Developing the TOTUS Data System

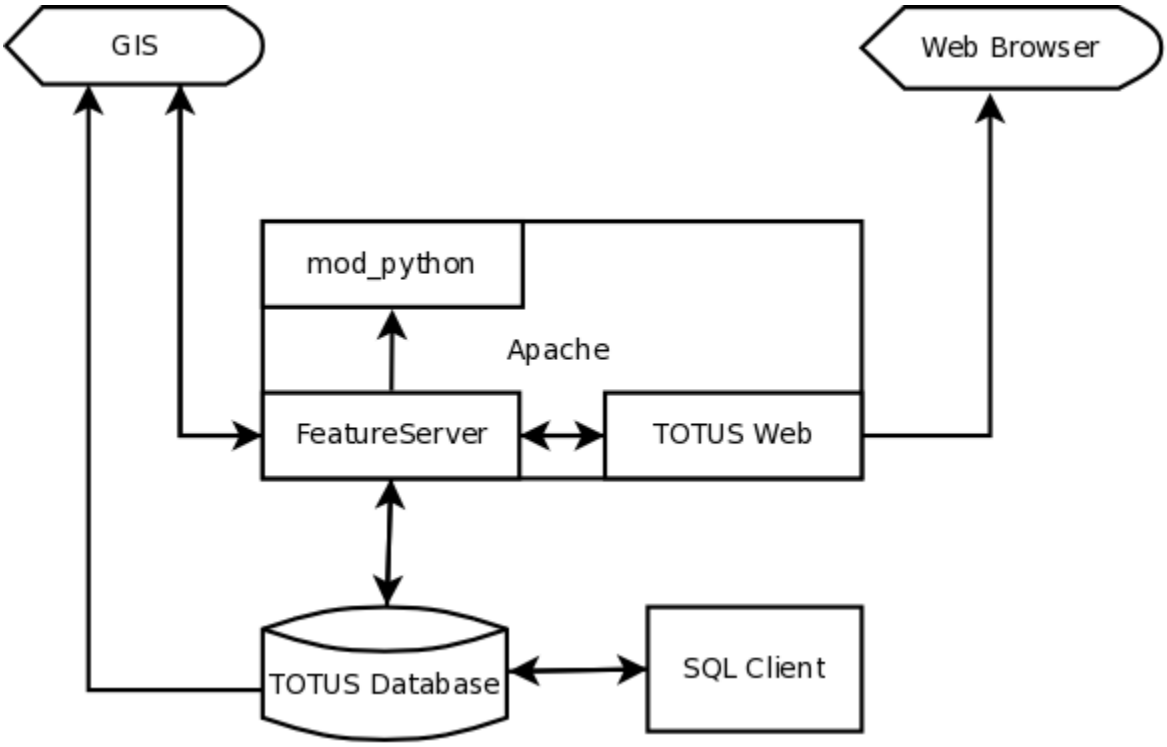
Introduction: Towards Sustainable Urban Forms (TOTUS)

The changes that modern urban forms undergo have far reaching consequences and implications. The addition of a complex motorway through a once quiet suburb will have huge impacts on the community that resides in such an urban form. Some changes, like the addition of a new community in a once rural district, lead to impacts on infrastructure provisioning by the additional demand placed on the road, water and power network; which feedbacks into long term management of public and ecosystem health.

The aim of the TOTUS system is to assist in identifying relationships and interactions between some aspects of a changing urban form. The requirements for the underlying system are:

- the system should be open and accessible to all stakeholders
- it should not be constrained to one vendor software or protocol for interacting with the system
- a light version of the system should be easily distributed
- the system should be flexible and allow deployment to any geographical region (if TOTUS's minimum requirements are met)

With that mind the TOTUS system has been constructed as follows:



The TOTUS system follows the client-server architecture and is a three-tier model consisting of a data store, a data access layer and a presentation layer accessible through standard HTTP protocols.

At the core of TOTUS is a relation database that stores the data in a manner that allows identifying and modelling urban forms interactions. The data it contains include:

- a route-able spatial road network
- traffic model output
- demographic information
- exposure model
- energy model

The TOTUS data store is a spatial database and accessible using a variety of Geographical Information Systems (GIS). In addition the data access layer supports the Web Feature Service (WFS) of the Open GIS Consortium (OGC) for querying the underlying data sets.

Due the relational nature of the data store the most flexible way to interact with the system is to use Structure Query Language (SQL) queries, but that depends on inherent knowledge of the underlying system. This document will assist in gaining a more detailed understanding of the data tier.

TOTUS Infrastructure

TOTUS is built upon open source technologies to ensure it can be easily distributed and accessed by all stake holders.

PostgreSQL

PostgreSQL is the leading open source, standards compliant, enterprise class database management system in the world. It is an object-relational database system that brings object oriented and relational design together in one system. It also has the ability to be used as a document store with the addition of the eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) data types. It boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions, online or hot backups, a sophisticated query planner (optimizer), and write ahead logging for fault tolerance. PostgreSQL supports international character sets and is locale-aware for sorting and formatting data. It is highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate.

PostgreSQL is truly a developer's system and supports more than a dozen programming languages for developing stored procedures and native applications using one of the many library interfaces supported. It has hundreds of built-in functions for taking care of anything from string operations, regular expressions, mathematics to cryptography and includes a framework that allows developers to create their own custom data types together with their own functions and operators. With all this freedom the PostgreSQL user community can extend the system by adding functionality as encapsulated extension modules, which can be easily managed with PostgreSQL's extension framework. PostgreSQL is not only a powerful database system capable of running enterprise grade systems, it is a development platform upon which to develop in-house, web, or commercial software products that require a capable RDBMS.

It was an obvious choice for TOTUS.

PostGIS

PostGIS is an official extension module for PostgreSQL that adds support for storing and manipulating geographical objects. PostGIS is standards compliant and follows the Simple Features for SQL specification from the Open Geospatial Consortium (OGC).

PostGIS builts upon well known open source GIS software libraries such as the Geometry Engine Open Source (GEOS) geometry library for some geometry operations, the PROJ.4 re-projection library for coordinate re-projections and the GDAL library for raster support. This makes PostGIS a powerful and open GIS tool for PostgreSQL. It supports the storage of geometry objects of types points, line-strings, polygons, multi-points, multi-line-strings, multi-polygons and geometry collections. PostGIS has a raft of spatial predicates and operators for various spatial measurement and analysis and it utilises R-tree-over-GiST (Generalised Search Tree) spatial indexes for high speed spatial querying with index selectivity support to to provide high performance query plans for mixed spatial/non-spatial queries. In addition PostGIS has raster and topology support.

PostGIS allows importing spatial data, writing complex spatial queries and algorithms using SFS and standards compliant SQL, of which the results can be visualized directly in a GIS. It's a cornerstone in TOTUS's foundation.

pgRouting

pgRouting builds upon the functionality provided by a PostGIS/PostgreSQL geo-spatial database by adding configurable, cost driven routing capability. pgRouting provides an interface between a topological road network and Boost's graph algorithms. The results of a topological road query is passed to Boost, converted to a directed graph, routed and the results returned to PostgreSQL.

The version used in TOTUS supports the *Dijkstra shortest path algorithm*; the *A-star (A*) algorithm*, a shortest path algorithm using a heuristic function; the *Shooting star (Shooting*) algorithm* which utilises a rule system for mimicking real road networks with turn restrictions, traffic lights and one way streets; the *driving distance algorithm* for calculating the area that can be covered in a given time and a solution for the well known *T ravelling Salesperson Problem*.

