Herstellung von Oracle Spatial Networks und Routing mit Oracle Spatial 11gR2

Annemarie Becher / Erik Lipski

grit GmbH

Werne - Olpe – Berlin

Agenda

- Einleitung
- Projektdarstellung
- Herstellung von routingfähigen Netzwerken
- Oracle 11gR2 Routing Funktionen
- Fazit

Profile der grit GmbH

- Seit mehr als 20 Jahren im GIS-Bereich tätig
- Unterstützung zahlreicher Kunden aus Verwaltung und Wirtschaft
  - Land Berlin, Freistaat Thüringen, Freistaat Sachsen usw.
  - Leitungsbetreiber in ganz Deutschland
- Beratung in den Bereichen
  - GDI/INSPIRE-Infrastrukturen und Web-Applikationen
  - AFIS, ATKIS und ALKIS (AAA)
  - Homogenisierung und Qualitätssicherung
  - Softwareentwicklung

Projektdarstellung

- Einbindung von
  - Wanderwegen
  - Point Of Interest (POI)

  In ein routingfähiges Netzwerk auf Grundlage von NAVTEQ - Daten
- Berechnung und Präsentation von optimierten Routen in einem WebClienten

Hardware / Software

Datenbank DMZ

DB – Server

- Oracle DB 11gR2

Oracle Enterprise Linux 5.x

Applikation DMZ

APP – Server

- Apache WebServer
- Apache Tomcat
  - Java SDO API
  - WebClient
  - OGC Service

---

64 Bit Linux

Datengrundlage

- NAVTEQ Daten im Shape Format
  - Sachdaten im Shape (DBF – Datei)
  - Shapes: streets und zlevels
- Wanderwege
  - Alphanumerisch Sachdaten in Excel
  - Graphische Koordinaten in CSV
- Point of Interest (POI)
  - Naturaussichtspunkte, Naturschutzpunkte o.Ä.
  - Wie bei Wanderwegen

NAVTEQ - Datenformate

- Wie wurden die Shape Daten übernommen?
  - Mittels „shp2sdo“
- Redundante Daten aus Shape in Oracle Spatial übernommen
- Aufbereitung der eingespielten Shape – Daten
  - Doppelte Punkte löschen
  - Straßen ohne Nodes löschen
- Einbindung der Daten ins Oracle Routing Datenmodel
  - Tabellen
  - Metadaten

Untersuchte Oracle Funktionen

- PL/SQL Funktionen zur Herstellung / Aufbereitung der Daten zur Erstellung eines routingfähigen Netzes
  - SDO_LRS / SDO_GEOM
  - SDO_NET
- Java 11gR2 und Java API 10g
  - Stabilität
  - Performance
  - Speichernutzung

