Java-Bibliothek oracle.sdoapi

Der Zugriff auf SQL-Objekte ist ein wenig umständlich und sollte unbedingt in entsprechenden Klassen gekapselt werden. Zur Arbeitserleichterung stellt aber Oracle seit dem Release 8.1.6 eine Klassenbibliothek namens `oracle.sdoapi` zur Verfügung, die den Zugriff auf `SDO_GEOMETRY`-Objekte unterstützt. Sie kann bis Oracle Spatial 10.1 genutzt werden und wurde inzwischen von der Klassenbibliothek `oracle.spatial` abgelöst.

Datenmodell

Die Geometrie-Klassen von `sdoapi` sind im Paket `oracle.sdoapi.geom` zusammengeführt. Abbildung 1.1 stellt die entsprechende Vererbungshierarchie dar[1]. Sie ähnelt den Vererbungshierarchien des OGC-Simple-Feature-Modells und des Datenmodells von SQL/MM Spatial. Auf unterster Ebene finden wir die Objekte mit geradlinig verbundenen Eckpunkten. Die Klassen `CurveString` und `CurvePolygon` erlauben Kreisbögen als Verbindungen. Für deren Oberklassen `Curve` und `Surface` gibt es keine korrespondierenden `SDO_GEOMETRY`-Objekte. Für jede Klasse, die eine einzelne Geometrie repräsentiert, existiert eine entsprechende `Collection`-Klasse. `Geometry` ist die übergeordnete Geometrieklasse.

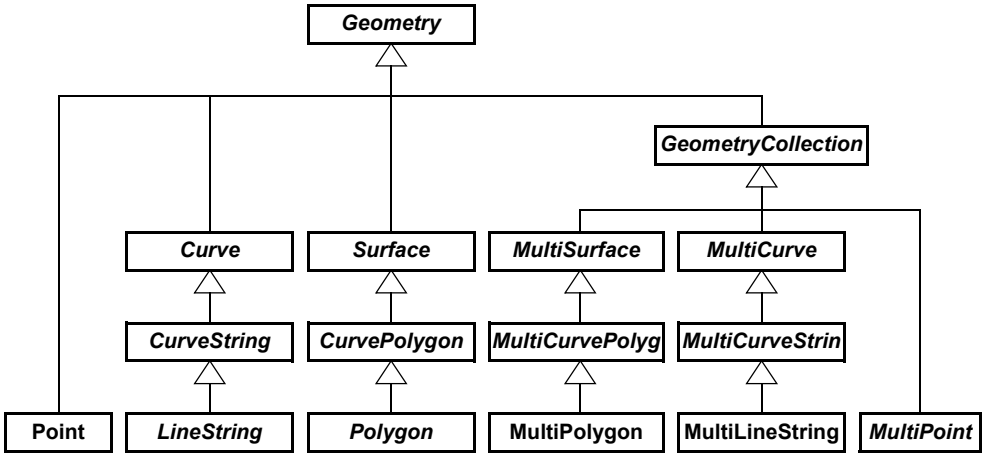

Abb. 1.1: Vererbungshierarchie im Paket `oracle.sdoapi.geom`.

Abbildung 1.2 zeigt eine Auswahl weiterer Beziehungen. Objekte der Typen `CurvePolygon` bzw. `Polygon` bestehen jeweils aus einem äußeren und optional aus mehreren inneren Ringen. Für die Beschreibung von Geometrien gibt es zwei unterschiedliche Klassen: Während Objekte vom Typ `CoordPoint` zwei- oder dreidimensionale Punkte repräsentieren, stellen Segmente die Verbindung zwischen zwei Punkten dar. Als Spezialisierung von `Segment` existieren `LinearSegment` für geradlinige Verbindungen und `CircularArc` für Kreisbögen. Für jede Geometrie repräsentiert `Envelope` das minimal umgebende Rechteck.

[1] Auch hier handelt es sich eigentlich um Schnittstellen, die durch interne Klassen aus dem Paket `oracle.sdoapi.geom.impl` implementiert werden.

Das räumliche Bezugssystem `SpatialReference` stammt aus dem Paket `oracle.sdoapi.ref`. Die Klassen des Pakets `oracle.sdoapi.geom` stellen jeweils Zugriffsmethoden auf die einzelnen Eigenschaften und Bestandteile der Objekte zur Verfügung.

