Urban Data Consistency in RDF: A Case Study of Curitiba Transportation System

Mirian Halfeld-Ferrari[1], **Carmem S. Hara**[2], **Nádia P. Kozievitch**[3], **Flavio R. Uber**[2,4]

[1]Université d'Orléans, INSA CVL, LIFO EA 4022 FR-45067, Orléans, France

[2]Universidade Federal do Paraná, Curitiba-PR, Brazil

[3]Universidade Tecnológica Federal do Paraná, Curitiba-PR, Brazil

[4]Universidade Estadual de Maringá, Maringá-PR, Brazil

mirian@univ-orleans.fr, carmem@inf.ufpr.br,
nadiap@utfpr.edu.br, flavio.uber@gmail.com

Abstract. *Urban Computing has an important role in providing new tools for urban mobility. In this paper integrity constraints and blank nodes are used in an RDF database to minimize extra updates (called side effects) to guarantee consistency during required updates. A study case using a real scenario on Curitiba/Brazil transportation database is presented to support users identifying inconsistencies and preventing undesirable updates. Experiments demonstrated shorter execution time and meaningful results when compared with similar strategies.*

1. Introduction

The growth of urban centers sets several challenges to human well-being of which many are associated to urban mobility, such as traffic jams, energy consumption, security, mobility, longer travel times, health issues due to emissions, and stress. New approaches are required to provide tools for managing a city, reconciling operations of several systems so that their performance fits to better serve its inhabitants. In this scenario, urban computing has emerged, in order to assist and improve the management of resources and decision making, by exploring data generated by a diversity of sources in urban spaces, such as sensors, traffic devices, and vehicles.

Urban related data can come in different formats and the problem of integrating these multitude of heterogeneous data sources carries the same challenges of traditional integration applications. Moreover, there is a movement towards making urban data freely available to the public. This is in line with the principles of Linked Data, which is a set of recommended best practices for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web. RDF is the standard model of the Semantic Web. Its simple model, based on triples (subject, property, object), makes data interpretable by machines and humans. Moreover, RDF may come along with ontological information which associates semantics to the published information. In particular, the ability to express equivalences using *sameAs* properties can be used to state that individuals in distinct datasets refer to the same entity in the real world[1]. This ability allows one to traverse across several data sources, which is an important and desirable feature in the context of urban computing.

Linking data may result in huge datasets[2]. Keeping the consistency of such large datasets is a challenge. One traditional approach for keeping data consistent is to al-

[1]https://www.w3.org/TR/owl-ref/

[2]The 2016 release of DBPedia contains more than 1 billion triples.

low users to define constraints that are checked whenever the database is updated. In [Halfeld-Ferrari et al. 2017], we have propose a system, called BNS, that detects RDF constraint violations, and either refuses the update, or computes a set of compensation actions in order to guarantee their satisfaction. In this paper, we show how BNS can be applied on a urban setting, using as case study transportation data of Curitiba, the capital of Paraná State in Brazil.

Curitiba has developed and implemented mass transport corridors, and mobility solutions using Bus Rapid Transit (BRT) systems in the 1970s, which has been featured in the Scientific American magazine [Rabinovitch and Leitman 1996]. The complete system, according to Curitiba's Institute of Research and Urban Planning (IPPUC)[3], includes about 482 routes, distributed among 9940 bus stops, 23 bus terminals and 17 categories of streets [Kozievitch et al. 2016]. There are different types of routes: express routes, that defines the city spoke-shaped structural axes, inter-district and local lines, that fill the space between spokes, besides feeder, worker's routes and the city center line. Each type of route is identified by buses of different colors. We are going to focus on express and the city center lines, which are illustrated in Figure 1. Express routes run in exclusive lanes using high capacity bi-articulated buses (called *Expresso*, in Portuguese), which transport up to 250 passengers per bus. They are identified by red vehicles. The city center line circles the downtown area in two directions, providing easy access to the commerce, services, and points-of-interest, reducing the need for driving to the city and finding parking spots. As opposed to the huge *Expressos*, these routes use white minibuses, that pass every 8 minutes and complete the circle in around 35 minutes. They also have a cheaper fare, and only accepts payment with the transportation card. In this scenario, this paper shows how the Curitiba urban transportation database were exported in RDF format and how constraints were managed by the BNS system in order to maintain the database consistency after insertion and deletion operations.

