Event-Processing und Stream-Analyse

Arne Brüning, ORACLE Deutschland B.V. & Co. KG

Industrie 4.0 oder das Internet der Dinge, egal aus welcher Perspektive man die Aufgabe betrachtet: Die echtzeitnahe Analyse von hochvolumigen Daten, das Durchsuchen nach relevanten Mustern und die zeitnahe Reaktion sind üblicherweise Bestandteil eines jeden Szenarios. Die Werkzeuge dafür sind der Kinderstube entwachsen und die zweite Generation der Event-Processing-Tools betritt die Bühne. Auch bei Oracle ist eine neue Generation am Start und wirbt auf der einen Seite mit erweiterten Fähigkeiten wie Geo-Processing und Machine-Learning. Auf der anderen Seite wurde die Administrier- und Benutzbarkeit verbessert. Dieser Artikel gibt einen Überblick und eine Einführung in das aktuelle Oracle-Portfolio.

Grundsätzlich handelt es sich bei Lösungen zur Stream-Analyse um Systeme, die große Mengen an Daten aus verschiedenen Quellen zur Laufzeit nahezu in Echtzeit filtern, aggregieren und nach bestimmten Mustern durchsuchen können. Die Resultate werden dann an Geschäftsanwendungen weitergeleitet. Laut dem Beratungsunternehmen Forrester Research *[1]* besteht eine Streaming-Analytics-Lösung aus den folgenden Bausteinen:

- *Transformation*
  Die hochvolumigen Datenströme können zu einem großen Teil nicht benötigte Daten („noise“) und teilweise nur zu einem kleinen Teil Nutzdaten („signal“) enthalten. Über geeignete Filtermechanismen filtern die Systeme die anfallenden Daten entsprechend, um eine Konzentration auf die relevanten Anteile zu ermöglichen. In dieser Phase lassen sich auch fehlerhafte Daten aussortieren. Meldet beispielsweise ein Temperatursensor für ein paar Millisekunden eine um einige Hundert Grad erhöhte Temperatur, wird es sich wahrscheinlich um einen Messfehler handeln.

- *Correlation*
  Nachfolgend werden die erhobenen Daten mit anderen Datenströmen in Bezug gebracht. In einem Verkehrsdaten-System lässt sich so aus den Daten der einzelnen Sensoren eine Verkehrslage erstellen.

- *Enrichment*
  Zusätzlich zur Korrelation der Messdaten untereinander kann eine Anreicherung durch zusätzliche Daten von verschiedenen Datenbanken nützlich sein. Beispielsweise lassen sich die erhobenen Daten des Netzwerk-Monitorings eines Telekommunikations-Providers gegen eine Datenbank mit vorhandenen Netzwerk-Kapazitäten abgleichen.

- *Time Windows*
  Wichtig ist die zeitliche Komponente der Events, weshalb diese als zeit-

liche Ströme vorgehalten werden. Im Finanzwesen ist zum Beispiel bei der Chart-Analyse das sogenannte „W-Pattern“ von Belang. Dafür ist wichtig zu wissen, dass innerhalb eines bestimmten Zeitraums ein Kurs fällt, steigt, wieder fällt und wieder steigt, sodass der Graph eine „W“-Form annimmt. Rein durch die aktuellen Werte ist dies nicht zu ermitteln, deshalb ist die zeitliche Komponente notwendig.

- *Pattern Matching*
  Ist die beschriebene Vorverarbeitung getan, lässt sich gezielt nach Mustern suchen. Diese sind von der fachlichen Anwendung abhängig und die durchlaufenden Daten werden ständig nach den definierten Mustern durchsucht. Dabei kann auch das Ausbleiben eines Events ein wichtiges Muster sein. Wird am Flughafen ein Koffer an einem Fließband aufgegeben und taucht dieser nicht nach einer definierten Zeit am geplanten Ausgang wieder auf, ist er vermutlich irgendwo vom Band gefallen.

