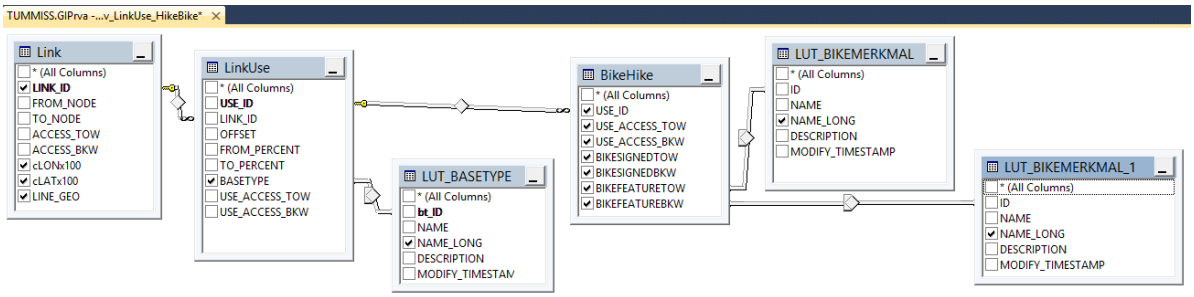
Import Daten der Graphenintegrations-Plattform (GIP) mit MS SSIS



1 Zusammenfassung

Die Graphenintegrations-Plattform (GIP) enthält österreichweit eine Vielzahl an Verkehrsinfrastrukturdaten in hoher Detailliertheit. Die GIP beinhaltet Daten für alle Verkehrsformen inkl. Schiffe, U-Bahnen, Seilbahnen, Busse, KFZ, Radfahrer, Fußgänger etc.

Alle 2 Monate werden die 318 Tabellen aus der Oracle Datenbank in einfach interpretierbare Formate exportiert. Besonders interessant ist der OGD Routingexport [1], der alle Verkehrsdaten für das Routen von Verkehrsteilnehmern, u.a. Radfahrer und Fußgänger, enthält.

Obwohl es behördliche Daten sind, gibt es keine Garantie auf Aktualität und Fehlerfreiheit. Eine gesunde Skepsis bei der Interpretation der Daten ist daher angebracht.

Design

Als Frontend der Daten wird QGIS verwendet. Die Daten selbst werden in einer SQL Spatial Datenbank vorgehalten, die mit dem ETL-Tool SSIS (SQL Server Integration Services) vollautomatisiert importiert werden. Der Schwerpunkt liegt auf Daten für Radfahrer und Fußgänger.

Performanz

Der Datenimport läuft in 2 Schritten ab: (i) Download der Daten von Open Data Österreich [2] und (ii) Import in eine MS SQL-Datenbank.

Data In	Data Out	Zeit	Aktivität
504 MB 1 ZIP	3,46 GB in 14 Files	5,1 min	Download ZIP Archiv mit ~16 MBit/sek und extrahieren
1,73 GB in 5 Files	23,5 Mio Zeilen in 5 Tabellen	2,9 min	5 CSV-Files in 5 Tabellen importieren
16,0 Mio Knoten	2,1 Mio Geo- Linien	2,5 min	Konvertierung lat/lon Linienzug zu räumlichen Daten

Trotz Standard-HW ist die Performance beeindruckend. Zum Vergleich einige Betreiberangaben [3] mit eigenen Daten:

- GIP Tabelle Link einspielen: 2 min Oracle mit SQL Loader - 21 s SQL Server mit SSIS

- GIP IDF Export einspielen: 7 min Postgres, 36 min Oracle - 2,9 min SQL Server (nur 1/2 IDF Export)

2. Entwicklung

Plattform für Entwicklung, Test und Debugging ist Visual Studio Community 2019.

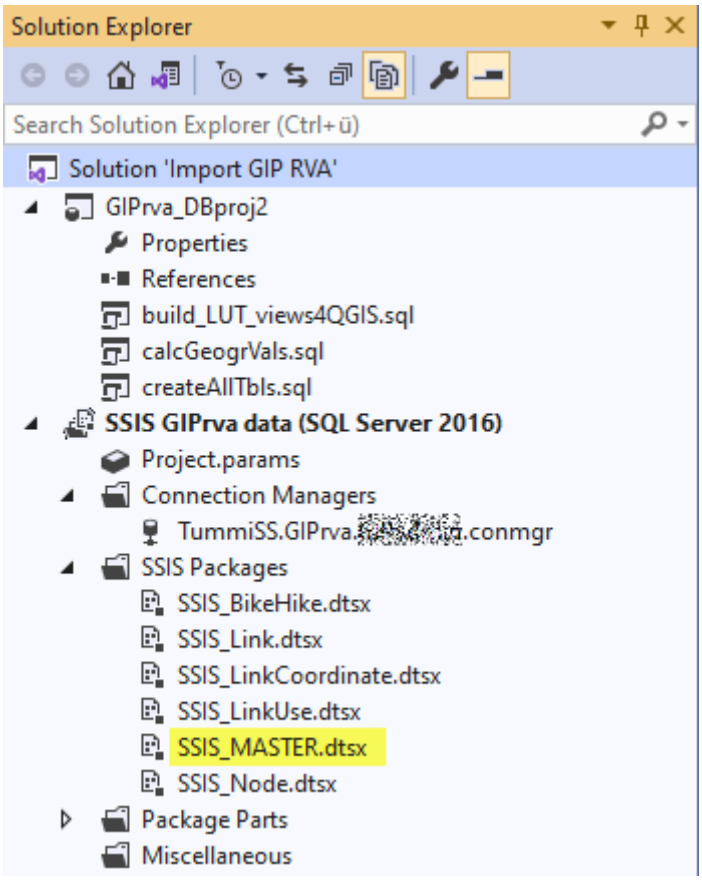
Entwicklungsumgebung

Client (Host)
PC: Intel i5-9600K @ 3,70 GHz; 32 GB RAM; 697 GB SSD; 1TB HD
Betriebssystem: Win 10 pro 64 bit
Visual Studio Comunity 2019 + SS Data Tools + SS Integration Services
VirtualBox V6.1

Virtual Database Server (Guest #1)
Windows Server 2012 R2
SS Management Studio 2014
SQL Server 2014 Developer (64-bit)

SSIS Projekt

Die Hauptkomponente des SSIS Projektes ist "SSIS_MASTER.dtsx". Mit Rechtsklick und Execute Package wird der Importprozess gestartet. Die übrigen SSIS_*-Pakete werden von der Hauptkomponente aufgerufen und importieren jeweils einen CSV-File.

