www.ordix.de € 3,20

1|2019

ORDIX® news

einfach. besser. informiert.

Zwischen den Welten: SQL oder NoSQL

6 | MySQL Document Store

11 | Analytische Funktionen in Oracle – Anwendungsgebiete und Vorteile

15 | Graylog – Zuverlässiges und kostenneutrales Logging & Monitoring von verteilten Systemen

18 | AI Meets Fintech – Aufbau einer Data Science Pipeline

44 | Performance-Faktoren von IBM DataStage – Ein Blick in die Architektur & Performance

BIG DATA ODER SMART DATA

WANN WERDEN IHRE DATENMENGEN SMART?

Das Schlagwort Big Data begegnet uns nach wie vor in den unterschiedlichsten Domänen. Spannend wird es aber dann, wenn Big Data zu Smart Data werden – wenn wir also Nutzen, Qualität und Sicherheit der Datenmengen erkennen und sicherstellen können.

Mit unserer Erfahrung und unseren Technologielösungen helfen wir Ihnen, das Potenzial Ihrer Daten voll auszuschöpfen. Oder möchten Sie selbst zum Experten auf diesem Gebiet werden? Dann bilden wir Sie mit unseren Seminaren zum Big Data Professional aus.

Wenn Sie neugierig geworden sind, informieren Sie sich unter:
www.ordix.de/bigdata

Groß und klein

Paderborn, Mai 2019

Gute Nachricht: Die Arbeitslosenzahl ist so niedrig wie noch nie seit der Wiedervereinigung, im April gab es mehr als 45 Millionen Beschäftigte in Deutschland.

Schlechte Nachricht(en): Bayer, ThyssenKrupp, VW, Daimler, Siemens, Deutsche Bank, Commerzbank, RWE, EON usw. drohen jeder mit der Entlassung von mehreren Tausend Beschäftigten. Da kommen schnell hohe sechsstellige Zahlen zusammen.

Wie passen die beiden Nachrichten zusammen? Relativ einfach, ein gesunder Mittelstand und viele sorgsam operierende Unternehmer in Deutschland fangen die Misswirtschaft von angestellten Managern mit überdimensionierten Gehältern auf. Ganz nebenbei vernichten diese Manager viele Milliarden Euro von gutgläubigen Anlegern (allein jüngst bei Bayer, VW oder auch Daimler).

Was fällt der Politik dazu ein? Nun, unser Wirtschafts- und Finanzminister hätten gerne noch größere Konzerne (Bankenzusammenschlüsse, ThyssenKrupp + Tata, Siemens und Alstom, vielleicht BMW/Daimler usw.), weil das wirklich viel bringt, natürlich auch Steuereinnahmen, denn die großen Konzerne versteuern ja überall richtig brav ihre Gewinne. Im Gegenzug werden viele kleine und mittlere Unternehmen durch immer mehr Vorgaben, Verpflichtungen und unsinnige Bürokratie, aber auch durch die Konzerne gegängelt.

Sehr vieles bei uns ist rückwärtsgewandt – das beginnt bei der Diskussion über E-Autos. Der Diesel wird doch nicht durch E-Autos von heute auf morgen abgeschafft, aber wir brauchen dringend eine weitere Alternative neben Diesel und Benziner. Und zwar jetzt, nicht erst in fünf, zehn oder gar zwanzig Jahren. Und diese muss CO_2-neutral sein, zumindest zum Zeitpunkt der Fahrt. Dass die großen deutschen Automobilkonzerne in ihrer Raffgier den Zug der Zeit verschlafen haben (s.o.), ist eigentlich deren Problem (leider nicht nur).

Das Vorgehen, nichts Neues oder Radikales zu wagen, geht weiter beim Thema Braunkohle und dem Ausstieg aus dem Abbau und Alternativen bei der Energiegewinnung. Aber, ach, die Arbeitsplätze müssen erhalten bleiben. Warum wird dann vor allem den Energiekonzernen das Geld in den Rachen geschmissen? Stoppt den Abbau so schnell wie möglich und gebt das Geld denen, die dann keine Arbeit mehr haben, aber nicht den Konzernen!

Hat in Berlin einmal einer darüber nachgedacht, wie viele Arbeitsplätze im Bereich Solarenergie in den letzten Jahren durch die Politik kaputt gemacht wurden? Diese Arbeitsplätze wären zukunftsorientiert gewesen, jetzt jammern alle darüber, dass Solartechnik fast nur noch aus China kommt.

Warum werden z.B. keine Smartphones in Deutschland produziert oder hergestellt? Was Apple, Samsung oder Huawei können, sollte doch hier auch klappen. Aber wir Deutschen überlegen immer zuerst, wenn was Neues kommt, warum es nicht kommen kann, siehe Windenergie, Solartechnik oder eben auch E-Autos.

Ich kann mich des Gefühls nicht erwehren, dass sich die meisten (insbesondere Ältere) am liebsten in ihr Schneckenhaus zurückziehen und hoffen, dass alles so bleibt, wie es vorgestern war. Und wenn die Jugend endlich mal aktiv wird und friedlich Protest anmeldet (Friday's for Future), werfen ihnen unsere Politiker vor, dass sie die Schule schwänzen und das reale Leben „erfahrenen" Menschen überlassen sollen[1]. Danke Herr Lindner, solche Klugscheißer habe ich Anfang der 70er auch ungemein geschätzt, ach, da waren Sie ja noch nicht mal geboren!

Dann widme ich mal die nächsten Sätze den Profis, die die Artikel in dieser ORDIX® news geschrieben haben. Vieles hat dieses Mal mit Software-Entwicklung, anderes mit Betriebsaspekten zu tun. Insbesondere um das Stichwort JSON drehen sich gleich zwei Artikel (MySQL Document Store und JSON in Oracle).

Bei Oracle beschäftigen wir uns mit neuen (Polymorphic Table Functions), aber auch mit alten (Analytische Funktionen) Features. Und ein Artikel zu Standardverfahren in APEX und das Dateihandling sind ebenfalls im Angebot.

Neben Oracle, MySQL (und NoSQL) sind wir noch mit folgenden Datenbanken vertreten: Cassandra und PostgreSQL. Cassandra betrachten wir unter dem betrieblichen Aspekt und bei PostgreSQL beschreiben wir im ersten Artikel einer neuen Reihe einige Enterprise Features.

Last not Least finden Sie Interessantes zum Thema agile Software-Entwicklung (Module statt Microservices), zur Performance bei DataStage und sogar künstliche Intelligenz findet mit dem Artikel „AI meets Fintech" einen Eingang in die ORDIX® news – obwohl ich immer noch ein Befürworter der menschlichen Intelligenz bin, selbst wenn ich nicht der Erfinder des gesunden Menschenverstandes bin[2].

Wenn Sie das lesen, waren Sie und ich hoffentlich zur Europawahl gegangen und haben Ihr Kreuz für ein wirklich starkes, gemeinsames Europa gemacht, womit Rechtspopulisten ausscheiden.

Viele Grüße

Wolfgang Kögler

[1] Von Kindern könne man nicht erwarten, dass sie alle globalen Zusammenhänge sehen. Das sei vielmehr „eine Sache für Profis". (Quelle: https://www.derwesten.de/politik/christian-lindner-ueber-guido-reil-und-die-afd-das-ist-ein-schlechter-charakterzug-id218398521.html)

[2] Den Titel hat sich ja schon unser geehrter Verkehrsminister Scheuer selbst verliehen. Nur erstaunlich, dass viele studierte und wissenschaftlich arbeitende Menschen anderer Meinung sind, sofern sie nicht von VW, BMW oder Daimler bezahlt werden.

Zwischen den Welten – SQL oder NoSQL

AI Meets Fintech

Big Data / Datawarehouse

6 ..... MySQL Document Store
Zwischend den Welten – SQL oder NoSQL
Warum sollte man sich zwischen SQL und NoSQL entscheiden müssen? Dieser Artikel gibt Ihnen einen Einblick in den JSON Document Store, der beide Welten verbindet und in dem diese problemlos kombiniert werden können.

9 ..... Einrichtung und Betriebsaspekte
Apache Cassandra
Welche Aspekte sind bei der Einrichtung und der Erstkonfiguration eines Cassandra-Clusters zu beachten? Welche Berechtigungen bekommen die Nutzer und wie sieht es eigentlich mit einem Backup aus?

18 ..... AI Meets Fintech
Aufbau einer Data Science Pipeline
Olaf Hein demonstriert in diesem Artikel einen erprobten Ansatz für den Aufbau einer Data Science Pipeline. Dabei werden die unterschiedlichen Anforderungen aus dem Betrieb, die Anwendungsentwicklung und Data Science berücksichtigt.

44 ..... Performance-Faktoren von IBM DataStage
Ein Blick in die Architektur und Performance von IBM DataStage
Schlagwörter wie „Cluster Computing“ und „horizontale Skalierbarkeit“ scheinen in den letzten Jahren vor allem durch Big-Data-Technologien einen neuen Trend bekommen zu haben. Schon davor hatte IBM DataStage einige derartige parallele Konzepte vereinigt und für die breite ETL-Kundschaft im Angebot.

Oracle

11 ..... Analytische Funktionen in Oracle
Anwendungsgebiete und Vorteile
Analytischen Funktionen gibt es bei Oracle schon seit der Version 8i. Am Beispiel einiger ausgewählter Funktionen stellen wir Ihnen den sinnvollen Einsatz in einigen Anwendungsfällen aus der Praxis vor.

29 ..... JSON in Oracle – Von einem JSON-Objekt
zur relationalen Datenbank
Auch wenn Oracle JSON als Datentyp nicht unterstützt, gibt es die Möglichkeit, mit JSON-Objekten in einer relationalen Datenbank zu arbeiten – wir zeigen Ihnen wie.

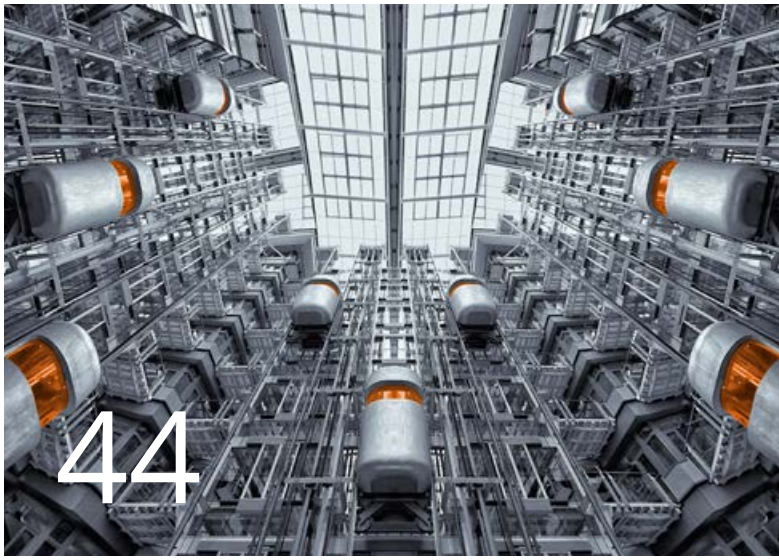
37 ..... Standardverfahren in APEX
Dateien in Oracle APEX
Die direkte Speicherung von Dateien in der Datenbank ist ein Vorteil von APEX. Wir stellen Ihnen ein Standardverfahren vor, wie Dateien im Oracle Application Express hochgeladen und bereitgestellt werden können.

41 ..... Eine für Alle
Polymorphic Table Function
Mit dem Datenbank-Release 18c ist die Polymorphic Table Function eingeführt worden. In diesem Artikel erklären wir die grundlegende Funktionalität anhand eines kleinen Beispiels.

PostgreSQL

26 ..... PostgreSQL - Datenbank der Zukunft (Teil I)
PostgreSQL Enterprise Features
In dieser neuen Reihe machen wir Sie mit den Features der Open-Source-Datenbank PostgreSQL bekannt und verdeutlichen, dass eine Vielzahl dieser Features bereits denen eines kommerziellen RDBMS gleichkommt.

PostgreSQL Enterprise Features



Performance-Faktoren von IBM DataStage

Entwicklung

15 ..... Graylog
Zuverlässiges und kostenneutrales Logging und Monitoring von verteilten Systemen
Informationen vom Nutzerverhalten über Fehlererkennung und Performance-Risiken bis hin zur Intrusion-Detection können durch angepasstes Logging gesammelt werden. Das Tool Graylog bietet hier die Möglichkeit, dies über verteilte Systeme zu tun.

31 ..... Module statt Microservices
Auch mit klassischer Java-EE-Entwicklung ist agile Entwicklung möglich
Agilität und Effizienz schließen sich nicht aus – Microservices und Monolithen sind kein Widerspruch. Wir vergleichen beide Möglichkeiten und zeigen die Vor- und Nachteile.

Microsoft

48 ..... Microsoft SharePoint (Teil II)
Suche und Kommunikation
SharePoint bietet eine sehr leistungsfähige Suchfunktion. Welche Möglichkeiten diese Suche bietet und mit welchen Tools aus der Office-365-Produktfamilie diese kombiniert werden kann, erläutern wir im zweiten Teil der Reihe.

Aktuell

22 ..... Ready for IT? – Ausbildung und Karriere bei ORDIX
24 ..... Seminarübersicht 2019

Impressum

Herausgeber:	ORDIX AG Aktiengesellschaft für Softwareentwicklung, Beratung, Schulung und Systemintegration, Paderborn
Redaktion/Layout:	Jens Pothmann, Elisabeth Herick
V.i.S.d.P.:	Christoph Lafeld, Wolfgang Kögler
Anschrift der Redaktion:	ORDIX AG | Karl-Schurz-Straße 19a | 33100 Paderborn Tel.: 0 52 51 10 63 -0 | Fax: 0180 1673490
Auflage:	6.500 Exemplare
Druck:	Druckerei Bösmann, Detmold
Bildnachweis:	© pexels.com © pixabay.com © pixabay.com | Comfreak | Astronaut Weightless Space © pexels.com | Aleksandar Pasaric | Architektur bewölkt business draußen © pixabay.com | taddtography | Sun Tunnel Lucin Utah © unsplash.com | Frantzou Fleurine | Rock by Sea © pixabay.com | Aufzug Berlin Ludwig Erhard Haus
Autoren:	Daniel Dyck, Markus Fiegler, Tobias Flüter, Olaf Hein, Matthias Jung, Wolfgang Kögler, Lars Hendrik Korte, Christian Rädisch, Adi Schmidt, David Sedlbauer, Tobias Ummler, Andreas Upgang, Dennis Vinueza
Copyright:	Alle Eigentums- und Nachdruckrechte, auch die der Übersetzung, der Vervielfältigung der Artikel oder von Teilen daraus, sind nur mit schriftlicher Zustimmung der ORDIX AG gestattet.
Warenzeichen:	Einige der aufgeführten Bezeichnungen sind eingetragene Warenzeichen ihrer jeweiligen Inhaber. ORDIX® ist eine registrierte Marke der ORDIX AG.
Haftung:	Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden.
Kontaktadresse:	redaktion@ordix.de

Sie können die Zusendung der ORDIX® news jederzeit ohne Angabe von Gründen schriftlich (z.B. Brief, Fax, E-Mail) abbestellen.

MySQL Document Store

Zwischen den Welten: SQL oder NoSQL

NoSQL-Datenbanken setzen sich immer mehr durch. Gerade die Entwickler schätzen den hohen Freiheitsgrad, der aus dem Verzicht auf starre Regeln entsteht, den jedoch traditionelle SQL-Datenbanken über Schemata und Tabellenstrukturen erzwingen. Aber warum sollte man sich zwischen den beiden Welten entscheiden müssen, wenn es doch Produkte gibt, die beide Welten verbinden? Dieser Artikel stellt den JSON Document Store der MySQL-Version 8.0 vor.

Sicht der Dinge ...

Für den klassischen SQL-Anwender besteht die Welt nach wie vor aus Datenbanken (hier: **company**), Tabellen (**employee_sql**), Spalten (**id**, **vorname**, ...) und Datensätzen.

```
mysql> select * from company.employee_sql;
+----+----------+-----------+----------+
| id | vorname  | nachname  | firma    |
+----+----------+-----------+----------+
|  1 | Matthias | Jung      | ORDIX AG |
|  2 | Theo     | Test      | ORDIX AG |
|  3 | Max      | Mustermann | ORDIX AG |
+----+----------+-----------+----------+
```

Beispiel eines klassischen relationalen Datenmodells am Beispiel der Tabelle **employee_sql**.

Der NoSQL-Nutzer sieht zwar auch die Datenbank (**company**; hier als Objekt **db** instanziert). Die Tabelle, Spalten und Datensätze subsummiert er jedoch unter den Begrifflichkeiten **Collection**, **Document** und **Field**.

```
company JS> db.employee_json.find()
[
    {
        "_id": "00005c35f7f40000000000000001",
        "department": 1,
        "firma": "ORDIX AG",
        "nachname": "Jung",
        "vorname": "Matthias"
    }, …
```

Dasselbe Datenmodell eines Mitarbeiters als JSON-Objekt.

Etwas unstrukturiert ...

Bereits seit der Version 5.7 gibt es bei MySQL den Datentypen JSON. Er erlaubt es, pro Zeile einer InnoDB Tabelle 1 GB an JSON-Informationen zu speichern. Der Datentyp prüft dabei beim **INSERT** der Zeile, dass der zu speichernde Inhalt validen JSON-Code repräsentiert und hilft so, Syntaxfehler zu vermeiden.

```
mysql> create table employee_json (
             nr int primary key,
             jdoc json ) engine = innodb;
```

```
mysql> insert into employee_json (nr, jdoc)
values (1, ' {"vorname": "Matthias", "nachname":
"Jung", "Firma": "ORDIX AG"} ');
```

So wurden in MySQL 5.7 JSON-Objekte in Tabellen gespeichert.

Gerade etwas komplexere Strukturen mussten vom Entwickler auf diese Weise sehr sorgsam in die klassische SQL-Syntax integriert werden. Dies löste nicht immer nur Freude aus. Auch aus diesem Grund wurde in die aktuelle Version 8.0 ein neues Protokoll integriert, welches auch von der neuen MySQL Shell (in JavaScript, Python und SQL) umgesetzt wird: Das X-Protocol!

X für ein U ...

Für den einen oder anderen altgedienten DBA (und der Autor zählt sich dazu) ist die neue Syntax zugegebenerweise etwas gewöhnungsbedürftig. Entwickler werden aber sicher ihre wahre Freude daran haben. Das oben – in klassischer SQL-Manier – implementierte Beispiel sieht unter Verwendung des X-Protokolls (im JavaScript-Modus) wie folgt aus:

```
mysqlsh root:root@localhost:3306/company
company JS> db=session.getSchema('company')
company JS> db.createCollection('employee_
json');
company JS> db.employee_json.add({
       vorname: "Matthias",
       nachname: "Jung",
       firma: "ORDIX AG"});
company JS> db.employee_json.find()
       [
          {
          "_id": "00005c35f7f40000000000000001",
          "firma": "ORDIX AG",
          "nachname": "Jung",
          "vorname": "Matthias"
          }
       ]
1 document in set (0.0039 sec) Zwischenüber-
schrift
```

CRUDe Zeiten

Das Akronym CRUD steht für: **Create**, **Read**, **Update** und **Delete**. Damit sind die vier grundlegenden Operationen des Datenmanagements gemeint. Das Erzeugen von Datensätzen erfolgt mit der Methode **add()**:

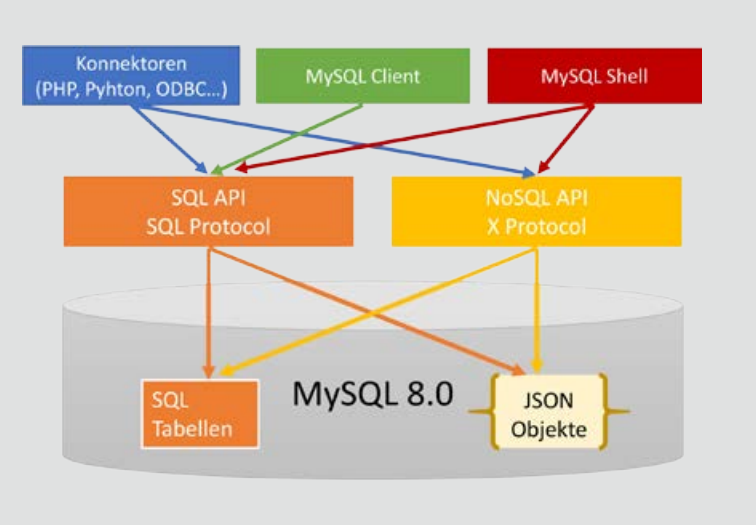


Abb. 1: MySQL-Nutzer haben nun die Wahl, mit welchem Protokoll sie auf ihre Daten zugreifen wollen.

```
db.employee_json.add({Vorname: "Max", Nachna-
me: "Mustermann", Firma: "ORDIX AG"})
```

Das Lesen (Suchen) kann über die Methode **find()** gestartet werden:

```
db.employee_json.find("Vorname = 'Max'");
[
    {
        "Firma": "ORDIX AG",
        "Nachname": "Mustermann",
        "Vorname": "Max",
        "_id": "00005c35f7f40000000000000008"
    }
]
```

Zum Löschen und Ändern von Daten stehen **modify()** und **remove()** bereit:

```
db.employee_json.modify("id    =    '00005c35f
7f40000000000000008'").set("Vorname", "Maximi-
lian");
```

```
db.employee_json.remove("id    =    '00005c35f
7f40000000000000008'")
```

Import- und Exportgeschäft

Auch der Import von bestehenden Daten, z.B. aus JSON-Dokumenten, ist einfach realisierbar. Dateien (auch gerne in einem komprimierten Format) können einfach über eine Import-Funktion der MySQL-Shell geladen werden:

```
mysqlsh root:root@192.168.56.101:64460/com-
pany --import employee_json.json.bz2
```

Brücken bauen, Welten verbinden

Das Datenmodell unserer kleinen Testdatenbank besteht aus einer klassischen relationalen Tabelle (department)

und einer Collection **employee_json** mit JSON-Dokumenten. In der Collection soll ein Feld mit einer Fremdschlüsselinformation aufgenommen werden, über die dann der Name der Abteilung aus der relationalen Tabelle dazugejoined werden kann. Dazu kann das Objekt einfach um ein Attribut erweitert werden. Eine Strukturänderung wie bei klassischen Tabellen (**alter table**… **add column**…) ist nicht notwendig.

```
company JS> db.employee_json.modify('nachname
= "Jung"').set("department", 1);
```

Zugegeben, die Syntax, um die Inhalte des Document-Stores mit relationalen Tabellen zu verknüpfen, ist etwas sperrig, mit ein bisschen Übung jedoch kein Problem:

```
mysql> select department, JSON_
UNQUOTE(doc->"$.nachname")
       from department  join employee_json
       on JSON_UNQUOTE(doc->"$.department")
       = department.nr;

+----------+----------------------------+
| department | JSON_UNQUOTE(doc->"$.nachname")|
+---------+----------------------------+
| IT       | Jung                       |
+---------+----------------------------+
```

Generier Dir einen Index …

MySQL bietet keinen Weg an, ein JSON-Dokument attributspezifisch zu indizieren. Aber auch hier gibt es einen Ausweg. Seit MySQL 5.7 besteht die Möglichkeit, generierte (andere Hersteller nennen es „virtuelle“) Spalten zu erzeugen. Diese wiederum können mit einem Index belegt werden. Um in unserem Beispiel die Information (Attribut) „nachname“ zu verschlagworten, muss zunächst eine solche Spalte erstellt und anschließend indiziert werden.

```
mysql> alter table employee_json add nachna-
me_json varchar(20) as (JSON_UNQUOTE(doc->"$.
nachname"));

mysql> create index i_nachname on employee_
json(nachname_json);
```

Bei der gezielten Suche nach einem Nachnamen innerhalb des JSON-Dokumentes sollte nun der Index verwendetet werden:

```
mysql> explain SELECT * FROM company.employee_
json
       where JSON_UNQUOTE(doc->"$.nachname") =
'Jung';
********** 1. row **************
           id:   1
  select_type:   SIMPLE
        table:   employee_json
   partitions:   NULL
         type:   ref
possible_keys:   i_nachname
          key:   i_nachname
      key_len:   83
          ref:   const
         rows:   1
     filtered:   100
        Extra:   NULL
```

Fazit

Mit den aktuellen Document-Store-Funktionalitäten bietet MySQL die Möglichkeit, die Konzepte von klassischen NoSQL-Datenbanken innerhalb bekannter Infrastrukturlösungen (MySQL) umzusetzen. Dabei lassen sich die beiden Welten „SQL“ und „NoSQL“ problemlos kombinieren und nebeneinander betreiben. Für den DBA hat dies den Vorteil, dass kein neues Produkt erlernt und in Betrieb gebracht werden muss. All die für MySQL bereits entwickelten Prozesse – vom Backup-, über das Monitoring bis hin zu Maßnahmen der Betriebsstabilität und -sicherheit (HA- & DR-Lösungen) – können einfach weitergeführt werden und müssen nicht für ein weiteres Datenbankprodukt erneut entwickelt werden. Selbst die altbekannten administrativen SQL-Kommandos können weiter genutzt werden, wenn der DBA nicht gewillt ist, sich der neuen JavaScript Syntax anzunehmen.

Der Entwickler freut sich über die Freiheiten, die ihm die JSON-Dokumente bei der Entwicklung bieten. Zudem muss er bei der Entwicklung nicht mehr in SQL-Kommandos denken, sondern kann objektorientiert mit seinen Daten arbeiten.

Matthias Jung
(info@ordix.de)

SEMINAREMPFEHLUNG

ORDIX Seminar „MySQL Administration“

https://seminare.ordix.de/seminare/oracle/mysql/mysql-administration.html

Links

[1] Weiterführende Informationen zu MySQL Enterprise:
https://www.mysql.com/de/products/enterprise/document_store.html

Bildnachweis

Einrichtung und Betriebsaspekte

Apache Cassandra

Es gab bereits einige Artikel in der ORDIX® news zu Themen wie NoSQL-Datenbanken und auch Cassandra. In diesem Artikel soll es etwas technischer werden. Wir schauen uns die Einrichtung eines Cassandra-Clusters an und gehen auf einige Punkte beim Betrieb einer solchen Umgebung ein.

Architektur

Cassandra wird in einem Cluster betrieben, wobei die zu speichernden Daten im besten Fall mehrfach repliziert auf die Knoten verteilt werden. Empfohlen wird hier eine 3-fach-Replikation. Ein Cluster kann aus nur einem Knoten bestehen, also ein sogenannter Single-Node-Cluster, der z.B. bei lokaler Entwicklung oder Tests sehr nützlich sein kann. Die maximale Anzahl an Knoten ist theoretisch unbegrenzt. In der Praxis könnten aber Monitoring- und Wartbarkeitsgründe gegen eine extrem große Zahl sprechen, wobei einige größere Unternehmen sogar Cluster mit mehr als 1.000 Knoten betreiben.

Die Knoten eines Clusters sind in einem Ring angeordnet, in dem es keinen führenden (Master-)Knoten gibt. Alle sind gleichberechtigt innerhalb des Clusters und es gibt daher keinen Single-Point-Of-Failure. Ein Zugriff auf die Daten ist über jeden Knoten möglich, da sie sich diesbezüglich untereinander austauschen.

Vorüberlegungen

Beim Aufbau eines Cassandra-Clusters ist es von Vorteil, im Vorfeld zu definieren, wie dieser ausgelegt sein soll. Auf wie viele Rechenzentren soll der Cluster verteilt werden? In welchen Racks werden die Server verbaut sein? Wie nah stehen die Racks innerhalb eines Rechenzentrums zusammen? Das sind Fragen, die es zu beantworten gilt, um eine größtmögliche Performance innerhalb der Cassandra zu erreichen. Diese Vorüberlegungen spiegeln sich dann dementsprechend auch in der Konfiguration wider.

Installation und erste Konfiguration

Die Installation der Software ist sehr einfach möglich über das Extrahieren eines Tarballs auf allen Knoten, die zu dem Cassandra-Cluster gehören sollen. Danach könnte theoretisch bereits der Dienst manuell gestartet werden, der dann mit einer Standardkonfiguration laufen würde. Auf diese Weise wäre aber jeder Knoten für sich ein eigener Single-Node-Cluster und könnte keine Replikation der Daten mit den anderen Knoten durchführen. Daher ist vor dem ersten Start die Konfiguration anzupassen, um einige Netzwerkeinstellungen zu definieren.

In der zentralen Konfigurationsdatei **`cassandra.yaml`** sind grundlegend erst einmal zwei Einstellungen vorzunehmen. Zum einen ist zu definieren, auf welcher IP-Adresse der Cassandra-Dienst gestartet werden soll (Standard ist localhost), zum anderen ist eine Liste mit IP-Adressen als Seeds einzutragen (siehe Abbildung 1). In dieser Liste werden bis zu drei weitere Knoten des gesamten Clusters eingetragen. Mit diesen verbindet sich Cassandra beim Start, um sich über die Beschaffenheit der Cluster-Umgebung auszutauschen. Dieser Austausch geschieht mittels des Gossip Protocols, wobei die Knoten über deren Standort (Rechenzentrum, Rack) und Status „tratschen“ (engl.: to gossip). Falls sich der Cluster über mehrere Rechenzentren verteilt, sollte aus jedem Rechenzentrum mindestens ein Knoten in die Seed-Liste aufgenommen werden.

