Geo-Tabellen im HomeRun-Projekt

Internes technisches Papier
03.11.2017 22:54

Jörg Roth
TH Nürnberg
90489 Nürnberg
Joerg.Roth@th-nuernberg.de

Zusammenfassung: Dieses Papier beschreibt die Geo-Tabellen der HomeRun-Datenbank.

1 Einleitung ... 3

2 Die Geodaten-Strukturen ... 4

2.1 Das Grundelement – die Domain ... 4
2.2 Klassifizierungen von Objekten ... 5
2.2.1 Das grundlegende Klassifizierungsschema ... 5
2.2.2 LSI-Klassen ... 5
2.2.3 Eigenschaften ... 6
2.2.4 OSM-Tags ... 8
2.2.5 Adressen ... 8
2.2.6 Zugehörigkeiten zu Land, Bundesland, Stadt etc. ... 9
2.3 Zusammenfassen von Objekten – Relations ... 10
2.4 Geometrien ... 11
2.5 Weitere geometrische Eigenschaften ... 13
2.6 Topologien ... 14
2.6.1 Generelles ... 14
2.6.2 Sackgassen ... 16
2.6.3 Routing unter Einbeziehung von Flächen ... 16

3 Weitere Datenquellen ... 17

3.1 NASA-Höhenprofile ... 18
3.2 Global Administrative Areas ... 18
3.3 Wikipedia ... 18

4 Datenbank-Tabellen ... 19

4.1 domain ... 19
4.2 domain_properties ... 21
4.3 domain_addresses ... 22
4.4 relation ... 22
4.5 is_ins ... 23
4.6 is_near ... 24
4.7 crossing ... 24
4.8 link ... 25
4.9 areacrossing ... 26
4.10 lsiclasses ... 26
4.11 properties ... 27
4.12 address_tags ... 27

1 Einleitung

Der Geo-Datenbestand entsteht aus dem Import aus der Datenquelle *Open Street Map* (*OSM*). In der Regel wird dazu die Datei **`germany`** verwendet, die alle gesammelten Geo-Objekte von Deutschland enthält. Beim Import werden die folgenden Arbeiten ausgeführt:

- In Linienzügen, Flächen und zusammengesetzten Objekte werden die Referenzen auf Position und Flächen durch Kopien ersetzt. Als Resultat wird jedes Geo-Objekt durch eine eigene vollständige Geometrie beschrieben.
- In OSM sind Flächen von Linienzügen nur dadurch unterscheidbar, dass Anfangs- und Endpunkte einer Punktfolge zusammenfallen. Anhand von OSM-Schlüsseln oder Objekttypen kann dies nicht entschieden werden. Dieser Test wird beim Import durchgeführt und Geometrien entsprechend gekennzeichnet.
- Punkt- und linienförmige Geometrien besitzen keine flächenartige Ausprägung. Viele Such-Operationen erfordern jedoch die Hinterlegung einer Fläche. Daher wird zu solchen Objekten ein zusätzliches Flächenobjekt konstruiert, indem die Geometrie durch die so genannte Puffer-Operation 'aufgeblasen' wird. Die Linien- oder Punktgeometrien bleiben jedoch erhalten, so dass eine Anwendung sich später die für den Anwendungsfall geeignete geometrische Repräsentation aussuchen kann. (Bemerkung: gilt nur noch für die das *depro-Format*).
- OSM kann weder Multipolygone noch mehrfache Linien in einzelnen Linien- oder Flächenobjekten darstellen (erst durch die *relations*). Darüber hinaus werden ausgedehnte Linienobjekte (z.B. Autobahnen) in viele Teilstücke zerlegt. Beim Importieren wird versucht, passende Einzelobjekte wieder zu einem Gesamtobjekt zusammenzufassen. Dies ist eine sehr komplizierte Funktion, da die Zusammengehörigkeit heuristisch durch Namensgleichheit und geometrische Nähe festgestellt werden muss. Zusammengesetzte Objekte werden in einer weiteren Tabelle verwaltet, die Einzelobjekte stehen aber weiter in der Haupttabelle zur Verfügung.
- Die zeichenbasierte Klassifikationsbeschreibung über Tags (siehe auch Abschnitt 2.2.3) wird über ein Regelwerk interpretiert und auf das eigene nummernbasierte Klassifikationsschema abgebildet. Näheres dazu wird in einem eigenen Dokument "LSI-Klassifikation im HomeRun-Projekt" beschrieben.
- Liegt kein Objektname vor, wird ein automatisch generierter Name eingetragen.
- Es wird eine Wegenetz-Topologie aus den Wegen berechnet, so dass Routing-Algorithmen realisiert werden können.

Der gesamte Import dauert für die mehreren Million Datensätze zurzeit mehrere Tage. Die Datenbank umfasst dann die folgenden Tabellen:

Tabelle	**Zweck**
`domain`	Elementare Geo-Objekte wie Wege, Wälder, Grundstücke, Punktobjekte. Einträge enthalten eine Klassifikation, Namen, die Geometrie sowie Felder für den räumlichen Index.
`domain_properties`	Weitere Eigenschaften zu Domains (z.B. Öffnungszeiten von Restaurants)
`domain_addresses`	Adressen-Einträge zu dieser Domain (z.B. Straßenname, Hausnummer)
`relation`	Zusammenfassung von Geo-Objekten zu einem übergeordneten Objekt
`is_ins, is_near`	Dokumentiert "enthalten in", z.B. "in Stadt", "in Bundesland" sowie die Nähe zu einer Beschriftung
`crossing`	Kreuzungen im Wegenetz

areacrossing	Kreuzungen zwischen Flächen und Wegen (experimentell)
link	Verbindungen zwischen Kreuzungen im Wegenetz
lsiclasses	Abbildung Objekt-Klassennummer zur Beschreibung
properties	Typenspezifikation von Domain-Eigenschaften
address_tags	Liste aller Adress-Schlüssel (z.B. **addr:street**)

Es gibt zwei Datenbank-Varianten:

- *depro-Format*: es wird eine Standard-SQL-Datenbank *ohne* räumliche Erweiterung verwendet. Die räumliche Indexierung wird über den so genannte *Extended Split Index* geleistet, der in klassischen SQL-Spalten abgelegt werden kann. Eine Zugriffsbibliothek leistet die notwendigen Index-Berechnungen. Geometrien werden als *Well-Known-Binary* in BLOBs gespeichert.
- *PostGIS*: es wird die räumliche Erweiterung *PostGIS* verwendet. Diese übernimmt die räumliche Indexierung. Geometrien können als *Well-Known-Binary* oder *Well-Known-Text* abgefragt werden. PostGIS erlaubt darüber hinaus die Formulierung von geometrisch/geographischen Funktionen, beispielsweise Flächenberechnungen auf der Erdoberfläche. Aus diesem Grund können viele vorberechnete Eigenschaften des depro-Formats entfallen.

Die Formate unterscheiden sich in einigen Tabellen. Bei unterschiedlichen Tabellen werden die gemeinsamen Spalten sowie die Spalten der Format-Varianten dargestellt (Abschnitt 4).

2 Die Geodaten-Strukturen

2.1 Das Grundelement – die Domain

Jedes Geo-Objekt wird in einer einzelnen Datenzeile in der **domain**-Tabelle gespeichert. Geo-Objekte sind

- Straßen, Wege, Parkplätze
- Bäume, Seen, Hügel, Felsen, Wälder
- Stadtgrenzen, Stadteilgrenzen
- Restaurants, Geschäfte, Ämter
- etc.

Geo-Objekte können durch Flächen-, Linien oder Punktgeometrien beschrieben werden. Die Geometrien sind dabei im Objekt vollständig definiert, d.h. Polygonpunkte werden nicht durch Verweise auf eine weitere Tabelle dargestellt.
Neben der Geometrie beschreiben folgende Informationen ein Objekt:

- der *Typ*, z.B. "Wald", "italienisches Restaurant"
- der *Name*, z.B. "Pizzeria Luna"
- *weitere Eigenschaften*, z.B. "Nadelwald", "Öffnungszeiten von 11:00-24:00"

Der Typ eines Objekts wird durch die so genannte *LSI-Klasse* (siehe Abschnitt 2.2.2) beschrieben und in einer eigenen Spalte gespeichert. Auch für den Namen gibt es eine spezielle Datenbank-Spalte. Für weitere Eigenschaften existiert kein strukturiertes Schema in der Datenbank, da es zu unterschiedliche Muster gibt. Daher werden weitere Eigenschaften durch das Beschreibungsschema von OSM in Form vom Schlüssel-Wert-Paaren dargestellt (siehe Abschnitte 2.2.2 und 2.2.3).

2.2 Klassifizierungen von Objekten

2.2.1 Das grundlegende Klassifizierungsschema

Es gibt vier Eigenschaften, die ein Objekt beschreiben:

- *Basistyp*: ein Objekt kann konkret oder eine Relation sein.
- *LSI-Klasse*: einem Objekt wird eine Klassifikationsnummer zugeordnet, die die Art des Objektes kennzeichnet (z.B. "Baum", "Straße").
- *OSM-Tags*: Weitere Eigenschaften liegen als Schlüssel-Werte-Paare vor. Es gibt dazu verschiedene Rubriken, z.B. Namen-Tags, Adress-Tags.
- *Is-In*: Hier wird eine geometrische "Enthaltensein"-Beziehung aufgebaut. Is-In beschreibt vorwiegend, ob eine Domain geometrisch in einer Stadt oder in einem Bundesland liegt.

Zum Basistyp: die meisten Domains sind *konkret*, d.h. sie beschreiben ein ganz bestimmtes Objekt der Umwelt. Objekte vom Basistyp *Relation* beschreiben Objekte der Umwelt, die aus verschiedenen Einzelobjekten aufgebaut sind. Der Zusammenhang des Objektes zu seinen Bestandteilen wird über eine eigene Tabelle beschrieben (siehe Abschnitt 2.3).

