Die Oracle-Datenbank jenseits von Entity-Relationship-Modellierung – vorhandenes Wissen repräsentieren und neues mit RDF Graph generieren

Karin Patenge, ORACLE Deutschland B.V. & Co. KG

Die Oracle-Datenbank vereint in einzigartiger Weise unterschiedliche Datenmodelle. So finden relationale Daten neben JSON- oder XML-Dokumenten ihren Platz ebenso wie raumbezogene Daten oder unstrukturierte Inhalte wie Multi-Media-Dateien. Sogar die spaltenbasierte Datenhaltung im Arbeitsspeicher wird seit der Einführung der In-Memory-Option mit Version 12.1.0.2 durch die Datenbank unterstützt. Weniger bekannt, aber dafür im Informationszeitalter umso spannender ist, dass auch ein Datenmodell speziell für Begriffe („Ressourcen“) und deren Beziehungen („Relationships“) untereinander vorgehalten wird. Termini, die in diesem Zusammenhang immer wieder fallen, sind „Semantische Netze“, „Ressource Description Framework“ (RDF), „Linked Data“ oder „Ontologien“. Die Datenbank stellt dafür das Feature „RDF Graph“ innerhalb der Option „Spatial and Graph“ bereit.

Wissens- und Informations-Zusammenhänge zu repräsentieren, bewegt sich sehr häufig außerhalb der Grenzen einer relationalen Datenmodellierung. Denn in einem Entity-Relationship-Modell (ER) müssen Entitäten, deren Attribute und die Beziehungen zwischen den Entitäten vorab definiert werden. In den weiteren Modellierungsschritten werden daraus Tabellen, Spalten mit zugeordneten Datentypen und Foreign Key Constraints, die über ein SQL-Skript in der jeweiligen Datenbank angelegt sind. Änderungen an der Applikation, die darauf abzielen, zusätzliche Informationen oder Zusammenhänge aufzunehmen, müssen im Datenmodell relativ aufwändig nachgezogen werden.

Eine Alternative ist die Repräsentation von Informationen und Informations-Zusammenhängen als Graphen in einem Knoten- und Kantenmodell. Die Knoten stellen dabei Begriffe oder beliebige Ressourcen dar. Über die Kanten werden Zusammenhänge zwischen diesen abgebildet. *Abbildung 1* veranschaulicht das Graphenmodell in Form eines einfachen Anwendungsbeispiels, eines Familienverbands. Die Knoten sind die einzelnen Mitglieder, die Kanten zeigen, wie sie miteinander verwandt sind.

Die Oracle-Datenbank als Triple Store

Knoten und Kanten sind in der Datenbank als sogenannte „Triples“ gespeichert. Das sind Aussagen, bestehend aus Subjekt („Subject“), Prädikat („Property“) und Objekt („Object“), für die der Datentyp „SDO_TRIPLE_RDF_S“ zum Einsatz kommt *(siehe Listing 1)*.

Eine zu speichernde Aussage gemäß *Abbildung 1* ist nun „Maria (s) isMotherOf (p) Mia (o)“. Für die einzelnen Triple-Teile gilt:

```
-- Table with triples
create table family_rdf_data (
   id integer generated as identity (start with 1 increment by 1) primary key,
   triple sdo_rdf_triple_s
)
/

-- Create model
execute sem_apis.create_sem_model(
   'family',             -- Model name
   'family_rdf_data',    -- Table name
   'triple'              -- Column name
)
/
```

Listing 1: Anlegen einer Triple-Tabelle sowie Registrieren eines semantischen Modells (Graph)

- Ein Subjekt (s) ist entweder ein Uniform Resource Identifier (URI) oder ein leerer Knoten („blank node“)
- Ein Prädikat (p) ist immer ein URI
- Ein Objekt (o) kann ein URI, ein Literal oder auch ein leerer Knoten sein