Noch mehr „Klatsch und Tratsch“

Zur Unterstützung der Knoten bei ihrem Informationsaustausch ist es sinnvoll, zu definieren, wo genau sich ein Knoten befindet. Bei der Aufteilung des Cassandra-Clusters auf mehrere Rechenzentren ist in diesem Zusammenhang der Parameter **`endpoint_snitch`** in der **`cassandra.yaml`** zu prüfen. Für den Aufbau in einer produktiven Umgebung wird empfohlen, diesen Wert auf **`GossipingPropertyFileSnitch`** zu setzen.

Nun wertet der Cassandra-Dienst die Angaben bzgl. **`dc=`** und **`rack=`** in der Datei **`cassandra-rackdc.properties`** aus.

Bsp.:

```
dc=rechenzentrum1
rack=rack1
```

Diese Konfigurationsänderungen sind selbstverständlich auf allen Knoten durchzuführen und der Cassandra-Dienst ist danach neu zu starten.

```
seed_provider:
 - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
      - seeds: "IP_RZ1_A,IP_RZ1_B,IP_RZ2_A"
```

Abb. 1: Seed-Konfiguration in der cassandra.yaml

Anhand dieser Informationen kann nun Cluster-intern bestimmt werden, wie „weit“ einzelne Knoten voneinander entfernt sind und welches somit der schnellste Weg zum Speichern und Lesen von Daten ist.

Berechtigungssystem

In der Standardkonfiguration eines Cassandra-Knotens ist die Überprüfung der Authentifizierung (wer darf sich anmelden?) und Autorisierung (was darf ein Benutzer tun?) nicht aktiviert, was in der **`cassandra.yaml`** an folgenden Parametern zu sehen ist:

```
authenticator:      AllowAllAuthenticator
authorizer:         AllowAllAuthorizer
```

Dadurch ist es für jeden Benutzer möglich, auch ohne eine valide Benutzername-Passwort-Kombination, Zugriff auf die Daten innerhalb der Datenbank zu bekommen und alle Aktionen (Lesen, Löschen, Ändern, …) mit diesen Daten durchzuführen. Diese Parameter sollten zwingend z.B. wie folgt angepasst werden:

```
authenticator:      PasswordAuthenticator
authorizer:         CassandraAuthorizer
```

Damit wird die Anmeldung mit einem gültigen Benutzernamen und Passwort aktiviert und die Rechte zum Arbeiten mit den gespeicherten Daten können so für jeden Benutzer bzw. auch jede Rolle angepasst werden.

Neben der reinen Authentifizierung und Autorisierung ist es empfehlenswert, die Kommunikation der Cassandra-Knoten untereinander sowie vom Client in den Cluster hinein mittels TLS zu verschlüsseln. Hierzu bietet die **`cassandra.yaml`** entsprechende Möglichkeiten. Voraussetzung hierfür ist natürlich das Vorhandensein einer entsprechenden PKI zum Erstellen und Signieren von Zertifikaten, die dann für die sichere Kommunikation genutzt werden.

Backup

Manche sind der Meinung, dass ein Backup in einem Cassandra-Cluster nicht nötig ist, weil die Daten mit einer 3-fach-Replikation bereits genug gesichert sind. So könnten bis zu zwei Knoten ausfallen und die Daten wären immer noch verfügbar. Manchmal ist aber der Mensch das Problem, denn wenn ein Anwender ungewollt Daten über ein **`delete`**- oder **`truncate`**-Kommando löscht, so wird diese Löschung natürlich auch repliziert. Um dem vorzubeugen, können verschiedene Vorkehrungen getroffen werden.

Sapshots werden für den aktuellen Stand eines Nodes erstellt. Es kann definiert werden, ob dieser Snapshot für den gesamten Node (**`nodetool snapshot`**), explizit für die Daten eines speziellen Keyspaces (**`nodetool snapshot keyspace1`**) oder nur auf Tabellenebene (**`nodetool snapshot –kt keyspace1.tabelle1`**) durchgeführt werden soll. Snapshots werden im Filesystem als Hardlinks auf Datenbankdateien einer Tabelle realisiert.

Der Parameter **`auto_snapshot`** sollte aktiviert bleiben. Dadurch ist sichergestellt, dass vor dem Löschen einer Tabelle oder eines Keyspaces automatisch ein Snapshot erstellt wird. Neben der Erstellung von Snapshots sollte auch das Filesystem, in dem die Datenverzeichnisse liegen, regelmäßig gesichert werden. Bestenfalls liegen diese auf einem Logical Volume eines LVM und können auch hier über die entsprechende Snapshot-Funktion gesichert werden.

Lars Hendrik Korte
(info@ordix.de)

Anwendungsgebiete und Vorteile

Analytische Funktionen in Oracle

Analytische Funktionen sind mit der Oracle Version 8i eingeführt und ständig erweitert worden. Die Fähigkeit, komplexe Fragestellungen bei Datenauswertungen im reinen überschaubaren SQL-Code abzuwickeln, hat analytische Funktionen sehr beliebt gemacht. Die Alternative wäre ein komplexer SQL-Code oder der Umweg über den prozeduralen Code. In diesem Artikel werden die Besonderheiten und Grenzen der analytischen Funktionen in Oracle am Beispiel einiger ausgewählte Funktionen vorgestellt. Abgerundet wird der Artikel mit dem sinnvollen Einsatz und einigen Anwendungsfällen aus der Praxis.

Was sind analytische Funktionen?

Als analytische Funktionen in einer Oracle-Datenbank werden Funktionen bezeichnet, die auf einer Menge der Ergebnisdatenzeilen ausgeführt werden. Anders als bei Gruppenfunktionen mit einer **GROUP BY**-Klausel, bei denen nur die Aggregatsdatenzeilen ausgegeben werden, werden bei analytischen Funktionen immer alle Datenzeilen ausgegeben, inklusive der Ergebnisse der Funktionen, die auf einer Menge der Ergebnisdatenzeilen ausgeführt wurden (siehe Abbildung 1). Die Menge der Datenzeilen, auf denen eine analytische Funktion aufgerufen wird, kann pro Datensatz unterschiedlich sein.

Ein zweites wesentliches Merkmal analytischer Funktionen ist die Fähigkeit, auf benachbarte Datenzeilenwerte zuzugreifen. Ein Zugriff auf diese Datenzeilenwerte ohne analytische Funktionen ist in der Regel nur mit einem zusätzlichen Tabellenzugriff verbunden, was zum einem einen komplizierteren Code und zum anderen einen erhöhten Ressourcenverbrauch – sowie eine höhere Laufzeit – zur Folge hat.

Anwendungsgebiete und Vorteile

Analytische Funktionen werden vorwiegend im Data-Warehouse-Umfeld eingesetzt. Die Hauptanwendungsfälle sind unter anderem Aufsummierung von Werten in einem Datenzeilenbereich, gleitende Durchschnitte, zeilenübergreifende Datenwertvergleiche oder Rankings.

Der wesentliche Vorteil analytischer Funktionen ist die Vermeidung der mehrfachen Zugriffe auf dieselbe Tabelle innerhalb einer SQL-Anweisung. Die meisten SQL-Anweisungen mit Unterabfragen, Inline-Views oder Self-Joins lassen sich mithilfe analytischer Funktionen so umschreiben, dass lediglich ein Tabellenzugriff notwendig ist. Daraus resultieren in der Regel ein reduzierter Ressourcenverbrauch und eine schnellere Laufzeit. Eine Implementierung ohne analytische Funktionen führt darüber hinaus zu einem sehr umständlichen, schlecht wartbaren und teilweise auch prozeduralen Code.

Syntax analytischer Funktionen

Abbildung 2 zeigt die Syntax der analytischen Funktionen. Neben einer Gruppenfunktion, die als analytische Funktion ausgeführt werden kann, können optional eine **PARTITION BY**- und eine **ORDER BY**-Klausel angegeben werden.

Mithilfe der **PARTITION BY**-Klausel kann eine Gruppierung der Ergebnismenge durchgeführt werden, ähnlich einer **GROUP BY**-Klausel bei einer Gruppierung. Alle Zeilen einer Ergebnismenge, die dem Gruppierungskriterium entsprechen, werden zusammengefasst. Ohne Angabe der **PARTITION BY**-Klausel werden alle Datenzeilen als eine Partition betrachtet.

Die **ORDER BY**-Klausel gibt dagegen eine Sortierung innerhalb einer Partition vor. Das Besondere an der **ORDER BY**-Klausel ist, dass hier eine Teilmengenextraktion aktiviert wird, bei der nicht alle Zeilen zu einer Partition zusammengefasst werden, sondern nur ein bestimmtes „Fenster“. Das standardmäßige Fenster besteht aus Zeilen der jeweiligen Partition bis zur aktuellen Zeile.

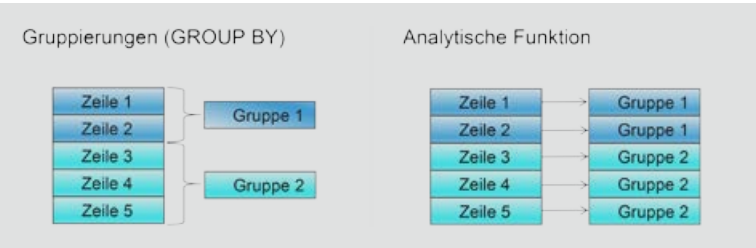


Abb. 1: Netzwerk-Architektur

```
SELECT <spalte>,
  funktion([argumente]) OVER (
    [PARTITION BY <spalte>]
    [ORDER BY <spalte> |
     ORDER BY <spalte> WINDOWING-Klausel]
  ) ana_spalte
FROM <tabelle>
WHERE ...
```

Abb. 2: Syntax analytischer Funktionen

Syntax WINDOWING-Klausel:

```
ROWS | RANGE BETWEEN <startpunkt> AND <endpunkt>
ROWS | RANGE BETWEEN <startpunkt>
```

Startpunkt / Endpunkt	Beschreibung
UNBOUNDED PRECEDING	alle bisherigen Zeilen ab Beginn der Partition
<N> PRECEDING	die letzten <N> Zeilen vor der aktuellen Zeile
UNBOUNDED FOLLOWING	alle nachfolgenden Zeilen bis zum Ende der Partition
<N> FOLLOWING	die nächsten <N> Zeilen ab der aktuellen Zeile
CURRENT ROW	die aktuelle Zeile

Abb. 3: Start- und Endpunkt der WINDOWING-Klausel.

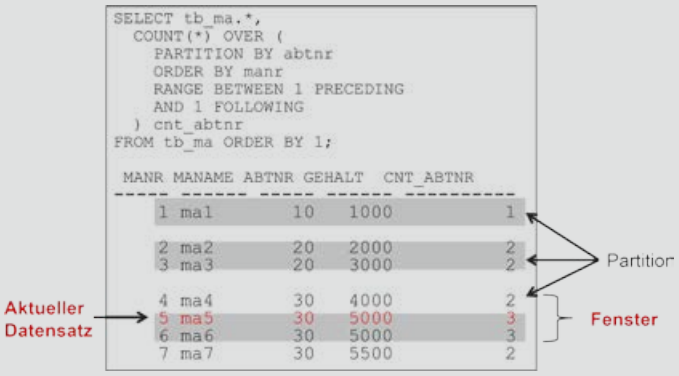


Abb 4: Beispiel WINDOWING-Klausel

```
SELECT tb_ma.*,
  DENSE_RANK() OVER (ORDER BY gehalt) dense_rank,
  RANK()       OVER (ORDER BY gehalt) rank,
  ROW_NUMBER() OVER (ORDER BY gehalt) row_number
FROM tb_ma ORDER BY 1;

 MANR MANAME ABTNR GEHALT DENSE_RANK       RANK ROW_NUMBER
----- ------ ----- ------ ---------- ---------- ----------
    1 ma1       10   1000          1          1          1
    2 ma2       20   2000          2          2          2
    3 ma3       20   2000          2          2          3
    4 ma4       20   3000          3          4          4
```

Abb. 5: Beispiel mit Ranking-Funktionen

Teilmengenextraktion mit WINDOWING-Klausel

Ein „Fenster“ definiert den Bereich von Datensätzen, auf dem eine analytische Funktion ausgeführt werden soll (Teilmengenextraktion). Voraussetzung zur Verwendung der **WINDOWING**-Klausel ist die **ORDER BY**-Klausel (sortierte Ergebnismenge), d.h. mit der **ORDER BY**-Klausel wird die **WINDOWING**-Klausel aktiviert.

In der **WINDOWING**-Klausel werden entweder Start- und Endpunkt oder nur der Startpunkt des Fensters angegeben. Wird lediglich der Startpunkt angegeben, so ist die aktuelle Zeile automatisch der Endpunkt. Mögliche Werte für Start- und Endpunkt und die genaue Syntax der **WINDOWING**-Klausel stellt Abbildung 3 dar.

Die Einschränkung des Fensters ist zum einen auf Zeilenebene mit dem Schlüsselwort **ROWS** und zum anderen auf logischer Ebene mit **RANGE** möglich. Bei der Einschränkung auf Zeilenebene (**ROWS**) kann über ein Offset die Anzahl der Zeilen angegeben werden, auf die sich ein Fenster beziehen soll (siehe Abbildung 4).

Über **RANGE** kann dagegen ein Fenster auf der logischen Werteebene definiert werden. Die Spalten, auf die sich der logische Wertebereich beziehen soll, werden in der **ORDER BY**-Klausel angegeben und können vom Datentyp **NUMBER**, **DATE** oder **TIMESTAMP** sein.

Ranking

Ranking ist in der Praxis einer der Hauptanwendungsfälle der analytischen Funktionen. Mit einem Ranking lässt sich eine Datenzeilen-Rangfolge ermitteln oder Top-n-Auswertungen durchführen. Dazu stellt Oracle folgende Funktionen zur Verfügung:

- **DENSE_RANK**
- **RANK**
- **ROW_NUMBER**

Die Funktionen **DENSE_RANK** und **RANK** liefern bei gleichen Werten den gleichen Rang zurück. Die Funktion **ROW_NUMBER** setzt bei gleichen Werten die Zählung dagegen fort. Die **RANK**-Funktion hinterlässt darüber hinaus im Gegensatz zu **DENSE_RANK** nach gleichen Werten eine Lücke in der Rangfolge (siehe Abbildung 5).

Zugriff auf benachbarte Datenzeilenwerte

Ein weiteres Anwendungsbeispiel für analytische Funktionen ist der Zugriff auf benachbarte Datenzeilenwerte. Dieser lässt sich mit folgenden Funktionen realisieren:

- **LAG**
- **LEAD**

Soll auf die vorhergehenden Datenzeilen zugegriffen werden, so kann die **LAG**-Funktion verwendet werden. Die

LEAD-Funktion kann dagegen auf die darauffolgenden Datenzeilen zugreifen. In der Praxis werden diese beiden Funktionen häufig für zeilenübergreifende Datenfeldvergleiche wie Differenzbildung oder prozentuale Veränderungen eingesetzt. Wie aus Abbildung 6 ersichtlich, greift die Variante ohne analytische Funktion zweimal auf die Tabelle **TB_MA** zu und benötigt aufgrund der mehr gelesenen Datenblöcke auch eine längere Laufzeit.

Werteverteilung der Datenzeilenwerte

Auch für die Ermittlung der Werteverteilung der Datenzeilenwerte können analytische Funktionen verwendet werden. Dazu stellt Oracle folgende Funktionen zur Verfügung:

- **CUME_DIST**
- **PERCENT_RANK**
- **NTILE**
- **RATIO_TO_REPORT**

Die beiden Funktionen **CUME_DIST** und **PERCENT_RANK** liefern den Anteil der Zeilen an der Gesamtanzahl aller Zeilen, die kleiner oder gleich dem Datenzeilenwert der aktuellen Zeile sind. Der Rückgabewert der Funktionen liegt zwischen 0 und 1, wobei bei der Funktion **CUME_DIST** die 0 exklusive und bei der Funktion **PERCENT_RANK** die 0 inklusive ist.

Bei der Funktion **CUME_DIST** werden außerdem gleiche Werte derselben höchsten kumulativen Werteverteilung zugeordnet und bei der Funktion **PERCENT_RANK** der niedrigsten kumulativen Werteverteilung.

Sollen Datenzeilen in n gleich große Mengen aufgeteilt werden, um z.B. einen sehr großen Datenbestand in mehrere Tabellen aufzuteilen, so kann die **NTILE**-Funktion verwendet werden. Die Anzahl der Mengen kann mit einem Übergabeparameter bestimmt werden. Dabei muss allerdings berücksichtigt werden, dass gleiche Werte unterschiedlichen Mengen zugeordnet werden können, da die Funktion **NTILE** nicht sicherstellt, dass gleiche Werte immer nur einer Menge zugeordnet werden.

Mit der **RATIO_TO_REPORT**-Funktion kann schließlich – anders als bei den Funktionen **CUME_DIST** und **PERCENT_RANK**, wo die Werteverteilung anhand der Anzahl der Zeilen durchgeführt wird **–** eine Werteverteilung bezogen auf die Werte ermittelt werden. Dabei liefert die Funktion **RATIO_TO_REPORT** den Anteil eines Wertes an der Gesamtsumme aller Werte einer Spalte zurück (siehe Abbildung 7).

Anwendungsbeispiele aus der Praxis

Die typischen Kandidaten für den Einsatz von analytischen Funktionen sind SQL-Anweisungen mit mehrfachen Zugriffen auf dieselben Tabellen, wie es häufig bei Unterabfragen, Inline-Views oder Self-Joins der Fall ist. Die Abbildungen 8, 9 und 10 stellen weitere typische Anwendungsfälle für den Einsatz von analytischen Funktionen dar. Alle diese Beispiele haben gemeinsam, dass bei der Variante mit analytischer Funktion im Gegensatz zu der Variante ohne analytische Funktion lediglich ein Tabellenzugriff notwendig ist.

```
-- Beispiel ohne analytischer Funktion
SELECT manr, maname, abtnr, gehalt, gehalt-gehalt_vor diff FROM (
  SELECT manr, maname, abtnr, gehalt, (
    SELECT * FROM (
      SELECT m2.gehalt FROM tb_ma m2 WHERE m2.manr < m1.manr
      ORDER BY m2.manr DESC
    ) WHERE ROWNUM < 2
  ) gehalt_vor
  FROM tb_ma m1
) ORDER BY manr;

-- Beispiel mit analytischer Funktion
SELECT manr, maname, abtnr, gehalt, gehalt-gehalt_vor diff FROM (
  SELECT manr, maname, abtnr, gehalt,
    LAG(gehalt) OVER (ORDER BY manr) gehalt_vor
  FROM tb_ma
) ORDER BY manr;

   MANR MANAME        ABTNR     GEHALT GEHALT_DIFF
------- -------- ---------- ---------- -----------
      1 ma1              10       1000
      2 ma2              20       2000        1000
      3 ma3              20       3500        1500
      4 ma4              30       4300         800
```

Abb 6: Beispiel mit Zugriff auf Vorgängerdatensatz

```
SELECT manr, abtnr, gehalt,
  CUME_DIST()     OVER (ORDER BY abtnr) cd_abtnr,
  PERCENT_RANK() OVER (ORDER BY abtnr) pr_abtnr,
  NTILE(3)        OVER (ORDER BY manr)  nt_3_manr,
  ROUND(RATIO_TO_REPORT(gehalt) OVER (), 3) rtr_geh
FROM tb_ma ORDER BY 1;

MANR ABTNR GEHALT   CD_ABTNR   PR_ABTNR  NT_3_MANR    RTR_GEH
---- ----- ------ ---------- ---------- ---------- ----------
   1    10   1000         ,2          0          1        ,083
   2    20   2000         ,8        ,25          1        ,167
   3    20   2000         ,8        ,25          2        ,167
   4    20   3000         ,8        ,25          2         ,25
   5    30   4000          1          1          3        ,333
```

Abb 7: Beispiel mit Werteverteilung

```
-- Beispiel ohne analytischer Funktion
SELECT m1.*, m2.avg_gehalt
FROM tb_ma m1, (
  SELECT abtnr, AVG(gehalt) avg_gehalt FROM tb_ma GROUP BY abtnr
) m2 WHERE m1.abtnr = m2.abtnr
ORDER BY manr;

-- Beispiel mit analytischer Funktion
SELECT tb_ma.*, AVG(gehalt) OVER (PARTITION BY abtnr) avg_gehalt
FROM tb_ma ORDER BY manr;

  MANR MANAME        ABTNR     GEHALT AVG_GEHALT
------ -------- ---------- ---------- -----------
     1 ma1              10       1000        1000
     2 ma2              20       2000        2500
     3 ma3              20       3000        2500
     4 ma4              30       4000        5125
     5 ma5              30       6000        5125
     6 ma6              30       5000        5125
     7 ma7              30       5500        5125
```

Abb 8: Beispiel mit Durchschnittsgehalt pro Abteilung

```
-- Beispiel ohne analytischer Funktion
SELECT manr, maname, abtnr, gehalt FROM tb_ma WHERE manr IN (
  SELECT MIN(manr) FROM tb_ma WHERE (abtnr, gehalt) IN (
    SELECT abtnr, MAX(gehalt) FROM tb_ma GROUP BY abtnr
  ) GROUP BY abtnr, gehalt
) ORDER BY 1;

-- Beispiel mit analytischer Funktion
SELECT manr, maname, abtnr, gehalt FROM (
  SELECT tb_ma.*, ROW_NUMBER()
    OVER (PARTITION BY abtnr ORDER BY gehalt DESC, manr) rn
  FROM tb_ma
) WHERE rn = 1 ORDER BY manr;

  MANR MANAME         ABTNR     GEHALT
------ -------- ---------- ----------
     1 ma1              10       1000
     3 ma3              20       3000
     5 ma5              30       6000
```

Abb. 9: Beispiel mit bestverdienendem Mitarbeiter pro Abteilung

```
-- Beispiel ohne analytischer Funktion
SELECT manr, maname, abtnr, gehalt, (
  SELECT SUM(gehalt) FROM tb_ma m2
  WHERE m2.abtnr = m1.abtnr AND m2.manr <= m1.manr
) sum_gehalt
FROM tb_ma m1 ORDER BY manr;

-- Beispiel mit analytischer Funktion
SELECT manr, maname, abtnr, gehalt,
  SUM(gehalt) OVER (PARTITION BY abtnr ORDER BY manr) sum_gehalt
FROM tb_ma ORDER BY manr;

  MANR MANAME         ABTNR     GEHALT SUM_GEHALT
------ -------- ---------- ---------- ----------
     1 ma1              10       1000       1000
     2 ma2              20       2000       2000
     3 ma3              20       3000       5000
     4 ma4              30       4000       4000
     5 ma5              30       6000      10000
     6 ma6              30       6000      16000
     7 ma7              30       5500      21500
```

Abb. 10: Beispiel mit kummuliertem Gehalt pro Abteilung

Funktion	WINDOWING-Klausel zulässig	ORDER BY-Klausel
AVG	Ja	erlaubt
COUNT	Ja	erlaubt
CUME_DIST	Nein	notwendig
DENSE_RANK	Nein	notwendig
FIRST_VALUE	Ja	erlaubt
LAG	Nein	notwendig
LAST_VALUE	Ja	erlaubt
LEAD	Nein	notwendig
LISTAGG	Nein	nicht erlaubt
MAX	Ja	erlaubt
MIN	Ja	erlaubt
NTH_VALUE	Ja	erlaubt
NTILE	Nein	notwendig
PERCENT_RANK	Nein	notwendig
RANK	Nein	notwendig
RATIO_TO_REPORT	Nein	nicht erlaubt
ROW_NUMBER	Nein	notwendig
SUM	Ja	erlaubt

Abb. 11: Besonderheiten der Syntax analytischer Funktionen

Besonderheiten und Grenzen analytischer Funktionen

Analytische Funktionen sind in der **WHERE**-Klausel nicht zulässig und können nur in der **SELECT**-Liste und **ORDER BY**-Klausel verwendet werden. Der Grund der Einschränkung liegt in der Reihenfolge der SQL-Verarbeitung, da die analytischen Funktionen erst vor der **ORDER BY**-Klausel und nach der **Join**-Verknüpfung, **WHERE**-, **GROUP BY**- und **HAVING**-Klauseln ausgewertet werden. Soll das Ergebnis einer analytischen Funktion in einer **WHERE**-Klausel eingeschränkt werden, so kann dies z.B. mithilfe einer Inline-View vorgenommen werden.

Außerdem ist die **ORDER BY**- und **WINDOWING**-Klausel nicht bei allen analytischen Funktionen erlaubt, bei einigen Funktionen aber sogar zwingend erforderlich (siehe Abbildung 11).

Fazit

Zusammenfassend kann gesagt werden, dass die analytischen Funktionen die Komplexität der SQL-Anweisungen deutlich vereinfachen und gleichzeitig auch häufig die Performance verbessern. Aus diesem Grund werden diese Funktionen in der Praxis vor allem im Data-Warehouse-Umfeld eingesetzt. Die vielen Vorteile können auch den Einsatz z.B. in OLTP-Systemen interessanter gestalten.

Markus Fiegler
(info@ordix.de)

SEMINAREMPFEHLUNG

ORDIX Seminar „SQL-Workshop für Experten“

https://seminare.ordix.de/seminare/oracle/entwicklung/oracle-sql-workshop-für-experten.html

Graylog

Zuverlässiges und kostenneutrales Logging und Monitoring von verteilten Systemen

Informationen vom Nutzerverhalten über Fehlererkennung und Performance-Risiken bis hin zur Intrusion-Detection können durch angepasstes Logging gesammelt werden. Nicht nur für verteilte Systeme bietet sich der Einsatz eines zentralen Logging-Systems an, um die Informationen an einem Ort sammeln und auswerten zu können. So ist es auch denkbar, die Logs der Anwendung und die Logs der benutzten Infrastruktur zusammen an einem Ort verwalten zu können.

Logging

Aus Log-Nachrichten können viele unterschiedliche Informationen gewonnen werden. Informationen vom Nutzerverhalten über Fehlererkennung und Performance-Risiken bis hin zur Intrusion-Detection können durch angepasstes Logging gesammelt werden. Der häufigste Fall ist aber weiterhin das Erkennen von Fehlern. Gerade wenn der Betrieb von anderen Personen als den Entwicklern durchgeführt wird, ist es unabdingbar, dass es einen einfachen Zugriff auf alle Log-Nachrichten des verteilten Systems gibt. Eine Herausforderung bei verteilten Systemen, die ggf. auch noch mehrschichtig aufgebaut sind, ist die Vielzahl der Log-Nachrichten (ggf. verstreut über mehrere Logdateien). So gibt es in der Regel eine Log-Datei pro verteiltes System. In mehrschichtig aufgebauten Systemen kann es für einige Schichten noch zusätzliche Log-Dateien geben, beispielsweise ein „Slow Query Log“ für eine MySQL-Datenbank. Wenn die Systeme auch noch horizontal skaliert werden, explodiert die Anzahl der Log-Dateien.

Zentrales Logging

Ein zentrales Logging sammelt die anfallenden Log-Nachrichten ein und stellt sie zentral für die Auswertung zur Verfügung. Ein zentrales Logging muss folgende Aufgaben übernehmen:

- Transfer: Empfangen/Abholen von Log- und Monitor-Daten
- Datenhaushalt: Die Log- und Monitor-Daten werden in der Regel sofort verarbeitet, sie werden aber auch persistiert.
- Autorisierung des Zugriffs auf die Log-Daten
- Homogenisierung: Die Log- und Monitor-Daten werden in ein einheitliches Format konvertiert, damit über Dateigrenzen hinweg nach gleichen Feldern gesucht werden kann.
- Analyse: Neben der Fehlerauswertung können auch weitere Zusammenhänge aus den Log-Nachrichten extrahiert werden:
  1. Eine Analyse des Nutzerverhaltens ist möglich (Benefit: Anpassung der Software an das Nutzerverhalten).
  2. Die Frequenz der Service-Aufrufe lässt sich ermitteln (Ressourcen für Services können somit besser dimensioniert werden – überflüssige Services können identifiziert werden).
  3. Der zeitliche Verlauf von Performance-Daten ermöglicht eine Prognose: frühzeitiges Erkennen von Performance-Engpässen.
- Alarmierung: Falls definierte Schwellwerte erreicht bzw. überschritten werden, dann soll das zentrale Logging Alarmierungen anstoßen z.B. in Form von Alarmierungs-E-Mails.

Zentrales Logging umgesetzt mit Graylog

Für die Implementierung eines zentralen Loggings, gibt es etliche Alternativen. Eine weit verbreitete Möglichkeit ist der ELK-Stack, allerdings ist die kostenfreie Version für Produktionseinsätze nicht ratsam, da u.a. eine Benutzerverwaltung fehlt. In diesem Artikel werde ich Graylog vorstellen.