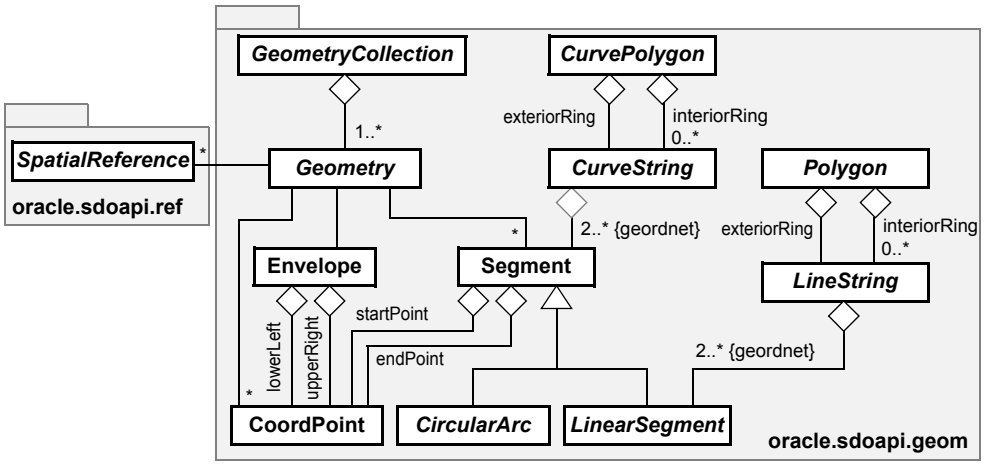

Abb. 1.2: Ausgewählte Beziehungen in `oracle.sdoapi`.

Schnittstelle GeometryAdapter

Für die Umwandlung zwischen der Datenbankrepräsentation als `STRUCT` und der jeweiligen Geometrieklasse aus `oracle.sdoapi.geom` ist ein Objekt zuständig, das die Schnittstelle `GeometryAdapter` aus dem Paket `oracle.sdoapi.adapter` erfüllt. Für die Umwandlung gibt es die Methoden `importGeometry`, um ein `Geometry`-Objekt zu erhalten, und `exportGeometry`, um ein `Geometry`-Objekt zurück zu konvertieren. Da die Schnittstelle `GeometryAdaper` nicht nur die Umwandlung zwischen Datenbankrepräsentationen und Objekten aus `oracle.sdoapi.geom` spezifiziert, sondern anstelle der Datenbankdarstellung auch andere Repräsentationen unterstützen soll, ist der jeweils andere Parameter nicht vom Typ `STRUCT`, sondern ein allgemeines Java-`Object`. Deswegen wird bei `exportGeometry` zusätzlich über einen Parameter vom Typ `Class` die gewünschte Zielklasse angegeben. Beide Methoden werfen Ausnahmen, wenn die Geometrie fehlerhaft ist oder ein nicht unterstützter Ausgabe- bzw. Eingabetyp übergeben wurde:

```
Object exportGeometry (Class outputType, Geometry geom)
   throws GeometryOutputTypeNotSupportedException, InvalidGeometryException

Geometry importGeometry (Object inputData)
   throws GeometryInputTypeNotSupportedException, InvalidGeometryException
```

Da der `GeometryAdapter` nur eine Schnittstelle ist, stellt sich die Frage, wie man ein Objekt erhält, das diese Schnittstelle erfüllt. Dazu dient die Klasse `OraSpatialManager`, die einige Adapter bereithält und bei der sich weitere Adapter registrieren lassen können. Die Klassenmethode `getGeometryAdapter` stellt einen spezifischen Adapter zur Verfügung:

```
static GeometryAdapter getGeometryAdapter (
  String formatName,            // Umwandlungsformat "SDO"
  String formatVersion,         // "8.1.6"
  Class inputType,              // Eingabeklasse bei importGeometry
  Class outputType,             // Ausgabeklasse bei exportGeometry
  Class passthroughOutputType,  // null
  Connection conn)              // Datenbankverbindung
```

Der Parameter `formatName` enthält den Namen des Umwandlungsformats, der für die Datenbankrepräsentationen von Oracle Spatial „SDO“ lautet. Der Parameter `formatVersion` gibt die Version dieser Datenrepräsentation an. Da diese sich seit dem Oracle-Release 8.1.6 nicht verändert hat, ist hier zurzeit „8.1.6“ anzugeben. Die Parameter `inputType` und `outputType` geben die Klasse an, aus der bzw. in die ein `Geometry`-Objekt umgewandelt werden soll. Dies ist `oracle.sql.STRUCT.class`. Den fünften Parameter `passthroughOutputType` können wir mit `null` ignorieren. Als letztes Argument ist die Datenbankverbindung zu übergeben. Sollte kein passender Adapter vorliegen, wird von der Methode `null` zurückgegeben.

