osm2graphium

Konzept

# Import und Segmentierung

Phase 1

* lies alle Ways, die der definierten Filterung entsprechen
* merke alle Node-IDs dieser Ways
* identifiziere alle Nodes, die als Segmentierungspunkte gelten
  + Node-ID darf nicht Start-oder End-Node-ID des Ways und die Anzahl muss größer 1 sein
  + => SegmentationNodesSink
* identifiziere alle Nodes, die als Barriers markiert sind
  + Node-ID darf nicht Start-oder End-Node-ID des Ways und die Anzahl muss größer 1 sein
  + sind ebenfalls Segmentierungspunkte
  + => BarrierNodesSink
  + sollte noch weiter diskutiert werden, ob Barriers überhaupt berücksichtigt werden müssen!
* merke zu diesen Nodes die Way-IDs (Speicher?)
  + generell müssen initial zu allen Node-IDs die entsprechenden Way-IDs gemerkt werden
* grundsätzlich müsste die Datenstruktur in etwa so aussehen:
  + Map<Long(nodeId), Refs>
  + Refs:
    - Long[] (wayIds)
    - byte (Segmentierungspunkt = 1, End-Node = 0)

Phase 2

* lies alle Ways, die der definierten Filterung entsprechen
* segmentiere jene Ways, die Segmentierungspunkte aufweisen
* merke Zuordnung neu segmentierte Ways zu originalen Ways ODER erstelle neue Way-ID, die auf die originale Way-ID rückschließen lässt
* erstelle Connections an Segmentierungspunkten
* erstelle Connections an regulären End-Nodes

Phase 3

* lies alle Relations und bearbeite Connections je nach Restrictions etc.

# Filterung

Die Filterung von Ways erfolgt mit Hilfe des Osmosis TagFilters. Hier können Keys und Key-Value-Pairs zur Filterung angegeben werden.

# ID-Generierung bei Segmentierung

Eine Segmentierung eines Ways ergibt eine Aufteilung in zwei (neue) Ways an eine m Kreuzungspunkt (Node). Die neuen Ways benötigen dabei jeweils neue Way-IDs.

Anforderung: Die Segmentierung eines Ways soll in jedem Change Set zu der selben neuen Way-ID führen. Die ID sollen somit stabil bleiben.

Ansatz: Verknüpfung der originalen Way-ID mit StartNodeIndex und EndNodeIndex (die beiden Indizes definieren die jeweilige Segmentierung.

Derzeitige Lösung:

IDneu = Long.MAX\_VALUE – (IDoriginal \* 10000000 + startNodeIndex \* 1000 + endNodeIndex)

# Node-Liste

Für die Segmentierung der Ways müssen jene Nodes ermittelt werden, die Kreuzungspunkte darstellen. Kreuzungspunkte sind jene Nodes, die von mehr als einem Way referenziert werden.

Dazu werden alle Ways, die dem Filterkriterium entsprechen, gelesen und deren Nodes in der folgenden Datenstruktur gespeichert:

Map<Long(NodeId), List<WayRef>

WayRef: Long (WayId)

byte (Type)

Type: 0 .. End Node

1 .. Normal Node

2 .. Segmentation Node

Zu jeder Node-ID wird eine Liste von Way-ID gespeichert. Zu jeder Way-ID existiert ein Type, der Auskunft darüber gibt welcher Art der Node in diesem Way ist. So kann ein Node im Way 1 ein End Node und im Way 2 ein zwischengelagerter Node sein.

Ablauf des Einfügens von Node-IDs und Way-IDs in die Datenstruktur:

* Identifizierung von End Nodes (Node-ID ist gleich jener des ersten oder letzte Nodes des Ways); diese erhalten den Type 0, ansonsten Type 1
* Ist noch kein Eintrag zu der Node-ID vorhanden   
  => Einfügen der Node-ID mit der zugehörigen Way-ID und dem entsprechenden Type
* Ist bereits ein Eintrag zu der Node-ID vorhanden   
  => Hinzufügen der Way-ID und dem entsprechenden Type  
  => Da nun die Anzahl der Way-IDs zu einer Node-ID größer als 1 ist, muss der Type jedes Ways zu dieser Node-ID überprüft und angepasst werden (normale Nodes werden nun zu Segmentation Nodes (Type 2), End Nodes dürfen keine Segmentation Nodes werden).

**Beispiel:**

1

8

2

7

3

5

6

4

w1

w2

w3

|  |  |  |
| --- | --- | --- |
| Analyse Weg | Node-ID | {Way-ID / Type} |
| w1 | 1 | w1/0 |
|  | 2 | w1/1 |
|  | 3 | w1/1 |
|  | 4 | w1/0 |
| w2 | 3 | w2/0, w1/2 (Type von w1 > 0 und Anzahl Ways > 1 => neuer Type 2) |
|  | 5 | w2/1 |
|  | 6 | w2/0 |
| w3 | 7 | w3/0 |
|  | 2 | w3/2, w1/2 (Type von w1 > 0 und Anzahl Ways > 1 => neuer Type 2) |
|  | 8 | w3/0 |

# Restrictions

Laut OSM Wiki (<http://wiki.openstreetmap.org/wiki/DE:Relation:restriction>):

Eine restriction-Relation hat mindestens folgende drei Mitglieder:

* genau ein way mit Rolle from: Der Weg, von dem aus die Beschränkung gilt.
* genau ein way mit Rolle to: Der Weg, zu dem die Beschränkung gilt.
* entweder ein node oder ein oder mehrere way's mit Rolle via: Im Normalfall der Punkt, der die Kreuzung beider Wege beschreibt.

Für die Verarbeitung sind nur Restricitions relevant, deren From- und To-Ways laut Filterung importiert werden.

# OSM Tags

## Oneway

A way is oneway if and only if it has:

* oneway=yes or
* oneway=-1 (denotes opposite direction to node order) or
* junction=roundabout or
* highway=motorway or
* highway=motorway\_link
* and oneway ≠ no

## Name

The following tags may contain the name of a street:

* name=\*, "name:<2digit-language-code>", int\_name, ...
* ref=\*, "nat\_ref", "loc\_ref", "int\_ref"

Weiters können Ways zu Routes zusammengefügt werden. Dies erfolgt über Relationen. Route-Relationen können ebenfalls Straßennamen beinhalten.

## Access

Ein Way ist für die Benützung nur erlaubt, wenn es kein Tag „access“ gibt oder dieses den Wert „yes“ besitzt.

Weiters kann die Benützung noch feingranularer definiert sein – für uns relevant:

* …

Zur Berechnung der Accesses siehe auch:

<http://wiki.openstreetmap.org/wiki/Computing_access_restrictions>

# Osmosis – Graph ausschneiden (Preprocessing Step)

osmosis --read-pbf file="europe-latest.osm.pbf" --bounding-polygon file="dachi.poly" –write-pbf file="germany.osm.pbf"