2.2.2 LSI-Klassen

OSM bietet ein einfaches Verfahren an Objekte zu klassifizieren, indem Geo-Objekten beliebig viele Paare von Schlüssel-Wert (jeweils eine Zeichenkette) zugeordnet werden. Diese Paare unterliegen keiner Restriktion, müssen also nicht aus einer festen Liste stammen. Jeder Teilnehmer, der einen Datensatz beisteuert, kann beliebige Paare eintragen. Bezüglich vieler Objekttypen gibt es Konsens über sinnvolle Richtlinien für die Zusammensetzung von Paaren, die Einhaltung der Richtlinien wird aber nicht technisch überprüft. Ein Beispiel für eine OSM-Klassifikation eines Weges:

```
highway=residential,maxspeed=30,oneway=yes,surface=paved,
width=5,name=Eigenheimstraße,postal_code=01217
```

Diese Beschreibung ist für eine effiziente Suche nach bestimmten Objekttypen zu komplex. Aus diesen Gründen wird für den importierten Datenbestand ein eigenes Klassifikationsschema verwendet. Dieses verwendet einen einzelnen Zahlenwert, um Klassen zu definieren (Abb. 1). Die erste Ziffer gibt eine Hauptklasse an, z.B. organisatorisch, bebaute Fläche oder offene Fläche. Ist für ein Objekt eine genauere Klassifikation möglich, so wird eine weitere Ziffer verwendet. Man kann bis zur maximalen Stellenzahl (derzeit fünf) weitere Stellen verwenden und erhält eine immer genauere Klassifikation. Durch die Ziffernvergabe entsteht eine einfache baumartige Zuordnung von Klassen und Unterklassen. Derzeit werden ca. 700 Klassen unterschieden.

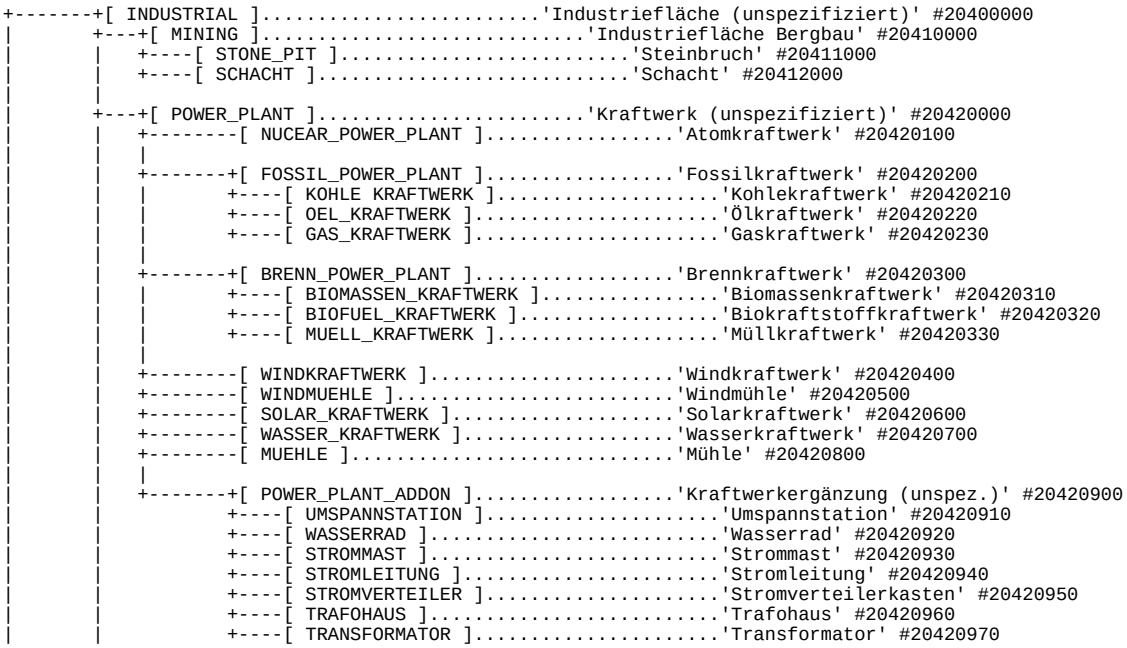


Abb. 1: Ausschnitt des LSI-Klassen-Baums

2.2.3 Eigenschaften

Geo-Objekte können Eigenschaften haben. Diese werden aus den OSM-Tags ausgelesen und für jedes Domain-Objekt abgelegt. Die Liste der möglichen Eigenschaften pro Objekt wird durch seine LSI-Klasse gesteuert.
Da es pro Objekt mehrere Eigenschaften geben kann wurde eine eigene Tabelle **`domain_properties`** verwendet. Die Beziehung von Eigenschaften zu Geo-Objekten wird durch die eindeutige Objekt ID (**`d_id`**) gebildet.
Damit die "Klassen" von Eigenschaften nicht in jedem Eigenschaftseintrag wiederholt werden, gibt es eine weitere Tabelle **`properties`**. Diese stellt gewissermaßen die Obermenge möglicher Eigenschaften dar. Ein Eintrag in **`properties`** sieht beispielsweise so aus:

- **`p_id=0`**: ein eindeute ID für jeden Eigenschaftstyp,
- **`type=METERS`**: der Datentyp der Eigenschaft,
- **`key=distance`**: der Schlüssel entweder in OSM oder Wikipedia,
- **`unit=m`**: die Einheit.

Steht in der OSM-Beschreibung eines Objektes beispielsweise das Schlüssel-Wert-Paar

`distance=720.0`

dann wird daraus ein Eintrag in **`domain_properties`** angelegt mit

- **`d_id=`** <d_id des entsprechenden Domain-Objektes>
- **`slot=1`**, wenn das z.B. eine Eigenschaft der ersten LSI-Klasse des Objekte ist
- **`p_id=0`**, Verweis auf den **`properties`**-Eintrag oben
- **`value="720.0"`**
- **`int_value=720`**
- **`double_value=720.0`**

Die Spalten **`int_value`** und **`double_value`** beinhalten die Zahlenwerte, wenn

- Der Text der Eigenschaft vom Muster her eine Mengen-Definition beinhaltet *und*
- der Text vom Muster her die richtige Mengeneinheit definiert.

Darüber hinaus:

- **`double_value`** enthält immer einen Wert, wenn **`int_value`** einen Wert besitzt.
- **`int_value`** enthält immer einen Wert, wenn **`double_value`** einen Wert einer ganzen Zahl besitzt.

Ein Beispiel:
Die Eigenschaft **`wikipedia:Kosten`** hat den Typ **`EURO`**. Ein Eigenschaftstext lautet "2 Mio. Euro". Dann steht in **`int_value`** sowie **`double_value`** der Wert 2000000. Steht dagegen als Eigenschaftstext "Sehr hoch", oder "500 DM", dann sind **`int_value`** und **`double_value`** beide NULL. Würde der Eigenschaftstext "100,0 €" lauten, so wäre auch **`int_value`** auf 100 gesetzt, da es sich bei 100,0 zwar um ein geschriebenen Komma-Zahl handelt, diese vom Wert her eine ganze Zahl ist.

Die Einträge der Spalte **`type`** in **`properties`** ist ein String, der selbst nicht mehr in einer weiteren Tabelle steht. Die Bedeutung ist oft offensichtlich. Hier die Liste der möglichen Werte:

properties.type	**Beschreibung**	**Beispiele – properties.key**	**Beispiele – domain_properties.value**	**properties.unit**
ACCESS	Zugang zu Wegen	**motor_vehicle, access, motorcar, motorcycle**	yes, no, private, customers	
BICYCLE	Dürfen Fahräder?	**bicycle**	yes, no	
BOOL	Wahrheitswert	**segregated, shelter, bicycle, noexit**	yes, no	
BOOL_LIMITED	Wahrheitswert mit "Limited"	**wheelchair**	yes, no, limited	
BUILDINGROOF	Typ eines Daches	**building:roof**	gabled, tile, flat	
COLOUR	Farben	**colour, roof:colour, buoy:colour, fire_hydrant:colour**	#FFD700, #8C6954, #df872b	
EURO	Geld-Werte	**wikipedia:Jahresetat, wikipedia:Umsatz, wikipedia:Bilanzsumme, wikipedia:Baukosten**	45,5 Mio. Euro, 204.520.000 €, 741 Mio. Euro	
FENCE_TYPE	Aussehen eines Zauns	**fence_type**	chain_link, wood, wire	
FLOAT	Komma-Zahlen	**frequency, wikipedia:Bevölkerungsdichte, wikipedia:pH-Wert**	235 Einwohner je km², 16.7	
FOOT	Darf man zu Fuß?	**foot, footway**	yes, no, designated, sidewalk	
GOODBAD	Gut und schlecht	**trail_visibility**	good, bad, excellent, horrible	
GRADE	Kategorie von Wegen	**tracktype**	grade1, grade2, grade4	
HORSE	Dürfen Pferde?	**horse**	yes, no, permissive, agricultural	
INCLINE	Steigung	**incline**	17.0%, -18.0%, down, up	
INT	Ganze Zahlen	**population, level, wikipedia:Betten, wikipedia:Besucherzahlen**	1122, 14 Ortsteile	
INT_HALFS	Ganze Zahlen und "Halb"	**building:levels**	6, 16, 0.5, 2.5	
INT_YES_NO_UNKNOWN	Ganze Zahlen, Ja und Nein und unbekannt	**capacity:disabled**	yes, no, unknown	
KM2	Große Flächen	**wikipedia:Fläche, wikipedia:Einzugsgebiet**	19,91 km², ca. 12,4 km²	km²
M2	Kleine Flächen	**wikipedia:Platzfläche, wikipedia:Grundfläche,**	14,4 m², ca. 6170 m², 13.290 m²	m²

		wikipedia:Querschnitt		
M3	Volumina	**wikipedia:Volumen wikipedia:Bemessungshochwasser, wikipedia:Speicherraum**	13.570.000 m³, 1 Mio m³, 182 Mio. m³	m³
METERS	Distanzen	**distance, height, maxheight, wikipedia:Höhe, wikipedia:Straßenlänge**	30,6 m, 56 m	m
MW	Leistung	**wikipedia:Kraftwerksleistung, wikipedia:Leistung**	843 Megawatt brutto, 392 MW	MW
ONEWAY	Ist es Einbahnstraße?	**oneway**	yes, no	
ORIENTATION	Himmelsrichtung	**roof:orientation**	n, along, ne, se	
PERCENT	Anteile	**wikipedia:Ausländeranteil**	4 %, 12,8 %	%
RATIO	Verhältinis (insb. von Steigungen)	**wikipedia:Böschungsneigung luftseitig, wikipedia:Böschungsneigung wasserseitig**	1:2,5, 1:0,69	
ROOFMATERIAL	Material eines Daches	**building:material, roof:material**	stone, glass, thatch, gravel	
ROOFSHAPE	Aussehen eines Daches	**roof:shape**	hipped, pyramidal, pitched, gabled	
SACSCALE	Wanderskala	**sac_scale**	hiking, alpine_hiking	
SMOOTHNESS	Begehbarkeit von Wegen	**smoothness**	good, bad,excellent	
SPEED	Geschwindigkeit	**maxspeed, zone:maxspeed**	50, 100, 55 km/h, 24 km/h	km/h
STRING	Jeder Text	**line, operator, description, religion, wikipedia:Primärenergie, wikipedia:Eigentümer**	Tu 13:30-17:30, Rathaus	
SURFACE	Oberfläche (z.B. von Wegen)	**surface**	stone, mud, wood, gravel	
TONS	Gewicht	**maxweight**	24.0, 5.5	t

2.2.4 OSM-Tags

Damit die OSM-Tags der Beschreibung nach dem Import weiterhin verfügbar sind, werden sie in jedem **`domain`**-Datensatz in der Datenbank gespeichert. Typischerweise würde man eine Suche erst einmal nach Datensätzen über die LSI-Klasse machen. Liegt ein Datensatz aber einmal als Ergebnis einer Datenbankanfrage vor, kann man so weitere Details über ein Objekt erfragen. Es wird nicht empfohlen, Inhalte der OSM-Tags in Suchbedingungen zu verwenden, da solche Suchen z.B.

```
WHERE tags LIKE '%highway=residential%'
```

sehr lange dauern.
Die verschiedenen OSM-Tags werden bestimmten inhaltlichen Gruppen zugeordnet.

