Importazione delle mappe OSM nella KB di Km4City

Versione

2.3

Ultimo aggiornamento

20/06/2018

Storia del documento

10/04/2017 Completamento della stesura della bozza di proposta iniziale

14/04/2017 Integrazione di aspetti implementativi

02/05/2017 Proposta l'introduzione della classe Hamlet. Superato nominatim. Definiti con maggiore precisione gli aspetti implementativi.

08/05/2017 Migliorata la classe Hamlet istanziando anche le frazioni e quartieri rappresentati come nodi, e introducendo le coordinate geospaziali. Modificata la mappatura degli elementi stradali. Riorganizzati gli aspetti implementativi.

24/05/2017 Definito un algoritmo di riconciliazione tra province, comuni e toponimi del grafo stradale della Regione Toscana, ed i corrispondenti elementi del grafo stradale di Km4City, definendo anche le query SPARQL che devono essere eseguite nei diversi passaggi dell'algoritmo, ma senza implementare un software che automatizzi l'esecuzione dell'algoritmo (sviluppo futuro). Definita ed implementata una strategia per la riconciliazione dei servizi sul grafo stradale di Open Street Map

29/05/2017 Risolta la criticità sulla riconciliazione dei servizi sul grafo OSM. Introdotto un servizio REST che associa ad un punto con latitudine e longitudine arbitrarie, l'ID OSM di una regione, provincia, comune, way o nodo, a seconda della richiesta dell'utente. Il servizio è implementato sotto forma di una servlet denominata Latlon2osm.

01/06/2017 Aggiunta la mappatura tra le Regioni delle mappe OSM e le corrispondenti classi km4c:Region di nuova introduzione nella Smart City Ontology. Predisposizione di materiale pensato appositamente per la generazione di triple relative ad uno specifico aspetto (una particolare proprietà di una particolare classe) senza dover procedere alla generazione da zero dell'intero grafo.

08/06/2017 Introdotto il capitolo dedicato alla rappresentazione della mappatura delle paline del TPL della Regione Toscana sui nodi del grafo stradale di OSM, con indicazione del contesto, delle finalità e dei dettagli implementativi relativi alla versione corrente, oltre agli immancabili sviluppi futuri.

21/06/2017 Definita una mappatura e realizzata un'implementazione per l'importazione delle corsie e delle restrizioni di accesso, di svolta e di massimo/minimo, riferite sia alle strade nel loro complesso, che alle singole corsie, con distinzione tra mezzi di trasporto e gestione delle restrizioni condizionate.

26/06/2017 Introdotto uno strumento eseguibile da riga di comando ed un file batch di Windows ad esso associato, per l'automatizzazione dell'algoritmo di riconciliazione del grafo stradale della Regione Toscana con quello generato a partire da Open Street Map.

Aggiornata la documentazione.

29/06/2017 Introdotto uno strumento eseguibile da riga di comando per l'importazione dei numeri civici dalle basi dati della Regione Toscana, rilevato che i numeri civici presenti sulle mappe Open Street Map non sono sufficienti per i nostri scopi. Modificati gli script SQL e SML rispettivamente di preparazione dei dati per la triplificazione del grafo stradale OSM e di configurazione di Sparqlify, di modo da gestire il caso in cui i numeri civici da considerare per la generazione delle triple siano quelli importati dalla Regione Toscana e non quelli nativi di Open Street Map.

05/07/2017 Introdotti accorgimenti per il miglioramento delle prestazioni, ed in particolare l'indicizzazione delle geometrie PostGIS e la riscrittura di alcune interrogazioni così da trarre il massimo vantaggio dalla disponibilità degli indici. Aggiornata la documentazione per quanto riguarda il procedimento di triplificazione del grafo stradale, e la rappresentazione della riconciliazione delle paline TPL sui nodi OSM.

12/07/2017 Esteso l'utilizzo degli indici geospaziali e più in generale l'adozione di misure per il miglioramento dell'efficienza, anche alla parte di preparazione dei dati per la generazione delle triple del grafo stradale di Km4City a partire dalle mappe OSM, intervenendo diffusamente sullo script SQL che predispone le tabelle relazionali di appoggio che vengono poi sfruttate da Sparqlify per la generazione delle triple.

07/08/2017 Introdotta la proprietà highwayType sui RoadElement, ottenuta riprendendo tal quale il valore del tag highway presente sull'elemento way di OSM a partire dal quale il RoadElement è generato.

24/08/2017 Riportate su questo documento alcune note che avevo preso su di un appunto separato, riguardanti la dislocazione degli strumenti, delle sorgenti dati, e degli output, oltre ai modi d'impiego degli strumenti e alle procedure per la triplificazione e la pubblicazione in test delle triple.

05/09/2017 Fixes. Alcune parti della documentazione, soprattutto per quanto riguarda la tecnica seguita per stabilire i legami di inclusione tra enti territoriali, e la generazione degli elementi stradali nel caso di toponimi formati da più di una way OSM, non erano state aggiornate.

29/09/2017 Migliorata la documentazione relativa al tool di riconciliazione dei servizi sul grafo stradale importato da OSM, inserendo una descrizione di maggiore dettaglio dell'algoritmo, ed inoltre un esempio di invocazione, e rimuovendo la parte relativa alle criticità che sono state nel frattempo superate.

04/12/2017 Documentate le modifiche apportate sulle restrizioni di svolta, che adesso fanno sempre riferimento ad elementi stradali come sorgenti e destinazioni, non più a strade, ed includono inoltre il nodo. Documentata l’importazione dei numeri civici che non hanno su OSM una strada associata. Documentata la modifica sull’elementType, per cui gli elementi che si trovano sulle rotatorie hanno adesso la proprietà istanziata due volte, una per indicare che si trovano in una rotatoria, e l’altra per indicare che si trovano su di un tronco di carreggiata.

18/12/2017 Aggiornata la parte relativa alla riconciliazione delle paline degli autobus con i nodi OSM, e alla relativa triplificazione. Aggiornata la parte relativa alla riconciliazione di un punto generico individuato attraverso una coppia di coordinate, con un elemento OSM di interesse, andando in particolare a descrivere la restrizione, che è di interesse soltanto nel caso in cui si debba eseguire la riconciliazione verso un nodo, secondo cui il nodo debba poter essere raggiunto a piedi, restrizione che viene utilizzata quando il servizio viene invocato per la riconciliazione di una palina dell’autobus dal momento che le paline dell’autobus sono tipicamente raggiunte a piedi.

20/12/2017	Aggiornata la mappatura dei numeri civici, descrivendo il recupero dei numeri civici dalle way di OSM (outline degli edifici), che fino a questo momento non veniva fatto.
29/01/2018	Migliorata la descrizione del mapping dei numeri civici.
08/06/2018	AdministrativeRoads
20/06/2018	AdministrativeRoad: aggiunte proprietà adRoadNameGeneric e adRoadNameSpecific

Abstract

Importazione delle mappe stradali di Open Street Map: corrispondenze tra elementi delle mappe OSM e classi della Smart City Ontology, implementazione dell'importazione. Riconciliazione tra il grafo stradale della Regione Toscana ed il grafo stradale importato da Open Street Map: definito un algoritmo, definite le query SPARQL che devono essere eseguite nei diversi passaggi, ed implementata l'esecuzione automatica dell'algoritmo. Definita ed implementata una strategia per la riconciliazione dei servizi sul grafo stradale di Open Street Map. Definita e implementata una strategia per la mappatura di un punto con coordinate geo-spaziali arbitrarie, su di un nodo del grafo stradale OSM (ed opzionalmente su altre entità OSM come regioni, province, comuni, elementi stradali). Descritta l'implementazione della rappresentazione del legame tra paline del TPL della Regione Toscana e nodi del grafo stradale OSM, realizzata nella forma di nuovi componenti di ETL da integrare opportunamente nel flusso esistente. Introdotte sul grafo stradale KM4C le corsie e le restrizioni di accesso, svolta e max/min relative sia alle strade nel loro complesso che alle singole corsie, con distinzione tra mezzi di trasporto e gestione delle restrizioni condizionate. Introdotto uno strumento eseguibile da riga di comando per l'importazione dei numeri civici dalle basi dati della Regione Toscana, rilevato che i numeri civici presenti sulle mappe di Open Street Map sono insufficienti per i nostri scopi. Introdotti alcuni accorgimenti finalizzati al miglioramento delle prestazioni, in particolare l'introduzione di nuove tabelle relazionali e di opportune indicizzazioni, che unitamente a modifiche mirate sulle interrogazioni di modo da trarre il massimo vantaggio dalle indicizzazioni, ha permesso di conseguire in brevissimo tempo risultati oltremodo incoraggianti soprattutto sulla riconciliazione di punti arbitrari su elementi OSM, e sulla generazione delle triple del grafo stradale Km4City a partire dalle mappe OSM. Nel giugno 2018 sono state introdotte in KB anche le AdministrativeRoad importate da OSM: frammenti di strade sovracomunali, dotati di un proprio nome, diverso da quello della strada sovracomunale.

Indice degli argomenti

Finalità

Ci si pone l'obiettivo di determinare quale corrispondenza possa essere instaurata tra la rappresentazione XML delle mappe stradali utilizzata in Open Street Map, e l'ontologia del grafo stradale del progetto Km4City, con il fine ultimo di importare i dati di interesse estratti dalle Open Street Map, all'interno della Knowledge Base di Km4City. Ci si pone inoltre l'obiettivo di individuare una strategia implementativa che consenta di realizzare tale mappatura.

Ci si pone inoltre l'obiettivo di riconciliare molteplici elementi che erano precedentemente collegati al grafo stradale importato dalla Regione Toscana, con i corrispondenti elementi appartenenti al grafo stradale popolato tramite l'importazione delle mappe di Open Street Map, e di predisporre opportuni strumenti ad uso delle applicazioni per un efficiente utilizzo dei dati presenti nella KB.

Ipotesi di mappatura delle OSM sulla Smart City Ontology

Si delinea in questa sezione un'ipotesi di mappatura dei dati contenuti sulle carte stradali di Open Street Map, verso l'ontologia del grafo stradale di Km4City.

Mappatura delle regioni

Le regioni sono rappresentate nei documenti OSM attraverso elementi XML `relation` contraddistinti attraverso:

- un tag `type` valorizzato a `boundary`;
- un tag `boundary` valorizzato a `administrative`;
- un tag `admin_level` valorizzato a `4`;

come descritto sul Wiki di Open Street Map.

Nella Smart City Ontology, la classe che rappresenta le regioni è la `km4c:Region`.

Nel generare una nuova istanza di `km4c:Region` importando i dati da OSM, si propone di valorizzare:

- `rdf:type` con `km4c:Region`;
- `dct:identifier`, con la stringa `OS`, seguita dalla valorizzazione dell'attributo `id` dell'elemento `relation` di OSM che rappresenta la regione che si deve importare (con aggiunta di zeri iniziali fino al raggiungimento delle undici cifre), seguita dalla stringa `RG`;
- `foaf:name`, con la valorizzazione del tag `name` della `relation` di OSM che rappresenta la regione che si deve importare;
- `dct:alternative`, con la valorizzazione del tag `short_name` della `relation` di OSM che rappresenta la regione che si deve importare;
- `km4c:hasProvince`, v. Mappatura del legame tra regioni e province.

Mappatura delle province

Le province sono rappresentate nei documenti OSM [1] attraverso elementi XML `relation` contraddistinti attraverso:

- un tag[2] `type` valorizzato a `boundary`;
- un tag `boundary` valorizzato a `administrative`;
- un tag `admin_level` valorizzato a `6`;

come descritto sul Wiki[3] di Open Street Map.

Nella Smart City Ontology, la classe che rappresenta le province è la `km4c:Province`[4].

Nel generare una nuova istanza di `km4c:Province` importando i dati da OSM, si propone di valorizzare:

[1] Documenti XML che rappresentano le mappe in Open Street Map

[2] Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag

[3] http://wiki.openstreetmap.org/wiki/Tag:boundary%3Dadministrative#10_admin_level_values_for_specific_countries

[4] Lapo Bicchielli, Analisi e sviluppo di un sistema di acquisizione, rappresentazione ed accesso di informazioni geografiche e di servizio, statiche ed in tempo reale, per Smart City, Tesi di laurea magistrale, Anno accademico 2012 - 2013

- `rdf:type` con `km4c:Province`;
- `dct:identifier`, con la stringa `OS`, seguita dalla valorizzazione dell'attributo `id` dell'elemento `relation` di OSM che rappresenta la provincia che si deve importare (con aggiunta di zeri iniziali fino al raggiungimento delle undici cifre), seguita dalla stringa `PR`;
- `foaf:name`, con la valorizzazione del tag `name` della `relation` di OSM che rappresenta la provincia che si deve importare;
- `dct:alternative`, con la valorizzazione del tag `short_name` della `relation` di OSM che rappresenta la provincia che si deve importare;
- `km4c:hasMunicipality`, v. Rappresentazione del legame tra le province e i comuni;
- `km4c:isInRegion`, v. Mappatura del legame tra regioni e province.

Mappatura del legame tra regioni e province

Per ciascuna provincia si va ad individuare la regione il cui confine include quello della provincia, utilizzando le funzioni rese disponibili dall'estensione PostGIS di Postgres, e si realizza così la corrispondenza tra ciascuna provincia e la regione in cui la stessa si trova.

Il legame è rappresentato valorizzando opportunamente la proprietà `km4c:hasProvince` sull'istanza della regione, e la proprietà `km4c:isInRegion` dell'istanza della provincia, con gli URI dell'istanza collegata.

Mappatura dei comuni

I comuni sono rappresentati nei documenti OSM[5] attraverso `relation` contraddistinte attraverso:

- un tag[6] `type` valorizzato a `boundary`;
- un tag `boundary` valorizzato a `administrative`;
- un tag `admin_level` valorizzato a `8`;

come descritto sul Wiki[7] di Open Street Map.

Nella Smart City Ontology, la classe che rappresenta i comuni è la `km4c:Municipality`[8].

Nel generare una nuova istanza di `km4c:Municipality` importando i dati da OSM, si propone di valorizzare:

- `rdf:type` con `km4c:Municipality`;
- `dct:identifier`, con la stringa `OS`, seguita dalla valorizzazione dell'attributo `id` dell'elemento `relation` di OSM che rappresenta il comune che si deve importare (con aggiunta di zeri iniziali fino al raggiungimento delle undici cifre), seguita dalla stringa `CO`;
- `foaf:name`, con la valorizzazione del tag `name` della `relation` di OSM che rappresenta il comune che si deve importare;
- `dct:alternative`, con la valorizzazione del tag `ref:catasto` della `relation` di

[5]Documenti XML che rappresentano le mappe in Open Street Map

[6]Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag

[7]http://wiki.openstreetmap.org/wiki/Tag:boundary%3Dadministrative#10_admin_level_values_for_specific_countries

[8]Lapo Bicchielli, Analisi e sviluppo di un sistema di acquisizione, rappresentazione ed accesso di informazioni geografiche e di servizio, statiche ed in tempo reale, per Smart City, Tesi di laurea magistrale, Anno accademico 2012 - 2013

OSM che rappresenta il comune che si deve importare;

- `km4c:isPartOfProvince`, v. Rappresentazione del legame tra le province e i comuni.

Rappresentazione del legame tra le province e i comuni

Nei documenti OSM[9], non esiste una mappatura esplicita tra comuni e province.

Tuttavia, `osmosis`[10] nel popolare il database relazionale Postgresql[11] (Simple[12] schema con estensione PostGIS[13]) carica sul database una vasta quantità di informazioni perimetrali attraverso le quali è possibile ricostruire i legami tra le province ed i comuni anche senza utilizzare lo strumento di localizzazione `nominatim`[14] cui si era ipotizzato in un primo momento di dover ricorrere.

In particolare, già oggi la relazione tra ciascun comune e la provincia a cui lo stesso appartiene, viene stabilita andando ad individuare per ciascun comune, tra tutte le province, quale sia la cui linea di confine racchiude completamente la linea di confine del comune, e si stabilisce con essa la corrispondenza.

Importazione di toponimi formati da più di un elemento stradale

I toponimi[15] formati da più di un elemento stradale sono rappresentati nei documenti OSM[16] nella forma di `relation` taggate[17] come segue:

- un tag `type` valorizzato a `route`;
- un tag `route` valorizzato a `road` oppure assente;
- un tag `network` assente o comunque non valorizzato a `e-road`;
- almeno una `way`[18] taggata con `highway` diverso da `proposed`[19].

Per ciascuna delle `relation` così individuate, si propone di introdurre nella KB[20] una nuova istanza della classe `km4c:Road`.

Nella nuova `km4c:Road` si propone di valorizzare il `dct:identifier` con la stringa `OS`, seguita dalla valorizzazione dell'attributo `id` della `relation` che rappresenta la strada preceduto da tanti zeri quanti ne occorrono per raggiungere le undici cifre, seguita dalla stringa `LR`.

Per la valorizzazione della proprietà `km4c:roadType`, essendo un dato che non viene esplicitamente rappresentato in OSM, si propone di comparare la parte iniziale del nome della strada (recuperabile dal tag `name` della `relation`) con l'elenco di tutte le possibili denominazioni urbanistiche generiche[21]. Se una corrispondenza viene trovata, si propone di utilizzare tale denominazione urbanistica generica per la valorizzazione della proprietà. Se nessuna corrispondenza

[9]Documenti XML che rappresentano le mappe in Open Street Map
[10]http://wiki.openstreetmap.org/wiki/Osmosis
[11]https://www.postgresql.org/
[12]http://wiki.openstreetmap.org/wiki/Osmosis/Detailed_Usage_0.45#PostGIS_Tasks_.28Simple_Schema.29
[13]http://postgis.net/
[14]http://nominatim.openstreetmap.org/
[15]Comunemente, via
[16]Documenti XML che rappresentano le mappe in Open Street Map
[17]Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag
[18]Un elemento `member`, figlio della `relation`, con attributo `type` valorizzato a `way` e attributo `ref` valorizzato con l'`id` dell'elemento `way` referenziato. L'elemento `way` corrisponde in questo caso ad un `km4c:RoadElement`.
[19]La valorizzazione `proposed` per il tag `highway` di una `way` di OSM, indica che il tratto di strada rappresentato dall'elemento `way` esiste in realtà soltanto sulla carta
[20]La Knowledge Base di Km4City
[21]http://www.laputa.it/denominazioni-urbanistiche-generiche/#15/45.6686/13.1041

viene trovata, si propone di non istanziare la proprietà.

Per la valorizzazione della proprietà `km4c:roadName`, si propone di utilizzare la valorizzazione del tag `name` della `relation`, rimuovendo se del caso la denominazione urbanistica generica come individuata al paragrafo precedente.

Per la valorizzazione della proprietà `km4c:extendName` si propone di utilizzare tal quale la valorizzazione del tag `name` della `relation`.

Per la valorizzazione della proprietà `dct:alternative` si propone di utilizzare tal quale la valorizzazione del tag `alt_name` della `relation`, se presente. Se il tag non è presente si propone di non istanziare la proprietà.

Gli elementi stradali di cui la strada si compone, sono rappresentati dagli elementi `member`, figli della `relation`, che recano un attributo `tag` valorizzato a `way`. L'attributo `ref` di tali elementi è l'`id` dell'elemento `way` di OSM che rappresenta il singolo elemento stradale.