Ein wesentlicher Vorteil von Graylog ist, dass die kostenlose Variante eine integrierte Benutzerverwaltung mitbringt. Diese kann auch mit vorhandenen Benutzerverwaltungssystemen wie z.B LDAP zusammenarbeiten.

Installation von Graylog

Graylog ist aus verschiedenen Komponenten aufgebaut (siehe Abbildung 1). Um Einstellungen zu persistieren, wird eine Mongo-DB eingesetzt. Elasticsearch wird benötigt, um die Log-Nachrichten zu indexieren und zu persistieren. Die Graylog-UI übernimmt dann das Zusammenspiel der Komponenten und bietet zahlreiche In- und Outputs, um mit anderen Systemen zu kommunizieren. Graylog kann auf verschiedene Art und Weise installiert werden [1]. Ein schneller Einstieg in Graylog lässt sich durch die Installation über Docker gewinnen, da hier kein Betriebssystem-spezifisches Wissen erforderlich ist. Die Installation in Docker-Containern kann mit den in Abbildung 2 dargestellten Befehlen durchgeführt werden.

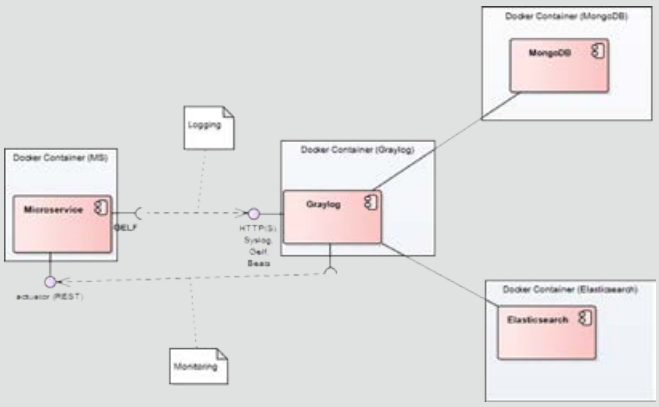


Abb. 1: Graylog Komponenten

```
docker run --name mongo -d mongo:3

docker run --name elasticsearch \
-e "http.host=0.0.0.0" -e ES_JAVA_OPTS="-Xms4g -Xmx4g" \
-d docker.elastic.co/elasticsearch/elasticsearch-oss:6.5.4

docker run --link mongo --link elasticsearch \
-p 9000:9000 -p 12201:12201/udp -p 514:514 \
-e GRAYLOG_WEB_ENDPOINT_URI="http://127.0.0.1:9000/api" \
-d graylog/graylog:2.5
```

Abb. 2: Installation von Graylog in Docker-Containern

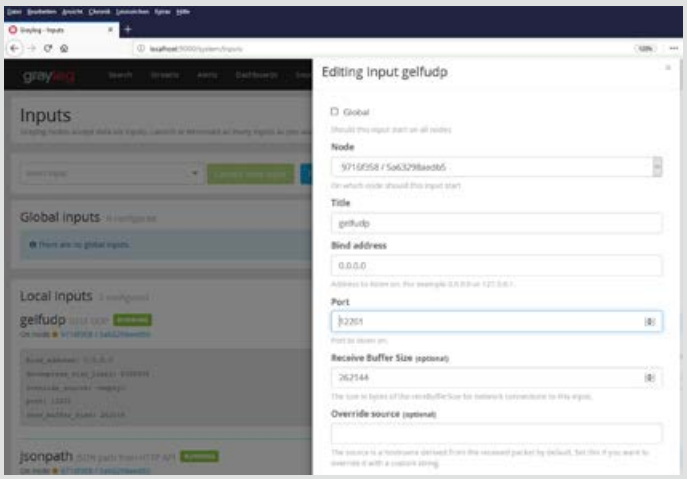


Abb. 3: Graylog gelfUDP Input

Dabei werden die Komponenten MongoDB, Elasticsearch und Graylog (siehe Abbildung 1) in eigenen Docker-Instanzen ausgeführt und netzwerktechnisch verbunden. Bei der Installation richten wir den Zugriff auf die Graylog-UI per Browser ein (localhost:9000). Das Login erfolgt mit dem Benutzer „admin“ und dem Passwort „admin“. Als zweiten Transportweg für Log-Nachrichten richten wir den UDP-Port 12201 ein.

Logging von Microservices

Am Beispiel eines Microservices, der in Java mit dem Spring Boot Framework geschrieben ist, zeige ich eine Möglichkeit, Log-Nachrichten an Graylog zu übermitteln (siehe Abbildung 1). Das Transferieren der Log-Nachrichten an Graylog kann ein **`logback-gelf-appender`** übernehmen. Dazu muss folgende Dependency mit im **`pom.xml`** des Microservices aufgenommen werden:

```
<dependency>
<groupId>de.siegmar</groupId>
    <artifactId>logback-gelf</artifactId>
    <version>1.0.3</version>
</dependency>
```

Anschliessend muss der **`logback-gelf-appender`** noch in der l**`ogback-spring.xml`** konfiguriert werden [2].

Damit Graylog die Log-Nachrichten verarbeitet, muss noch ein **`GELF-UDP-Input`** angelegt werden. Der Input muss auf dem Port 12201, der in der **`logback-spring.xml`** eingetragen ist, eingestellt werden (siehe Abbildung 3).

Logging von Docker-Containern

Nicht nur **`Log`**-Nachrichten von Microservices können per **`GELF`** an Graylog gesendet werden. Alle **`Log`**-Nachrichten eines Docker Containers können per **`GELF`** an Graylog gesendet werden, da Docker das **`GELF`**-Format unterstützt:

```
docker run --log-driver=gelf --log-opt
gelf-address=udp://127.0.0.1:12201 <(Micro-
service-)ContainerName>
```

Monitoring von Microservices

Neben dem Erkennen von Fehlern durch das Logging besteht noch die Notwendigkeit, entstehende Performance-Probleme frühzeitig zu erkennen. Hierfür ist es hilfreich, den Microservice anhand von Metriken zu überwachen.

Diese Art der Überwachung wird auch als Monitoring bezeichnet (siehe Abbildung 1).

Für das Monitoring greift Graylog auf RESTful-Schnittstellen der zu überwachenden Microservices zu (siehe Abbildung 1). Die Auswertung der dadurch angebotenen Daten wird in Graylog konfiguriert. Das Spring-Boot-Framework bietet zu diesem Zweck einen sogenannten Actuator an. Um Microservices zu monitoren, nehmen diese die Dependency **`spring-boot-starter-actuator`** in die **`pom.xml`** auf.

```
<dependency>
<groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator
    </artifactId>
</dependency>
```

Der **`actuator`** stellt mehrere **`metrics`**-Endpoints zur Verfügung [3]. Defaultmäßig sind alle Endpoints bis auf **`shutdown`** aktiviert. Damit die Endpoints über das Web erreichbar sind, müssen sie veröffentlicht werden. Defaultmäßig sind die Endpoints **`health`** und **`info`** als veröffentlicht markiert. Der Endpoint **`info`** zeigt defaultmäßig eine leere Seite und der Endpoint **`health`** den Status UP. Informationen, die im Endpoint Info angezeigt werden, können in der Datei **`application.properties`** hinterlegt werden [4]. Es können weitere Endpoints als veröffentlicht markiert werden, indem die Konfiguration beispielsweise in der Datei **`application.properties`** angepasst wird:

```
management.endpoints.web.exposure.
include=health,info,metrics
```

Damit Graylog nun an die Metriken kommt, kann in Graylog ein REST-API-Client-Input konfiguriert werden (siehe Abbildung 4), der den Mircoservice unter der actuator-URL beispielsweise alle fünf Minuten polled.

Die aktuell verwendete IP des Microservices ist mit folgendem Befehl einsehbar:

```
docker network inspect bridge
```

Alternativ kann ein Portainer **–** ein Docker-Container zur Verwaltung von Docker **–** genutzt werden [5] [6], um die verwendeten IPs und noch viele weitere Informationen über die installierten Docker-Container zu erhalten.

Fazit

Graylog bringt eine Menge an eingebauten Features inklusive In- und Outputs mit. Neben dem generischen Ansatz, über einen REST-Client auf beispielsweise die Metriken eines Microservices zugreifen zu können, gibt es noch Plugins für herstellerspezifische Lösungen. Die Plugins können über den Graylog-Marketplace [7] nachinstalliert werden.

Ich hoffe, ich konnte Ihnen einen Einblick in eine mögliche Umsetzung eines zentralen Loggings mit Graylog vermitteln. Gerne beraten wir von der ORDIX AG Sie auch bei der Einführung von Graylog in Ihrem Unternehmen.

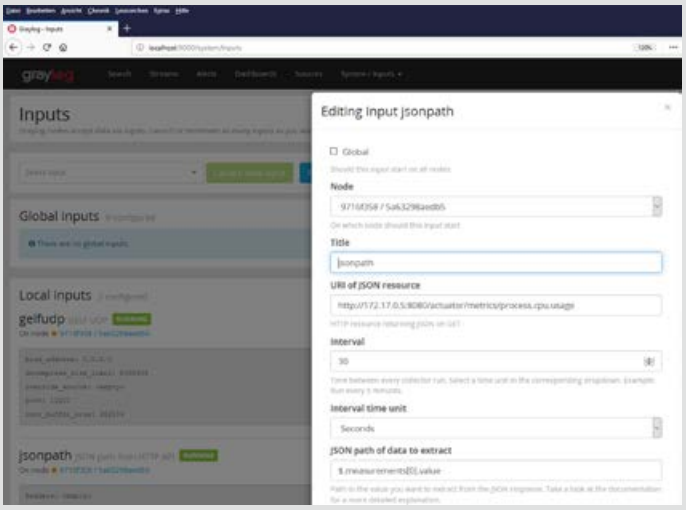


Abb. 4: Graylog JSON Input

Links

[1] Installationsdokumentation Graylog
http://docs.graylog.org/en/3.0/pages/installation.html

[2] Blog von Matthias Hryniszak zum Thema Graylog2
http://padcom13.blogspot.com/2017/03/spring-boot-application-with-graylog2.html

[3] Spring Boot Actuator: Production-ready features
https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html

[4] Customizing Spring Boot Actuator
https://www.baeldung.com/spring-boot-info-actuator-custom

[5] Docker Management
https://www.portainer.io

[6] ORDIX Blog-Beitrag von Benedikt Ahle zum Thema Docker
https://blog.ordix.de/technologien/portainer-was-kann-die-gui-fuer-docker

[7] Add-on-Marketplace für Graylog
https://marketplace.graylog.org/

[8] Gelf Dokumentation
http://docs.graylog.org/en/3.0/pages/gelf.html

Glossar

Graylog Extended Log Format (GELF)
GELF ist ein offenes von Graylog spezifiziertes Format, um strukturierte Lognachrichten zu unterstützen.

Christian Rädisch
(info@ordix.de)

AI Meets Fintech

Aufbau einer Data Science Pipeline

Maschinelles Lernen und künstliche Intelligenz werden für den geschäftlichen Erfolg immer wichtiger. Mit der Hadoop-Plattform und Frameworks wie TensorFlow oder scikit-learn, kann eine Data-Science-Umgebung sehr leicht aufgebaut werden. In stark regulierten Branchen, wie zum Beispiel der Finanzindustrie, sind vor dem produktiven Einsatz allerdings viele regulatorische und technologische Hürden zu überwinden. Dieser Artikel stellt einen erprobten Ansatz für den Aufbau einer Data Science Pipeline in einem stark regulierten Umfeld vor. Dabei werden insbesondere die unterschiedlichen Anforderungen aus dem Betrieb, der Anwendungsentwicklung und dem Data Science berücksichtigt.

Data Science Pipeline

Im Zusammenhang mit Data Science und maschinellem Lernen wird oft von der Entwicklung und dem Trainieren von Modellen geredet. Etwas vereinfacht besteht ein Modell aus einem oder mehreren Algorithmen, die als Teil eines Computerprogramms ausgeführt werden. Beim Trainieren werden die Algorithmen ausgewählt und konkrete Parameter festgelegt. Die Arbeitsweise der Data Scientisten unterscheidet sich von der klassischen Anwendungsentwicklung vor allem dadurch, dass Daten eine wesentliche Rolle spielen. In einem iterativen Prozess werden die Daten in mehreren Schritten verarbeitet, analysiert, bereinigt und zum Erstellen und Trainieren von Modellen verwendet. Die trainierten Modelle werden anschließend in der Produktion eingesetzt. Die Erkenntnisse aus der Produktion werden in der nächsten Iteration verwendet, um das Modell zu verbessern. Die Modellentwicklung kann grob in die Schritte „Data Engineering“, „Maschinelles Lernen“ und „Produktiver Betrieb“ aufgeteilt werden (siehe Abbildung 1). Typische Anwendungsfälle in der Finanzindustrie sind zum Beispiel Betrugsfallerkennung, Umsatzkategorisierung oder die Risikobewertung.

Data Engineering

Das Data Engineering besteht aus mehreren Teilschritten, in denen die Rohdaten analysiert und für das Trainieren eines Modells aufbereitet werden (siehe Abbildung 2). Am Anfang geht es darum, mit den verfügbaren

Daten zu experimentieren und sie zu verstehen. Dies wird auch als „Data Exploration“ bezeichnet. Daraus ergeben sich dann Ideen und Erkenntnisse, die bei der Erstellung eines Modells hilfreich sind. Als Nächstes werden die Daten bereinigt. Dabei werden zum Beispiel fehlerhafte Datensätze oder Ausreißer eliminiert. Bei der anschließenden Transformation werden die Daten aufbereitet, sodass weitere Analysen vereinfacht werden. Typische Transformationen sind das Normieren von Werten oder das Hinzufügen von neuen Attributen (Features) aus weiteren Datenquellen wie zum Beispiel Wetterdaten oder Ortsangaben. In der Praxis ist das Data Engineering oft der zeitaufwendigste Arbeitsschritt.

Maschinelles Lernen

Beim maschinellen Lernen wird auf Basis der Erkenntnisse aus dem Data Engineering das Modell erstellt (siehe Abbildung 3). Die Modellentwicklung erfolgt dabei oft mit Python. Als Bibliotheken kommen häufig Spark MLlib, scikit-learn oder TensorFlow zum Einsatz. Anschließend wird das Modell mit vorhandenen Daten trainiert. Dafür werden in der Regel große Datenmengen und auch sehr viel Rechenleistung benötigt. Sowohl die Daten als auch die Rechenleistung werden oft von einem Hadoop Cluster bereitgestellt. Die existierenden Daten werden in Trainingsdaten und Validierungsdaten aufgeteilt. Das Modell wird dann mit den Trainingsdaten gefüttert, sodass es selbständig lernt. Das fertig trainierte Modell wird anschließend mit Validierungsdaten überprüft, die jedoch nicht Teil der Trainingsdaten sein dürfen. Das trainierte Modell muss mit diesen neuen und unbekannten Daten zuverlässige Ergebnisse liefern.

Produktiver Betrieb

Als Nächstes wird aus dem trainierten Modell eine Anwendung, die im produktiven System installiert und betrieben wird (siehe Abbildung 4). Anwendungen können zum Beispiel regelmäßig ausgeführte Batch-Jobs, kontinuierlich laufende Streaming-Jobs oder in Realtime antwortende REST-Services sein. Zur professionellen Anwendungsentwicklung gehören automatisierte Tests und ein kontrolliertes Deployment-Verfahren. Bei den Tests geht es primär nicht um die fachliche Korrektheit des Modells, denn die wurde bereits mit den Validierungsdaten überprüft. In dieser Phase werden vor allem Stabilität und Kompatibilität des Modells getestet. Continuous-Integration-Systeme, wie zum Beispiel Jenkins oder Bamboo, kommen dafür oft zum Einsatz. Die Anwendung wird anschließend paketiert, abgenommen, in ein zentrales Repository eingeliefert, im produktiven System installiert und dann in Betrieb genommen. In der Produktion muss das Modell überwacht werden. Dafür ist es wichtig, dass die Anwendung aussagekräftige Monitoringdaten erfasst und abspeichert. Bei Auffälligkeiten oder einem fehlerhaften Verhalten muss das Modell sofort deaktiviert und schnellstmöglich durch eine korrigierte Version ersetzt werden. Bei der Inbetriebnahme aktualisierter Modelle wird oft der „Champion/Challenger“-Ansatz angewendet. Dabei werden das alte und das neue Modell parallel betrieben. Das alte Modell wird dabei als Champion betrachtet und weiterhin aktiv genutzt. Das neue Modell ist der Challenger und muss im produktiven Betrieb beweisen, dass es besser ist. Erst wenn es diesen Beweis geliefert hat, wird das alte Modell deaktiviert und das neue Modell aktiv genutzt.

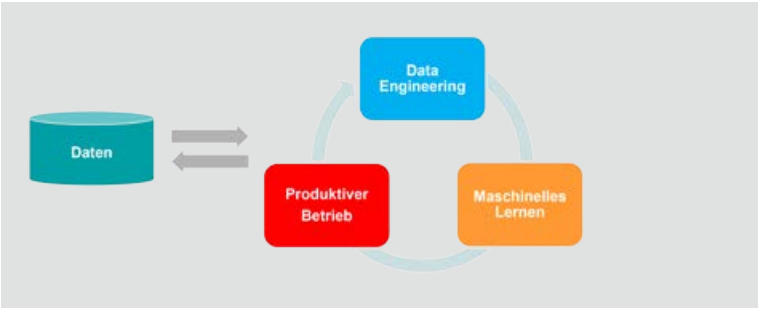


Abb. 1: Data Science Pipeline

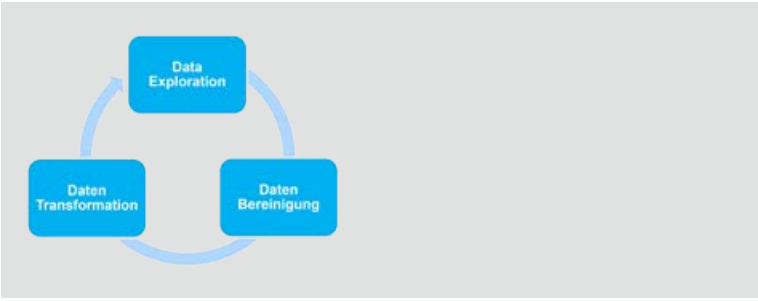


Abb. 2: Data Engineering

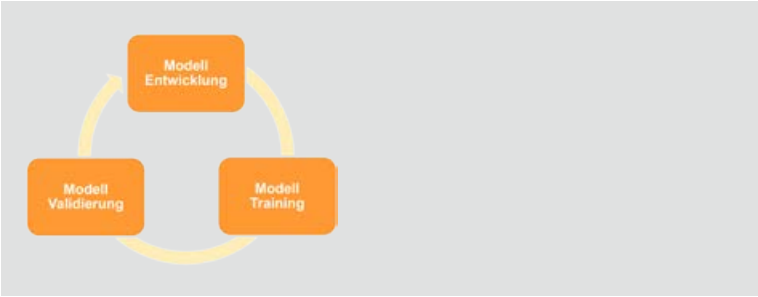


Abb. 3: Maschinelles Lernen

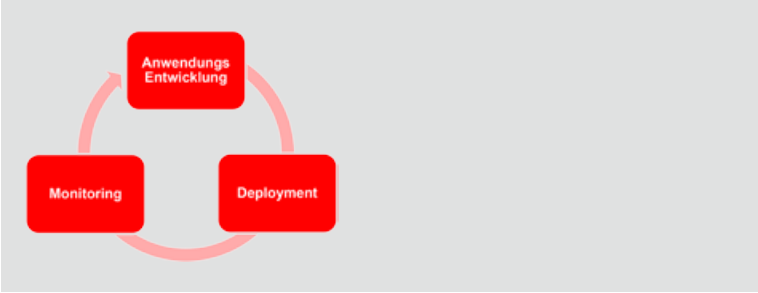


Abb. 4: Produktiver Betrieb

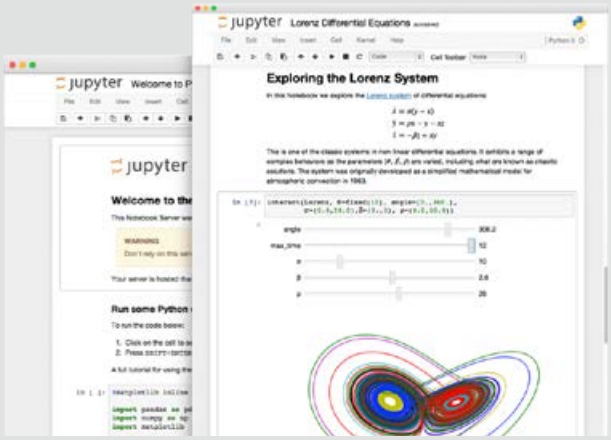


Abb. 5: Jupyter Notebook

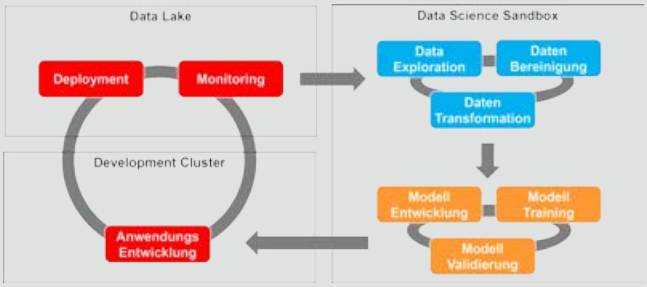


Abb. 6: Development+Cluster

Data-Science-Anforderungen

Im Idealfall möchten Data-Scientisten mit möglichst vielen produktiven Daten arbeiten. In vielen Fällen sind das personenbezogene Daten, die besonders schützenswert sind. Eine Anonymisierung oder Pseudonymisierung der Daten würde dieses Problem lösen. Allerdings sind anonymisierte Daten für die explorative Analyse und das Training der Modelle oft nutzlos. Ein Anwendungsfall ist die automatische Kategorisierung von Umsatzdaten, um ein automatisches Haushaltsbuch zu erstellen. Bei einer einfachen Anonymisierung könnte die IBAN maskiert werden. Damit ist der direkte Bezug zu einem Konto und somit zu einer Person nicht mehr möglich. Durch die Analyse des Verwendungszweckes der einzelnen Datensätze ist der Personenbezug in den meisten Fällen aber wieder herstellbar. Wird jetzt zusätzlich der Verwendungszweck maskiert, dann sind die Daten für die Analyse und Modellentwicklung aber unbrauchbar, da der Verwendungszweck für eine korrekte Kategorisierung zwingend notwendig ist.

Neben den Daten spielen auch die eingesetzten Werkzeuge eine wichtige Rolle. Zusätzlich zu Bash-Scripten und SQL-Abfragen ist insbesondere Python sehr beliebt und weit verbreitet. Für Python gibt es eine Vielzahl von analytischen und mathematischen Bibliotheken. NumPy, Pandas, Matplotlib, NLTK, scikit-learn und TensorFlow sind nur einige davon. Die Entwicklung dieser Bibliotheken schreitet schnell voran. Um die neuesten Features nutzen zu können, müssen die neuen Versionen regelmäßig und zeitnah bereitgestellt werden. Für die explorative Arbeit mit den Daten werden oft Notebook-Systeme wie Jupyter oder Zeppelin eingesetzt. Dabei handelt es sich im Wesentlichen um Webanwendungen, in denen Code-Blöcke auf einer Webseite eingegeben und direkt ausgeführt werden können. Das Ergebnis wird sofort ansprechend im Browser dargestellt und kann mit beschreibenden Texten angereichert werden. Insbesondere können die Daten innerhalb eines Notebooks sehr leicht mit Diagrammen und Tabellen visualisiert werden (siehe Abbildung 5).

Data Lake

Die beschriebenen Anforderungen können sehr gut mit einem Hadoop Data Lake umgesetzt werden. HDFS, HBase oder Kudu sind für die Speicherung großer Datenmengen sehr gut geeignet. SQL-Abfragen auf strukturierten Daten können sehr leicht mit Hive, Impala und Spark SQL ausgeführt werden. Für weniger strukturierte Daten oder komplexere Aufgaben können Spark Jobs genutzt werden. Neben der Arbeit mit Kommandozeilen-Tools wie den verschiedenen Shells für HDFS, Hive, Impala und Spark können auch Notebook-Systeme wie Jupyter und Zeppelin sehr gut in einen Hadoop-Cluster integriert werden.

Somit bietet Hadoop eine ideale Plattform für eine Data Science Pipeline. Für die Arbeit der Data-Scientisten werden sogenannte Edge Nodes bereitgestellt. Auf diesen Knoten wird neben den verschiedenen Hadoop-Kommandozeilen-Tools und Shells auch ein Notebook-System installiert und betrieben. Die Analysten melden sich auf diesen Knoten an und arbeiten von dort mit den Daten im Cluster. Dabei hat der Analyst die Wahl, seine Berechnungen, zum Beispiel mit Spark, im Cluster auszuführen oder aber die Daten in den lokalen Hauptspeicher des Edge Nodes zu laden und dort zum Beispiel mit Pandas, NumPy oder ScyPy zu arbeiten. Häufig wird auch eine Kombination genutzt. In dem Fall werden die Daten zuerst mit Spark im Cluster gelesen und vorverarbeitet und dann in ein lokales Pandas Data Frame konvertiert. Mit diesem wird dann lokal gearbeitet. Die Ergebnisse werden dann wieder mit Spark im Cluster gespeichert. Da oft direkt auf den Edge Nodes gearbeitet wird, dürfen diese bezüglich des verfügbaren Hauptspeichers und der Anzahl der CPU-Cores nicht unterdimensioniert sein.

Betriebsanforderungen

Die fertige Applikation soll am Ende in Produktion betrieben werden. Nicht selten müssen SLAs bezüglich Verfügbarkeit und Antwortzeit eingehalten werden. Das bedingt, dass die produktiven Modelle in einer kontrollierten und stabilen Umgebung laufen müssen. Ein direkter Zugriff auf das produktive System zur Ausführung von Ad-hoc-Auswertungen oder zum Trainieren von Modellen würde dem widersprechen. Weiterhin können in einer produktiven Umgebung

nicht ohne Weiteres neue Versionen von Bibliotheken installiert und ausprobiert werden. Die Gefahr von Inkompatibilitäten und ungewollten Nebeneffekten ist zu groß. Zu guter Letzt muss der Betrieb auch sicherstellen, dass die Daten vor unberechtigtem Zugriff geschützt werden. Neben einem umfassenden Rechtemanagement müssen dazu oftmals weitere Data-Governance-Anforderungen umgesetzt werden. Diese Anforderungen stehen in einem Konflikt mit den Anforderungen der agilen Arbeitsweise der Data-Scientisten.

Data Science Sandbox

Durch ein ausgefeiltes Management von Ressourcen und Rechten können viele dieser Probleme innerhalb eines einzelnen Hadoop-Clusters gelöst werden. Allerdings ist es sehr aufwendig (wenn nicht gar unmöglich), eine abgesicherte Umgebung zu schaffen, in der sich die Data-Scientisten ungestört austoben können, ohne das produktive System zu gefährden. Einfacher und sicherer ist der Aufbau einer dedizierten „Data Science Sandbox“. Dabei handelt es sich um einen eigenständigen zweiten Hadoop-Cluster, der für Data Engineering und maschinelles Lernen genutzt wird und neben dem primären Hadoop Data Lake betrieben wird. Der Data Lake dient als zuverlässige Datenquelle und als Umgebung für den Betrieb der abgenommenen Anwendungen. Data Scientisten erhalten keinen direkten Zugriff auf den Data Lake. Für ihre Arbeit verwenden sie ausschließlich die Sandbox. Data Engineering und maschinelles Lernen finden innerhalb der Sandbox statt. Durch diese Konstruktion können strenge, regulatorische Vorgaben im Data Lake umgesetzt werden, ohne die explorative Arbeit der Data-Scientisten zu beeinträchtigen.

Die Sandbox wird, wie auch der Data Lake selbst, als produktives System behandelt. Das bedeutet, dass die gleichen Maßnahmen zum Schutz der Daten vor unberechtigtem Zugriff gelten. In der Praxis müssen sich die Benutzer authentifizieren und innerhalb der Sandbox dürfen die Daten nur von autorisierten Anwendern gelesen bzw. verändert werden. Für die Sandbox gelten allerdings deutlich niedrigere Anforderungen an die Verfügbarkeit und die einzuhaltenden SLAs. Im Gegenzug bekommen die Anwender sehr viele Freiheiten. Insbesondere können innerhalb der Sandbox relativ problemlos neue Hadoop-Dienste oder Data-Science-Bibliotheken installiert und ausprobiert werden.

Closed Loop

Wie bereits beschrieben, ist es wichtig, dass die Modelle Monitoring-Daten erzeugen und speichern. Mit diesen Daten wird das Verhalten des Modells überwacht. Im Idealfall werden die Monitoring-Daten zusammen mit den neuen Eingabedaten kontinuierlich in die Sandbox übertragen. Damit schließt sich der Kreis (closed-loop). Innerhalb der Sandbox können die Daten analysiert und mit den erwarteten Ergebnissen abgeglichen werden. In weiteren Iterationen werden diese Daten dann für die Optimierung und Weiterentwicklung der Modelle verwendet.