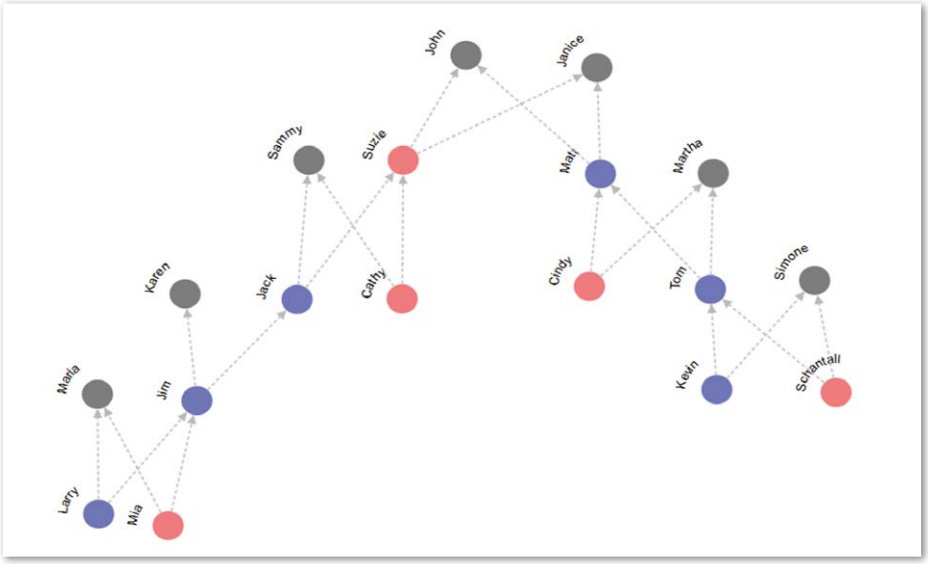

Abbildung 1: Ausschnitt aus einem Familienverband als Graphenmodell.

```
-- Maria ist die Mutter von Mia.
SQL> INSERT INTO family_rdf_data VALUES (
   SDO_RDF_TRIPLE_S(
   'family',
   '<http://www.example.org/family/Maria>',
   '<http://www.example.org/family/motherOf>',
   '<http://www.example.org/family/Mia>'))
/

-- Schantall ist (vom Typ) weiblich.
SQL> INSERT INTO family_rdf_data VALUES (
   SDO_RDF_TRIPLE_S(
   'family',
   '<http://www.example.org/family/Schantall>',
   '<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>',
   '<http://www.example.org/family/Female>'))
/

-- Cathy hat die Größe von 5.8 (Fuß).
SQL> INSERT INTO family_rdf_data VALUES (
   SDO_RDF_TRIPLE_S('family',
   '<http://www.example.org/family/Cathy>',
   '<http://www.example.org/family/height>',
   '"5.8"^^xsd:decimal'));

SQL> commit
/

-- Extrahieren der einzelnen Triple-Bestandteile aus der Spalte TRIPLE
SQL> select id,
   f.triple.get_model() m,       -- Semantisches Modell
   f.triple.get_subject() s,     -- Subjekt
   f.triple.get_property() p,    -- Prädikat
   f.triple.get_object() o       -- Objekt
from family_rdf_data f
/
```

Listing 2

Listing 2 zeigt beispielhaft drei Triples.

Neues Wissen generieren

Das Speichern von Triples ist aber nur die halbe Sache. Wirklichen Mehrwert bringt das regelbasierte Ableiten von neuen Informationen. Nehmen wir die zwei folgenden Aussagen an:

- Matt ist der Vater von Tom
- Tom ist der Vater von Kevin

Dann kann daraus geschlussfolgert werden, dass auch folgende Aussage gilt: Matt ist ein Großelternteil von Kevin. Diese Aussage selbst ist nicht explizit in den Triples enthalten, aber durch eine selbstdefinierte Regel ohne Weiteres ableitbar. Dazu wird zunächst eine sogenannte „Rulebase“ als Container für Regeln in Bezug auf das Family Model angelegt. Hinzu kommt dann eine GrandParent-Regel *(siehe Listing 3)*. *Listing 4* zeigt die Struktur jeder Rulebase.

