Nur gegen Cache

SAP HANA – In-Memory optimierte Anwendungsszenarien programmieren

Stefan Kühnlein, Holger Seubert

Welche Möglichkeiten zur Umsetzung von Anwendungen bietet SAP HANA und wie können Softwarearchitekten und Entwickler diese nutzen? Neben einer kurzen Vorstellung möglicher Anwendungsszenarien von SAP HANA werden unterschiedliche Möglichkeiten zur Programmierung einer SAP HANA-Anwendung aufgezeigt.

Hauptspeicherresidente Speicherung polyglotter Daten

Betrachtet man Anforderungen der Unternehmen genauer, stellt man fest, dass sich die heutigen Herausforderungen verändert haben. Unternehmen sind darauf angewiesen, Informationen über verschiedenste Kanäle zu sammeln und zeitnah auszuwerten. Bisherige Verfahren und Architekturen können bei der Speicherung und zeitnahen Analyse großer, unterschiedlicher Datenmengen an die Grenzen von Machbarkeit und Wirtschaftlichkeit stoßen.

Insbesondere die Notwendigkeit zur Voraggregation der Daten führt bei großen Datenmengen zu einer hohen Redundanz und erschwert die zeitnahe Erstellung von Analyseergebnissen. Dazu kommen die unterschiedlichen Datenformate, die zu verarbeiten sind und bestehende Architekturen vor neue Herausforderungen stellen. Um diesen Anforderungen entgegenzukommen, wurde mit Polyglot Persistence ein Architektur-Pattern entwickelt, das die Speicherung der Daten je nach ihrer Herkunft in ein passendes Datenbanksystem übernimmt. Dieses Konzept birgt allerdings neue Risiken, vor allem im Betrieb, da mit Polyglot Persistence nicht nur eine, sondern mehrere Datenbanken betrieben werden.

Allein die Datenmengen, die durch soziale Medien entstehen und für die Informationsgewinnung zunehmend wichtiger werden, sind riesig. Wurden früher via Twitter nur Freunde informiert, wird der Kurznachrichtendienst heute zum Beispiel auch für Marketing oder Recruiting eingesetzt.

Mit SAP HANA hat SAP eine Datenbank entwickelt, die neuste Hardware-Technologien mit einem In-Memory-Datenmanagement verbindet und dadurch neue Chancen für die Entwicklung von Anwendungen ermöglicht. So kann mit Verwendung von SAP HANA auf die Voraggregation von Daten zwecks kennzahlenbasierter Auswertung verzichtet werden, und die Verarbeitung von strukturierten, unstrukturierten und graphorientierten Daten in derselben Datenbank erfolgen. Dadurch kann die Anzahl der Datenbanken sowie ETL-Strecken reduziert und die komplexe Systemlandschaft vereinfacht werden.

Mit SAP HANA stellt SAP nicht nur eine Datenbank auf einer dedizierten Hardware zur Verfügung, sondern auch eine Plattform für die Anwendungsentwicklung von Echtzeitanwendungen. So liefert SAP HANA funktionale Komponenten wie GeoSpatial, Text Mining, Bibliotheken für Predictive Analysis und Data Mining, deren Algorithmen für die In-Memory-Verarbeitung optimiert und sehr eng an die Datenbank gekoppelt sind.

Programmieransätze

Für die Entwicklung von Anwendungen auf der SAP HANA-Plattform stehen dem Entwickler zwei Architekturansätze zur Verfügung, mit denen er die Daten und Funktionen der Plattform nutzen kann.

Die Entwicklung einer *nativen Anwendung* erfolgt mit den SAP HANA-Entwicklungswerkzeugen (Eclipse- oder Web-basiert). Native Anwendungen werden direkt auf der SAP HANA-Plattform ausgeführt und benötigen keinen zusätzlichen Anwendungsserver. Die Entwicklung der datenintensiven Geschäfts- und Kontrollflusslogik kann entweder mit SQLScript oder mit JavaScript im Kontext der Plattformkomponenten SAP HANA Extended Application Services (SAP HANA XS) erfolgen.