Development-Cluster

Für den vollständigen Round-Trip fehlt noch eine Kleinigkeit. Das trainierte Modell muss von der Anwendungsentwicklung in eine robuste Applikation überführt werden. Dafür wird eine Umgebung benötigt, in der effizient entwickelt und getestet werden kann. Insbesondere in stark regulierten Branchen, wie der Finanzindustrie, bekommen Entwickler keine Rechte, um direkt mit produktiven Systemen und schützenswerten Daten zu arbeiten. Somit können die Anwendungsentwickler weder den produktiven Data Lake noch die Data Science Sandbox für ihre Arbeit nutzen. Die Lösung ist ein dritter Hadoop-Cluster: der Development-Cluster. Dieser Cluster wird für Anwendungsentwicklung, Test und Abnahme verwendet. Im Development-Cluster dürfen keine schützenswerten Daten gespeichert werden. Entwicklung und Test finden mit synthetisch generierten oder mit zuverlässig anonymisierten Daten statt. Die Installation der abgenommenen Anwendungen im produktiven Data Lake erfolgt über einen kontrollierten Deployment-Prozess. Dazu wird die fertige Software paketiert und von den Entwicklern in einem zentralen Repository, wie zum Beispiel einem Nexus-Repository, abgelegt. Das Betriebsteam verteilt und installiert dann die Software aus dem Repository im Cluster.

Fazit

Die Anforderungen aus Data Science, Betrieb und Anwendungsentwicklung sind nicht leicht zu vereinbaren. Mit dem Aufbau eines Data Lakes, einer Data Science Sandbox und eines Development-Clusters können diese Anforderungen aber elegant umgesetzt werden (siehe Abbildung 6). Dadurch bekommt das Data-Science-Team die notwendigen Freiheiten für Experimente, ohne produktive Systeme zu gefährden. Durch einen zusätzlichen Development-Cluster wird sichergestellt, dass auch die Anwendungsentwicklung effizient arbeiten kann. Der Datenschutz wird vereinfacht, da in den einzelnen Clustern immer nur die absolut notwendigen Daten gespeichert werden (Datensparsamkeit). Von Nachteil ist, dass drei Hadoop-Cluster bereitgestellt und parallel betrieben werden müssen. Abhängig von den konkreten Anforderungen können noch weitere Cluster notwendig sein. So wurden Punkte wie Backup & Disaster Recovery oder Upgrades der Hadoop-Software noch nicht berücksichtigt.

Olaf Hein
(info@ordix.de)

Bildnachweis

© pixabay.com | taddtography | Sun Tunnel Lucin Utah

Ausbildung und Karriere bei der ORDIX AG

Ready for IT?

Mit aktuell fast 40 dualen Studenten geht die ORDIX AG entschieden gegen den Fachkräftemangel vor. Jedes Jahr starten 12-15 Studenten in den vier großen Standorten der ORDIX AG ihr duales Studium zum Bachelor. Dies ist für viele Studienanfänger der Einstieg in die IT-Welt. Wie so ein Einstieg aussehen kann, zeigen wir Ihnen hier.

Ausbildung hat bei ORDIX Tradition

Bereits im Jahr 1995 wird die ORDIX AG Ausbildungsbetrieb und beginnt damit bereits 5 Jahre nach der Gründung mit der Ausbildung der eigenen Fachkräfte. Und einige der Auszubildenden der ersten Jahre arbeiten noch heute bei uns.

Der erste Auszubildende der ORDIX AG hat in dieser Ausgabe noch einen Artikel veröffentlicht. Wissen Sie, wer es ist? (Antworten gerne an redaktion@ordix.de)

Ausbildungskonzept früher/heute

Wurde früher die konventionelle Ausbildung zum Fachinformatiker angeboten, umfasst das heutige Portfolio einige Ausbildungsgänge mehr.

Und waren es vor 20 Jahren noch konventionelle Ausbildungsgänge mit IHK-Abschluss, sind es heute duale Studiengänge. Durch die Digitalisierung, die Fülle an Daten, die damit einhergeht und die IT-Sicherheit dieser Daten entstanden in den letzten Jahren ganz neue Ausbildungswege bzw. Studienfachrichtungen.

Mittlerweile arbeitet die ORDIX AG mit Partner-Hochschulen im ganzen Bundesgebiet zusammen. Diese Kooperationen ermöglichen es uns, sieben unterschiedliche Studiengänge anzubieten.

Das Studienangebot im Einzelnen

- Duales Studium zum Bachelor of Science in Angewandte Informatik (w/m/d)
- Duales Studium zum Bachelor of Science in IT-Security (w/m/d)
- Duales Studium zum Bachelor of Science in Wirtschaftsinformatik (w/m/d)
- Duales Studium zum Bachelor of Science in Informatik (w/m/d)
- Duales Studium zum Master of Science in Informatik (w/m/d)
- Duales Studium zum Master of Science in Data Science (w/m/d)
- Duales Studium der angewandten Mathematik und Informatik (w/m/d)

Studium muss nicht nur Theorie sein!

Ein Studium besteht natürlich auch aus Theorie. Aber bei einem dualen Studium bei ORDIX kommt auch noch eine Menge Praxis dazu.

Die Aufteilung der Theorie- und Praxisphasen wird von der Partnerhochschule und dem jeweiligen Studienmodell festgelegt. Dies ist von Hochschule zu Hochschule unterschiedlich.

Mit welcher Hochschule/welchen Hochschulen kooperiert die ORDIX AG?

Nordrhein-Westfalen

- FHDW Paderborn
- FHDW Bergisch Gladbach
- FH Aachen (Studienort Köln)

Hessen

- Hochschule Darmstadt
- Hochschule RheinMain
- Berufsakademie Rödermark

Bayern

- International Co-operative Studies (I.C.S.) e.V.

Weiterbildung während der Praxisphasen

Alle dualen Studenten bei der ORDIX AG erhalten zusätzlich Unterstützung durch praxisbezogene Seminare im Seminarzentrum Wiesbaden. Diese Schulungen dienen dazu, den Studenten neben notwendigem Basiswissen auch maßgeschneidertes Know-how zu vermitteln.

Nach dem Bachelor ist noch nicht Schluss

Auch für diese Variante bietet die ORDIX AG ein praxisnahes Konzept – mit einem individuellen dualen Master-Studienprogramm.

Dabei nutzt der Student das Know-how und das Wissensnetzwerk der ORDIX AG für die Abschlussarbeit. Erfahrene Datenbankspezialisten, Entwickler und Projektmanager stehen hier mit Rat und Tat zur Seite.

Und nach dem Studium ?

Die Übernahme ist das uneingeschränkte Ziel unserer Ausbildung! Dies setzen wir seit mehr als 20 Jahren um. Die unterschiedlichen Karrieremöglichkeiten bieten hierbei einen gelungenen Berufseinstieg.

DUALES STUDIUM BEI DER ORDIX AG:

- Verbindung von Theorie und Praxis für eine optimale Ausbildung
- Spannende Projekte und viele unterschiedliche Themengebiete begegnen den Studenten
- Die Studiengebühren werden übernommen und es wird ein Gehalt gezahlt
- Zusätzliche Weiterbildungen durch den Besuch fachspezifischer Kurse im Seminarzentrum Wiesbaden
- Flexible Arbeitszeiten sorgen für eine sehr gute Work-Life-Balance
- 30 Tage Urlaub
- Regelmäßig Mitarbeiter-Events

ORDIX® seminare

Seminarzentrum:
ORDIX AG
Kreuzberger Ring 13
65205 Wiesbaden
Tel.: 0611 77840-00

Online-Anmeldung, aktuelle Seminarinhalte und Termine unter:
https://seminare.ordix.de

Unser Seminarstandort ist Wiesbaden.
Die hier angegebenen Termine entsprechen dem ersten Seminartag.
Die Preise gelten pro Seminar pro Teilnehmer in Euro zzgl. ges. MwSt., Inhouse-Preise auf Anfrage.

Big Data und Data Warehouse

BIG Data				
DB-BIG-01	Big Data: Informationen neu gelebt	1 Tage	590,00 €	26.08. | 04.11.
DB-BIG-02	Big Data: Apache Hadoop Grundlagen	3 Tage	1.390,00 €	24.06. | 23.09. | 02.12.
DB-NSQL-01	Einführung in NoSQL-Datenbanken	2 Tage	1.090,00 €	27.06. | 26.09. | 05.12.
Data Warehouse				
DB-DB-03	Data Warehouse Grundlagen	3 Tage	1.290,00 €	19.08. | 05.11.

PostgreSQL

DB-PG-01	PostgreSQL Administration	5 Tage	1.990,00 €	12.08. | 21.10.

Oracle

Entwicklung				
DB-ORA-01	Oracle SQL	5 Tage	1.990,00 €	29.07. | 23.09. | 04.11.
DB-ORA-01A	Oracle SQL Workshop für Experten	3 Tage	1.490,00 €	01.07. | 07.10. | 02.12.
DB-ORA-02	Oracle Datenbankprogrammierung mit PL/SQL Grundlagen	5 Tage	1.890,00 €	03.06. | 12.08. | 18.11.
DB-ORA-34	Oracle Datenbankprogrammierung mit PL/SQL Aufbau	3 Tage	1.390,00 €	15.07. | 02.09. | 09.12.
DB-ORA-42	Oracle PL/SQL für Experten - Performance Analyse & Laufzeitopt.	3 Tage	1.390,00 €	26.08. | 28.10. | 09.12.
DB-ORA-49E	Oracle 12c Neuheiten für Entwickler	3 Tage	1.390,00 €	19.08. | 21.10. |09.12.
DB-ORA-53	Oracle Text	3 Tage	1.490,00 €	16.09. | 16.12.
DB-ORA-51	Oracle Spatial	3 Tage	1.490,00 €	22.07. | 11.11.
DB-ORA-46	Oracle APEX Anwendungsentwicklung Grundlagen	3 Tage	1.490,00 €	05.08. | 07.10.
DB-ORA-47	Oracle APEX Anwendungsentwicklung Aufbau	3 Tage	1.490,00 €	19.08. | 28.10.
Administration				
DB-ORA-03	Oracle Datenbankadministration Grundlagen	5 Tage	1.990,00 €	24.06. | 09.09. | 25.11.
DB-ORA-04	Oracle Datenbankadministration Aufbau	5 Tage	1.990,00 €	08.07. | 16.09. | 02.12.
DB-ORA-07	Oracle Tuning - Theorie und Interpretation von Reports	5 Tage	2.290,00 €	01.07. | 23.09. | 18.11.
DB-ORA-08	Oracle Grid Infrastructure und Real Application Cluster (RAC)	5 Tage	2.290,00 €	15.07. | 02.09. | 11.11.
DB-ORA-49A	Oracle 12c / Oracle 18c Neuheiten	5 Tage	2.190,00 €	03.06. | 26.08. | 14.10. | 02.12.
DB-ORA-49B	Oracle 18c Neuheiten	2 Tage	1.090,00 €	06.06. | 29.08. | 17.10. | 05.12.
DB-ORA-52W	Oracle Lizenz Workshop Webinar	1 Tag	590,00 €	Termine auf Anfrage
DB-ORA-33	Oracle Security	4 Tage	1.890,00 €	12.08. | 21.10. | 16.12.
DB-ORA-35	Oracle Cloud Control	3 Tage	1.590,00 €	24.06. | 16.09. | 09.12.
DB-ORA-55	Oracle ASM für Single Instance	3 Tage	1.490,00 €	03.06. | 26.08. | 14.10. | 16.12.
DB-ORA-56	Oracle Tenant Technologie (Multi Tenant / Single Tenant)	3 Tage	1.490,00 €	22.07. | 07.10.
DB-ORA-57	Single Sign On mit Oracle	3 Tage	1.490,00 €	08.07. | 28.10.
Backup und Recovery				
DB-ORA-32	Oracle Backup und Recovery mit RMAN	5 Tage	1.990,00 €	29.07. | 21.10
DB-ORA-31	Oracle Data Guard	4 Tage	1.890,00 €	05.08. | 04.11.
MySQL				
DB-MY-01	MySQL Administration	4 Tage	1.490,00 €	15.07. | 21.10.

IBM Datenbanksysteme

Informix				
DB-INF-01	IBM Informix SQL	5 Tage	1.790,00 €	01.07. | 07.10.
DB-INF-02	IBM Informix Administration	5 Tage	1.990,00 €	22.07. | 11.11.
DB2				
DB-DB2-01	IBM Db2 für Linux/Unix/Windows SQL Grundlagen	5 Tage	1.990,00 €	12.08. | 21.10.
DB-DB2-02	IBM Db2 für Linux/Unix/Windows Administration	5 Tage	1.990,00 €	24.06. | 02.09. | 25.11.
DB-DB2-05	IBM Db2 für Linux/Unix/Windows Monitoring und Tuning	3 Tage	1.490,00 €	08.07. | 16.09. | 09.12.
DB-DB2-06	IBM Db2 für Linux/Unix/Windows Backup und Hochverfügbarkeit mit HADR	3 Tage	1.490,00 €	15.07. | 28.10. | 02.12.

Microsoft

Entwicklung				
MS-SQL-01	Querying Data with Transact-SQL	5 Tage	2.190,00 €	05.08. | 18.11.
MS-SQL-07	Updating Your Skills to Microsoft SQL Server 2017	2 Tage	1.290,00 €	29.07. | 28.10.
Administration				
MS-SQL-02	Verwalten einer SQL Datenbankinfrastruktur	5 Tage	2.190,00 €	24.06. | 16.09. | 25.11.
MS-SQL-11	Microsoft SQL Server for Oracle DBAs	4 Tage	1.890,00 €	19.08. | 04.11.
MS-SQL-17W	Microsoft SQL Server 2017 Upgrade Webinar	1 Tag	99,00 €	auf Anfrage

Rechenzentrum

ANSIB-01	Konfigurationsmanagement mit Ansible	3 Tage	1.390,00 €	19.08. | 11.11.
E-DOCK-01	Docker DevOps Workshop	1 Tag	450,00 €	02.07. | 27.09. | 03.12.
SM-NAG-01	Systemüberwachung mit Nagios - Workshop	3 Tage	1.190,00 €	auf Anfrage

Web und Application-Server

INT-04	Apache HTTP Server Administration	3 Tage	1.290,00 €	01.07. | 16.12.
INT-07	Tomcat Konfiguration und Administration	3 Tage	1.290,00 €	05.08. | 21.10. | 09.12
INT-08	WebSphere Application Server Installation und Administration	3 Tage	1.390,00 €	auf Anfrage
INT-12	WildFly Application Server Administration	3 Tage	1.290,00 €	03.06. | 02.09. | 09.12.
DB-ORA-50	Oracle WebLogic Administration Grundlagen	3 Tage	1.390,00 €	11.06. | 02.09. | 25.11.

IT-Security

IT-SEC-01	IT-Sicherheit für Projektmanager und IT-Leiter - ein Überblick	3 Tage	1.690,00 €	03.06. | 02.09. | 18.11
SEC-06	Security Awareness für Mitarbeiter	1 Tag	990,00 €	24.10.
SEC-02	Certified Information Systems Security Professional (CISSP)	5 Tage	3.490,00 €	24.06. | 09.09. | 23.09. | 02.12.
SEC-03	Certified Information Security Manager (CISM)	3 Tage	2090,00 €	21.10. | 29.10.
SEC-04	Certified Information Systems Auditor (CISA)	4 Tage	2.290,00 €	21.05. | 07.10. | 05.11.
Begleitende Coachings zu CISSP (SEC-02C), CISM (SEC-03C), CISA (SEC-04C)			800,00 €	

Projekt- und IT-Management

Klassisches Projektmanagement				
PM-01	IT-Projektmanagement praxisorientiert	3 Tage	1.690,00 €	24.06. | 09.09. | 02.12.
PM-15	Projektmanagement für Führungskräfte - ein Überblick	2 Tage	1.290,00 €	05.08. | 04.11.
PRINCE-01	PRINCE2® Foundation	3 Tage	1.160,00 €	15.07. | 16.09. | 25.11.
PRINCE-02	PRINCE2® Practitioner	3 Tage	1.655,00 €	17.07. | 18.09. | 27.11.
PRINCE-03	PRINCE2® kompakt	5 Tage	2.560,00 €	15.07. | 25.11.
PM-06	Projekte souverän führen - Systemisches Projektmanagement	4 Tage	1.790,00 €	08.07. | 28.10.
PM-05	Projektcontrolling in der IT	2 Tage	1.290,00 €	19.08. | 16.10. | 09.12
PM-07	Krisenmanagement in Projekten - Projektkrisen vorbeugen & meistern	2 Tage	1.290,00 €	17.10.
PM-14	Anforderungsmanagement in IT-Projekten	2 Tage	1.290,00 €	03.06. | 16.09. | 18.11.
Agiles Projektmanagement				
AGIL-01	Agil führen - Neue Konzepte für Ihre Führung im agilen Umfeld	3 Tage	1.490,00 €	17.06. | 04.11.
SCRUM-01	Agiles Projektmanagement mit Scrum - Mit agilem Vorgehen mehr ...	2 Tage	1.290,00 €	23.09.
SCRUM-02	Scrum Vorbereitung zur Zertifizierung - So einfach kann es klappen	1 Tag	690,00 €	25.09.
SCRUM-04	Scrum Product Owner - Produkte erfolgreich entwickeln	3 Tage	1.490,00 €	17.09.
KB-01	KANBAN in der IT - Prozesse & Projekte mit Hilfe von Kanban optimieren	2 Tage	1.290,00 €	26.09.
PM-T-01	Testmanagement für agile und klassische Projekte	2 Tage	1.290,00 €	05.06. | 18.09. | 20.11.
PM-08	Hybrides Projektmanagement	2 Tage	1.290,00 €	14.10.
IT-Management, IT-Strategie und IT-Organisation				
MGM-03	IT-Management - Die IT nachhaltig zum Erfolg führen	3 Tage	1.690,00 €	16.12.
PM-29	Systemische Führung - Führung unter Berücksichtigung aller Aspekte	3 Tage	1.690,00 €	30.09.
K-12	Plötzlich IT-Führungskraft - Von Anfang an richtig führen	4 Tage	1.790,00 €	09.09.
MGM-07	IT-Strategie - strategische IT-Planungen	3 Tage	1.690,00 €	18.11.
MGM-02	IT-Architekturen - Ihre IT-Landschaft sinnvoll gestalten	3 Tage	1.690,00 €	14.10.
PM-10	IT-Controlling - Methoden zur Steuerung der IT	3 Tage	1.690,00 €	02.09.
MGM-04	Geschäftsprozessmanagement (BPM)	3 Tage	1.690,00 €	24.06. | 25.11.
PM-CH-01	Change Management in der IT - Veränderungen reibungslos einführen	3 Tage	1.690,00 €	01.07. | 11.11.
ITIL-01	ITIL® V3 Foundation	3 Tage	952,50 €	08.07. | 09.09. | 04.11.
ITIL-02	ITIL® V3 Practitioner	2 Tage	1.410,00 €	11.07. | 12.09. | 07.11.
ITIL-03	ITIL® V3 kompakt	5 Tage	2.375,00 €	08.07. | 09.09. | 04.11.
PM-28	IT-Organisation - Optimierung Ihrer IT-Organisation	3 Tage	1.690,00 €	21.08. | 25.11.

Kommunikation und Selbstmanagement

K-04	Konfliktmanagement - Mehr Sicherheit in unsicheren Situationen	2 Tage	1.100,00 €	25.10.
K-05	Zeit- und Selbstmanagement - Mit weniger Stress mehr erreichen	2 Tage	1.090,00 €	08.11.
K-10	Coaching von IT Mitarbeitern - Führungskraft als Coach	3 Tage	1.090,00 €	17.12.
K-20	Beratungs Know-How für IT-Experten - Kunden & Kollegen gut beraten	3 Tage	1.390,00 €	24.09.
K-30	Burn-out vorbeugen, bessere Work-Life-Balance erreichen	3 Tage	1.340,00 €	22.07. | 21.10.
K-40	Heikle Gespräche	3 Tage	630,00 €	18.07. | 04.10.

Betriebssysteme & Monitoring

Unix/Linux				
BS-01	Unix/Linux Grundlagen für Einsteiger	5 Tage	1.790,00 €	15.07. | 09.09. | 04.11.
BS-02	Linux Systemadministration	5 Tage	1.790,00 €	29.07. | 23.09. | 11.11.
BS-25	Unix Power Workshop für den Datenbank- & Applikationsbetrieb	5 Tage	1.890,00 €	12.08. | 25.11.
BS-27	Neuerungen SUSE Linux Enterprise Server 12	3 Tage	1.290,00 €	03.06. | 16.09. | 16.12.
BS-09	Linux Cluster mit Pacemaker und Corosync	3 Tage	1.490,00 €	26.08. | 18.11.
Solaris				
BS-03-11	Solaris 11 Systemadministration Grundlagen	5 Tage	1.990,00 €	01.07. | 14.10.
BS-04-11	Solaris 11 Systemadministration Aufbau	5 Tage	1.990,00 €	22.07. | 02.12.
BS-06-11	Solaris 11 für erfahrene Unix/Linux-Umsteiger	3 Tage	1.990,00 €	auf Anfrage
IBM AIX				
AIX-01	IBM AIX Systemadministration Grundlagen	5 Tage	1.990,00 €	16.09. | 09.12.
AIX-04	IBM AIX Systemadministration Power Workshop	3 Tage	1.390,00 €	08.07. | 07.10.
AIX-02	IBM AIX Installation, Backup und Recovery mit NIM	3 Tage	1.390,00 €	22.07. | 11.11.

Entwicklung

Allgemeines				
OO-01	Einführung in die objektorientierte Programmierung und UML	3 Tage	1.190,00 €	24.06. | 23.09.
E-SWA-01	Softwarearchitekturen	5 Tage	1.890,00 €	23.09. | 09.12.
Script-Sprachen				
PSHELL-01	Windows PowerShell Für Administratoren	3 Tage	1.490,00 €	01.07. | 23.09. | 09.12.
P-PERL-01	Perl Programmierung	5 Tage	1.790,00 €	15.07. | 09.09. | 16.12.
P-UNIX-01	Shell, Awk und Sed	5 Tage	1.690,00 €	08.07. | 07.10. | 02.12.
P-PYTH-01	Python Programmierung	4 Tage	1.590,00 €	08.07. | 09.09. | 25.11.
XML				
P-XML-01	XML Grundlagen	3 Tage	1.290,00 €	26.08. | 07.10.
Java				
P-JAVA-01	Java Programmierung Grundlagen	5 Tage	1.790,00 €	29.07. | 14.10.
P-JAVA-03	Java Programmierung Aufbau	5 Tage	1.790,00 €	12.08. | 11.11.
P-JEE-08	Java Performance Tuning	3 Tage	1.490,00 €	15.07. | 14.10. | 02.12.
P-JAVA-11	Java Neuheiten	2 Tage	990,00 €	06.06. | 05.09. | 07.11.
Java EE				
P-JAVA-12	Java EE Power Workshop	5 Tage	1.890,00 €	19.08. | 04.11.
J-HIB-01	Java Persistence API mit Hibernate	5 Tage	1.790,00 €	01.07. | 02.09. | 25.11.
INT-05	Java Web Services	3 Tage	1.290,00 €	29.07. | 28.10.
P-JEE-06	Spring Power Workshop	5 Tage	1.790,00 €	08.07. | 07.10. | 02.12.
MICRO-01	Microservices Workshop mit Spring Boot	5 Tage	1.790,00 €	22.07. | 21.10. | 09.12.
Web- und GUI-Entwicklung				
P-PHP-01	PHP Programmierung	5 Tage	1.790,00 €	16.09. | 18.11.
P-JEE-05	Rich Internet Application mit JSF und Primefaces	5 Tage	1.790,00 €	26.08. | 11.11.
P-JEE-05A	Webanwendungen mit JavaServer Faces (JSF)	5 Tage	1.790,00 €	09.09. | 25.11.
E-ANG-02	Webanwendungen mit Angular	3 Tage	1.590,00 €	07.08. | 20.11.
E-TYPSC-01	TypeScript Grundlagen	2 Tage	1.190,00 €	05.08. | 18.11.
Tools und Verfahren				
P-CI-01	Continuous Integration (CI) Workshop	3 Tage	1.390,00 €	03.06. | 02.09. | 04.11.

PostgreSQL- Datenbank der Zukunft? NEUE REIHE: Teil I

PostgreSQL Enterprise Features

Open-Source-Datenbanken wie PostgreSQL haben am Markt einen schweren Stand. Eine oft missverstandene Eigenart ist, dass viele der erweiterten Features, die ein kommerzielles RDBMS ausmachen, dort nicht oder nur rudimentär umgesetzt seien. Tatsächlich stellt PostgreSQL eine Vielzahl von Features bereit, die denen in kommerziellen RDBMS gleichkommen. Einige dieser Features wollen wir Ihnen in diesem Artikel aufzeigen. In den kommenden Ausgaben werden diese Eigenschaften dann mit größerer Ausführlichkeit beleuchtet.

Hochverfügbarkeit (High Availability)

In kommerziellen RDBMS ist oft eine HA-Variante implementiert, bei der neben dem regulären Datenbankserver ein zweites System parallel läuft, welches ein 1:1-Abbild des ersten darstellt. Das Ausfallsystem läuft dabei in einem Modus der ständigen Wiederherstellung von Archivdaten des Primärsystems.

Oft ist das Zweitsystem auch in einer lesenden Variante verfügbar, also für reine Abfragen ohne Datenänderung. Sollte das Primärsystem - geplant oder ungeplant - nicht verfügbar sein, so existiert eine Möglichkeit, das Zweitsystem zum Primärsystem zu machen und den produktiven Betrieb weiter laufen zu lassen. Bei Oracle ist diese Lösung als Dataguard bekannt, bei IBM Db2 und Informix als HADR bzw. HDR. Die PostgreSQL-Variante hiervon lautet Log Shipping Standby Server und der Funktionsumfang ist standardmäßig Teil der normalen PostgreSQL-Installation.

Wie bei den anderen RDBMS verwendet auch PostgreSQL das gleiche Prinzip. Basierend auf einem Backup vom Hauptsystem werden Datenänderungen über Archive auf das Zweitsystem übertragen. Ein solches Backup hat unter PostgreSQL zudem bereits den Vorteil, dass es sich mit wenigen Einstellungen zu einer lauffähigen Variante und somit zu einem Abbild des primären Datenbankservers umfunktionieren lässt. Ein weiterer Vorteil ist zudem, dass sich das zweite System ohne zusätzliche Kosten in den Lesemodus versetzen lässt, was unter kommerziellen Datenbanksystemen wie Oracle mit Zusatzkosten zu Buche schlägt.

Bei PostgreSQL gibt es hierbei zwei Varianten:

Die erste ist das Wiederherstellen der WALs auf dem Standby-System. Diese Variante ist naturgemäß asynchron, da ein WAL-Archiv erst wiederhergestellt werden kann, wenn es abgeschlossen und auf das Standby-System transferiert wurde. Das Risiko eines Datenverlustes ist zwar relativ gering, aber doch vorhanden, da bei einem Totalausfall des Primärsystems kein solch abschließender Transfer mehr erfolgen kann. Transaktionsinformationen, die im letzten WAL vorhanden waren, können daher unter Umständen verloren sein. Um dieses Risiko weitestgehend zu minimieren, kann das Verfahren so abgeändert werden, dass nach Ablauf des bestimmten Zeitintervalls automatisch ein Transfer eingeleitet wird.

Die zweite Variante nennt sich Streaming Replication. Dabei werden Transaktionen über das Streaming-Replication-Protokoll vom Primärsystem auf das Standby-System gesendet, sobald die Bestätigung zur Datenänderung (`Commit`) gesendet wird. Sollte das Primärsystem ausfallen, sind so allenfalls die Änderungen im Puffer nicht mehr übertragen worden.

Streaming Replication kann auch synchron eingestellt werden. In synchronen Fall ist das Risiko des Datenverlustes noch geringer als im asynchronen Fall: Zu einem Datenverlust kann es nur dann kommen, wenn beide Systeme gleichzeitig versagen. Aus diesem Grund sind einzelne HA-Server immer in getrennten Rechenzentren unterzubringen. Sind allerdings die Netzwerkantwortzeiten zwischen den Systemen zu groß, so kann es bei synchroner Replikation zu Performance-Einbrüchen kommen, da das Primärsystem immer auf Antworten vom Standby wartet.

Passwortverschlüsselung

Da PostgreSQL seine Benutzer intern verwaltet, müssen zugriffsberechtigte Benutzer über ein Passwort verfügen. Diese werden automatisch verschlüsselt und im Cluster gespeichert. Da der bis zur Version 9.6 verwendete Verschlüsselungsalgorithmus MD5 mittlerweile als zu unsicher gilt, wurde ab Version 10 der SCRAM-SHA-256-Algorithmus integriert.

Auch wenn keine SSL-Verschlüsselung verwendet wird, ist es möglich, zumindest die Passwörter nicht unverschlüsselt über das Netzwerk zu versenden, was mit entsprechenden Einträgen in der `pg_hba.conf` leicht zu erreichen ist.