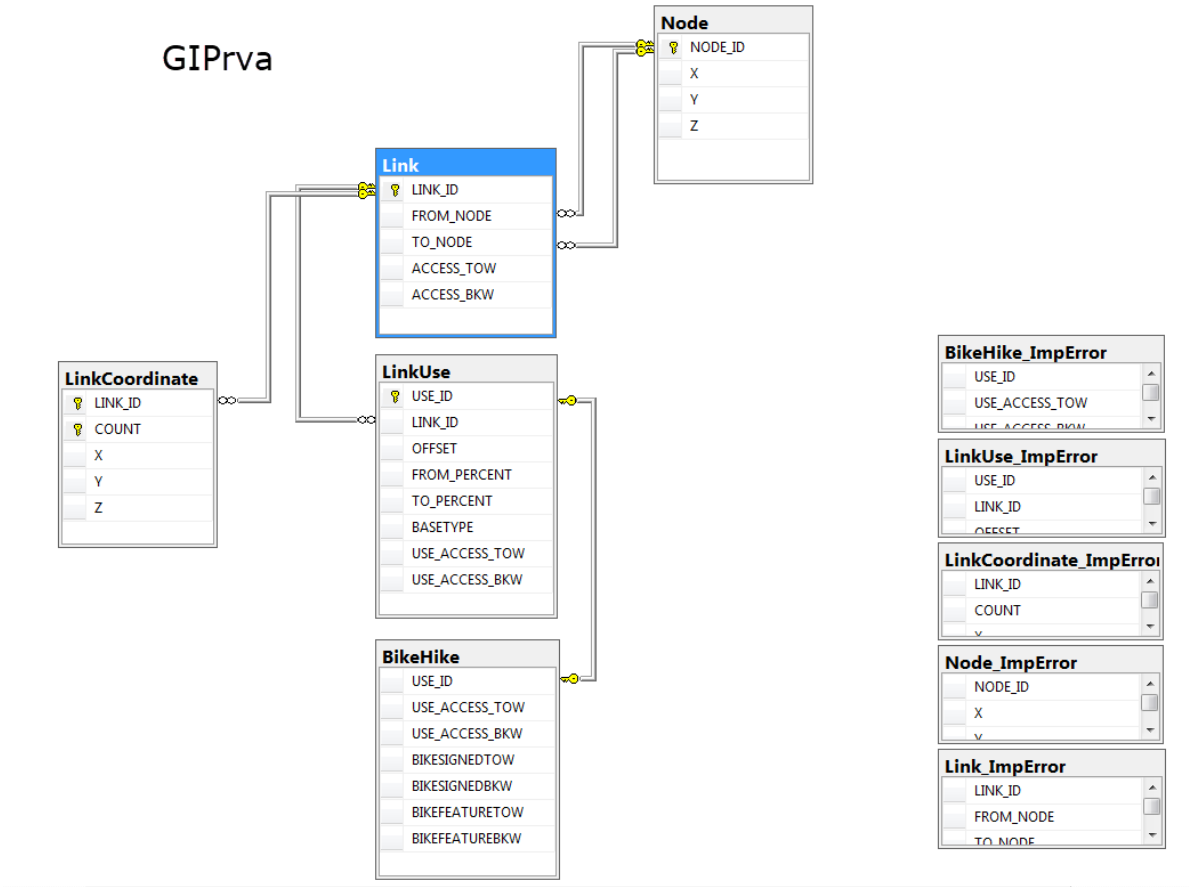


"createAllTables" ist ein Script, das die i.a. gefüllten Datentabellen löscht und neu aufbaut. Der Importprozess beginnt mit diesem Schritt. Script "calsGeogrVals" konvertiert die Links in einen räumlichen Datentyp. Mit diesem Schritt endet der Importprozess.

"build_LUT_views4QGIS" ist ein Script, das bei Bedarf manuell ausgeführt wird. Es generiert die Look-Up-Tabellen und einige beispielhafte Views für das Frontend QGIS.

Datenmodell

Die für diese Anwendung relevanten Daten sind in 5 Tabellen abgelegt. Fünf weitere Tabellen sammeln alle Datensätze, die beim Import einen Fehler hervorbringen.



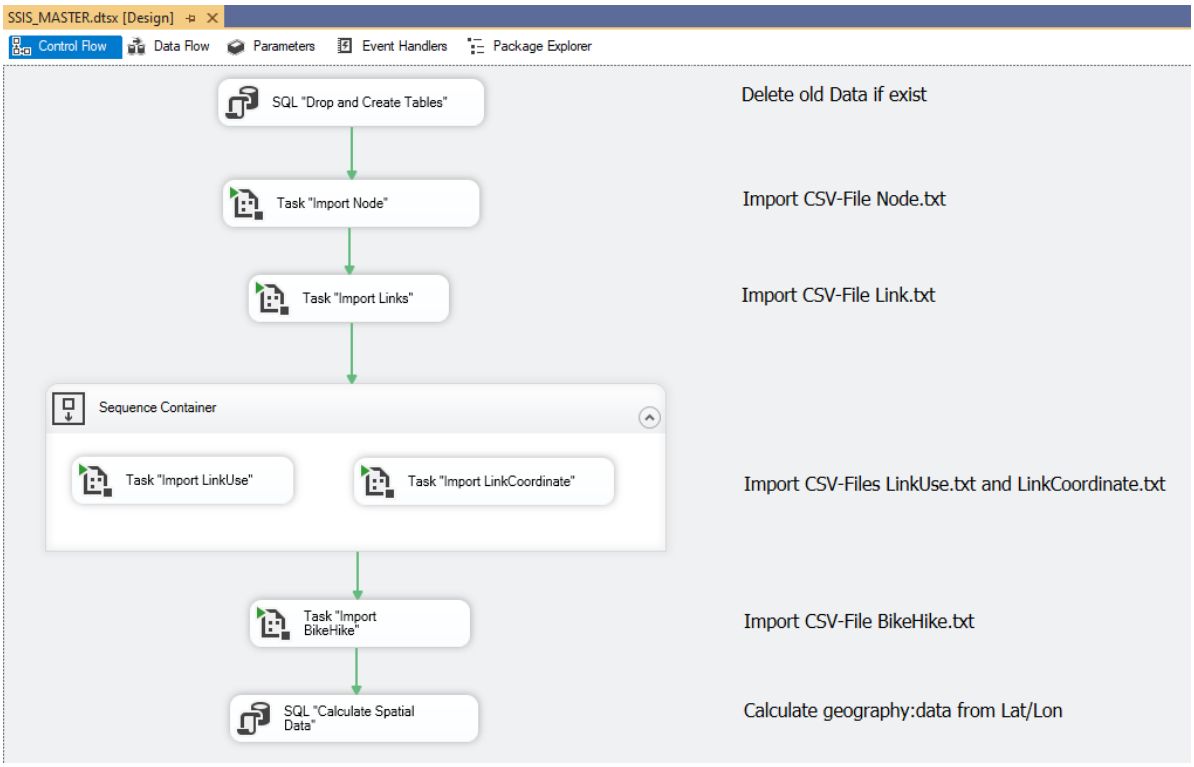
Alle Details zu diesen Objekten sind in [1] zu finden. Hier nur ein kurzer Überblick zum Verständnis:

Tabelle	Inhalt
Link	Anfangs- und Endpunkt des Wegabschnittes (Link) und über alle Nutzungsstreifen aggregierte Zugangsinformationen (Access) in und gegen die Link-Richtung
Node	Koordinaten des Link Anfangs- und Endpunktes
LinkCoordinate	Koordinaten aller Knoten auf dem Link
LinkUse	Informationen zu den einzelnen Nutzungsstreifen
BikeHike	Zusatzinformationen für Radfahrer und Fußgänger

Die Verknüpfung zwischen den Tabellen LinkUse und BikeHike ist vom Typ 1:1 bzw. genauer 1:1...0.

Ablaufsteuerung

Die Ablaufsteuerung definiert den zeitlichen Ablauf der einzelnen Aufgaben. Die grünen Pfeile definieren einen Weitergang bei "Erfolg" der Aufgabe. In dieser Ebene gibt es keine explizite Fehlerbehandlung.