Nicht native Anwendungen werden außerhalb der SAP HANA-Entwicklungsplattform (z. B. in Java, C++, ABAP) entwickelt, getestet und bereitgestellt und greifen über einen entsprechenden Treiber (z. B. JDBC, ODBC, ADBC) auf die Daten und SQL-Funktionen der SAP HANA-Plattform zu.

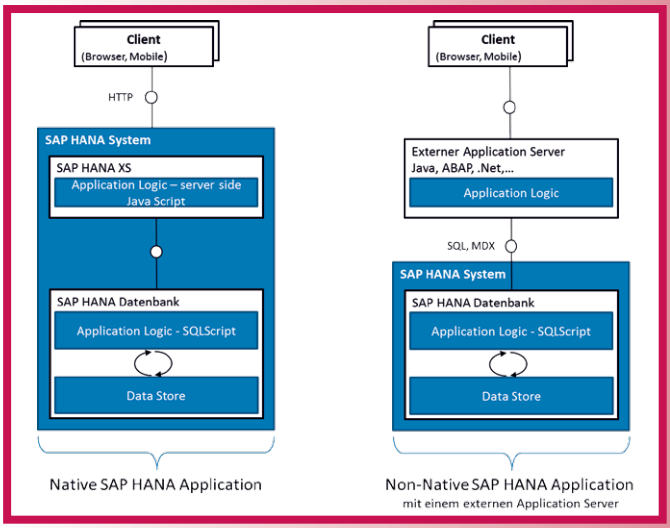

Abb. 1: Gegenüberstellung nativer und nicht nativer SAP HANA-Anwendungen (Quelle: [SAPHANA])

Abbildung 1 veranschaulicht die grundlegende Architektur für native und nicht native SAP HANA-Anwendungen. So zeigt die linke Abbildung die zweischichtige Architektur einer nativen Anwendung, bei der mittels eines Browsers über das HTTP-Protokoll auf die Anwendung zugegriffen wird. Die rechte Seite zeigt die dreischichtige Architektur einer nicht nativen SAP HANA-Anwendung, bei der der Anwender über einen Browser oder Fat Client auf die Applikationslogik mit dem jeweiligen Protokoll zugreift.

Abbildung 2 stellt die zur Verfügung stehenden Programmiersprachen und zugehörigen Umgebungen der SAP HANA-Plattform dar. Je nach Anwendungsfall beziehungsweise Architekturausrichtung wird eine der entsprechenden Technologien ausgewählt, um die Leistungsfähigkeit von SAP HANA optimal für die Anwendungslogik zu nutzen. So kann zum Beispiel die Entwicklung der Logik einer nativen SAP HANA-Anwendung mit Hilfe der Core Data Services (CDS), HDBtable und SQLScript erfolgen. Mit serverseitigem JavaScript (XSJS) wird ebenfalls Anwendungslogik implementiert, die direkt auf der SAP HANA-Plattform ausgeführt wird. Der Zugriff auf die Persistenzschicht erfolgt entweder direkt mittels des JavaScript-Database-API oder unter Verwendung des OData-

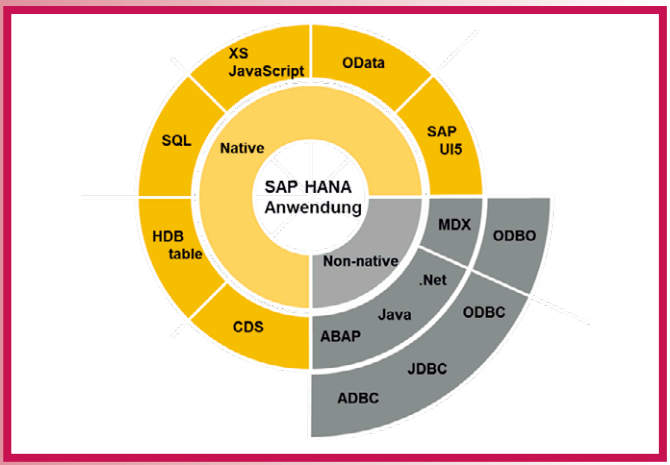

Abb. 2: Übersicht der Techniken für die Entwicklung von Anwendungen auf Basis von SAP HANA (Quelle: [SAPHANA])