- *Business Logic*
  Der eigentliche Wert der Streaming Analysis entsteht, wenn Muster gefunden und infolgedessen Geschäftsanwendungen informiert werden, sodass diese in geeigneter Form darauf reagieren. So kann etwa das Verkehrsleitsystem auf einen Stau reagieren und die dynamische Verkehrsführung ändern, das gefundene W-Pattern veranlasst das Trading-System, dem Analysten eine Empfehlung für eine Finanz-Transaktion zu geben, oder der verlorene Koffer wird gesucht.

Gerade im Umfeld der ausgerufenen Industrie 4.0 ist hier am Industriestandort Deutschland eine gesteigerte Nachfrage nach Systemen zur Predictive Maintenance zu beobachten. Dies mag daran liegen, dass es sich hierbei nicht um ein abstraktes Forschungsprojekt, sondern um eine Maßnahme zur kurzfristigen Kosteneinsparung handelt. Die Erwartung ist, über ein Stream-Analyse-System einen bevorstehenden Maschinenausfall besser als bisher vorhersagen zu können. Wenn dann ein Wartungstechniker das System zeitnah reparieren kann, lassen sich Ausfallzeiten und gegebenenfalls Folgeschäden vermeiden, was direkt einen Kostenvorteil ergibt. Da die Ausfall-Szenarien durchaus komplex sein können, wird diese Analyse aber nicht alleine von einem an die Maschine angeschlossenen Steuer-PC durchgeführt. Oftmals sind weitere Daten erforderlich, etwa Wetterdaten, Daten von anderen Systemen oder auch Planungsdaten. Daher ist ein zentrales System notwendig, das die Anforderungen gemäß obiger Definition erfüllt.

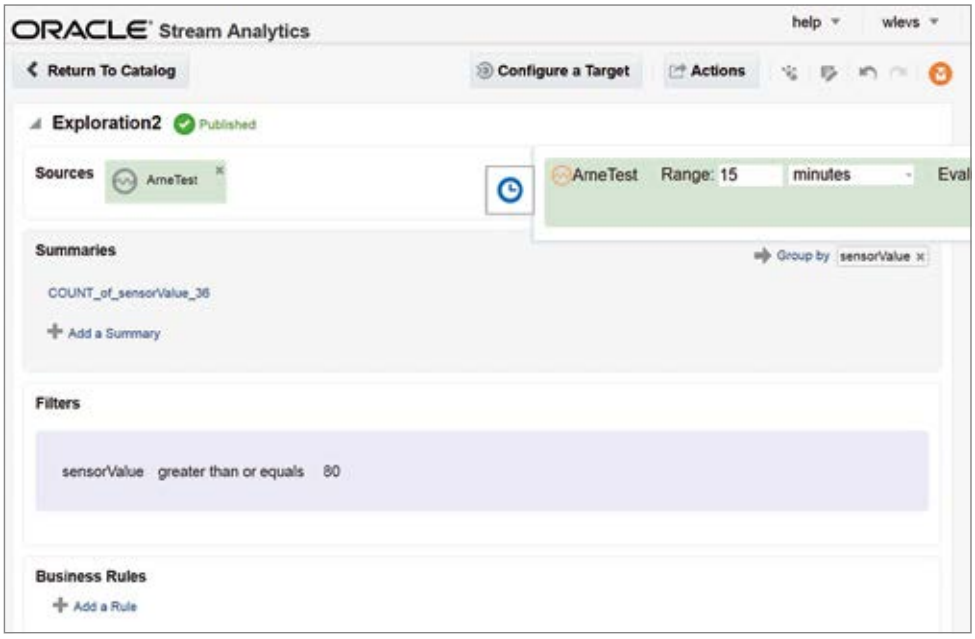

Abbildung 1: Live-Operationen auf den Stream

Streaming Analytics mit Oracle

Event-Processing-Produkte haben bei Oracle eine lange Historie. Das Produkt Complex Event Processing (CEP) wurde bereits zu BEA-Zeiten entwickelt, kam durch die Übernahme zu Oracle und wurde hier später in Oracle Event Processing (OEP) umbenannt. Entsprechend der langen Zeit am Markt ist auch die Liste der Referenzkunden. So nutzt zum Beispiel Japans größte Telefongesellschaft NTT OEP für das Netzwerk-Monitoring.