Tag-Klasse	Bedeutung
`tags`	Eigenschaften des Objektes, z.B. Öffnungszeiten, Maximal-geschwindigkeit
`tags_name`	Verschiedene Namen des Objektes, z.B. Namen in verschiedenen Sprachen, Autobahnnummer
`tags_addr`	Postadresse des Objektes
`tags_relative`	Zusammenhang zu anderen Objekten, z.B. 'liegt in'
`tags_link`	Links zu Wikipedia, Bildern etc.
`tags_organizational`	Verwaltungsinformationen zum Objekt
`tags_others`	noch nicht klassifizierte weitere Tags

2.2.5 Adressen

OSM kann Post-Adressen zu Objekte ausdrücken. Diese sind in den Tags **`addr:…`**, **`postal_code`** sowie in einigen `contact:…` hinterlegt. Die zusammengefügte Liste all dieser Tags ist in dem Domain-Spalte **`tags_addr`** schon Komma-getrennt hinterlegt. Zu-

sätzlich wird für jeden einzelnen Eintrag in der Tabelle **domain_addresses** jeder Wert einzeln abgelegt und kann damit über SQL recherchiert werden.
Die Schlüssel liegen selbst in einer eigenen Tabelle **address_tags**, damit diese nicht wiederholt abgelegt werden müssen. Der Zugriff zu den Schlüsseln erfolgt über einen Integer-Wert a_id. Für einige Adress-Tags ("well-known tags") gibt es eine feste Zuordnung:

a_id	OSM-Tag	Bedeutung/Bemerkung	Beispiel
1	**postal_code**	Postleitzahl; gemäß OSM gibt es einen Unterschied zu **addr:postcode**	90489
2	**addr:postcode**	Postleitzahl; gemäß OSM gibt es einen Unterschied zu **postal_code**	90489
3	**addr:country**	Länderkennzeichen	DE
4	**addr:state**	Bundesland	Bayern
5	**addr:city**	Stadt	Nürnberg
6	**addr:suburb**	Adress-Zusatz zur Straße (wird selten verwendet)	Nonnhof
7	**addr:street**	Straße	Ostendstraße
8	**addr:housenumber**	Hausnummer	6
9	**addr:housename**	Eigentlich: Ersatz für Hausnummer, häufig als Zusatz zur Hausnummer verwendet	Tierheim
10	**addr: interpolation**	Wie kann die nächste Hausnummer gebildet werden (z.B. nur ungerade auf dieser Seite)	odd
11	**contact:phone**	Die Telefonnummer **phone** wird beim import auf **contact:phone** abgebildet und auch hier abgelegt	+49 123 497
12	**contact:fax**	Die Faxnummer **fax** wird beim import auf **contact:fax** abgebildet und auch hier abgelegt	+49 123 987

2.2.6 Zugehörigkeiten zu Land, Bundesland, Stadt etc.

Die Tabelle **is_ins** dokumentiert die Zugehörigkeiten zu übergeordneten Strukturen.

Tabellenspalte (is_in*)	Übergeordnete Struktur
1	Staat
2	Bundesland
3	Regierungsbezirk
4	Kreisfreie Stadt, Landkreis
5	Amtsgemeinde
6	Stadt, Gemeinde
7	Stadtteil
a,b,c	Weitere Zugehörigkeiten (zur Zeit: Größere Region, Region)

Die Spalten **is_in<nr>** beschreiben n-zu-1-Zugehörigkeiten, d.h. eine Domain ist immer in genau einem Land, einem Bundesland etc. Die Einträge können auch **null** sein, wenn die Zugehörigkeit nicht festgestellt werden kann oder nicht sinnvoll existiert. So gibt es beispielsweise Straßen, die in zwei Bundesländern verlaufen, diese enthalten dann für

is_in2 den Wert **null**. Enthalten ist jeweils die **d_id** der übergeordneten Domain. Zu den Spalten 1...7 gibt es jeweils drei Einträge (z.B.**is_in1**, **is_in1b**, **is_in1c**), so dass Objekte auch zu maximal drei übergeordneten Objekten gehören können.
Aus Performance-Gründen werden die Einträge 1 (*Staat*) und 2 (*Bundesland*) normalerweise nicht berechnet, insb. da deren Nutzen gering ist.
Darüber hinaus kann es Zugehörigkeiten zu *Regionen* geben. Da es davon mehrere geben kann, gibt es bis zu drei Felder **is_ina**, **is_ina** und **is_inc**, um das auszudrücken. Die Felder **is_ina**, **is_ina** und **is_inc** ist als ungeordnete Menge zu verstehen.
Die Tabelle hat eine {0,1}-zu-1-Beziehung zur Tabelle **domain**. Denkbar wäre gewesen, dass die **is_in**-Spalten in einer Datenzeile eines domain-Eintrags abgelegt werden. Allerdings erzeugt das einen zu großen zeitlichen Aufwand beim Anlegen einer Datenbank, daher wurden die **is_in**-Einträge in einer eigenen Tabelle abgelegt. Inhaltlich gehören die Zeilen in der **is_ins**-Tabelle allerdings zu den jeweiligen **domain**-Datenzeilen.
Auch Domains vom Basistyp Relation können einen Eintrag in der **is_ins**-Tabelle haben. Ein Eintrag in einer bestimmten **is_in<nr>**-Spalte signalisiert dabei, dass *alle* Bestandteile der Relation an dieser Stelle denselben Eintrag besitzen. Gibt es nur eine Abweichung, dann sind nicht alle Bestandteile in derselben übergeordneten Struktur enthalten, und in der entsprechenden Spalte wird der Wert **null** gesetzt.
Manchmal sind die Grenzen von übergeordneten Objekten wie Städte oder Stadtteile nicht als Flächenobjekte verfügbar. In diesem Fall kann die Zugehörigkeit über eine zweite Tabelle geschätzt werden. Es wird der räumliche Abstand in Metern zu relevanten Beschriftungen eingetragen.

Tabellenspalte (is_near*)	**Übergeordnete Struktur**
1	Staat
2	Bundesland
3	Landkreis, Stadt, Kleinstadt, Dorf
4	Vorort, Stadtteil
a,b,c	Weitere Zugehörigkeiten (zur Zeit: Region, Lokalität, Insel, Kleininsel)

Auch hier gibt es den Spalten 1...4 jeweils drei Einträge (z.B.**is_near1**, **is_near1b**, **is_near1c**), wobei dort die Abstände zu den Beschriftungsobjekten aufsteigend einsortiert werden, d.h. **is_near3** enthält beispielsweise die nächstgelegene Stadtbeschriftung und **is_near3b** diejenige mit dem zweitgrößten Abstand.
Die Abstände in Metern befinden sich in den Spalten **distance***, also z.B. **distance6b** für die Stadt mit dem zweitgrößten Abstand.

2.3 Zusammenfassen von Objekten – Relations

Es gibt Objekte, die aus mehreren Objekten inhaltlich zusammengesetzt werden. Beispiele:

- Eine Stadtgrenze besteht aus mehreren Teilgrenzen. Beispielsweise besteht die Grenze von Nürnberg aus der Grenze zu Erlangen, zu Fürth und zum Nürnberger Land. Jede einzelne Grenze wird von zwei Städten oder Landkreisen verwendet, daher ist es sinnvoll, diese als Einzelobjekt zu erhalten. Die gesamte Grenze von Nürnberg lässt sich daher am besten als Sammlung dieser Teilgrenzen beschreiben.
- Der Radwanderweg "Rothenburg bis Kehlheim" besteht aus verschiedenen einzelnen Radwegen, die gedanklich zu einer übergeordneten Struktur zusammengefasst werden. Die einzelnen Wege sollten aber weiterhin vorhanden bleiben, da man diese Radwege auch einzeln befahren kann.

- Beschreibende Eigenschaften können für jedes Objekte nur einmal vergeben werden. So kann jede Straße nur eine einzige Maximalgeschwindigkeit haben. Es kann aber sein, dass sich diese Eigenschaften abschnittsweise ändern. So zerfällt eine Autobahn in viele Einzelteile mit unterschiedlichen Maximalgeschwindigkeiten. Damit das übergeordnete Objekte (z.B. "A3") weiterhin erkennbar ist, kann man diese aus den Abschnitten zusammensetzen.
- Es gibt Objekte mit komplexen Geometrien. So ist die Wasserfläche eines Sees durch das Ufer minus der Inseln definiert. In OSM können flächenartige Geometrien nur durch ein einzelnes geschlossenes Polygon definiert werden (also insb. ohne Löcher). Damit komplexe Geometrien (allgemein wären das Multipolygone mit Löchern) ausgedrückt werden können, kann man Objekte zusammenfassen (wobei "zusammenfassen" auch "abziehen" bedeuten kann).

Das Zusammenfassen von Objekten wird in OSM durch so genannte *relations* beschreiben. Bei den Beispielen oben kann man unterschiedliche Arten von relations erkennen:
- Eine inhaltliche Zusammenfassung verschiedener Objekte zu einem "neuen" größeren Objekt (z.B. Radwanderweg "Rothenburg bis Kehlheim").
- Die Zusammenfassung von Objekten, um im Objekte verschiedene Eigenschaften ausdrücken zu können (z.B. bei einer Autobahn).
- Die Zusammenfassung von Objekten, um komplexe Geometrien ausdrücken zu können (z.B. ein See).