Passwörter, die noch im MD5-Modus in neuere Versionen übernommen wurden, können etwa durch die Definition eines Ablaufdatums für den User auch in die neue Variante überführt werden.

Netzwerkverschlüsselung

PostgreSQL unterstützt nativ SSL-Verschlüsselung für Datenbankzugriffe über das Netzwerk. Hierzu kann sowohl ein CA-signiertes als auch ein selbst erstelltes Zertifikat verwendet werden. Benutzer und Clients müssen, wie bei PostgreSQL allgemein üblich, autorisiert werden, um eine verschlüsselte Verbindung zu verwenden und können diese dann benutzen.

Netzwerkverschlüsselung kann auch für eine verschlüsselte Replikationslösung verwendet werden, da für den Datentransfer über das Netzwerk ein Zugriff über einen speziellen Replikationsbenutzer benötigt wird. Dieser Benutzer muss zwar nicht zwingend vorhanden sein, ist aber aus Sicherheitsgründen empfehlenswert.

Datenverschlüsselung

PostgreSQL stellt über seine mitgelieferten Erweiterungen eine Vielzahl von Verschlüsselungsmethoden zur Verfügung, um sensible Daten vor unbefugtem Zugriff zu schützen.

Zur verschlüsselten Ablage von Passwörtern innerhalb der Datenbank (gemeint ist hier kein Datenbankpasswort für PostgreSQL, sondern z.B. Passwörter für ein externes Programm) existieren bereits vorgefertigte Hashing-Funktionen. Diese Funktionen verfügen über einige Sicherheitsmechanismen, etwa:

- unterschiedliche Verschlüsselung gleicher Passwörter (zufällige Salt-Zeichenfolge)
- verschlüsselte Passwörter können aus unterschiedlichen Algorithmen berechnet werden und trotzdem nebeneinander existieren
- Steuerung der Schnelligkeit des genannten Algorithmus' (durch Erhöhung der Iterationen), eventuelle Angreifer benötigen aus diesem Grund ebenfalls länger, um ein Passwort durch Brute-Force-Angriffe herauszufinden.

Zur verschlüsselten Ablage von Daten, die wieder unverschlüsselt ausgelesen werden müssen, stellt PostgreSQL PGP- und RAW-Verschlüsselung zur Verfügung. Das stellt sicher, dass nur Benutzer, welche über den korrekten Schlüssel verfügen, in der Lage sind, im Klartext auf die Daten zuzugreifen.

Externe Authentifizierung über LDAP

Für die Anmeldung an PostgreSQL können auch LDAP-User verwendet werden. Voraussetzung dafür ist, dass sie in der Datenbank als lokale User angelegt werden. In der Konfiguration des Datenbank-Clusters kann mit Optionen, die ähnlich einem LDAP-Connection-String aufgebaut sind, eine Authentifizierung des Users über das LDAP stattfinden.

Der Vorteil hierbei: Die Passwörter von Datenbank-Usern müssen nicht separat gepflegt werden und weichen auch nicht von denen der LDAP-User ab.

Bei dieser Zugriffsmethode wird momentan noch keine ldaps-URL zum verschlüsselten Zugriff auf das LDAP

Glossar

Relationales Datenbankmanagementsystem (RDBMS)
Ein RDBMS verwaltet eine oder mehrere Datenbanken mitsamt ihren Objekten, sorgt u.a. für die Einhaltung von Constraints (Randbedingungen) und für die Bearbeitung gleichzeitig stattfindender Datenabfragen.

(Datenbank-)Cluster
Stellt unter PostgreSQL das Äquivalent zu einer Datenbankinstanz dar. In einem Cluster sind immer mehrere Datenbanken vorhanden, die voneinander unabhängige Daten besitzen, aber dieselben Tablespaces und User verwenden können. Pro Server können mehrere Cluster parallel existieren.

Write-Ahead-Log (WAL)
Entspricht unter PostgreSQL den Transaktionsprotokollen. Datenänderungen werden in WALs protokolliert, um sie im Fehler- oder sogar Restore-Fall wiederholbar zu machen. Sie dienen in Replikationsszenarien außerdem zur Datenübertragung zwischen einzelnen Datenbank-Clustern.

(Standard-)Page
Ist die kleinste Einheit, in der – in diesem Fall – Datenbanken alle Daten einlesen und wieder speichern. Unter PostgreSQL sind diese standardmäßig 8 KB groß.

The-Oversized-Attribute-Storage-Technique (TOAST)
Ist das PostgreSQL-Konzept für die Speicherung von Daten mit überdurchschnittlicher Größe. Um diese Daten nicht in einer einzelnen Page zu speichern, kann mittels TOAST der Wert in einer für den User nicht sichtbaren Untertabelle abgelegt werden. TOAST kommt nur für Datentypen mit einer variablen Repräsentation zum Einsatz.

Quellen/Links

[Q1] PostgreSQL Wiki:
https://www.postgresql.org/docs/

Bildnachweis

unterstützt. Um dennoch eine sichere Verbindung über das Netzwerk aufzubauen, muss, wie oben beschrieben, die Netzwerkverschlüsselung über SSL verwendet werden.

Auditing

Nativ unterstützt PostgreSQL schon von vornherein das Tracing von SQL-Statements und deren Laufzeiten sowie von An- und Abmeldungen der User am Cluster.

Zudem lebt PostgreSQL sehr stark von seinen zahlreichen Erweiterungen, die zum Teil zur Installation hinzugehören. Eines davon ist die Extension pgaudit. Hiermit lassen sich auch Datenänderungen oder -zugriffe auf Tabellen überwachen. Auch ganz bestimmte User können auf diese Weise überwacht werden. Die Installation des Moduls erfolgt, wie fast immer unter PostgreSQL, pro Datenbank und wird beim Neustart des Clusters aktiv.

Komprimierung

PostgreSQL verwendet ein Konzept zu Speicherung von Daten, die nicht in eine Standard-Page passen. Dabei ist es möglich, die Daten sowohl direkt als auch komprimiert zu speichern. Das unter dem Namen TOAST bekannte Verfahren kommt nur für Datentypen mit variabler Länge zum Tragen. Dabei kann der Wert innerhalb der Tabelle als auch transparent für den User in eine externe Untertabelle ausgelagert werden.

Die Speicherung mit TOAST findet automatisch statt, sobald die Länge des zu speichernden Werts einen bestimmten Schwellwert überschreitet. Der DBA kann einstellen, ob der Wert zusätzlich komprimiert werden soll. Für die meisten Datentypen, für die TOAST angewendet werden kann, wird standardmäßig sowohl ausgelagert als auch komprimiert.

Dennis Vinueza
(info@ordix.de)

JSON in Oracle

Von einem JSON-Objekt zur relationalen Datenbank

Spricht man von Daten- und Informationsaustausch zwischen Server und Client ist häufig von JSON die Rede. Der textbasierte Datentyp JSON muss für JavaScript lesbar sein und dient nur dem Zweck des Informationsaustausches zwischen Anwendungen. Auch wenn Oracle JSON als Datentyp nicht unterstützt, gibt es jedoch die Möglichkeit, mit JSON-Objekten in einer relationalen Datenbank zu arbeiten.

Warum sollte man das tun?

Bei einem Datenaustausch ist es häufig notwendig, die ausgetauschten Daten dauerhaft in einer Datenbank zu speichern, um sie später weiter zu verarbeiten. Um das Speichern und die Weiterverarbeitung von JSON-Strings in Datenbanken zu vereinfachen, stellt Oracle seit der Version 12c einige Funktionen bereit. Diese werden in diesem Artikel vorgestellt.

Seit Oracle 12c ist es möglich, mit JSON-Strings in Oracle-Datenbanken zu arbeiten. Obwohl Oracle JSON als Datentyp nicht unterstützt, gibt es dennoch einige Funktionen, die zur Verarbeitung dieses Datentyps dienen. Vor allem die Version 18c brachte wichtige Neuerungen in Bezug auf JSON mit sich. Neben der großen Anwendungsvielfalt gibt es keinerlei Einschränkung bei der Abfrage auf eine JSON-Tabelle. Wichtig zu wissen ist, dass die JSON-Tabelle nicht atomar ist. Ob dies nun ein Fluch oder Segen ist, muss jeder selbst entscheiden.

Und wie genau geht das jetzt?

CREATE (siehe Abbildung 1)
Zunächst wird eine Tabelle benötigt, in welcher das JSON-Objekt gespeichert werden kann. Dies geschieht wie gewohnt durch den DDL-Befehl **CREATE TABLE**. Um JSON nun in dieser Tabelle speichern zu können, ist nur eine Spalte mit dem Datentyp **CLOB** nötig. Das Speichern eines JSON-Objektes in einer **CLOB**-Spalte ist erst seit der Oracle-Version 18c möglich . Vorher mussten die Datentypen VARCHAR2 oder RAW verwendet werden, welche einen begrenzten Speicherplatz von 32k Bytes besitzen. Der Vorteil an dem Datentyp **CLOB** ist der fast „unbegrenzte“ Speicherplatz.

Des Weiteren wird ein **CHECK**-Constraint verwendet, der auf diese Spalte gelegt wird. Dieser prüft durch die Funktion **IS JSON**, ob eingefügte Daten dem JSON-Standard entsprechen (möglich seit Oracle Version 12.1). So wird sichergestellt, dass nur JSON-Objekte in dieser Spalte liegen. Dieser Constraint ist Pflicht, wenn Daten aus der Tabelle selektiert werden sollen.

INSERT (siehe Abbildung 2)
Durch den DML-Befehl **INSERT INTO** kann in der zuvor erstellten Tabelle ein JSON-Objekt gespeichert werden. Bei jedem Einfügen wird geprüft, ob der eingefügte String den JSON-Standards entspricht.

Ist dies nicht der Fall, wird der Fehler:
ORA-02290: CHECK-Constraint <constraint_name> verletzt geworfen. Jetzt liegt das JSON-Objekt als ein String in der **CLOB**-Spalte der Tabelle. Aber wie gelangen diese nun an die einzelnen Attribute des JSON-Objekts in diesem Datensatz?

SELECT (siehe Abbildung 3)
Um die JSON-Daten in einer Tabelle zu selektieren, können diese mit dem **SELECT**-Befehl abgefragt werden. Wichtig hierbei ist es, der JSON-Tabelle in der **FROM**-

```
CREATE TABLE projekt(
 eigenschaft CLOB
    CONSTRAINT con_ck CHECK(eigenschaft IS JSON)
);
```

Abb. 1: JSON-Tabellen erstellen

```
INSERT INTO projekt VALUES('{"projektnr" : 1999}');
```

Abb. 2: Einfügen in eine JSON-Tabelle

```
SELECT j.eigenschaft.projektnr AS "Projektnummer"
FROM projekt j;
```

Abb. 3: Selektieren aus JSON-Tabelle

Abb 4: JSON-Data in einer Tabelle

```
SELECT JSON_QUERY
(eigenschaft,'$.projektnr' WITH CONDITIONAL WRAP-
PER)
FROM projekt;
```

Abb. 5: Path-Expression

Klausel einen Alias zu geben. Das ist die vorgegebene Syntax, welche eingehalten werden muss, da sonst die Abfrage nicht funktioniert. Die Punkt-Notation, welche in Version 18c neu dazukam, hilft bei der Navigation durch ein JSON-Objekt. Hierbei wird mit dem Punkt von einer Ebene auf die darunterliegende Ebene gesprungen (Tabelle➔Tabellenspalte➔JSON-Attribut).

In der Version 12c musste die Path-Expression verwendet werden, die ebenfalls durch ein JSON-Objekt navigiert (Abbildung 5). Dies ist auch in 18c noch möglich. Da aber die Punktnotation in der Simplizität der Path-Expression deutlich überlegen ist, wird seit 18c vorwiegend mit der Punktnotation gearbeitet.

Ein JSON-Objekt besteht aus mehreren Key-Value-Paaren. Durch die **SELECT**-Abfrage (Abbildung 3) wird der JSON-String wie folgt aufgeteilt: Zuerst wird für jeden Key eine Spalte angelegt. Gleichnamige Key's nutzen dieselbe Spalte. Für jedes eingefügte JSON-Objekt wird beim Selektieren eine neue Zeile angelegt. Danach werden die Values der JSON-Objekte in die dazugehörige Spalte eingefügt. Besitzt ein JSON-Objekt nicht alle „Spalten-Keys" bleibt das Feld leer (null) (siehe Abbildung 4).

Indizierung

Wie in fast allen Bereichen der Informatik spielt die Performance eine wichtige Rolle. So auch hier. Durch das Anlegen eines Indexes auf eine Spalte wird die Zugriffszeit auf die Spalte deutlich verkürzt. Kurz: Indizes beschleunigen eine Abfrage. Oracle bietet die Möglichkeit, JSON-Daten mittels dem **JSON Search Index**, welcher seit 12.1 verfügbar ist, zu indizieren.

```
CREATE SEARCH INDEX index01 ON
projekt(eigenschaft) FOR JSON;
```

Durch diesen Befehl ist es möglich, einen **JSON Search Index** auf einer JSON-Spalte anzulegen. Bei einem Test mit einer Datenmenge von 5 Millionen Datensätzen benötigt eine Abfrage ohne Index im Durchschnitt 109,66 Sekunden. Nach dem Anlegen des Indexes und der gleichen Abfrage hat sich die Laufzeit im Durchschnitt auf 0,055 Sekunden reduziert.

Auch wenn der **JSON Search Index** die Abfragezeit drastisch verkürzt, ist es ratsam den Oracle-Text-Index zu verwenden. Dieser erzielte bei dem gleichen Test eine durchschnittliche Laufzeit von 0,009 Sekunden und ist somit sechs Mal schneller als der **JSON Search Index**. Mit folgendem Befehl kann ein Oracle-Text-Index auf eine Spalte angelegt werden:

```
CREATE INDEX index02 ON projekt(eigenschaft)
INDEXTYPE IS CTXSYS.CONTEXT;
```

Diese Werte sind von vielen verschiedenen Faktoren abhängig und können entsprechend variieren. Trotzdem kann man daran gut erkennen, welchen Einfluss ein Index auf einer Spalte bei einer Abfrage hat.

Fazit

Abschließend das wichtigste noch einmal zusammengefasst:

CREATE
Beim Erstellen einer JSON-Tabelle wird nur eine Spalte mit dem Datentyp **CLOB** benötigt. Der **CHECK**-Constraint, welcher prüft, ob eingefügte Daten dem JSON-Standard entsprechen, ist Pflicht.

INSERT
Beim Einfügen muss darauf geachtet werden, dass das eingefügte Objekt ein gültiges JSON-Objekt ist. Ansonsten wird das Objekt nicht eingefügt und der **CHECK-CONSTRAINT** wirft einen Fehler.

SELECT
Beim Selektieren aus der JSON-Tabelle muss der JSON-Tabelle ein Alias gegeben werden. Über diesen Alias können dann Daten über die Punkt-Notation oder die Path-Expression selektiert werden.

Auch wenn Oracle den Datentyp JSON nicht unterstützt, ist es dennoch möglich, mit altbekannten Mitteln ein JSON-Objekt in einer Tabelle zu speichern. Auch das Abfragen der Daten aus der Tabelle ist, abgesehen von einigen kleinen Änderungen, ein alter Hut.

Daniel Dyck
(info@ordix.de)

Module statt Microservices

Auch mit klassischer Java-EE-Entwicklung ist agile Entwicklung möglich

Agilität ist der Treiber der digitalen Transformation. Nach dem Vorbild von Netflix, Google und Co. arbeiten kleine Teams an autonomen Komponenten, die zeitnah Kundenanforderungen umsetzen. Microservices erblicken das Licht der Welt. Unglaublich sind die Produktivität und Großzügigkeit, mit der Infrastruktur-Lösungen für diesen Mikrokosmos erschaffen und oftmals kostenfrei zur Verfügung gestellt werden. Unglaublich sind aber auch die Aufwände, die ein solcher Kosmos in der Entwicklung und im Betrieb mit sich bringt.

Da liegt der Wunsch nach effizienteren Lösungen nahe. Effizienz war eine Vorgabe der Vergangenheit, einer Zeit, in der sogenannte Monolithen in einem weitgehend industrialisierten Entwicklungsprozess erstellt und administriert wurden. Agilität und Effizienz schließen sich nicht aus - Microservices und Monolithen sind kein Widerspruch: Java Module lassen sich unabhängig entwickeln, in eine Anwendung integrieren und im Java-Application-Server ausführen.

Monolith versus Microservice

Der Rahmen dieses Artikels ist sicher zu eng, um eine umfassende Beschreibung des Architektur-Konzepts zu Microservices zu geben. Stattdessen sei auf die Definition in [1] verwiesen, um nur einige Aspekte von Microservices herauszugreifen, die für diesen Kontext sinnvoll erscheinen.

Oftmals wird in Vorträgen und Artikeln (z.B. im Artikel von James Lewis und Martin Fowler [1]) die Diskussion zur Microservice-Architektur damit eingeleitet, die Eigenschaften von Monolithen zu kritisieren. Ein Beispiel für einen Monolithen ist in Abbildung 1 dargestellt.

Die Kritik bezieht sich darauf, dass der innere Aufbau der Monolithen aus Komponenten besteht, die in vielfältiger Weise miteinander verbunden sind. Dies führe zu einer Komplexität, die nicht einfach beherrschbar sei. Diese Komplexität ist in Abbildung 1 nicht zu sehen: einerseits ist die Anzahl der Komponenten aus Gründen der Darstellung beschränkt. Andererseits sind die Komponenten in einer typischen Schichtenarchitektur aufgebaut. Komponenten haben nur Abhängigkeiten zu Komponenten derselben Schicht oder aber zu Komponenten der darunterliegenden Schicht. Die Reduktion der Abhängigkeiten einzelner Komponenten ist ein zentraler Treiber für diese Architektur. Offenbar ist es in einer solchen Technologie theoretisch möglich, ein System zu planen, das eine überschaubare Komplexität aufweist.

In der Praxis ist Software einem steten Wandel unterworfen: in der Entwicklungs-Phase wird sie kontinuierlich erweitert. In der Test-Phase beheben Entwickler Fehler. In der Wartungsphase verändert sich die Software aufgrund von Fehlern und Erweiterungen. Wahrscheinlich gilt der zweite Hauptsatz der Thermodynamik auch für ein Software-System: wird ein System sich selbst überlassen, so strebt es den Zustand maximaler Unordnung an - im Laufe der Zeit ist mit einem Zuwachs an Komplexität zu rechnen.

Die Komplexität (Umfang + Abhängigkeiten) des Monolithen behindert eine agile Weiterentwicklung der Software. Um die Komplexität in den Griff zu bekommen, werden Konzepte für die fachliche Logik und die Struktur des Systems geschrieben und stellen die Grundlage für Veränderungen der Software dar. Planlose Veränderungen im Monolithen werden mit der Höchststrafe geahndet: Wartungsaufwände wachsen dramatisch an, fachliche Konsistenz geht verloren.

Dieser Nachteil von Monolithen ist in einer hochveränderlichen Welt ein wesentlicher Grund für eine Anpassung

der Software-Architektur. Mit Microservices wird die Komplexität reduziert. Ein Microservice ist klein und unabhängig. Unter diesen Voraussetzungen ist die Komplexität gering. Agile Vorgehensmodelle lassen sich für Microservices einsetzen, um die Software rasch an neue Anforderungen anzupassen.

Die Abbildung 2 zeigt die Anordnung von zwei Microservices (A, B). Beide Microservices verfügen über ihre eigene Datenbank. Neben den Microservices ist für die Integration in den Betrieb etwas Infrastruktur erforderlich. Das API-Gateway stellt die Fassade nach außen dar: Alle Anfragen an Microservices seiner Domäne richten sich an das API-Gateway und werden von diesem an die

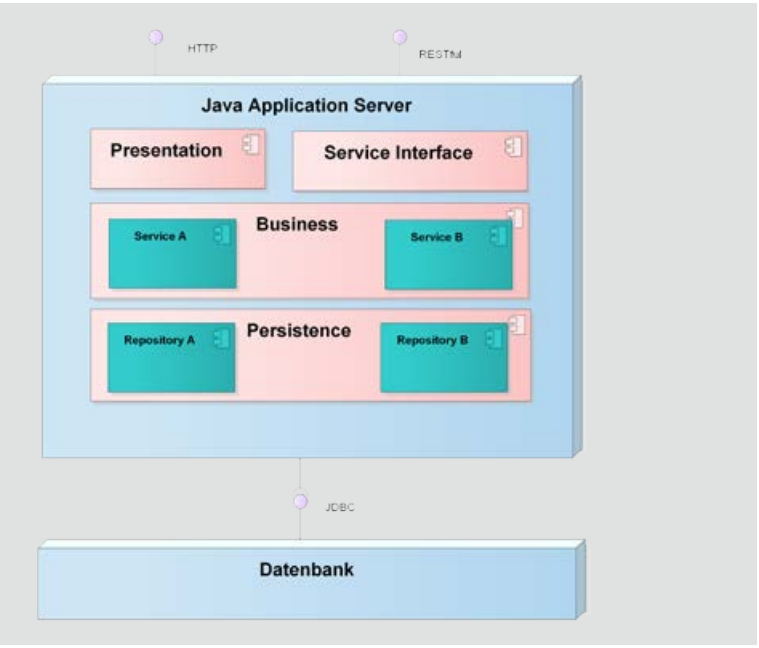


Abb. 1: Monolithische Java-EE-Architektur

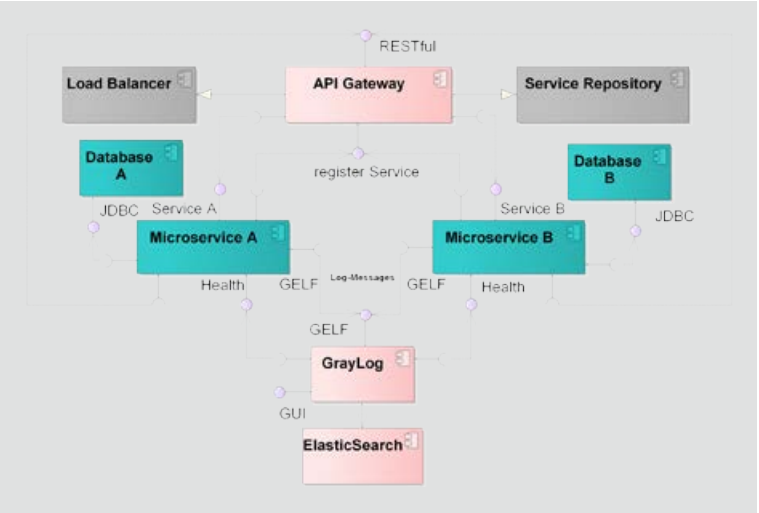


Abb. 2: Beispiel einer Microservice-Architektur

Microservices weitergeleitet. Das ist für den Service-Client praktisch - er muss nur die Verbindung zum API-Gateway berücksichtigen.

Darüber hinaus sind redundante Instanzen der Microservices zum Zwecke der Skalierung oder Hochverfügbarkeit nach außen unsichtbar. Das API-Gateway ist selbstkonfigurierend: Instanzen der Microservices registrieren sich beim API-Gateway. So hat das Gateway einen Überblick über alle Microservices seiner Domäne. Die zweite Infrastruktur-Komponente Graylog im unteren Teil von Abbildung 2 übernimmt Aufgaben des zentralen Loggings und Monitorings. Eine zentrale Instanz zur Analyse von Log-Dateien ist bei einer größeren Anzahl von Microservice-Instanzen erforderlich, um die Nachvollziehbarkeit von fachlichen oder technischen Problemen zu ermöglichen. Im Sinne der organisatorischen Autonomie scheint auch ein lokales Monitoring für das Microservice-Cluster sinnvoll zu sein.

Den ersten Eindruck, dass für den Betrieb des Microservices unterstützende Services benötigt werden, vermittelt die Abbildung 2. Neben der fachlichen Logik berücksichtigen die Entwickler die Kommunikation mit diesen Services. Das ist nur sinnvoll möglich, indem spezielle Client-Bibliotheken eingesetzt werden. Neben derartigen technischen Randbedingungen wird die Autonomie von Microservices nicht selten auch durch fachliche Abhängigkeiten eingeschränkt: ein Microservice greift auf das Service-Angebot eines anderen Microservices zu. Ziel des fachlichen Zuschnitts von Microservices muss es sein, die Autonomie weitgehend zu erhalten. Mit jeder fachlichen Abhängigkeit verringert sich die Fähigkeit, eine schnelle Anpassung durchzuführen.

Agilität stellt auch Anforderungen an den Build-Prozess und die Inbetriebnahme. Einerseits soll schnell reagiert werden: anstelle der drei oder vier Release-Termine im Jahr wird ein Rollout nach Bedarf angestrebt. Andererseits wächst die Anzahl der Software-Lösungen naturgemäß: Viele Microservices werden einen Monolithen ablösen. Automatisierung von Build und Delivery sind erstrebenswert. Auch hierfür kann das Entwicklungs-Team auf technische Unterstützung zurückgreifen - brand-neu und oftmals ohne Lizenzkosten.

In der Summe liegt der Schluss nahe, dass Agilität durch Microservices mit einem umfangreichen Technologie-Stack und erheblichen Aufwänden erkauft wird.

Autonome Services im Monolithen

Die Unabhängigkeit von Microservices ist kein Ergebnis einer Technologie. Sie ist das Ergebnis einer Zerlegung: der Architekt zerlegt ein IT-System in möglichst unabhängige Komponenten.

Die Kapselung von Microservices wird unterstützt, indem eine Komponente mit einem Stück lauffähiger Software gleichzusetzen ist. Abhängigkeiten zwischen Microservices entstehen nur durch die Verwendung von Services über

Netzwerkschnittstellen. Alternativ bietet das Java Platform Module System (aka Jigsaw) die Möglichkeit, Module zu kapseln obwohl sie zusammen mit anderen Modulen in derselben Java Virtual Machine ausgeführt werden. Es ist denkbar, dass ein Modul die fachliche Aufgabe eines Microservices übernimmt. Module lassen sich separat entwickeln und werden erst im Build-Prozess zu einer Anwendung zusammengebaut. So entsteht eine monolithische Anwendung, beispielsweise eine Java-EE-Anwendung, die aus weitgehend autonomen Modulen gebildet wird.

Sicher ist auch diese technische Option keine Universal-Lösung für alle IT-Anforderungen dieser Welt. Der Vorteil liegt in einer erheblichen Reduktion der administrativen Aufwände, einer Verringerung des Technologie-Stacks und gegebenenfalls die Nutzung bestehender Strukturen in Unternehmen mit Java-Enterprise-Erfahrung.

Netzwerk-Kommunikationspfade zwischen Microservices entfallen und reduzieren damit nicht nur die Netzwerklast und verbessern die Ausführungsgeschwindigkeit, sondern verringern auch Aufwände der Konfiguration, des Monitorings, der Autorisierung oder des Transaktionsmonitorings.

Einsatz von JPMS in Java-EE-Monolithen

Die Modularisierung durch das Java Platform Module System (JPMS) bringt Neuerungen in vielen Bereichen wie Entwicklung, Kompilierung und Laufzeitumgebung mit sich. In den Artikeln aus der Reihe „Neuheiten Java 9“ der letzten Ausgaben [2, 3] haben wir bereits ausführlich über die Neuerungen berichtet: welche Aspekte von JPMS lassen sich auf die Java-EE-Umgebung übertragen? Aktuelle Applikations-Server (wie beispielsweise der in unserem Beispiel verwendete WildFly 15) können zwar durch JPMS modularisierte Anwendungen ausführen, jedoch wird hier noch immer auf den Classpath gesetzt, statt den neu eingeführten Modulepath zu verwenden.

Dennoch ist JPMS hilfreich für die Entwicklung von Java-Enterprise-Anwendungen. Hier bietet JPMS dem Entwickler die Möglichkeit, abgeschlossene Module zu erstellen, die durch die starke Kapselung nur über klar definierte Schnittstellen angesprochen werden. Wird in der Entwicklungsumgebung der Modulpfad zum Einsatz gebracht, bekommt der Entwickler einen Kompiler-Fehler, sobald er versucht auf nicht-exportierte Klassen zuzugreifen.

Beispiel-Architektur

Als Beispiel-Projekt [4] wird ein RESTful-WebService mit Datenbankanbindung für einen Bookstore umgesetzt. Abbildung 3 zeigt die dabei verwendete Layer-Architektur unter Verwendung von JPMS. Mit JPMS werden aus den Komponenten Java-Module mit wohl-definierten Schnittstellen. Das Projekt ist in vier Module aufgeteilt.

- **Repository:** Dieses Modul kapselt den Zugriff auf die Persistenz, wie beispielsweise auf eine Datenbank. Für die Implementierung der Schnittstelle bietet sich das DAO-Entwurfsmuster an.
- **Service:** Das Service-Modul beinhaltet die Implementierung der Business-Logik. Zur Umsetzung greift das Service-Modul auf das Persistenz-Modul zu.
- **Fassade:** Die Fassade bildet die Schnittstelle zwischen dem Interface und der Geschäftslogik. Sie koordiniert die Abwicklung der fachlichen Logik
- **Interface (RESTful):** Dieses Modul enthält die Implementierung einer RESTful-Schnittstelle. Diese Netzwerk-Schnittstelle stellt die Services über das Netzwerk zur Verfügung.