The advantages of the database routing approach for TOTUS are that the urban spatial data and attributes can be modified by many clients, like Quantum GIS and uDig either through JDBC, ODBC, or directly using PostgreSQL's procedural SQL or standards compliant SQL. Any data changes can be reflected instantaneously through the routing engine and there is no need for pre-calculation. In addition the edge cost parameter can be dynamically calculated through SQL and its value can come from multiple fields or tables, which allows TOTUS to apply different costs based on the mode of transportation.

pgRouting is used in TOTUS for transferring the traffic model output data to an OpenStreetMap (OSM) road network, routing based modelling and traffic model output aggregations.

FeatureServer

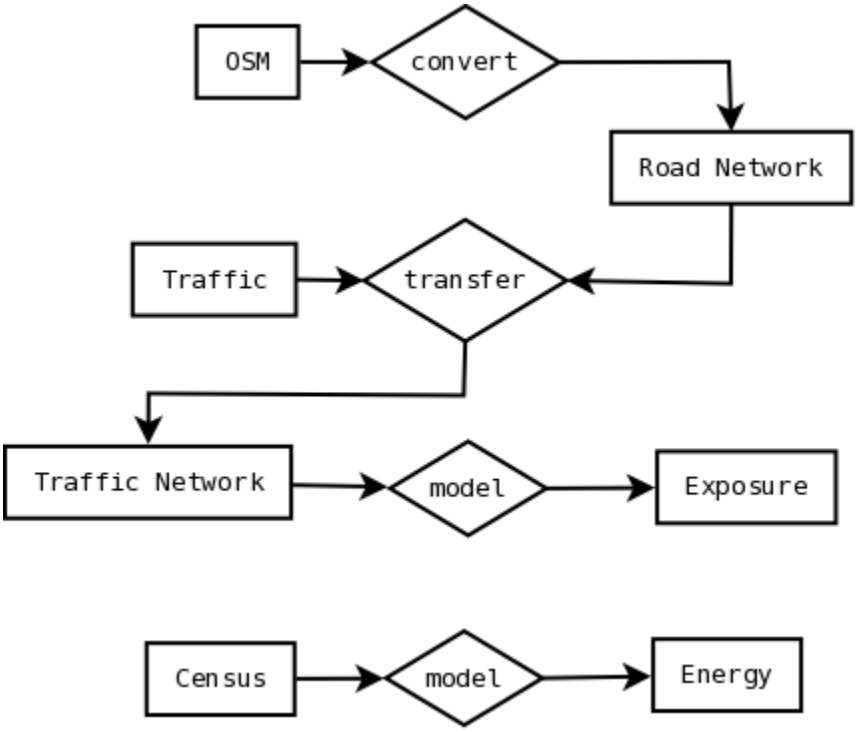
FeatureServer is an implementation of a RESTful (Representational State Transfer) Geographic Feature Service, written in Python. It implement Create, Read, Update and Delete (CRUD) functionality by mapping standard HTTP methods to database data operations. With FeatureServer you can fetch a representation of a feature or a collection of features, add new data to the service, or delete data from the service. It supports a variety of input data sources and output services which makes it a powerful and flexible middle ware that can be used as a data query layer, aggregator or format translator.

FeatureServer supports interacting with a variety of data sources such as OSM, OGR (part of GDAL), PostGIS, SpatiaLite, SQLite and WFS and data write (and some read-write) services for CSV, DXF (AutoCAD), GeoJSON, the GPS eXhange Format (GPX), HTML, Keyhole Markup Langauge (KML), OSM, the TomTom Point of Interest (POI) OV2 format, ESRI Shapefiles and SQLite with SpatiaLite

FeatureServer was chosen as the data access layer for TOTUS because of it's flexible and extensible architecture. FeatureServer was easily extended to support TOTUS's custom data sources.

Data sources

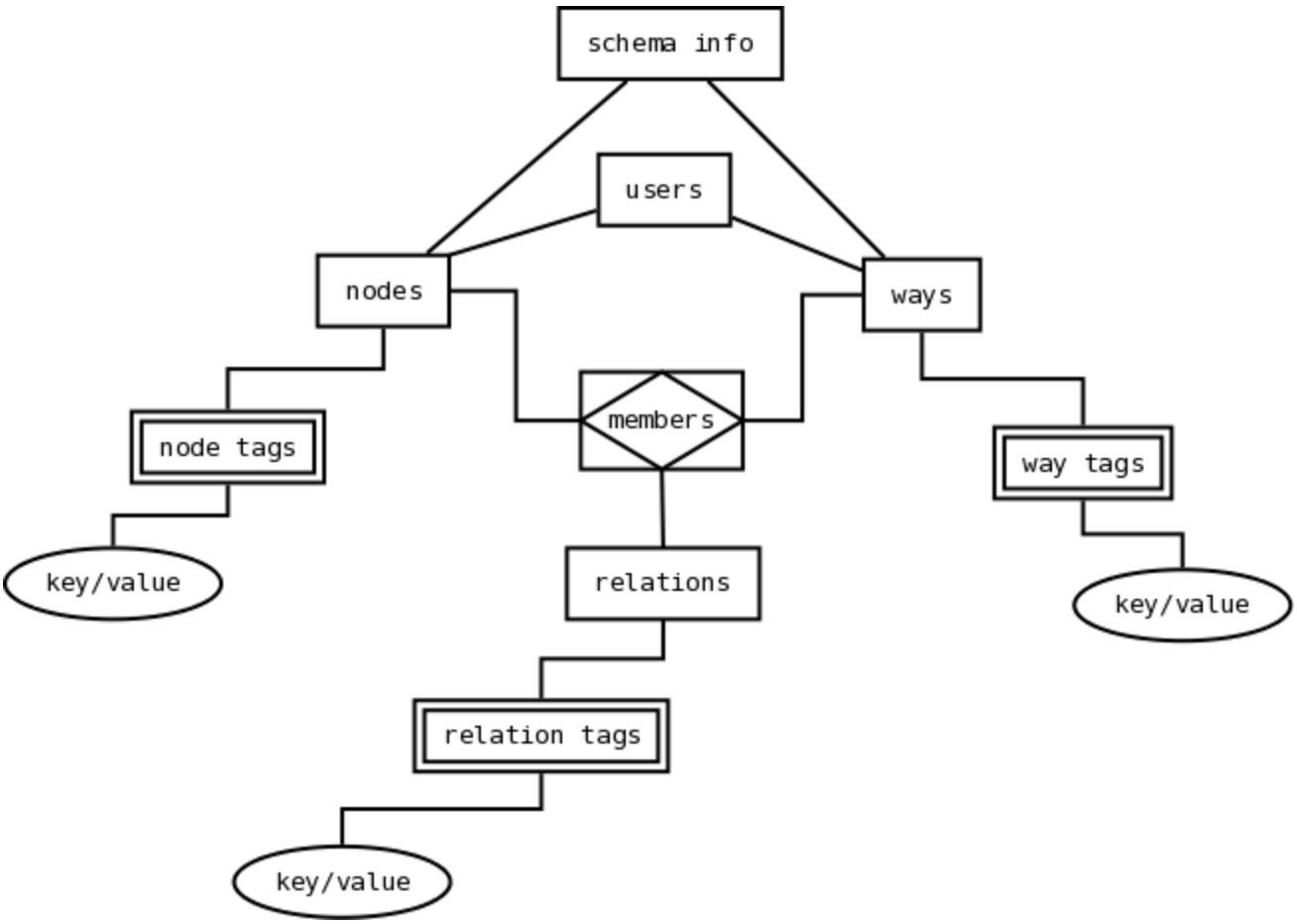
To be able to identify relationships within a changing urban form TOTUS requires a route-able spatial road network, output from a traffic model and demographic information. Exposure and energy demand information is derived using these data sources.



OpenStreetMap

OSM is used as the primary data source for TOTUS. It was chosen because of its data coverage, it is open data license (released under the Open Data Commons Open Database License) and its generic database schema which is flexible and can store virtually any spatial datasets. In addition for New Zealand it contains the Land Information New Zealand (LINZ) datasets.

