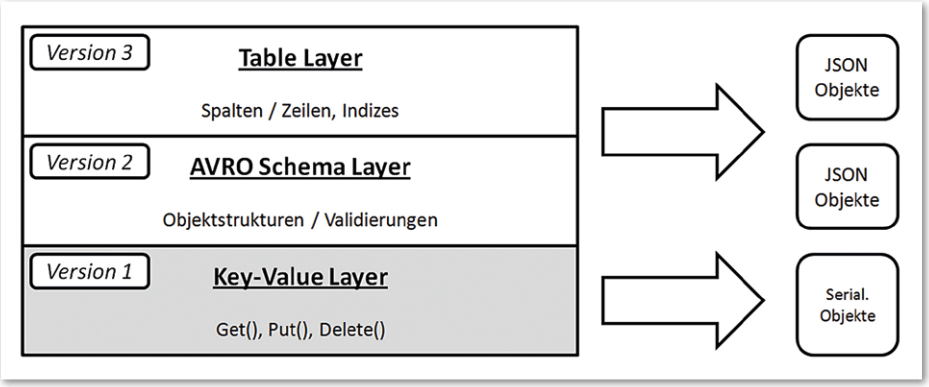


Abbildung 1: Auch das neue Tabellenmodell speichert die Daten tatsächlich als Key-Value-Paare ab

```
kv-> table create -name EMP
EMP-> add-field -type INTEGER -name empno
EMP-> add-field -type STRING  -name ename
EMP-> add-field -type DOUBLE  -name sal
EMP-> add-field -type STRING  -name job
EMP-> primary-key -field empno
EMP-> exit
Table EMP built.

kv-> plan add-table -name EMP -wait
Plan 19 ended successfully
```

Listing 1: Definition einer Tabelle „EMP“ in der NoSQL DB

Es können numerische, alphanumerische und strukturierte Datentypen in Form von Records oder Arrays verwendet werden – „DATE“ oder „TIMESTAMP“ stehen jedoch noch nicht zur Verfügung. Der Primärschlüssel ist Pflicht, er wird logischerweise für den Key der als Key-Value-Paare gespeicherten Daten verwendet. Tabellen können auch in einer Parent-Child-Beziehung miteinander verknüpft sein. Das wird so abgebildet, dass der Primärschlüssel der Parent-Tabelle in die Child-Tabelle(n) übernommen wird. Sobald die Tabelle erstellt wurde, können Zeilen abgelegt werden – dies kann (wie immer bei der NoSQL DB) per Java-Programm geschehen *(siehe Listing 2)*. Alternativ kann auch mit dem CLI gearbeitet werden *(siehe Listing 3)* – die Tabellenzeile wird dann mit dem Befehl „put table“ im JSON-Format übergeben.

Analog dazu kann man mit dem Kommando „get table“ Zeilen lesen. Der Zugriff erfolgt hier allerdings allein über den Primärschlüssel oder (wie sich noch zeigen wird) über per „Secondary Index“ indizierte Spalten. Die freie Selektion, wie bei SQL-Datenbanken üblich, ist in der NoSQL-Datenbank nach wie vor nicht möglich.

Wie *Listing 4* zeigt, ist der Zugriff auf eine konkrete Tabellenzeile zunächst nur mit dem Primary Key möglich – man erkennt sofort, dass der „Key Value Store“ der NoSQL DB die technische Grundlage ist. Allerdings ist das noch nicht alles: Mit den neu eingeführten „Secondary Indexes“ wurde eine Möglichkeit geschaffen, auch über andere Spalten als den „Primary Key“ zu suchen. *Listing 5* zeigt, wie ein „Secondary Index“ erzeugt und genutzt wird.

```
import oracle.kv.*;
import oracle.kv.table.*;

public class write {
   public static void main(String args[]) {
       // Verbindung zur NoSQL DB öffnen
       KVStore store= KVStoreFactory.getStore (
           new KVStoreConfig(
               "workshopstore"
               new String[] {"sccloud031:10100", "sccloud031:10200"}
           )
       );
       // Handle zur definierten Tabelle EMP holen
       TableAPI tableH = store.getTableAPI();
       Table myTable = tableH.getTable("EMP");
       Row row = myTable.createRow();
       row.put("empno", 7839);
       row.put("ename", "MILLER");
       row.put("sal", 3000d);
       row.put("job", "SALESMAN");
       // Tabellenzeile speichern
       tableH.put(row, null, null);
       store.close();
   }
}
```