Die letzte Art von relation ist für unsere Zwecke eigentlich unpassend. Die Geometrien in der Datenbank können nämlich mehr beschreiben als die Geometrien in OSM, insb. Multipolygone mit Löchern. Daher versucht das Import-Verfahren solche relations aufzulösen und die ausgerechneten Geometrien in Einzelnobjekten zu speichern. Wenn z.B. ein See durch den Inhalt des Ufer-Polygons minus der Inseln definiert ist, werden die Inselflächen beim Import abgezogen und das Polygon mit Löchern in der Geometrie gespeichert. Ist der Auflösungsprozess beim Import erfolgreich, enthält das Objekt in der Datenbank später keinen Hinweis darauf, dass es ursprünglich durch eine relation definiert war.
In den meisten Fällen können die relations nicht auf diese Art aufgelöst werden und werden daher als relation in der Datenbank gespeichert. Hierzu wird eine weitere Tabelle (**`relation`**) verwendet, die jedem übergeordneten Objekt seine Bestandteile zuordnet. Ein untergeordnetes Objekt kann eine "Rolle" einnehmen, für die OSM verschiedene Vorschläge macht.

2.4 Geometrien

Die wesentliche geometrische Eigenschaft eines Objektes ist die Projektion auf die Erdoberfläche (genauer, den Ellipsoiden). Damit wird ein Objekt im Wesentlichen durch eine zweidimensionale Struktur beschrieben. Open Street Map unterstützt folgende Geometrien (gemäß der "Simple Features"):
- Point: Punkt-Objekte, OSM-Objekte vom Typ **`node`**
- Line String: nicht geschlossene Linien-Züge, OSM-Objekte vom Typ **`way`**
- Linear Ring: geschlossene Linien-Züge, OSM-Objekte vom Typ **`way`**, mit gleichem Anfangs- und Endpunkt
- Polygon (ohne Holes): geschlossene Linien-Züge, OSM-Objekte vom Typ **`way`**, mit gleichem Anfangs- und Endpunkt von denen klar ist, dass die Fläche und nicht die Grenze das eigentlich Objekt beschreiben.

Abb. 2 (oben) zeigt diese elementaren Geometrietypen.

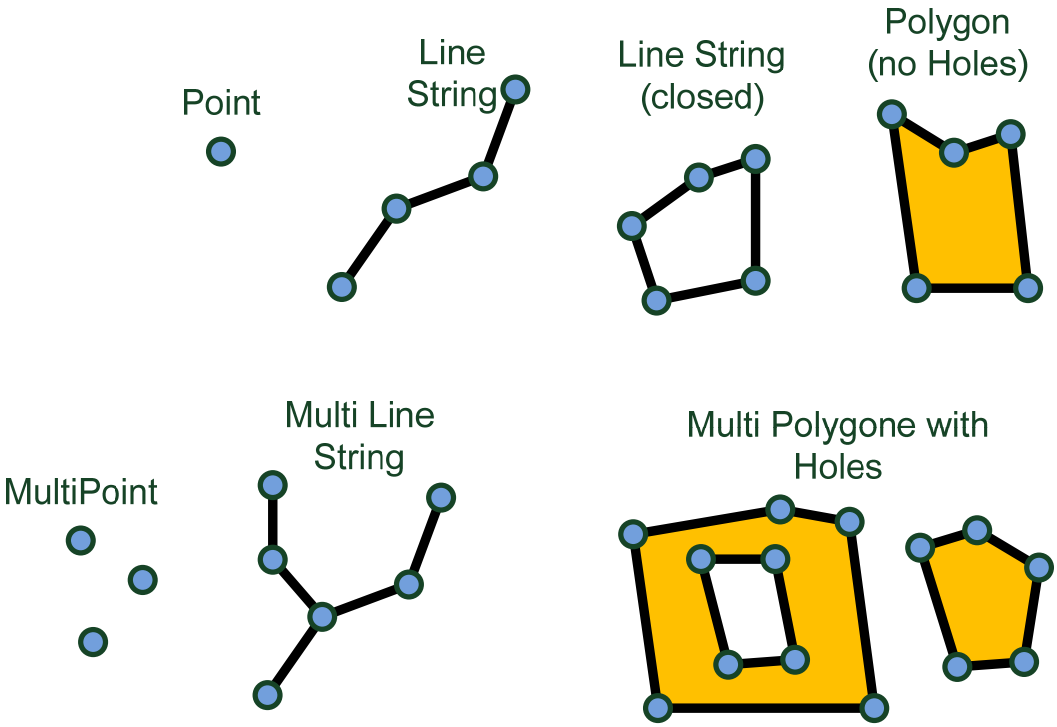


Abb. 2: Unterstützte Geometrien

Durch OSM-Einträge des Typs **`relation`** können elementare Geometrien zu weiteren Geometrien entstehen (Abb. 2 unten). Zu nicht flächenförmigen Objekten wird im depro-Format der Datenbank immer zusätzlich eine Flächengeometrie gespeichert. Diese entsteht, indem eine Breite (je nach Objekttyp) berechnet wird, und über die geometrische Buffer-Funktion die Punkt- oder Liniengeometrie "aufgeblasen" wird (Abb. 3).

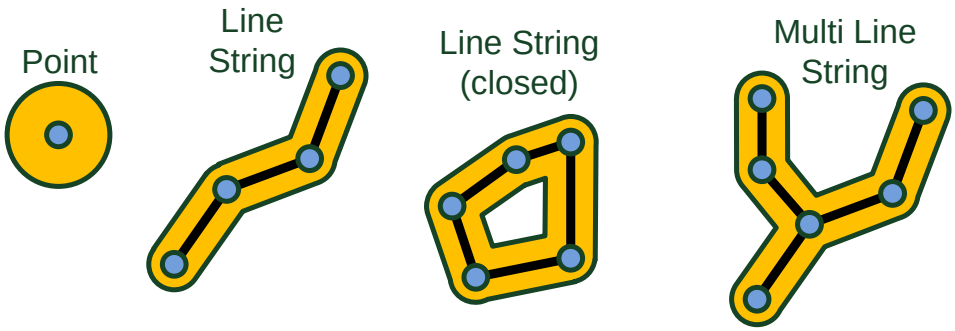


Abb. 3: Ableitung von Area-Geometrien aus nicht-Area-Geometrien (nur depro-Format)

Die folgende Tabelle zeigt, wie alle OSM-Geometrien in der Datenbank abgelegt werden, zusätzlich, welcher Simple Feature Typ bei der Deserialisierung aus der Datenbank entsteht.

OSM-Typ	Kennzeichen (Spalte `geometry`)	Relevante Geometrie-Spalten	Simple Feature
`node`	**`P`**	**`geodata_point`** zusätzlich Buffer in **`gao_geometry`**	Point
`way` (open)	**`L`**	**`geodata_line`** zusätzlich Buffer in **`gao_geometry`**	LineString (offen)
`way` (closed)	**`C`**	**`geodata_line`** zusätzlich Buffer in **`gao_geometry`**	LineString (geschlossen) Achtung: *nicht* Linear-Ring
`way` (area)	**`A`**	**`gao_geometry`**	Polygon (ohne holes)
`relation` Members nur nodes	**`P`**	**`geodata_point`** zusätzlich Buffer in **`gao_geometry`**	Point oder MultiPoint
`relation` Member nur ways (open oder closed)	**`L`**	**`geodata_line`** zusätzlich Buffer in **`gao_geometry`**	LineString, MultiLine-String
`relation` Members nur areas *oder* unterschiedliche Geometrien	**`A`**	**`gao_geometry`**	Polygon (mit holes), MultiPolygon
`relation` mit unbestimmbarer Geometrie	**`U`**	keine	keines

Das Geometrie-Kennzeichen **`U`** ("unknown") wird dann vergeben, wenn eine Geometrie einer Relation (also die Summengeometrie aller Member) nicht bestimmt werden kann. Die Gründe dafür können sein:

- Beim Zusammensetzen der Teilgeometrien ist ein Fehler passiert (üblicherweise in der JTS-Bibliothek).
- Die zusammengesetzte Geometrie ist zu groß. Als Beispiel: es gibt Relations der Art "alle Landstraßen von Franken". Die Summengeometrie würde ein große Fläche abdecken, daher sehr oft bei geometrischen Suchen überprüft werden. Demgegenüber hat das Summenobjekt keine inhaltliche Bedeutung.
- Das Summenobjekt ist nicht ausreichend durch einen geeigneten Namen oder eine Klasse beschrieben, d.h. es ist zweifelhaft, ob dieses Objekt als Summenobjekt ein gewünschtes Suchergebnis wäre.

Eine entsprechende Relation (insb. dessen Geometrie) kann später immer noch über die Tabelle **`relation`** aufgelöst werden.

2.5 Weitere geometrische Eigenschaften

Neben den Projektionen auf die Erdoberfläche werden in den Objekten weitere geometrische Eigenschaften gespeichert. Einerseits ist dies die Höhe (berechnet anhand der NASA-Höhenprofile) andererseits sind diese aus der Geometrie abgeleitete Größen (z.B. Flächeninhalt). Die abgeleiteten Größen ermöglichen es geometrische Bedingung in eine SQL-Abfrage zu integrieren, ohne die Geometrien einzeln auszuwerten.

Die folgende Tabelle zeigt alle weiteren geometrischen Eigenschaften eines Objektes.