TOTUS uses the simple schema representation of the OSM data model, as required by the OSM planet file loader.



The OSM simple data model is an Entity-Attribute-Value (EAV) model with post-ingestion validation. It constructs spatial features from nodes and ways and has the ability to store relationships between map features.

A node is the atomic element in the OSM data schema and used to construct all linear features. It holds the only geometry needed by OSM to represent linear map features and may or may not have any attributes. Attribute are stored as an extensible list in what is a EAV representation of spatial features. Nodes are allowed to be stranded (isolated from road network) when marked as an amenity (a Point Of Interest), but are usually the constituent parts of a spatial feature.

The *node* table is one of the three data primitive tables in the OSM physical model. It holds all the information needed to identify a vertex and revision it's changes. The OSM simple schema used by TOTUS only stores the most recent version of all data primitives. Node attributes are stored in the *node_tags* table as an extensible set of attribute type/value pairs that can hold any attribution associated with the vertex, eg. a roundabout, amenity, etc. Most nodes only exist to describe ways, however they may still have attributes that provided more information about the linear feature they are part of.

A way identifies any linear map features, eg. a road, ferry route, railway line, cycle way, tramping track, area, etc. Polygon features are represented as closed ways, but multi-polygons features can only be represented using relations, eg. relationship amongst all it's constituent area features. The *ways* table hold all the version and user information, as well as (for convenience) the PostGIS geometry constructed from the way's children node geometry. The *way_nodes* store the link between a single way and its constituent vertices. As a linear feature a way must have at least 2 bounding nodes. Each node has a sequence position within the way which is used to maintain digitisation direction. The extensible list of way attribution are stored as generic key value pairs in *way_tags*. Although any properties can be stored here, OSM encourages a standard way of denoting tags to allow for interoperability of tools using OSM data.

Relations are used to construct complex features from nodes, ways or even other relations and may be used to represent entities, eg. turn restrictions. Relations may group related features together, eg. public transport route. A relation consist of a feature or relation members, each with it's own role in the relation. As with other primitives relations may have an arbitrary number of attribute tags. The relation's version and user information is stored in *relations*, the data primitives that are members of the relation are stored in *relation_members* and any attribution associated with the relation, eg. public transport route number is present in *relation_tags*.

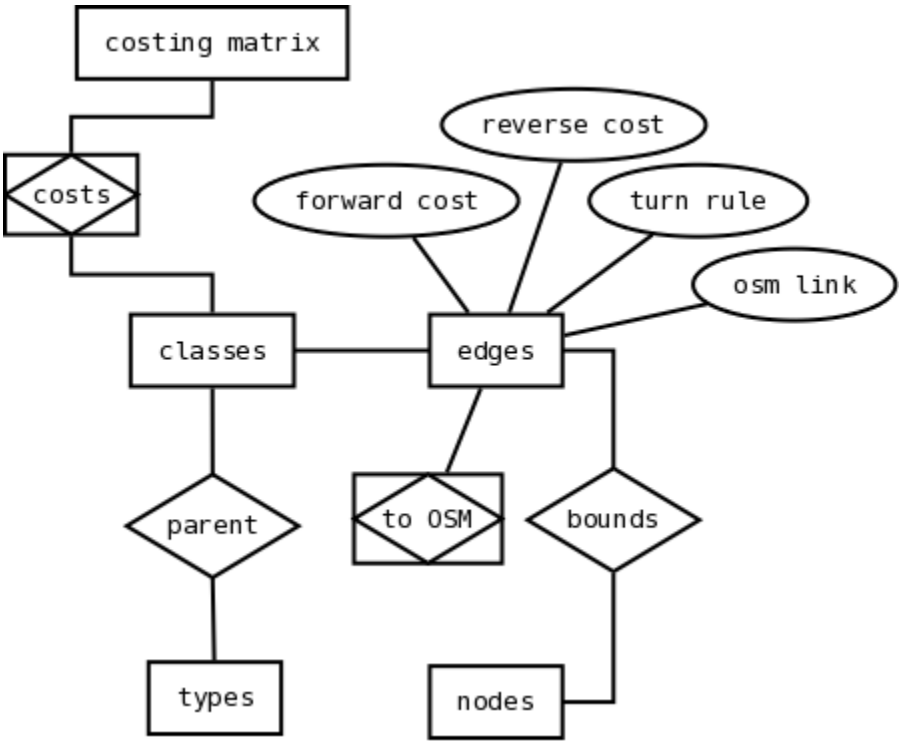
The OSM simple schema contains meta data that assist in re-visioning the content and structure of the simple schema. This information is dependent on by the schema objects and OSM planet file loader when loading data to the target schema. This allows the loader to support incremental data loads and use the revision history to only update the records affected in a change set. The meta data include *users* which hold the OSM users that have made changes to loaded dataset. OSM revisions all changes made to data primitives and their attribution, however OSM simple does not hold any historical data, only the latest. *schema_info* holds the version number of the OSM simple schema, which allows the OSM loader to know which schema to target when loading the data.

The current version of TOTUS is deployed on the Auckland region and utilises the OSM data for Auckland only.

Network

The OSM simple schema data is not suitable for network routing. By nature it is topological, but not necessary planar and cannot be effectively used by *pgRouting*. TOTUS uses the *pgRouting* OSM importer to create the correct planar network topology from an OSM planet file. This utility depends on a configuration file to load only the network classes TOTUS is interested in for routing purposes. The network schema is created from the same OSM planet XML file as imported into the OSM schema.

TOTUS uses a slighly modified version of *pgRouting*'s topological network schema for OSM data, as created by their OSM importer.



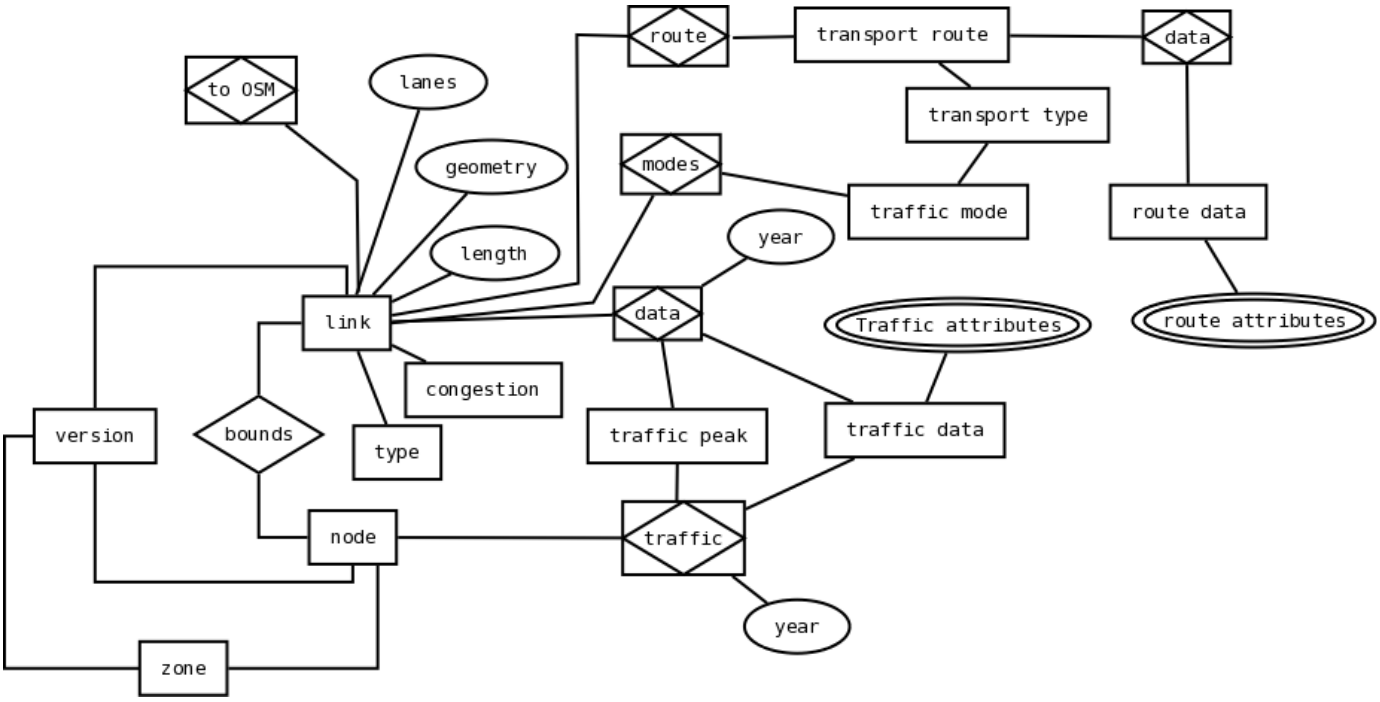
The *types* and *classes* classify the network edges into network types, eg. highway, each with their own classes, eg. motorway. This information is used by the routing functions to consider only sub-networks, eg. only highway classes when calculating routes and to apply rudimentary costing to the traversal of an edge. The *types* are the top level map feature types in the OSM data set and TOTUS only import cycleway, highway, junction and track type. The *classes* are the individual classes for the network types taken from the attribute values in the OSM map feature keys and used to scale the cost of traversing the network edges belonging to a specific network class.