Listing 2: Das Java-Programm speichert eine Tabellenzeile in die NoSQL DB

```
kv-> connect store -host localhost -port 10100 -name workshopstore
Connected to workshopstore
Connected to workshopstore at localhost:10100.
kv-> put table
 -name EMP
 -json ,{"empno":815, "ename":"JONES", "sal": 3500, "job":"MANAGER"}`

Operation successful, row inserted.
```

Listing 3: Tabellenzeilen mit dem CLI in die NoSQL DB speichern

```
kv-> get table -name EMP
{"empno":7844,"ename":"FORD",  "sal":2500.0,"job":"CLERK"}
{"empno":815, "ename":"JONES", "sal":3500.0,"job":"MANAGER"}
{"empno":7839,"ename":"MILLER","sal":3000.0,"job":"SALESMAN"}

3 rows returned.

kv-> get table -name EMP -field "empno" -value 7844
{"empno":7844,"ename":"FORD","sal":2500.0,"job":"CLERK"}

kv-> get table -name EMP -field "ename" -value "SMITH"
Error handling command get table -name EMP -field "ename" -value
"SMITH": Field is not part of PrimaryKey: ename
```

Listing 4: Tabellenzeilen mit dem CLI aus der NoSQL DB abrufen

Die technische Grundlage eines „Secondary Index" ist einfach: Beim Erstellen generiert die NoSQL DB zusätzliche Key-Value-Paare, mit denen der Index modelliert wird. Dabei dienen die Inhalte der indizierten Spalte als Key – der „Primary Key" der indizierten Zeile ist dagegen der „Value". Beim Abrufen von Zeilen wird folgerichtig zunächst mithilfe dieses Index der Primärschlüssel und damit erst die konkreten Daten abgerufen. Da die NoSQL DB keinen Query-Optimizer besitzt, muss der „Index Lookup" von der Anwendung explizit angefordert werden – in *Listing 4* ist das am Parameter „-index" erkennbar.

Die in den Listings 4 und 5 vorgestellten Möglichkeiten lassen sich natürlich auch per API aus Java-Programmen heraus nutzen. Auf die bereits erwähnten „Child Tables" oder Array-Datentypen soll hier aus Platzgründen nicht mehr im Detail eingegangen werden – die Dokumentation der NoSQL DB enthält dazu weitere Informationen.

Security-Funktionen

Die bisherigen Versionen der Oracle NoSQL DB enthielten keinerlei Security-Funktionen; weder war ein Login zur Arbeit mit der NoSQL DB vorgesehen, noch wurden die Daten auf dem Transportweg in irgendeiner Art und Weise verschlüsselt.

Die vorliegende Version 3.0 ändert dies: Die Kommunikation über das Netzwerk kann nun mit SSL verschlüsselt und der Zugang zur NoSQL DB per Login geschützt werden. Eine bestehende NoSQL-DB-Installation lässt sich nachträglich sichern, indem per Admin-Werkzeug eine Security-Konfiguration hinzugefügt wird – das geschieht, grob skizziert, mit folgenden Schritten:

- Herunterfahren aller NoSQL-DB-Clusterknoten
- Erstellen einer neuen Security-Konfiguration auf einem Knoten per CLI
- Kopieren dieser Konfiguration (Verzeichnis „security") auf alle anderen Knoten
- Aktivieren der Konfiguration auf allen Knoten per CLI
- Starten aller Knoten
- Anonymer initialer Login
- Erstellen des ersten ADMIN-Users

Danach sind Zugriffe nur noch nach erfolgreichem Login möglich. Beim Erstellen der Sicherheitskonfiguration wird ein „External Password Store" festgelegt – für die Enterprise Edition kann und soll-

```
kv-> plan add-index -name idx_emp_ename -table EMP -field ename
Started plan 19. Use show plan -id 19 to check status.
      To wait for completion, use plan wait -id 19
kv-> plan wait -id 19
Plan 19 ended successfully

kv-> get table -name EMP -field ename -value FORD
Error handling command get table -name EMP -field ename -value FORD: Field is not part of PrimaryKey: ename
kv-> get table
  -name EMP -field ename -value FORD -index idx_emp_ename
{"empno":7844,"ename":"FORD","sal":2500.0,"job":"CLERK"}
```