Das Hauptpaket ruft 2 SQL-Tasks, zu Vor- und Nachbereitung der Datenbank und 5 Sub-Pakete, die jeweils eine Tabelle befüllen, auf.

Die Reihenfolge, in der die Tabellen befüllt werden, wird durch die referentielle Integrität bestimmt. Die erste zu befüllende Tabelle darf daher keinen Fremdschlüssel enthalten. Tabellen mit Fremdschlüssel können erst befüllt werden, wenn die zugehörigen Tabelle, wo die Fremdschlüssel Primärschlüssel sind, bereits befüllt wurden.

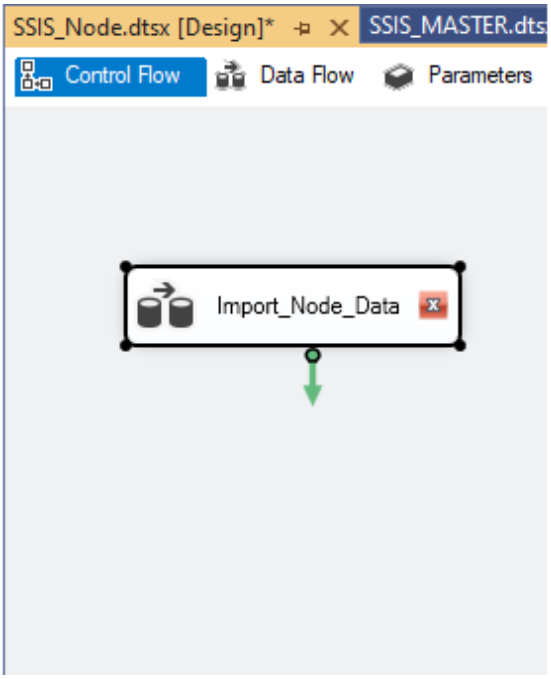
Die referentielle Integrität gewährleistet, dass nur konsistente Datensätze in die Datenbank gelangen. Der einfachere Ansatz, die Fremdschlüssel erst nach dem Import hinzuzufügen, funktioniert bestens, wenn die Datensätze vollständig konsistent sind. Aber ein einziger Fehler genügt, damit der nachträgliche Aufbau der Fremdschlüssel misslingt und die ganze Datenbank unbrauchbar wird.

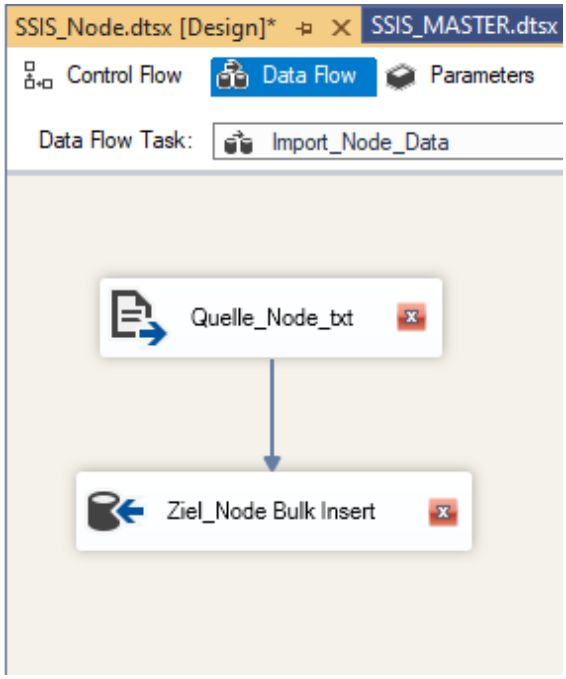
Beispiel SSIS Paket "Import Node" erstellen

In Visual Studio ist ein SSIS-Paket recht schnell erstellt. Die Schritte sind:

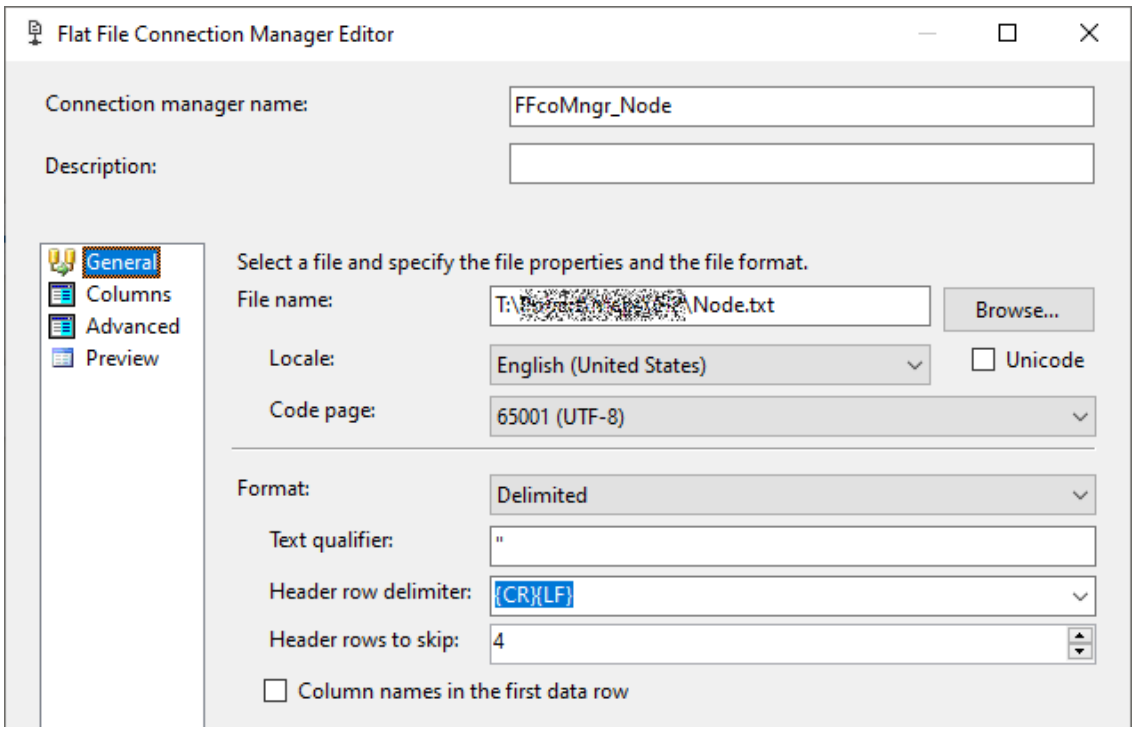
1. Im Solution Explorer: rechtsklick aus SSIS-Packages/ „New SSIS-Package“
2. umbenennen in "SSIS_Node.dtsx"
3. Datenflusstask in Control Flow Fenster ziehen, umbenennen in "Import_Node_Data"
4. Fenster „Data Flow“ anwählen
5. aus SSIS Toolbox/Other Sources „Flatfilequelle“ hinüberziehen und umbenennen in "Quelle_Node_txt"
6. aus SSIS Toolbox/Other destinations „OLE DB-Ziel“ hinüberziehen und umbenennen in "Ziel_Node Bulk Insert"