In addition, TOTUS applies rudimentary costing per road class based on a route mode option. The calling route function needs to specify the route costing options to apply per road class to prevent certain road types, not allowed by the mode of transport, to be considered as a candidate edge, eg. cycling not allowed on motorway therefore the motorway road class is allocated a very high course to deter cycling on the motorway. The *costing_options* table holds the information for the costing options which include distance only, pedestrian, cycle and vehicle routing. The *class_costs* holds the cost matrix for the different classes for each route option. The cost itself is used as a scaling factor to the base cost of traversing the edge.

The topological information is stored in *nodes* and *edges*. The *nodes* table contain the vertices of the directed graph which bounds a graph edge. The *edges* table holds all the information needed for *pgRouting* which include the network class the edge belongs to, it's great-circle distance length, the name of the road the edge is part of, the start and end locations of the edge which is used by A-star's heuristic filter, a cost for traversing the edge in reverse, a turn restriction rule implemented as a list of subsequent edges to follow, the cost of the final edge in the turn restriction and the bounding source and target nodes. The length of the edge is used as the default cost for traversing the edge from source to target node, whilst the reverse cost is used for going from target to source node. One way's have a direction of flow from source to target node and are assigned a huge reverse cost to prevent navigating through it's target node against traffic flow.

Traffic model

The traffic model schema was designed to hold all traffic data as an extensible set per link and to represent traffic routes.



TOTUS is deployed on Auckland and uses the Auckland Region Council (ARC)'s EMME model output as traffic model. The ARC's EMME provides a modelled view of traffic flow through topological network links, which is an abstracted view of the real network and may not correspond to the OSM road network features and needs to be transferred manually.

The traffic data model of TOTUS needed to support different versions of the same model run for the same year, same model runs for different years with same or different link topology and same or different traffic attributes. In addition it needed to support predetermined grouping of modelled links as routes, eg. public transport routes, freight routes, etc. The data schema was designed with these requirements in mind.

At the core of the traffic model schema we have the topological modelled links, their model meta data, transport modes and the model zones.

The lookup table *link_types* contains the road types of the modelled links, which may or may not correspond to the OSM map feature types. Any relationships between these road types and OSM ones are used in transferring traffic model links to OSM network edges. The *congestion_functio n* is model specific and hold information about the different congestion functions applied to the links during modelling. The different transport modes supported by a modelled link for which the traffic flow is relevant is stored in *transport_mode*. The transport modes may include bus, rail, ferry, etc. Some traffic models classify the modelled links into zones and apply inter-zone modelling as well, these are filtered from TOTUS, but the information about the modelled areas are stored in *zones*.

The *link* table contains topological, version and physical attribution about the modelled geometry. The attribution include the length of the roads modelled by this link, the number of lanes simulated and the congestion function applied. The *node* data is created from the modelled links, unless provided. Each link may have traffic data modelled for multiple transport modes; the *link_transport_mode* relates a link to all it's transport modes.

The traffic model provides an extensible set of traffic attributes for each link for different traffic peaks, eg. morning, inter and evening peak. The lookup table *traffic_peak* contains the definition for each traffic peak and allows defining custom profiles. *traffic_attribute* defines a traffic attribute, eg. link traversal time, total vehicles per 2hr, and provides information on how to interpret such an attribute by defining it's data type. Traffic attributes may different from one model and/or year run to the next and may not always be present and each have a version.

The *traffic_data* tables holds the actual instances of the traffic attributes, eg. the attribute values for the defined traffic attributes types. This table contains unique attribute value pairs and is fully normalised.*link_traffic_data* links an traffic model geometry link to all instances of its traffic attributes for each traffic peak and model year, whereas *node_traffic_data* store any traffic data present for the link nodes.

Another purpose of the traffic data model is to store model information for transport routes. Tables have been defined to represent a modelled view of transport routes, eg. public or freight routes, which consist of a sequence of traffic model links and holds the route link's attributes, eg. number of public transport routes allocated. The route itself may also have attributes, eg. like 1 ton trucks only. *route_attribute* define a pre-linked route attribute definition, whereas *route_data* hold the instance data for a route attribute.

Each transport route are associated with a type of transport and consist of a type, eg. public, a transport mode, eg. rail, a vehicle, eg. train and a description. These are stored in *transport_type*. *transport_route* describes the route itself and contains route identifier, description and transport type information; *transport_route_data* links a route to all of its transport attributes and *transport_route_link* which defines the route path usig the the traffic model links (in order) that makes up the pre-linked path.

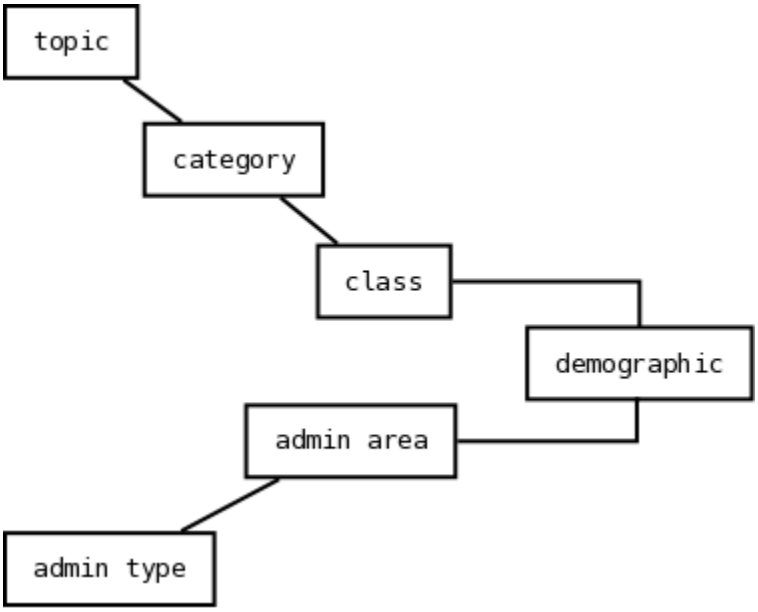
As mentioned earlier the traffic model output may not always correspond to the real work and TOTUS may require a mechanism of relating an traffic model link to one or more OSM network edges. Each OSM network edge assigned to an traffic model link is allocated a fraction of the traffic flow of the traffic model link. This information is used mostly for aggregated traffic model traffic information on a sub-network, a dynamic route, eg. pupil trip to school, etc. *link_network* relates an traffic model link with one or more OSM network edges and is vital in aggregating traffic model output derived variables, such as the Traffic Impact Factor (TIF).