Auch schon sehr lange ist OEP Embedded verfügbar, das auf kleinen Devices wie dem Raspberry Pi läuft. Da die Embedded-Variante weitestgehend identisch mit der Server-Variante ist, können auf dem Device die gleichen Event Processing Networks ablaufen wie auf dem Server. Ausgenommen sind lediglich ein paar komplexe Operationen, mit denen die Device-Hardware überfordert wäre.

Intern setzt OEP auf Frameworks wie Spring und OSGi auf. Sie werden genutzt, um OEP extrem leichtgewichtig umzusetzen. Die Kehrseite war bislang: Zur Nutzung von OEP waren Programmierkenntnisse und Kenntnisse der Frameworks notwendig. In der Praxis hat sich aber gezeigt, dass die Algorithmen zur Pattern-Suche oftmals noch gar nicht fertig sind und eher explorativ entwickelt werden. Diese Algorithmen werden nicht von Entwicklern, sondern von den Experten der jeweiligen fachlichen Domänen entwickelt. Daher hat Oracle den Stream Explorer, heute Stream Analytics, entwickelt. Einmal richtig aufgesetzt, können damit auch Anwender ohne Programmierkenntnisse die entsprechenden Analysen interaktiv erstellen.

Oracle Stream Analytics 12.2.1

Das aktuelle Server-Produkt für Event-Processing und Stream-Analyse ist OSA Version 12.2.1. Voraussetzung sind laut Preisliste Coherence Enterprise Edition oder die WebLogic Suite, die Coherence wiederum enthält. Die Installation ist sehr einfach. Der Download enthält wie bei Oracle üblich eine „.jar“-Datei. Diese wird einfach mit „java -jar /media/sf_Share/OSA/fmw_12.2.1.0.0_osa_generic.jar“ aufgerufen, um den Universal Installer zu starten.

Neben dem Zielverzeichnis kann man als einzige Option auswählen, ob die Beispiele dazu installiert werden sollen.

Wie vom WebLogic Server bekannt, muss als Nächstes eine Domain angelegt werden. Dies erfolgt wie üblich über das „config"-Skript, unter Linux beispielsweise mit „oep/common/bin/config.sh". Die wenigen Optionen sind auch hier schnell geklärt: Domain erstellen oder ändern, mit oder ohne Beispielen, Password für Administrator und Keystore – eine Minute später steht die Domain bereit. Gestartet wird der Server dann per „startwlevs.sh" aus dem Domainverzeichnis.

Überprüft werden kann der laufende Server, wenn man per Browser den bei der Installation eingegebenen Port (Standard: „9002") öffnet. War die Installation erfolgreich, wird man auf die URL „*/wlevs*" umgeleitet. Dort meldet sich der Event Processing Visualizer. Ändert man die URL auf „/sx", gelangt man auf die Hauptseite von Oracle Stream Analytics *(siehe Abbildung 1)*.

Am Design der beiden Einstiegsseiten lassen sich sofort die Zielgruppen erkennen. Die Hauptseite von Oracle Stream Analytics ist bewusst im typischen Oracle-JET-Design einfach gehalten, das auch die Oracle Cloud Services prägt. Anhand von Templates, die typische Use Cases abbilden, kann der Fachanwender in die Modellierung einsteigen. Die darunterliegende Technik ist hier bewusst versteckt. Der Event Processing Visualizer hingegen bietet dem Entwickler oder Administrator Zugriff auf alle darunterliegenden Details.