Eigenschaft	Spalten der Tabelle `domain`	Beschreibung
Höhe	**`min_height`** **`max_height`**	Höhen gemäß NASA-Höhenprofile des niedrigsten und höchsten Punktes der Geometrie

Darüber hinaus gibt es im depro-Format folgende Eigenschaften:

Eigenschaft	Spalten der Tabelle `domain`	Beschreibung
Flächeninhalt	**`area_size`** **`gao_area`**	Flächeninhalt in m^2 (**`area_size`**) und $Grad^2$ (**`gao_area`**). Es wird der Flächeninhalt der Area-Geometrien verwendet (z.B. auch Punkt- und Liniengeometrien haben einen Flächeninhalt>0)
Linienlänge	**`line_length`**	Für Liniengeometrien: die Länge der Linie in Metern.
Minimaler umgebender Kreis	**`bounding_circle_lat`** **`bounding_circle_lon`** **`bounding_circle_rad_m`** **`bounding_circle_rad_deg`**	Der minimal umgebende Kreis der Area-Geometrie, gegeben durch Zentrum (lat, lon) und Radius (ausgerechnet in Grad und Metern). Bei Punkt- und Liniengeometrien wird die (vergrößerte) Area-Geometrie verwendet.
Enthaltener Punkt	**`inside_point_lat`** **`inside_point_lon`**	Ein Punkt, der Mitglied der Geometrie ist. Dieser Punkt ist der naheliegendste Punkt der Geometrie zum Zentrum des minimal umgebenden Kreises. Es kann auch das Zentrum des minimal umgebenden Kreises sein, wenn es zur Geometrie gehört.
Minimal umgebendes Rechteck	**`gao_mbr_n`** **`gao_mbr_w`** **`gao_mbr_s`** **`gao_mbr_e`**	Das minimal umgebende Rechteck, gegeben durch vier Grenz-Linien.
Anzahl	**`cnt_geom_points`** **`cnt_geom_components`**	Anzahl der Geometriepunkte und Geometriekomponenten.

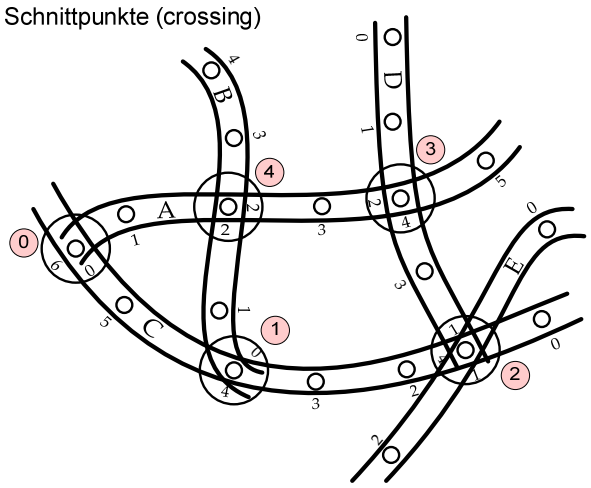
2.6 Topologien

2.6.1 Generelles

Straßenobjekte können sich zwar in Kreuzungen überlappen, aus der Tabelle **`domain`** kann man das aber nicht erkennen. Für die Routenplanung ist daher die Information über die Topologie erforderlich. Diese wird in den Tabellen **`link`** und **`crossing`** zur Verfügung gestellt

- **`crossing`** (Abb. 4): Für jede Kreuzung und jede einmündende Straße gibt es einen Eintrag in dieser Tabelle. Jede **`CrossingID`** spezifiziert eindeutig einen Kreuzungspunkt zwischen zwei oder mehr Straßen, aber das Tupel (**`CrossingID`**, **`PosNr`**) ist eindeutig für die Tabelle.

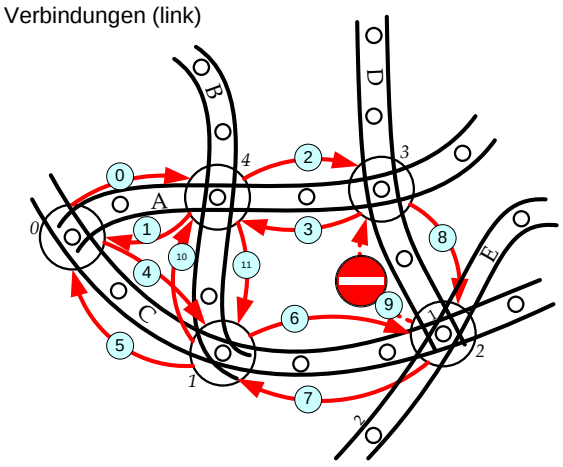
- **`link`** (Abb. 5): Für jede *gerichtete* Verbindung zwischen zwei Kreuzungspunkten gibt es einen Eintrag. Zwei verbundene Kreuzungspunkte werden durch zwei **`link`**-Einträge repräsentiert. Gibt es mehrere Wege zwischen zwei Kreuzungen (eher selten), sind es sogar 4, 6, etc. **`link`**-Einträge.



CrossingID	PartNr	DomainID	PosNr
0	0	A	0
0	1	C	6
1	0	B	0
1	1	C	4
2	0	C	1
2	1	D	4
2	2	E	1
3	0	A	4
3	1	D	2
4	0	A	2
4	1	B	2

Bedeutungen
CrossingID: laufende Nummer der Kreuzung
PartNr: laufende Nummer der Straße in dieser Kreuzung (0…)
DomainID: gao_id der Straßendomain
PosNr: Punktnummer (0…) des LineStrings der Straßendomain für diese Kreuzung
Pos (nicht im Bild): Long/Lat-Position der Kreuzung

Abb. 4: Modellierung von Straßen-Kreuzungen



LinkID	CrossingIDfrom	PartnrFrom	CrossingIDto	PartnrTo	DomainID	Length [m]	LSIClass	Tag
0	0	0	4	0	A	50	32611	
1	4	0	0	0	A	50	32611	
2	4	0	3	0	A	99	32612	
3	3	0	4	0	A	99	32612	
4	0	1	1	1	C	31	32615	
5	1	1	0	1	C	31	32615	
6	1	1	2	0	C	9	32613	
7	2	0	1	1	C	9	32613	
8	3	1	2	1	D	80	32611	
9	2	1	3	1	D	80	32611	C
10	1	0	4	1	B	66	32617	
11	4	1	1	0	B	66	32617	

Bedeutungen
LinkID: laufende Nummer der Verbindung
CrossingIDfrom: Start-Kreuzung
PartnrFrom: Einmündungsnummer Start
CrossingIDto: End-Kreuzung
PartnrTo: Einmündungsnummer Ende
PosFrom (nicht im Bild): Long/Lat Pos der Start-Kreuzung
Length: Länge der Verbindung in Metern
LSIClass: LSI-Klasse der Straße zur Verbindung
Tag: Weitere Beschreibung
<Blank>: nichts besonderes
‚C': Gegen die Einbahnstraße – so darf nicht gefahren, nur gegangen werden

Abb. 5: Modellierung von Straßen-Segmenten

Eine beispielhafte Benutzung dieser Tabellen für die Wegeplanung:

- Angenommen, man startet in einer bestimmten Straße (repräsentiert durch einen Eintrag in der **`domain`**-Tabelle). Dann hat diese Straße eine bestimmte **`DomainID`**.
- Man erhält alle Kreuzungen dieser Straße, indem man die Zeilen aus **`crossing`** ausgibt, die auf diese Domain verweisen. Diese Einträge haben bestimmte **`CrossingID`**s.
- Weiterführende Straßen, die von diesen Kreuzungen abgehen, erhält man, indem man die **`link`**-Tabelle nach solchen Einträgen fragt, deren **`crossingIDfrom`** aus der gesuchten Liste von Kreuzungen stammt. Diese **`link`**-Einträge verweisen wiederum auf Ziel-Kreuzungen (**`crossingIDto`**) sowie auf das **`domain`**-Objekt, zu dem dieser Straßenabschnitt gehört. Darüber hinaus beinhaltet ein **`link`**-Objekt Routing-relevante In-

formationen wie die Länge in Metern, die Maximalgeschwindigkeit, den Straßentyp (**`LSIClass`**), sowie ggfs. Einbahnstraßenrichtungen.

Mit diesen Informationen kann man z.B. eine Routenplanung über A* durchführen.

2.6.2 Sackgassen

Die Idee hinter einer **`crossing`** ist die Verbindung zwischen zwei oder mehr Straßen. Das bedeutet aber, dass die Endpunkte von Sackgassen nicht durch einen **`crossing`**-Eintrag berücksichtigt werden. Bei strenger Auslegung der Bedeutung einer Kreuzung würde sich eine Situation wie in Abb. 6 links dargestellt ergeben.

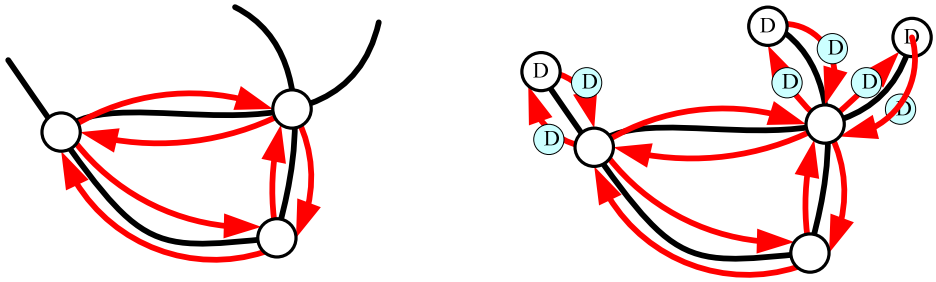


Abb. 6: Berücksichtigung von Sackgassen

Für Routing-Aufgaben ist diese Situation zuerst sinnvoll. Möchte man aber in einer Sackgasse eine Route beginnen oder enden, hat man das Problem, dass dieser Straßenabschnitt nicht durch ein **`link`**-Objekt repräsentiert wird. Als Konsequenz kann man die Kosten oder Maximalgeschwindigkeiten nicht berechnen. Deshalb wurden in einem zusätzlichen Generierungsschritt alle *Endpunkte* von Straßen, die keine Kreuzungen sind, als **`crossing`** hinzugenommen. Es ergibt sich daher schließlich die Situation in Abb. 6 rechts.
`crossing`-Einträge, die auf dieser Art hinzugenommen wurden, haben eine Kennzeichnung **`deadend`**='D' und können so leicht in der Datenbank erkannt werden. Darüber hinaus sind **`link`**-Einträge, die ein solches **`crossing`**-Objekt verbinden, auch mit der Kennzeichnung **`deadend`**='D' markiert. Solche **`crossings`** und **`links`** können nie in der Mitte eines kürzesten Wege vorkommen, sondern nur am Anfang oder Ende.

Dazu folgende Bemerkungen:

- **`crossing`**-Einträge mit **`deadend`**='D' liegen nur auf einer einzelnen Straße und gehören nur zu einem einzelnen **`link`**-Eintrag.
- **`link`**-Einträge mit **`deadend`**='D' verbinden eine **`crossing`** mit **`deadend`**='D' und eine mit **`deadend`**=<blank>. (**`link`**-Einträge, die zwei **`crossings`** mit **`deadend`**='D' verbinden, werden derzeit nicht generiert.)
- Alle **`link`**-Einträge einer Straße decken *alle* Wegpunkte der Straßengeometrie ab. Als Beispiel: Eine Straße hat 100 Wegpunkte, dann gibt es beispielsweise einen **`link`** für Wegpunkte 0…11, 11…33, 33...65 und 65....99. Nur die **`crossings`** der Wegpunkte 0 und 99 können (müssen aber nicht) die Kennzeichnung **`deadend`**='D' haben.
- Für Straßen mit *n* Wegpunkten gibt es grundsätzlich je eine **`crossing`** für Wegpunkt 0 und *n*-1.