Standards. Für die Erstellung der Benutzungsoberflächen, die an die serverseitigen JavaScript- oder OData-Services gebunden werden, kann das integrierte SAPUI5 HTML5 SDK oder auch jede andere Open-Source-Bibliothek verwendet werden.

Für die Anbindung von nicht nativen SAP HANA-Anwendungen stehen – in Abhängigkeit der verwendeten Programmiersprache – die entsprechenden Treiber zur Verfügung. So erfolgt der Zugriff auf die SAP HANA-Datenbank für Java-Anwendungen via JDBC, für C/C++/.NET-Anwendungen via ODBC und ADO.NET oder für ABAP-Anwendungen via ADBC.

Szenario: Wetteranalyse via Twitter

Im folgenden Beispiel wird die Speicherung und Auswertung mit Hilfe der in SAP HANA integrierten Textanalyse von Wetterdaten aus dem sozialen Netzwerk Twitter gezeigt. Hierzu wird exemplarisch nach Tweets mit Tags wie `#wetter` oder `#wetterstation` gesucht. Mit Hilfe der integrierten Textanalyse von SAP HANA, sowie den geografischen Daten der Tweets, kann eine „soziale" Wetterkarte erstellt werden.

Für die Speicherung der Tweet-Nachrichten wird eine Tabelle „TWEETS" (s. Abb. 3) in der In-Memory-Datenbank angelegt.

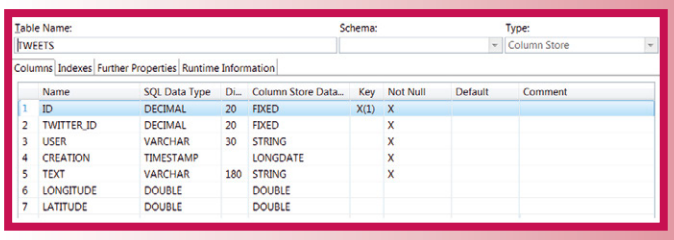
Table Name: TWEETS | Schema: | Type: Column Store

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Di...	Column Store Data...	Key	Not Null	Default	Comment
1	ID	DECIMAL	20	FIXED	X(1)	X		
2	TWITTER_ID	DECIMAL	20	FIXED		X		
3	USER	VARCHAR	30	STRING		X		
4	CREATION	TIMESTAMP		LONGDATE		X		
5	TEXT	VARCHAR	180	STRING		X		
6	LONGITUDE	DOUBLE		DOUBLE				
7	LATITUDE	DOUBLE		DOUBLE				

Abb. 3: Definition der Tabelle „TWEETS" in Eclipse

Native Anwendungsentwicklung

Native SAP HANA-Anwendungen nutzen Technologie und Services, die durch die integrierten SAP HANA Extended Application Services (SAP HANA XS) bereitgestellt werden. Dabei sind folgende Rahmenbedingungen zu berücksichtigen:

- Alle Artefakte der Anwendung werden im internen Repository der SAP HANA-Plattform gespeichert und deployed.
- Serverseitige Anwendungslogik wird entweder mit SQLScript oder mit Hilfe von serverseitigem (SAP HANA XS) JavaScript erstellt.
- Die Darstellung der Benutzeroberfläche erfolgt ausschließlich im Client (Browser oder Mobile), typischerweise mit HTML5 (SAPUI5).

Mit den SAP HANA Extended Application Services (SAP HANA XS) wird des Weiteren ein Satz von serverseitigen JavaScript Application Programming Interfaces (API) zur Verfügung gestellt, mit denen eigene Anwendungen erweitert werden.