Lastly the traffic data model holds meta data about the versions of the traffic model and transport model, as well as the year of the data used in model run. This information is stored in *version*.

Demographic

TOTUS relies on demographic information at the mesh block level for deriving energy related variables by producing simple energy models. Auckland TOTUS uses Statistics New Zealand's census dataset. Census data is provided as a collection of data sets for each instance of an administration area (geographies). For New Zealand these are mesh block, area unit, ward, territorial authority and regional council.

The administrative hierarchy is represented as a tree with each level of a specific administrative type. Demographic data is assigned to the mesh blocks only since each higher administrative level can determine their census data using aggregation. In general Census data is provided as a collection of data per topic. Each topic has a set of categories, each with it's own set of classifications. The demographic information is assigned to the class. These aspects are all incorporated in the design of the demographic data model.

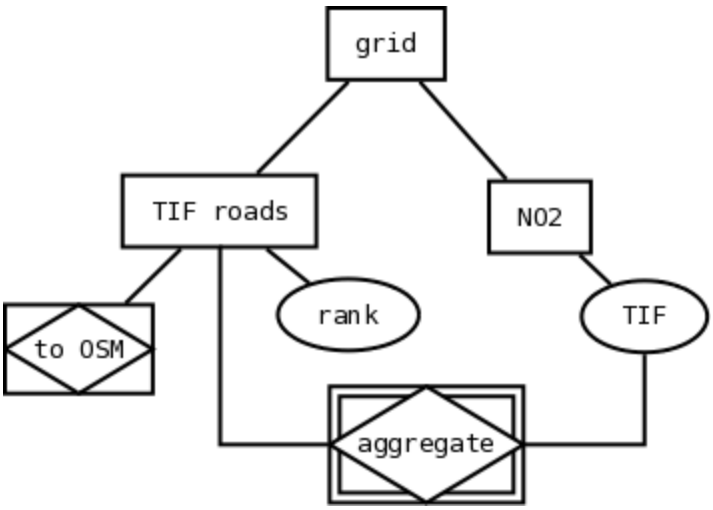


The demographic data schema in TOTUS defines the administrative types or levels of the area hierarchy in the table *admin_type* and stores the actual administrative areas in *admin_area*. The administrative areas are defined within an administrative or geographic hierarchy, but do allow for non-hierarchical data as well when its parent administrative place is missing. The census topics include about people, households, etc. and stored as a lookup list in *topic*. Similarly *category* stores the category for a given topic, eg. age in 5 year groups and *class* holds the different classes for a category, eg. 0 - 4 years. The *demographic* table holds the instance of a statistic of the human population, which is the tally per demographic class.

TOTUS modules

Exposure

The exposure module of TOTUS is used to model NO2 on a grid using the traffic model data transferred to the OSM road network. The Traffic Impact Factor (TIF) is calculated for a 100 meter grid spanning the geographic extent of the TOTUS instance and is used to estimate NO2 exposure values for a given 100 meter cell.



The *grid* table holds the 100 m grid for TOTUS's geographic extent and is meant to be the default implementation of the NO2 model. It is possible to create different resolution grids and perform a custom NO2 model run. The grid is only created for areas that overlap the traffic model zones. The traffic impact factor (TIF) is an indicator of the importance of a network edge's traffic flow contribution to exposure intensity at pre-determined distance from the edge. TIF is calculated by choosing a grid cell (centre of cell), choosing a pre-determined number of closest cells and applying a distance based dispersion function to their traffic volume. For efficiency the traffic impact factor network edges are stored in *grid_tif_edge*. This is by far the largest table in TOTUS and account for nearly half of the size.

NO2 is calculated for each using the linear relationship NO2 has with the TIF value aggregated from all network edges that contributed traffic volume to cell. The model values are stored in the table *no2*.

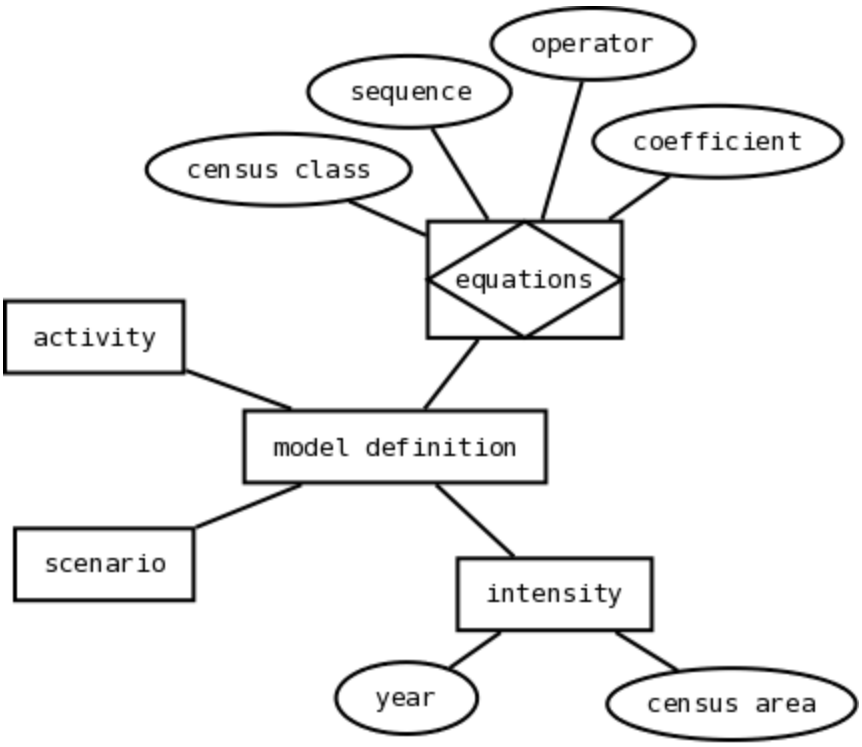
Energy

Energy intensity is derived from census demographic data at the granularity of census mesh blocks. The energy schema is designed to be generic and to allow some form of aggregation to be applied to a set of model equation parts, each which derives information from a single census demographic statistic.

Being generic the schema needs to store meta data needed to configure a energy model run. The table *activity* defines the energy activity to model energy intensity for, eg. household heating, total energy consumption, whilst *scenario* contains the various scenarios that can be applied to apply to modelling energy intensity, eg. current accounts, continuity.

The model definition provides the aggregation rules to use when producing an energy intensity value for a specific energy producing activity and scenario. It is a simple linear equation:

```
Energy Intensity = some demographic data * coefficient + OPERATOR (some
other demographic data, coefficient)
```