Lesen von Geodaten

Mit diesen Informationen sind wir nun in der Lage, Geodaten aus der Datenbank in ein Java-Programm einzulesen. Dazu müssen die notwendigen Klassen von `sdoapi` importiert und mögliche Ausnahmen abgefangen werden. Nachdem der `GeometryAdapter` vorliegt, können die angefragten Geometrien umgewandelt und mit den entsprechenden Methoden auf deren Bestandteile zugegriffen werden.

```
// Einlesen von Geoobjekten mit Hilfe des Paketes oracle.sdoapi
import java.sql.*;           // Import der JDBC-Klassen
import oracle.sql.STRUCT;    // Import von STRUCT
// Import der Klassen des SDO-API:
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.GeometryAdapter;
import oracle.sdoapi.geom.*;

public class SdoapiReadExample {

  public static void main (String[] args) {
    try {
      // Verbindung zur Datenbank aufbauen
      Class.forName("oracle.jdbc.driver.OracleDriver");
      String url = "jdbc:oracle:thin:@"+DBHOST+":"+DBPORT+":"+DBNAME;
      Connection db = DriverManager.getConnection(url, DBUSER, DBPASSWORD);

      // GeometryAdapter holen
      GeometryAdapter adapter = OraSpatialManager.getGeometryAdapter
                              ("SDO", "8.1.6", STRUCT.class, null, null, db);
      if (adapter == null) {
        db.close();
        throw new ClassNotFoundException("Kein GeometryAdapter!");
      }

      // Anfrage ausführen
      Statement statement = db.createStatement();
      String sql = "SELECT * FROM Gemeinden WHERE gkz = 3403000";
      ResultSet result = statement.executeQuery(sql);
```

```
      if (result.next()) {
        System.out.print(result.getString("name") + " - ");
        Point centrum =
                (Point)adapter.importGeometry(result.getObject("centrum"));
        System.out.println(centrum.getX() + " / " + centrum.getY());
        Polygon pol =
               (Polygon)adapter.importGeometry(result.getObject("gebiet"));
        CurveString outerRing = pol.getExteriorRing();
        System.out.println("#Eckpunkte: " + outerRing.getNumPoints());
      }
      result.close();
      statement.close();
      // Verbindung schließen
      db.close();
    } // try
    // Fehlerbehandlung
    catch (Exception ex) {
      System.err.println("SdoapiReadExample.main: " + ex);
    }
  } // main
} // class
```

Für die Übersetzung und die Ausführung des Programms muss nun zusätzlich die Klassenbibliothek `sdoapi` (hier in der Datei `sdoapi.zip`) zugreifbar sein:

```
rem  Java-Datei übersetzen:
javac -classpath .;./lib/ojdbc14.jar;./lib/sdoapi.zip SdoapiReadExample.java
rem  Programm ausführen:
java -classpath .;./lib/ojdbc14.jar;./lib/sdoapi.zip SdoapiReadExample
```

Das Programm erzeugt dann folgende Ausgaben:

```
Oldenburg - 8.22261313155162 / 53.1468217508227
Anzahl der Eckpunkte: 272
```

Abspeichern von Geodaten

Um Geodaten in der Datenbank abspeichern zu können, werden Objekte benötigt, die die Schnittstelle `Geometry` erfüllen. Diese können wir entweder dadurch erhalten, dass wir die Objekte zuvor aus der Datenbank einlesen, oder indem wir neue Objekte erzeugen. Für den zweiten Ansatz dient eine `GeometryFactory` (aus dem Paket `oracle.sdoapi.geom`), die man über die Klassenmethode `getGeometryFactory` der Klasse `OraSpatialManager` erhält:

```
static GeometryFactory getGeometryFactory ()
static GeometryFactory getGeometryFactory (SpatialReference spatialRef)
```

Die erste Variante liefert eine `GeometryFactory`, die eingesetzt werden kann, falls kein räumliches Bezugssystem verwendet wird, während man bei der zweiten Form das gewünschte Bezugssystem angibt. Die `GeometryFactory` stellt eine Reihe von Methoden zum Erzeugen neuer Objekte der `Geometry`-Klassenhierarchie zur Verfügung.