Für die weitere Verwendung muss die Regelbasis noch indiziert werden. Es wird ein sogenanntes „Entailment“ angelegt *(siehe Listing 5)*. Nun ist der Moment gekommen, die Triples zu durchsuchen und daraus neue Informationen abzuleiten. Auf Basis der zuvor definierten Regel soll mit einem zusätzlichen Filter herausgefunden werden, wer Großvater welchen Enkelkinds ist. Dazu dient ein „SELECT“-Statement mit der Tabellen-Funktion „SEM_MATCH“ *(siehe Listing 6)*. *Tabelle 1* zeigt das Ergebnis in Tabellenform, *Abbildung 2* grafisch insgesamt neun abgeleitete Großvater-Enkelkind-Beziehungen.

Der Prozess des Ableitens von neuen Beziehungen heißt auch „Inferencing“. Weiterführende Funktionen in „Spatial and Graph“ verknüpfen das in Form von Triples gespeicherte Wissen mit relationalen Daten. Dazu kommt der „SEM_RELATED“-Operator zum Einsatz.

Ein ebenfalls sehr anschauliches Beispiel ist als „Oracle By Example“-Tutorial (OBE) in der „Oracle Learning Library“ *(siehe „http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/11g/r1/prod/datamgmt/nci_semantic_network/nci_Semantics_les01.htm“)* verfügbar. Es zeigt Schritt für Schritt, wie domänenspezifisches Wissen – konkret über onkologische und andere Erkrankungen – in Form von Triples modelliert und gespeichert

ist sowie mit Diagnosen in einer Patientenkartei verknüpft werden kann. Die Gesamtheit dieser Triples sind auch als Thesaurus des National Cancer Institute *(siehe „https://ncit.nci.nih.gov/ncitbrowser“)* bekannt *(siehe Listing 7)*. Ein anderer häufig verwendeter Begriff für domänenspezifisches Wissen ist „Ontologie“. Dazu in Beziehung gebracht, werden relational modellierte Daten über PatientInnen. Diese sind in der Tabelle „PATIENTS_DATA“ vorgehalten *(siehe Listing 8)*.

Mithilfe eines semantischen Index auf der „DIAGNOSIS“-Spalte lässt sich dann beispielsweise ermitteln, welche Patientin beziehungsweise welcher Patient an einer Unterklasse der Krankheit „Bone Sarcoma“ (Osteosarkom) leidet *(siehe Listing 9)*.

Die auf Daten anwendbaren Fragestellungen, die als Graphen modelliert sind, können vielfältiger Natur sein. Die wohl bekanntesten beschäftigen sich mit der Analyse von Beziehungen in sozialen Netzwerken:

1. Wer ist mit wem über wie viele Knoten vernetzt?
2. Wer ist die einflussreichste Person in einem Netzwerk?
3. Wo gibt es eine Häufung von Individuen mit ähnlichen Interessen?

Das sind drei der möglichen Fragestellungen. Antworten darauf geben Graphen-basierte Algorithmen.

Graphen-basierte Analysen jenseits einer relationalen Datenbank und SQL

Heutige und zukünftige IT-Architekturen orientieren sich immer mehr an immens und schnell wachsenden Datenmengen, die aus unterschiedlichsten Quellen zur Verfügung stehen. Es sind also flexible und skalierbare Architekturen gefragt, die in kurzen Zeiteinheiten und an unterschiedlichen Orten sehr viele Daten aufnehmen, filtern, prozessieren und analysieren können.

Das führt zu Architektur-Komponenten, die jenseits eines idealerweise einheitlichen Unternehmens-Data-Warehouse zu finden sind. Hier kommen zunehmend Technologien wie Hadoop, NoSQL oder Event Processing zum Einsatz, die häufig im Zusammenhang mit dem Thema „Big Data“ genannt werden.

```
SQL> exec sem_apis.create_rulebase('family_rb')
/

SQL> Insert into mdsys.semr_family_rb
  values(
  'grandparent_rule',
  '(?x :parentOf ?y) (?y :parentOf ?z)',
  NULL,
  '(?x :grandParentOf ?z)',
  SEM_ALIASES(
      SEM_ALIAS('','http://www.example.org/family/')
  )
  )
/
```