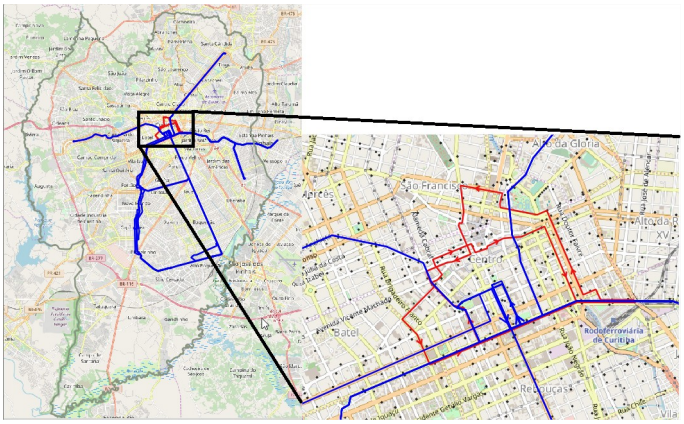

Figure 1. Corridors for BRT system (blue) along city center bus line (red). The zoom on the center district (right) presents the bus stops as black spots.

Figure 2. *Expresso* **bus and a tube stop.**

The rest of the paper is organized as follows. In section 2 we present the overall architecture of the system, and an example of the application of constraints on keeping the

[3]`http://www.ippuc.org.br/` – Last visited on Feb 10, 2017.

database consistency. Details of the BNS system are given in Section 3. In section 4 we describe our new implementation of the system, which integrates the constraint management system with a database system, and present some experimental results. Related work are discussed in Section 5, and Section 6 concludes the paper.

2. System Architecture

In this section we present the system components, the RDF database and the BNS system, as well as a motivating example to illustrate our main ideas. The RDF database has been generated by importing data from the Curitiba Urbanization Company (URBS). The resulting database contains RDF schema properties (such as *Class, Subclass, Domain, Range, Property and Sub-property*), as well as RDF triples. As an example, consider tables `Route` and `Stop`, illustrated in Figures 3(a) and (b), respectively, extracted from the URBS database. By joining these tables, we generated a set of triples, illustrated in Figure 3(c), that associates an express route (such as 302) to a bus stop by property `ExpressLineStop`[4] We also store the class of each subject. For example, we keep the information that 302 is an `ExpressLine` and that `Estação Tubo Germânia` is a *tube* stop, as the one illustrated in Figure 2. The resulting RDF dataset contains 110,000 triples and 25,000 class instances using data from bus lines, bus stops and bus line characterization.

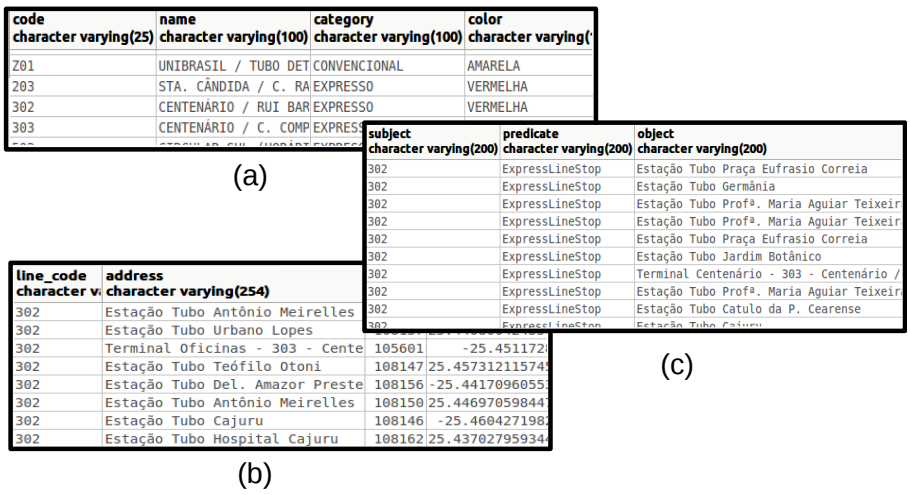

Figure 3. From the URBS Database to RDF triples.