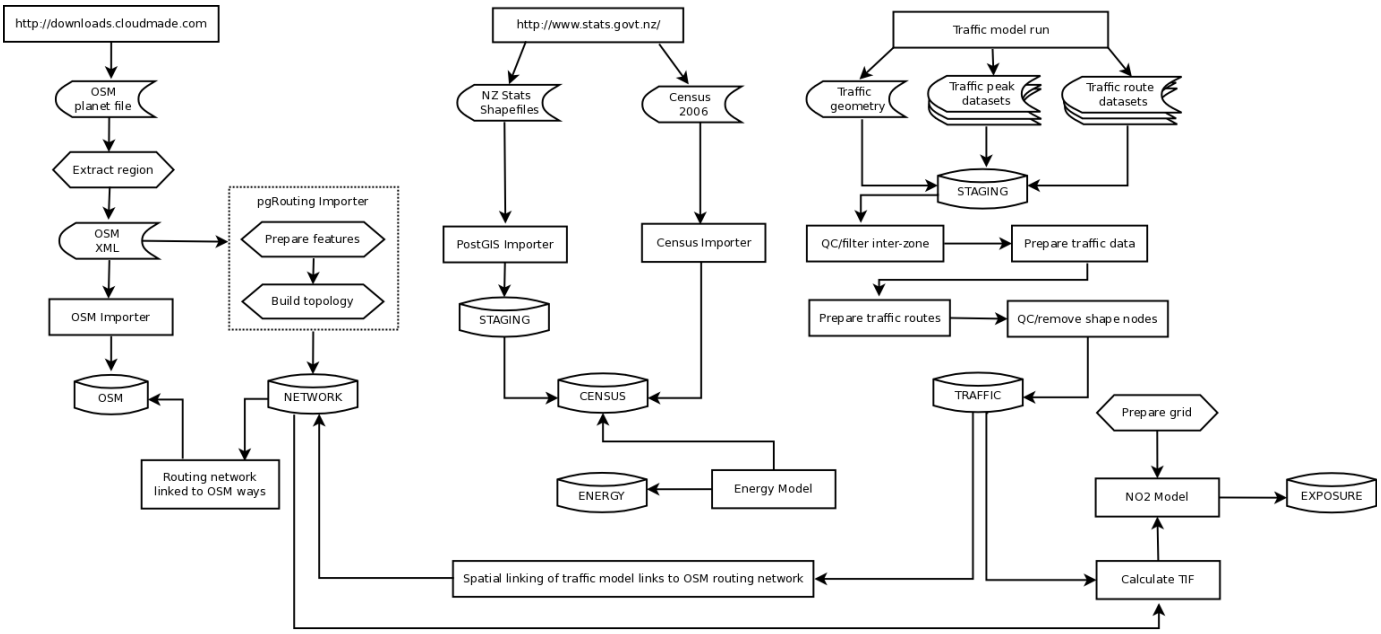
model_definition hold the information about a specific model and contains details about the activity simulated for which scenario and how many parts exists of the equation. *model_definition_part* defines the individual parts of the equation that are aggregated to produce a final intensity value. Each model definition part is defined for a demographic class, eg. household heating and consist of a coefficient the apply, the operation to perform on the demographic *count* field, eg. POW (count, 0.09) and the sequence of the equation part in the whole model equation.

The model output, when run, is stored in the *intensity* table. This is the output of a model when run on mesh block census demographic data. A stored procedure will take care of generating and executing the SQL statements that will derive the Energy intensity value for a specific energy activity and scenario.

Implementation details

Preparing the TOTUS data store

The TOTUS data store supports incremental updates, but the current tool chain that prepares the system do not fully support incremental loads. Ideally the system should be able to augment the OSM data set, which triggers an update on the road network (not using the pgRouting tool chain) and re-linking those road edges to the traffic model topology. Similarly when an existing traffic model is updated. When a new traffic model dataset is loaded it automatically links the traffic model network to the OSM routing network. The current system is prepared as follows:



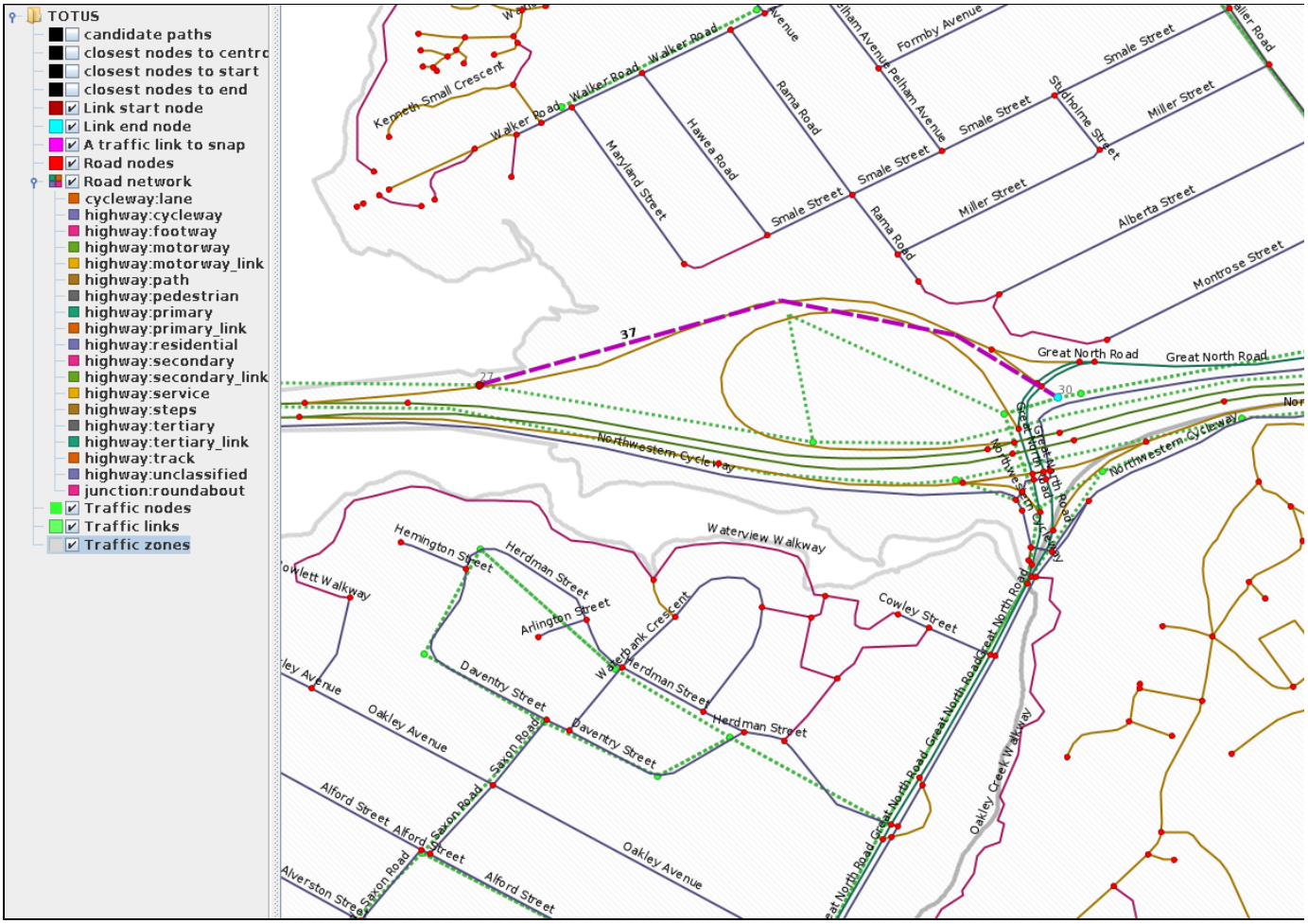
When preparing a new TOTUS system an empty TOTUS data store is prepared prior to loading, this includes all schema objects and procedures. The definition of the TOTUS data store is stored in a Subversion repository and deployed using *Apache Ant* which is a Java library and command-line tool for defining and managing complex build processes. It has an expressive XML syntax with which the conditional logic needed for managing the inter-dependencies that exist amongst the TOTUS data sources and modules can be expressed. When the build process is called upon for the first time a clean data load is performed.

The primary data source of TOTUS is the OSM dataset. The TOTUS OSM loader utilises *osmosis* (version *0.36*) to extract a certain geographic extent from a planet file downloaded from *CloudMade*, then import it to the OSM simple schema prepared by the *Ant* build process. The same geographic OSM extract is passed to the OSM pgRouting importer *osm2pgrouting* (pre-relation version), which has been slightly modified to work with the specific TOTUS schemas and build process. The importer determines when line strings crosses through shape nodes and turn them into real road nodes, splitting the road edges accordingly and updating all of it's attribution. Only the OSM road map features and attributes of interest to routing is parsed and persisted. All routing edges are linked with their OSM counterparts to allow modifications of the OSM to be tracked in pgRouting and for extracting any other attributes or spatial features of interest to TOTUS. With time TOTUS will have it's own pgRouting data loader to replace *osm2pgrouting* and have the ability to modify the routing network using an external interface; eg. how would the NO2 exposure of individuals within an urban form change if we were to connect two no-exit roads and route traffic through it.