Auch wenn die Kommunikation zwischen den Modulen nicht über das Netzwerk stattfindet, sondern über eine API, ist die Verwendung des Interface-Symbols gerechtfertigt: Die Komponenten kommunizieren über wohl-definierte Schnittstellen mit wohldefinierten Kommunikations-

Abb. 3: Beispiel-Architektur des Java-EE-Monolithen

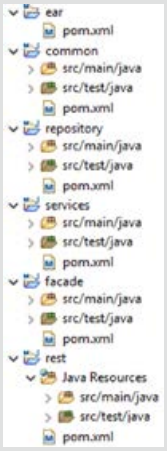


Abb. 4: Verzeichnisstruktur der Maven-Projekte

```
public class BookServiceBeanTest {
   …
   private int index;
   private HashMap<Long, Book> db;
   BookServiceBean service;
   BookDao dao;
   @BeforeEach
   public void before() {
      service = new BookServiceBean();
      dao = mock(BookDao.class);
      service.dao = dao;
      db = new HashMap<>();
      index = 0;
      when(dao.save(any(Book.class))).then(i -> {
         Book c = i.getArgument(0);
         c.setId(Long.valueOf(index++));
         db.put(c.getId(), c);
         return c;
      });
      when(dao.getByIsbn(anyString())).then(i -> {
         String id = i.getArgument(0);
         for (Book b : db.values()) {
            if (b.getIsbn().equals(id))
               return b;
         }
         throw new NoResultDaoException();
      });
…
   }
   @Test
   public void testSimpleAddAndGetBookByIsbn() {
      Book book = createBook(isbn, title);
book = service.addBook(book);
      book = service.getBook(book.getIsbn());
      assertEquals(isbn, book.getIsbn());
      assertEquals(title, book.getTitle());
      verify(dao).save(any(Book.class));
      verify(dao).getByIsbn(anyString());
   }
…
}
```

Abb. 5: Ausschnitt aus einer JUnit-Testklasse

pfaden. Die API-Schnittstellen definieren einen Vertrag zwischen den Modulen, der die zu unterstützenden Methoden und Objekte festlegt.

Build-Prozess-Beispiel

Als Build-Management-Tool setzen wir in unserem Beispiel auf Maven. Maven bietet die Möglichkeit, unsere Module unabhängig voneinander in unterschiedlichen Projekten zu entwickeln und unsere Enterprise-Anwendung automatisiert zu erzeugen.

Mit dem Kommando **`mvn archetype:generate`** erzeugt Maven vorgefertigte Musterprojekte für die jeweiligen Module. Da es sich bei den Modulen bis auf das REST-Modul um einfache Java Projekte handelt, bei denen am Ende eine JAR-Datei erzeugt wird, ist der Archetyp **`maven-archetype-simple`** geeignet. Bei dem REST-Modul handelt es sich um eine Web-Applikation, die mit dem Archetyp **`maven-archetype-webapp`** erstellt wird.

Zusätzlich zu den Modulen auf der linken Seite der Abbildung 3 werden noch zwei weitere Projekte benötigt. Dabei handelt es sich zum einen um ein weiteres einfaches Java-Projekt, das **`common`**-Projekt, um die gemeinsamen Entitäten und Schnittstellen mit allen Projekten zu teilen und zum anderen ein **`EAR`**-Projekt, um zum Schluss alle Module in einer einzigen ausführbaren Enterprise-Applikation zusammenzufügen.

Für das **`EAR`**-Projekt wird kein für uns passender Archetyp mitgeliefert. Doch das Erzeugen eines solchen Projekts ist relativ simpel. Hier reicht es aus, das Projektverzeichnis und die Konfigurationsdatei **`POM.xml`** manuell zu erzeugen. Die Abbildung 4 zeigt, wie die Verzeichnisstruktur nach der Erstellung der einzelnen Projekte aussieht. Die vollständige Konfiguration der jeweiligen **`POM`**-Dateien würde hier deutlich den Rahmen sprengen und kann online im GitHub [4] nachgeschlagen werden.

Um die komplette Enterprise-Applikation aus unabhängigen Modulen zu erzeugen, müssen zuerst die einzelnen Module mit **`mvn install`** in das lokale Repositoy installiert werden. In einer professionellen Umgebung wird jedes Team sein Modul in das zentrale Repository übertragen.

Anschließend fügt das Kommando **`mvn package`** des **`EAR`**-Projekts alle Module zu einem Enterprise Application Archive zusammen.

So lässt sich jedes Modul in einem Team entwickeln. Die Abhängigkeit zu den übrigen Projekten reduziert sich auf die angebotenen Schnittstellen und die verwendeten Data-Transfer-Objekte, die beim Aufruf der Schnittstellenmethoden verwendet werden.

Beispiel-Implementierung

Nachdem die grundlegende Projektstruktur der Module steht, kann mit der eigentlichen Implementierung der einzelnen Module begonnen werden. Hier bieten die durch die Aufteilung entstehenden kleinen Komponenten die Möglichkeit, diese mit Test Driven Development agil zu entwickeln.

Beim Schreiben der Unit-Tests wird nun durch JPMS sichergestellt, dass nur Klassen des eigenen Moduls verwendet werden können. Alle benötigten Schnittstellen der anderen Module können durch ein Mocking-Framework (wir verwenden hier Mockito) ersetzt werden.

Diese komplette Isolation der Module ist genau das, was wir durch die Verwendung von JPMS erreichen wollten. Der Ausschnitt eines Unit-Tests aus dem Beispiel-Projekt ist in Abbildung 5 zu sehen.

Um allen Modulen Zugang zu den gemeinsamen Entitäten und Schnittstellen zu ermöglichen, wird das bereits erwähnte **`common`**-Projekt verwendet. Dies verhindert die

sonst entstehende Abhängigkeitsverkettung der Module und würde der unabhängigen Entwicklung der einzelnen Projekte entgegenwirken.

Um die Auflösung der Schnittstellen zu den gehörenden Implementierungen kümmert sich dann später der Applikation-Server via Dependency Injection.

Moduldeskriptor

Die zu veröffentlichenden als auch die benötigten Pakete eines Moduls werden bei JPMS durch einen Moduldeskriptor beschreiben. Der Moduldeskriptor ist eine einzelne Klasse im Source-Verzeichnis (bei den meisten Maven-Projekten **`../src/main/java`**) mit der Bezeichnung module-info.java. Diese Klasse wird allen Modulen hinzugefügt.

Der Moduldeskriptor beschreibt mit **`exports`** alle Pakete, deren mit **`public`** deklarierte Klassen öffentlich zugänglich gemacht werden sollen und mit **`requires`** alle Abhängigkeiten zu anderen Modulen. Beispiele für einen solchen Moduldeskriptor zeigen die Abbildungen 6 und 7.

Wie in diesem Beispiel gezeigt, können des Weiteren mit **`exports to`** auch direkt Module definiert werden, die Zugriff auf das jeweilige Paket haben sollen. Dadurch wird wiederum allen anderen Modulen der Zugriff auf das jeweilige Paket untersagt.

Da auf die Implementierungen der Module aus Abbildung 3 nur über die im **`common`**-Projekt enthaltenen Schnittstellen zugegriffen werden soll, enthält ihr Moduldeskriptor keine exports (siehe Abbildung 7). Dafür ist hier das Weitergeben der Abhängigkeit mit dem Zusatz **`transitive`** zu sehen. Dies ist nötig, falls die Schnittstellen mit den Entitäten arbeiten und somit vorausgesetzt wird, dass beide Module dieselbe Abhängigkeit zu den Entitäten besitzen. In Abbildung 8 werden die Abhängigkeiten in einem Modul-Graph dargestellt.

Die Module in Abbildung 8 sind hinsichtlich ihrer (erwarteten) Stabilität sortiert. Unten befinden sich die stabilen Module, oben die Module, die eine hohe Änderungs-Wahrscheinlichkeit aufweisen. Es ist auffällig, dass viele Module von dem Modul **`bookshop-common`** abhängen. Daher sollte die Änderungswahrscheinlichkeit dieses Moduls gering sein.

Nun befinden sich in diesem Modul einerseits die Domain-Objekte, andererseits alle Klassen, die die Schnittstellen der Module beschreiben. Zur Verbesserung der Stabilität ist es ratsam, stattdessen ein Modul mit Domain-Objekten und für jede Schnittstelle eines Moduls ein separates API-Modul vorzusehen.

Universell gilt aber: die Stabilität eines Moduls ergibt sich nicht durch die Anordnung in einem Modul-Graph oder aus der Anzahl der abhängigen Module. Die Stabilität ist das Ergebnis einer stabilen Anforderungsumgebung und eines analytisch gut durchgeplanten Designs.

Als letzter Punkt zu den Moduldeskriptoren sollte noch erwähnt sein, dass JPMS standardmäßig alle Zugriffe mit Reflection auf nicht öffentliche Klassen unterbindet. Da jedoch viele Bibliotheken (wie JUnit oder auch Mockito) Reflection benutzen, sollten alle Module mit **open** deklariert werden, um dies zu erlauben.

Beispiel-Laufzeitumgebung

Zum Ausführen des Beispiel-Projekts wurde OpenJDK 11 mit einem Wildfly-15-Server verwendet.

```
open module bookshop.common {
   exports de.ordix.java.bookshop.model;
   exports de.ordix.java.bookshop.exceptions;
   exports de.ordix.java.bookshop.dao.interfaces to book-
shop.repository, bookshop.services;
   exports de.ordix.java.bookshop.services.interfaces to
bookshop.services, bookshop.facade;
   exports de.ordix.java.bookshop.facade.interfaces to
bookshop.facade, bookshop.rest;
   requires java.xml.bind;
   requires java.persistence;
   requires beta.jboss.ejb.api_3_2;
}
```

Abb. 6: Moduldeskriptor von bookshop.common

```
open module bookshop.repository {
   requires beta.jboss.ejb.api_3_2;
   requires transitive bookshop.common;
   requires java.persistence;
   requires java.annotation;
}
```

Abb. 7: Moduldeskriptor von bookshop.repository

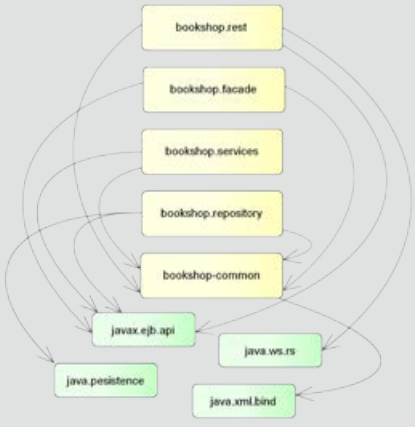


Abb. 7: Moduldeskriptor von bookshop.repository

Abb. 9: Response des GET-Requests in einem Web-Browser

Glossar

Java Platform Module System (JPMS)
Das Java Platform Module System (JPMS) ermöglicht Entwicklern, Java-Anwendungen auf Artefaktebene zu modularisieren

Links

[1] Lewis, James und Fowler, Martin:
Microservices. A definition of this new architectural term
https://martinfowler.com/articles/microservices.html

[2] ORDIX® news, 2/2017
Neuheiten Java 9 (Teil I). Java 9 - Was lange währt".

[3] ORDIX® news, 2/2018
Neuheiten Java 9 (Teil II). Java 9 macht schlank, kann aber noch mehr?".

[4] GitCoin – Beispielprojekt:
https://www.ordix.de/gitcoin/sdb/microservices-im-monolithen

[5] Reinhold, Mark: The State of the Module System:
http://openjdk.java.net/projects/jigsaw/spec/sotms/

David Sedlbauer
(info@ordix.de)

Abbildung 9 zeigt die Rückgabe auf den http-Aufruf (GET-Request **`/api/books`**) durch einen Browser. Zurückgegeben wird eine Liste aller registrierten Bücher.

Module – nicht Microservices

Auch wenn Module sich, genauso wie Microservices, in separaten Projekten von unterschiedlichen Teams entwickeln lassen, muss auf die Unterschiede dieser beiden Architektur-Ansätze eingegangen werden. Während Module über eine Programmier-Schnittstelle (API) miteinander verbunden sind, ist die Kopplung von Microservices durch eine Netzwerkschnittstelle sehr viel unabhängiger. Microservices lassen sich in unterschiedlichen Programmiersprachen oder Technologien entwickeln. Die Repräsentation der Schnittstellendaten kann sich auf Client- und Server-Seite voneinander unterscheiden.

Der Prototyp eines Microservices ist vollständig autonom: er realisiert alle Schichten der Anwendung: von der Serviceschnittstelle bis hin zur Datenbank. Ein derartiger Schnitt ist bei der Verwendung von Modulen prinzipiell nicht unmöglich, aber nicht naheliegend. Der in diesem Artikel skizzierte Ansatz basiert auf einer gewissen Beständigkeit der Domain-Objekte, auf Persistenz- und Service-Layer und natürlich nicht zuletzt auf dem Schema in der Datenbank. Agilität sollte sich in der Modul-Architektur nur in den oberen Schichten von Abbildung 8 niederschlagen.

Das Ergebnis der Modul-Architektur ist zweifelsfrei ein Monolith. Die Logik vieler Microservices sind in einer Deployment-Unit zusammengefasst. Das kann organisatorische Vorteile mit sich bringen, weil dadurch der Administrationsaufwand verhältnismäßig gering ist. Andererseits wird die Flexibilität von Microservices nicht erreicht: die individuelle Skalierung einzelner Services, die Spontanität eines Updates oder Service-Austauschs oder die Resilienz (Widerstandsfähigkeit) gegen temporäre Service-Ausfälle.

Fazit

Ab Java 9 lassen sich in Java Module erzeugen, die nur über wohldefinierte Schnittstellen angesprochen werden können. Damit liefert Java eine Hilfestellung, eine monolithische Anwendung aus weitgehend unabhängigen Komponenten zu erzeugen. Unerwünschte Abhängigkeiten zwischen diesen Komponenten werden damit unterbunden. So lässt sich ein wesentlicher Vorteil von Microservices - nämlich die Erstellung weitgehend autonomer kleiner Komponenten - in einer monolithischen Architektur umsetzen. Infrastrukturaufwände für Microservice-Landschaften werden überflüssig. Unternehmen können auf bewährte Serviceeinheiten oder Cloud-Systeme zurückgreifen.

Die Vorteile von Monolithen - Kommunikation ohne Latenzzeiten, Verfügbarkeit der Komponenten sowie bewährtes Transaktions- und Autorisierungs-Management – lassen sich weiterhin nutzen. Was für Beton gilt, gilt auch für Monolithen: Es kommt darauf an, was man daraus macht.

Standardverfahren in APEX

Dateien in Oracle APEX

Das Hochladen und Bereitstellen von Dateien in einer Oracle Application Express (APEX)-Anwendung ist eine gängige Anforderung. Dieser Artikel stellt die Standardverfahren vor, um eine solche Anforderung umzusetzen.

Dateien in einer APEX-Applikation

Mithilfe von Oracles Application Express (APEX) ist es möglich, schnell und einfach – dank deklarativer Entwicklung – lauffähige und ansprechende Webapplikationen zu erstellen. Durch die tiefe Integration in die Oracle-Datenbank können auch all deren Vorteile genutzt werden. Einer dieser Vorteile ist die Speicherung von Dateien direkt in der Datenbank, die über die Applikation geladen werden. Als Dateien kommen beispielsweise Rechnungen als PDF oder Anschreiben im Word-Format in Frage. Des Weiteren können über die APEX-Oberfläche dem Anwender diese Dateien aus der Datenbank zur Verfügung gestellt bzw. zum Herunterladen angeboten werden.

Das Element „Datei durchsuchen“

Die zentrale Komponente für das Hochladen einer Datei über die APEX-Applikation in die Datenbank ist das Element „Datei durchsuchen“. Dieses Element verfügt über all die Funktionalitäten, die notwendig sind, um eine lokale Datei vom Anwender in die Datenbank zu übermitteln. Dazu gehören: das Öffnen des „Datei öffnen“-Dialogs des Browsers – wie in Abbildung 1 zu sehen –, das Übertragen der Datei vom Client zum Server und das Bereitstellen der Datei innerhalb von Oracle zur Weiterverarbeitung. Für das Übermitteln der Datei wird eine Seitenweiterleitung, z. B. über eine Schaltfläche, benötigt.

Zu den wichtigsten Eigenschaften dieses Elements gehört die Festlegung, wohin die hochgeladene Datei geschrieben werden soll. Zur Auswahl stehen dabei zwei Speicherorte (Attribut: Speichertyp):

- Standardtabelle **`APEX_APPLICATION_TEMP_FILES`**
- Benutzerdefinierte Tabelle mit einer BLOB-Spalte

Diese beiden Speicherorte werden nun genauer vorgestellt.

Hochladen einer Datei in die Standardtabelle …

Die von Oracle vorgegebene, standardmäßige Zieltabelle für Dateien ist **`APEX_APPLICATION_TEMP_FILES`**. Genau genommen handelt es sich hierbei nicht um eine Tabelle, sondern um eine View, die auf eine interne Tabelle (**`wwv_flow_file_objects$`**) referenziert. Hinweis: Da in der View auf eine gültige APEX-Session-ID geprüft wird, liefert diese außerhalb von APEX, wie im SQL-Developer, keine Zeilen zurück.

Damit die Tabelle **`APEX_APPLICATION_TEMP_FILES`** nicht „unendlich“ wächst bzw. mit veralteten Dateien gefüllt bleibt, werden die hochgeladenen Dateien automatisch gelöscht. Zur Auswahl stehen die beiden Löschzeitpunkte:

- am Ende vom Request
- am Ende der Session

Dadurch ist der Entwickler gezwungen, die hochgeladene Datei in eine eigene Tabelle zu kopieren bzw. weiterzuverarbeiten. Dazu wird ein PL/SQL-Prozess benötigt, der eben jene Aktion bei der Seitenverarbeitung durchführt. Dieser Prozess kann wie in der Abbildung 2 aufgebaut sein.

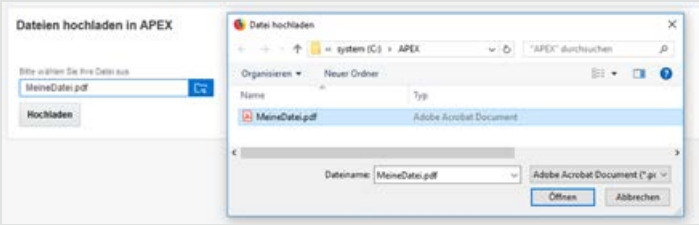


Abb. 1: Das Element „Datei durchsuchen“ auf einer Applikationsseite mit dem geöffneten Browser-Fenster zum Auswählen einer Datei.

```
BEGIN
  INSERT INTO dateien_apex_demo1 (dateiname, datei)
  SELECT filename, blob_content
    FROM apex_application_temp_files
   WHERE name = :Px_UPLOAD;
  COMMIT;
END;
```

Abb. 2: PL/SQL-Prozess-Code für eine einzelne Datei mit der Standardtabelle. Hierbei wird mittels einer SELECT-Abfrage auf die View APEX_APPLICATION_TEMP_FILES die Datei geholt, die zuletzt über das Element „Datei durchsuchen“ hochgeladen wurde.

```
DECLARE
  dateinamen APEX_T_VARCHAR2;
  datei      APEX_APPLICATION_TEMP_FILES%ROWTYPE;
BEGIN
  dateinamen := APEX_STRING.SPLIT (
                  p_str => :Px_UPLOAD
                , p_sep => ':' );
  FOR i IN 1 .. dateinamen.COUNT
  LOOP
    SELECT *
      INTO datei
      FROM apex_application_temp_files
     WHERE name = dateinamen(i);

    INSERT INTO datei_tabelle (dateiname, datei)
    VALUES (datei.filename, datei.blob_content);
  END LOOP;
  COMMIT;
END;
```

Abb. 3: PL/SQL-Prozess-Code für mehrere Dateien mit der Standardtabelle. Im ersten Schritt wird die Liste der Dateinamen aus dem Element geholt und mittels der Funktion SPLIT im Package APEX_STRING am Komma getrennt. Es liegt eine Liste der Dateien vor, welche nun in einer Schleife Datei für Datei gelesen und in eine eigene Zieltabelle eingefügt werden.

```
CREATE TABLE dateien
( id NUMBER GENERATED ALWAYS AS IDENTITY
, dateiname VARCHAR2(100)
, mime_type VARCHAR2(100)
, charset VARCHAR2(20)
, zuletzt_aktualisiert DATE
, datei BLOB
, CONSTRAINT pk_dateien PRIMARY KEY (id)
);
```

Abb. 4: Anweisung zum Erstellen einer Tabelle mit Spalten für die Datei als auch die Metainformationen.

Des Weiteren kann festgelegt werden, ob nur eine oder gleich mehrere Dateien hochgeladen werden können (Attribut „Mehrere Dateien zulassen“). Ist dieses Attribut auf „ja“ eingestellt, so muss der in der Abbildung 2 gezeigte PL/SQL-Code angepasst werden. Da nun mehrere Dateien aus der Standardtabelle gelesen und in eine eigene Tabelle eingefügt werden müssen, wird eine Schleife benötigt. Somit wird der Code, wie in Abbildung 3 dargestellt, erweitert. Im Element selbst ist als Wert der jeweilige Dateiname hinterlegt. Bei mehreren Dateien werden die Namen jeweils durch ein Komma jeweils getrennt.

Eine weitere Eigenschaft vom Element ist die Einschränkung auf bestimmte Dateitypen. Diese Angabe bestimmt, welche Dateitypen im „Datei durchsuchen“-Dialog des Browsers angezeigt werden sollen. Für PDF-Dateien wird beispielsweise „application/pdf“ angegeben. Hinweis: Der Benutzer hat dennoch die Möglichkeit, einen anderen Dateityp als PDF hochzuladen, in dem er im Browser-Dialog „alle Dateien“ auswählt und eine entsprechende Datei wählt. Es sollte also noch zusätzlich auf der Serverseite eine Prüfung erfolgen.

...in eine benutzerdefinierte Tabelle

Alternativ zur Verwendung der Standardtabelle kann auch eine eigene Tabelle als direktes Ziel zum Hochladen verwendet werden. Dies bietet die Möglichkeit, eine bessere Kontrolle über das Wachstum der Tabelle (Quota) als auch über weitere Eigenschaften (Tablespace, Rechte, etc.) zu behalten. Die zu definierende Tabelle benötigt mindestens eine Spalte vom Typen BLOB, um die Datei aufzunehmen. Zudem können Spalten für Metainformationen sinnvoll sein, welche zusätzlich vom Element „Datei durchsuchen“ gesetzt werden können. Dazu gehören:

- MIME-Type: die Art der Datei (PDF, Text etc.)
- Dateiname: der Name der hochgeladenen Datei
- Zeichensatz: dieser Wert wird nicht direkt vom Element gefüllt, sondern von einem anzugebenden Element abgeleitet
- Datum der letzten Aktualisierung des BLOBs (wird für das Browser-Caching verwendet)

Eine beispielhafte Tabellendefinition ist in Abbildung 4 zu finden.

Zum Hochladen einer Datei wird wieder das Element vom Typen „Datei durchsuchen“ verwendet. Der Speichertyp ist hier aber diesmal „BLOB Spalte in Element-Quelle definiert“. An dieser Stelle können auch die zuvor genannten weiteren Dateieigenschaften unter Angabe der Spaltennamen der Zieltabelle eingetragen werden. Unter der Quelle (Typ: Datenbankspalte) des Elements muss die Tabellenspalte angegeben werden, in die die Datei geladen werden soll. Diese Einstellungen sind in Abbildung 5 zu finden.

Neben dem eigentlichen Element zum Hochladen wird noch ein Prozess benötigt, welcher die eigentliche DML-Operation bei der Seitenweiterleitung ausführt. Dieser Prozess vom Typen „Automatische Zeilenverarbeitung (DML)“ erwartet die Angabe der Zieltabelle sowie die Angabe der Primärschlüsselspalte und des Elements, welches den Wert des Schlüssels enthält, der vorab durch eine Berechnung o. Ä. bestimmt wird. Alternativ dazu kann das Element auch leer bleiben, wenn die ID durch eine Sequenz (Identity-Spalte) oder einen Trigger bestimmt wird. Im Anschluss an diesen DML-Prozess können noch weitere benutzerdefinierte (PL/SQL-) Prozesse gestartet werden, die diese hochgeladene Datei weiterverarbeiten.

Hinweis: Wird die Seite über eine Schaltfläche abgeschickt, so muss diese unter ihren Eigenschaften im Attribut „Datenbankaktion“ auf „SQL-INSERT-Aktion“ gestellt sein, damit der DML-Prozess aufgerufen wird.

Herunterladen einer Datei mittels Schaltfläche

Um dem Anwender der Applikation Dateien aus der Datenbank zur Verfügung zu stellen (im Sinne von Download im Browser), gibt es in APEX mehrere Möglichkeiten. Zwei Wege sollen hier einmal exemplarisch vorgestellt werden. Eine Möglichkeit besteht darin, dem Benutzer auf einer

Seite eine Schaltfläche zu präsentieren. Durch Anklicken dieser Schaltfläche wird der Download-Vorgang für eine Datei gestartet und der Anwender erhält den typischen Browser-Dialog zum Speichern bzw. direkten Anzeigen (browser- und dateitypabhängig).

Dafür wird neben einer Schaltfläche eine weitere Seite als Hilfsseite benötigt, die als Ziel dient. Diese Seite soll keinen eigenen Inhalt besitzen, sondern wird nur benötigt, um einen PL/SQL-Prozess zu starten, welcher die Datei zum Download bereitstellt. Als Zeitpunkt wird der frühestmögliche Zeitpunkt auf der Seite gewählt: vor Header.

Die Abbildung 6 zeigt solch einen exemplarischen PL/SQL-Prozess. In diesem werden die folgenden Aktionen durchgeführt:

- Die zu liefernde Datei wird mittels **`SELECT INTO`**-Statement geladen.
- Größe der Datei ermitteln, damit diese beim Download-Dialog im Browser angezeigt werden kann (Restdauer ermitteln, etc.).
- HTTP-Header erstellen mittles der Packages **`HTP`** und **`OWA_UTIL`**. Hier werden im Wesentlichen drei Werte mitgegeben:
  - MIME-Header: hier statische PDF; diese könnte jedoch auch dynamisch eingefügt werden
  - die Größe der Datei (**`Content-Length`**)
  - Anzeige der Datei (**`Content-Disposition`**): entweder direkt im Browser (**`inline`**) oder über den Speichern-Dialog (**`attachment`**). Hier ist dieser Wert statisch hinterlegt, könnte aber auch dynamisch mittels einer Variablen realisiert werden. Der Dateiname wird ebenfalls hinterlegt (**`filename`**).
- Der eigentliche Download der Datei mittels der Prozedur **`DOWNLOAD_FILE`** im Package **`WPG_DOCLOAD.`**
- Stoppen der APEX Engine mit der Prozedur **`STOP_APEX_ENGINE`** im Package **`APEX_APPLICATION`**.

Dies ist nur eine Beispiel-Implementierung und könnte an einigen Stellen noch dynamisch(-er) gestaltet werden. Alternativ könnte dieser Prozess auch als Applikations-Prozess erstellt werden, welcher dann mittels einer dynamischen Aktion aufgerufen wird.

Bereitstellen von Dateien in einem Bericht

Eine weitere Möglichkeit zum Herunterladen von Dateien besteht darin, in einem Bericht pro Zeile jeweils einen Download-Link bereit zu stellen. Dies hat den Vorteil, dass auf einer Seite mehrere Dateien heruntergeladen werden können.

Dazu wird beispielsweise ein klassischer Bericht auf einer Tabelle erstellt, die auch Dateien beinhaltet. Da der Bericht Spalten vom Typen **`BLOB`** nicht direkt ausgeben kann, muss hier ein „Umweg“ gegangen werden. In den Spalteneigenschaften des Berichts wird die entsprechende Spalte ausgewählt. Der Typ wird von „Text“ auf „BLOB herunterladen“ geändert und die entsprechenden BLOB-Attribute eingestellt (aus welcher Tabelle kommt die Da-

Abb. 5: Einstellungen am Element für das Hochladen in die zuvor erstellte eigene Zieltabelle

```
DECLARE
  vb_datei BLOB;
  vc_dateiname VARCHAR2(100);
  vn_size NUMBER;
BEGIN
  SELECT datei, dateiname
    INTO vb_datei, vc_dateiname
    FROM dateien
   WHERE id = :Px_ID;

  vn_size := DBMS_LOB.GETLENGTH(vb_datei);

  HTP.INIT;
  OWA_UTIL.MIME_HEADER('application/pdf', FALSE);
  HTP.P('Content-Length: ' || vn_size);
  HTP.P('Content-Disposition: attachment;
         filename="'||vc_dateiname||'"');
  OWA_UTIL.HTTP_HEADER_CLOSE;
  WPG_DOCLOAD.DOWNLOAD_FILE(vb_datei);
  APEX_APPLICATION.STOP_APEX_ENGINE;
END;
```

Abb. 6: PL/SQL-Code zum Laden einer Datei aus der Tabelle DATEIEN und Bereitstellen dieser im Browser

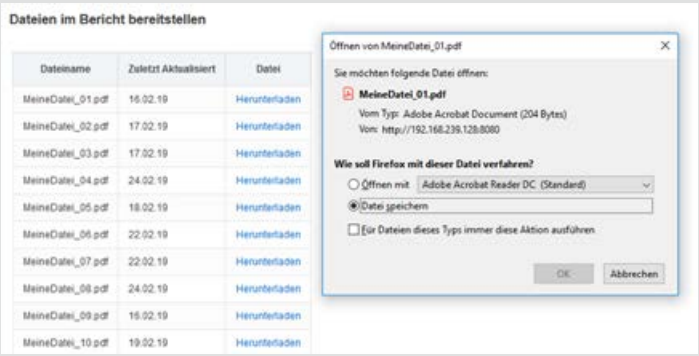


Abb. 7: Die Dateien in der Quelltabelle werden innerhalb eines Berichts aufgelistet. Beim Klick auf den jeweiligen „Herunterladen"-Link wird die Datei ausgeliefert und der entsprechende Browserdialog erscheint.