Datenmodel SDO_NET

Node Table

Node_ID
- N1
- N2
- …
- …
- Nn

Link Table

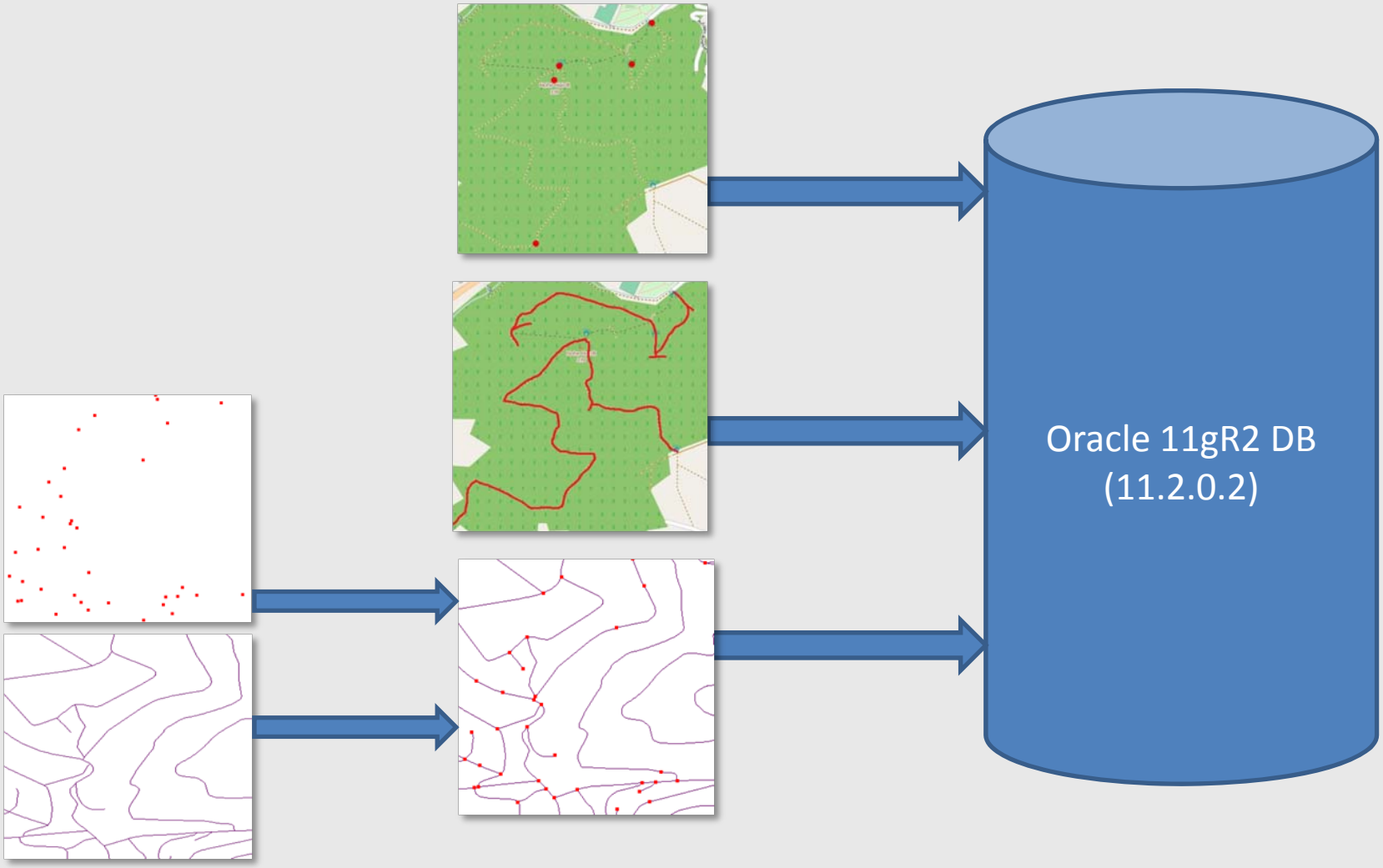
Link_ID,	Start_node_id,	End_node_id,	Cost , etc.
L1	N2	N1	15,457
L2	N2	N3	21,151
…			
…			
Ln	Nn	N1	182,123

Datenmodel SDO_NET

- Wichtig
  - SDO_NET.CREATE_PATH_TABLE('NET_001_PATHS', 'PATH_GEOMETRY');
  - SDO_NET.CREATE_PATH_LINK_TABLE('NET_001_LINKS');
  - SDO_NET.CREATE_PARTITION_TABLE('NET_001_PART_TAB');

- Optional
  - SDO_NET.CREATE_SUBPATH_TABLE('NET_001_SUBPATH_TAB', 'SUBPATH_GEOMETRY');

Herstellung routingfähiger Netzwerke



Herstellung routingfähiger Netzwerke

- **Filterung von NAVTEQ- Daten anhand der Klassifizierung**
  - 1 = Autobahnen oder Kreuze
  - 2 = Bundesstrassen und Autobahnen,
  - 3 = Bundesstrassen, Landstrassen und Autobahnen,
  - 4 = Kreisstrasse und Landstrassen, Bundesstrasse,
  - **5 = Strassen,**
  - **6 = Strasse, Weg,**
  - **7 = Strasse, Weg ,**
  - **8 = Nicht definiert**

Herstellung routingfähiger Netzwerke

- Verwendete SDO_GEOM Funktionen
  - SDO_INTERSECTION, SDO_BUFFER, SDO_DISTANCE, SDO_LENGTH usw.
- Verwendete SDO_LRS Funktionen
  - CONVERT_TO_LRS_GEOM, CONVERT_TO_STD_GEOM
  - GET_MEASURE, MEASURE_RANGE, PROJECT_PT, und FIND_OFFSET
  - CLIP_GEOM_SEGMENT, GEOM_SEGMENT_START_PT, GEOM_SEGMENT_END_PT
  - usw.