7. beide verbinden

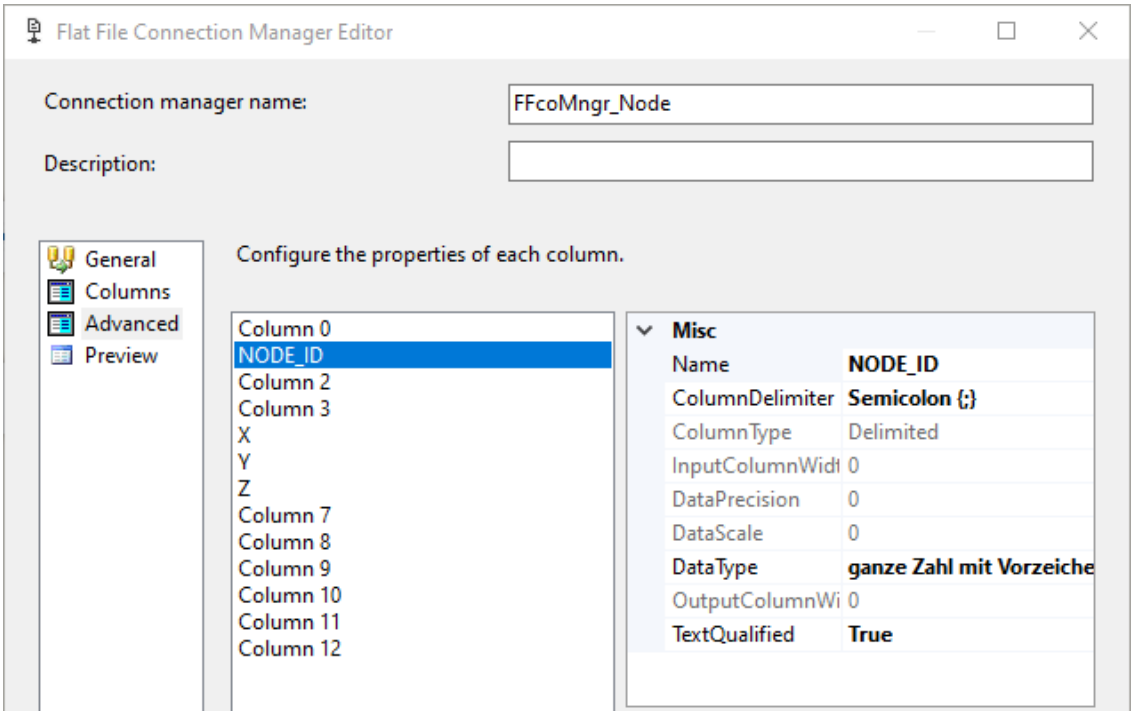




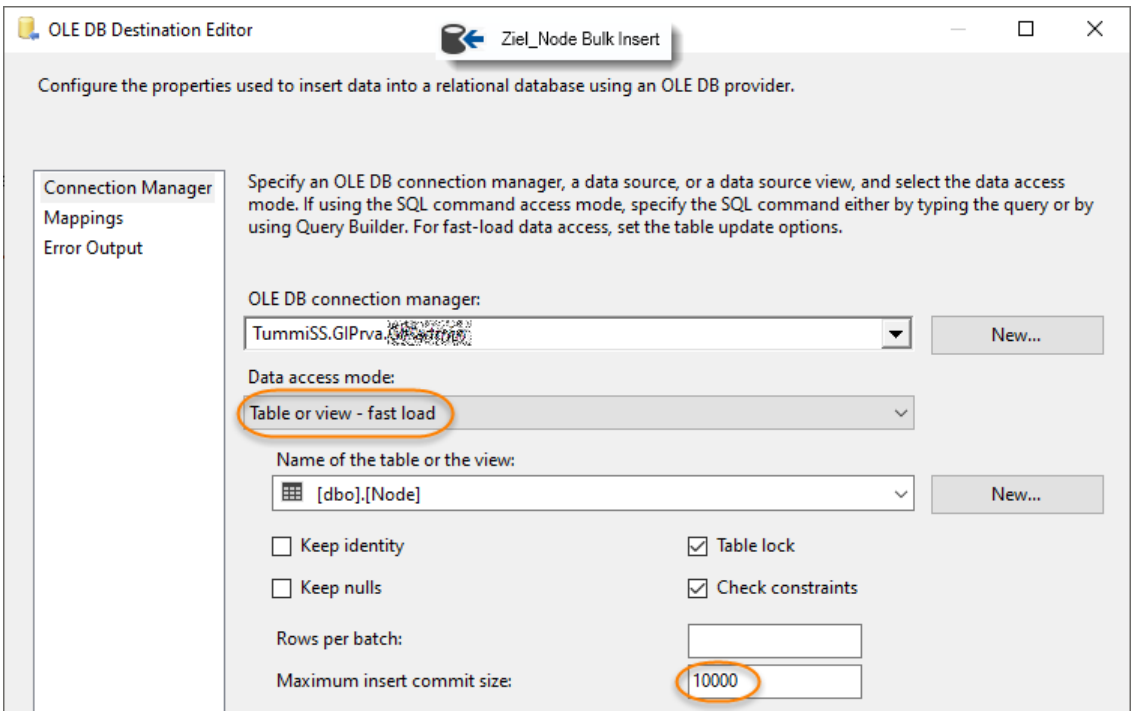
8. Im nächsten Schritt wird der Connection Managers der Quelle definiert. Dazu Doppelklick auf Quelle_Node_txt/ Flat file connection Manager/ New:



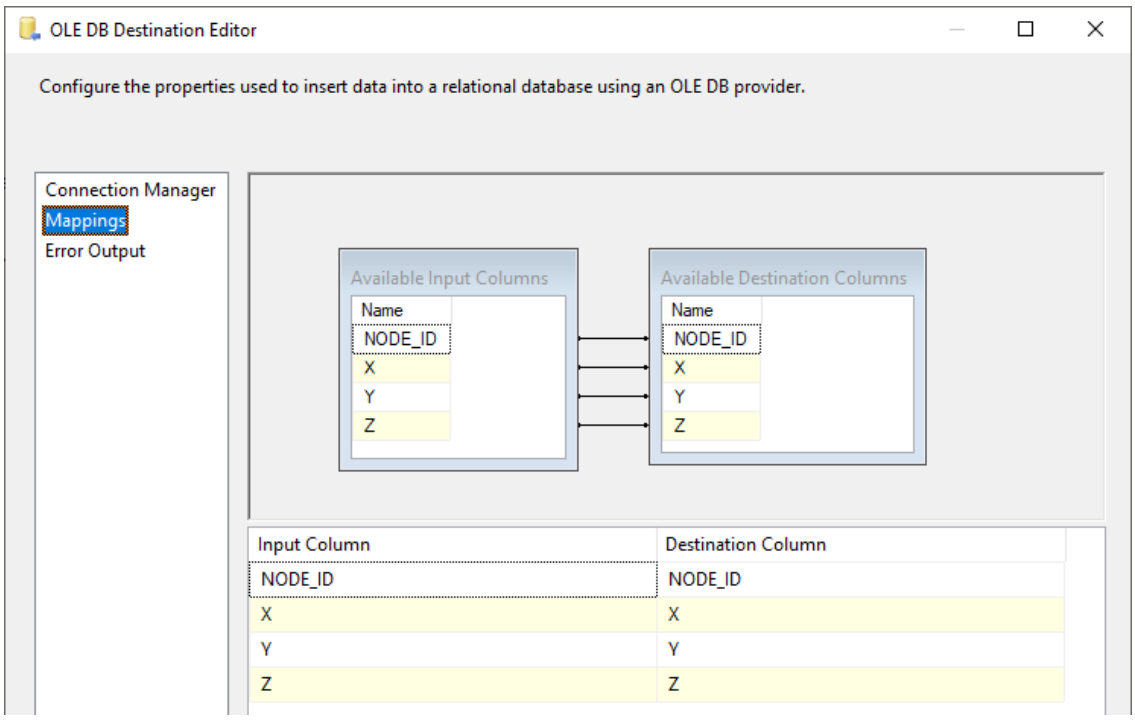
9. In „Advanced“ Namen und Datentyp der zu importieren Spalten definieren. Den Rest belassen:



10. Connection Managers des Zieles definieren. Dazu Doppelklick auf "Ziel_Node Bulk Insert". Die maximale Einfügungcommitgröße wird auf 10 000 Zeilen gesetzt:



11. Bei korrekter Benennung sollte das Mapping automatisch erfolgen:



Die Kernaufgabe - Lesen des CSV-Files und Schreiben in die DB-Tabelle - ist damit realisiert, das Paket ist im Prinzip lauffähig. Was noch fehlt ist die Fehlerbehandlung.

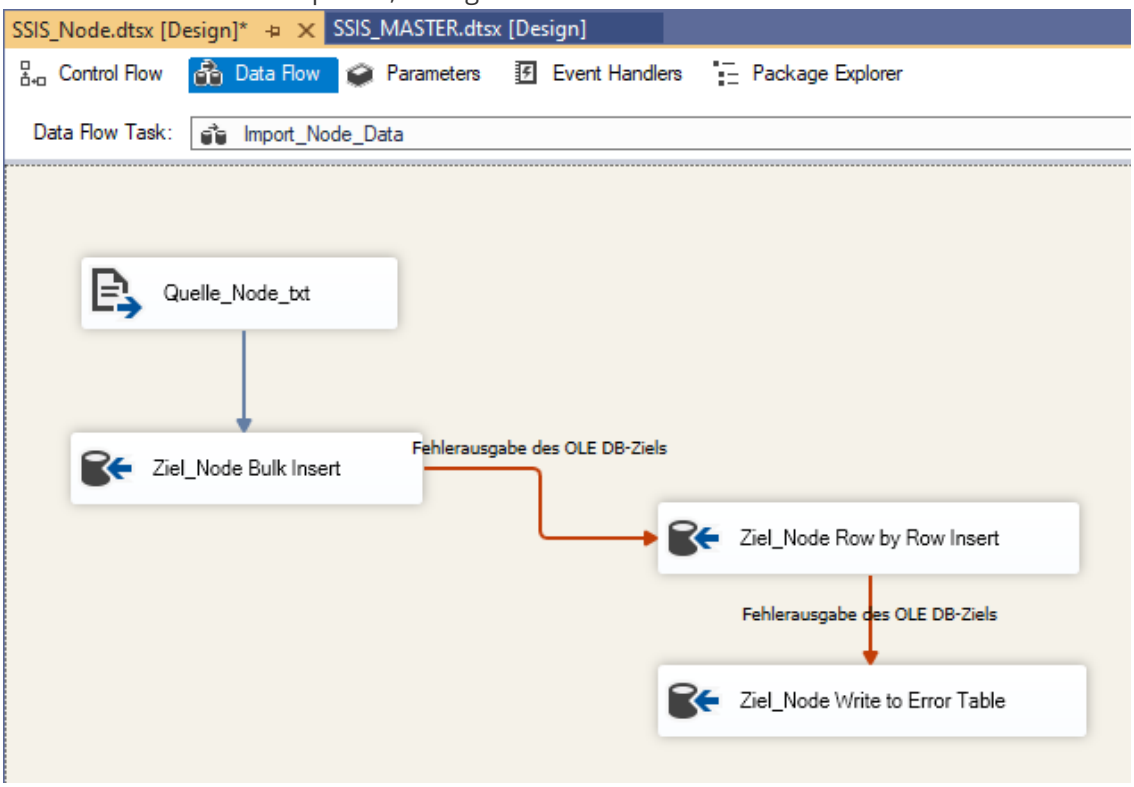
Fehlerbehandlung hinzufügen

Der Massenimport mit Fast Load ist durch den stapelweisen Datentransfer sehr schnell, hat aber den Nachteil, im Fehlerfall den ganzen Stapel zu verlieren. Eine Lösung besteht darin, den Import zu kaskadieren [4]:

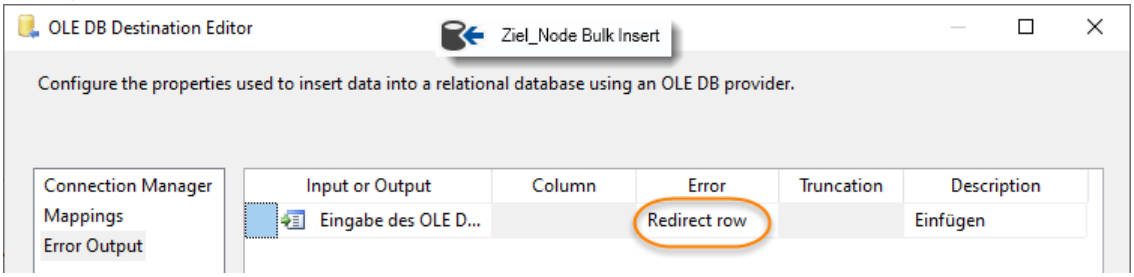
- Massenimport in 10 000er Stapeln
- Tritt darin ein Fehler auf, wird der 10 000er Stapel an ein Row by Row Importer weitergegeben
- dieser trägt alle OK-Zeilen in die Zieltabelle ein und die fehlerhaften Zeilen in die Fehlertabelle
- Fehlercode und Nr. der fehlerhaften Spalte werden ebenfalls in die Fehlertabelle geschrieben

Ein weiterer Vorteil ist die spezifischere Fehlerausgabe beim zeilenweisen Import.