Si propone di generare un nuovo `km4c:RoadElement` per ciascuno dei segmenti lineari nodo-nodo che compongono gli elementi `way` di cui al paragrafo precedente, di valorizzarne le proprietà secondo quanto descritto nel capitolo relativo alla Rappresentazione degli elementi stradali, e di introdurre all'interno dell'istanza della `km4c:Road` che stiamo creando, una nuova istanza della proprietà `km4c:containsElement` per ciascuno dei `km4c:RoadElement` creati, valorizzandola con l'`id` del `km4c:RoadElement`.

Si propone di valorizzare la proprietà `km4c:inMunicipalityOf` con l'istanza appropriata della classe `km4c:Municipality` individuata secondo quanto descritto nel capitolo dedicato alla Rappresentazione del legame tra i toponimi e i comuni.

Si propone di introdurre all'interno della Smart City Ontology la proprietà `km4c:inHamletOf`, sulla classe `km4c:Road`, da valorizzare con un'opportuna istanza della classe `km4c:Hamlet`, di cui si propone l'introduzione nella Smart City Ontology, istanza ottenuta secondo quanto descritto nel capitolo relativo alla Rappresentazione del legame tra i toponimi e i comuni.

Importazione di toponimi formati da un solo elemento stradale

Gli elementi `way`, presenti in documenti OSM[22], che non sono referenziati da alcuna delle `relation` di cui al capitolo sull'Importazione di toponimi formati da più di un elemento stradale, e che sono tuttavia taggati[23] con `highway` in qualsiasi modo valorizzato ad eccezione di `proposed`, descrivono in OSM un toponimo[24] formato da un solo elemento stradale.

Per ciascuno degli elementi `way` di OSM così individuati, si propone di introdurre nella KB[25] una nuova istanza della classe `km4c:Road` e di valorizzarne le proprietà in completa analogia con quanto visto per il caso dei toponimi formati da più elementi stradali, con le seguenti differenze:

- si propone di valorizzare la proprietà `dct:identifier` con la stringa `OS`, seguita dal valore dell'attributo `id` della `way` con aggiunta di zeri a sinistra fino al raggiungimento delle undici cifre, seguita dalla stringa `SR`;
- si propone di valorizzare le proprietà `roadType`, `roadName` ed `extendName` leggendo dal tag `name` della `way` invece che della `relation`;

[22]Documenti XML che rappresentano le mappe in Open Street Map
[23]Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag
[24]Comunemente, via
[25]La Knowledge Base di Km4City

- si propone di valorizzare la proprietà `dct:alternative` utilizzando il tag `alt_name` della `way`, se presente, e di non istanziare la proprietà nel caso non sia disponibile una denominazione alternativa;
- si propone di valorizzare la proprietà `km4c:inMunicipalityOf` con l'istanza della classe `km4c:Municipality` individuata attraverso la mappatura descritta nel capitolo dedicato alla Rappresentazione del legame tra i toponimi e i comuni;
- si propone di introdurre all'interno della Smart City Ontology la proprietà `km4c:inHamletOf`, sulla classe `km4c:Road`, da valorizzare con un'opportuna istanza della classe `km4c:Hamlet` di cui si propone l'introduzione, ottenuta secondo quanto descritto nel capitolo relativo alla Rappresentazione del legame tra i toponimi e i comuni.

Si propone di generare una nuova istanza della classe `km4c:RoadElement`, corrispondente all'unico elemento stradale componente il toponimo, e di valorizzarla come descritto nel capitolo relativo alla Rappresentazione degli elementi stradali**,** per poi introdurre all'interno dell'istanza della `km4c:Road`, una nuova istanza della proprietà `km4c:containsElement`, valorizzandola con l'`id` del `km4c:RoadElement`.

Rappresentazione del legame tra i toponimi e i comuni

Nei documenti OSM[26], il legame tra i toponimi[27] da un lato, ed i comuni e le frazioni dall'altro, non è rappresentato in modo esplicito.

Per stabilire il legame, si propone di stabilire la corrispondenza al livello del singolo elemento stradale, di legare cioè ciascun elemento stradale di ciascun toponimo, al comune e alla frazione o quartiere in cui si trova, utilizzando i punti estremi dell'elemento stradale.

In particolare, si propone di associare un elemento stradale ad un comune, nel caso in cui l'estremità iniziale, oppure l'estremità finale, dell'elemento stradale, ricada entro la superficie coperta dal comune.

Invece, si propone di associare un elemento stradale ad una frazione o quartiere, nel caso in cui entrambi i nodi, quello iniziale e quello finale, dell'elemento stradale, ricadano entro la superficie coperta dalla frazione o quartiere. Se la frazione, quartiere, o zona delimitata e denominata all'interno del territorio comunale, è rappresentata in OSM attraverso un nodo (quindi un punto) invece che attraverso una way o una relation (quindi un perimetro), si propone di non associare a quella frazione o quartiere alcun elemento stradale.

Per l'implementazione di tali corrispondenze, si propone di utilizzare l'informazione che `osmosis`[28] carica sul database relazionale, senza ricorrere, come si era ipotizzato in un primo momento, a strumenti di geo-localizzazione esterni (`nominatim`[29]). Per maggiori dettagli, si veda anche il capitolo relativo alla Rappresentazione del legame tra le province e i comuni.

Si propone di instaurare il legame tra i toponimi e i comuni, frazioni o quartieri di appartenenza, ereditandolo dal legame instaurato per gli elementi stradali che compongono il toponimo. In particolare, si stabilisce la corrispondenza tra un toponimo ed un comune, frazione o quartiere, se uno almeno degli elementi stradali di cui il toponimo si compone, ricade all'interno del comune, frazione o quartiere.

Rappresentazione delle frazioni e dei quartieri

Si propone di introdurre una nuova classe nella Smart City Ontology, la classe `km4c:Hamlet`, per

[26]Documenti XML che rappresentano le mappe in Open Street Map
[27]Comunemente, vie
[28]http://wiki.openstreetmap.org/wiki/Osmosis
[29]http://nominatim.openstreetmap.org/

rappresentare una frazione o un quartiere all'interno di un comune.

Si propone di istanziare un nuovo `km4c:Hamlet` per:

- ciascuna `way` di OSM taggata con `boundary` valorizzato ad `administrative` e con `admin_level` che assuma un valore maggiore di 8;
- ciascuna `relation` di OSM taggata con `type` valorizzato a `boundary`, ed inoltre con `boundary` valorizzato ad `administrative`, e con `admin_level` che assuma un valore superiore ad 8;
- ciascun `node` di OSM taggato con place valorizzato a `suburb`.

Da ciò si deduce che in realtà le istanze della classe `km4c:Hamlet` non rappresentano soltanto le frazioni e i quartieri propriamente detti, ma più in generale una zona all'interno dei confini di un comune a cui sia stata conferita una specifica denominazione, e che sia rappresentata su OSM in uno dei modi descritti.

Si propone di comporre l'URI di ciascuna istanza di `km4c:Hamlet` utilizzando l'URI del comune in cui la stessa si trova, seguito da una slash e dal nome della frazione o del quartiere, privato dei caratteri di spaziatura e di altri caratteri speciali eventualmente presenti.

Per ciascuna istanza di tale classe, si propone di istanziare la proprietà `km4c:inMunicipalityOf` da valorizzare con l'URI del comune in cui il quartiere o frazione si trova, e la proprietà `foaf:name` da valorizzare con la denominazione del quartiere o della frazione, instaurando il legame con il comune secondo quanto descritto nel capitolo relativo alla Rappresentazione del legame tra i toponimi e i comuni.

Inoltre, per ciascuna istanza di tale classe si propone di aggiungere due proprietà, `lat` e `long`, che descrivono in modo orientativo la posizione del `km4c:Hamlet`. Nel caso dei `km4c:Hamlet` istanziati a partire da una `way` o da una `relation` di OSM, si propone di valorizzare le proprietà `lat`, `long` con il centroide del perimetro rappresentato dalla `way` o dalla `relation`. Nel caso in cui invece la frazione, quartiere o zona sia rappresentata attraverso un `node` di OSM, si propone di valorizzare le proprietà `lat`, `long` ereditandole dalle corrispondenti proprietà definite sul nodo.

Rappresentazione degli elementi stradali

Sia nel caso dei toponimi[30] formati da un unico elemento stradale, sia nel caso dei toponimi formati da più elementi stradali, il singolo elemento stradale è rappresentato nei documenti OSM[31] attraverso elementi `way` taggati[32] come `highway`[33].

Va sottolineato che il tag `highway` non è in realtà utilizzato per i soli tratti di autostrada. Esso è invece applicato a tutti gli elementi `way` che rappresentano un elemento stradale, indipendentemente dall'importanza e dalle caratteristiche peculiari della stessa.

Va anche sottolineato che la presenza del tag `highway` è determinante per individuare in OSM le rappresentazioni degli elementi stradali. Gli elementi `way` infatti, rappresentano di per sé generiche linee, e vengono infatti utilizzati anche per rappresentare la forma degli edifici, i confini amministrativi, e molto altro. Soltanto quando l'elemento `way` è taggato come `highway`, esso rappresenta un elemento stradale, e contiene sia informazioni relative all'andamento del tratto di strada, sia altre informazioni su caratteristiche peculiari del tratto di strada rappresentato, queste

[30] Comunemente, vie

[31] Documenti XML che rappresentano le mappe in Open Street Map

[32] Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag

[33] `highway` è il nome del tag, che viene valorizzato in modo diverso a seconda del tipo di strada

ultime espresse nella forma di tag apposti sullo stesso elemento `way`.

Si propone quindi di generare una nuova istanza della classe `km4c:RoadElement`, per ciascuno dei segmenti che compongono ciascuno degli elementi `way` taggati come `highway`, e di valorizzare le proprietà della nuova istanza come descritto nel seguito di questo capitolo.

Pertanto, per ciascuna way di OSM che rappresenta un elemento stradale, si propone di individuare la sequenza dei nodi che delineano il tracciato della way, e per ciascun tratto rettilineo compreso tra due nodi successivi, di instanziare un nuovo `km4c:RoadElement`. Conviene osservare che tutti gli elementi stradali istanziati a partire da uno stesso elemento way di OSM, avranno le proprietà valorizzate in modo identico, dal momento che su OSM, sono tutti riconducibili ad un unico elemento stradale. Le uniche differenze si avranno sulle proprietà `km4c:startsAtNode`, `km4c:endsAtNode`, `km4c:route` (segmento di congiunzione tra i due nodi), e `length`. La trattazione che segue, in cui si vanno a proporre le valorizzazioni delle diverse proprietà dei `km4c:RoadElement` sulla base di quanto riscontrato su OSM, sottintende quanto appena descritto.

Si propone di comporre la URI del `km4c:RoadElement` concatenando il prefisso con la stringa `OS` seguita dall'id dell'elemento `way` associato seguita dalla stringa `RE`, seguita da una slash, e dalla posizione del `km4c:RoadElement` all'interno della `way` di OSM da cui è derivato, da intendere come numero di sequenza del nodo iniziale del `km4c:RoadElement` all'interno della sequenza dei nodi che delineano il tracciato della `way`.

Per la valorizzazione della proprietà `km4c:elementType`, si propone di utilizzare una tra le valorizzazioni ad oggi già presenti nella KB[34]. Si presenta di seguito l'elencazione delle valorizzazioni ad oggi utilizzate per questa proprietà[35], con indicazione per ciascuna delle condizioni necessarie per la sua applicazione, da valutare in ordine di elencazione, con la sola eccezione degli elementi che si trovano sulle rotatorie, per i quali la proprietà viene sempre istanziata con la valorizzazione "di rotatoria", eventualmente aggiunta ad altra valorizzazione nel caso in cui tutte le condizioni di cui ad uno dei punti che precedono la rotatoria nel seguente elenco risultino soddisfatte:

- `di parcheggio`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, assuma la valorizzazione `service`, ed il tag `service` assuma il valore `parking_aisle`;
- `di tronco carreggiata`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:
  - `motorway`;
  - `trunk`;
  - `primary`;
  - `secondary`;
  - `tertiary`;
  - `unclassified`;
  - `residential`;
  - `service` (con il distinto tag `service` non valorizzato a `parking_aisle`);
- `raccordo, bretella, svincolo`: si propone di utilizzare questa valorizzazione

[34]La Knowledge Base di Km4City
[35]Estratte interrogando la KB attraverso l'interfaccia Web disponibile all'indirizzo http://log.disit.org/spqlquery/

quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:

  - `motorway_link`;
  - `trunk_link`;
  - `primary_link`;
  - `secondary_link`;
  - `tertiary_link`;
  - `escape`;
  - `motorway_junction`;

- `di parcheggio strutturato`: si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, contenga un tag `amenity` valorizzato a `parking`;
- `di rotatoria`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:
  - `mini_roundabout`;
  - `turning_circle`;
  - `turning_loop`;

  oppure quando sia presente un tag `junction` valorizzato a `roundabout`;
- `di incrocio`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:
  - `traffic_signals`;
  - `stop`;
  - `give_way`;
- `di casello/barriera autostradale`: si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, abbia un elemento figlio `nd` che faccia riferimento tramite l'attributo `ref` ad un elemento `node` che includa un tag `barrier` valorizzato a `toll_booth`;
- `di piazza`: si propone di utilizzare questa valorizzazione quando sull'elemento `way` di OSM che rappresenta l'elemento stradale, sia presente un tag `area` valorizzato a `yes`;
- `controviale`: nell'attesa di meglio caratterizzare gli elementi stradali classificati come controviali nell'attuale KB, si propone di non fare ricorso a questa valorizzazione;
- `pedonale`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:
  - `pedestrian`;
  - `living_street`;

    - footway;
    - bridleway;
    - steps;
    - path;
    - crossing;
    - elevator;
- di passaggio a livello: si propone di utilizzare questa valorizzazione quando l'elemento way di OSM che rappresenta l'elemento stradale, abbia un elemento figlio nd che faccia riferimento tramite l'attributo ref ad un elemento node di OSM che includa un tag railway valorizzato a level_crossing;
- in area di pertinenza: si propone di utilizzare questa valorizzazione quando il tag highway dell'elemento way di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:
    - bus_stop;
    - emergency_access_point;
    - rest_area;
    - services;
- di area a traffico strutturato: si propone di utilizzare questa valorizzazione quando l'elemento way di OSM che rappresenta l'elemento stradale, contenga un tag lanes in qualsiasi modo valorizzato, e quando nessuna delle valorizzazioni precedenti sia risultata applicabile;
- di area a traffico non strutturato: si propone di utilizzare questa valorizzazione come residuale, per le situazioni in cui nessuna delle valorizzazioni precedenti sia risultata applicabile.

Si propone di valorizzare la proprietà km4c:elementClass riutilizzando una delle valorizzazioni già oggi in uso nella KB. Si riporta nel seguito l'elencazione delle valorizzazioni ad oggi in uso, ciascuna con l'indicazione delle condizioni che dovrebbero verificarsi affinché la stessa possa essere applicata, da valutare in ordine di elencazione:

- extraurbana secondaria: si propone di utilizzare questa valorizzazione quando il tag highway dell'elemento way di OSM che rappresenta l'elemento stradale, assuma una tra le valorizzazioni seguenti:
    - primary
    - secondary
    - tertiary
- autostrada: si propone di utilizzare questa valorizzazione quando il tag highway dell'elemento way di OSM che rappresenta l'elemento stradale, sia valorizzato a motorway;
- urbana di scorrimento: si propone di utilizzare questa valorizzazione quando il tag highway dell'elemento way di OSM che rappresenta l'elemento stradale, sia valorizzato a

`unclassified`[36];

- `extraurbana principale`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, sia valorizzato a `trunk`;
- `urbana di quartiere`: si propone di utilizzare questa valorizzazione quando il tag `highway` dell'elemento `way` di OSM che rappresenta l'elemento stradale, sia valorizzato a `residential`;
- `locale/vicinale/privata ad uso privato`: si propone di utilizzare questa valorizzazione come residuale, quando nessuna delle valorizzazioni precedenti risulti applicabile.

Per la valorizzazione della proprietà `km4c:composition`, per la quale le due valorizzazioni attualmente in uso sono `carreggiata unica` oppure `carreggiate separate`, si propone di utilizzare la valorizzazione `carreggiate separate` laddove esista una `relation` che sia taggata con `type` valorizzato a `dual_carriageway` e che referenzi l'elemento stradale oggetto di mappatura[37]. Si propone di utilizzare la valorizzazione `carreggiate separate` anche nel caso delle autostrade e delle strade extraurbane principali (si veda in proposito anche la valorizzazione della proprietà `km4c:elementClass`). Si propone di utilizzare la valorizzazione `carreggiata unica` in tutti i rimanenti casi.

Per la valorizzazione della proprietà `km4c:elemLocation`, si propone di riutilizzare le valorizzazioni che sono attualmente già impiegate nell'attuale KB. Si propone di seguito l'elencazione di tali valorizzazioni, ciascuna con indicazione delle condizioni al ricorrere delle quali ciascuna valorizzazione dovrebbe essere applicata, da valutare in ordine di elencazione:

- `galleria, ponte e rampa`: si propone di utilizzare questa valorizzazione quando siano verificate le condizioni seguenti:
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `bridge` valorizzato a `yes` oppure referenziato da una `relation` taggata con `type` valorizzato a `bridge`;
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `tunnel` valorizzato a `yes`;
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `highway` che assuma una tra le seguenti valorizzazioni:
    - `motorway_link`;
    - `trunk_link`;
    - `primary_link`;
    - `secondary_link`;
    - `tertiary_link`;
    - `escape`;
- `ponte e rampa`: si propone di utilizzare questa valorizzazione quando siano verificate le

[36]The word 'unclassified' is a historical artefact of the UK road system and does not mean that the classification is unknown; you can use highway=road for that. http://wiki.openstreetmap.org/wiki/Key:highway

[37]Che abbia cioè un elemento figlio `member` con attributo `type` valorizzato a `way` e con attributo `ref` valorizzato con l'`id` dell'elemento `way` che si sta mappando

condizioni seguenti:

  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `bridge` valorizzato a `yes` oppure referenziato da una `relation` taggata con `type` valorizzato a `bridge`;
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `highway` che assuma una tra le seguenti valorizzazioni:
    - `motorway_link`;
    - `trunk_link`;
    - `primary_link`;
    - `secondary_link`;
    - `tertiary_link`;
    - `escape`;

- `galleria e rampa`: si propone di utilizzare questa valorizzazione quando siano verificate le condizioni seguenti:
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `tunnel` valorizzato a `yes`;
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `highway` che assuma una tra le seguenti valorizzazioni:
    - `motorway_link`;
    - `trunk_link`;
    - `primary_link`;
    - `secondary_link`;
    - `tertiary_link`;
    - `escape`;
- `ponte e galleria`: si propone di utilizzare questa valorizzazione quando siano verificate le condizioni seguenti:
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `bridge` valorizzato a `yes` oppure referenziato da una `relation` taggata con `type` valorizzato a `bridge`;
  - l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `tunnel` valorizzato a `yes`;
- `ponte`: si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `bridge` valorizzato a `yes` oppure referenziato da una `relation` taggata con `type` valorizzato a `bridge`;
- `rampa`, si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `highway` che assuma una tra le seguenti valorizzazioni:
  - `motorway_link`;

    - trunk_link;
    - primary_link;
    - secondary_link;
    - tertiary_link;
    - escape;
- galleria: si propone di utilizzare questa valorizzazione quando l'elemento way di OSM che rappresenta l'elemento stradale, sia taggato con tunnel valorizzato a yes;
- a raso: si propone di utilizzare questa valorizzazione come residuale, nel caso in cui nessuna delle precedenti valorizzazioni risulti applicabile.