Metadaten anlegen

```
INSERT INTO USER_SDO_NETWORK_METADATA
(
  NETWORK, NETWORK_CATEGORY, GEOMETRY_TYPE,   NO_OF_HIERARCHY_LEVELS,
     NO_OF_PARTITIONS,
  LINK_DIRECTION,  NODE_TABLE_NAME, NODE_GEOM_COLUMN, NODE_COST_COLUMN,
  LINK_TABLE_NAME, LINK_GEOM_COLUMN, LINK_COST_COLUMN,
  PATH_TABLE_NAME, PATH_GEOM_COLUMN, PATH_LINK_TABLE_NAME,
     PARTITION_TABLE_NAME,
     SUBPATH_TABLE_NAME,SUBPATH_GEOM_COLUMN
)
 VALUES
(
  'NAV_001,  'SPATIAL',  'SDO_GEOMETRY‘,    1, 1, 'UNDIRECTED',
  'NET_NODES_001',  'LOCATION',  NULL,
  'NET_STREETS_001', 'GEOM',  'COST',
  'NET_001_PATHS', 'PATH_GEOMETRY', 'NET_001_LINKS',
     'NET_001_PART_TAB',
     'NET_001_SUBPATH_TAB','SUBPATH_GEOMETRY'
);
```

Validierung des Netzwerkes

- SDO_NET.VALIDATE_NETWORK('NAV_001')
- SDO_NET.VALIDATE_LINK_SCHEMA('NAV_001‘)
- SDO_NET.VALIDATE_NODE_SCHEMA('NAV_001‘)
- SDO_NET.VALIDATE_PARTITION_SCHEMA('NAV_001‘)
- SDO_NET.VALIDATE_PATH_SCHEMA('NAV_001‘)
  - True oder False

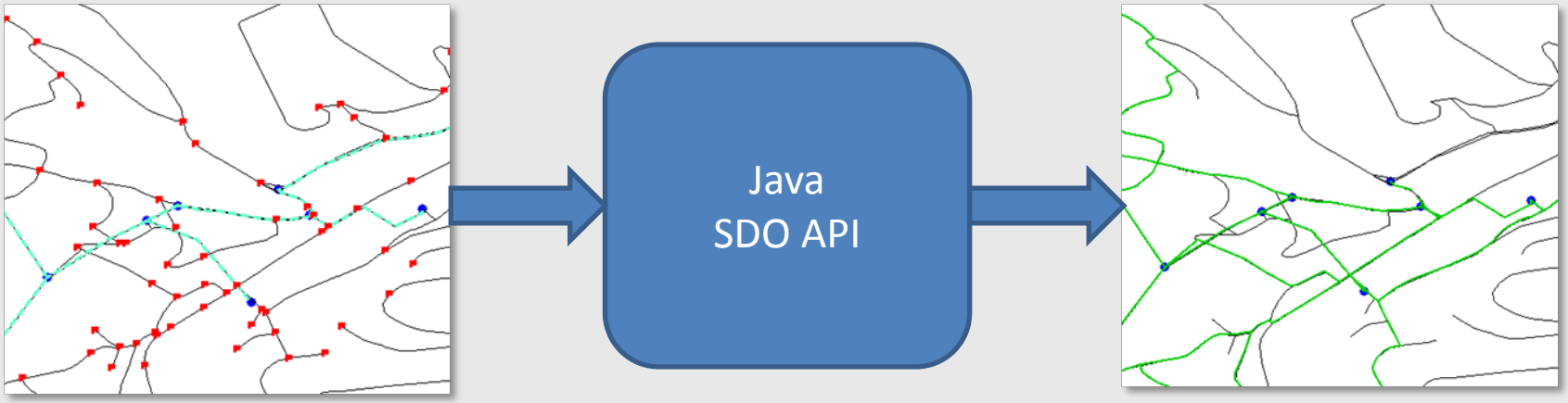
- SDO_NET.GET_ISOLATED_NODES('NAV_001')
  - Array mit Nodes_id