The BNS system allows the user to define constraints such as: ”every express line stop must be a fast boarding stop”. Suppose that there are 3 types of stops: tube, terminal stop, and street stop, and that only the first two are considered as fast boarding. Consider now an update to the system, including a new stop, $s1$ to the express line 302. According to the constraints, $s1$ must be a fast boarding stop, and thus cannot be a street stop. However, if $s1$ is already stored in the database as a street stop, either: (i) the update operation is generating an inconsistency in the database, and thus should be rejected; or (ii) the database is inconsistent and $s1$ is indeed a fast boarding stop. The BNS system can help the detection of such inconsistencies because for every insertion or removal operation, it computes all additional updates that need to be executed in order to satisfy the existing

[4]In the examples we kept the same identifiers used in the original tables instead of URIs in order to simplify the exposition.

constraints. In the above example, it will generate an operation to remove $s1$ from the class of street stops. However, removal operations as side-effects of facts in the database indicate potential sources of inconsistencies, which should be double checked by the user.

Figure 4 illustrates the proposed process. Given an RDF database D, we consider the existence of a set of application constraints ($\mathcal{C}$) and RDF semantic constraints ($\mathcal{A}$) defined on D. The input to the BNS System is a set of update operation (upd). The system computes the set of additional updates (side effects) U_S to keep the consistency in database, according to $\mathcal{A}$ and $\mathcal{C}$. It is possible that operations in U_S may be contradictory among themselves. For example, one insertion operation in upd requires a stop to be fast boarding, while another update in upd requires the same stop to be a street stop. Since the *update* operations are contradictory, the BNS system automatically rejects the update set. If U_S does not contain contain contradictory operations, then U_S is checked against facts in the database. If there are side-effects that contradict facts already stored, user validation is required before applying the updates on the database. Next section presents details on the type of constraints considered by the BNS system, as well as of the process for side-effects computation.

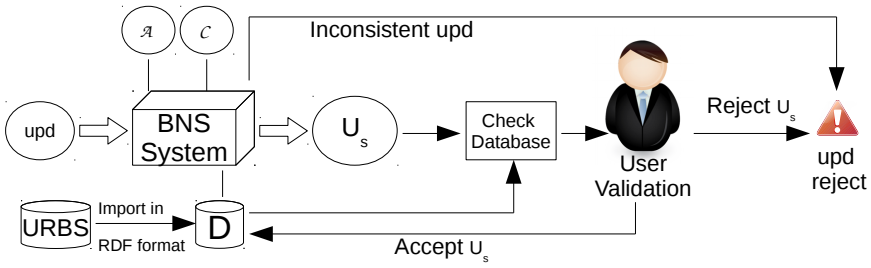

Figure 4. BNS System Process.

3. The Blank Node System (BNS)

Traditionally, whenever a database is updated, if constraint violations are detected, either the update is refused or compensation actions, which we call side-effects, must be executed in order to guarantee their satisfaction. In this section, we present the main ideas of the BNS system, which has proposed in [Halfeld-Ferrari et al. 2017]. It tackles the problem of "active rules" for RDF and computes the side-effects required by update operations. BNS is similar to the FKAC system [Flouris et al. 2013], but introduces blank nodes as free nulls, which are used as placeholders for unknown required data. As we will show in Section 4, this approach minimizes the number of side-effects and produce more meaningful results than the FKAC system. These features are important to help and reduce the burden of user validation.