Ein räumliches Bezugssystem erhält man über den `SRManager`, dessen Methode `retrieve` für eine angegebene Bezugssystemnummer ein entsprechendes `SpatialReference`-Objekt

bestimmt (oder `null` zurückgibt, falls es kein Bezugssystem mit der angegebenen Nummer in der Datenbank gibt). Der `SRManager` stammt wie `SpatialReference` aus dem Paket `oracle.sdoapi.ref` und wird durch eine Methode der Klasse `OraSpatialManager` bereitgestellt:

```
static SRManager getSpatialReferenceManager (Connection conn)
```

Damit kann eine neue Gemeinde wie folgt in die Datenbank einfügt werden:

```
// Erzeugen und Speichern von Geoobjekten mit Hilfe des Paketes oracle.sdoapi
import java.sql.*;           // Import der JDBC-Klassen
import oracle.sql.STRUCT;    // Import von STRUCT

// Import der Klassen des SDO-API:
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.GeometryAdapter;
import oracle.sdoapi.geom.*;
import oracle.sdoapi.sref.SpatialReference;
import oracle.sdoapi.sref.SRManager;

public class SdoapiStoreExample {

  public static void main (String[] args) {
    try {
      // Verbindung zur Datenbank aufbauen
      Class.forName("oracle.jdbc.driver.OracleDriver");
      String url = "jdbc:oracle:thin:@"+DBHOST+":"+DBPORT+":"+DBNAME;
      Connection db = DriverManager.getConnection(url, DBUSER, DBPASSWORD);

      // GeometryAdapter holen
      GeometryAdapter adapter = OraSpatialManager.getGeometryAdapter
                     ("SDO", "8.1.6", STRUCT.class, STRUCT.class, null, db);
      if (adapter == null) {
        db.close();
        throw new ClassNotFoundException("Kein GeometryAdapter!");
      }
      // Bezugssystem holen
      SRManager srm = OraSpatialManager.getSpatialReferenceManager(db);
      SpatialReference cs = srm.retrieve(8307);  // alte SRID
      if (cs == null) {
        db.close();
        throw new ClassNotFoundException("Keine SpatialReference!");
      }
      // GeometryFactory holen
      GeometryFactory gf = OraSpatialManager.getGeometryFactory(cs);

      // Insert vorbereiten
      String sql = "INSERT INTO Gemeinden (gkz,name,einw,centrum,gebiet) " +
                   "VALUES (1056025,'Helgoland',1499,?,?)";
      PreparedStatement insert = db.prepareStatement(sql);

      // Punkt setzen
      Point centrum = gf.createPoint(7.936,54.204);
      Object obj = adapter.exportGeometry(STRUCT.class,centrum);
      insert.setObject (1,obj);

      // Gebiet setzen
      double coords[] = {7.923,54.210, 7.944,54.194, 7.952,54.197,
                         7.945,54.209, 7.938,54.211, 7.923,54.210};
```

```
      LineString ring = gf.createLineString(coords);
      Polygon gebiet = gf.createPolygon(ring,null);   // null, da keine Löcher
      obj = adapter.exportGeometry(STRUCT.class,gebiet);
      insert.setObject (2,obj);

      // Einfügen
      insert.executeUpdate();
      insert.close();

      // Verbindung schließen
      db.close();
    } // try

    // Fehlerbehandlung
    catch (Exception ex) {
      System.err.println("SdoapiStoreExample.main: " + ex);
    }
  } // main
} // class
```

Nach Übersetzung und Ausführung ist die Gemeinde „Helgoland“, die zum Bundesland Schleswig-Holstein gehört, neu in der Datenbank gespeichert:

```
SELECT gkz, name, einw FROM Gemeinden WHERE name = 'Helgoland';
       GKZ NAME                           EINW
---------- ------------------------------ ----------
   1056025 Helgoland                      1499

SELECT centrum FROM Gemeinden WHERE name = 'Helgoland';
CENTRUM
--------------------------------------------------------------------------------
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(7,936, 54,204, NULL), NULL,NULL)

SELECT gebiet FROM Gemeinden WHERE name = 'Helgoland';
GEBIET
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, 8307, NULL, SDO_ELEM_INFO_ARRAY(1,1003,1),
   SDO_ORDINATE_ARRAY(7,923, 54,21, 7,944, 54,194, 7,952, 54,197, 7,945,
                      54,209, 7,938, 54,211, 7,923, 54,21))
```