Validierung des Netzwerkes

- SDO_NET.GET_NO_OF_LINKS('NAV_001')
- SDO_NET.GET_NO_OF_NODES('NAV_001')
- SDO_NET.GET_NO_OF_HIERARCHY_LEVELS ('NAV_001')
  - Anzahl der Links, Nodes oder der Hierarchien im Netzwerk

- SDO_NET.GET_INVALID_LINKS('NAV_001')
- SDO_NET.GET_INVALID_NODES('NAV_001')
  - Array mit Link_id / Node_id

Routing



11gR2 - “Load on Demand” API

- Vorgehen
  - JDBC Datenbankverbindung aufbauen
  - Laden der Netzwerk Metadaten
  - LOD Network Manager
  - LOD Cached Network I/O
  - Network Analyst
    - Routing mit A*Star
    - Routing mit Dijkstra
    - Weitere Funktionen wie z.B. benachbarte Nodes
  - Ergebnisse als Spatial Geometrie abspeichern

Untersuchte Routing Algorithmen

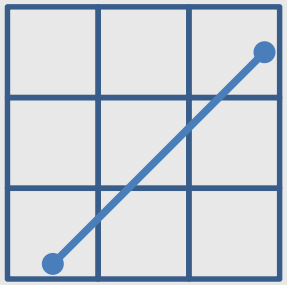
- NetworkAnalyst
  - shortestPathDijkstra
    - Berechnung des kürzesten Pfad zwischen einem Startnode und einem oder mehreren Nodes
    - LogicalSubPath subPath = analyst.shortestPathDijkstra(new PointOnNet(nodeStart), new PointOnNet(nodeEnd), null);
  - shortestPathAStar
    - Berechnung des kürzesten Pfad zwischen zwei Punkten
    - LogicalSubPath subPath = analyst.shortestPathAStar(new PointOnNet[]{new PointOnNet(nodeStart)}, new PointOnNet[]{ new PointOnNet(nodeEnd)}, null, null, new DummyLinkLevelSelector(1));

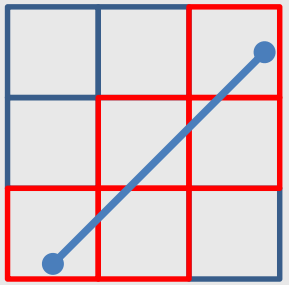
Ergebnisse / Performance

- Testgebiet
  - Links: 7.500
  - Nodes: 5.500
  - 347 Routing Vorgänge
- ShortestPathDijkstra Funktion
  - Routing aller Vorgänge 598 ms
  - Durchschnitt 1,7ms
- ShortestPathAStar Funktion
  - Routing aller Vorgänge 881 ms
  - Durchschnitt 2,5ms

Kommunikation

- zwischen Oracle DB 11gR2 (11.2.0.2) und Java Routing API
- Was wird übertragen?
  - Alle Daten aus den angeschnittenen Netzwerk - Partitionen





- Welche Datenmengen?
  - Hängt von der Größe der Netzwerk - Partitionen ab
    - Links und Nodes, aber ohne Spatial Geometrie
    - Userdata für dynamische Kostenberechnung