Per la valorizzazione della proprietà km4c:length calcolando con l'apposita funzione PostGIS la distanza tra i due nodi che delimitano l'elemento stradale. Infatti, il tag length è talvolta apposto sulle way di OSM, ma i km4c:RoadElement sono in generale dei frammenti di way, e quindi la loro lunghezza è inferiore a quella riportata nel tag length eventualmente presente sulla way di OSM.

Per la valorizzazione della proprietà km4c:width si propone di utilizzare il tag width oppure est_width dell'elemento way di OSM che rappresenta l'elemento stradale. Anche in questo caso, in fase di implementazione, si propone di validare la valorizzazione riportata in OSM, analizzarla, e compararla con opportuni valori di soglia, di modo da ricondursi ad una delle tre valorizzazioni riscontrabili nell'attuale KB, che sono segnatamente:

- minore di 3,5 mt;
- tra 3,5 mt e 7 mt;
- maggiore di 7 mt.

Nel caso in cui il dato relativo alla larghezza non fosse disponibile, si propone di utilizzare la valorizzazione non rilevato anch'essa riscontrabile già nell'attuale KB.

Per la valorizzazione della proprietà km4c:operatingStatus si propone di procedere nel seguente modo:

- quando il tag highway dell'elemento way di OSM che rappresenta l'elemento stradale, sia valorizzato a construction, si propone di utilizzare la valorizzazione in costruzione;
- quando il tag abandoned dell'elemento way di OSM che rappresenta l'elemento stradale, sia valorizzato a yes, oppure il tag disused sia valorizzato a yes, si propone di utilizzare la valorizzazione in disuso;
- nel caso in cui nessuna delle valorizzazioni precedenti risulti applicabile, si propone di utilizzare la valorizzazione in esercizio.

Per la valorizzazione della proprietà km4c:speedLimit si propone di utilizzare il tag maxspeed dell'elemento way di OSM che rappresenta l'elemento stradale. Anche in questo caso, in OSM è possibile esprimere la velocità massima in un'unità di misura diversa dalla predefinita (km/h). Anche in questo caso si propone quindi di analizzare in fase di implementazione il valore del tag, convertirlo se necessario, e rappresentarlo come numero intero, per uniformarsi alle valorizzazioni già presenti nella KB. Si propone inoltre di non istanziare la proprietà nel caso in cui il tag non risulti presente.

Per la valorizzazione della proprietà km4c:trafficDir, si propone di adottare una tra le

valorizzazioni seguenti, da valutare in ordine di elencazione:

- `tratto stradale chiuso in entrambe le direzioni`: si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `highway` valorizzato a `construction`, ed anche quando l'elemento `way` di OSM sia taggato con `access` valorizzato a `no`;
- `tratto stradale aperto nella direzione positiva (da giunzione NOD_INI a giunzione NOD_FIN)`: si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `oneway` valorizzato a `yes`;
- `tratto stradale aperto nella direzione negativa (da giunzione NOD_FIN a giunzione NOD_INI)`: si propone di utilizzare questa valorizzazione quando l'elemento `way` di OSM che rappresenta l'elemento stradale, sia taggato con `oneway` valorizzato a -1;
- `tratto stradale aperto in entrambe le direzioni (default)`: si propone di utilizzare questa valorizzazione come residuale, nel caso in cui nessuna delle valorizzazioni precedenti risulti applicabile.

Si propone di valorizzare la proprietà `km4c:managingAuthority` con l'istanza opportuna della classe `km4c:Municipality` individuata come descritto nel capitolo dedicato alla Rappresentazione del legame tra i toponimi e i comuni.

Si propone di introdurre all'interno della Smart City Ontology la proprietà `km4c:inHamletOf`, sulla classe `km4c:RoadElement`, da valorizzare con un'opportuna istanza della classe `km4c:Hamlet` di cui si propone l'introduzione, ottenuta come descritto nel capitolo dedicato alla Rappresentazione del legame tra i toponimi e i comuni.

Si propone di introdurre all'interno della Smart City Ontology la proprietà `km4c:highwayType`, da valorizzare riprendendo tal quale la valorizzazione del tag `highway` presente sull'elemento `way` di Open Street Map a partire dal quale l'elemento stradale di Km4City è generato. La proprietà è per certi versi ridondante, andando ad accedere allo stesso dato sorgente cui si accede per la valorizzazione delle proprietà `km4c:elemLocation` e `km4c:elementClass`, ma è stata comunque introdotta perché richiesta per un più agevole calcolo dei flussi di traffico.

Rappresentazione dei numeri civici e degli accessi

Nei documenti OSM[38], i numeri civici sono rappresentati attraverso il tag[39] `housenumber`, che viene applicato non soltanto alle private abitazioni ma a qualsiasi genere di edificio, pubblico o privato, indipendentemente dalla propria destinazione.

Il tag `housenumber` di OSM è di regola applicato ad elementi `node` (ed è tipicamente il caso delle abitazioni private). Residualmente, il tag può trovarsi applicato anche su elementi `way` che rappresentano outline di edifici. Entrambi i casi sono gestiti (il caso degli edifici dal dicembre ’17).

In particolare, si propone di utilizzare, per il recupero dell'informazione relativa ai numeri civici e alla loro localizzazione, i seguenti elementi `node`:

1. gli elementi `node` che siano taggati con un indirizzo completo, attraverso l'apposizione dei seguenti tre tag: `housenumber` (numero civico), `street` (toponimo completo) e `city`

[38]Documenti XML che rappresentano le mappe in Open Street Map

[39]Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag

(comune);

2. gli elementi `node` che siano taggati con il numero civico, e siano posti in relazione con il toponimo attraverso un apposito elemento `relation` taggato con `type` valorizzato a `associatedStreet`;
3. gli elementi `node` che siano taggati con il numero civico, e siano nodi di giunzione di un elemento stradale;
4. gli elementi `node` che siano taggati con il numero civico e con il `place`, anche in assenza della `street`.

Per ciascuno dei `node` così individuati, si propone di generare una nuova istanza della classe `km4c:StreetNumber` valorizzandone le proprietà come segue:

- `dct:identifier`: si propone di valorizzarlo con la stringa `OS`, seguita dall'`id` dell'elemento `node` di OSM con aggiunta di zeri a sinistra fino al raggiungimento delle undici cifre, seguito dalla stringa `NN`. Si propone di utilizzare questo identificativo anche come parte distintiva della URI dell'istanza;
- `km4c:classCode`: la numerazione duplicata, con numeri rossi e neri, pare esistere soltanto a Firenze, Genova e Savona[40]. Per queste città, guardando ai dati presenti su OSM, un approccio ragionevole potrebbe essere quello di ricercare la lettera R maiuscola o minuscola nel valore del tag `housenumber`: se la lettera è presente, si propone di utilizzare la valorizzazione `Rosso`, altrimenti `Nero`. Per tutte le altre città, si propone di utilizzare la valorizzazione `Privo colore`;
- `km4c:number`, `km4c:exponent`, `km4c:extendNumber`: in OSM, il numero civico è inserito sempre in formato esteso all'interno del tag `housenumber`. Per la valorizzazione di queste tre proprietà, si propone quindi di procedere in fase di implementazione ad analizzare lo `housenumber`, individuando la parte numerica e la parte letterale eventualmente presente, e valorizzando quindi le tre proprietà di conseguenza, quindi la parte numerica in `km4c:number`, la parte letterale in `km4c:exponent`, e il numero completo in `km4c:extendNumber`;
- `km4c:hasExternalAccess`: si propone di valorizzare questa proprietà con l'istanza della classe `km4c:Entry` (v. seguito) che contiene la posizione in termini di coordinate geospaziali, a cui il numero civico si trova;
- `km4c:belongToRoad`: si propone di valorizzare questa proprietà in modo diverso a seconda dei casi, ed in particolare:
    1. nel caso dei nodi che siano taggati con un indirizzo completo, si propone di individuare, cercando tra le `relation` che rappresentano toponimi formati da più di un elemento stradale e tra le `way` che rappresentano toponimi formati da un solo elemento stradale, l'elemento di OSM che rappresenta il toponimo indicato nel `node`, e di associare il `node` a tale elemento, valorizzando in particolare la proprietà `km4c:belongToRoad` con l'URI della `km4c:Road` associata a tale elemento;
    2. nel caso di numeri civici che siano associati al toponimo per tramite dell'apposita `relation` taggata con `type` valorizzato a `associatedStreet`, si propone di individuare all'interno di tale `relation` il membro di tipo `way`, che rappresenta l'elemento stradale su cui il numero civico si trova. Noto l'elemento stradale, l'individuazione del toponimo è pressoché immediata, infatti nel caso dei toponimi

[40] http://www.ilsecoloxix.it/p/genova/2014/09/06/ARe3j4qB-numeri_perche_genova.shtml

formati da un solo elemento stradale, l'elemento stradale rappresenta anche il toponimo, mentre nel caso soltanto leggermente più complesso dei toponimi formati da più di un elemento stradale, noto l'elemento stradale (`way` di OSM) è possibile risalire alla `km4c:Road` andando ad individuare quale sia la `relation` di tipo `route` in OSM che referenzia la `way` che rappresenta l'elemento stradale;

3. nel caso dei numeri civici che siano nodi di giunzione di elementi stradali, il legame con l'elemento stradale è immediato, e noto l'elemento stradale in cui il numero civico si trova, il legame con il toponimo è immediato o pressoché immediato, come descritto al punto precedente;
4. nel caso in cui il numero civico abbia soltanto il `place`, e nessuna indicazione relativa alla strada, questa proprietà non viene istanziata.

- `km4c:place`: nel caso di numeri civici che abbiano un tag place, la proprietà è istanziata e valorizzata esattamente con il valore del tag place così come lo stesso si trova su OSM.

Inoltre, per ciascuno degli `km4c:StreetNumber` istanziati, si propone di generare una nuova istanza della classe `km4c:Entry`, di legarla al nodo che rappresenta il numero civico attraverso la proprietà `km4c:hasExternalAccess` dello `km4c:StreetNumber`, e di valorizzarne le rimanenti proprietà nel seguente modo:

- `dct:identifier`: si propone di valorizzarlo con la stringa `OS`, seguita dall'`id` dell'elemento `node` di OSM taggato con il numero civico di cui si sta rappresentando il punto di accesso, con aggiunta di zeri a sinistra fino al raggiungimento delle undici cifre, seguito dalla stringa `NE`. Si propone di utilizzare questo identificativo anche come parte distintiva della URI dell'istanza;
- `km4c:entryType`: si propone di considerare il punto di accesso come un `Accesso diretto esterno`;
- `geo:lat`, `geo:long`: si propone di valorizzare le due proprietà con le coordinate geospaziali del `node` di OSM taggato con il numero civico di cui si sta rappresentando il punto di accesso, riportate come valorizzazioni degli attributi `lat` e `lon` dello stesso elemento `node`;
- `km4c:porteCochere`: nel caso in cui l'elemento `node` di OSM che rappresenta il punto di accesso al numero civico, sia taggato con `motorcycle` o `motorcar` valorizzati a `yes`, si propone di utilizzare la valorizzazione `Accesso carrabile`. Diversamente, si propone di utilizzare la valorizzazione `Accesso non carrabile`;
- `km4c:belongToElement`: questa proprietà deve essere valorizzata con l'elemento stradale su cui il numero civico si trova, individuato sulla base della distanza tra l'accesso e ciascuno degli elementi stradali componenti la strada su cui il numero civico si trova.

Inoltre, alcuni edifici (elementi `way` taggati come `building`) hanno `addr:street` e `addr:housenumber`. Per ognuna di queste `way`, si va a recuperare il nodo della `way` taggato come `entrance` = {`yes`, `main`, `staircase`}, o come `building=entrance` (deprecato ma molto utilizzato in passato), o come `barrier`, e si mappa quel nodo esattamente come si farebbe se l'indirizzo fosse stato scritto direttamente su quel nodo invece che sulla way a cui lo stesso appartiene. Se nessun nodo della way è taggato come ingresso, viene considerato come ingresso il nodo della `way` più vicino alla strada su cui l'edificio si affaccia.

Rappresentazione delle milestone

I milestone sono rappresentati nei documenti OSM[41] attraverso tag[42] `highway` valorizzati a `milestone` apposti su elementi `node`.

Si propone di generare una nuova istanza della classe `km4c:Milestone` della Smart City Ontology, per ciascun elemento `node` di OSM taggato con `highway` valorizzato a `milestone`, e con `distance` valorizzato con l'indicazione chilometrica, e di stabilire il collegamento tra l'istanza del `km4c:Milestone` e l'istanza del `km4c:RoadElement` andando a verificare, tra tutti gli elementi `way` di OSM, quale tra essi contiene il riferimento al nodo taggato come `milestone`, ed associando quindi l'istanza del `km4c:Milestone` con l'istanza del `km4c:RoadElement` corrispondente all'elemento `way` in tal modo individuato.

Si propone inoltre di valorizzare la proprietà `km4c:text` dell'istanza di `km4c:Milestone` leggendo dal tag `distance` presente sui nodi OSM taggati come `milestone`.

Si propone inoltre di valorizzare la latitudine, la longitudine e la geometria dell'istanza del `km4c:Milestone`, andando a leggere dai corrispondenti attributi del nodo OSM taggato come `milestone`.

Rappresentazione delle modificazioni temporanee della viabilità

Nella Smart City Ontology di Km4City[43], è prevista la possibilità di valorizzare la proprietà `km4c:accessToElement` di un elemento stradale con un'istanza della classe `km4c:EntryRule` che descrive una modificazione temporanea della viabilità, ad esempio a causa di lavori o manifestazioni.

Nei documenti OSM[44], questo tipo di situazioni viene rappresentato attraverso l'utilizzo del prefisso `temporary`[45]. In particolare, laddove un elemento stradale sia oggetto di modifiche temporanee di una o più delle sue proprietà, sono introdotte sull'elemento delle nuove proprietà con prefisso `temporary` che affiancano le precedenti e che comandano su di esse per un intervallo temporale ben delimitato, specificato nella valorizzazione delle stesse proprietà `temporary`.

Per quanto detto, si propone di rappresentare le modifiche temporanee della viabilità nel modo seguente:

- nel caso in cui l'elemento `way` di OSM che rappresenta l'elemento stradale interessato dalla variazione, sia taggato[46] con `temporary:access` valorizzato a `no`, si propone di valorizzare la proprietà `km4c:accessToElement` del `km4c:RoadElement` corrispondente, con una istanza della classe `km4c:EntryRule` che riporti la valorizzazione `Direzione flusso di traffico` sulla proprietà `km4c:restrictionType` e la valorizzazione `Chiusa in entrambe le direzioni` sulla proprietà `km4c:restrictionValue`, e che abbia inoltre la proprietà `km4c:hasRule` valorizzata con il riferimento all'indietro verso l'istanza del `km4c:RoadElement` oggetto della variazione;
- nel caso in cui l'elemento `way` di OSM che rappresenta l'elemento stradale oggetto della

[41]Documenti XML che rappresentano le mappe in Open Street Map
[42]Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag
[43]Per maggiori informazioni sul progetto, si visiti http://www.km4city.org/
[44]Documenti XML che rappresentano le mappe in Open Street Map
[45]http://wiki.openstreetmap.org/wiki/Proposed_features/temporary_(conditional)
[46]Un tag apposto su di un elemento di un documento OSM, è un elemento figlio `tag` con un attributo `k` valorizzato con il nome del tag, ed un attributo `v` che reca la valorizzazione del tag

variazione, sia taggato con `temporary:oneway` valorizzato a `yes`, si propone di valorizzare la proprietà `km4c:accessToElement` del `km4c:RoadElement` corrispondente, con un'istanza della classe `km4c:EntryRule` che riporti la valorizzazione `Direzione flusso di traffico` sulla proprietà `km4c:restrictionType` e la valorizzazione `Chiusa in direzione negativa` sulla proprietà `km4c:restrictionValue`, e che abbia inoltre la proprietà `km4c:hasRule` valorizzata con il riferimento all'indietro verso l'istanza del `km4c:RoadElement` oggetto della variazione. Se però, in abbinamento al tag `temporary:oneway`, l'elemento `way` di OSM che rappresenta l'elemento stradale oggetto della variazione, ha anche un tag `temporary:access` valorizzato a `destination`, allora si propone di valorizzare la proprietà `km4c:restrictionType` con la dicitura `Passaggio bloccato`, e la proprietà `km4c:restrictionValue` con la dicitura `Blocco fisico sulla giunzione finale`;

- nel caso in cui l'elemento `way` di OSM che rappresenta l'elemento stradale oggetto della variazione, sia taggato con `temporary:oneway` valorizzato a `-1`, si propone di valorizzare la proprietà `km4c:accessToElement` del `km4c:RoadElement` corrispondente, con un'istanza della classe `km4c:EntryRule` che riporti la valorizzazione `Direzione flusso di traffico` sulla proprietà `km4c:restrictionType` e la valorizzazione `Chiusa in direzione positiva` sulla proprietà `km4c:restrictionValue`, e che abbia inoltre la proprietà `km4c:hasRule` valorizzata con il riferimento all'indietro verso l'istanza del `km4c:RoadElement` oggetto della variazione. Se però, in abbinamento al tag `temporary:oneway`, l'elemento `way` di OSM che rappresenta l'elemento stradale oggetto della variazione, ha anche un tag `temporary:access` valorizzato a `destination`, allora si propone di valorizzare la proprietà `km4c:restrictionType` con la dicitura `Passaggio bloccato`, e la proprietà `km4c:restrictionValue` con la dicitura `Blocco fisico sulla giunzione iniziale`.

Rappresentazione delle estese amministrative

Il concetto di estesa amministrativa non pare essere rappresentato esplicitamente nei documenti OSM[47], né pare possibile derivarne una rappresentazione poggiando su fonti esterne da incrociare con i dati di OSM.

Infatti, un'estesa amministrativa è per definizione un insieme di elementi stradali esattamente delimitati raccolti dall'ente gestore della strada sotto un'unica denominazione, ortogonale rispetto alla toponomastica.

Tali segmenti stradali, in generale, non esistono in OSM (e non sono quindi accorpabili, neppure attingendo il criterio di accorpamento da una fonte esterna), dal momento che gli utenti che contribuiscono ad OSM non posseggono in generale questo tipo di informazioni, e quindi definiscono i segmenti stradali (elementi `way` di OSM) in un modo che, in generale, non ricalca le definizioni degli elementi stradali utili alla definizione delle estese amministrative.

Rappresentazione delle restrizioni di svolta

La Smart City Ontology non prevedeva originariamente la possibilità di descrivere vincoli di direzione obbligatoria, che sono invece determinanti per poter calcolare efficacemente un percorso.

Ci si pone quindi il problema di come sia possibile introdurre tali vincoli all'interno dell'ontologia.

Si propone di introdurre una nuova classe `km4c:TurnRestriction` per ciascun elemento

[47]Documenti XML che rappresentano le mappe in Open Street Map

`relation` di OSM che sia taggato con `type` valorizzato a `restriction`.