Next the traffic model dataset is loaded to TOTUS. The dataset expected consist of a shape file defining the traffic model zones, a shape file defining the modelled link topology and link attribution, a set of spreadsheets that contains all the traffic attributes for each link for each traffic peak defined, a shape file defining the topology for the modelled routes and a set of spreadsheets containing their traffic attributes for each traffic peak. The raw traffic data set is loaded to a set of staging tables, from which the data is converted into the TOTUS traffic model schema. This entails preparing all the lookup tables for the new data set loaded (that is for each model year); adding new traffic zones, nodes and links for each model year; constructing the link transport types from the model type string and snapping the traffic modelled route geometry to the traffic links. Next the route and traffic attributes are de-constructed from a transposed view into a extensible row set and linked to it's parent traffic link. When done we have a fully attributed traffic model populated in TOTUS. Next the traffic modelled links need to be spatially linked to the OSM road network.

Linking traffic model output to OSM

This is not a trivial exercise if the level of abstraction applied to the road network during modelling has obfuscated the real road network, which may mean that the topological traffic model network has little resemblance to the real OSM road network. For Auckland and the EMME model this is certainly the case, although it mostly effects complex motorway interchanges close to dense suburbs and local roads. The topology surrounding *traffic model link 37* illustrate the problem:



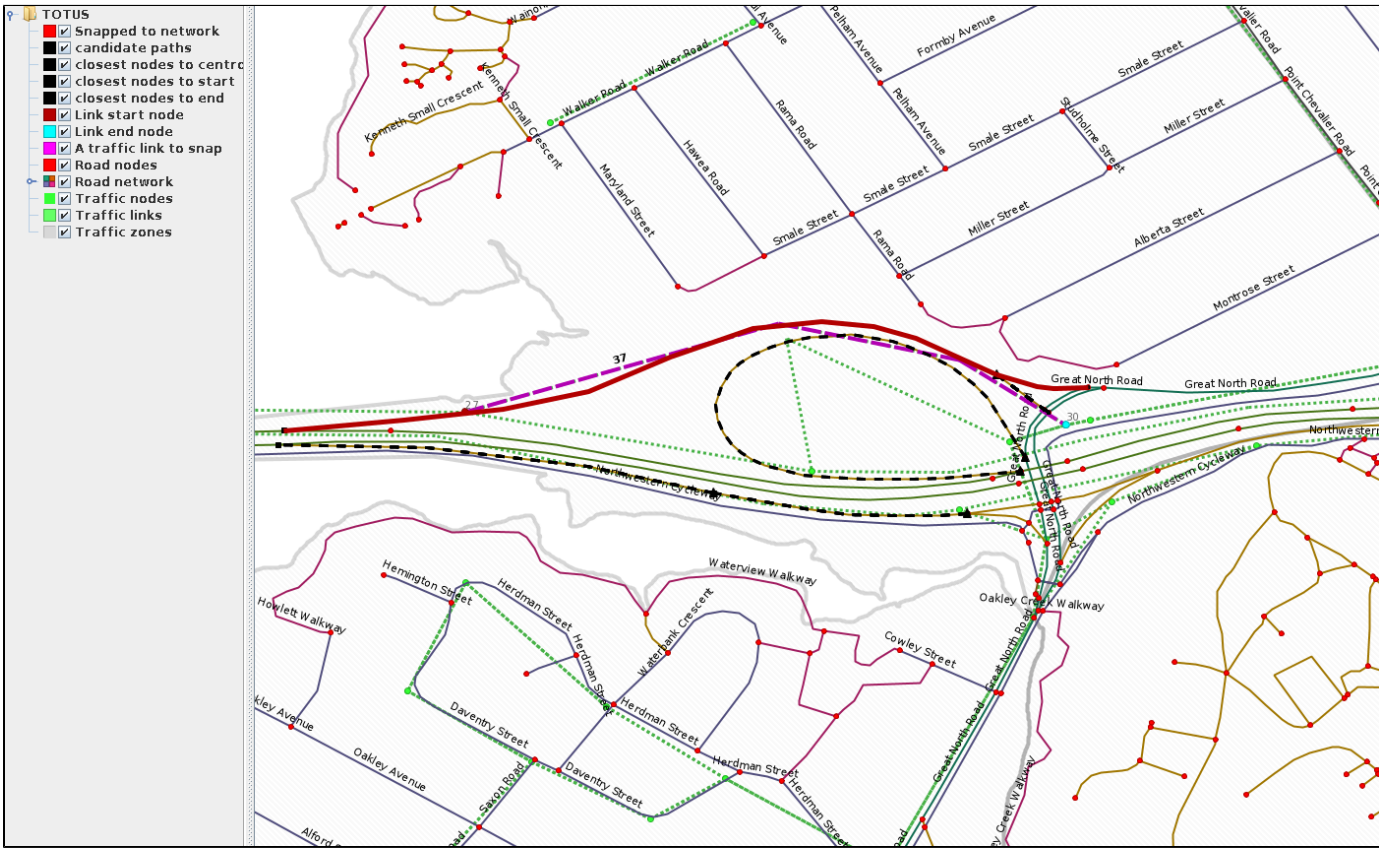
The traffic model output is transferred to the real road network by searching for a road segment or sequence of road segments that best matches the spatial characteristics of the modelled link. That limits the search to being a purely spatial one, but the accuracy of snapping is greatly improved if a rudimentary mapping between the modelled link types and OSM road map features exists. The spatial search includes finding a sequence of road segments that follow the modelled link closely. In particular its direction of flow should correspond to the digitization direction of traffic link; its shape should be similar; its start and end must be in the vicinity of those of the modelled link; the minimum bounding rectangle of the two features should correspond sufficiently and its great circle length should be similar. The suitability of the candidate segments are evaluated according to these criteria.

We start by exploring the neighbourhood of the traffic link's start and end nodes and determine the three closest OSM road nodes to each. For Auckland TOTUS the traffic model distinguishes between motorway, motorway ramps and non-motorway link types. This information is used to exclude OSM nodes and edges that do not correspond in type, eg for the example above (*link 37*) we only include nodes attached to motorway link segments. To ensure that we have sufficient coverage for curved modelled links we also include five OSM nodes (of the correct type) closest to the centroid of the link feature. The number of nodes to choose can be configured.

Next a set of vehicle routes are generated between each of the OSM road nodes (cross matrix), but only considering road segments of the correct type. The routes are costed as a normal vehicle traversing the network, considering way-ness and costing according to network classes, excluding turn restrictions (not currently supported). Each set of network segments are considered a candidate for receiving the link's traffic data and each are evaluated to determine which one matches best. Prior to doing so we filter the paths that did not reach their destination OSM node, these are incomplete and should not be considered. Next we need to extend the traffic links to allow projecting of shape points of the individual segment edges onto the EMME link for calculating match statistics. We compare the projected line string to the real segment line-string for determining it's suitability index. We do so by projecting project each segment's line-string sub-segment parts of the candidate network path to the extended traffic link; by doing so we filter sub-segments that do not follow the traffic link in the same direction. This leaves gaps which automatically disqualifies the candidate path. For each sub-segment part we determine the angular deviation it has from projected line string; its offset from the projected line string and its difference in length. We aggregate the square of each delta for each of these variables to calculate the root mean square error of these. In addition to the RMSEs we retain the total candidate path displacement, when its start and end node are projected onto the traffic link; calculate the percentage overlap in the MBR of the candidate path and the traffic link and aggregate the total route cost of the candidate path. These errors are normalised to determine indicators, which are combined to calculate the suitability index (SI) for each candidate path as:

```
SI = ( MBR * 2.0 + COST / 2.0 + PROJ_RMSE + ANGULAR_RMSE / 5.0 +
OFFSET_RMSE + LENGHT / 5.0 + DISPLACEMENT ) / 7.0
```

The candidate path with the best SI is retained and its constituent segment edges are linked to the traffic modelled link. The fraction of the traffic flow that needs to be assigned to the individual edges are determined by edge length. It may happen that some traffic links failed to snap to the road network due to a mismatch in topologies or that poor candidate paths were chosen due to complex topologies or corner cases. These are reported and then snapped using the distance and the suitable map class as weights discarding features that do not match the way-ness of the traffic link. For *link 37* this results in the Great North road off ramp.