So erfolgt mit dem bereitgestellten Datenbank-API der Zugriff mittels SQL auf Tabellen, Views, Analytical- und Caclulation-Views sowie auf Stored Procedures. Ebenfalls besteht die Möglichkeit, mittels HTTP-Requests Daten von einem Service abzurufen oder immer wiederkehrende Aufgaben automatisiert zu starten. Tabelle 1 stellt die zur Verfügung stehenden APIs vor.

Für das Lesen von Tweets, die mit den HashTag `#wetter` oder `#wetterstation` gekennzeichnet sind, wird im folgenden Beispiel sowohl das Outbound Connectivity API als auch das Job Schedule API verwendet. Hierzu bietet Twitter entsprechende REST-Services, um nach bestimmten Tweets zu suchen.

Die Abfrage der neusten Tweets mit den HashTags `#wetter` oder `#wetterstation` erfolgt in einem serverseitigen JavaScript.

API	Beschreibung
Database	Die Programmierschnittstelle erlaubt dem Entwickler, eine Verbindung zur Datenbank aufzubauen. So können z. B. Transaktionen gestartet, geschlossen oder abgebrochen werden sowie SQL-Statements und SQL-Prozeduren ausgeführt werden.
Outbound Connectivity	Die Programmierschnittstelle ermöglicht Zugriffe auf HTTP-Ziele, die Services zur Verfügung stellen, die durch die SAP HANA XS-Anwendung genutzt werden können.
Request Processing	Mit dieser Programmierschnittstelle kann sowohl lesend als auch schreibend auf den Kontext des aktuellen HTTP-Requests zugegriffen werden. So kann der Entwickler unter Verwendung dieses API den Inhalt des Request und des Response verändern.
Job Schedule	Die zeitliche Steuerung von immer wiederkehrenden Aufgaben, die im Hintergrund ausgeführt werden sollen, kann mit Hilfe dieser Programmierschnittstelle realisiert werden.
Trace	Mit dieser Programmierschnittstelle kann auf die verschiedensten Trace-Levels zugegriffen und entsprechende Traces durch die Anwendung generiert werden. Die erzeugten Trace-Dateien können im SAP HANA Studio in der Perspektive *Administration* unter der Sektion *Diagnosis Files* eingesehen werden.

Tabelle 1: Mit den dargestellten APIs können verschiedene Funktionen innerhalb einer nativen SAP HANA-Anwendung realisiert werden

Damit dieses Script in regelmäßigen Abständen durch die Job Schedule aufgerufen wird, wird die Abfrage der aktuellsten Tweets in einer JS-Funktion eingebettet (s. Listing 1).

```
function collectWetterTweets() {
  var conn = $.db.getConnection("opitz.wetteranalyse::wetteranalyse");
  try {
    var index = 0;

    var tweets = JSON.parse(getTweets(getLastId(conn)).asString());
    if (tweets !== null && tweets.statuses !== null
        && tweets.statuses.length > 0) {
      var count = tweets.statuses.length;
      for (index = 0; index < count; index++) {
        saveTweet(conn, tweets.statuses[index]);
      }
    }
    conn.commit();

  } catch (ex) {

    $.trace.error(ex.message);
  } finally {
    if (!conn.isClosed()) {
      conn.close();
    }
  }
}
```

Listing 1: Abfrage der aktuellsten Tweets in einer JS-Funktion

Für die Abfrage der Tweets wird das von Twitter bereitgestellte REST-API GET search/tweets verwendet. Zur Nutzung der Schnittstelle ist eine Identifikation über OAuth erforderlich. Für die Authentifizierung zur Benutzung des REST-API zur Suche von Tweets ist die application-only Authentication ausreichend, da für diesen Kontext keine benutzerspezifischen Daten benötigt werden. Mit Listing 2 kann das sogenannte Inhaber-Token von Twitter innerhalb der SAP HANA XS-Engine angefordert werden.

```
function getBearerToken() {
  var dest = $.net.http.readDestination("opitz.wetteranalyse", "wetter");
  var client = new $.net.http.Client();
  var req = new $.web.WebRequest($.net.http.POST,
      "/oauth2/token?grant_type=client_credentials");
  req.headers.set("Authorization", getBased64EncodeBearerToken());
  req.headers.set("Content-Type:",
      "application/x-www-form-urlencoded;charset=UTF-8");
  client.request(req, dest);
  var access = JSON.parse(client.getResponse().body.asString());
  return access.access_token;
}
```