Glossar

Oracle Application Express (APEX)
Oracle Application Express ist eine webbasierte Softwareentwicklungsumgebung, die in einer Oracle-Datenbank ausgeführt wird. Es wird vollständig unterstützt und ist standardmäßig in allen Oracle-Database-Editionen enthalten.

Binary Large Object (BLOB)
Binary Large Objects sind große binäre Datenobjekte wie z. B. Bild- oder Audiodateien.

Data Manipulation Language (DML)
DML ist der Teil einer Datenbanksprache, die verwendet wird, um Daten zu schreiben, zu lesen, zu ändern und zu löschen.

MIME-Type
Der Mime-Type bezeichnet die Art einer Datei. Der MIME-Type besteht aus zwei Teilen, die durch einen Schrägstrich getrennt werden (z.B. application/pdf, application/msword).

Quellen

[Q1] „Oracle Database Online Documentation 11g Release 2 (11.2)", Kapitel „How to Upload and Download Files in an Application" https://docs.oracle.com/cd/E11882_01/appdev.112/e11945/up_dn_files.htm#HTMAD008

tei, Primärschlüssel dieser Tabelle, MIME-Type-Spalte etc.). Da dieser Spaltentyp einen numerischen Wert erwartet, wird in der Select-Abfrage im Bericht für den **BLOB**-Wert ein numerischer Wert eingesetzt (hier: die Größe der Datei, ermittelt durch die Funktion **GETLENGTH** im Package **DBMS_LOB**). Die **SELECT**-Abfrage:

```
SELECT id
     , dateiname
     , zuletzt_aktualisiert
     , DBMS_LOB.GETLENGTH(datei) AS datei
  FROM dateien
ORDER BY id
```

Abbildung 7 zeigt das Ergebnis: APEX bietet im Bericht pro Zeile einen Download-Link an, welcher beim Anklicken die jeweilige Datei liefert.

Zusammenfassung

Es lässt sich festhalten, dass die Implementierung der Funktionalität zum Hochladen von Dateien in der Applikation recht einfach und ohne allzu großen Aufwand zu realisieren ist. Die bereitgestellten Komponenten sind hochgradig deklarativ konfigurierbar. Das Bereitstellen von Dateien hingegen ist je nach „Einsatzort" verschieden aufwendig: auf Basis eines Reports ist diese Aufgabe einfach umsetzbar, einzelne Dateien hingegen erfordern im Verhältnis dazu ein wenig mehr Aufwand.

Andreas Upgang
(info@ordix.de)

Eine für Alle

Polymorphic Table Function

Tabellenfunktionen sind schon seit längerer Zeit Bestandteil von Oracle-Datenbanken. Die Polymorphic Table Functions sind mit dem neuen Oracle-Datenbank-Release 18c als neuer Typ von Tabellenfunktionen eingeführt. Der folgende Artikel beschreibt die grundlegende Funktionalität von Polymorphic Table Functions und zeigt zudem ein kleines Beispiel.

Definition

Der Begriff polymorph kommt aus dem Griechischen und bedeutet so viel wie „viele Formen“. In der objektorientierten Programmierung gibt es polymorphe Objekte, mit der Besonderheit, verschiedene Formen (Eigenschaften/Datentypen) annehmen zu können.

Polymorphic Table Functions (PTF) sind benutzerdefinierte Funktionen, die wie eine herkömmliche Tabelle in der **FROM**-Klausel einer **SELECT**-Anweisung aufgerufen werden können (siehe Abbildung 1).

Polymorphic Table Functions werden nicht explizit für eine Tabelle, sondern in der Regel für mehrere Tabellen entwickelt. Die zu verarbeitende Tabelle oder Tabellen werden als Eingabeparameter beim Funktionsaufruf mitgegeben. Zum Zeitpunkt der Erstellung der PTF ist der Zeilenaufbau der Eingabetabelle nicht bekannt. Auch der Zeilenaufbau der Rückgabestruktur ist zum Kompilierzeitpunkt der PTF nicht bekannt.

Die Tatsache, dass Polymorphic Table Functions Bestandteil von ANSI 2016, unterstreicht die Bedeutung, die PTFs mittlerweile in der Praxis, vor allem aber auch bei anderen Datenbankherstellern erreicht hat.

Verschiedene Rollen der Function

Das Package **DBMS_TF** ist neu in 18c und liefert die Schnittstelle zum Implementieren von Polymorphic Table Functions. Der Entwickler hat damit die Möglichkeit, bis zu vier Unterfunktionen zu nutzen:

- **DESCRIBE-**Funktion (obligatorisch)
- **OPEN**-Prozedur (optional)
- **FETCH_ROWS**-Prozedur (optional)
- **CLOSE**-Prozedur (optional)

Zudem beinhaltet das Package mehrere Datentypen, die bei der Entwicklung von PTFs genutzt werden können. Dies wird in den kommenden Abschnitten eingehender betrachtet.

Verschiedene Rollen der Function

PTFs bringen drei verschiedene Rollen mit sich.

- Polymorphic-Table-Function-Autor
- Query-Autor
- DBMS

Der Polymorphic-Table-Function-Autor erstellt den Mechanismus der Polymorphic Table Function. Er generiert eine Ergebnistabelle und liefert die Schnittstelle, diesen Mechanismus zu nutzen. Der Autor der PTF benötigt kein Wissen über technische Details, die hinter den Funktionen des **DBMS_TF**-Packages liegen.

Der Query-Autor nutzt die öffentliche Schnittstelle der Polymorphic Table Function. Er kann diese Schnittstelle im **FROM**-Teil einer **SELECT**-Anweisung nutzen.

Das DBMS ist verantwortlich für die Kompilierung, Ausführung und Zustandsverwaltung der PTF.

Welche Möglichkeiten bieten Polymorphic Table Functions?

Grundsätzlich bieten Polymorphic Table Functions die Möglichkeit, Zeilen und Spalten in der Ausgabestruktur hinzuzufügen, sowie das Ausblenden von Zeilen und Spalten. Zudem ist es möglich die Werte von neuen Zeilen und Spalten durch Berechnungen, Aggregationen und eigene Analysen zu belegen.

Typen von Polymorphic Table Functions

Es gibt zwei Typen von Polymorphic Table Functions:

- Row Semantic
- Table Semantic

Bei einer Row-Semantic-PTF wird auf Basis des aktuellen Datensatzes eine neue Spalte für diesen generiert. Alternativ kann eine neue Zeile auf Basis des aktuellen Datensatzes generiert.

```
SELECT *
FROM ptf_pkg.excl_cols(mitarbeiter,
COLUMNS(BERUF,VORGESETZTER));
```

Abb. 1: Die SELECT-Funktion

Eine Table-Semantic-PTF zeichnet sich dagegen dadurch aus, dass alle zuvor gelesenen Zeilen betrachtet werden, um eine neue Zeile bzw. eine neue Spalte zu generieren. Diese Funktionalität eignet sich, um Daten zu aggregieren oder zu analysieren.

Die Eingabe der Polymorphic Table Function kann, wie bei analytischen Funktionen, partitioniert und geordnet werden. Beim Aufruf der Polymorphic Table Function muss die Partitionierung mit angegeben werden.

Welche Anwendungsfälle gibt es für polymorphe Tabellenfunktionen?

Anwendungsfälle für Polymorphic Table Functions leiten sich von den Typen der Funktionen ab.

Für Row Semantic PTFs ist ein vereinfachtes Beispiel das Berechnen eines Gesamtverdiensts, wenn in einem Datenmodell Gehalt, Provision und Boni erfasst werden. Zudem ist es, möglich Spalten zu trennen, umzubenennen, auszublenden oder zu bearbeiten.

Eine weitere Möglichkeit ist das Zusammenfassen von Zeilen in einer Spalte in einem einheitlichen Format wie beispielsweise JSON oder XML.

Table Semantic PTFs eignen sich hervorragend, um Tabellen mittels benutzerdefinierter Analysen auszuwerten. Dieses kann beispielsweise mittels Aggregationen oder Window-Funktionen erfolgen.

Funktionsweise

Eine Polymorphic Table Function muss in einem PL/SQL-Package implementiert werden. Die Schnittstelle zu der Polymorphic Table Function muss nicht zwangsläufig in dem Package implementiert werden. Sie kann auch als Stand-Alone-Function umgesetzt werden. Diese Schnittstelle muss mindestens eine Tabelle entgegennehmen. Tabellenargumente sind entweder **`WITH`**-Klauselabfragen oder Objekte auf Schemaebene, die in einer **`FROM`**-Klausel zulässig sind, wie beispielsweise Tabellen, Views oder Tabellenfunktionen.

Weitere Argumente für Polymorphic Table Functions können die Standard-Argumente sein, wie z.B. eine Cursor-Variable oder Collection, die an eine herkömmliche Tabellenfunktion übergeben werden können.

Mit Oracle Database Release 18c wird das Konzept des variablen Pseudooperators eingeführt. Ein variabler Pseudooperator kann eine Liste von Werten annehmen, muss aber mindestens ein Argument besitzen. Während der SQL-Kompilierung werden die Pseudo-Operatoren in entsprechende **`DBMS_TF`**-Typen konvertiert und anschließend an die **`describe`**-Funktion übergeben. Ein Beispiel hierfür ist der Pseudo-Operator Columns, zu sehen in Abbildung 1.

So kann mittels eines Pseudo-Operators eine Liste an Spalten an eine Polymorphic Table Function übergeben werden. Zudem muss das PTF-Package eine **`describe`**-Funktion beinhalten. Dazu in dem folgenden Abschnitt mehr.

Der eigentliche Inhalt der PTF kann sowohl in dem Package als auch als Funktion auf Schemaebene, die aus dem Package heraus aufgerufen wird, implementiert werden.

Die describe-Funktion

Die **`describe`**-Funktion beschreibt die Ergebniszeile der PTF. Die **`describe`**-Funktion wird während des Parsens der PTF aufgerufen. Die Parameter der **`describe`**-Funktion müssen mit den Parametern der Schnittstelle der PTF übereinstimmen. Zu beachten ist hierbei, dass die **`describe`**-Funktion die in dem Package **`DBMS_TF`** definierten Datentypen benötigt. Zudem benötigt die **`describe`**-Funktion mindestens einen Parameter des Typs **`DBMS_TF.TABLE_T`**.

Die Pseudo-Operatoren werden während der SQL-Kompilierung des PTF-Aufrufs in den entsprechenden **`DBMS_TF`**-Typen konvertiert und anschließend an die **`describe`**-Funktion übergeben.

Konstante skalare Werte werden unverändert und alle anderen Werte als **`NULL`** an die **`describe`**-Funktion übergeben. Die **`describe`**-Funktion liefert einen **`DBMS_TF.DESCRIBE_T`**-Datensatz zurück. Dieser beschreibt die Ergebniszeile der PTF und informiert die Datenbank über die zu verarbeitenden Datentypen der Zeile.

In Abbildung 2 wird ein Beispiel für eine **`describe`**-Funktion dargestellt. Die **`describe`**-Funktion nimmt genau wie die PTF-Schnittstelle eine Tabelle und eine Spaltenliste auf. Wie bereits erläutert unterscheiden sich die Parameter zwischen Schnittstelle und **`describe`**-Funktion ausschließlich in den Datentypen (siehe Abbildung 2).

Der Package Body

Im Package Body wird die Table Function implementiert. In Abbildung 3 wird ein Beispiel gezeigt, dass die übergebenen Spalten ausblendet. Dazu werden im ersten Schritt alle Spalten der übergebenen Tabelle durchlaufen. Im zweiten Schritt werden zu jeder gelieferten Spalte die übergebenen auszublendenden Spalten durchlaufen.

Anschließend bekommt die aktuelle Spalte ein **`pass_through`**-Flag zur Ausgabe gesetzt. Ist der Spaltenname

der übergebenen Tabelle ungleich des Spaltennamens der übergebenen Spaltenliste, so wird das **pass_through**-Flag auf **TRUE** gesetzt, ansonsten auf **FALSE**.

Im letzten Schritt wird die innere Schleife verlassen, wenn das **pass_through**-Flag nicht auf **TRUE** steht (siehe Abbildung 3).

Fazit

Mit den polymorphen Tabellenfunktionen liefert Oracle ein umfangreiches neues Feature mit vielfältigen Möglichkeiten nach ANSI 2016.

PTFs bieten die Möglichkeit, Logik für Tabellen zu implementieren die unabhängig von der Input-Tabelle ist. Da auch der Output dieser Logik vorher nicht bekannt sein muss, sind diese Funktionen sehr dynamisch einsetzbar. Durch klar definierte Schnittstellen ist es für Entwickler möglich, polymorphe Tabellenfunktionen zu entwickeln, ohne großartiges Wissen über technische Details zu besitzen.

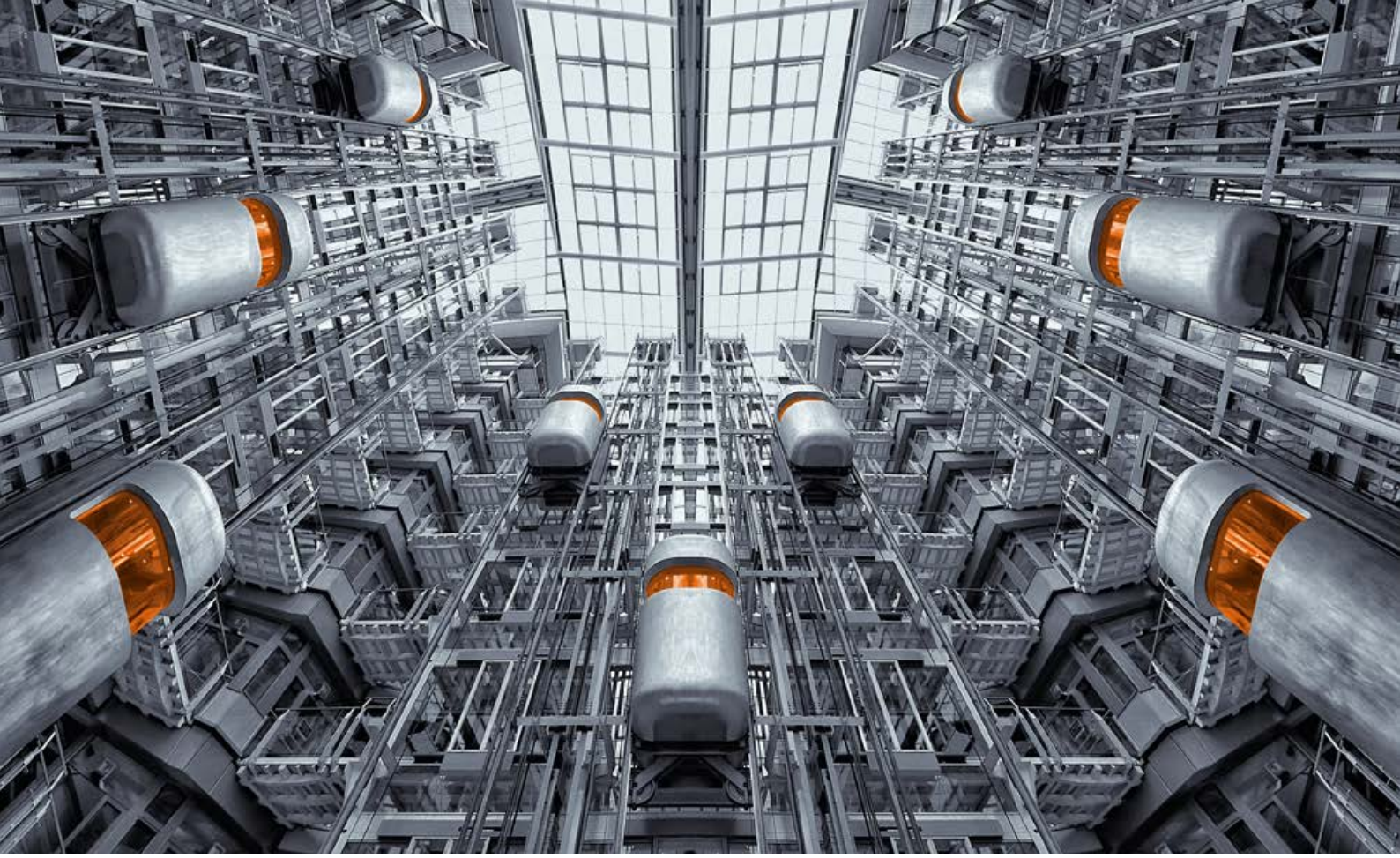
Tobias Flüter
(info@ordix.de)

```
CREATE OR REPLACE PACKAGE ptf_pkg AS
  FUNCTION excl_cols(tab IN TABLE,
                     col IN COLUMNS)
  RETURN TABLE PIPELINED
  ROW POLYMORPHIC USING ptf_pkg;
 FUNCTION describe (tab IN OUT DBMS_TF.table_t,
                    col IN     dbms_tf.columns_t)
    RETURN DBMS_TF.describe_t;
END ptf_pkg;
/
[…]
SELECT *
FROM ptf_pkg.excl_cols(mitarbeiter,
COLUMNS(BERUF,VORGESETZTER));
```

Abb. 2: Die DESCRIBE-Funktion

```
CREATE OR REPLACE PACKAGE BODY ptf_pkg AS
  FUNCTION describe (tab IN OUT DBMS_TF.table_t,
                     col IN     dbms_tf.columns_t)
    RETURN DBMS_TF.describe_t
  AS
  BEGIN
    FOR i IN 1 .. tab.column.count() LOOP
      FOR j IN 1 .. col.count() LOOP
        tab.column(i).pass_through := (tab.column(i).de-
scription.name != col(j));
        EXIT WHEN NOT tab.column(i).pass_through;
      END LOOP;
    END LOOP;
    RETURN NULL;
  END;
END ptf_pkg;
/
```

Abb. 3: Der Package Body



Performance-Faktoren von IBM DataStage

Ein Blick in die Architektur und Performance von IBM DataStage

Das Extrahieren, Transformieren und Laden von Daten gehört für viele Unternehmen, zur alltäglichen Routine. Um diesen Prozess abzubilden, gibt es eine Fülle von verschiedenen ETL-Tools am Markt. Eines davon ist IBM DataStage, welches Bestandteil des IBM InfoSphere Softwareangebots ist. Schlagwörter wie „Cluster Computing“ und „horizontale Skalierbarkeit“ scheinen in den letzten Jahren vor allem durch Big-Data-Technologien ein neuer Trend geworden zu sein. Schon davor hat das etablierte und als Marktführer bezeichnete ETL-Tool IBM DataStage einige derartige parallele Konzepte vereinigt und für die breite ETL-Kundschaft im Angebot.

IBM DataStage

IBM DataStage ist Bestandteil von IBMs InfoSphere-Information-Server-Softwareangebots. Das Portfolio umfasst verschiedene informationsverarbeitende Applikationen mit der Zielsetzung, dem Kunden eine zentrale, agile End-2-End-Infrastruktur für die Verarbeitung und Erzeugung von präzisen, konsistenten, aktuellen und kohärenten Informationen zu bieten. Seit mehreren Jahren gilt IBM DataStage als einer der etablierten Marktführer im Bereich ETL. Ein großer Vorteil der Software-Suite sind die „Parallel Processing“-Möglichkeiten und die daraus resultierende Performance. Der Ursprung der parallelen Funktionen geht auf die Firma Ascential Software zurück. Diese kaufte zu damaliger Zeit die Rechte an Orchestrate, einem ETL-Werkzeug mit gleichnamigem Framework für parallele Shell-Ausführungen, und integrierte diese in DataStage. Dies erlaubt DataStage eine hohe parallele Skalierbarkeit [2-4].

Der nachfolgende Artikel gibt einen Einblick in die Architektur der Software und erörtert einige Performance-Aspekte von IBM DataStage. Für einen gelungenen

Einstieg zu dem Thema ETL und insbesondere DataStage bietet sich die dreiteilige ORDIX® news Reihe „ETL im Data Warehouse am Beispiel IBM DataStage“ an.

Architektur

DataStage kann grob in drei Komponenten aufgeteilt werden, die anhand der Abbildung 1 im Folgenden beschrieben werden.

- **DataStage-Repository:** Bezeichnet den Speicherort der Metadaten von IBM DataStage und insbesondere der jeweiligen ETL-Strecken. Das Repository liegt in einer relationalen Datenbank und ist nicht beschränkt auf IBM DB2. Die parallele Job-Engine, auf denen ein paralleler DataStage-Job aufbaut, basiert auf einer proprietären Skriptsprache namens OSH (Orchestrate). Datastage-Jobs werden in OSH-Skripte übersetzt und ebenfalls im Repository neben weiteren Informationen wie beispielsweise Datenbankverbindungen, selbstdefinierten Parametern, Routinen und software-spezfischen Informationen abgelegt.
- **DataStage-Client-Applikationen:** Bestehen aus mehreren grafischen Benutzeroberflächen, die das Konfigurieren, Überwachen, Designen und Verwalten von ETL-Strecken und Metadaten erlauben. Die Client-Applikationen müssen sich über einen DataStage-Server mit einem Repository verbinden, da dies der Speicherort aller Informationen und Metadaten ist. Innerhalb des DataStage-Designers werden die Schnittstellen über eine grafische Oberfläche erstellt und anschließend in die Skriptsprache OSH kompiliert.
- **DataStage Engine:** Bezeichnet die Server-Applikation, die auf einem oder mehreren Servern als Single- oder Multiserver-Cluster-Konfiguration installiert ist. Zur Laufzeit eines parallelen Jobs wird auf jedem konfigurierten Serverknoten das Kompilat der Schnittstelle verteilt und ausgeführt. Abhängig von der jeweilige Schnittstelle verbinden sich dann die Knoten mit den verschiedenen Datenquellen und Zielsystemen und extrahieren, transformieren und laden Daten gemäß der definierten Datentransformationen.

Datenpartitionierung & Parallele Verarbeitung

Der mit unter größte Performance-Faktor wird durch die Anzahl der bereitgestellten Knoten in Verbindung mit den darunter liegenden Servern bestimmt. In der Praxis verwendet man mehrere Server mit der DataStage-Installationsinstanz, auf denen wiederum ein oder mehrere Knoten konfiguriert sind. Jeder Knoten verarbeitet zu der Laufzeit einer Schnittstelle die jeweiligen Daten. Entscheidend ist dabei die Aufteilung der Daten auf die verschiedenen Knoten. Dies kann über sogenannte „Partitioners“ innerhalb der Schnittelle definiert werden. Die zu verarbeitenden Daten werden partitioniert und über das Netzwerk auf die verschiedenen Knoten aufgeteilt. Auf jedem Knoten wird anschließend die Verarbeitung unabhängig voneinander parallel ausgeführt. Die Art der Partitionierung

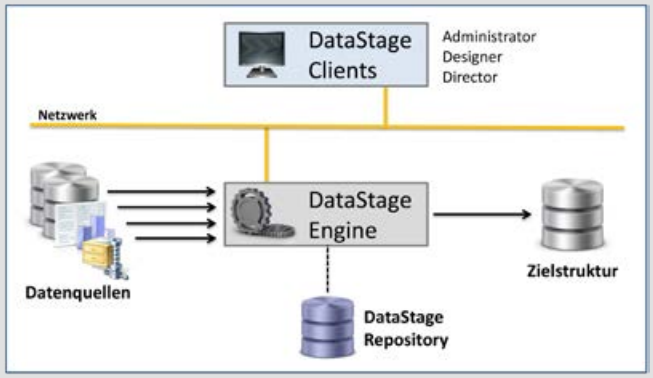


Abb. 1: Architektur DataStage
(Eigene Darstellung in Anlehnung an DSXchange (2008) S. 3)

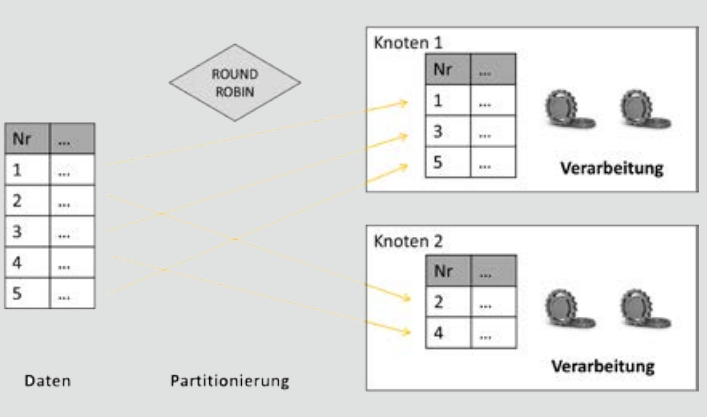


Abb. 2: Datenpartitionierung mit Round Robin

wird über den Partitioner festgelegt. Es wird unterschieden zwischen schlüsselbasierten Partitionierungen und nicht schlüsselbasierten Partitionierungen. Bei nicht-schlüsselbasierten Partitionierungen können die Datensätze unabhängig der jeweiligen Spaltenwerte verteilt werden. Hierunter zählen Partitionierungsfunktionen wie „Round Robin“, „Random“, „Entire“ und „Same“. Bei der schlüsselbasierten Partitionierung werden einzelne Spaltenwerte berücksichtigt. Beispielsweise kann die Datenverteilung anhand der Hash-Werte bestimmter Spalten erfolgen oder durch eine Modulus-Berechnung auf Integer-basierten-Spalten.

Einige Verarbeitungsschritte, wie beispielsweise das Bilden einer Summe oder das Joinen verschiedener Datenströme, setzen eine korrekte Aufteilung der Daten voraus. Nur Daten, die gemeinsam auf einem Knoten liegen, werden gemeinsam verarbeitet. Wir nehmen an, dass einzelne Umsätze von jeweils verschiedenen Produktkategorien vorliegen. Möchte man beispielsweise die Umsatzsummen von den jeweiligen Produktkategorien ermitteln, so müssen die jeweiligen Umsatzzahlen einer Produktkategorie in der gleichen Partition liegen.

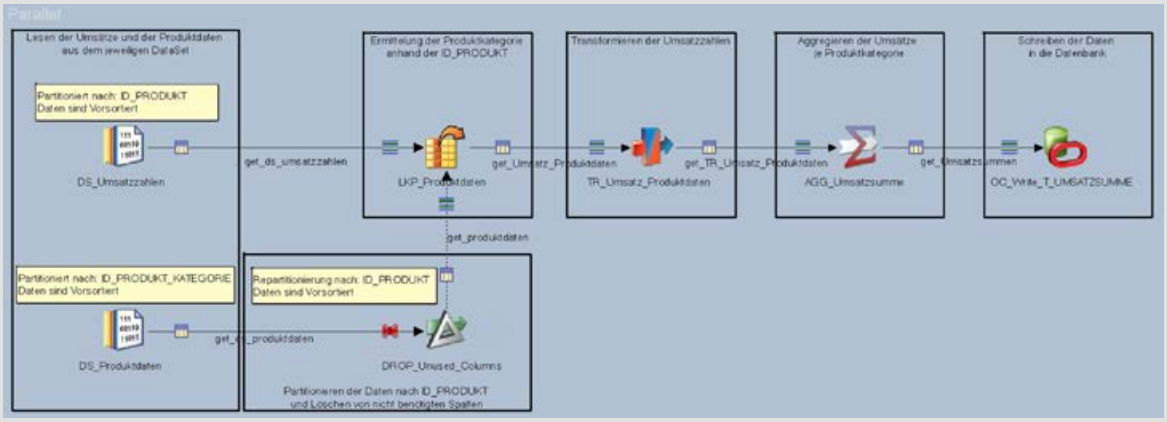


Abb. 4: DataStage-Job-Umsatzsummierung

Ziel der Partitionierung für eine performante Verarbeitung ist eine gleichmäßige Verteilung der Daten auf den verschiedenen Knoten unter der Maßgabe, die Anzahl an Repartitionierungen gering zu halten. Abbildung 2 zeigt die Partitionierung von fünf Datensätzen auf zwei verschiedenen Knoten. Die Partitionierung erfolgt nicht-schlüsselbasiert anhand einer Round-Robin-Funktion. Diese teilt die Datensätze, wie bei der Verteilung von Karten bei einem Kartenspiel, gleichmäßig auf. Jeder Knoten führt anschließend unabhängig von einander die anstehenden Verarbeitungsschritte aus.