2.6.3 Routing unter Einbeziehung von Flächen

Traditionelle Routing-Verfahren berücksichtigen nur Linien-Objekte. In den Rohdaten befinden sich jedoch auch Flächen-Objekte, die als "Wege" benutzt werden können, z.B. Fußgängerzonen oder Parkplätze.

Zur Vorbereitung entsprechender Routing-Verfahren wurde die Tabelle **areacrossing** eingerichtet. Diese Tabelle stellt lediglich die experimentellen Grundlagen für weitere Verfahren bereit. Weitere Tabellen müssen später folgen.
Die Tabelle **areacrossing** hat dieselbe Struktur wie **crossing** und auch eine ähnliche Bedeutung. Allerdings werden hier nur Kreuzungen zwischen

- zwischen Linien- und Flächenobjekten sowie
- zwischen Flächenobjekten untereinander

aufgenommen. Abb. 7 demonstriert, welche Kreuzungen gemeint sind (dunkel dargestellt).

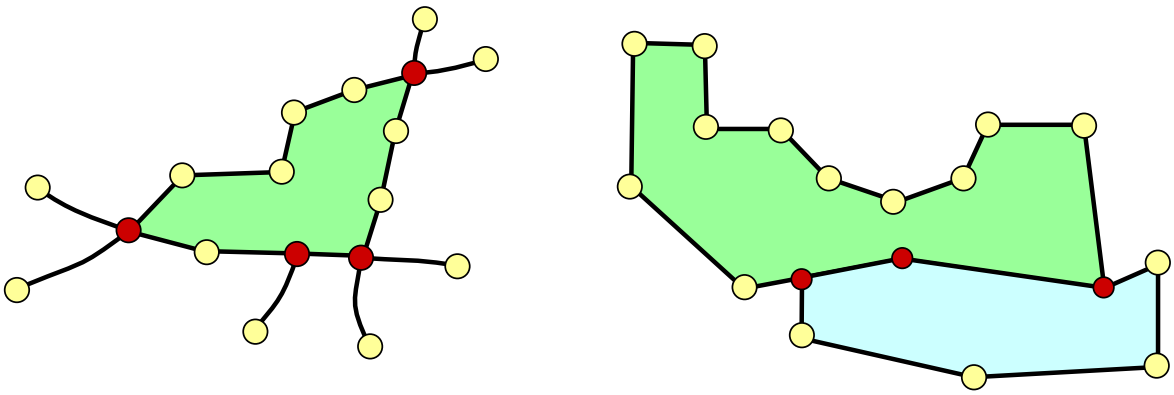


Abb. 7: Vorbereitung für Routing mit Flächen

Genau wie **crossing** entsprechen mehrere Einträge mit einer ID einer Kreuzung der realen Welt. Ein einzelner Eintrag identifiziert eine Einmündung bzw. ein Flächenkontakt. Von allen Einträgen in **crossing**/**areacrossing** mit derselben ID muss es mindestens einen Eintrag geben, der zu einer Fläche gehört (gäbe es nur Linien-Objekte, wäre diese Kreuzung ja schon durch die Tabelle **crossing** hinreichend beschrieben). Damit hat eine Kreuzung zwischen Linien und Flächenobjekt einen Eintrag in **crossing** und einen in **areacrossing** mit derselben ID.

Ein Mechanismus (**areanav**) beim Import generiert für alle Flächen, die Routen beinhalten können (z.B. Fußgängerzonen), künstlich kürzeste Wege innerhalb der Fläche. Damit können weitere Ansätze der Routenplanung komplett auf ein Wegenetz aufbauen und müssen die Flächen nicht gesondert behandeln. Diese künstlich angelegten Objekte haben folgende Belegung:

tags_ organizational	Fest mit **source=generated by areanav group** belegt
importid	**AN**<4-stellige laufende Nummer> <IMPORTID der Fläche>, z.B. **AN0001_W44129997**

Zusätzlich wird die Fläche mit folgendem Zusatz markiert:

tags_ organizational	Zusatz **info=considered by areanav group**

3 Weitere Datenquellen

Die reinen Daten aus OSM werden beim Import durch weitere Quellen ergänzt, um den Informationsgehalt und die Qualität zu verbessern.

3.1 NASA-Höhenprofile

OSM-Daten enthalten nur die 2D-Projektionen von Geometrie-Punkten auf die Erdoberfläche. Möchte man die Höhe integrieren, muss man die Höheninformation aus anderen Quellen verwenden.
Das NASA-Projekt *ASTER* (*Advanced Spaceborne Thermal Emission and Reflection Radiometer*) hat die Erdoberfläche vermessen und stellt diese Information zur Verfügung. Pro Flächenelement von 1° x 1° wurden 3601 x 3601 Höhenpunkte gemessen (entspricht einer Messung pro Bogensekunde in Latitude- und Longitude-Richtungen). Beim Import von OSM-Geometrien wird für jeden 2D-Punkt eine Höhe abgeleitet. Hierbei wird ein Interpolationsverfahren angewendet, da ein Geometrie-Punkt in der Regel nicht exakt auf einem eingemessenen Punkt liegt.

3.2 Global Administrative Areas

Leider können viele administrative Grenzen (insb. von Städten) nur schlecht über die reinen OSM-Daten importiert werden. Der Grund liegt meist darin, dass Grenzen in OSM aus Redundanzgründen aus Relations zusammengesetzt werden. Liegt nur ein kleiner Fehler in den Teilgeometrien vor, der verhindert, dass die Relations zu einem geschlossenen Polygon zusammengesetzt werden können, wird der Aufbau einer entsprechenden Geometrie verhindert, bzw. es entsteht eine falsche (in der Regel viel zu kleine) Geometrie.
Als Lösung wurden die wichtigsten Grenzen aus einer weiteren Quelle importiert: *Global Administrative Areas* (gadm.org). Die importierten Grenzen werden hier *Admin Shapes* genannt. Das Vorgehen hierbei:

- Gibt es zu einem Admin Shape eine hinreichend ähnlich Domain (gemessen an Name, LSI-Klasse und Geometrie), so wird das Admin Shape ignoriert.
- Gibt es zu einem Admin Shape eine ähnliche Domain (gemessen an Name und LSI-Klasse) mit einer signifikant unterschiedlichen Geometrie, so wird das Admin Shape importiert und die existierende Domain bekommt eine LSI-Klasse vom Typ **`OBSOLETE`** (z.B. **`OBSOLETE_KREISFREIE_STADT_LANDKREIS`**). Innerhalb des Domain-Eintrags der Admin Shape wird im Feld **`tags_link`** ein Verweis auf alle obsoleten Domains hinterleg.

Einträge in der Tabelle **`domain`**, die auf Admin Shapes basieren, haben folgende spezielle Belegungen:

tags_link	Liste der Domain d_ids, die durch diesen Eintrag obsolete wurden, Format: **`obsoletes=<d_id>;<d_id>;…`** NULL wenn keine Domain obsolete wurde
tags_ organizational	Fest mit **`source=http://gadm.org/country,note=imported by adminShape; NOT osmdb`** belegt
importid	**`GADM`**<5-stellige laufende Nummer>, z.B. **`GADM00001`**, **`GADM00002`**

3.3 Wikipedia

Für viele Geo-Objekte aus OSM gibt es einen Verweis auf Wikipedia-Artikel. Liegt solch ein Verweis vor, wird der entsprechende Artikel beim Import geladen und interpretiert. Die Ziele:

- Es soll eine Kurzbeschreibung des Geo-Objektes in einen Eintrag der Tabelle **`domain`** hinterlegt werden.

- Die Liste der Eigenschaften (Tabelle **domain_properties**) wird um Angaben auf der Wikipedia-Seite erweitert. Hierzu wird ausgenutzt, dass viele Eigenschaften als Schlüssel-Werte-Paare in so genannten Infoboxen hinterlegt werden. Diese können automatisiert ausgelesen werden.

Damit die neuen Eigenschaften sich nahtlos in die Tabellenstruktur einfügen, wird zu jedem möglichen Eintragstyp einer Wikipedia-Infobox ein Eintrag in die Tabelle **properties** gemacht. Zurzeit belegen die Wikipedia-Einträge die **p_id** 10000-19999. Die Schlüssel tragen grundsätzlich das Präfix '**wikipedia:**'.

Tabellen, die durch Wikipedia-Einträge beeinflusst werden und die relevanten Spalten:

- **properties**:

p_id	Eine eindeutige ID der Eigenschaft, für wikipedia-Einträge 10000-19999
type	Ein Schlüsselwert, ähnlich einem Datentyp, z.B. **INT**, **STRING**, **METERS**, **KM2**
key	Der Eigenschaftsname **wikipedia:**<infobox-Bezeichnung> z.B. **wikipedia:Einwohnerzahl**
unit	Die Einheit die von **type** abhängt, z.B. **m**, **km/2**, **m²**

- **domain_properties**:

d_id	Die Domain, zu dem Wikipedia-Eintrag
slot	ist sonst die Nummer des LSI-Eintrags für diese Eigenschaft, bei Wikipedia-Eigenschaften grundsätzlich -1
p_id	Die ID der Eigenschaft in **properties**
value	Der Wert-Text in der Wikipedia-Infobox (mit allen Zusätzen)
int_value	Wenn aus dem Wert eine ganze Zahl gewonnen werden konnte
double_value	Wenn aus dem Wert eine Kommazahl gewonnen werden konnte

- **domain**:

wikipedia_description	Der Kurzbeschreibungstext dieser Domain aus Wikipedia

4 Datenbank-Tabellen

In diesem Kapitel werden noch einmal die Tabellenstrukturen aller Tabellen dargestellt.