Tali `relation` sono infatti deputate a descrivere esattamente i vincoli di svolta in corrispondenza di un'intersezione tra strade.

In particolare, tali `relation` hanno:

- un membro di tipo `way` che rappresenta la strada (o il tratto di strada) di provenienza, il cui `member_role` è valorizzato a `from`;
- un membro di tipo `way` che rappresenta la strada (o il tratto di strada) di destinazione, il cui `member_role` è valorizzato a `to`;
- un nodo che rappresenta il punto in cui le due `way` si incrociano (oppure una `way`, nel raro caso in cui si debba descrivere un vincolo di direzione obbligatoria tra due `way` che non hanno nodi a comune sul grafo stradale), il cui `member_role` è valorizzato a `via`;
- un tag `restriction` la cui valorizzazione indica esattamente qual'è il tipo di restrizione che si incontra quando proveniendo dalla `way` "`from`" ci si immetta sulla `way` "`to`".

Una descrizione maggiormente dettagliata di questo tipo di relazioni è disponibile sul Wiki[48] di Open Street Map.

Ciò che si propone di importare nella Smart City Ontology è soltanto la URI della strada (toponimo) di provenienza, la URI del toponimo di destinazione, e il tipo di restrizione che si incontra quando dal toponimo di provenienza ci si voglia immettere sul toponimo di destinazione.

In particolare, si propone pertanto di valorizzare:

- la proprietà `km4c:where` dell'istanza della `km4c:TurnRestriction` con la URI dell'elemento stradale di provenienza, che è l'elemento stradale che si trova nella OSM way di provenienza, e che ha il nodo richiamato dalla restrizione di svolta su OSM come nodo iniziale o finale;
- la proprietà `km4c:toward` dell'istanza della `km4c:TurnRestriction` con la URI dell'elemento stradale di destinazione, che è l'elemento stradale che si trova nella OSM way di destinazione, e che ha il nodo richiamato dalla restrizione di svolta su OSM come nodo iniziale o finale;
- la proprietà `km4c:restriction` dell'istanza della `km4c:TurnRestriction` con la valorizzazione del tag `restriction` della `relation`.

Nel caso molto particolare[49] in cui la restrizione di svolta sia temporanea, allora avremo l'elemento `relation` di OSM taggato con due o più delle seguenti etichette:

- `day_on`, giorno di inizio della restrizione;
- `day_off`, giorno di fine della restrizione;
- `hour_on`, orario di inizio della restrizione;
- `hour_off`, orario di fine della restrizione.

Soltanto al verificarsi di questo particolare caso, si propone di introdurre sull'istanza della `km4c:TurnRestriction` omonime proprietà, e di valorizzarle esattamente come sono valorizzati i tag.

[48] http://wiki.openstreetmap.org/wiki/IT:Relation:restriction
[49] Una sola relazione per tutto il centro Italia

Nel caso anch'esso particolare in cui la restrizione non si applichi a determinate categorie di utenza[50], allora avremo l'elemento `relation` di OSM taggato con `except`, ed il tag valorizzato con l'elencazione delle tipologie di utenza alle quali la restrizione non si applica. Soltanto in questo particolare caso, si propone di istanziare la proprietà `km4c:except` sull'istanza della `km4c:TurnRestriction` e di valorizzarla con la valorizzazione del tag `except` di OSM.

Si propone di assegnare all'istanza di `km4c:TurnRestriction` una URI formata dalla URI della `km4c:Road` di provenienza, seguita dalla stringa `/restriction/turn/`, seguita dal `dct:identifier` della `km4c:Road` di destinazione.

Si propone di assegnare all'istanza di `km4c:TurnRestriction` anche il tipo `km4c:Restriction`, di nuova introduzione, generalizzazione di `km4c:TurnRestriction`.

Rappresentazione delle restrizioni di accesso

Una visione d'insieme sulla rappresentazione delle restrizioni di accesso in Open Street Map è fornita nel Wiki[51].

Si propone di rappresentare ciascuna restrizione di accesso all'interno di KM4C come un'istanza della classe `km4c:AccessRestriction`.

Si propone di assegnare a ciascuna istanza di `km4c:AccessRestriction` una URI formata dalla URI della `km4c:Road`, o del `km4c:RoadElement`, o del `km4c:Node` cui la restrizione si applica, concatenata con la stringa `/restriction/access/`, concatenata con la valorizzazione della proprietà `km4c:who` dell'istanza (o con `everybody` se non valorizzata), concatenata con la valorizzazione della proprietà `km4c:direction` dell'istanza (o con `alldirections` se non valorizzata), eventualmente concatenata con la valorizzazione della proprietà `km4c:condition` dell'istanza al netto dei caratteri speciali (o con `unconditioned` se non valorizzata). Ciascuna parte componente la URI dovrebbe essere separata dalle altre per tramite di una barra. Nel caso in cui la restrizione sia riferita ad una particolare corsia (lane), alla URI come sin qui proposta dovrebbe essere giustapposta la stringa `/lanes/` seguita dal numero d'ordine della corsia, procedendo da sinistra verso destra rispetto a chi guida.

Si propone di istanziare per ciascuna istanza di `km4c:AccessRestriction`, il seguente insieme di proprietà:

- `km4c:where`, da valorizzare con la URI della `km4c:Road`, o del `km4c:RoadElement`, o del `km4c:Node` a cui la restrizione si applica;
- `km4c:who`, da valorizzare con una stringa che indica il mezzo di trasporto cui la restrizione si applica. Nel caso in cui la restrizione si applichi a tutti i mezzi di trasporto indistintamente, si propone di non istanziare questa proprietà;
- `km4c:direction`, da valorizzare con `forward` oppure con `backward` nei soli casi in cui la restrizione si applichi ad una sola delle due direzioni della `km4c:Road` o del `km4c:RoadElement`. Nel caso in cui la restrizione si applichi ad entrambe le direzioni, si propone di non istanziare questa proprietà;
- `km4c:access`, è la proprietà principale, che indica il tipo di accesso, da valorizzare con la valorizzazione del tag access di OSM;
- `km4c:condition`, da valorizzare con la condizione al verificarsi della quale la restrizione

[50] Poche decine di casi nell'Italia centrale
[51] http://wiki.openstreetmap.org/wiki/OSM_tags_for_routing#Access-Restrictions, paragrafo “Access-Restrictions”

si applica. Nel caso in cui la restrizione non sia di tipo condizionale (quindi, nel caso in cui la chiave del tag su OSM non termini con `:conditional`), si propone di non istanziare questa proprietà.

Definita la proposta per quanto riguarda la rappresentazione delle restrizioni di accesso in OSM, si presenta nel seguito una proposta di mappatura dei tag di OSM sulle `km4c:AccessRestriction` come appena delineate:

- nel caso in cui su di un `node`, `way` o `relation` di OSM sia apposto un tag la cui chiave inizi per `access`, che sta solitamente ad indicare una restrizione che si applica a tutte le tipologie di veicoli (residualmente, possono aversi chiavi che iniziano per access e proseguono con l'indicazione del mezzo di trasporto cui la restrizione si applica, da trattare come nel seguito specificato), si propone di istanziare una `km4c:AccessRestriction` valorizzando la proprietà `km4c:where` con la URI dell'istanza di KM4C cui la restrizione si applica, ed inoltre la proprietà `km4c:access` con la valorizzazione del tag `access`. Nel caso in cui lo stesso elemento di OSM sia taggato con l'indicazione dei giorni in cui la restrizione si applica (chiavi `day_on`, `day_off`, `date_on`, `date_off`) si propone di valorizzare la proprietà `km4c:condition` riportando la valorizzazione di `day_on` o di `date_on`, seguita da un trattino, e dalla valorizzazione di `day_off` o di `date_off`. Nel caso in cui lo stesso elemento di OSM sia taggato con l'indicazione dell'orario in cui la restrizione si applica (chiavi `hour_on`, `hour_off`) si propone di valorizzare la proprietà `km4c:condition` riportando la valorizzazione del tag `hour_on` seguita da un trattino e dalla valorizzazione del tag `hour_off`. Nel caso in cui l'indicazione dei giorni e degli orari coesistano sullo stesso elemento, si propone di riportare sulla proprietà `km4c:condition` per prima cosa i giorni come già indicato, quindi gli orari come già indicato, separando le due condizioni con uno spazio. Nel caso in cui la chiave comprenda anche l'indicazione `:forward` o `:backward` si propone di istanziare e valorizzare coerentemente anche la proprietà `km4c:direction`. Nel caso, residuale, in cui la chiave del tag, pur iniziando per `access`, comprenda poi anche l'indicazione di un mezzo di trasporto (residuale perché in questa circostanza la maggior parte dei contributori inserisce direttamente il mezzo di trasporto senza indicare `access`) allora si propone di valorizzare anche la proprietà `km4c:who` riportando come stringa l'indicazione del mezzo di trasporto cui la restrizione si applica;
- nel caso in cui su di un `node`, `way` o `relation` di OSM sia apposto un tag la cui chiave inizia con l'indicazione di un mezzo di trasporto [52], si propone di istanziare una `km4c:AccessRestriction` valorizzando `km4c:where` con la URI dell'elemento KM4C cui la restrizione si applica, `km4c:who` con l'indicazione del mezzo di trasporto (riportando quindi esattamente tal quale la chiave del tag), e `km4c:access` con la valorizzazione del tag di OSM che impone la restrizione di accesso. Nel caso in cui lo stesso elemento di OSM sia taggato con l'indicazione dei giorni in cui la restrizione si applica (chiavi `day_on`, `day_off`, `date_on`, `date_off`) si propone di valorizzare la proprietà `km4c:condition` riportando la valorizzazione di `day_on` o di `date_on`, seguita da un trattino, e dalla valorizzazione di `day_off` o di `date_off`. Nel caso in cui lo stesso elemento di OSM sia taggato con l'indicazione dell'orario in cui la restrizione si applica (chiavi `hour_on`, `hour_off`) si propone di valorizzare la proprietà `km4c:condition` riportando la valorizzazione del tag `hour_on` seguita da un trattino e dalla valorizzazione del tag `hour_off`. Nel caso in cui l'indicazione dei giorni e degli orari coesistano sullo stesso elemento, si propone di riportare sulla proprietà `km4c:condition` per prima cosa i giorni

[52] Per l'elencazione dei mezzi di trasporto previsti ed utilizzabili come chiavi dei tag di restrizione di accesso, si veda il paragrafo “Land-based transportation” alla pagina http://wiki.openstreetmap.org/wiki/Key:access

come già indicato, quindi una virgola, e infine gli orari come già indicato. Nel caso in cui la chiave del tag termini per `:forward` o per `:backward` si propone di istanziare e valorizzare coerentemente anche la proprietà `km4c:direction`;

- nel caso in cui un `node`, `way`, o `relation` di OSM abbia un tag con chiave `oneway` valorizzato a `yes` oppure a `1` si propone di istanziare una `km4c:AccessRestriction` valorizzando `km4c:where` con la URI dell'elemento KM4C cui si applica la restrizione, `km4c:who` con `vehicle`, `km4c:direction` con `backward`, e `km4c:access` con `no`. Se invece il tag è valorizzato a `-1`, si propone di valorizzare `km4c:direction` con `forward`, ferme restando le rimanenti valorizzazioni. Se la chiave del tag contiene anche, come prefisso o come suffiso, separata da `oneway` con i due punti, l'indicazione del mezzo di trasporto cui la restrizione si applica, allora si propone di valorizzare la proprietà `km4c:who`, invece che con `vehicle`, con l'indicazione del mezzo di trasporto cui la restrizione si applica;
- nel caso in cui un nodo, una way o una relation di OSM sia taggato con `cycleway`, e la valorizzazione del tag comprenda la stringa `opposite`, significa (o dovrebbe significare secondo le indicazioni di OSM) che ci troviamo su di una strada a senso unico, che i cicli possono percorrere in senso inverso rispetto agli altri veicoli, sfruttando una corsia a loro dedicata. In questo caso, si propone di istanziare una `km4c:AccessRestriction` valorizzando `km4c:where` con la URI dell'elemento KM4C cui la restrizione si applica, `km4c:who` con `bicycle`, `km4c:direction` con l'indicazione della direzione che è consentita ai cicli diversamente dagli altri veicoli, `km4c:access` con `yes`.

Stante quanto sin qui descritto, se la chiave del tag comprende anche l'indicazione `:conditional`, allora si propone di valorizzare anche la proprietà `km4c:condition` riportando tal quale la condizione al ricorrere della quale la restrizione si applica (cioè, la parte della valorizzazione del tag che si trova dopo il simbolo `@`). Nel caso in cui nella valorizzazione del medesimo tag, siano riportate più valorizzazioni ciascuna con associata la condizione al ricorrere della quale la valorizzazione si applica, si propone di istanziare una `km4c:AccessRestriction` per ciascuna delle diverse valorizzazioni che il tag può assumere, valorizzando `km4c:condition` con la condizione al ricorrere della quale la specifica valorizzazione si applica.

Si propone di assegnare a ciascuna istanza di `km4c:AccessRestriction` anche il tipo `km4c:Restriction`, di nuova introduzione, generalizzazione di `km4c:AccessRestriction`.

Rappresentazione delle restrizioni di altezza, peso, misure in genere

Si propone di istanziare una nuova `km4c:MaxMinRestriction` (classe di nuova introduzione) per ciascun tag di OSM la cui chiave comprenda uno dei valori seguenti: `maxweight`, `maxaxleload`, `maxheight`, `maxwidth`, `maxlength`, `maxdraught`, `maxspeed`, `minspeed`, `maxstay`.

Si propone di istanziare per ciascuna `km4c:MinMaxRestriction` le seguenti proprietà:

- `km4c:where`, da valorizzare con la URI dell'elemento KM4C cui la restrizione si applica;
- `km4c:what`, da valorizzare con uno dei seguenti valori: `maxweight`, `maxaxleload`, `maxheight`, `maxwidth`, `maxlength`, `maxdraught`, `maxspeed`, `minspeed`, `maxstay`. La scelta del valore da utilizzare dipende da quale tra i valori indicati sia rintracciabile all'interno della chiave del tag che rappresenta la restrizione in OSM;
- `km4c:limit`, da valorizzare con la valorizzazione del tag che rappresenta la restrizione in

OSM;

- `km4c:condition`, da valorizzare con l'eventuale condizione specificata nella valorizzazione del tag che rappresenta la restrizione in OSM. Se la restrizione non è soggetta a condizioni (quindi, se la chiave del tag che rappresenta la restrizione in OSM non comprende l'indicazione `:conditional`), si propone di non istanziare la proprietà;
- `km4c:direction`, da valorizzare con la direzione di traffico cui si applica la restrizione, se ricorre la circostanza (cioè se nella chiave del tag che rappresenta la restrizione in OSM è possibile individuare l'indicazione `forward` piuttosto che `backward`). Se la restrizione si applica ad entrambe le direzioni, si propone di non istanziare la proprietà.

Si propone di assegnare a ciascuna istanza di `km4c:MaxMinRestriction` una URI formata dalla valorizzazione della proprietà `km4c:where`, seguita dalla stringa `/restriction/maxmin/` seguita dalla valorizzazione della proprietà `km4c:what`, eventualmente seguita dalla valorizzazione della proprietà `km4c:direction`, eventualmente seguita dalla valorizzazione della proprietà `km4c:condition` al netto dei caratteri speciali. Ciascuna delle parti componenti la URI deve essere separata dalle altre per tramite di una barra. Nel caso in cui la restrizione sia riferita ad una particolare corsia (lane), alla URI come sin qui proposta dovrebbe essere giustapposta la stringa `/lanes/` seguita dal numero d'ordine della corsia, procedendo da sinistra verso destra rispetto a chi guida.

Si propone di assegnare a ciascuna istanza di `km4c:MaxMinRestriction` anche il tipo `km4c:Restriction`, di nuova introduzione, generalizzazione di `km4c:MaxMinRestriction`.

Rappresentazione delle corsie

In OSM, è possibile indicare il numero di corsie di una strada o di un elemento stradale aggiungendo un tag `lanes`. Nella chiave del tag `lanes` possono coesistere chiavi di direzione nel caso di strade o elementi stradali a doppio senso di circolazione, ma anche chiavi di mezzi di trasporto, per indicare quante corsie sono riservate a quali mezzi di trasporto, ed anche chiavi che indicano restrizioni di utilizzo della corsia. Quando nella chiave del tag compare soltanto la chiave `lanes` eventualmente affincata dalla direzione e dal mezzo di trasporto e nulla più, la valorizzazione del tag deve essere interpretata come il numero delle corsie presenti.

Invece, quando nella chiave del tag `lanes` compaiono anche altre chiavi, ad esempio relative a restrizioni, o la chiave `turns` che serve per specificare qual'è la direzione che dovrà poi prendere chi si immetta su ciascuna delle diverse corsie, allora si sta andando a caratterizzare la corsia, non si sta più facendo una conta. In particolare, i tag con siffatte chiavi, hanno delle valorizzazioni che comprendono, separate da caratteri di pipe, la valorizzazione del tag per ciascuna delle corsie. L'ordine è importante: il primo valore si riferisce alla prima corsia partendo dalla sinistra di chi guida. Bisogna fare attenzione: se la chiave comprende un'indicazione di direzione, significa che le corsie che sto andando a caratterizzare sono soltanto quelle che vanno in quella direzione, e quindi il primo valore sulla sinistra si riferirà non alla prima corsia alla sinistra di chi guida in assoluto, bensì alla prima corsia alla sinistra di chi guida, e che sia destinata a procedere nella direzione in cui sta procedendo chi guida.

Le informazioni relative alle corsie, che si propone di importare nella KB di KM4C, sono:

- la conta delle corsie, separatamente per ciascun mezzo di trasporto e direzione laddove specificati;
- le restrizioni eventualmente presenti su ciascuna corsia (di svolta, di accesso, di misure);
- le indicazioni riguardo l'utilizzo che deve essere fatto della corsia (se si tratti cioè di una

corsia riservata a chi debba svoltare a destra piuttosto che a sinistra piuttosto che procedere dritto).

Per memorizzare tali informazioni, si propone di introdurre una nuova classe `km4c:Lanes`, prevedendo per le istanze di tale classe la possibilità di istanziare le seguenti proprietà:

- `km4c:where`, da valorizzare con la URI dell'elemento KM4C in cui si trovano le corsie;
- `km4c:direction`, da valorizzare eventualmente con la stringa `forward` piuttosto che `backward` a seconda che il set di corsie che l'istanza rappresenta sia quello che procede nella direzione positiva piuttosto che negativa (la direzione positiva è quella coerente con l'ordine in cui sono stati inseriti i nodi di giunzione della strada o dell'elemento stradale in OSM);
- `km4c:lanesCount`, da valorizzare con un'istanza di una classe di nuova introduzione, la `km4c:LanesCount`, per la quale si propone di istanziare una proprietà `km4c:undesignated` per indicare la conta delle corsie che non siano state marcate come riservate a particolari tipologie di utenza, ed inoltre proprietà km4c che prendano il nome dal mezzo di trasporto, da valorizzare con la conta delle corsie riservate allo specifico mezzo di trasporto, se disponibile;
- `km4c:lanesDetails`, da valorizzare nel solo caso in cui vi siano corsie per le quali è specificato un vincolo di utilizzo (corsia per svoltare a destra piuttosto che per procedere dritti) oppure una restrizione specifica per la corsia (su questa corsia non si può transitare oppure la corsia è riservata ad una determinata tipologia di utenza, o ancora il limite minimo di velocità su questa corsia è di tot km orari). Nel caso in cui sia valorizzata, questa proprietà è valorizzata con la URI di un `rdf:Seq`, una collezione ordinata di istanze della classe `km4c:Lane`, che rappresentano una singola corsia, e contengono (come vedremo nel seguito) proprietà idonee a descrivere particolari vincoli attivi sulla corsia rappresentata. Si propone di assegnare a questo genere di contenitori, una URI composta dalla URI dell'istanza di `km4c:Lanes` che referenzia il contenitore, cui si giustappone la stringa `/details`.