Listing 3: Rulebase anlegen und mit Regeln füllen

```
Name          Null       Type
-----------   --------   --------------
RULE_NAME     NOT NULL   VARCHAR2(30)
ANTECEDENTS              VARCHAR2(4000)
FILTER                   VARCHAR2(4000)
CONSEQUENTS NOT NULL     VARCHAR2(4000)
ALIASES                  RDF_ALIASES
```

Listing 4

```
exec sem_apis.drop_entailment('rdfs_rix_family')
/

SQL> begin
  sem_apis.create_entailment(
  'rdfs_rix_family',
  sem_models('family'),
  sem_rulesbases('RDFS','family_rb'));
end;
/

-- Statistiken
SQL> exec sem_apis.analyze_entailment('rdfs_rix_family')
/
```

Listing 5: Regelbasis indizieren

```
SQL> select x "Großvater", y "Enkelkind"
  from table(sem_match(
  '{?x :grandParentOf ?y . ?x rdf:type :Male}',
  sem_models('family'),
  sem_rulebases('RDFS','family_rb'),
  sem_aliases(sem_alias('','http://www.example.org/family/')),
  null))
/
```

Listing 6: Abfrage mit Inferencing

```
SQL> select count(*) from nci_rdf_data;

  COUNT(*)
---------
  408997
```

Listing 7: Anzahl der Triples im NCI-Thesaurus

```
describe patients_data
/
-- Ergebnis:
Name                          Null?      Type
----------------------------  --------   -------------------------
ID                            NOT NULL   NUMBER
NAME                                     VARCHAR2(35)
AGE                                      NUMBER
DIAGNOSIS                                VARCHAR2(200)
DIAG_LENGTH                              NUMBER
```

Listing 8: Triple-Tabelle für den NCI-Thesaurus

```
   select distinct diagnosis, sem_distance(123)
   from patients_data
   where sem_related(
       diagnosis,
       '<http://www.w3.org/2000/01/rdf-schema#subClassOf>',
       '<http://www.mindswap.org/2003/nciOncology.owl#Bone_Sarcoma>',
       sem_models('nci'),
       sem_rulebases('owlprime'), 123) = 1
order by sem_distance(123) asc
/
```

Listing 9: Semantische Analyse von Diagnosedaten im Zusammenhang mit dem NCI-Thesaurus