Listing 5: Erstellung und Nutzung eines Secondary Index in der NoSQL DB

te ein Oracle-Wallet verwendet werden; die Community-Edition nutzt eine Passwortdatei. Wichtig ist, dass dort nicht die Passwörter der in der NoSQL DB eingerichteten User abgelegt sind, sondern vielmehr das Passwort zum Zugriff auf den Java-Keystore – der wiederum enthält die Schlüssel für die SSL-Kommunikation. Die User-Verwaltung, also das Anlegen und Verwalten von Nutzerkonten, findet mit dem Admin-CLI statt. *Listing 6* zeigt das Erzeugen eines neuen Users („SCOTT“).

Zugriffe auf die NoSQL DB erfordern nun ein Login – das Java-Programm in *Listing 2* würde nun so nicht mehr funktionieren. *Listing 7* zeigt die zur Verbindung auf eine gesicherte NoSQL DB erforderlichen Java-Codeabschnitte.

In der aktuellen Version 3.0 dienen Usernamen allein dem Login: Ein Rechtekonzept, nach dem unterschiedliche User unterschiedliche Dinge tun dürfen, gibt es (noch) nicht. Ein Login ist nun auch erforderlich, wenn Daten, wie schon in *Listing 3* dargestellt, mit dem CLI geschrieben oder gelesen werden sollen *(siehe Listing 8)*.

Weitere neue Funktionen

Neben dem Tabellen-API und den Security-Funktionen sind noch weitere neue Features im Release enthalten:

- Die Knoten eines NoSQL-DB-Clusters können in verschiedene Zonen unterteilt sein. Knoten in der „Primary Zone“ arbeiten wie bisher – sie können als Master und als Replika fungieren. Fällt ein Master aus, so wählen die Replikas einen neuen Master. Knoten in der „Secondary Zone“ fungieren dagegen nur

```
$ java -jar /opt/oracle/nosqldb/sw/lib/kvstore.jar runadmin
   -host localhost -port 10100
   -security KVROOT1/security/client.security

Could not login as anonymous: Authentication failed (12.1.3.0.5)
Login as:root
root's password: ******
Logged in admin as root
Redirecting to master at rmi://sccloud031:10300

kv-> plan create-user -name SCOTT -password tiger -wait
Executed plan 21, waiting for completion...
Plan 21 ended successfully
```

Listing 6: Einrichtung eines neuen Nutzerkontos in der NoSQL DB

```
:
KVStoreConfig conf = new KVStoreConfig(
 "workshopstore",
 new String[] {"sccloud031:10100", "sccloud031:10200"}
);

// SSL-Kommunikation einschalten
Properties secProps = new Properties();
secProps.setProperty(
 KVSecurityConstants.TRANSPORT_PROPERTY,
 KVSecurityConstants.SSL_TRANSPORT_NAME
);
secProps.setProperty(
 KVSecurityConstants.SSL_TRUSTSTORE_FILE_PROPERTY,
 "/opt/oracle/nosqldb/ex/client.trust"
);
conf.setSecurityProperties(secProps);

// konkrete Verbindung mit Authentifizierung
KVStore store= KVStoreFactory.getStore (
 conf,
 new PasswordCredentials("SCOTT", new String("tiger").toCharArray()),
 null
);
:
```

Listing 7: Java-Verbindung zur NoSQL DB: Passwort-Login und SSL-Kommunikation

```
kv-> connect store -host localhost -port 10100 -name workshopstore
-security KVROOT1/security/client.security
Login as:SCOTT
SCOTT's password:******
Connected to workshopstore
Connected to workshopstore at localhost:10100.

kv-> get table -name EMP
{"empno":8000,"ename":"MILLER","sal":3000.0,"job":"SALESMAN"}
{"empno":7844,"ename":"FORD","sal":2500.0,"job":"CLERK"}
:

4 rows returned.
```

Listing 8: Arbeit mit dem CLI auf einer gesicherten NoSQL-DB-Umgebung

als Replika – auch wenn ein Master ausfällt, nehmen sie nicht an der Wahl des neuen Masters teil.