Su ciascuna istanza della classe `km4c:Lane` si ha:

- una proprietà `km4c:where` valorizzata con la URI dell'istanza della classe `km4c:Lanes` che rappresenta il gruppo di corsie a cui la corsia che stiamo rappresentando con la classe `km4c:Lane` appartiene;
- una proprietà `km4c:position`, valorizzata con un numero intero che indica la posizione della corsia all'interno del gruppo di corsie a cui la stessa appartiene, iniziando a contare dalla sinistra di chi guida;
- opzionalmente, una proprietà `km4c:turn`. La proprietà è valorizzata nel solo caso in cui sulla corsia rappresentata sia specificato un vincolo di destinazione, del tipo “corsia riservata a chi debba svoltare a destra”, che in OSM è rappresentato attraverso l'apposizione di tag con chiave `turns`. Quando è valorizzata, è valorizzata con una stringa che è esattamente la valorizzazione del tag `turns` di OSM relativa alla corsia che si sta rappresentando;
- opzionalmente, una proprietà `km4c:restrictions`. La proprietà è valorizzata soltanto nel caso in cui sulla corsia rappresentata sia specificata una o più restrizioni di accesso o di max/min. In questo caso, la proprietà è valorizzata con la URI di un `rdf:Bag`, contenitore non ordinato di istanze della classe `km4c:Restrictions` che rappresentano le diverse restrizioni che sono attive sulla specifica corsia. Si propone di assegnare a questo genere di contenitori una URI composta dalla URI dell'istanza della classe `km4c:Lane` che referenzia il contenitore, con giustapposta la stringa `/restrictions`.

Si propone di assegnare a ciascuna istanza della classe `km4c:Lane`, una URI composta a partire

dalla URI dell'istanza della classe `km4c:Lanes` a cui la corsia appartiene, giustapponendo una barra, e un numero intero che indica la posizione relativa della corsia all'interno del proprio gruppo di corsie, procedendo dalla sinistra di chi guida verso destra.

Sugli elementi KM4C che contengono corsie, si propone di introdurre un nuova proprietà `km4c:lanes` da valorizzare con un'istanza di `km4c:Lanes`. Nel caso di elementi che contengono corsie che procedono in due distinti sensi di marcia, si avranno due istanze della proprietà `km4c:lanes`, delle quali una punterà ad un'istanza di `km4c:Lanes` che rappresenta il set di corsie destinate all'utenza che procede in direzione positiva, e l'altra punterà ad un'istanza di `km4c:Lanes` che rappresenta il set di corsie destinate all'utenza che procede in direzione negativa.

Si propone di assegnare alle istanze di `km4c:Lanes` una URI formata dalla concatenazione della URI dell'elemento KM4C in cui le corsie si trovano, con la stringa `/lanes`, e con una stringa che indichi la direzione del set di lanes (`alldirections` se la direzione non è specificata sul tag OSM), con le diverse parti della URI separate sempre da una barra.

Si propone di assegnare alle istanze di `km4c:LanesCount`, delle URI formate dalla URI della `km4c:Lanes` cui appartengono, concatenata con la stringa `/count`.

La mappatura tra le informazioni contenute in OSM e la rappresentazione delle corsie in KM4C così come appena delineata, è facilmente desumibile da quanto sin qui detto, e la si lascia desumere al lettore.

Rappresentazione delle AdministrativeRoad

Ricorre talvolta il caso, guardando su Open Street Map, di strade “lunghe” (tipicamente sovracomunali), rappresentate cioè attraverso relations di OSM, i cui membri sono delle OSM way che hanno anch’esse un proprio nome, diverso da quello della relation. Con ciò si modella in OSM il caso delle strade sovracomunali che, attraversando ad esempio un centro abitato, assumono per un certo tratto un nome diverso. Tra l’altro, è ricorrente che la strada assuma più nomi diversi durante il suo attraversare anche di una stessa municipalità, ed a maggior ragione, è ricorrente che la strada assuma nomi diversi in comuni diversi. Ci sono poi dei tratti in cui nessun nome è assegnato alla strada in aggiunta a quello sovracomunale. Anche qualora la strada abbia un nome “locale”, va tuttavia ricordato che essa non cessa di essere anche “sovracomunale”. Le due casistiche non sono cioè l’una alternativa all’altra. Esse invece coesistono, così come coesistono le due denominazioni, che devono essere entrambe conservate.

L’ontologia di Km4City è predisposta sin dalle origini per modellare questo tipo di situazioni. Infatti, esiste in essa il concetto di strada amministrativa, AdministrativeRoad, ciascuna delle cui istanze rappresenta proprio una porzione delimitata (dotata di nome) di una strada sovracomunale. La strada sovracomunale è invece rappresentata come istanza di km4c:Road.

Per ciascuna AdministrativeRoad è previsto dall’ontologia di Km4City, che si conservi:

- un codice identificativo, che riempie la proprietà dct:identifier, e che viene poi utilizzato anche per comporre la URI;
- il nome della porzione di strada, che riempie la proprietà km4c:adRoadName;
- il nome della strada sovracomunale cui la AdministrativeRoad appartiene, che riempie la proprietà dct:alternative;
- la categoria amministrativa della strada sovracomunale (statale, regionale, provinciale, etc.), che riempie la proprietà km4c:adminClass e che altro non è che il km4c:roadType della km4c:Road sovracomunale;
- l’ente proprietario della AdministrativeRoad, che riempie la proprietà km4c:ownerAuthority;
- gli elementi stradali che compongono la AdministrativeRoad. Per quanto detto, anche la strada

sovracomunale, rappresentata come istanza di km4c:Road, avrà, tra gli altri, anche quegli elementi stradali come componenti. Il legame è stabilito attraverso due proprietà inverse: la km4c:hasRoadElement, e la km4c:formingAdminRoad.

Una volta recuperate, nel processo di ingestion, ed in particolare nello script SQL, tutte le informazioni di interesse relative alle diverse tipologie di strade e agli elementi stradali che le compongono, e una volta che tali informazioni siano state collocate in ordine all'interno di apposite tabelle in vista di una successiva triplificazione, è pressoché triviale estrarre da tali tabelle l'informazione relativa alle AdministrativeRoad. Infatti, si parte dalla tabella RoadRelationIdentifier che contiene gli identificativi delle strade sovracomunali, da essa si aggancia la RoadRelationType che contiene la categoria amministrativa delle strade sovracomunali, da essa si aggancia la RoadRelationExtendName che contiene le denominazioni complete delle strade sovracomunali, dopodiché si va sulla tabella relation_members di OSM per andare a prendere le way che compongono la strada sovracomunale di interesse, dopodiché si va sulla tabella OSM way_tags per andare a recuperare in join stretta il nome della way componente (l'istanziazione di una AdministrativeRoad ha infatti senso nel solo caso in cui il frammento di strada abbia una propria denominazione che si aggiunge a quella della strada sovracomunale), dopodiché si va sulla extra_toponym_city per esplodere sugli elementi stradali che compongono la AdministrativeRoad (i segmenti lineari nodo nodo), dopodiché dall'elemento stradale si va a recuperare l'autorità competente sul frammento di strada grazie alla tabella RoadElementManagingAuthority.

In aggiunta a quanto sin qui detto, sono state introdotte due nuove proprietà sulle AdministrativeRoad istanziate a partire dai dati presenti su Open Street Map, che non erano presenti nell'ontologia, e che non sono state introdotte, ritenendo le proprietà esistenti già sufficienti a caratterizzare completamente il concetto. Le due proprietà sono la adRoadNameGeneric e la adRoadNameSpecific, che corrispondono semanticamente alle proprietà roadType e roadName delle km4c:Road. Anche il metodo di creazione ricalca quello delle Road: una volta noto il nome completo (km4c:adRoadName) della AdministrativeRoad, si determina il tipo di strada cercando tra i nomi di strada generici (v. configurazione iniziale del database PostgreSQL) quale sia il più lungo che fa scopa con la parte iniziale dell'adRoadName. Quello è l'adRoadNameGeneric. Ciò che resta dell'adRoadName una volta eliminato l'adRoadNameGeneric, è l'adRoadNameSpecific.

Ipotesi di implementazione della mappatura

Si delineano in questa sezione alcune ipotesi implementative riguardo gli strumenti e le modalità di utilizzo degli stessi per la produzione di triple RDF coerenti con la Smart City Ontology, a partire dai documenti OSM[53].

L'approccio Linked Geo Data

Il problema di come poter implementare una mappatura tra i documenti OSM[54] e corrispondenti insiemi di triple RDF non è nuovo in letteratura. E' stato infatti precedentemente affrontato ad esempio nell'ambito del progetto Linked Geo Data[55].
In tale lavoro, si muove dal presupposto che sia possibile mappare ciascuna proprietà di ciascuna entità di OSM su di una tripla RDF, concentrandosi soltanto sulla specifica proprietà, senza prendere in esame le altre proprietà dello stesso oggetto OSM.
Questo requisito, che risultava verificato nell'ambito di Linked Geo Data dal momento che in quel lavoro ciò che si ricercava era una traduzione pressoché letterale dei documenti OSM in insiemi di triple RDF, non risulta invece soddisfatto nel nostro caso, il che costituisce probabilmente il principale impedimento all'adozione dell'approccio Linked Geo Data per l'implementazione della mappatura tra OSM e Km4City.

[53]Documenti XML che rappresentano le mappe in Open Street Map
[54]Documenti XML che rappresentano le mappe in Open Street Map
[55]http://linkedgeodata.org/

Inoltre, nel progetto Linked Geo Data l'attenzione è esclusivamente rivolta agli elementi `way`, `node` e `tag`, che sono soltanto una delle possibili tipologie di elementi che si possono individuare all'interno di un'entità OSM. Ad esempio, rivestono una grande importanza all'interno di un documento OSM gli elementi `relation`, che descrivono strade composte da più elementi stradali, stabiliscono corrispondenze tra toponimi e numeri civici, ed altro ancora.

Ciò è già sufficiente per convincersi che, per le differenti finalità, la mappatura che viene eseguita nell'ambito del progetto Linked Geo Data si discosta significativamente dalla mappatura che deve essere eseguita per le finalità di Km4City.

Tuttavia, lo strumento che viene utilizzato nell'ambito di Linked Geo Data per implementare la scrittura delle triple RDF a partire dai dati conservati sul database relazionale, Sparqlify[56], è invece di grande interesse anche per i nostri scopi.

Sparqlify: introduzione

Sparqlify è un applicativo server, i cui dati sono conservati in un database relazionale, ma vengono esposti all'esterno sotto forma di triple RDF (è quindi interrogabile con SPARQL), sulla base di uno o più documenti di configurazione che vengono forniti come parametri al momento dell'avvio.

E' tuttavia anche possibile, lanciando l'applicativo con parametri opportuni, far sì che non venga avviato il server, e venga invece soltanto prodotto in output (`stdout`) un insieme di triple che altro non sono che le righe del database relazionale trasformate in base al contenuto del documento di configurazione. Si tratta a tutti gli effetti di un dump del database, e infatti l'opzione che istruisce l'applicativo a lavorare in questa modalità è `-D`.

Sparqlify: installazione

Si riepilogano nel seguito i passi necessari per l'installazione, su di una macchina Ubuntu 14.04, di tutti gli applicativi necessari, e quindi in particolare:

- `osmosis`, che legge le mappe di Open Street Map, e le scrive su di un database relazionale (auspicabilmente Postgres) opportunamente predisposto;
- PostgreSQL: l'RDBMS su cui sono materialmente conservati i dati, che è per fortunata coincidenza la scelta raccomandata da Open Street Map, e uno dei soli due motori supportati da Sparqlify (l'altro è Apache Hive).
- Sparqlify, che legge dal database relazionale, ma si presenta all'esterno come un motore di database RDF, interrogabile quindi con SPARQL, sulla base di un'opportuna configurazione che deve essere fornita e che consente di mappare i dati del database relazionale sulle triple RDF.

L'installazione di `osmosis` è triviale, e può essere eseguita rapidamente seguendo le indicazioni fornite sul Wiki[57] di Open Street Map.

L'installazione di PostgreSQL è anch'essa piuttosto semplice. Si può fare riferimento anche in questo caso al Wiki[58] di Open Street Map, nella parte in cui viene descritto il setup di `osmosis` per la scrittura su database PostgreSQL con estensione PostGIS. A seconda della versione di Ubuntu in uso, i package che viene data indicazione di installare nella pagina del Wiki, potrebbero non essere disponibili. In quel caso si dovrà cercare tra i package disponibili, i più simili (è sufficiente fare riferimento alla denominazione del package) a quelli indicati, ed installarli in luogo di quelli indicati. Ad esempio, sul Wiki si indica di installare `postgresql-9.4-postgis-2.1`, che non è però

[56] http://aksw.org/Projects/Sparqlify.html
[57] http://wiki.openstreetmap.org/wiki/Osmosis/Installation#Linux
[58] http://wiki.openstreetmap.org/wiki/Osmosis/PostGIS_Setup

disponibile su Ubuntu 14.04, mentre è disponibile `postgresql-9.3-postgis-2.1`.

L'installazione di Sparqlify è complessa. Al momento in cui si scrive, nessuna delle distribuzioni pronte all'uso cui viene fatto riferimento sulla documentazione PDF[59] di Sparqlify o sul file README[60] del repository GitHub è realmente disponibile, e neppure la compilazione manuale giunge a termine perfettamente, in nessuna delle due modalità indicate negli stessi documenti. La strada che si è individuata per l'installazione di Sparqlify è stata infine questa:

- scaricamento del repository da GitHub[61];
- modifica del sorgente, ed in particolare aggiunta all'interno del file `sparqlify-cli/src/main/java/org/aksw/sparqlify/web/Main.java` attorno alla riga 200, della parte evidenziata in grassetto qui sotto:
```
DatabaseMetaData dbMeta = conn.getMetaData();
String dbProductName = dbMeta.getDatabaseProductName();
if(dbProductName.equals("PostgreSQL")) {
 dbProductName = "Postgresql";
}
```
- installazione di Java 8, e impostazione della JAVA_HOME sulla home di Java 8;
- compilazione “Assembly based” indicata nel file README (all'ultimo passo, fallisce, ma esegue comunque quanto basta per essere utile ai nostri scopi);
- creazione dello script `sparqlify.sh` come suggerito sulla documentazione PDF (par. 1.1.1 Building Sparqlify from Source), ma utilizzando come classpath l'archivio `sparqlify-cli-0.8.2-jar-with-dependencies.jar` che la precedente compilazione ha prodotto nella directory `sparqlify-cli/target`. Questo script, è il nostro eseguibile.

Sparqlify: utilizzo

Per il caricamento dei dati sul database relazionale via `osmosis`, si può fare riferimento alla sezione PostGIS Tasks (Simple Schema)[62] del Wiki per quanto riguarda lo schema del database, e al Wiki di Open Street Map[63] per la parte di creazione del database, e per il comando (da adattare in base alle specifiche esigenze) che esegue il caricamento dei dati sul database relazionale.

Per l'utilizzo di Sparqlify dopo il caricamento dei dati sul database relazionale, si può fare riferimento alla documentazione PDF[64] di Sparqlify, integrata con le istruzioni testuali che vengono presentate a video quando si lancia Sparqlify senza alcun parametro (i nomi dei parametri indicati sulla documentazione PDF, non sono infatti corretti). Nel lanciare Sparqlify, è necessario ricordare che è necessario prima portarsi nella directory di installazione di Sparqlify, e poi lanciare il comando, ed è necessario inoltre ricordare che la variabile d'ambiente JAVA_HOME deve essere stata preventivamente impostata sulla folder principale dell'installazione di Java 8, che deve essere presente sulla macchina.

Per quanto detto, il problema dell'aggiornamento dei dati pare poter essere ricondotto al problema di aggiornare i dati sul database relazionale da cui Sparqlify attinge. A questo proposito, va rilevato che su geofabrik[65], per ogni *subregion*, viene reso quotidianamente disponibile un file `osc`, Open Street

[59] https://github.com/downloads/AKSW/Sparqlify/Sparqlify-UserManual-And-TechnicalDocumentation.pdf
[60] https://github.com/AKSW/Sparqlify/blob/master/README.md
[61] `git clone git://github.com/AKSW/Sparqlify.git`
[62] http://wiki.openstreetmap.org/wiki/Osmosis/Detailed_Usage_0.45#PostGIS_Tasks_.28Simple_Schema.29
[63] http://wiki.openstreetmap.org/wiki/Osmosis/PostGIS_Setup
[64] https://github.com/downloads/AKSW/Sparqlify/Sparqlify-UserManual-And-TechnicalDocumentation.pdf
[65] http://download.geofabrik.de/

Change, e che `osmosis`, eseguito con un opportuno set di parametri[66], è in grado di eseguire puntualmente sul database relazionale le modifiche indicate nel file `osc`.

In realtà, la situazione è leggermente complicata dal fatto che per rendere le query inserite nel file di configurazione di Sparqlify le più semplici possibile, è necessario predisporre delle tabelle sul database relazionale che contengano tutti e soli i dati necessari a Sparqlify per la produzione delle triple. Si rende cioè necessario un passaggio intermedio: dopo l'aggiornamento del dati sul database relazionale eseguito utilizzando `osmosis`, è necessario eseguire un opportuno script SQL sul database relazionale che prepara i dati che saranno poi letti attraverso semplici query contenute nel file di configurazione di Sparqlify.

Generate poi le triple, si pone il problema di caricare le modifiche su Virtuoso[67].

Virtuoso prevede una funzionalità[68] ad-hoc, che accetta in ingresso un intero grafo, valuta le differenze tra il grafo inviato in ingresso ed il grafo correntemente memorizzato, ed esegue azioni mirate di inserimento, modifica e cancellazione. Questa funzionalità è tuttavia soggetta ad un insieme di restrizioni che devono essere approfondite. Nel caso in cui si determini che la stessa può essere effettivamente utilizzata nel nostro specifico caso, la proposta è quella di generare ogni giorno l'intero grafo delle triple RDF aggiornato, utilizzando Sparqlify, e inviarlo in ingresso a questa funzionalità di Virtuoso.

Nel caso in cui invece tale funzionalità non fosse utilizzabile, allora la proposta sarebbe quella di tentare di emularla, estraendo ogni giorno tramite Sparqlify le sole triple che devono essere inserite o modificate (attraverso una configurazione opportuna che includa dei tagli temporali), ed andando poi a ricercare le triple che devono essere eliminate, leggendo direttamente dal file `osc`, che è un documento in formato XML, per poi eliminarle con un job di ETL o con un software ad-hoc.