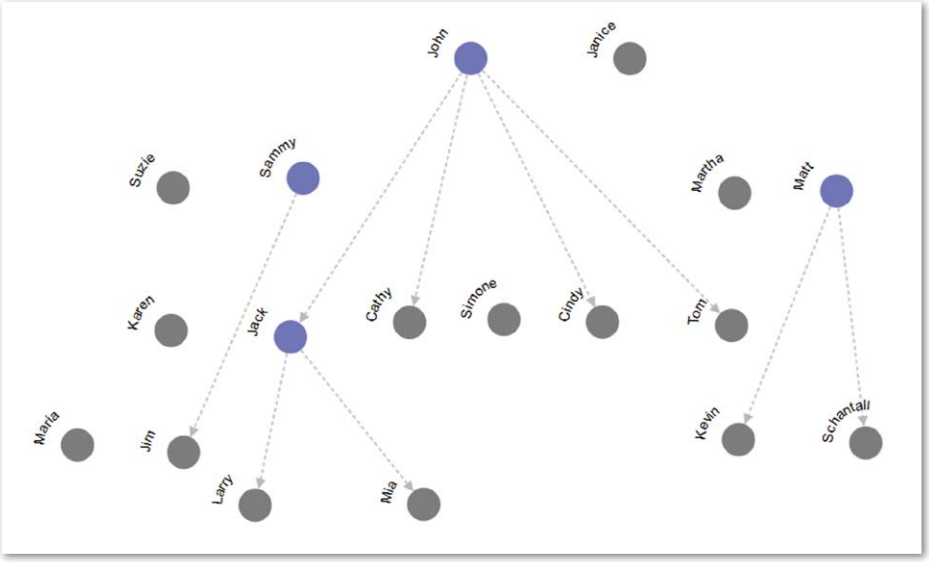

Abbildung 2: Abgeleitete Großvater-Enkelkind-Beziehungen

Großvater	Enkelkind
http://www.example.org/family/Jack	http://www.example.org/family/Larry
http://www.example.org/family/Jack	http://www.example.org/family/Mia
http://www.example.org/family/John	http://www.example.org/family/Cathy
http://www.example.org/family/John	http://www.example.org/family/Cindy
http://www.example.org/family/John	http://www.example.org/family/Jack
http://www.example.org/family/John	http://www.example.org/family/Tom
http://www.example.org/family/Matt	http://www.example.org/family/Kevin
http://www.example.org/family/Matt	http://www.example.org/family/Schantall
http://www.example.org/family/Sammy	http://www.example.org/family/Jim

Tabelle 1

Eine eigene Kategorie innerhalb der NoSQL-Datenbanken sind die sogenannten „Graph-Datenbanken". Oracle hat mit der NoSQL-Datenbank und deren Aufsatz „Graph" eine solche im Portfolio. Dieses Produkt ist zudem integraler Teil eines anderen Produkts – Oracle Big Data Spatial and Graph.

Die nachfolgenden Ausführungen beschreiben, wie in der Oracle-NoSQL-Datenbank ein Graph bestehend aus Knoten und Kanten mit zusätzlichen Attributen angelegt wird, die ein soziales Netzwerk abbilden, und wie dieses Netzwerk mit bereits vorgefertigten Funktionen im Hinblick auf die einflussreichsten Personen analysiert wird. Der für die Analyse verwendete Algorithmus ist „Page Rank". Er wird unter anderem verwendet, um die Link-Popularität einer Web-Ressource (ein Link, ein Dokument etc.) zu ermitteln.

Das Code-Beispiel wurde in Anlehnung an das Posting „Identifying Influencers with the Built-In Page Rank Analytics in Oracle Big Data Spatial and Graph" *(siehe „https://blogs.oracle.com/bigdataspatialgraph/entry/identifying_influencers_with_the_built")* im Blog „Adding Location and Graph Analysis to Big Data" erstellt und auf der Oracle Big Data Lite Virtual Machine v4.2.1 getestet. Dazu kommt eine für Graphen spezifische Skriptsprache zum Einsatz *(siehe Listing 10 und Abbildung 3)*.

Oracle Big Data Spatial and Graph besitzt eine Engine, die parallel und In-Memory Graph-Analysen wie beim Page Rank ausführt, und zwar sowohl auf der Oracle-NoSQL-Datenbank als auch auf Apache HBase. Grundlage dafür ist PGX, ein sogenanntes „Graph Analysis Framework" *(siehe „http://www.oracle.com/technetwork/oracle-labs/parallel-graph-analytics/overview/index.html")*, über das mehr als dreißig gängige Graphen-Algorithmen für die Bereiche „Classical", „Community Detection", „Path Finding", „Ranking", „Recommendation" und „Structure Evaluation" implementiert sind, die mit 107 und mehr Knoten und Kanten umgehen können. Anstelle von SQL wird über APIs wie Java, Gremlin oder Python auf die Daten zugegriffen.

Fazit

Die in ihrem Ursprung relationale Oracle-Datenbank hat sich zu einem multi-modalen Datenbanksystem entwickelt. Die Erweiterung als Graph-Datenbank, die Implementie-