Due to the nature of this snapping method, relying mostly on spatial similarity, we may end up with gaps between candidate route paths of two connected traffic links. We find the gap nodes by searching for nodes that have only one connected traffic edge per road class, find neighbouring edges of the same road class as the edge connected to gap node inside the gap. Next we iterate through neighbours of the gap node's neighbouring edge following topology in the correct direction. Iteration halts when the next traffic road node is found or when candidates are too far away. Some paths may loop back unto themselves and truncated at the loop node. When two candidate edges are available to fill an EMME gap the closest one is chosen. When done all these gap edges are linked to the traffic model and all portions for the edges linked to the same traffic link are updated. Now we can start using the OSM traffic model network for exposure modelling.

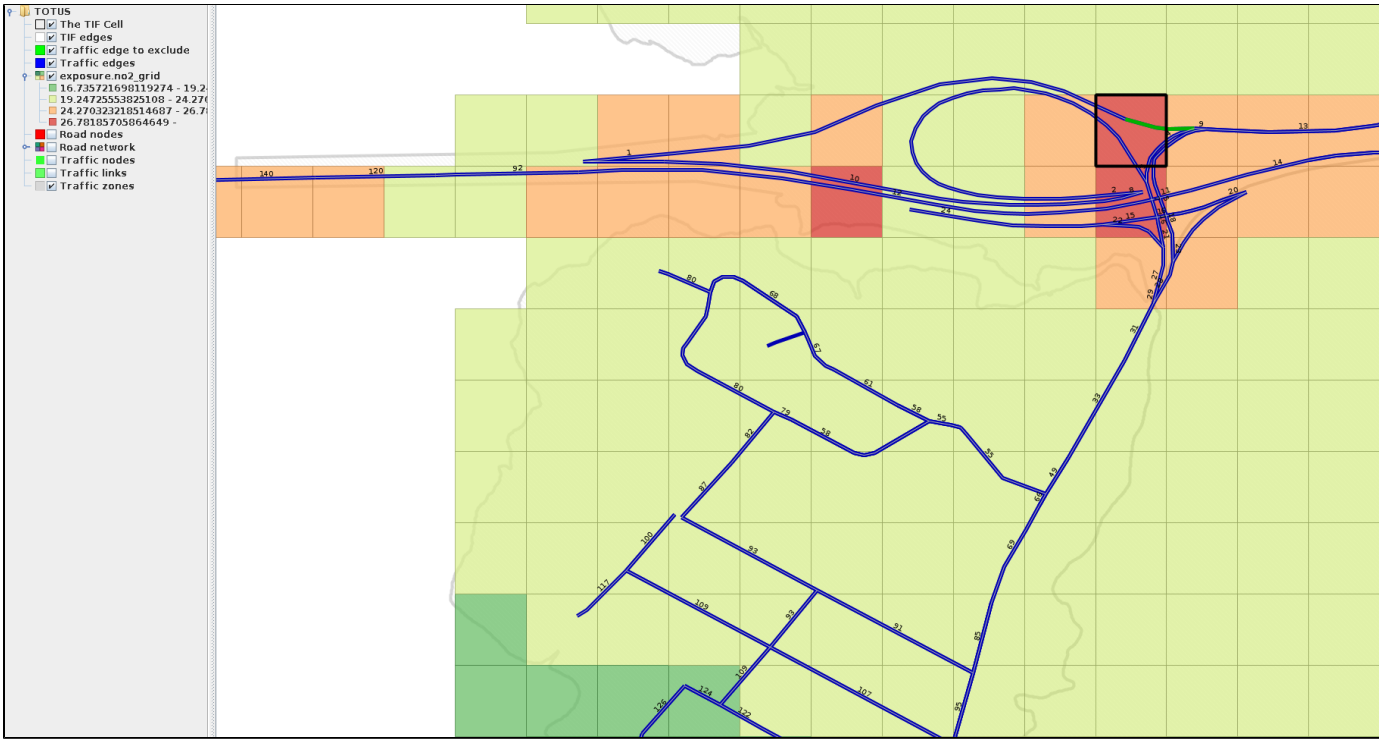
Preparing the exposure model

The exposure module in TOTUS consist of a set of database functions for deriving NO2 values from traffic model output for the edges on a given grid. The spatial extent and size of the grid is configurable, as long as it falls within the spatial extent of the traffic zones (as defined by the traffic model). The traffic impact factor (TIF) is determined for a given number of traffic network edges close to the centroid of each grid cell. All edges within a certain distance of the centroid of the cell are ignored, mostly the ones that start and flow out of the grid cell, because TOTUS models the influence of the neighbourhood on a given cell. The TIF value for each edge for a given grid cell is derived from the traffic flow assigned to it by the traffic model snapped to the road network by applying a distance based dispersion factor, each TIF edge is assigned a rank based on their distance from the centre of the cell.

```
TIF = SUM (0 .. N) (TRAFFIC VOLUME * POW ( DISTANCE TO CELL, DISPERSION
FACTOR))
```

The TIF values for all contributing TIF edges for a given cell are aggregated and used to predict the NO2 exposure for the cell.

```
NO2 (ug/m3) = 0.00171 TIF + 11.9
```



Preparing an energy model

The demographic data set for Auckland TOTUS is loaded from the NZ statistics spreadsheets using the TOTUS census importer, which converts the de-normalized and tabular view of the demographic data to a row-set dataset which are fully normalised at the mesh-block level. The census database schema in TOTUS holds demographic data for a set of topics each with their own categories. Each category consist of one or more classes, each of which may be assigned a count per mesh-block area. The energy intensity database schema in TOTUS has been designed to facilitate deriving energy intensity from one or more demographic variables. It contains meta data about the definition of a modelling equation and the energy intensity output of the model run. Each part of an energy equation is derived from one demographic class instance, eg. derive energy demand from household demographic classes by performing some set of operations (as defined in the model). The energy schema allows a user to define different model definitions, as in apply different coefficients or equation parts to the same census classes to model some scenario, eg. continual growth, current accounts, etc. These scenario need to be defined prior to configuring a model and is currently done in the energy schema loader. Each energy intensity may be associated with a specific energy activity, eg. household heating. As with scenarios these are defined in the energy schema loader.

The energy modelling is facilitated using TOTUS database functions; one to configure the model and another to use the model definition to populate the energy intensity table and return the results.

Eg. to define a simple energy model that defines that energy intensity is ten times the number of people in an mesh-block.

```
SELECT *
    FROM energy.configure_model_run (
      'TOTAL_RESIDENTS_MODEL',
      'Energy intensity is 10 times the number of people in the selected
area',
      'ALL',
      'CURRENT',
      ARRAY [ ('TOTAL', '', 10.0)::energy.definition_part ]
    );
```

And to create the model's energy intensity data set for 2006:

```
SELECT *
    FROM energy.model_intensity (
      'TOTAL_RESIDENTS_MODEL'::VARCHAR,
      2006::SMALLINT
    )
   LIMIT 1;
```

which can then be viewed in a GIS.

Conclusion

This document described the TOTUS data system and how it can be used to model interactions within a changing urban form.