Sparqlify: aspetti positivi e criticità

L'aspetto positivo dell'utilizzo di Sparqlify è la sua estrema flessibilità, oltre alla possibilità di utilizzarlo non soltanto come generatore di triple RDF da importare in un diverso database RDF, ma anche direttamente come database RDF (supporta però soltanto SPARQL 1.0).

Per estrema flessibilità si intende in particolare la possibilità, attraverso opportuna configurazione, di eseguire qualsiasi genere di filtraggio e trasformazione dei dati contenuti nel database relazionale, ai fini della generazione delle triple RDF corrispondenti.

La principale criticità è data dal fatto che l'utilizzo di Sparqlify per la produzione delle triple RDF, invece che di un ETL o di un software sviluppato ad-hoc, implica la necessità di avere disponibili sul database relazionale tutti i dati necessari per la produzione delle triple. Non è infatti possibile, durante l'esecuzione di Sparqlify, eseguire invocazioni ad applicativi esterni per recuperare informazioni mancanti.

Questo ha portato a valutare la possibilità di utilizzare uno strumento di geo-localizzazione esterno `nominatim`[69], per recuperare tutta l'informazione necessaria per la produzione delle triple ma non codificata all'interno delle mappe OSM (ad esempio le relazioni di appartenenza tra comuni e province).

In fasi successive di analisi, si è in realtà verificato che caricando le mappe su di un database relazionale con estensione PostGIS, tutta l'informazione necessaria a stabilire questo genere di relazioni e tutte le altre necessarie, era disponibile sul database relazionale, che doveva soltanto essere opportunamente interrogato, facendo ricorso alle funzioni SQL aggiuntive che proprio l'estensione

[66] `osmosis --read-xml-change --write-pgsimp-change`
[67] Il database RDF correntemente in uso nel progetto Kn4City
[68] https://www.mail-archive.com/virtuoso-users@lists.sourceforge.net/msg07075.html
https://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtRDFBulkLoaderWithDelete
[69] http://nominatim.openstreetmap.org/

PostGIS mette a disposizione.

L'implementazione attuale, che utilizza appunto questo genere di approccio, sembra dimostrare l'effettiva possibilità di recuperare tutta l'informazione di interesse dal database relazionale popolato da `osmosis`, senza dover ricorrere ad alcun applicativo esterno.

Una seconda criticità di Sparqlify è il tentativo che lo strumento compie di ottimizzare le query che vengono inserite all'interno del file di configurazione, con notevole dispendio di tempo e perdita di controllo. Per questo motivo, nella revisione corrente il carico è stato spostato più verso il database, rendendo le query riportate all'interno del file di configurazione di Sparqlify le più semplici possibile.

Integrazione dei dati della Regione Toscana

Il grafo stradale dell'attuale KB[70] è popolato a partire dai dati rilasciati dalla Regione Toscana.

Ciò che si sta invece ipotizzando, è di utilizzare una diversa fonte, le mappe OSM, per il popolamento dello stesso grafo.

Ciascuna delle due sorgenti dati è detentrice esclusiva di talune informazioni.

Si pone il problema di come poter trarre il massimo beneficio da tale “abbondanza di dati”.

A questo proposito, si propone di utilizzare Open Street Map come fonte dati primaria, e di predisporre delle tabelle aggiuntive sul database relazionale, da popolare con ETL/applicativi ad-hoc, per ospitare i dati provenienti dalla Regione Toscana che non siano disponibili sulle mappe di Open Street Map.

Nel configurare Sparqlify, si andranno poi ad interrogare opportunamente le tabelle popolate tramite importazione da OSM e quelle diversamente popolate con i dati provenienti in via esclusiva dalla Regione Toscana, di modo da produrre triple RDF adeguate rispetto alle esigenze delle applicazioni.

Linee guida per l'installazione su di una macchina vergine

E' necessario innanzi tutto installare osmosis, Postgresql e Sparqlify secondo le indicazioni fornite nel capitolo dedicato alla Sparqlify: installazione.

Dopodiché, è necessario procedere alla prima importazione dei dati OSM all'interno di un database relazionale, che può essere eseguita utilizzando `osmosis` come descritto nel capitolo Sparqlify: utilizzo. Va rimarcato che al momento del primo caricamento dei dati, è di gran lunga conveniente importare una mappa che copra tutta l'area di interesse per l'installazione, eventualmente eccedendo, piuttosto che tentare successivamente di aggiungere nuove mappe del tutto distinte da quella caricata nella prima installazione. Infatti, mentre applicare gli aggiornamenti rilasciati da (ad esempio) geofabrik[71] su di una mappa tagliata dallo stesso geofabrick è un'operazione triviale, lo è molto meno tentare di aggiungere ad esempio la mappa del centro Italia ad un'installazione su cui era stata originariamente caricata la mappa delle sole isole.

Una volta eseguita la prima importazione dei dati tramite `osmosis`, è necessario predisporre e popolare le tabelle da cui Sparqlify andrà a leggere per generare le triple RDF. Si tratta di un processo in due fasi:

1. creazione e popolamento delle tabelle relazionali di appoggio, che devono essere create sullo stesso database relazionale su cui è stata caricata la mappa OSM via osmosis. Lo script SQL che crea le tabelle e le popola opportunamente è disponibile sul repository[72];
2. generazione delle triple RDF attraverso l'esecuzione di Sparqlify istruito attraverso un

[70] La Knowledge Base di Km4City
[71] https://www.geofabrik.de/
[72] `osm_ingestion/sparqlify`, sotto-cartella `install` se si deve triplificare l'intero grafo stradale, sotto-cartella `update` se si deve triplificare soltanto una specifica tipologia di oggetti (classe) e non l'intero grafo

opportuno file di configurazione, con estesione `sml`, anch'esso pubblicato sul repository. A ciascuno script SQL è affiancato il documento `sml` corrispondente.

A proposito degli script SQL, è probabilmente il caso di evidenziare quanto segue:

- la delimitazione spaziale dell'installazione, delimita il perimetro di interesse per la generazione delle triple RDF, indipendentemente dalla mole di dati che `osmosis` ha originariamente caricato sul database relazionale. Ad esempio, si possono caricare con `osmosis` i dati relativi all'intera Italia, e poi estrarre le triple RDF soltanto per la Città Metropolitana di Cagliari;
- la delimitazione spaziale dell'installazione si basa sul fatto che tutti gli enti territoriali sono rappresentati all'interno di OSM attraverso delle `relation` che comprendono come membri un insieme di `way` che tracciano il perimetro dell'ente territoriale, e che sono taggate (le `relation`) opportunamente come descritto nei capitoli Mappatura delle province e Mappatura dei comuni (per le regioni, resta tutto invariato ad eccezione del tag `admin_level` che questi enti è valorizzato a 4). Infatti, ciò che si deve specificare all'atto della configurazione è l'ID OSM della o delle relation di interesse, scelte tra quelle che rappresentano enti territoriali. In verità, si potrebbero anche indicare relation arbitrarie: ciò che lo script SQL fa è di andare ad individuare il perimetro della `relation` attraverso i membri di tipo `way`, ed utilizzare quel perimetro come delimitazione spaziale dell'installazione;
- dal momento che l'intero flusso si conclude con l'esportazione delle triple o quadruple su di un file di testo che deve essere successivamente e separatamente caricato su di un database RDF, e quindi tutto ciò che non è esplicitamente rappresentato sul file di testo va perso, è forse il caso di rimarcare che il nome del grafo sotto cui vengono collocate tutte le triple RDF generate nell'ambito di una stessa esecuzione, che può essere specificato personalizzando opportunamente lo script SQL di configurazione iniziale, è rilevante nel solo caso in cui Sparqlify venga lanciato nella modalità che genera quadruple [73] (esplicitando quindi il contesto). Nel caso della generazione delle triple infatti, il contesto non viene scritto sul file di testo destinato ad essere poi caricato dul database RDF, e quindi va perso.

Dopo ciascun aggiornamento dei dati di OSM (caricamento di files `osc`, si veda in proposito anche il capitolo Sparqlify: utilizzo), sarà necessario ripetere le due fasi indicate di preparazione delle tabelle relazionali di appoggio, e di triplificazione, per ottenere le triple RDF aggiornate.

Linee guida per l'aggiornamento di una parte del grafo stradale

Nel caso in cui si desideri aggiornare soltanto una specifica proprietà di una specifica classe del grafo stradale, o nel caso in cui si debba integrare il grafo stradale con una specifica proprietà di una specifica classe di nuova introduzione, non è necessario procedere all'importazione dell'intero grafo stradale come descritto nel paragrafo Linee guida per l'installazione su di una macchina vergine.

Si può invece fare riferimento al materiale contenuto nel repository[74] SVN, specificamente pensato per le attività di aggiornamento mirato.

Procedure operative e dislocazione di strumenti e dati

Lo sviluppo delle triple deve avvenire sulla 110, a partire dal database `pgsimple_ita` che si trova sul Postgres della 110. I dati del Postgres della 110, e anche le triple sviluppate sulla 110 a partire da quei dati, si trovano su `/disk1` della 110.

[73] Aggiungendo l'opzione `--format nquads` al momento del lancio dello `sparqlify.sh`

[74] Cartella `osm_ingestion`, in particolare il file `sparqlify/readme.txt` e la cartella `sparqlify/update`.

Ci sono degli script belli che pronti in `/disk1` della 110 per generare senza fatica le triple a partire dai dati Postgres: sono alla posizione `/disk1/triple_osm/Toscana`, all'interno delle diverse cartelle delle province, ciascuno personalizzato per la provincia di interesse. Servono chiaramente da modello: possono essere utilizzati per la triplificazione di qualsiasi porzione di territorio previo opportuno adattamento.

Una volta prodotte le triple, per caricarle in test, bisogna prima di tutto spostarle su /media/triples/OSM che è un percorso sempre montato sulla 110, che però va a pescare da un'altra parte (che poi è la stessa cosa che succede per `/disk1`).

Poi bisogna andare sulla 208, e lì succede che il percorso che se lo faccio dalla 110 è `/media/triples/`, mi diventa `/media/TriplesTest/`, perché `/media` è montata su entrambe le macchine ma va a pescare in due posti diversi, questo per inciso.

Invece quello che ci interessa di più è che per caricare le triple sul Virtuoso e renderle disponibili ai client, bisogna, dalla 208, eseguire lo script `update-osm.sh` che si trova sulla home, e che guarda caso va a pescare in `/media/TriplesTest/OSM/ecceteraeccetera`, dopo aver avuto cura di personalizzarlo scrivendoci qual è la cartella/le cartelle da cui si vuole leggere, tra quelle che si trovano in OSM. Attenzione che lui prende tutti i file che si trovano in quelle cartelle e tenta di caricarli.

Sulla base di tutto questo, per fare il backup di un grafo stradale, bisogna andare sulla 208, aprire l'interactive sql (da riga di comando isql, oppure da interfaccia web) e lanciare una stored procedure opportuna con parametri opportuni, che è questa qui:

```
dump_one_graph('http://www.disit.org/km4city/resource/OSM/firenze'
, '/media/TriplesTest/OSM/backup/firenze/backup_firenze_',
50000000);
```

anche questa chiaramente da personalizzare a seconda di che cosa esattamente si debba backuppare.

Questo grazie al fatto che una volta nella vita la ho creata questa stored procedure, usando il codice fornito da Virtuoso nella sua stessa documentazione, a questa pagina:

`http://docs.openlinksw.com/virtuoso/rdfperfdumpandreloadgraphs/`

dove c'è anche scritto qual è la procedura che deve essere creata e come deve essere utilizzata per fare il restore di quegli stessi dati. Notare che la procedura di recupero si aspetta di avere dei file ttl, non sono sicuro che gli vadano bene anche i ttl gz, sarebbe da verificare, se non gli vanno bene vanno prima decompressi, e poi lanciata la procedura, perché la procedura quella che genera il backup, lo genera compresso.

Riconciliazione del grafo stradale della RT con quello di OSM

In questa sezione è descritta un'ipotesi di riconciliazione tra il grafo stradale della Regione Toscana ed il grafo stradale derivante dall'importazione delle mappe stradali da Open Street Map. In particolare, si presenta un algoritmo per la mappatura automatica delle province, dei comuni, dei toponimi e degli elementi stradali presenti nel grafo stradale della Regione Toscana verso i corrispondenti del grafo stradale di Open Street Map e viceversa, fornendo alcune indicazioni utili all'esecuzione manuale dello stesso algoritmo attraverso il lancio manuale di query SPARQL e l'osservazione dei risultati. Nella parte finale del capitolo, si descrive invece in che modo sia possibile utilizzare un command-line tool sviluppato ad-hoc in laboratorio, in abbinamento con uno script batch di Windows (per il lancio in ambiente Linux l'interessato produrrà uno shell script traendo ispirazione dallo script fornito per l'ambiente Windows), per automatizzare l'esecuzione dell'algoritmo.

Riconciliazione delle province

Il metodo proposto per la riconciliazione delle province è triviale, ed è implementabile attraverso

un'unica query `CONSTRUCT` di SPARQL nella quale si vadano a recuperare le istanze della classe `km4c:Province` separatamente dal grafo RDF che contiene le triple relative al grafo stradale della Regione Toscana e dal grafo RDF che contiene invece le triple generate importando i dati da OSM, e si vada in particolare a recuperare per ciascuna istanza la valorizzazione della proprietà `dct:alternative` che contiene la sigla della provincia. Imponendo l'uguaglianza tra le sigle delle province, si stabilisce la corrispondenza tra le istanze presenti nel grafo della Regione Toscana e le istanze presenti nel grafo generato leggendo da OSM, e si vanno a scrivere tali corrispondenze in un grafo RDF apposito, andando a valorizzare per ciascuna delle due istanze messe in relazione, la proprietà `owl:sameAs`. Una prima versione della query SPARQL che implementa questa mappatura è disponibile sulla macchina Ubuntu di sviluppo[75].

Riconciliazione dei comuni

Il metodo proposto per la riconciliazione delle province è triviale, ed è implementabile attraverso un'unica query `CONSTRUCT` di SPARQL nella quale si vadano a recuperare le istanze della classe `km4c:Municipality` separatamente dal grafo RDF che contiene le triple relative al grafo stradale della Regione Toscana e dal grafo RDF che contiene invece le triple generate importando i dati da OSM, e si vada in particolare a recuperare per ciascuna istanza la valorizzazione della proprietà `dct:alternative` che contiene il codice catastale del comune. Imponendo l'uguaglianza tra i codici catastali, si stabilisce la corrispondenza tra le istanze presenti nel grafo della Regione Toscana e le istanze presenti nel grafo generato leggendo da OSM, e si vanno a scrivere tali corrispondenze in un grafo RDF apposito, andando a valorizzare per ciascuna delle due istanze messe in relazione, la proprietà `owl:sameAs`. Una prima versione della query SPARQL che implementa questa mappatura è disponibile sulla macchina Ubuntu di sviluppo[76].

Riconciliazione dei toponimi: pre-requisiti

Nell'ipotesi corrente, la riconciliazione dei toponimi deve essere eseguita a livello comunale. E' quindi innanzi tutto necessario procedere alla creazione di due grafi, estratti rispettivamente dal grafo stradale complessivo della Regione Toscana e dal grafo stradale complessivo importato da Open Street Map, andando ad eseguire un taglio sia geografico (sul comune), sia a livello di proprietà (non tutte le proprietà delle diverse classi sono necessarie per poter procedere correttamente alla mappatura secondo l'ipotesi corrente, ed anzi la presenza di proprietà non necessarie potrebbe addirittura compromettere il buon esito dell'operazione portando a stime di tempo di esecuzione troppo lunghe o a tempi di esecuzione effettivi troppo lunghi con conseguente timeout della transazione). Una query SPARQL che può essere utilizzata come traccia per la generazione di un grafo che contenga tutta e sola l'informazione necessaria per la riconciliazione dei toponimi nel comune di interesse (e che deve essere eseguita due volte, una volta leggendo dal grafo della Regione Toscana, e una seconda volta leggendo dal grafo importato da OSM), è memorizzata sulla macchina Ubuntu di sviluppo[77].

Riconciliazione dei toponimi: inizializzazione

Dopo che siano stati opportunamente generati i due grafi come descritto al paragrafo precedente, si propone di procedere stabilendo una prima triviale corrispondenza tra i toponimi della Regione Toscana e quelli di OSM: si mettono in corrispondenza i toponimi (proprietà `km4c:extendName` della `km4c:Road`) uguali a meno della capitalization. Si stabilirà così un primo insieme (ridottissimo) di legami, che serviranno da base per lo sviluppo dei legami successivi. La query SPARQL che realizza questo genere di corrispondenze è memorizzata sulla macchina Ubuntu di sviluppo[78].

[75]File province.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)
[76]File municipality.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)
[77]File road_init.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)
[78]File road_seed.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)

Riconciliazione dei toponimi: step iterativo

Una volta stabilite le prime corrispondenze come delineato nel paragrafo precedente, si propone di procedere come segue.

Si supponga di aver instaurato, in fase di inizializzazione, una relazione per via Firenze. Abbiamo quindi individuato un toponimo "Via Firenze" sul sottografo OSM, un toponimo "Via Firenze" sul sottografo RT, e abbiamo memorizzato sul database RDF una relazione bidirezionale tra le due istanze attraverso l'utilizzo della proprietà owl:sameAs.

A questo punto, andiamo ad individuare sul sottografo OSM l'insieme di tutti i toponimi che intersecano Via Firenze, e lo stesso facciamo sul sottografo RT. Avremo così individuato due insiemi distinti di toponimi, tipicamente di cardinalità ridotta, che sono le rappresentazioni OSM e le corrispondenti rappresentazioni RT dei toponimi che intersecano Via Firenze.

Andiamo quindi a stabilire le corrispondenze tra i toponimi presenti in questi due piccoli insiemi che abbiamo estratto. Lo facciamo imponendo un criterio molto stringente: stabiliamo la corrispondenza nel solo caso in cui il toponimo (km4c:extendName) OSM ed il toponimo RT siano tali per cui uno costituisce la parte finale dell'altro. Siccome i toponimi completi iniziano con una denominazione generica, chiedere che uno costituisca la parte finale dell'altro, corrisponde pressoché a chiedere che siano uguali, ma se erano uguali, li avevamo già individuati in fase di inizializzazione, ed infatti imponendo un vincolo così stringente non riusciamo ad accrescere la nostra conoscenza (a generare nuove corrispondenze).

Allora, imponiamo un vincolo meno stringente: vogliamo che uno dei due toponimi costituisca la parte iniziale dell'altro. Potrebbe sembrare un vincolo debole, ma non lo è, perché dobbiamo ricordare che oltre a questo, abbiamo anche ristretto la ricerca tra i soli toponimi che intersecano Via Firenze. Stiamo quindi dicendo che se due toponimi, uno OSM e uno RT, che intersecano entrambi Via Firenze, hanno un km4c:extendName tale per cui uno costituisce la parte iniziale dell'altro, allora i due toponimi sono rappresentazioni diverse della stesso toponimo nel mondo reale. Supponiamo che così facendo, riusciamo a trovare nuove corrispondenze. In questo caso, le scriviamo sul database RDF, e torniamo a testare il vincolo più stringente, il primo, perché avendo instaurato nuove corrispondenze, è possibile che tra i toponimi che intersecano queste nuove corrispondenze, ve ne siano che soddisfano il vincolo più stringente. Se invece non riusciamo a stabilire alcuna corrispondenza, allentiamo ancora le maglie, e diciamo che ci va bene stabilire una corrispondenza nel caso in cui tutte le parole di un toponimo siano presenti anche nell'altro (che va sempre sommata al vincolo che i due toponimi devono intersecare via Firenze). Supponiamo che così facendo riusciamo a stabilire delle nuove corrispondenze. In questo caso, le scriviamo, e ripartiamo con i test applicando il più stringente tra i vincoli, perché può darsi che tra i toponimi che intersecano le nuove corrispondenze, ve ne siano uno o più che soddisfano quel vincolo. Se invece non fossimo ancora riusciti a stabilire nuove corrispondenze, procederemmo ancora oltre allentando ulteriormente il vincolo, ed "accontentandoci" di avere `km4c:roadName` uguale, tralasciando la denominazione generica, e via di seguito, secondo questo approccio, giù per la scala dei vincoli, che è composta da quattordici "gradini". Il processo di riconciliazione si arresta quando, arrivati all'ultimo gradino, non si riesce a stabilire alcuna ulteriore corrispondenza.