4.1 domain

Die Geo-Objekte sind in der Tabelle **domain** gespeichert.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
d_id	integer	Ja	Eindeutige ID	123456
basetype	character(1)	Ja	Basistyp der Domain: C: konkret R: Relation	C
lsiclass1	integer	Ja	Klassennummer gemäß LSI	35151
lsiclass2	integer	Ja	Weitere Klassennummer gemäß LSI (oder 0)	0
lsiclass3	integer	Ja	Weitere Klassennummer gemäß LSI (oder 0)	0

realname	character varying(256)	Nein	Echter Name	Uferstraße
nametype	integer	Ja	Wie "gut" ist der echte Name (1: sehr gut: >=100: künstlich vergeben)	2
wikipedia_description	character varying(5000)	Nein	Beschreibung aus Wikipedia	Die Stadt Nürnberg…
geometry	character(1)	Ja	Art der Geometry A: Fläche (Polygon) L: Linie C: Geschlossene Linie P: Punkt U: unbekannt	L
tags	character varying(3000)	Nein	OSM-Tags, die Eigenschaften des Objektes beschreiben	highway=residential, maxspeed=30
dmetaphone_primary	character varying(6)	Nein	Double Metaphone (primary) von realname	AFRSTR
dmetaphone_alternate	character varying(6)	Nein	Double Metaphone (alternate) von realname, wenn von primary verschieden	AFRSTR
tags_name	character varying(3000)	Nein	OSM-Tags, die den Namen des Objektes beschreiben	name=Uferstraße
tags_addr	character varying(3000)	Nein	OSM-Tags, die die Post-Adresse beschreiben	addr:city=Mönchengladbach, addr:country=DE, addr:housenumber=65, addr:postcode=41061, addr:street=Hindenburgstraße
tags_relative	character varying(3000)	Nein	OSM-Tags, die den Zusammenhang zu anderen Objekten beschreiben	is_in=Enzkreis,Karlsruhe,Baden-Württemberg,Bundesrepublik Deutschland,Europe
tags_link	character varying(3000)	Nein	OSM-Tags, die Referenzen (meist URLs) enthalten	source:ref=http://wiki.openstreetmap.org/wiki/Import/Catalogue/Strassen_NRW
tags_organizational	character varying(3000)	Nein	OSM-Tags, die Verwaltungsinformationen enthalten	osmarender:renderName=no
tags_others	character varying(3000)	Nein	weitere, bisher nicht klassifizierte OSM-Tags	TMC:cid_58:tabcd_1:Class=Area, TMC:cid_58:tabcd_1:LCLversion=8.00,TMC:cid_58:tabcd_1:LocationCode=530
importid	character varying(64)	Nein	Eindeutige ID von der Datenquelle (z.B. OSM) vergibt. Implizites Namensschema: GADM...: generiert durch "adminshapes" N...: Nodes W... : Ways R... : Relations AN…: generiert durch "areanav"	W4999499
min_height	double precision	Nein	Höhe des tiefsten Punktes der Geometrie in m	375.6
max_height	double precision	Nein	Höhe des höchsten Punktes der Geometrie in m	411.5

Im PostGIS-Format gibt es zusätzlich folgende Spalte:

Name	**Datentyp**	**Nicht NULL?**	**Bemerkung**	**Beispiel**
geom	geography (GeometryZ,4326)	Nein	Geometrie (Punkt, Linie oder Multipolygon)	POLYGON Z ((10.8775174 49.6948161 269.29562037119, …

Im depro-Format gibt es zusätzlich folgende Spalten:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
area_size	double precision	Nein	Flächeninhalt in m^2. Bei Linien oder Punkten wird die berechnete Flächengeometrie herangezogen.	500.23
bounding_circle_lat	double precision	Nein	Zentrum Latutide des minimal umgebenden Kreises	52.13654
bounding_circle_lon	double precision	Nein	Zentrum Longitude des minimal umgebenden Kreises	9.78656
bounding_circle_rad_m	double precision	Nein	Radius in Meter des minimal umgebenden Kreises	500.2
bounding_circle_rad_deg	double precision	Nein	Radius in Grad des minimal umgebenden Kreises	0.00014
inside_point_lat	double precision	Nein	Ein Punkt der Geometrie	52.13654
inside_point_lon	double precision	Nein	Ein Punkt der Geometrie	9.78656
line_length	double precision	Nein	Länge einer Liniengeometrie in m	110.5
cnt_geom_points	integer	Nein	Anzahl aller Geometriepunkte der "einfachsten" Geometrie (also z.B. der Punkt-Geometrie, wenn diese existiert)	1
cnt_geom_components	integer	Nein	Anzahl der Geometrie-Komponenten (z.B. Hülle, Löcher oder isolierte Linien)	1
geodata_line	bytea	Nein	Für geometry='L' oder 'C': WKB-serialisierte Liniengeometrie	*Binärdaten*
geodata_point	bytea	Nein	Für geometry='P': WKB-serialisierte Punktgeometrie	*Binärdaten*
gao_geometry	bytea	Nein	Für geometry='A': WKB-serialisierte Polygongeometrie, sonst Polygongeometrie des Buffers	*Binärdaten*
gao_index1	bigint	Nein	GAO-Index	505774560
gao_index2	bigint	Nein	Verschobener GAO-Index	1690493
gao_area	double precision	Nein	Flächeninhalt in $Grad^2$ der Geometrie (geometry)	4.292638e-008
gao_mbr_n	double precision	Nein	Bounding-Box-Grenze Nord	52.4417433
gao_mbr_w	double precision	Nein	Bounding-Box-Grenze West	9.670365
gao_mbr_s	double precision	Nein	Bounding-Box-Grenze Süd	52.441335
gao_mbr_e	double precision	Nein	Bounding-Box-Grenze Ost	9.67088833

4.2 domain_properties

Hier werden die Eigenschaften zu jedem domain-Eintrag abgelegt. Die Obermenge von Eigenschaften wird durch die LSI-Klasse bestimmt. Davon müssen in einem konkreten Fall aber nicht alle Belegungen vorliegen.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
d_id	integer	Ja	Zu welcher Domain gehört diese Eigenschaft	1015081
slot	integer	Ja	Zu welchem LSIClass_Eintrag (1, 2, 3) der Domain gehört diese Eigenschaft. -1 wenn Eigenschaft nicht aus der LSIClass gebildet (z.B. aus Wikipedia)	2
p_id	integer	Ja	Um welche Eigenschaft handelt es sich (Verweis auf Tabelle properties)	10
value	character varying(3000)	Ja	Wert der Eigenschaft	kebab
int_value	integer	Nein	Der Zahlenwert, wenn value (abzüglich aller White-Spaces) eine ganze Zahl darstellt	123
double_value	double precision	Nein	Der Zahlenwert, wenn value (abzüglich aller White-Spaces) eine Fließkommazahl darstellt	123.45

4.3 domain_addresses

Hier werden die Adress-Tags zu jedem domain-Eintrag abgelegt. Diese liegen zwar schon zusammengefasst in der Spalte tags_addr in der Tabelle vor, sind dort aber nur schwer in Bedingungen formulierbar. Die Tabelle **`domain_addresses`** legt zu jeder Domain so viele Zeilen ab, wie es Adress-Tags gibt.
Für einen Eintrag wird der OSM-Schlüssel selbst (z.B. addr:street) nicht abgelegt, sondern ist nur über einen Verweis mit Hilfe der Spalte **`a_id`** zugreifbar.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
d_id	integer	Ja	Zu welcher Domain gehört dieser Adress-Eintrag	1015081
a_id	integer	Ja	Um welchen Schlüssel handelt es sich, Verweis auf die Tabelle address_tags	7
value	VARCHAR(3000)	Ja	Wert des Adress-Eintrags	Grünreutherstraße

4.4 relation

Diese Tabelle verwaltet die m-n-Beziehung zwischen Domains untereinander. Eine Datenzeile verknüpft eine Domain vom Typ "relation" mit einer anderen Domain.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
d_id	integer	Ja	Die Relation-Domain (basetype='R') zu der ein Member-Objekt gehört	123456
importid	character varying(64)	Ja	Die Relation-Domain (basetype='R') zu der ein Member-Objekt gehört (Referenz auf importid). Dieser Eintrag wird noch aus historischen Gründen geführt. In Zukunft sollten alle Joins mit der Domain-Tabelle bezüglich Relation-Domain über d_id gehen.	R318
membernr	integer	Ja	Die laufende Reihenfolgenummer dieses Members (1, 2, 3…)	3
member_d_id	integer	Nein	Das Domain-Objekt, das zu einer Relation gehört	12345688
memberimportid	character varying(64)	Ja	Das Domain-Objekt, das zu einer Relation gehört (Referenz auf importid). Dieser Eintrag wird noch aus historischen Gründen geführt. In Zukunft sollten alle Joins mit der Domain-Tabelle bezüglich member über member_d_id gehen.	W35820867
role	character varying(64)	Nein	Die Rolle der Domain innerhalb der Relation gemäß OSM-Relation-Rollen-Tags	outer

4.5 is_ins

Für jeden domain-Eintrag gibt es 0 oder 1 Eintrag in der Tabelle **`is_ins`**. Eine Zeile dokumentiert die Zugehörigkeit zu übergeordneten Strukturen wie Stadt oder Bundesland.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
d_id	integer	Ja	Verweise auf die eindeutige ID in der domain-Tabelle	123456
is_in1	integer	Nein	Staat	123456
is_in1b	integer	Nein	Staat (weiterer Eintrag)	null
is_in1c	integer	Nein	Staat (weiterer Eintrag)	null
is_in2	integer	Nein	Bundesland	123456
is_in2b	integer	Nein	Bundesland (weiterer Eintrag)	null
is_in2c	integer	Nein	Bundesland (weiterer Eintrag)	null
is_in3	integer	Nein	Regierungsbezirk	123456
is_in3b	integer	Nein	Regierungsbezirk (weiterer Eintrag)	null
is_in3c	integer	Nein	Regierungsbezirk (weiterer Eintrag)	null
is_in4	integer	Nein	Kreisfreie Stadt, Landkreis	123456
is_in4b	integer	Nein	Kreisfreie Stadt, Landkreis (weiterer Eintrag)	null
is_in4c	integer	Nein	Kreisfreie Stadt, Landkreis (weiterer Eintrag)	null
is_in5	integer	Nein	Amtsgemeinde	123456
is_in5b	integer	Nein	Amtsgemeinde (weiterer Eintrag)	null
is_in5c	integer	Nein	Amtsgemeinde (weiterer Eintrag)	null
is_in6	integer	Nein	Stadt, Gemeinde	123456
is_in6b	integer	Nein	Stadt, Gemeinde (weiterer Eintrag)	null
is_in6c	integer	Nein	Stadt, Gemeinde (weiterer Eintrag)	null
is_in7	integer	Nein	Stadtteil	123456
is_in7b	integer	Nein	Stadtteil (weiterer Eintrag)	null
is_in7c	integer	Nein	Stadtteil (weiterer Eintrag)	null
is_ina	integer	Nein	Weitere Zugehörigkeiten	123456
is_inb	integer	Nein	Weitere Zugehörigkeiten	123456
is_inc	integer	Nein	Weitere Zugehörigkeiten	123456