Listing 2: Anfrage des OAuth 2 Bearer Tokens

Nach Erhalt des Inhaber-Tokens werden die neusten Tweets zu den Hashtags **`#wetter`** oder **`#wetterstation`** wie in Listing 3 abgerufen.

```
function getTweets(maxId) {
var searchQuery = "/1.1/search/tweets.json?q=%23wetter%20OR%20
%23wetterstation&max=1000";
  var dest = $.net.http.readDestination("opitz.wetteranalyse", "wetter");
  var client = new $.net.http.Client();
if (maxId !== null && maxId !== '') {
    searchQuery = searchQuery + "&since_id=" + maxId;
  }
  var req = new $.web.WebRequest($.net.http.GET,
      searchQuery);
  req.headers.set("Authorization", "Bearer " + getBearerToken());
  client.request(req, dest);
  return client.getResponse().body;
}
```

Listing 3: Abruf der neuesten Tweets zu unseren Wetter-Hashtags

Zuletzt müssen die Tweets in der Datenbank gespeichert und ausgewertet werden. Dies kann entweder mittels der Database-Bibliothek der SAP HANA XS-Engine oder unter Verwendung des OData-Service erfolgen.

Damit die Daten kontinuierlich über das REST-API ermittelt und in der Datenbank persistiert werden können, wird ein entsprechender Scheduler aufgesetzt. Hierzu ist in der SAP HANA XS-Engine (s. Tabelle 1) ein API implementiert, das es erlaubt, entsprechende Aufgaben zu bestimmten Zeiten beziehungsweise in bestimmten Zeitabständen durchzuführen. Ein solcher Job wird über eine spezielle Datei, der *.xsjob-Datei, aufgesetzt. Zum Lesen und Speichern der Wetter-Tweets in einem Intervall von 5 Minuten wird die Job-Definition aus Listing 4 aufgesetzt.

```
{
  "description": "Wetteranalyse mittels Twitter",
  "action": "opitz.wetteranalyse:wetteranalyse.xsjs::collectWetterTweets",
  "schedules":
  [
    {
    "description": "Lesen der Tweets mit den Hashtags #wetter und
#wetterstation",
    "xscron": "* * * * * */5 *",
    "parameter": { }
    }
  ]
}
```

Listing 4: Job-Definition zum Lesen und Speichern der Wetter-Tweets

Nicht native Anwendungsentwicklung

Nicht native Anwendungen, die auf die SAP HANA-Plattform zugreifen, können Services nutzen, die durch die integrierte SAP HANA XS-Technologie bereitgestellt werden. Die nicht native Anwendung wird im Vergleich zu der nativen Anwendung auf einem externen Applikationsserver ausgeführt.

Im Java-Umfeld kann dies sowohl direkt mittels JDBC-API oder indirekt über JPA (Java Persistence API) erfolgen. Sowohl die aktuelle Version von Hibernate 4.3.5 als auch EclipseLink 2.5.2 unterstützen den Zugriff auf die SAP HANA-Datenbank via JPA und JDBC. Die Rahmenbedingungen bei der Verwendung von EclipseLink und SAP HANA sind unter [EclipseLinkHana] zu finden.