Chiaramente, per ciascun livello, si vanno a sfruttare tutte le corrispondenze già stabilite, quindi si cercano corrispondenze tra le intersezioni di via Firenze e, separatamente, tra le intersezioni di tutte i toponimi per i quali abbiamo già stabilito una corrispondenza tra rappresentazione OSM e rappresentazione RT.

La query SPARQL che implementa questi step di riconciliazione è unica[79], e contiene (commentati) i vincoli corrispondenti a tutti i diversi "gradini". A seconda del gradino che dobbiamo testare, si dovrà procedere a decommentare il vincolo opportuno, ed eseguire la query. Le corrispondenze

[79]File road_step.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)

trovate dovranno quindi essere memorizzate sul database RDF.

In realtà, per alcuni "gradini", non si avrà un vincolo direttamente codificato nella query SPARQL, ma un riferimento ad altre query SPARQL che devono essere eseguite.

In tutte le query indicate, si trovano codificati i nomi dei grafi e sottografi[80], che dovranno essere eventualmente personalizzati preventivamente rispetto all'esecuzione delle stesse query.

Riconciliazione degli elementi stradali

Nella proposta corrente, la riconciliazione degli elementi stradali viene eseguita attraverso due query SPARQL.

La prima delle due[81], stabilisce un legame tra ciascuno dei `km4c:RoadElement` del sottografo RT che appartengono a `km4c:Road` riconciliate, e uno dei `km4c:RoadElement` del sottografo OSM appartenenti alla `km4c:Road` corrispondente, sulla base della somma delle distanze tra i due nodi estremi dell'elemento stradale.

La seconda[82], va a recuperare gli elementi stradali del sottografo OSM che sono rimasti privi di una corrispondenza, e associa ciascuno ad un elemento stradale del sottografo RT, sulla base dello stesso criterio.

Esecuzione automatica dell'algoritmo con il command-line tool rt2osm

E' stato prodotto in laboratorio ed è disponibile sul repository[83] uno strumento eseguibile da riga di comando denominato `rt2osm` che, in eventuale opportuno abbinamento con l'omonimo script batch di Windows (o con un corrispondente shell script per gli ambienti Linux) pubblicato sul repository nella stessa posizione, permette di automatizzare l'esecuzione dell'algoritmo di riconciliazione dei grafi stradali così come presentato sin qui in questo stesso capitolo.

In particolare, lo script batch oltre ad essere componente fondamentale dell'automazione, soprattutto per quanto riguarda la riconciliazione dei toponimi, costituisce anche un'utile guida all'utilizzo di `rt2osm` per chi si avvicini per la prima volta allo strumento, mostrando come lo stesso strumento debba essere utilizzato per riconciliare le province, i comuni, i toponimi e gli elementi stradali, ed indicando inoltre quali riconciliazioni sono propedeutiche rispetto ad altre.

Va rimarcato che il file batch è principalmente fornito a scopo esemplificativo. L'interessato, leggendone i commenti ed analizzandone i comandi, lo personalizzerà di volta in volta, escludendo e/o modificando le diverse parti, a seconda della specifica riconciliazione che deve essere eseguita.

L'installazione del software è triviale: il command-line tool (jar) e il file batch dovranno infatti semplicemente essere collocati sulla macchina che ospita l'istanza di Virtuoso sulla quale si deve operare (sulla quale si trovano cioè i grafi da riconciliare), ed opportunamente personalizzati. A seconda dei casi, accorgimenti particolari potrebbero essere necessari per assicurare che Virtuoso abbia visibilità dei file di triple che devono essere su di esso caricati, ed inoltre che lo strumento built-in di Virtuoso denominato `isql` sia visibile dalla posizione in cui viene collocato il file batch, ed inoltre che le librerie esterne necessarie per la corretta esecuzione di `rt2osm` siano visibili dalla posizione in cui viene collocato il jar. Si affida al lettore che si assume formato su questi aspetti, il compito di determinare che cosa esattamente debba essere fatto in ciascun caso di specie al fine di

[80]Grafo completo RT: http://www.disit.org/km4city/resource/GrafoStradale/Grafo_stradale_Grosseto
Grafo completo OSM: urn:km4city:OSM:test:grosseto
Sottografo RT: http://www.disit.org/km4city/resource/GrafoStradale/RT2OSM/RT_Graph/ComuneGrosseto
Sottografo OSM: http://www.disit.org/km4city/resource/GrafoStradale/RT2OSM/OSM_Graph/ComuneGrosseto
Grafo corrispondenze: http://www.disit.org/km4city/resource/GrafoStradale/RT2OSM/Road/ComuneGrosseto

[81]File road_element.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)

[82]File road_element_2.sparql alla posizione ubuntu@hadoopnodet05:~/rt2osm (macchina 192.168.0.110)

[83]osm_ingestion\rt2osm\tool

assicurare quanto indicato.

Riconciliazione dei servizi sul grafo stradale di Open Street Map

Ci si pone il problema di associare un'istanza di `km4c:Road` ed un'istanza di `km4c:Entry` ai servizi di Km4City, con il vincolo che le due istanze appartengano al grafo stradale generato attraverso l'importazione da Open Street Map, invece che al grafo stradale della Regione Toscana.

Approccio proposto e implementazione

L'approccio che si propone di utilizzare è quello di recuperare l'indirizzo human-readable di ciascun servizio da riconciliare, accedendo alle proprietà `schema:addressLocality`, `schema:streetAddress` e `km4c:houseNumber` dell'istanza del servizio memorizzata nell'RDF store, ed inviarlo in ingresso al servizio con interfaccia REST denominato Address/POI search by text[84], ottenendone la URI dell'istanza di `km4c:StreetNumber` che deve essere associata al servizio. Si propone quindi di utilizzare tale URI per recuperare, attraverso opportune query SPARQL sull'RDF store, la URI della `km4c:Entry` e della `km4c:Road` associate al numero civico. Si propone di utilizzare queste due ultime URI per valorizzare rispettivamente la proprietà `km4c:hasAccess` e `km4c:isInRoad` dell'istanza del servizio da riconciliare.

Un'implementazione dell'approccio descritto è disponibile sulla macchina Ubuntu[85] di sviluppo e sul corrispondente repository SVN[86].

Algoritmo di riconciliazione

In maggior dettaglio, la riconciliazione procede attraverso i seguenti passi:

1. INIZIALIZZAZIONE. Si compila la lista degli URI dei servizi che devono essere riconciliati. La lista degli URI da riconciliare può essere personalizzata predisponendo su di un file di testo la query SPARQL che estrae l'elenco degli URI dei servizi da riconciliare, e valorizzando il parametro `--target-query`, oppure `-t`, con il percorso e nome del file che contiene tale query SPARQL. Se la target query non è specificata, il tool individua tutte le istanze, di qualsiasi classe, per le quali siano istanziate tutte le seguenti proprietà: `schema:name`, `schema:addressLocality`, `schema:streetAddress`, `km4c:houseNumber`. Tali istanze sono quelle che il tool tenta in questo caso di riconciliare.
2. RECUPERO DEI DATI RELATIVI AL SERVIZIO DA RICONCILIARE. Per ciascuna istanza di servizio da riconciliare, vengono recuperate attraverso un'opportuna query SPARQL, le valorizzazioni delle proprietà `schema:name`, `schema:addressLocality`, `schema:streetAddress`, che devono essere obbligatoriamente valorizzate, altrimenti la riconciliazione per indirizzo non può essere sicuramente eseguita, e si passa direttamente a valutare la possibilità di una riconciliazione basata sulle coordinate geospaziali eventualmente disponibili per il servizio da riconciliare. Inoltre, se disponibili, vengono recuperate anche le proprietà: `km4c:houseNumber`, `geo:lat`, `geo:long`.
3. TENTATIVO DI RICONCILIAZIONE PER INDIRIZZO. Con i dati recuperati al punto 2, si interroga il servizio REST Address/POI search by text. Se non si ottengono risultati al primo tentativo, vengono eseguiti ulteriori tentativi, apportando piccole modifiche sul

[84] Documentazione completa del servizio alla pagina http://www.disit.org/drupal/?q=home&axoid=urn%3Aaxmedis%3A00000%3Aobj%3A1f27dffd-0d3a-44a5-9e2b-194dd85c3be3, paragrafo 4.12

[85] Archivio serv2osm.jar comprensivo di sorgenti, in ubuntu@hadoopnodet05:~/shared/serv2osm/dist

[86] osm_ingestion/serv2osm

numero civico e sulla denominazione della strada, nel tentativo di ottenere una risposta. Se in nessun modo si ottiene una risposta, il tentativo di riconciliazione basato sull'indirizzo si considera fallito, e si passa all'eventuale tentativo di riconciliazione basato sulle coordinate geospaziali del servizio. Invece, nel caso in cui si ottengano molteplici risultati, dovendo invece necessariamente associare al servizio un solo numero civico, si deve operare una scelta. Se si conoscono le coordinate geospaziali del servizio, la scelta viene operata selezionando tra tutti i numeri civici, quello che si trova più vicino al servizio. Diversamente, viene associato il primo tra tutti i numeri civici restituiti.

4. TENTATIVO DI RICONCILIAZIONE PER COORDINATE GEOSPAZIALI. Se per il servizio da riconciliare, sono istanziate le proprietà `geo:lat` e `geo:long`, allora nel solo caso in cui la riconciliazione per indirizzo di cui al precedente punto 3 sia fallita, si tenta la riconciliazione per posizione. In particolare, si interroga il database relazione su cui si trova la rappresentazione della mappa OSM a partire dalla quale è costruito il grafo strade di Km4City, alla ricerca della strada più vicina al punto in cui si trova il servizio, ed anche del comune entro i cui confini cade il punto in cui si trova il servizio. Noti la strada e il comune, si torna ad interrogare il servizio REST Address/POI search by text, utilizzando questa volta non l'indirizzo scritto sull'istanza del servizio da riconciliare, ma l'indirizzo recuperato da Open Street Map. Tuttavia, questo non ci garantisce di ottenere risultati. Infatti, benché la via richiesta esista certamente sul grafo strade di Km4City, può sempre accadere che per essa non esistano numeri civici, nel qual caso il servizio non ritorna risultati dal momento che esso restituisce soltanto numeri civici. Inoltre, può sempre accadere che la ricerca testuale fallisca, per motivi di indicizzazione od altro. Ad ogni modo, se eseguendo la ricerca con questo nuovo indirizzo così recuperato, si riescono ad ottenere risultati, allora la riconciliazione per posizione è riuscita, e si passa al servizio successivo.
5. RICONCILIAZIONE PARZIALE PER POSIZIONE. Invece, se non si ottengono risultati attraverso la procedura di cui al punto 4, allora si utilizza l'indirizzo di OSM recuperato allo stesso punto 4, per interrogare Virtuoso e recuperare l'istanza di `km4c:Road` che corrisponde a quell'indirizzo. Così facendo, si esegue una riconciliazione almeno parziale, e ci si garantisce che anche nel caso in cui una via non abbia numeri civici, o nel caso in cui il servizio di ricerca testuale non funzioni correttamente, se il servizio da riconciliare è georeferenziato, esso viene in ogni caso almeno parzialmente riconciliato.
6. FINALIZZAZIONE. Le triple risultanti dall'attività descritta ai precedenti punti, sono infine memorizzate in un opportuno file di testo. La posizione e il nome del file di testo sono specificati dall'utente attraverso il parametro `--output`, o `-o`. Una conta dettagliata riguardante l'esito delle riconciliazioni viene inoltre scritta sul log, livello `INFO`, al termine dell'esecuzione dell'algoritmo.

Esempio di utilizzo

L'eseguibile JAR che costituisce lo stato dell'arte nell'implementazione dell'algoritmo descritto, si trova alla seguente posizione:
`ubuntu@hadoopnodet05:~/shared/serv2osm/dist/serv2osm.jar`.

Invocandolo senza fornire in ingresso alcun parametro, si ottiene la guida all'utilizzo, con l'elencazione dei parametri che possono essere inviati in ingresso, e del loro significato.

Un esempio di invocazione è il seguente:

```
java -jar ./serv2osm.jar --virt-host-serv 192.168.0.999 --virt-
port-serv 1234 --virt-user-serv [nomeutente] --virt-pwd-serv
[password] --virt-host-osm 192.168.0.999 --virt-port-osm 1234 --
virt-user-osm [nomeutente] --virt-pwd-osm [password] --pgre-host-
osm 192.168.0.999 --pgre-port-osm 1234 --pgre-user-osm
```

```
[nomeutente] --pgre-pwd-osm [password] --pgre-db-osm [nomerdb] --
service-endpoint
"http://192.168.0.999:8080/smosmm/api/v1/location/?" -t
"/home/lnxusr/shared/serv2osm/dist/n3/digital_locations/target_ser
vices.sparql" --log verbose --output
"/home/lnxusr/shared/serv2osm/dist/n3/digital_locations/reconcilia
tions.ttl"
```

Attraverso questo comando, l'utente:

- specifica l'indirizzo e i dati di accesso dell'istanza di Virtuoso in cui si trovano i servizi che devono essere riconciliati, attraverso i parametri: `--virt-host-serv --virt-port-serv --virt-user-serv --virt-pwd-serv`
- specifica l'indirizzo e i dati di accesso dell'istanza di Virtuoso in cui si trova il grafo stradale di Km4City, attraverso i parametri: `--virt-host-osm --virt-port-osm --virt-user-osm --virt-pwd-osm`
- specifica l'indirizzo, i dati di accesso, e il nome del database relazionale dove si trovano i dati importati da Open Street Map tramite osmosis, che è poi lo stesso database relazionale cui si accede per la triplificazione del grafo stradale: `--pgre-host-osm --pgre-port-osm --pgre-user-osm --pgre-pwd-osm --pgre-db-osm`
- specifica la base URI del servizio REST Address/POI search by text, attraverso il parametro `--service-endpoint`
- specifica dove si trova la query SPARQL che deve essere utilizzata per estrarre la lista degli URI dei servizi che devono essere riconciliati, attraverso il parametro: `-t`
- specifica dove debbano essere scritte le triple risultanti dalla riconciliazione, attraverso il parametro `--output`
- specifica che devono essere scritti sul log tutti i dettagli relativi all'esecuzione di ciascuna riconciliazione, attraverso il parametro: `--log` valorizzato a `verbose`.

Il parametro `--idle` serve invece nel caso in cui si vogliano frapporre delle pause tra richieste successive al servizio REST piuttosto che a Virtuoso o al database relazionale. Se ne consiglia l'utilizzo soltanto in caso di comprovata necessità.

Mappatura di una coppia di coordinate geospaziali su entità OSM

Ci si pone il problema di ricondurre un punto arbitrario, caratterizzato da latitudine e longitudine, ad un'entità OSM, che può essere la regione, provincia, comune in cui il punto si trova, ma può essere anche la `way` (elemento stradale) o il `node` (giunzione di elementi stradali) OSM più vicino al punto, a seconda delle richieste dell'utente.

Il problema è risolto interrogando opportunamente il database relazionale Postgresql generato a partire dalla mappa di OSM via `osmosis`.

Il servizio è offerto attraverso un'interfaccia Web nella quale, a fronte di una richiesta HTTP nella quale l'utente specifica le coordinate del punto di interesse ed il tipo di risultato a cui è interessato (eventualmente), viene restituito in risposta l'ID OSM dell'entità richiesta.

Il file WAR pronto per essere deployato su Tomcat o similare, si trova nella cartella `latlon2osnode` del repository SVN `osm_ingestion`.

La riconciliazione di un punto arbitrario con un elemento stradale o con un nodo del grafo stradale OSM richiede circa 5 secondi. La riconciliazione con entità di livello superiore è pressoché istantanea.

La servlet deve essere configurata indicando i dati di accesso al database Postgresql con estesione

PostGIS, formato simple, generato con Osmosis. La configurazione è contenuta in un file properties la cui posizione deve essere opportunamente specificata all'interno del descrittore di deploy.

L'esigenza di procedere a questo genere di riconciliazione è sorta originariamente in conseguenza del fatto che si avevano le fermate degli autobus geo-referenziate ma scollegate dal grafo stradale, e invece per poterle utilizzare nell'algoritmo di calcolo del percorso, era necessario che ciascuna fermata dell'autobus potesse essere mappata su di un nodo appartenente al grafo stradale. Siccome le fermate dell'autobus si raggiungono a piedi, è fondamentale che il nodo OSM riconciliato con la palina sia raggiungibile a piedi. E' possibile imporre questo vincolo aggiungendo il parametro `restrictions` valorizzato a 1 alla query string della richiesta http.

La ricerca della strada e del nodo OSM più vicino è gerarchica: si individua prima il comune in cui il punto si trova, poi si individuano le way che ricadono almeno in parte entro quel comune e si individua tra queste quella più vicina, e poi si individua il nodo più vicino tra quelli che delineano il percorso della way individuata come la più vicina.

Rappresentazione del legame tra paline del TPL e nodi OSM

Ci si pone il problema di rappresentare il legame tra le paline delle fermate degli autobus (e più in generale, tra le fermate del Trasporto Pubblico Locale) e i nodi del grafo stradale di Open Street Map.

Il problema si pone perché abbiamo a disposizione i dati relativi alla denominazione e alla posizione (coordinate spaziali) delle fermate degli autobus dei diversi vettori, ma tale informazione non poteva essere sino ad oggi utilizzata per il calcolo dei percorsi, perché affinché si possa fare ciò, deve esistere una corrispondenza tra la fermata dell'autobus e un nodo del grafo stradale di Open Street Map. Infatti, gli algoritmi di calcolo del percorso lavorano soltanto sul grafo stradale di OSM.

Si tratta di un problema complesso, che è stato infatti scomposto in due diverse attività.

Da un lato, era necessario dotarsi di uno strumento che, date le coordinate geospaziali di un punto, fosse in grado di individuare, tra tutti i nodi appartenenti al grafo stradale di OSM, quello più vicino. Questo strumento è stato approntato ed è disponibile nella forma di una servlet, che può essere invocata inviando in ingresso latitudine e longitudine, ed ottenendo in risposta l'ID del nodo OSM più vicino. Per maggiori informazioni su questa parte del lavoro, si può fare riferimento al capitolo Mappatura di una coppia di coordinate geospaziali su entità OSM.