4.6 is_near

Für jeden domain-Eintrag gibt es 0 oder 1 Eintrag in der Tabelle **`is_near`**. Eine Zeile dokumentiert die Entfernung zu Beschriftungsobjekten von übergeordneten Strukturen wie Stadt oder Bundesland.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
d_id	integer	Ja	Verweise auf die eindeutige ID in der domain-Tabelle	123456
is_near1	integer	Nein	Staat	123456
is_near1b	integer	Nein	Staat (weiterer Eintrag)	null
is_near1c	integer	Nein	Staat (weiterer Eintrag)	null
is_near2	integer	Nein	Bundesland	123456
is_near2b	integer	Nein	Bundesland (weiterer Eintrag)	null
is_near2c	integer	Nein	Bundesland (weiterer Eintrag)	null
is_near3	integer	Nein	Landkreis, Stadt, Kleinstadt, Dorf	123456
is_near3b	integer	Nein	Landkreis, Stadt, Kleinstadt, Dorf (weiterer Eintrag)	null
is_near3c	integer	Nein	Landkreis, Stadt, Kleinstadt, Dorf (weiterer Eintrag)	null
is_near4	integer	Nein	Vorort, Stadtteil	123456
is_near4b	integer	Nein	Vorort, Stadtteil (weiterer Eintrag)	null
is_near4c	integer	Nein	Vorort, Stadtteil (weiterer Eintrag)	null
is_neara	integer	Nein	Weitere Zugehörigkeiten	123456
is_nearb	integer	Nein	Weitere Zugehörigkeiten	123456
is_nearc	integer	Nein	Weitere Zugehörigkeiten	123456
distance1	integer	Nein	Land	123456
distance1b	integer	Nein	Land (weiterer Eintrag)	null
distance1c	integer	Nein	Land (weiterer Eintrag)	null
distance2	integer	Nein	Bundesland	123456
distance2b	integer	Nein	Bundesland (weiterer Eintrag)	null
distance2c	integer	Nein	Bundesland (weiterer Eintrag)	null
distance3	integer	Nein	Regierungsbezirk	123456
distance3b	integer	Nein	Regierungsbezirk (weiterer Eintrag)	null
distance3c	integer	Nein	Regierungsbezirk (weiterer Eintrag)	null
distance4	integer	Nein	Kreisfreie Stadt, Landkreis	123456
distance4b	integer	Nein	Kreisfreie Stadt, Landkreis (weiterer Eintrag)	null
distance4c	integer	Nein	Kreisfreie Stadt, Landkreis (weiterer Eintrag)	null
distancea	integer	Nein	Weitere Zugehörigkeiten	123456
distanceb	integer	Nein	Weitere Zugehörigkeiten	123456
distancec	integer	Nein	Weitere Zugehörigkeiten	123456

4.7 crossing

`crossing` bildet zusammen mit link die Straßen-Topologie ab. Eine Zeile innerhalb von **`crossing`** repräsentiert eine Einmündung in eine Kreuzung.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
id	bigint	Ja	Laufende Kreuzungsnummer (0, 1, 2…)	3
partnr	integer	Ja	Laufende Einmündungsnummer (0, 1, 2…)	2
d_id	integer	Ja	domain-Identifikator der Einmündung	123456
posnr	integer	Ja	Nummer des Punktes in der Geometrie der Einmündungs-domain (0, 1, 2…) (Domain muss Linie sein)	0
deadend	character(1)	Ja	Sackgassen-Kennzeichen: <blank>: Kreuzung ist echte Kreuzung zwischen 2 oder mehr Straßen D: Kreuzung ist Endpunkt eine Sackgasse	D

Im PostGIS-Format gibt es zusätzlich folgende Spalte:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
location	geography(Point,4326)	Nein	Kreuzungspunkt	POINT(10.2095607 49.0928941)

Im depro-Format gibt es zusätzlich folgende Spalten:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
long	double precision	Ja	Longitude-Koordinate des Kreuzungspunktes	10.5944166
lat	double precision	Ja	Latitude-Koordinate des Kreuzungspunktes	48.05440166

4.8 link

Eine Zeile innerhalb von **`link`** entspricht einem Wegabschnitt von Kreuzung zu Kreuzung.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
id	bigint	Ja	Laufende Nummer (0, 1, 2…)	5
crossing_id_from	bigint	Ja	id der Startkreuzung	3146
crossing_partnr_from	integer	Ja	partnr der Startkreuzung	1
crossing_id_to	bigint	Ja	id der Endkreuzung	456789
crossing_partnr_to	integer	Ja	partnr der Endkreuzung	0
d_id	integer	Ja	domain-Identifikator der Domain, zu der dieses Segment gehört	123456
meters	integer	Ja	Länge des Segments in Metern	361
lsiclass	integer	Ja	Klassennummer gemäß LSI der zugehörigen Domain	35141
tag	character(1)	Ja	Einbahnstraßen-Klassifikation <blank>: in diese Richtung befahrbar C: in diese Richtung nicht befahrbar, da gegen die Einbahnstraße, nur Fußgänger	C
maxspeed	integer	Ja	Maximalgeschwindigkeit in km/h oder 0 wenn unbekannt	50
deadend	character(1)	Ja	Sackgassen-Kennzeichen: <blank>: Link ist Link zwischen 2 oder mehr echten Kreuzungen D: Link geht in eine Sackgasse, d.h. eine der zugehörigen crossings hat auch das Kennzeichen deadend='D'	D

Im PostGIS-Format gibt es zusätzlich folgende Spalte:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
location_from	geography(Point,4326)	Nein	Start-Kreuzungspunkt	POINT(10.2095607 49.0928941)

Im depro-Format gibt es zusätzlich folgende Spalten:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
long_from	double precision	Ja	Longitude-Koordinate des Start-Kreuzungspunktes	10.5944166
lat_from	double precision	Ja	Latitude-Koordinate des Start-Kreuzungspunktes	48.05440166

4.9 areacrossing

areacrossing ist die experimentelle Vorbereitung für das Flächen-Routing. Eine Zeile innerhalb von **areacrossing** repräsentiert eine Einmündung in eine Kreuzung (von einem Linien- oder Flächenobjekt).

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
id	bigint	Ja	Laufende Kreuzungsnummer (0, 1, 2…)	3
partnr	integer	Ja	Laufende Einmündungsnummer (0, 1, 2…)	2
d_id	integer	Ja	domain-Identifikator der Einmündung	123456
posnr	integer	Ja	Nummer des Punktes in der Geometrie der Einmündungsdomain (0, 1, 2…) (Domain muss Linie oder Fläche sein, mindestens eine Fläche pro id)	0

Im PostGIS-Format gibt es zusätzlich folgende Spalte:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
location	geography(Point,4326)	Nein	Kreuzungspunkt	POINT(10.2095607 49.0928941)

Im depro-Format gibt es zusätzlich folgende Spalten:

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
long	double precision	Ja	Longitude-Koordinate des Kreuzungspunkt	10.5944166
lat	double precision	Ja	Latitude-Koordinate des Kreuzungspunkt	48.05440166

4.10 lsiclasses

Mit dieser Tabelle kann über ein join direkt der Klassifikationstext zu einer Domain ausgegeben werden, z.B.

```
select A.fullname, B.description from domain A, lsiclasses B
where A.lsiclass1=B.id limit 100;
```

Für eine konkrete Anwendung ist diese Tabelle eher unwichtig. Der Nutzen liegt eher in Testabfragen über SQL. Darüber hinaus kann die Datenbank den Stand der LSI-Klassen 'konservieren', der zum Zeitpunkt des Imports vorlag. Das ist deshalb wichtig, da sich die LSI-Klassen in den jeweiligen Quellen ständig entwickeln.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
id	integer	Ja	LSI-Klassennummer	20502000
id_to	integer	Ja	Bis zu welcher Klassennummer geht diese Klasse (inkl. alle Unterklassen)	20502999
level	integer	Ja	Ebene dieser Klasse (0: ganz unten)	3
token	character varying(64)	Ja	Eindeutiger String, um diese Klasse zu identifizieren	SHOP
description	character varying(64)	Ja	Beschreibender Text	"Geschäft (unspezifiziert)"
simplified	character varying(64)	Ja	Beschreibender Text, vereinfacht für die Ausgabe	"Geschäft"
synonym1	character varying(64)	Nein	Liste möglicher Synonyme (Suchwort)	"Geschäft"
synonym2	character varying(64)	Nein		"Laden"
synonym3	character varying(64)	Nein		null
synonym4	character varying(64)	Nein		null
synonym5	character varying(64)	Nein		null
synonym6	character varying(64)	Nein		null
synonym7	character varying(64)	Nein		null
synonym8	character varying(64)	Nein		null

4.11 properties

Mit dieser Tabelle kann man Informationen über Eigenschaften bekommen. Die Eigenschaften selbst sind in einer eigenen Tabelle **`domain_properties`** abgelegt.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
p_id	integer	Ja	Eindeutige ID (0…)	66
type	character varying(64)	Ja	String, der den Datentyp der Eigenschaft spezifiziert	BOOL
key	character varying(64)	Ja	Der Schlüssel	building
unit	character varying(20)	Nein	Einheit wenn vorhanden	km/h

4.12 address_tags

Mit dieser Tabelle sind die Schlüssel der vorkommenden Adress-Tags gemäß OSM (in der Regel Tags die **`addr:...`**) abgelegt. Die zugehörigen Werte sind in einer eigenen Tabelle **`domain_addresses`** abgelegt. Diese Tabelle verweist dann auf die Schlüssel anhand der Spalte **`a_id`**.

Für einige Schlüssel sind die zugehörigen **`a_id`**-Werte fest eingetragen. Anwendungen können sich auf diese Zuordnungen verlassen und müssen diese nicht mehr anhand des **`key`**-Eintrags testen.

Name	Datentyp	Nicht NULL?	Bemerkung	Beispiel
a_id	integer	Ja	Eindeutige ID (1…)	7
key	character varying(64)	Ja	Der Schlüssel	addr_street