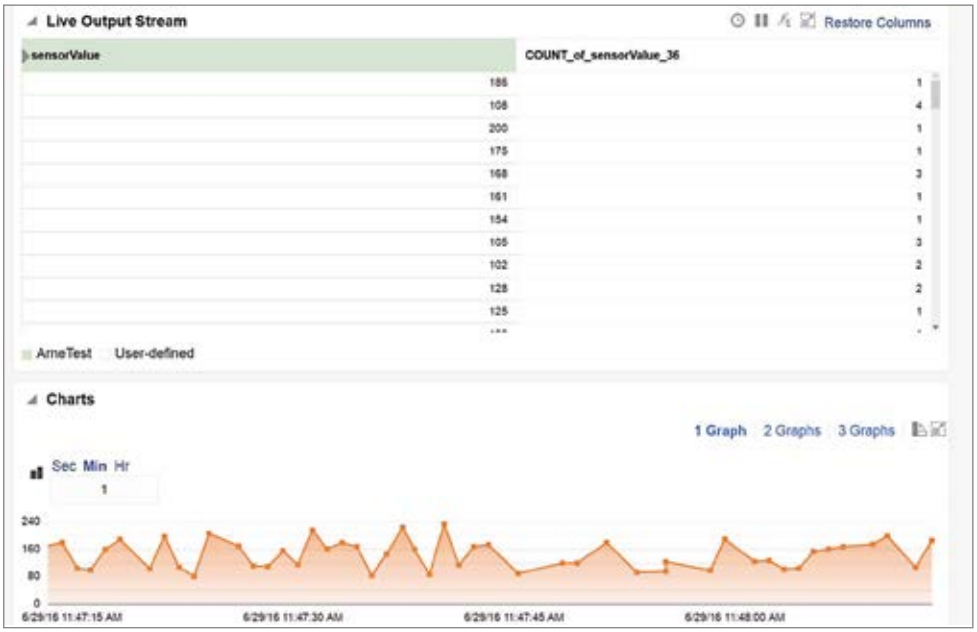

Abbildung 2: Ausgabe des Beispiel-Streams

Einstieg in Stream Analytics

Der Einstieg in Stream Analytics lässt sich leicht an einem Beispiel demonstrieren. Dazu wechselt man von der Willkommensseite auf den Reiter „Catalog". Sind die Beispiele mit installiert, finden sich hier bereits einige Beispielobjekte. Um einen Stream zu analysieren, wird zunächst natürlich ein solcher benötigt. Entwicklerfreundlich ist hier als ein möglicher Stream auch eine „.csv"-Datei vorgesehen. Beim Oracle Technology Network sind einige Beispieldateien zum Download angeboten, sodass man ohne große Vorbereitungen schnell erste Erfahrungen mit dem System machen kann.

In den Beispieldateien findet sich eine Datei „smalltrends.csv", die für dieses Beispiel genommen wurde. Im „Catalog" erstellt man per „Create New Item" einen neuen Stream und muss hier als Erstes den Typ auswählen. OSA bietet eine große Auswahl möglicher Streams. Neben Oracle-eigenen Technologien wie Coherence und offenen Standards wie REST findet sich hier auch Unterstützung für das im IoT-Umfeld wichtige Protokoll „MQTT" oder für das populäre Open Source Messaging System „Apache Kafka". Hier gibt es auch die Option, eine „.csv"-Datei als Stream-Quelle zu nehmen. Nach Auswahl der Datei wird man von OSA noch aufgefordert, einen Shape auszuwählen oder neu zu definieren. Dabei handelt es sich um eine Endbenutzer-Abstraktion des zu verwendenden Datentyps; hier wurde bewusst die Entwicklersprache vermieden.