Dall'altro, era necessario utilizzare tale strumento, per la produzione di file CSV che potessero essere importati negli algoritmi di calcolo del percorso, ed anche per la produzione di triple RDF da caricare sull'RDF store di Km4City. Questa seconda parte del lavoro, è quella che viene specificamente trattata in questo capitolo, ed è stata realizzata implementando componenti di ETL ad-hoc, da integrare nell'attuale flusso di ingestion e triplificazione dei dati provenienti dalla Regione Toscana relativi al Trasporto Pubblico Locale.

Overview

La produzione del file CSV ed il popolamento di una nuova tabella HBase contenente i soli dati relativi alla riconciliazione delle paline del TPL sui nodi OSM viene eseguita nella fase di ingestion dell'ETL di importazione dei dati della Regione Toscana relativi al TPL.

Invece, la produzione delle triple RDF in cui, attraverso la valorizzazione di un'opportuna proprietà, si va a stabilire una corrispondenza tra le paline e i nodi OSM, viene eseguita nella fase di triplification.

I job e le trasformazioni sino ad oggi prodotte, sono pensate nell'ottica di eseguire una prima rinconciliazione tra le paline e i nodi OSM, attraverso esecuzioni separate rispetto al flusso di ETL pre-esistente, per evitare di appesantirlo troppo. Una volta eseguita la riconciliazione iniziale, si dovrà prevedere l'integrazione di queste funzionalità all'interno dell'ETL pre-esistente, di modo che se dovessero intervenire delle variazioni sulle paline a regime, queste possano essere ingerite e triplificate senza che vi sia necessità di procedere a lanci di ETL ulteriori rispetto a quello usualmente

eseguito per la triplificazione dei dati del trasporto pubblico locale.

Ingestion

La parte di ingestion viene correntemente eseguita da un job denominato `Stop2OSM_FirstLaunch` il quale recupera i necessari parametri di configurazione, dopodiché inizializza se del caso il file CSV creandolo ed aggiungendo una riga di intestazione (trasformazione `CSVH`), ed esegue la trasformazione denominata `Stops2OSM` che esegue il lavoro vero e proprio di compilazione del file CSV e di scrittura sulla tabella HBase dei dati frutto della riconciliazione.

In particolare, la trasformazione legge dalla tabella HBase che contiene tutte le fermate del TPL[87], scarta quelle per le quali sia stata già eseguita la riconciliazione con un nodo OSM (scarta cioè quelle il cui ID sia rintracciabile nella tabella delle riconciliazioni[88]), e per le rimanenti esegue l'invocazione del servizio di riconciliazione delle coordinate geospaziali su nodi OSM con il vincolo di scartare i nodi che non siano raggiungibili a piedi, verifica che la risposta fornita dal servizio sia valida, e scrive la nuova corrispondenza in append sul file CSV. Poi, prepara la URI del nodo OSM a partire dall'ID dello stesso nodo secondo lo standard in uso in KM4C[89] e scrive la corrispondenza sulla tabella delle riconciliazioni, assieme al timestamp che individua il momento in cui la corrispondenza è stata instaurata, e che verrà utilizzato in fase di triplificazione per determinare se per tale corrispondenza sia stata già generata o meno una triple RDF. I componenti di ETL descritti in questo paragrafo sono pubblicati sul repository[90], sia quelli prodotti per la riconciliazione iniziale, sia quelli prodotti per essere integrati nell'ETL complessivo.

Triplification

Il job `Stops2OSM` che si trova conservato anch'esso nel repository `osm_ingestion`, ricalca i job omologhi responsabili della triplificazione degli altri dati importati dalla Regione Toscana per ciascun vettore, ivi compreso il job che esegue la triplificazione delle fermate al netto della riconciliazione con i nodi OSM, differenziandosi da essi soltanto per la parte di esportazione da HBase verso la tabella MySQL sorgente di Karma, e questo è vero sia per il job pensato per eseguire la riconciliazione iniziale, sia per il job responsabile della riconciliazione a regime.

Il job esegue quindi in particolare l'eliminazione della tabella MySQL sorgente Karma per la triplificazione della mappatura delle paline su nodi OSM, seguita dalla creazione della stessa tabella vuota, dal popolamento della stessa tabella con i dati prelevati dalla tabella HBase popolata in fase di ingestion filtrati per processo[91] e per timestamp (non devo ripetere la triplificazione di dati già esportati come triple RDF in un tempo precedente), dalla verifica della presenza di dati da triplificare, e dall'opportuna invocazione di Karma[92] per l'esecuzione vera e propria della triplificazione.

La trasformazione `Stops2OSM_ToSQL`, memorizzata alla stessa posizione del job di triplificazione `Stops2OSM` che la invoca, ricalca le omologhe trasformazioni che eseguiono il trasferimento dei dati dalle tabelle HBase verso le tabelle MySQL per gli altri dati importati dalla Regione Toscana per ciascun vettore, differenziandosi da queste soltanto per la tabella HBase sorgente, la tabella MySQL destinazione e per la presenza di parametri di ingresso a sostituire i riferimenti diretti alle variabili Pentaho. La trasformazione esegue niente più che la lettura dei dati dalla sorgente HBase e la scrittura degli stessi tal quali sulla corrispondente tabella MySQL.

La trasformazione `getTime`, anch'essa memorizzata nel repository, è stata inserita per convenienza, ma non è stata sviluppata nell'ambito di questo task. Si tratta infatti di una trasformazione già presente

[87] TPLBus_stops
[88] TPLBus_stops_osm
[89] http://www.disit.org/km4city/resource/OS###NO, con ### identificativo OSM del nodo
[90] osm_ingestion\busstops2osm
[91] Vettore (compagnia) di trasporto
[92] http://usc-isi-i2.github.io/karma/

ed utilizzata nell'ETL pre-esistente[93], che il nuovo job `Stops2OSM` si limita ad utilizzare tal quale.

Model

L'esportazione dei dati conservati su MySQL nella forma di triple RDF viene eseguita da Karma sulla base di un modello, sostanzialmente un documento di configurazione nel quale è descritta la mappatura tra i campi della tabella relazionale e le istanze, classi e proprietà della rappresentazione RDF degli stessi dati.

Il modello predisposto per la triplificazione del legame tra le paline del TPL ed i nodi OSM è anch'esso conservato nel repository `osm_ingestion`, sottocartella `Models` di `Triplification`, nome `TransferServiceAndRenting__basedNear__Node`.

Come si può ben capire osservandolo, il file non è prodotto manualmente, ma utilizzando un'apposita interfaccia Web dello stesso Karma. Maggiori informazioni sull'utilizzo di Karma sono disponibili nella documentazione del progetto SiiMobility[94].

Importazione dei numeri civici dalle basi dati della Regione Toscana

Ci si pone il problema di come poter importare nella mappa Open Street Map i numeri civici presenti sulle basi dati della Regione Toscana (che si assumono conservati in tabelle MySql) dal momento che i numeri civici nativamente presenti sulle mappe OSM non sono sufficienti per i nostri scopi.

La proposta è quella di generare nuovi nodi sulla nostra copia locale della mappa OSM, senza sottomettere la modifica alla community di Open Street Map, utilizzando ID negativi di modo che mai aggiornando la nostra copia locale via `osmosis`[95] possano generarsi conflitti tra i nuovi dati importati da Open Street Map ed i dati da noi introdotti sulla sola nostra copia locale.

Per generare nuovi nodi sulla nostra copia locale, la proposta è di non intervenire direttamente sul database relazionale, bensì di generare un documento XML di modifiche in formato idoneo ad essere digerito da `osmosis` (attraverso un apposito strumento eseguibile da riga di comando), dopodiché eseguire opportunamente `osmosis` inviando in ingresso il file di modifiche prodotto.

La proposta è quella di generare un file di modifiche che intervenga soltanto in aggiunta. Infatti, i nodi nativi di OSM taggati con numeri civici, sono in generale utilizzati anche come nodi di giunzione degli elementi stradali, e quindi rimuovendoli, perderemmo di precisione nella rappresentazione del tracciato delle strade, oltre a perdere tutto un insieme di altre informazioni eventualmente presenti sui nodi, aggiuntive rispetto all'indicazione del numero civico. Inoltre, modificare i nodi originari eliminando il solo tag del numero civico, sarebbe comunque inefficace in generale perché siccome l'aggiornamento delle mappe avviene al livello di nodo, relazione o way, quindi non di tag, qualsiasi modifica che nel corso del tempo venisse apportata sui nodi nativi OSM taggati come numeri civici, sarebbe eseguita dal contributor andando a riportare entro il tag di update del file OSC l'intero nodo con tutti i propri tag, compreso quindi in generale anche il numero civico, ed importando poi noi questo aggiornamento, torneremmo ad avere sul nodo nativo OSM il tag del numero civico, con conseguente duplicazione dei numeri civici.

La scelta di operare soltanto in aggiunta non è tuttavia priva di conseguenze: seguendo questo approccio, è infatti necessario non soltanto procedere all'aggiunta dei numeri civici provenienti dalla Regione Toscana preventivamente rispetto alla triplificazione del grafo strade OSM, ma anche introdurre nella configurazione della triplificazione un nuovo parametro attraverso cui si possa andare a specificare se i numeri civici che devono essere triplificati siano quelli nativi di OSM o quelli importati dalla Regione Toscana, ed utilizzare poi opportunamente questo parametro nello script SQL di preparazione dei dati per la triplificazione ed anche seppur marginalmente nello script SML di

[93] /media/Trasformazioni/TrasformazioneTPLBus/Triplification/getTime.ktr sulla 192.168.0.36
[94] http://www.disit.org/6990
[95] http://wiki.openstreetmap.org/wiki/Osmosis

configurazione di Sparqlify.

Lo strumento command-line per la generazione del file di modifiche

Lo strumento ad oggi approntato per la generazione del file OSC, documento XML indoneo per essere inviato in ingresso ad `osmosis` per l'applicazione delle modifiche sul database relazionale che costituisce la copia locale della mappa Open Street Map confinata ad un certo territorio di interesse, è un software eseguibile da riga di comando denominato `rthousenum`, pubblicato sul repository[96].

Il software legge da un database MySql sul quale si assume si trovino i dati importati dalla Regione Toscana, eseguendo una query del tutto personalizzabile che deve essere inviata come parametro di ingresso, e che consente di avere totale flessibilità rispetto a variazioni nella struttura del database sorgente, ed anche rispetto all'eventuale esigenza di importare di volta in volta soltanto una parte dei numeri civici disponibili sulle basi dati della Regione Toscana. L'applicativo utilizza il risultato di questa query per aggiungere elementi `create` con figli `node` opportunamente taggati, al documento OSC che dovrà poi essere inviato in ingresso ad `osmosis` per l'applicazione delle modifiche.

Per tutti i dettagli sull'utilizzo dello strumento da riga di comando, è possibile eseguirlo senza inviare alcun parametro in ingresso. Si otterrà una guida all'utilizzo con l'elencazione di tutti i parametri che il software accetta in ingresso, e del loro significato. Si tratta per grandi linee:

- dei parametri per la connessione al database MySql che contiene i dati della Regione Toscana;
- della query da eseguire sul database MySql per andare a selezionare i dati a partire dai quali i nuovi nodi devono essere generati;
- dei parametri per la connessione al database Postgresql che contiene la copia locale della mappa OSM che deve essere integrata con i nuovi numeri civici;
- dell'URL dell'endpoint del servizio `latlon2osnode` che viene utilizzato per determinare, data la latitudine e la longitudine di un punto in cui si trova un numero civico, quale sia la strada ad esso più vicina sulla mappa OSM (verosimilmente ed auspicabilmente benché non necessariamente, la strada su cui il numero civico si trova nel mondo reale), quale sia il comune in cui il numero civico si trova guardando sulla mappa OSM, e quale sia il nodo nativo OSM più vicino al numero civico che si sta introducendo, per fini di routing;
- del path del file di output, del livello di log, e dell'eventuale tempo di attesa prima di ogni chiamata a servizi esterni o basi di dati relazionali.

In riguardo all'output che viene generato da questo strumento, vale soltanto la pena di notare che esso valorizza l'attributo `addr:street` di ciascun nodo con il nome della strada così come esso appare sulla mappa OSM, seguito da punto e virgola, e dal nome della strada così come esso era invece scritto nei dati originari della Regione Toscana. E' importante che gli applicativi client siano sviluppati nella consapevolezza di questa ridondanza di informazione, e la sfruttino correttamente. Infatti, non sempre c'è corrispondenza tra le due denominazioni. Capita infatti che su OSM per alcune strade il contributor non abbia indicato una denominazione, così come capita che ne abbia indicata una completamente errata facendo riferimento ad una via completamente diversa da quella riportata sui dati della Regione Toscana, così come capita (ed è anzi praticamente la regola) che ci siano lievi differenze tra la denominazione presente in OSM e quella presente nei dati della Regione Toscana.

Vale inoltre la pena di notare che i nodi che rappresentano numeri civici importati dalla Regione Toscana sono taggati con un tag `source` valorizzato a `Regione Toscana`, il che permette di distinguerli del tutto agevolmente dagli altri in fase di triplificazione, senza doversi affidare a dettagli implementativi quali l'assegnazione di identificativi univoci negativi piuttosto che positivi.

Vale infine la pena di notare che i nuovi nodi generati sono taggati con `ref` valorizzato con l'ID del

[96] Repository osm_ingestion, cartella rthousenum.

nodo nativo OSM che si trova più vicino al nuovo nodo introdotto, e che appartiene al grafo stradale (è cioè nodo di giunzione di un elemento stradale).

L'applicazione delle modifiche con osmosis

Una volta generato il file di modifiche utilizzando lo strumento command-line sviluppato ad-hoc di cui al precedente paragrafo, le modifiche devono essere applicate alla copia locale della mappa OSM, rappresentata come database relazionale Postgresql con schema `pgsimple`.

A tale scopo, si deve utilizzare il tool da riga di comando `osmosis`, più volte richiamato in questa documentazione, andando a leggere il file OSC ed andando quindi ad applicare le modifiche in esso contenute al database corretto. Ciò può essere fatto lanciando un unico comando[97] come descritto sul Wiki[98] di `osmosis`.

Le modifiche sul sottosistema di triplificazione del grafo OSM

Si propone di introdurre una nuova tabella di configurazione, da denominare `extra_config_civic_num`, con un campo `civic_num_source`, che l'operatore di volta in volta sarà chiamato a valorizzare alternativamente con `Regione Toscana` piuttosto che `Open Street Map` ad indicare il fatto che nel preparare i dati per la triplificazione si devono tenere in considerazione i soli numeri civici importati dalla Regione Toscana piuttosto che i soli numeri civici disponibili nativamente sulla mappa di Open Street Map rispettivamente.

Esaurito ciò che l'operatore deve sapere, allo sviluppatore potrebbe invece essere utile la conoscenza di quanto segue relativamente allo script di preparazione della triplificazione, file SQL:

- la gestione dei numeri civici importati dalla Regione Toscana ha imposto una modifica nella tabella `extra_ways` popolata in fase di preparazione dei dati per la triplificazione, ed in particolare ha imposto il fatto di portare sulla target list due campi aggiuntivi, `start_node_id` e `end_node_id`, valorizzati con gli OSM ID dei nodi che rappresentano il nodo iniziale e finale del frammento di elemento stradale rappresentato sulla riga;
- tali ID, sono recuperati e portati fuori nelle tabelle `extra_tmp_1` ed `extra_tmp_2`, per poi essere fattivamente utilizzati nel popolare la tabella `extra_streetnumbers_on_nodes`, dove l'aggancio del numero civico con la strada e conseguentemente con l'elemento stradale avviene, nel caso particolare dell'importazione dei numeri civici dalla Regione Toscana, andando ad agganciare il riferimento al nodo nativo più vicino conservato nel tag `ref` del nodo importato, con appunto indifferentemente il campo `start_node_id` piuttosto che `end_node_id`;
- nel popolare la tabella `NodeStreetNumberRoad`, quella da cui in definitiva si attinge per la triplificazione, si va poi a verificare che la tipologia di nodo sia quella corretta, andando a scartare alternativamente i nodi taggati come numeri civici provenienti dalla Regione Toscana piuttosto che i nativi di Open Street Map. Invece, nel popolare le tabelle `RelationStreetNumberRoad` e `RelationStreetNumberRoad2`, ci si è assicurati di ritornare un insieme vuoto nel caso in cui la configurazione sia impostata su Regione Toscana, perché i numeri civici importati dalla Regione Toscana sono tutti quanti rappresentati come nodi taggati con numero civico ed indirizzo completo.

Lievi modifiche sono state apportate anche sullo script di configurazione di Sparqlify, file SML, in particolare nell'andare a valorizzare la proprietà di nuova introduzione `km4c:nearTo` con la

[97] Ad es. `osmosis --rxc file="/home/ubuntu/changefile.osc" --wsc database="pgresdbname" user="pgresdbuser" password="pgresdbpwd"`

[98] http://wiki.openstreetmap.org/wiki/Osmosis/Detailed_Usage_0.41

valorizzazione del tag `ref` nel solo caso dei numeri civici importati dalla Regione Toscana.

Quanto detto si applica sia agli script di install che a quelli di update relativi alle classi `km4c:StreetNumber` e `km4c:Entry`.

Massimizzare le performance utilizzando l'indicizzazione geo-spaziale

Attraverso l'opportuna indicizzazione delle tabelle che contengono geometrie PostGIS sul database Postgres che rappresenta la mappa OSM, e l'opportuna interrogazione di tali tabelle, è possibile ottenere risultati di estremo rilievo in termini di performance[99].

In particolare, immediatamente dopo la creazione del database, e ad ogni aggiornamento dei dati in esso contenuti, è di fondamentale importanza procedere alla creazione delle seguenti relazioni:

- una relazione che contiene gli OSM ID degli elementi `relation` di OSM che rappresentano enti territoriali, ciascuno con associata la geometria esatta del confine, e la geometria del bounding box dell'ente territoriale, entrambe indicizzate;
- una relazione contenente gli OSM ID degli elementi `way` di OSM che rappresentano elementi stradali;
- un indice opportuno per il campo `geom` della tabella `nodes`, e per il campo `linestring` della tabella `ways`.

Le query che si propone di eseguire per la creazione di tali relazioni sono pubblicate nel repository[100].

Questo accorgimento permette di conseguire risultati importanti in termini di prestazioni nella riconciliazione di un punto arbitrario su di un'entità di OSM. Si veda in proposito anche il capitolo dedicato: Mappatura di una coppia di coordinate geospaziali su entità OSM.

Inoltre, estendendo l'utilizzo dell'indicizzazione ed adottando accorgimenti opportuni, ed in particolare riscrivendo opportunamente le interrogazioni, nella fase di preparazione dei dati su database relazionale destinati poi ad essere utilizzati da Sparqlify per la generazione del grafo stradale Km4City a partire dalle mappe OSM, si sono conseguiti risultati di estremo rilievo anche in termini di efficienza anche in questo ambito. Al momento attuale, la produzione delle triple del grafo stradale di Km4City per una media provincia (Città Metropolitana di Cagliari) giunge a compimento in un tempo inferiore alle due ore.

[99] Riduzione ad un decimo del tempo di riconciliazione di un punto arbitrario su di un nodo del grafo stradale OSM, riduzione di due ordini di grandezza del tempo di individuazione dell'ente in cui si trova un punto arbitrario

[100] Repository osm_ingestion, cartella doc, script performance_optimization.sql