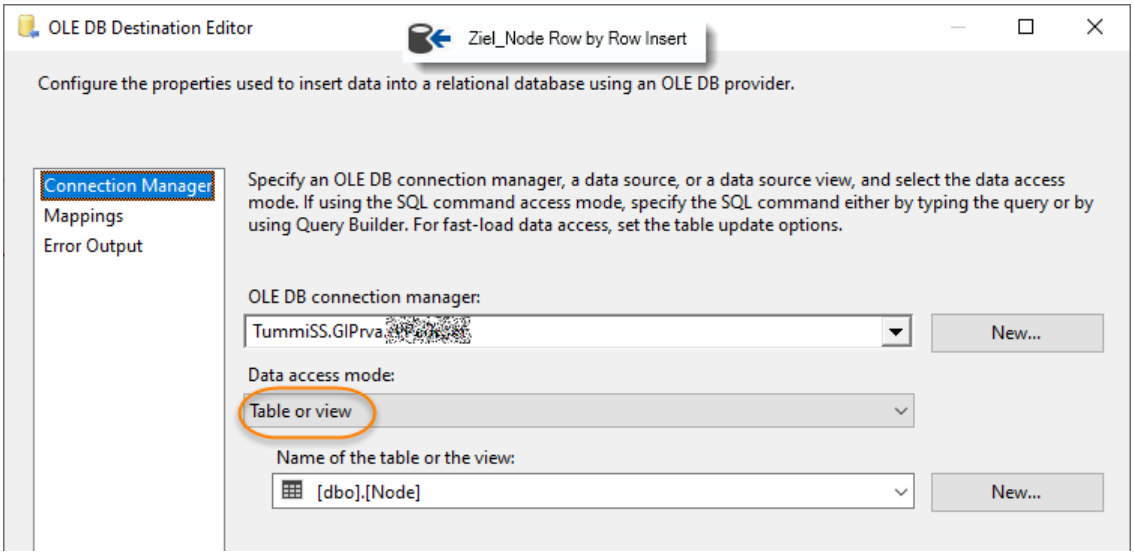
1. OLE DB-Ziel einfach 2x kopieren, einfügen und verbinden:

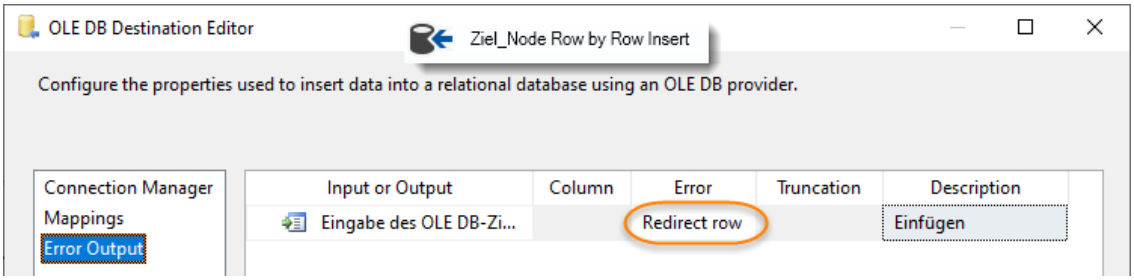


2. Konfigurieren "Ziel_Node Bulk Insert":

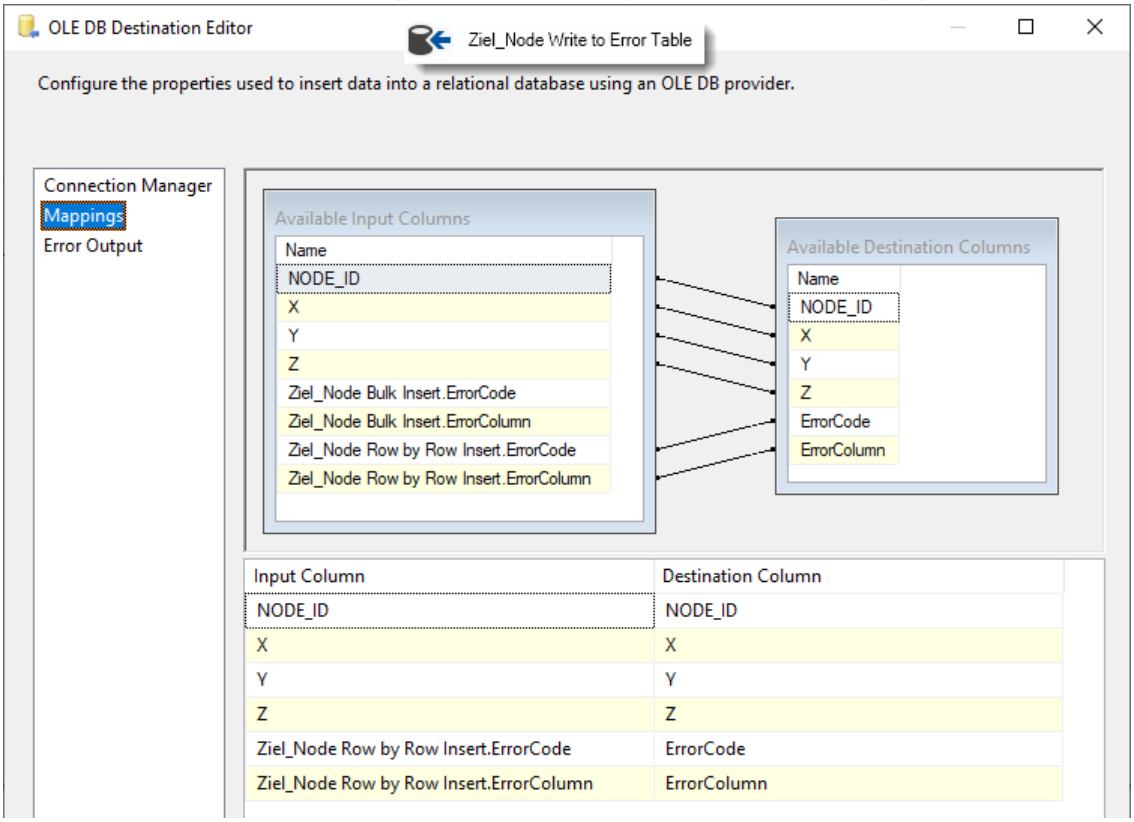


3. Konfigurieren "Ziel_Node Row by Row Insert"





4. Konfigurieren "Ziel_Node Write to Error Table". Connection Manager und Error Output wird wie bei "Ziel_Node Row by Row Insert" konfiguriert. In Mappings wird der Bulk Insert.Error verworfen und stattdessen der spezifischere Row by Row Insert.Error abgespeichert:



5. Fertig

Fallstricke beim Import

`Die` Klassiker sind Umlaute und der Kommapunkt. Für Umlaute verwendet GIP die UTF-8 Kodierung. In SSIS ist für Unicode (UTF-8) die code page 65001 und der Datentyp "Unicode (DT_WSTR)" zu verwenden, der Zeichensatz "Unicode" darf nicht angehakt werden. Das Dezimaltrennzeichen ist eine Garantie für graue Haare, wenn man die Interpretation dem System überlässt. Die Lösung heißt, das Gebietsschema (Locale) immer explizit zu setzen:

- Kommapunkt im CSV-File? - Im Flat File Connection Manager oder in den Eigenschaften der FlatFile-Quelle LocaleID= English(United States) setzen!
- SQL Server auf deutsch eingestellt? - in den Eigenschaften von OLE DB-Ziel LocaleID=German(Germany) setzen!

Konvertierung zu räumlichen Daten

Mit räumlichen Datentypen wird die Lage von Objekten, wie Punkt, Linie, Polygon etc. im Raum beschrieben. Seit SQL Server 2008 sind räumliche Datentypen in jeder Datenbank verfügbar. Nach dem Import liegt das Verkehrsnetz als Punktelinien mit Lat/Lon-Koordinaten vor. In SQL Server sind das LineString-Objekte mit dem Datentyp geography. Beispiel:

`LINESTRING (Lon Lat, Lon Lat, Lon Lat, ...) oder`

`LINESTRING (X Y,  X Y,  X Y, ...) oder`

`LINESTRING (15.2612702 48.5217664, 15.2615284 48.5217629, 15.261579 48.5217625)`

Zu beachten sind 2 Eigenheiten: (i) Punkt-Konstruktoren haben die Reihenfolge Lat/Lon und Linien-Konstruktoren Lon/Lat(!) und (ii) der räumliche Datentyp geht nur über Text. Beispiel für T-SQL Syntax:

`geography::STLineFromText('LINESTRING(Lon Lat, Lon Lat, Lon Lat)', SRID)`

Wobei SRID für Spatial Reference ID steht . SRID=4326 ist der Defaultwert und entspricht WGS84.