The system considers application constraints ($\mathcal{C}$), specific for each application domain, and intrinsic RDF/S semantic constraints ($\mathcal{A}$). When updates are submitted, these constraints are checked, generating additional updates required to restore the database consistency. In this section, we use the following syntax. *CI(e,c)*, which stands for class instance, defines that an entity e is of class c, such as *CI(302, ExpressLine)*. *PI(s,o,p)*, which stands for property instance, defines that subject s is linked to object o by property p. An example is *PI(302, Estação Tubo Germânia, ExpressLineStop)*. A constraint is a logical rule r whose left-hand side is denoted as *body(r)*, while the right-hand side is denoted as *head(r)*. Figure 5(b) presents some constraints in the context of the transportation system in Curitiba. In particular, constraints $r_4 - r_7$ define the ones used as examples in Section 2.

Application Constraint: Let c_1, c_2 be class labels, and p_1, p_2 be property labels. Application constraints in $\mathcal{C}$ have the forms presented in Figure 5(a). As an example, constraint r_1 is of Type 1 and defines that minibuses are vehicles that accept only payment by transportation cards. Constraints r_2 and r_3 are of Type 2 and requires every express route to have a start stop and end stop (as opposed to the center city line, which has a closed trajectory). Constraint r_6 is of Type 3, and defines that all stops for express lines are fast boarding stops. The following restrictions are imposed on $\mathcal{C}$: (1) for constraints r' of Type 2 there exists no constraint $r'' \in \mathcal{C}$ such that $head(r')$ and $body(r'')$ are unifiable; (2) for constraints r' of Type 3 there exists no constraint $r'' \in \mathcal{C}$ such that $body(r')$ and $head(r'')$ are unifiable.

Type 1:
$CI(X_1, c_1) \rightarrow CI(X_1, c_2)$
$CI(X_1, c_1) \rightarrow \neg CI(X_1, c_2)$
$PI(X_1, X_2, p_1) \rightarrow PI(X_1, X_2, p_2)$
$PI(X_1, X_2, p_1) \rightarrow \neg PI(X_1, X_2, p_2)$
Type 2:
$CI(X_1, c_1) \rightarrow PI(X_1, X_2, p_1)$
$CI(X_1, c_1) \rightarrow \neg PI(X_1, X_2, p_1)$
$CI(X_2, c_1) \rightarrow PI(X_1, X_2, p_1)$
$CI(X_2, c_1) \rightarrow \neg PI(X_1, X_2, p_1)$
Type 3:
$PI(X_1, X_2, p_1) \rightarrow CI(X_1, c_1)$
$PI(X_1, X_2, p_1) \rightarrow \neg CI(X_1, c_1)$
$PI(X_1, X_2, p_1) \rightarrow CI(X_2, c_1)$
$PI(X_1, X_2, p_1) \rightarrow \neg CI(X_2, c_1)$

(a)

r_1:	$CI(V, Minibus) \rightarrow CI(V, CardOnly)$
r_2:	$CI(L, ExpressLine) \rightarrow PI(S, L, StartStop)$
r_3:	$CI(L, ExpressLine) \rightarrow PI(S, L, EndStop)$
r_4:	$PI(S, L, ExpressLineStop) \rightarrow CI(S, FastBoarding)$
r_5:	$CI(S, Tube) \rightarrow CI(S, FastBoarding)$
r_6:	$CI(S, Terminal) \rightarrow CI(S, FastBoarding)$
r_7:	$CI(S, StreetStop) \rightarrow \neg CI(S, FastBoarding)$
r_8:	$PI(S, L, ExpressLineStop) \rightarrow CI(L, ExpressLine)$
r_9:	$PI(S, L, ExpressVehicle) \rightarrow CI(L, ExpressLine)$
r_{10}:	$PI(S, L, ExpressLineStop) \rightarrow CI(S, ExpressStop)$

(b)

Figure 5. Constraint types and definitions.