Neben den Partitioners existieren sogenannte „Collectors“. Diese sind dafür zuständig, die Daten von mehreren Partitionen für eine sequenzielle Verarbeitung wieder zu vereinen. Zum Beispiel erfolgt das Schreiben von Daten in eine Datei (Sequential-File) sequenziell. Sind die Daten vorher über mehrere Knoten partitioniert, so vereinigt ein Collector zunächst die Daten auf einem Knoten, bevor die Daten in die Datei geschrieben werden. Eine detaillierte Auflistung der möglichen Partitionierungsarten können in den kostenfreien erhältlichen IBM Redbooks [1] entnommen werden .

Parallele DataSets

Die Notwendigkeit, Daten zwischenzuspeichern, ist vielfältig. Beispielsweise benötigt man ein Set von Daten in mehreren Verarbeitungen und möchte ähnliche oder gleiche Vortransformationen nur einmal ausführen. Oder man möchte Jobabschnitte definieren mit Restart-Points, wodurch das Speichern von Zwischenergebnissen notwendig ist. Für dieses Speichern eignet sich die Verwendung von „DataSets“ mit der namensgleichen DataSet-Stage. Während in sequenziellen Jobs die Daten in sequenziellen oder Hash-basierten Dateien zwischengespeichert werden, so werden im Parallel Framework DataSets verwendet. Diese ermöglichen das parallele Speichern und Lesen von Daten auf den jeweils konfigurierten Knoten. Dabei wird die vorherige Datenpartitionierung und Datensortierung beibehalten. Zudem werden die Daten in einem DataStage-spezifischen binären Datenformat gespeichert, welches ein performanteres Lesen und Schreiben ermöglicht, da die Daten nicht erst gesondert interpretiert werden müssen.

Beispiel-Job Umsatzsummierung

Eine typische DataStage-Verarbeitung besteht aus mehreren von einander abhängigen Stages. Eine Beispielverarbeitung ist in Abbildung 3 dargestellt. Es wird angenommen, dass ein DataSet mit Umsätzen für einzelne Produktverkäufe existiert. Ein anderes DataSet beinhaltet zu verschiedenen Produkten die Produktkategorie. Ziel ist es, die Umsatzzahlen je Produktkategorie zu ermitteln. In einem ersten Schritt werden die Daten über die DataSet-Stages **`DS_Umsatzzahlen`** und **`DS_Produktdaten`** ausgelesen und die Umsatzzahlen mit Produktdaten angereichert (LookUp-Stage **`LKP_Produktdaten`**). Zuvor werden für den Reference-Stream **`get_produktdaten`** alle nicht benötigten Felder gelöscht, um die Datenmenge gering zu halten (Modify-Stage **`DROP_Unused_columns`**). In einem zweiten Schritt werden die Umsatzzahlen in das richtige Format transformiert (Transformer-Stage **`TR_Umsatz_Produktdaten`**). In einem dritten Schritt werden die Umsätze aggregiert zum Bilden einer Umsatzsumme (Aggregator-Stage **`AGG_Umsatzsumme`**). Schlussendlich werden die Daten in die Datenbank geschrieben (Oracle-Stage **`OC_Write_T_UMSATZSUMME`**).

Inter-Operator Transport Buffering

Die einzelnen Verarbeitungsschritte sind abhängig von dem vorherigen Verarbeitungsschritt und die Daten müssen jede Stage gemäß der Pfeile durchlaufen. In einem DataStage-Sequential-Job wird jeweils jeder Prozessschritt einzeln und sequenziell auf diese gesamte Datenmenge angewendet, bevor der zweite Prozessschritt beginnt. DataStage-Parallel verarbeitet die Daten blockweise.

Wenn ein Block in Schritt 1 erfolgreich mit Produktdaten angereichert wurde, dann wird dieser Block bereits an Prozessschritt 2 übergeben und beginnt mit der Transformierung, während Prozessschritt 1 den entsprechend nächsten Datenblock verarbeitet. Einige Stages benötigen alle Daten oder eine entsprechende Datengruppe, um mit der Verarbeitung fortfahren zu können. Beispielsweise erfordert die Summierung der Umsatzzahlen, dass alle Umsätze für eine Produktkategorie bereits fertig transformiert wurden. In dem Beispiel wird angenommen, dass die Daten bereits nach Produktkategorie vorsortiert sind. Daher werden die Daten vor Prozessschritt 3 entsprechend pro Produktkategorie gepuffert, bis eine Aggregierung stattfinden kann. Dieses Vorgehen wird als Inter-Operator Transport Buffering bezeichnet und sorgt für eine gleichmäßige Auslastung der Ressourcen, da die Daten sich jeweils auf die Prozesse verteilen können.

Operator Combination

Zur Laufzeit eines Jobs wird anhand der hinterlegten Konfigurationen und des Job-Designs ein Ausführungsplan für den Job erstellt. Dieser bestimmt die genauen Prozesse und die Ausführungstopologie für die Jobausführung. Bei dem Erstellen des Ausführungsplan wird versucht, die Logik von verschiedenen Stages zusammenzulegen, um so die Anzahl benötigter Prozesse auf jedem Knoten zu reduzieren. In der Regel würde pro Stage ein Prozess auf jedem Knoten gestartet werden. Über die Funktion **`Operator Combination`** werden so verschiedene Schritte in einem Prozess erledigt. Dadurch wird Overhead auf den einzelnen Knoten vermieden und die Gesamtperformance steigt. Während der Ausführung des unten dargestellten Jobs kann es beispielsweise passieren, dass anstelle des Auslesens des gesamten DataSets **`DS_Produktdaten`** und dem anschließenden Löschen einiger Spalten nur die benötigen Spalten ausgelesen werden.

Schlusswort

Die in den vorherigen Abschnitten besprochenen Faktoren ermöglichen DataStage eine sehr solide Performance. Dadurch kann sich das Produkt weiterhin mit an der Spitze des Gartner Magic Quadrants behaupten. Nichtsdestotrotz bilden die hier vorgestellten Eigenschaften lediglich die Grundlage für eine performante Verarbeitung. Letztendlich spielen auch weitere Faktoren eine Rolle:

- **Das Job-Design**: Werden Daten transferriert, die in der Verarbeitung nicht benötigt werden? Werden die Datenströme sinnvoll und effektiv verarbeitet? Ist die Verarbeitungsreihenfolge sinnvoll oder gibt es unnötige Berechnungen? Können Repartitionierungen und Sortierungen eingespart werden?
- **Das Server Sizing/Hardware**: Sind die Maschinen, auf denen die DataStage-Knoten laufen, ausreichend dimensioniert? Stimmt die Netzwerkperformance? Ist genügend RAM bzw. schneller Scratch-Speicher verfügbar?
- **Die Konfiguration:** Wurde den DataStage-Knoten ausreichend RAM und Speicherplatz für ihre Berechnungen zugeordnet? Stimmen die eingestellten Puffergrößen mit den Anforderungen überein?

Spannend bleibt auch der Blick in die Zukunft. Die ETL-Werkzeugauswahl ist vielfältig und besteht neben alteingesessenen Marktführern wie IBM und Informatica auch aus neuen Herausforderern wie Oracle, Talend oder Pentaho. Die Qual der Wahl wird zunehmend größer und bedarf einer präzisen Abbildung der unternehmensspezifischen Anforderungen an eine Softwarelösung. Neben erforderlichen Leistungsfähigkeiten müssen auch einzuhaltende Nebenbedingungen sorgfältig identifiziert werden. Mehr dazu, gibt es in einer der folgenden ORDIX® news.

Tobias Ummler
(info@ordix.de)

Links

[1] Richtlinien für die Entwicklung von hocheffizienten und skalierbaren Informationsintegrationsanwendungen
http://www.redbooks.ibm.com/abstracts/sg247830.html

[2] ORDIX® news Artikel 3/2011:
ETL im Data Warehouse am Beispiel IBM DataStage (Teil III)
https://www.ordix.de/ordix-news-archiv/3-2011.html

[3] ORDIX® news Artikel 1/2011:
ETL im Data Warehouse am Beispiel IBM DataStage (Teil II)
https://www.ordix.de/ordix-news-archiv/1-2011.html

[4] ORDIX® news Artikel 4/2010:
ETL im Data Warehouse am Beispiel IBM DataStage (Teil I)
https://www.ordix.de/ordix-news-archiv/4-2010.html

Quellen

[Q1] DSXchange (2008)
“White Paper What is DataStage?”; Stand: 23.04.2016:
www.dsxchange.net/uploads/White_Paper_-_What_Is_DataStage.pdf

[Q2] Gartner (2015)
“Magic Quadrant for Data Quality Tools”; Stand: 14.03.2016:
https://www.gartner.com/doc/reprints?id=1-2SGV9XI&ct=151118&st=sb

[Q3] Gartner (2018)
„Magic Quadrant for Data Integration Tools“; Stand: 25.01.2019:
https://www.informatica.com/data-integration-magic-quadrant.html#fbid=7XdXd8TvKD9

[Q3] IBM (2010); Stand: 25.01.2018
„IBM InfoSphere DataStage Parallel Framework Standard Practices“
http://www.redbooks.ibm.com/abstracts/sg247830.html

Bildnachweis

© pixabay.com | Aufzug Berlin Ludwig Erhard Haus

Microsoft SharePoint (Teil II):

Suche und Kommunikation

Seit der Version 2010 bietet SharePoint eine sehr leistungsfähige Suchfunktion, die aufgrund ihrer Vielseitigkeit auch als „Schweizer Taschenmesser" bezeichnet werden kann. In Kombination mit Tools aus der Office-365-Produktfamilie können die Kommunikationsmöglichkeiten beträchtlich erweitert werden.

Teil 1: Suche

Das schnelle Finden von benötigten Informationen stellt eine zentrale Herausforderung im modernen Arbeitsalltag dar und bildet eine Schlüsselrolle für Produktivität und Effizienz. Die Notwendigkeit leistungsfähiger Suchwerkzeuge wird umso deutlicher, wenn man sich das Ausmaß der Informationsüberflutung im heutigen Arbeitsalltag vor Augen führt. Hierzu ein paar Zahlen: So haben Studien ergeben, dass ein durchschnittlicher Knowledge Worker pro Tag ca. 63.000 Wörter an neuen Informationen erhält und verarbeiten muss. Zum Vergleich: Die durchschnittliche Länge eines englischsprachigen Romans auf Amazon beträgt 64.531 Wörter! [Quelle: Agnes Molinar: SharePoint 2016 Search Explained]

Überhaupt verarbeitet der heutige Mensch pro Tag mehr Informationen als ein Mensch im 1.500 Jahrhundert während seines gesamten Lebens [Quelle: Agnes Molinar: SharePoint 2016 Search Explained].

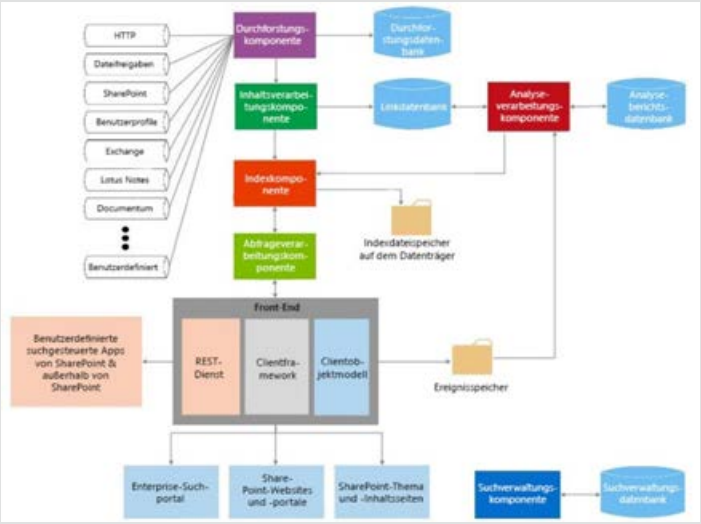


Abb. 1: Interaktion der Suchkomponenten (Quelle: Microsoft)

Dem gegenüber steht die tatsächliche User Experience bei der Suche: Viele Suchwerkzeuge, die in Unternehmen im Einsatz sind, haben das Manko, bei Suchanfragen zu viele potenzielle Treffer anzuzeigen, so dass sich sehr häufig die Arbeitserleichterung in Grenzen hält. Da das Nutzererlebnis geprägt ist vom Umgang mit Internet-Suchmaschinen, wird bei einer unzureichend konfigurierten Suche im Unternehmen oft sehr schnell völlig auf die Suche verzichtet.

Wenn wie bei einem Projekt Inhalte und Dokumente gezielt und strukturiert erstellt und abgelegt werden, können über Suchmechanismen, die eine gezielte – und gezielt verfeinerbare – Suche ermöglichen, sehr effiziente Arbeitsbedingungen geschaffen werden. Wertvolle Dienste leisten zudem Funktionalitäten, die die abgelegten Informationen automatisch gezielt nach vorgegebenen Kriterien auswerten und darstellen.

Für beides stellt SharePoint in Form von Suchcentern, anpassbaren Suchergebnisseiten und Such-Webparts (bzw. -Apps / -Add-Ins)[1] sehr leistungsfähige und granular konfigurierbare Werkzeuge zur Verfügung.

SharePoint-Suche

Mit der Übernahme der Enterprise Search Appliance der Firma FAST und der kontinuierlichen Integration der Search-Komponenten in die SharePoint-Produktfamilie seit der Version 2010 stellt Microsoft in den aktuellen SharePoint-Versionen ein sehr leistungsfähiges und anpassbares Enterprise-Suchwerkzeug bereit.

Dabei sind grundsätzlich zwei Ansätze zu unterscheiden: Über die Einrichtung und Einbindung von Such-Centern an verschiedenen Stellen der logischen und inhaltlichen SharePoint-Struktur steht eine „klassische" Suchfunktion über Sucheingabefeld und Ergebnisseiten zur Verfügung, die sehr flexibel an die spezifischen Anforderungen angepasst werden kann. Darüber hinaus kann die Suchfunktion in sog. suchgetriebenen Anwendungen innerhalb von Webparts oder über APIs auch programmatisch genutzt werden.

[1] Seit SharePoint 2013 fasst Microsoft auch Webparts unter der Sammelbezeichnung „Apps" zusammen. Seit SharePoint 2016 unter „Add-Ins"; zur Verdeutlichung der Funktionalität wird im Folgenden weiterhin die vormalige Bezeichnung verwendet

Suchtopologie

Voraussetzung insbesondere für Letzteres ist die seit SharePoint 2013 neu vorhandene Funktion des CONTINUOUS CRAWLINGS, durch die eine permanente Durchforstung der Datenquellen erfolgt (standardmäßig alle 15 Minuten, herabsetzbar auf 1 Minute), so dass eine sehr zeitnahe Ausgabe der Ergebnisse möglich ist.

Was die in SharePoint integrierte Such-Topologie anbelangt, reicht das Spektrum von Crawl- und Indexierungskomponenten bis hin zu Analytics-Komponenten (Abbildung 1): Durchsucht werden kann nicht nur die SharePoint-Umgebung, sondern darüber hinaus auch weitere Datenquellen wie Mail- und Fileserver, sowie u.U. auch weitere Informationsspeicher wie Datenbanken oder ERP-Systeme.

Gerade bei der Sucharchitektur propagiert Microsoft stark den Umstieg auf Hybrid-Modelle, indem Ressourcen-intensive Komponenten nach SharePoint Online ausgelagert werden. Neu seit SharePoint 2016 ist die Option, dass in Hybrid-Szenarien der lokale Suchindex einer On-Premise-Installation von Office 365 übernommen wird und bei jeder Suche Ergebnisse aus beiden Plattformen angezeigt werden.

Konfigurationsmöglichkeiten:

Die Anpassungsmöglichkeiten beginnen damit, dass verschiedene Suchergebnisquellen, d.h. Scopes, eingerichtet werden können und damit neben der Volltext-Suche gezielt Bereiche geschaffen werden können, die bei der Suche berücksichtigt werden sollen. Konfigurierbar sind diese Suchergebnisquellen über einen Abfrage-Generator, in dem mittels der Abfragesprache KEYWORD QUERY LANGUAGE (KQL) ggf. sehr differenziert Kriterien vorgegeben werden können. Neben Einstiegspunkten in der Seitenstruktur können hier auch inhaltliche Aspekte zum Tragen kommen. Sehr leistungsfähig im Hinblick auf gezielte Informationssuche wird dieses Instrument, wenn so z.B. Inhaltstypen als Scope angegeben werden.

Über die Abfragesprache KQL sind darüber hinaus sehr komplexe Anfragen an die Suche konfigurierbar, was vor allem bei suchgetriebenen Anwendungen eine große Rolle spielt. Neben KQL steht zusätzlich noch die etwas technischere/abstraktere Variante FAST QUERY LANGUAGE (FQL) zur Verfügung.

Grundlage für Abfragemechanismen und die Präsentationsmöglichkeiten für Suchergebnisse bildet das Konzept der durchforsteten und verwalteten Eigenschaften. Indem SharePoint automatisch nach der Erstellung und Befüllung von SharePoint-Inhalten bei der Durchforstung **`crawled properties`** (durchforstete Eigenschaften) erstellt, kann über zugeordnete **`managed properties`** (verwaltete Eigenschaften) über die SharePoint-Oberfläche auf diese zugegriffen werden. Reichen die automatisch vergebenen Eigenschaften der **`managed properties`** für den spezifischen Anwendungsfall nicht aus, so können manuell weitere Zuordnungen konfiguriert werden, so dass ein sehr vielseitiges Instrumentarium für Suchabfragen und suchgetriebene Anwendungen zur Verfügung steht.

Suchcenter – die „klassische" Variante:

Die Abfrage der Suche kann zum einen über SUCHCENTER erfolgen. Suchcenter können an verschiedenen Stellen innerhalb der logischen und inhaltlichen Struktur der SharePoint-Umgebung erstellt werden. Als Templates stehen ein Basissuchcenter und ein Unternehmenssuchcenter mit vielseitigen Anpassungsmöglichkeiten zur Verfügung.

Gerade wenn SharePoint-Inhalte für einen spezifischen Anwendungsfall strukturiert abgelegt werden (z.B. über Inhaltstypen) können mit einem Unternehmenssuchcenter bereits sehr vielfältige Filterungsoptionen (Refinements) über die hinterlegten Metadaten vorgegeben werden.

Ebenso vielfältig konfigurierbar ist die Ausgabe der Suchergebnisse. Dabei können die Suchergebnisseiten über Display-Templates exakt an den spezifischen Anwendungsfall angepasst werden. Neben der bei vielen Dokumenttypen möglichen Vorschaufunktion wären hier vor allem die gezielte Anzeige von einzelnen Metadaten-Informationen zu nennen, über die der Suchende schnell entscheiden kann, ob das Suchergebnis relevant ist oder nicht.

Suchgetriebene Anwendungen:

Im Gegensatz bzw. in Ergänzung zu gezielten Suchabfragen durch die Benutzer sind in SharePoint-Szenarien konfigurierbar, in denen der Benutzer Suchergebnisse automatisch beim Aufruf einer Seite angezeigt bekommt. Dabei kann es sich z.B. um gefilterte, aggregierte und bereits im Hintergrund weiterverarbeitete Daten aus verschiedenen Quellen innerhalb der SharePoint-Umgebung handeln, die im Rahmen von Webparts auf einer SharePoint-Seite angezeigt werden. Hierfür stellt SharePoint bereits eine Reihe von Webpart-Templates bereit, die für den spezifischen Anwendungsfall weiter konfiguriert werden können.

Darüber hinaus besteht die Möglichkeit, mittels APIs auch programmatisch auf die Suchumgebung zuzugreifen, sodass sehr vielfältige Anwendungsszenarien realisierbar sind.

Security-Aspekte bei der Suche

Ein weiteres Merkmal der SharePoint-Suche ist das Security-Trimming, wodurch jedem Benutzer nur die Suchergebnisse angezeigt werden, für die er auch berechtigt ist. Hierin liegt auch die Hauptproblematik bei der Anbindung externer Datenquellen, da die Angleichung der Security-Kontexte u.U. einen beträchtlichen Anpassungs-

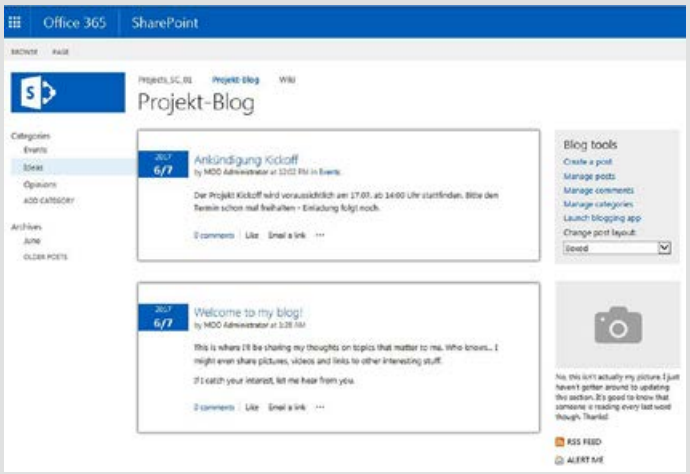


Abb. 2: Beispiel für eine Blog-Seite im Projekt-Teamroom

aufwand mit sich bringt. Bleibt man jedoch bei den zu durchsuchenden Komponenten im Microsoft-Umfeld, so bietet die SharePoint sehr viele Möglichkeiten, die gerade auch im Kollaborationsbereich sehr zielgerichtet genutzt werden sollen.

Suchqualität

Bei all den technischen Möglichkeiten, die die SharePoint-Suche bietet, darf nicht außer Acht gelassen werden, dass die Qualität der Suche sehr stark abhängig ist von der Qualität der Daten, die in SharePoint abgelegt werden. Grundsätzlich gilt, je gezielter Daten in Form von Metadaten und Inhaltstypen strukturiert werden, umso anforderungsspezifischer und gezielter sind diese über Suchanpassungen abfragbar. Auch hier werden die besten Ergebnisse über sorgfältige Planung im Vorfeld erzielt.

Gezielte Suche vs. machine learning - Delve und Office Graph

Bisher wurde nur die strukturierte Suche betrachtet und welche Mechanismen in SharePoint hierfür nutzbar sind. Wie sieht es aber aus mit Mechanismen, die die Informationsflut nach anderen Kriterien vorfiltern, z.B. anhand des Benutzerverhaltens und anderer aus dem Social Media-Bereich bekannter Kriterien?

Microsoft treibt die Entwicklung in diesem Bereich vor allem in der Cloud voran: Beim Einsatz von Office 365 wird das Spektrum der Suchmöglichkeiten nochmals erweitert, indem mit MICROSOFT OFFICE GRAPH als zugrundeliegender Technologie und DELVE als Oberfläche Funktionalitäten angeboten werden, die Informationen abhängig vom Benutzerverhalten anzeigen. OFFICE GRAPH, ein selbstlernender Algorithmus, vernetzt dabei innerhalb von Office 365 Informationen aus verschiedenen Quellen und stellt diese dem Anwender zur Verfügung. Dabei ist der Fokus weiter gespannt, indem die Interaktionen zwischen Personen und generierten Inhalten abgebildet werden und - je nach Datenschutz-Einstellung - neben Dateien auch Personen und Aktivitäten betrachtet werden.

Werden diese selbstlernenden Komponenten, die aktuell allerdings nur in Cloud- bzw. Hybrid-Szenarien zur Verfügung stehen, mit den gezielten Suchmöglichkeiten der SharePoint-Suche kombiniert, so ergeben sich sehr leistungsfähige Optionen für das Informationsmanagement – nicht nur in Projekten.

Teil 2: Kommunikation

Einige der Anforderungen an die Kommunikation im Projekt wurden bereits in dem vorangegangenen Artikel [1] angesprochen. Geht man davon aus, dass Projektkommunikation schnell und unkompliziert und doch verbindlich und nachvollziehbar erfolgen muss, dass unter Umständen das Projektteam flexibel um Spezialisten auch von extern erweitert werden muss und dass zum direkten Informationsaustausch Werkzeuge wie Chats oder im Idealfall auch Videokonferenzsysteme zur Verfügung stehen sollten, so stellt sich die Frage, inwieweit SharePoint dieses Anwendungsprofil abdecken kann.

Kommunikationstools in SharePoint

SharePoint stellt Tools und Templates zur direkten Kommunikation in den Teamrooms zur Verfügung, z.B. Blogs, Umfragen und Community-Templates (siehe Abbildung 2).

Blog-Seiten können in der Projektarbeit einen Kompromiss darstellen, zwischen „informeller Kommunikation“ als Alternative zum Mailverkehr und dennoch dokumentiertem Informationsaustausch. Dabei bietet sich die Möglichkeit, mit Kategorien zu arbeiten, so dass die Inhalte auch strukturiert und Foren erstellt werden können. Erweiterte Möglichkeiten in dieser Richtung bieten die Community-Templates, die ebenfalls in einem Projekt-Teamroom bereitgestellt werden können.

Weitere Kommunikationstools innerhalb der Office-365-Produktfamilie

Die Palette der Möglichkeiten wird durch die Einbettung in die Office-365-Produktfamilie noch beträchtlich erweitert. SharePoint stellt dabei eine Art Bindeglied zwischen den neuen Collaboration Tools der Office 365-Familie (z.B. „Teams“, „Skype for Business“, „Yammer“, „Planner“) dar. Wo die Social Features, die zwar mit der Version 2016 erneut weiter ausgebaut wurden, nicht ausreichen, können diese durch Kombination mit anderen Office-365-Diensten ergänzt werden. Beispielhaft kann hier die Verbindung mit einem der neuen Produkte aus der Office-365-Familie angeführt werden, mit „Teams“ (siehe Abbildung 3):

Hierbei handelt es sich um ein Chat-basiertes Tool, das je nach Anwendungsfall ebenfalls für die Projektkommunikation genutzt werden kann und in das beispielsweise Videokonferenz-Optionen via Skype for Business bereits integriert sind. Für das Dokumentenmanagement ist ein Konnektor auf eine bestehende SharePoint-Teamsite integriert, so dass beides möglich ist: eine leicht handhabbare Oberfläche zur Ad-Hoc-Kommunikation verbunden mit strukturierter Dokumentenablage.

In Projektszenarien, in denen vor allem ein spontaner Austausch zwischen Projektmitarbeitern (bzw. das spontane Hinzuziehen von Mitarbeitern/Experten auch außerhalb des Projekts) über verschiedene Kanäle (Chat, Videokonferenz, Blog) gefordert ist, kann SharePoint in Verbindung mit weiteren Office-365-Tools dem einzelnen Mitarbeiter auch viel Spielraum zum Teilen von Informationen lassen. Hier spielen insbesondere Tools wie „Teams“ oder „Yammer“ ihre Stärken aus. Auf diese Weise spielt sich die Kommunikation immer noch in einem für das Projekt vorgesehenen Rahmen ab (nicht über private WhatsApp-Gruppen etc.) und umfasst einen definierten Teilnehmerkreis (Mitarbeiter mit Firmen-Accounts).

Adi Schmid
(info@ordix.de)

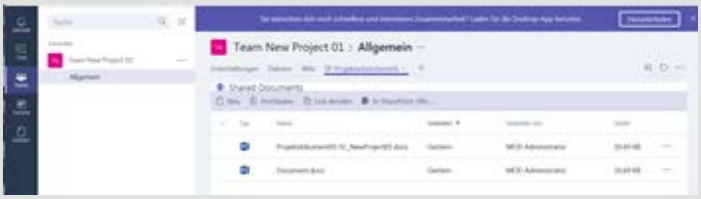


Abb. 3: „Teams“ mit Verlinkung auf die Dokumentenbibliothek des SharePoint-Projektarbeitsbereichs

Links

[1] Microsoft SharePoint (Teil I):
Plattform für das Informationsmanagement in Projekten
https://www.ordix.de/ordix-news-archiv/1-2018/

SO KÖNNTE IHR SEMINARRAUM AUSSEHEN!

PASSGENAUE SEMINARE IN INDIVIDUELLER UMGEBUNG

Neben der Digitalisierung, geänderten Geschäftsprozessen und vielen tiefgreifenden Veränderungen, die damit verbunden sind, ist eine stetige Weiterbildung Pflicht.

Wissen sichert Unternehmern und Mitarbeitern gleichermaßen Wettbewerbsvorteile. Deshalb wird die kontinuierliche Weiterbildung der Mitarbeiter in Zukunft immer stärker über den wirtschaftlichen Erfolg von Unternehmen entscheiden.

Lernen verändert sich. Wir geben Ihnen mit virtuellen Klassenräumen die Möglichkeit, Ihre Mitarbeiter kosten- und zeitsparend weiterzuentwickeln.

Seminarteilnehmer verschiedener Standorte kommen im virtuellen Klassenraum als Lerngruppe zusammen und haben die Gelegenheit, zeitgleich wie in einem richtigen Seminarraum zu lernen und miteinander in Echtzeit zu kommunizieren.

Sichern Sie sich Ihr Wissen unter:
seminare.ordix.de

ORDIX® seminare
einfach. gut. geschult.