Außerdem wird gefragt, ob gleich eine Exploration erstellt werden soll. Die Exploration ist die Abbildung des Streams mit allen darauf getätigten Operationen sowie den Live-Daten. Hat man ausgewählt, dass eine solche Exploration mit erzeugt werden soll, öffnet sich diese im Anschluss direkt im Exploration-Editor und der Anwender sieht sofort die Daten des Streams live durchlaufen. In diesem einfachen Beispiel sind es die Wertepaare für „sensorID" und „sensorValue". Hier kann der Anwender nun direkt anfangen, mit den Daten zu experimentieren, exploratives Vorgehen ist hier ausdrücklich vorgesehen.

Im Bereich „Sources" lassen sich auch weitere Streams mit dem vorhandenen verknüpfen („Correlation"). Der Bereich darunter enthält die der Summenbildung und Gruppierung. In diesem Beispiel erfolgt ein Summary auf „COUNT OF sensorValue" und ein „GROUP BY sensorValue". Ähnlichkeiten der darunterliegenden CQL-Abfragesprache zu SQL sind absolut gewollt.

Wie eingangs erwähnt, spielt bei der Stream-Analyse aber auch die zeitliche Komponente eine wichtige Rolle. Ohne weitere Änderungen würde das Aggregat nämlich immer „1" betragen. Grund dafür ist, dass sich innerhalb eines betrachteten Augenblicks immer nur ein Wert im Stream befindet. Um anzuzeigen, wie häufig ein bestimmter Wert auftrat, muss der Exploration also ein Zeitfenster mitgegeben werden. Dieses kann vorgeben werden, wenn man das Uhren-Symbol am rechten Rand anwählt. In diesem Beispiel sind 15 Minuten vorgegeben. Im dritten Bereich lassen sich Filter definieren („Transformation"); zu Demonstrationszwecken wurde ein einfaches „sensorValue greater than or equals 80" eingefügt. Im letzten Bereich lassen sich dann über Business Rules auch komplexere Muster suchen. Das Ergebnis wird dann live dargestellt, sowohl als durchlaufende Wertetabelle als auch als Graph *(siehe Abbildung 2)*.

Bislang sind die resultierenden Werte nur in der Design-Oberfläche dargestellt.

Ist der gewünschte Ergebnis-Stream gefunden, muss dieser zur weiteren Verwendung an ein verarbeitendes System weitergeleitet werden. Das kann, wie im obigen Beispiel, wieder ein neuer Datenstrom sein. Explorationen lassen sich auch beliebig verketten. So kann der resultierende Strom auch wieder ein Eingabestrom für eine weitere Exploration sein oder es wird nur nach bestimmten Mustern gesucht („Pattern Matching"). Typischerweise enthält der Ergebnis-Strom dann nur selten Daten, nämlich dann, wenn das gesuchte Muster gefunden wurde.

In jedem Fall lässt sich für die Exploration ein Target definieren, dies können unter anderem ein Service-Endpunkt oder ein Queuing-System sein. Dahinter können dann die Geschäftsanwendungen auf die gefundenen Daten reagieren („Business Logic"). Auch hier gibt es wieder die Option, den Ergebnis-Strom in eine „.csv"-Datei zu schreiben; die wurde in diesem Beispiel verwendet. Ist die Exploration dann soweit fertig, wird sie via „Publish" live geschaltet, bearbeitet die eingehenden Datenströme wie vorgegeben und meldet die Ergebnisse an die Geschäftsanwendungen.

Weitergehende Funktionalitäten

Dieses simple Beispiel zeigt nur einen kleinen Teil des Umfangs von OSA, die grundsätzliche Arbeitsweise von OSA sollte daran allerdings erkennbar sein. Einen Eindruck von der Mächtigkeit dieses Werkzeugs bekommt der Anwender beim Öffnen des dritten Reiters „Patterns". Diese Patterns bilden eine Vielzahl typischer Stream-Analysen ab. Beim Klick auf eines der Patterns bekommt der Anwender eine Kurzbeschreibung, einen Link auf die Dokumentation und ein YouTube-Video, das das Pattern kurz vorstellt.