Im Folgenden wird gezeigt, wie das bereits vorgestellte Szenario zur Speicherung von Twitter-Nachrichten mit einer Java-Anwendung und JPA in der Tabelle TWEETS persistiert werden kann. Im Gegensatz zur nativen SAP HANA-Anwendung werden die Daten direkt vom Streaming-API abgegriffen und nicht über einen REST-Service gelesen. Durch die Verwendung des Streaming-API wird auf den globalen Datenstrom der Tweets mit einer geringen Latenz zugegriffen. Ein entsprechender Streaming-Client veröffentlicht Nachrichten über Tweets und weitere Ereignisse, ohne dass ständig über die REST-Schnittstelle eine Abfrage gestartet werden muss.

Zur Speicherung der Tweets wird ein entsprechendes OR-Mapping der Tabellen für EclipseLink benötigt. Im Rahmen des Mappings der Tabellen auf die entsprechenden POJOs sind keine Besonderheiten zu beachten. Lediglich der SAP HANA JDBC-Treiber (ngdbc.jar) muss sich zusätzlich zu den obligatorischen Bibliotheken im Classpath der Anwendung befinden.

Das Streaming der Tweets zu den oben genannten HashTags erfolgt mit der twitter4j-Bibliothek. Diese Library stellt mit der Klasse **`TwitterStream`** eine entsprechende Implementierung bereit, mit der die Tweets gelesen werden können.

Die Klasse **TwitterStream** wird mit Hilfe der Factory **TwitterStreamFactory** erzeugt. Um den Stream auf bestimmte Hash-Tags zu begrenzen, wird eine **FilterQuery** entsprechend initialisiert und der Methode **filter()** des TwitterStreams übergeben (s. Listing 5).

```
initTwitterFactory();
TwitterStream stream = twitterfactory.getInstance();

stream.addListener(new TwitterStatusListener(em));
FilterQuery query = new FilterQuery();
query.track(new String[]{"#wetter","#wetterstation"});
stream.filter(buildFilterQuery());
```

Listing 5: Definition des Filters für das Twitter-Streaming-API

Zum Auswerten der Tweets wird eine Implementierung der Schnittstelle **TwitterStatusListener** benötigt. Die Zerlegung der Tweets in ihre Bestandteile sowie der Speicherung in der SAP HANA-Datenbank erfolgt in der zu implementierenden Methode **onStatus(Status status)** (s. Listing 6).

```
public class TwitterStatusListener implements StatusListener {
  private EntityManager entityManager = null;
  public TwitterStatusListener(EntityManager pEntityManager) {
    entityManager = pEntityManager;
  }

  @Override
  public void onStatus(Status status) {
    Tweet tweet = new Tweet();

tweet.setTwitter_id(status.getId());
    tweet.setUser(status.getUser().getName());
    tweet.setText(status.getText());
    tweet.setCreateTime(status.getCreatedAt());

    if (status.getGeoLocation() != null) {
      tweet.setLatidute(Double.valueOf(status.getGeoLocation()
          .getLatitude()));
      tweet.setLongidute(Double.valueOf(status.getGeoLocation()
          .getLongitude()));
    }
    tweet.setLanguage(status.getLang());
entityManager.getTransaction().begin();
    entityManager.persist(tweet);
    entityManager.getTransaction().commit();
  }
```

Listing 6: Speicherung der Tweets in der Datenbank

Auswertung der Tweets

Um nach bestimmten Informationen in den gespeicherten Tweets zu suchen, wird ein Volltextindex in SAP HANA mit folgendem SQL-Ausdruck angelegt

```
CREATE FULLTEXT INDEX "TWEETS_ANALYSIS"
  ON "TWEETS"("TEXT") FUZZY SEARCH INDEX ON
CONFIGURATION 'EXTRACTION_CORE_VOICEOFCUSTOMER' TEXT ANALYSIS ON
```

Neben der Volltextsuche mittels SQL/MM-Standard erlaubt der angelegte Index auch, Resultate der Textanalyse bei der Abfrage zu berücksichtigen. Über unterschiedliche Konfigurationen werden die von SAP HANA durchgeführten linguistischen Analysen definiert. Die Analyseergebnisse stehen in einer automatisch erstellten Tabelle mit dem Präfix **$TA_** im eigenen Schema. Die Textanalysen basieren auf bereitgestellten Wörterbüchern, die ebenfalls inhaltlich erweitert werden können.