Views zur Geo-Konvertierung

Nach dem Import der CSV-Tabellen sind die Koordinaten der Links als Lat/Lon-Werte in der Datenbank abgelegt. In dieser Form sind sie nutzlos, da sie kein räumliches Objekt repräsentieren. Die Konversion erledigt die View "v_L2_Pts2Line" , die im letzten Schritt des Importprozesses ausgeführt wird.

Die View "v_L2_Pts2Line" baut auf einer weiteren View auf. Diese Teilung in zwei Arbeitsschritten ist prinzipiell nicht erforderlich, erhöht aber die Verständlichkeit und damit auch die Wartbarkeit.

Alle Views werden vom Script "createAllTbls.sql", am Beginn des Importprozesses, generiert.

View v_L1_allPts

Anfangs-/Endpunkte und Zwischenpunkte der Linien sind in 2 verschiedene Tabellen untergebracht. Um sie geordnet verbinden zu können, wird den Anfangskoordinaten die Ordnungszahl "0", den Zwischenpunkten die Ordnungszahlen "1", "2" ... und den Endpunkten die Ordnungszahl 99999 zugeordnet. Die 3 Ergebnisse werden mit Union vereinigt.

```
`CREATE VIEW [dbo].[v_L1_allPts]`
`AS`
`/* Get FROM-Coordinates, set COUNT=0 */`
`SELECT Link.LINK_ID, 0 AS [COUNT], NodeF.X, NodeF.Y`
`FROM            Link INNER JOIN`
                        `Node AS NodeF ON Link.FROM_NODE = NodeF.NODE_ID`

`/* Get TO-Coordinates, set COUNT=99999, UNION */`
`UNION ALL`
`SELECT        Link.LINK_ID, 99999 AS [COUNT], NodeT.X, NodeT.Y`
`FROM            Link INNER JOIN`
                        `Node AS NodeT ON Link.TO_NODE = NodeT.NODE_ID`

`/* Get all other Intermediate-Coordinates, UNION */`
`UNION ALL`
`SELECT        LINK_ID, [COUNT], X, Y`
`FROM            LinkCoordinate
```

View v_L2_Pts2Line

Die innere Abfrage erhält von der äußeren Abfrage die Link_ID. Pro Link_ID werden Anfangspunkt, Zwischenpunkte und Endpunkt geordnet verkettet und in eine Geo-Linie (geography LineString) konvertiert. Die String-Formung ist ein bisschen tricky:

1. Das innere Select liefert Zeilen der Form ',XX YY'

2. "FOR XML Path('')" am Ende einer Query sammelt die Abfrageergebnisse in 1 XML-Element. Das Ergebnis ist nun XML = ,XX YY,XX YY ...
3. Die Funktion STUFF(XML, 1,1, '') ersetzt 1 Zeichen auf Position 1 mit einem Leerstring, d.h. der "," am Beginn wird gelöscht. Das Ergebnis ist nun STUFF = XX YY,XX YY ...
4. Somit wird konvertiert: geography::STLineFromText(LINESTRING(XX YY,XX YY ...), 4326)

```
/*
* Create geography linestring from points (lat,lon)
* for each group LINK_ID
*
* Ref.: https://stackoverflow.com/questions/48241483
*/

CREATE VIEW [dbo].[v_L2_Pts2Line]
AS
SELECT        LINK_ID, geography::STLineFromText('LINESTRING('
                    + STUFF(
                            (
                             SELECT        ',' + CONCAT([X], ' ', [Y])
                             FROM            v_L1_allPts AS t2
                             WHERE        t1.LINK_ID = t2.LINK_ID
                             ORDER BY [COUNT]
                             FOR XML PATH('')
                            )
                            , 1, 1, '')
                     + ')'
                     , 4326) AS LINE_GEO
FROM            v_L1_allPts AS t1
GROUP BY LINK_ID;
```

View v_L3_centerFromTo

Diese View berechnet näherungsweise die geographischen Mitte der Links durch Mittelung der der Anfangs- und Endpunktkoordinaten. Die Koordinaten der Mittenpunkte werden noch mit 100 multipliziert und die Kommastellen entfernt.

```
/*
* Calculate Center Coordinates of lines
* to speed up spatial queries against a bounding box
*/
CREATE VIEW dbo.v_L3_centerFromTo
AS
SELECT        L.LINK_ID, CAST((N1.X + N2.X) * 50 AS INT) AS cLONx100, CAST((N1.Y
+ N2.Y) * 50 AS INT) AS cLATx100
FROM            dbo.Link AS L INNER JOIN
                         dbo.Node AS N1 ON L.FROM_NODE = N1.NODE_ID INNER JOIN
                         dbo.Node AS N2 ON L.TO_NODE = N2.NODE_ID
```

Damit kann ein kombinierter Zweispalten-Index auf diesen Punkt gelegt werden, der eine sehr schnelle Vor-Einschränkung des Datengebietes erlaubt. Das ist für das zügige Arbeiten mit QGIS enorm wichtig, da die Daten bei jedem Verschieben/Zoomen vom Server neu geholt werden. Testergebnisse:

Tab. 1: Ausführungszeiten von Test-Abfragen mit Filter und Index Variationen auf Tabelle "Link"

Filter	ohne Index	mit Index
im Bereich mit cLONx100, cLANx100 (4x4 km, 5131 Zeilen)	0,31s	0,17 s (X, Y komb. Spaltenindex)
im Kreis mit STBuffer (R=2186m, 5130 Zeilen)	2,6 min	0,22 s (Geo-Index)

T-SQL für den Geo-Index:

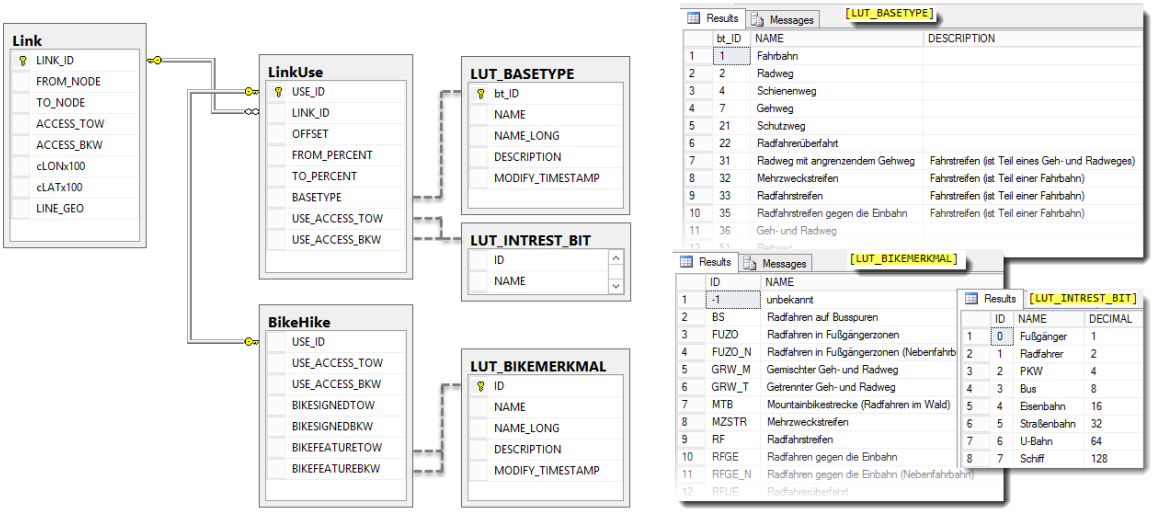
```
CREATE SPATIAL INDEX [spIdx_GIPLink] ON [dbo].[Link] ([LINE_GEO]) USING GEOGRAPHY_AUTO_GRID
```