Hier findet sich auch das Pattern „K-means Anomaly Detection", ein Machine-Learning-Algorithmus, der ein automatisches Clustering durchführt, also eine Gruppierung der Daten. Cluster mit einer geringen Menge von Werten werden als Anomalie angesehen. Hiermit lassen sich schnell und einfach Ausreißer im Datenstrom finden.

Machine-Learning-Verfahren werden voraussichtlich in Zukunft stark an Bedeutung gewinnen. So lange nur nach vordefinierten Mustern gesucht wird, werden auch nur diese gefunden. Das kann natürlich auch schon sehr hilfreich sein. Wenn beispielsweise das bekannte Muster für einen bevorstehenden Maschinenausfall erkannt und der Wartungstechniker rechtzeitig auf den Weg geschickt wurde, hat sich das System bewährt.

Interessant ist jedoch auch, nach Mustern zu suchen, die noch gar nicht bekannt sind. Wenn etwa im vorigen Beispiel der Wartungstechniker immer zu spät kommt, weil das Muster erst direkt vor dem Ausfall gefunden wird, dann müssen weitere Daten hinzugezogen werden. Beispielsweise von anderen Maschinen, die weiter vorne in der Verarbeitungskette stehen, oder Umgebungsdaten wie das aktuelle Wetter. Mit der Komplexität der Daten wächst aber auch der Aufwand für die Daten-Analysten, hierin nach geeigneten Mustern zu suchen. Machine-Learning-Algorithmen können hier eine große Hilfe sein.

Eine weitere Neuerung ist das sogenannte „Geofencing" inklusive Visualisierungs-Komponente. Die Datenströme können auch Tupel von Geo-Koordinaten enthalten. Oracle Event Processing kann überprüfen, ob sich die Koordinaten inner-/außerhalb von definierten Regionen befinden. Ein Ereignis kann also ausgelöst werden, wenn ein Objekt eine dieser Regionen verlässt. In der aktuellen Version bietet auch OSA das Geofencing an und liefert mit der Map gleich eine passende Darstellung dazu. Auf der Map lassen sich dann live die empfangenen Koordinaten und die definierten Regionen darstellen *(siehe Abbildung 3)*.

Die Engine unter OSA heißt „Oracle Event Processing" (OEP). Somit finden sich die Explorations auch im Event Processing Visualizer des OEP wieder. Auch können die Explorations aus OSA exportiert, in den JDeveloper importiert und hier als Event Processing Networks bearbeitet werden, etwa wenn externe Libraries zur Datenauswertung einbezogen werden sollen, was unter der OSA-Oberfläche nicht möglich ist.

Oracle Edge Analytics ist der Nachfolger von OEP Embedded. Auch hierauf können die gleichen Event Processing Networks ablaufen wie auf der Server-Variante, sodass über diesen Weg auch die Explorations auf einer Vielzahl von Devices zur Ausführung gebracht werden können, da als Systemvoraussetzung lediglich eine passende JVM benötigt wird.

In der aktuellen Version ist OEP aber nicht mehr die einzige Engine, die von OSA genutzt werden kann. Alternativ ist als Ablaufumgebung auch Apache Spark möglich. Beim aktuellen Stand der Entwicklung bietet OEP eine bessere „single node"-Performance sowie eine umfassendere Funktionalität. Spark hingegen kann die Last sehr gut in einem Cluster verteilen. Beide Engines haben unterschiedliche Stärken – gut für den Anwender, dass er hier die Wahl hat.