Für das beschriebene Beispiel wird eine Tabelle **$TA_TWEET_ANALYSIS** erzeugt, in der die Ergebnisse der Textanalyse abgelegt werden. Die Spalte **TA_TOKEN** der Tabelle speichert den entsprechend erkannten String, wie „22.0°C“ oder „Kiel“, und die Spalte **TA_TYPE** den erkannten Entity-Typ, wie „MEASURE“ oder „LOCALITY“.

Folgender SELECT-Ausdruck sucht Tweets, die in ihrem Text eine Kennzahl (wie Temperatur) und gleichzeitig den String „Gewitter“ enthalten

```
SELECT "TWEETS"."ID", "TWEETS"."TEXT",
  "$TA_TWEETS_ANALYSIS"."TA_TOKEN", snippets("TEXT")
FROM "$TA_TWEETS_ANALYSIS","TWEETS"
WHERE "$TA_TWEETS_ANALYSIS"."ID" = "TWEETS"."ID"
  AND "TA_TYPE" = 'MEASURE' AND CONTAINS(TEXT,'Gewitter');
```

Bei der Textanalyse werden ebenfalls Stimmungen erkannt, um so nach Tweets zu suchen, die sich positiv über das Wetter äußern

```
SELECT "TWEETS"."ID", "TWEETS"."TEXT",
  "$TA_TWEETS_ANALYSIS"."TA_TOKEN"
FROM "$TA_TWEETS_ANALYSIS","TWEETS"
WHERE "$TA_TWEETS_ANALYSIS"."ID" = "TWEETS"."ID"
  AND "TA_TYPE" = 'StrongPositiveSentiment';
```

Fazit

Für SAP HANA lassen sich im Wesentlichen zwei unterschiedliche Architekturansätze zur Entwicklung eigener Anwendungen unterscheiden: Native SAP HANA-Anwendungen, die direkt auf der SAP HANA-Plattform ausgeführt werden und keinen Applikationsserver benötigen. Bei der Umsetzung der Geschäftslogik kommt bei nativen Anwendungen JavaScript und datenbanknahes SQLScript zum Einsatz.

Im Weiteren stehen Standard-Schnittstellen wie JDBC, ODBC und ADO.NET zur Verfügung, um komplexere Geschäftslogik mit bewährten und bereits eingesetzten Methoden und Technologien zu entwickeln. Bei der Verwendung von ORM-Technologien ist, wie bei anderen Datenbanksystemen auch, auf die Spezifika von SAP HANA zu achten.

Wichtig zu betonen ist die Kombinierbarkeit beider Architekturansätze, um die Vorteile der nativen Anwendung mit zum Beispiel J2EE-Anwendungen zu verbinden.

Literatur und Links

[EclipseLinkHana] EclipseLink Development SAP HANA-Plattform, **http://wiki.eclipse.org/EclipseLink/Development/DatabasePlatform/HANAPlatform#Limitations_of_the_Platform**

[SAPHANA] SAP HANA Developer Guide, **http://help.sap.de/hana/SAP_HANA_Developer_Guide_en.pdf**

Stefan Kühnlein ist Solution Architekt bei OPITZ CONSULTING in München. Er unterstützt Kunden bei der Planung und Durchführung von sowohl klassischen Java- als auch von BPM- und serviceorientierten Projekten. Seit Mai 2014 leitet er das neu gegründete Kompetenzzentrum „SAP HANA Development“.
E-Mail: Stefan.Kuehnlein@opitz-consulting.com

Holger Seubert arbeitet als Senior Solution Architect im Bereich Datenbanken und Technologie bei SAP und ist auf die Implementierung und Einführung SAP HANA-basierter Lösungen spezialisiert.
E-Mail: h.seubert@sap.com