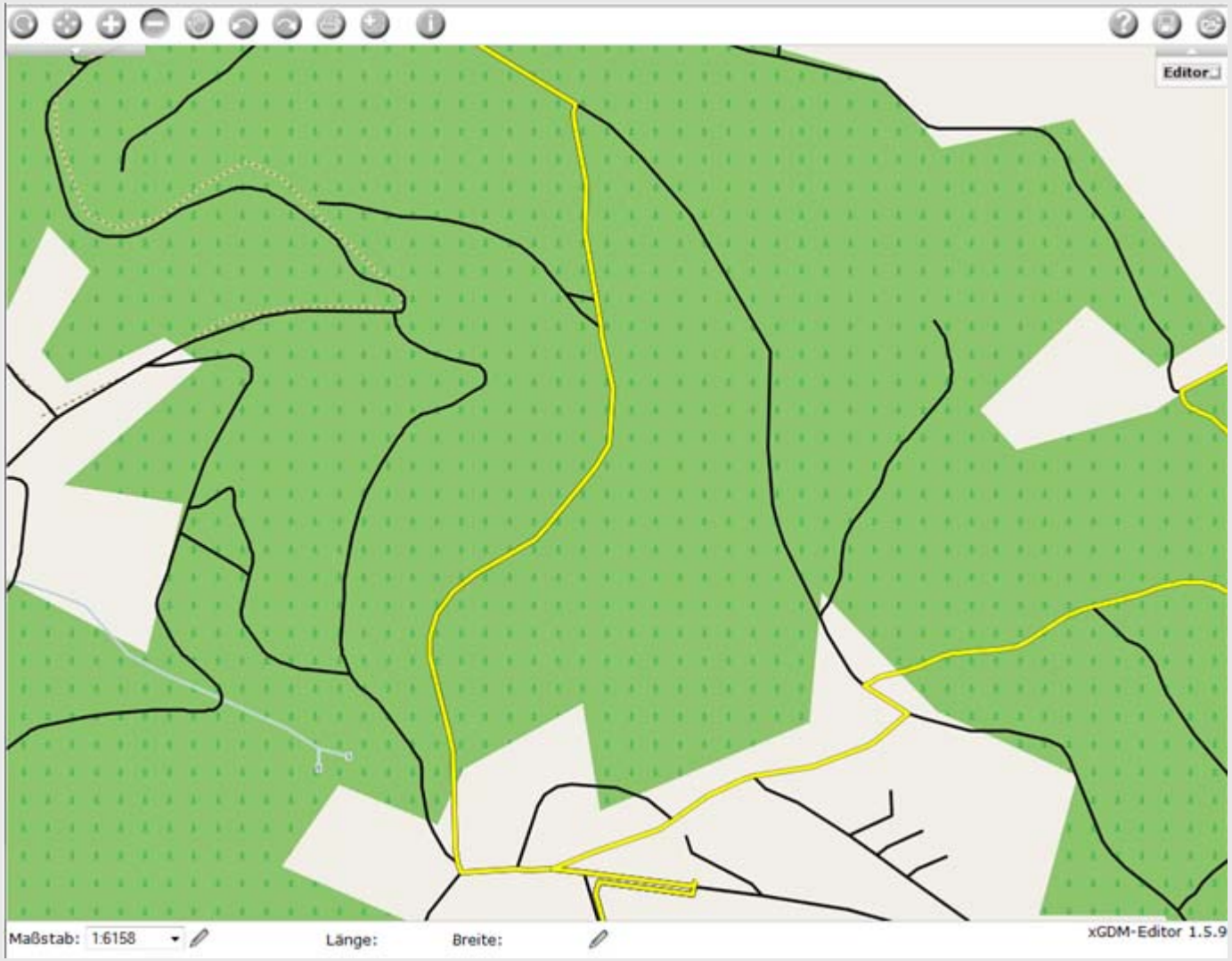
Offene Punkte / Probleme

- Problem bei Erstellung der gerouteten Geometrie: SDO Metainformationen (Genauigkeit) werden nicht aus der Datenbank berücksichtigt
  - Wahrscheinlich in der SDO API fest hinterlegt
  - Workaround: manuelles Erstellen der Spatial Geometrie über SDO_AGGR_CONCAT_LINES Funktion
- Partitionierung kann sehr viel Tablespace belegen
- Java API entspricht teils der Online – Dokumentation (Oracle® Spatial Java API Reference)
  - Reicht aber für erfahrene Java – Entwickler voll aus

Hinweise zu Routing mit Oracle 10g

- 10g API “In memory”
- Alle Daten des Netzwerkes werden zur Java Routing Engine übertragen
- Möglicher Workaround: Java JDK 64Bit
  - 64 Bit Speicheradressierung, so dass Routing auf größeren Netzwerken möglich ist
    - Entsprechende Hardware ist notwendig
  - Keine Partitionierung möglich
    - Ist nur eine Netzwerk Partition mit allen Daten

Visualisierung



Visualisierung

Fazit

- Mit der Oracle 11gR2 ist sehr viel besser und stabiler geworden ☺
  - Verarbeitung von großen Netzwerken ist durch den Einsatz der Netzwerk - Partitionierung möglich
  - Viele nützliche Funktion erleichtern die Arbeitsweise
    - PL/SQL
    - Java API
  - Stabile und sehr performante Java API
- Routing mit Oracle 11gR2 „Load on Demand“ ist für Produktiv Systeme bestens geeignet

Fragen ?

Vielen Dank für

Ihre Aufmerksamkeit