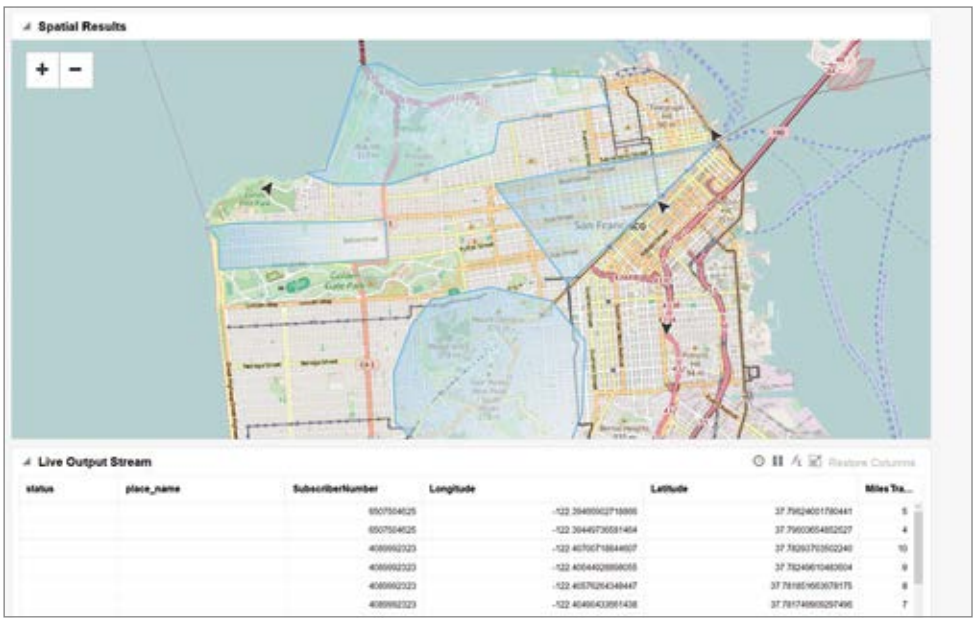

Abbildung 3: Geofencing-Visualisierung

Coherence kann sowohl als Quelle als auch als Ziel von OSA dienen. Am Beispiel von Predictive Maintenance könnten etwa alle Geräte ihre Messdaten in einem Coherence-Cluster ablegen. Aus diesem führt OSA dann die Stream-Analyse durch und eine mobile Wartungsanwendung liest beispielsweise benötigte Live-Daten, während Coherence mittels „Write-Through“ oder „Write-Behind“ die Daten in die Zielsysteme schreibt.

Selbstverständlich hat Oracle auch eine Integration mit Datenbanken im Angebot. Mit „Oracle GoldenGate for Big Data“ können Änderungen aus einer Datenbank über eine Kafka-Queue direkt in OSA weiterverarbeitet werden. Mit dem IoT Cloud Service (IoTCS) bietet Oracle einen PaaS Service an, der an anderer Stelle in dieser Ausgabe beschrieben wird. OSA ist ein wichtiger Bestandteil des IoTCS, umgekehrt ist der IoTCS eine Möglichkeit, OSA als Cloud-Service zu nutzen.

Fazit

Oracle Stream Analytics ist innerhalb von Minuten installiert und nach kurzer Zeit lassen sich Stream-Analysen durchführen, dank der neuen Oberfläche jetzt auch von Fachanwendern. Trotz der einfachen Endbenutzer-Oberfläche liegt unter Stream Analytics die volle Mächtigkeit von Oracle Event Processing, sodass ein Entwickler gegebenenfalls auch komplexere Event-Processing-Networks erstellen kann. Diese können dann auch auf einer Vielzahl von Devices laufen. Durch die SQL-ähnliche Abfragesprache CQL werden sich Datenbank-Entwickler hier schnell heimisch fühlen. Mit dem aktuellen Release wird die Funktionalität mit Machine-Learning und Geofencing wesentlich erweitert, durch die Öffnung in Richtung populärer Open-Source-Werkzeuge ist OSA jetzt auch einfacher in heterogene Umgebungen zu integrieren.

Quelle

[1] The Forrester Wave: Big Data Streaming Analytics, Q1 2016

Arne Brüning
arne.bruening@oracle.com