- Die neue Lesekonsistenz-Stufe „NONE_REQUIRED_NO_MASTER“ erzwingt das Lesen von einer Replika. Das ist sinnvoll, wenn die Ressourcen der Master-Knoten (obwohl sie zu einem Zeitpunkt theoretisch frei wären) dennoch (etwa für andere Zugriffe) geschützt werden sollen.

Fazit

Das neue Release 3.0 bringt die Oracle NoSQL DB einen großen Schritt nach vorn. Das Tabellen-API erlaubt das Hinterlegen von Datenstrukturen – die Praxis hat in vielen Fällen schon gezeigt, dass es ohne diese nicht geht – auch nicht in der Welt der NoSQL-Datenbanken. Damit können sich Entwickler auf eine Tabellenstruktur einigen, diese kontrolliert pflegen und weiterentwickeln. „Secondary Indexes“ erlauben einfache Zugriffe über ein Attribut, das nicht als Schlüssel definiert wurde – diese fertige Funktionalität spart große eigene Programmieraufwände. Dennoch bleibt die Datenbank eine NoSQL-Datenbank – ein Query Optimizer existiert nicht und das freie Formulieren von Abfragen ist nicht möglich.

Security-Funktionen sind im Unternehmenseinsatz ein Muss – das schließt sowohl die verschlüsselte Kommunikation als auch den Schutz der Installation durch Authentifizierung ein. Beides wird mit dem vorliegenden Release eingeführt.

Wer NoSQL-Technologien einsetzt oder sich damit beschäftigt, sollte sich die neue Oracle NoSQL DB auf jeden Fall näher ansehen – dem Produkt ist deutlich anzumerken, dass es für den Unternehmenseinsatz im Rechenzentrum gewappnet ist.

Weitere Informationen

[1] Download Oracle NoSQL DB
http://www.oracle.com/technetwork/products/nosqldb/downloads/index.html

[2] Oracle NoSQL DB Dokumentation
http://docs.oracle.com/cd/NOSQL/html/index.html

[3] Oracle NoSQL DB 3.0: Complete List of Changes
http://docs.oracle.com/cd/NOSQL/html/changelog.html

[4] Oracle NoSQL DB Frequently Asked Questions
http://www.oracle.com/technetwork/database/nosqldb/overview/nosqldb-faq-518364.html

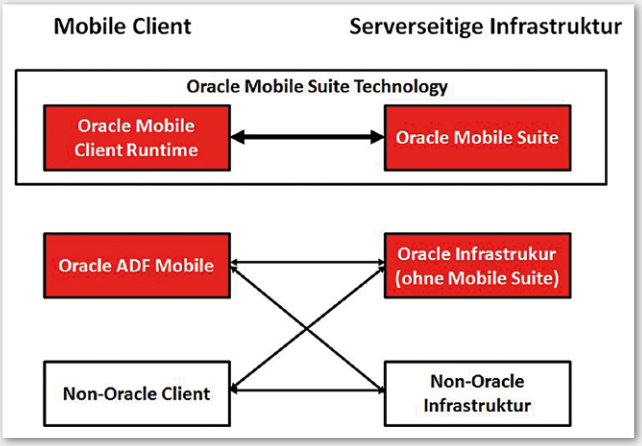
Carsten Czarski
Carsten.Czarski@oracle.com
http://twitter.com/cczarski
http://sql-plsql-de.blogspot.com

Artikel „Neue Oracle Product Suites für mobile Anwendungen“ von Dr. Jürgen Menge, ORACLE Deutschland B.V. & Co. KG, in der letzten Ausgabe auf Seite 22

Leider haben sich in dem Artikel Fehler eingeschlichen, für die sich der Autor entschuldigt.

Die Oracle Mobile Suite besteht aus folgenden Komponenten:

- Oracle Service Bus
- Oracle Applications Adapter
- Technologie-Adapter



Die Abbildung zeigt die korrekten Zusammenhänge zwischen den Produkten.