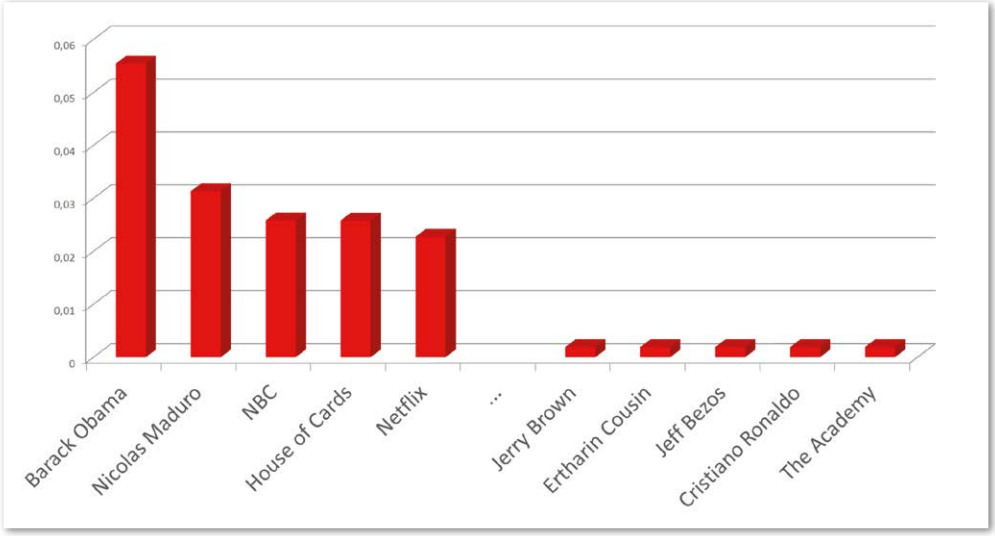

Abbildung 3: Ergebnis der Einfluss-Analyse im Netzwerk mit Page Rank

```
//
// Connect to the Oracle NoSQL Database
//
server = new ArrayList<String>();
server.add("bigdatalite:5000");
cfg = new PgNosqlGraphConfigBuilder()    \
 .setDbEngine(DbEngine.NOSQL)    \
 .setName("connections")            \
 .setStoreName("kvstore")           \
 .setHosts(server)          \
 .addEdgeProperty("lbl", PropertyType.STRING, "lbl") \
 .addEdgeProperty("weight", PropertyType.DOUBLE,
"1000000") \
 .setMaxNumConnections(1).build();

//
// Get an instance of OraclePropertyGraph
//
opg = OraclePropertyGraph.getInstance(cfg);
opg.clearRepository();

//
// Use the parallel data loader API to load a
// sample property graph in flat file format with
// a degree of parallelism (DOP) 2
//
vfile="../../data/connections.opv"
efile="../../data/connections.ope"
opgdl=OraclePropertyGraphDataLoader.getInstance();
opgdl.loadData(opg, vfile, efile, 2);

// Read through the vertices
opg.getVertices();

// Read through the edges
opg.getEdges();

//
// Use In-Memory Graph Analysis Framework to run
// the built-in analytical function, PageRank, to
identify influencers
//
```

Listing 10

```
analyst = opg.getInMemAnalyst();
rank = analyst.pagerank(0.0001, 0.85, 100).get();
// maximum error for terminating the iteration: e =
0.0001
// damping factor: d = 0.85
// maximum number of iterations: max = 100

rank.getTopKValues(5);

// Identify person at the given vertex
v1=opg.getVertex(11);
v2=opg.getVertex(601);
v3=opg.getVertex(421);

System.out.println("Top 3 influencers: \n " +
v1.getProperty("name") + \
      "\n " + v2.getProperty("name") +     \
      "\n " + v3.getProperty("name"));

// Top: Barack Obama
```

rung Objekt-relationaler Datentypen sowie die Unterstützung Dokument-zentrischer, semi-strukturierter Daten wie XML oder JSON sind zum Teil längst schon Realität und Bestandteil vieler Anwendungen.

SQL als Standard-Abfragesprache wurde und wird in geeigneter Weise erweitert, um direkt in der Datenbank auf all diese Daten in einheitlicher Art und Weise zuzugreifen und sie in ihren Zusammenhängen zu analysieren. Aber die Anforderungen und Technologien entwickeln sich weiter. Daher lohnt es sich, den eigenen Blick über die bekannten und bewährten Technologien im Umfeld von Data Warehouses hinaus zu richten. NoSQL, Hadoop & Co. lassen sich gut in bestehende IT-Infrastrukturen integrieren oder als Cloud Services nutzen. Bereits in naher Zukunft werden kaum mehr Wege daran vorbeiführen. Daher gilt es, sich bald das neue Wissen zu erschließen und es bestmöglich für das eigene Unternehmen einzusetzen.

Karin Patenge
karin.patenge@oracle.com
http://oracle-spatial.blogspot.com