Die Berechnung dauert etwa 3min, der zusätzliche Speicherbedarf ist erträglich. Zum Vergleich der Speicherbedarf der Indizes in % der Tabellengröße (796 MB):

1. Primärer Schlüssel "PK_Link": 0,4%
2. Kombinierter Index "cXY": 4%
3. Geo-Index "spIdx_GIPLink": 12%

3 Anwendungsunterstützung

Nach dem Import und der Aufbereitung der Geo-Daten stehen dem Anwender die Tabellen Link, LinkUse und BikeHike für Auswertungen zur Verfügung. Drei zusätzliche Nachschlag-Tabellen unterstützen bei der Interpretation der Tabellenwerte.



Beispiel Befahrbarkeit

Die Befahrbarkeit der Nutzungsstreifen ist als Bitmaske in ACCESS_TOW (in Zeichenrichtung) und ACCESS_BKW (gegen der Zeichenrichtung) angegeben. Die Befahrbarkeits-Bits sind in der Tabelle "LUT_INTEREST_BIT" definiert. In der Tabelle "Link" sind die Befahrbarkeits-Bits aller Nutzungsstreifen aggregiert.

Mit einfachen Bit-Operationen können die Bit-Werte in leichter lesbare Variablen-Werte transformiert werden. Beispiel für Bit 3 "PKW", Dezimal-Wert 4:

```
( access_tow | access_bkw & 0x4 ) / 4         AS cACC_Car,
( ( access_tow & 0x4 ) - ( access_bkw & 0x4 ) ) / 4  AS cACC_Car_Oneway
```

cACC_Car: 0=Fahrverbot; 1=zumindest in einer Richtung erlaubt
cACC_Car_Oneway: 0=keine Einbahn; 1= Einbahn in Zeichenrichtung; -1= Einbahn gegen Zeichenrichtung

Vollständige Query "v_Link_BikeUse":

```
SELECT link_id,
    access_tow,
    access_bkw,
    clonx100,
    clatx100,
    line_geo,
    ( access_tow | access_bkw & 0x2 ) / 2                AS cACC_BIKE,
    ( ( access_tow & 0x2 ) - ( access_bkw & 0x2 ) ) / 2  AS cACC_BIKE_ONEWAY,
    access_tow | access_bkw & 0x1                        AS cACC_Hike,
    ( access_tow | access_bkw & 0x4 ) / 4                AS cACC_Car,
    ( ( access_tow & 0x4 ) - ( access_bkw & 0x4 ) ) / 4  AS cACC_Car_Oneway
 FROM  dbo.link
```

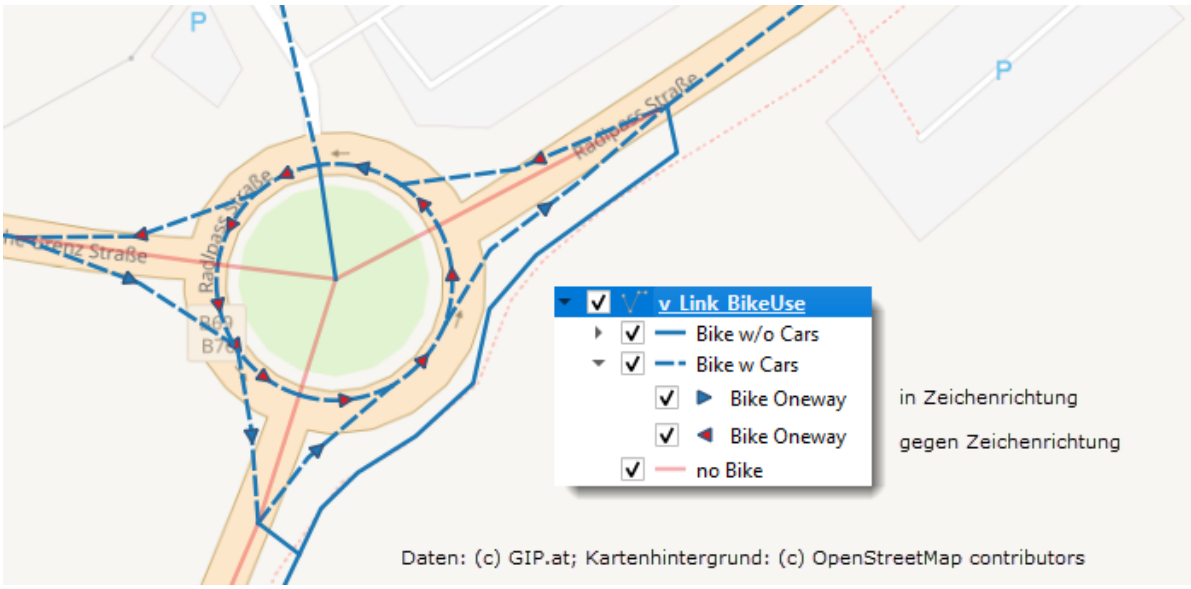
Beispiel QGIS

Die Verbindung zum SQL-Server wird im Data Source Manager mit "Add MSSQL/New" hergestellt. Der "Provider Specific Filter Expression":

`(cLONx100 BETWEEN 1523-4 AND 1523 + 4) AND (cLATx100 BETWEEN 4668-4 AND 4668 + 4)`

schränkt den Datentransfer auf den interessierenden Bereich ein und sorgt damit für einen schnellen Bildaufbau.

Abb 1: Visualisierung der über alle Fahrstreifen aggregierten Zugangsberechtigungen. Pfeile zeigen Einrichtungsfahrbahnen an. Pfeil in rot heißt: Zeichenrichtung ist entgegengesetzt der erlaubten Fahrrichtung.



4 Referenzen

[1] GIP.at: Dokumentation Intermodales Verkehrsreferenzsystem Österreich, http://open.gip.gv.at/ogd/0_dokumentation_gipat_ogd.pdf

[2] Open Data Österreich: A - Routingexport: ZIP Archiv mit den einzelnen IDF Tabellen aufgesplittet, http://open.gip.gv.at/ogd/A_routingexport_ogd_split.zip

[3] D. Geroe: DER GIP.AT OGD-EXPORT NEUIGKEITEN, TIPPS & TRICKS, AGIT 2019 Salzburg, https://www.gip.gv.at/assets/downloads/AGIT2019_OGD_Workshop.pdf

[4] Joydeep Das: Error Handling With OLEDB Destination – Fast Load (2017) http://sqlknowledgebank.blogspot.com/2017/02/error-handling-with-oledb-destination.html