We now turn to the problem of side-effects computation. Consider the insertion of the fact *CI(LineA, ExpressLine)*. The simple addition of this fact in the database renders it inconsistency because according to rules r_2 and r_3, every express line is required to have a start and end stop. Thus, the following side effects are produced: (i) rule r_2 generates *PI(Null*$_1$*, LineA, StartStop)*, and (ii) rule r_3 produces *PI(Null*$_2$*, LineA, EndStop)*. $Null_1$ and $Null_2$ are placeholders, indicating that $LineA$ has a start and end stops, although it is not yet known which ones.

Consider now the deletion of f = *PI(Stop2, LineA, StartStop)*. To avoid the violation of r_2 we cannot just eliminate f from the database D. The solution proposed by the FKAC system is to delete all facts the requires the existence of f, which results in cascading deletion operations. The application of constraint r_2, generates the removal of *CI(LineA, LineExpress)*, which in turn produces the removal of all *ExpressLineStop*s by r_8. Such a solution seems too radical. The BNS system adopts a different approach, which is similar to the "on delete set null" strategy, and generate a single side-effect operation, which is the inclusion of a blank node the replaces *LineA* start stop. That is, the removal of *PI(Stop2, LineA, StartStop)* generates as side-effect *PI(Null*$_1$*, LineA, StartStop)*.

RDF Intrinsic Semantic Constraint: Besides satisfying the application constraints, an RDF database should also respect the intrinsic RDF schema semantic constraints ($\mathcal{A}$). Thus, following the same reasoning used for constraints in $\mathcal{C}$, our approach proposes to generate additional updates in order to maintain consistency *w.r.t.* $\mathcal{A}$. Constraints in $\mathcal{A}$ are those presented in Tables 1 and 2. Here, $Cl(c)$ defines that c is a class, $Pr(p)$ defines p as a property, $Csub(c_1, c_2)$ defines subclass relationships, $Psub(p_1, p_2)$ defines sub-property relationships, and $Rng(p, c)$ and $Dom(p, c)$ define the range and domain of properties. Table 1 borrows from [Flouris et al. 2013] a subset of the RDF semantic constraints and

Table 2 presents the rules that have been modified and added in order to consider the existence of nulls.

Table 1. Subset of rules from [Flouris et al. 2013]

m_1:	Cl(x) ∧ Pr(y) → (x ≠ y)
m_2:	Cl(x) → Csub(x, rdfs:Resource)
m_3:	Ind(x) → CI(x, rdfs:Resource)
m_4:	Csub(x,y) ∧ Csub(y,z) → Csub(x,z)
m_5:	Pr(x) → Dom(x,y) ∧ Rng(x,z)
m_6:	CI(x,y) → Ind(x)
m_7:	CI(x,y) → Cl(y) ∨ (y=rdfs:Resource)
m_8:	PI(x,y,z) → Pr(z)
m_9:	CI(x,y) ∧ Csub(y,z) → CI(x,z)
m_{10}:	PI(x,y,z) ∧ Psub(z,w) → PI(x,y,w)

Table 2. Rules that involve blank nodes

b_1:	Pr(x) ∧ BN(y) → (x ≠ y)
b_2:	Ind(x) ∧ BN(y) → (x ≠ y)
b_3:	Cl(x) ∧ BN(y) → (x ≠ y)
b_4:	PI(x,y,z) ∧ $\neg BN(x)$ → Ind(x)
b_5:	PI(x,y,z) ∧ $\neg BN(y)$ ∧ $\neg Lit(y)$ → Ind(y)
b_6:	PI (x,y,z) ∧ Dom(z,w) ∧ $\neg BN(x)$ → CI(x,w)
b_7:	PI(x,y,z) ∧ Rng(z,w) ∧ $\neg BN(y)$ → CI(y,w) ∨ (Lit(y) ∧ (w=rdfs:Literal))

4. Experimental Study

In [Halfeld-Ferrari et al. 2017] we reported experiments of the BNS system implemented with the Standard ML of New Jersey compiler, and constraints defined on Berlin and LUBM benchmarks. While in the previous implementation data were stored on regular files, we have re-engineered the system in SML# [Ohori and Ueno 2011] to access data stored on a Postgres database, pushing some filter expressions to the DBMS. Although not all DBMS optimizations were explored, the execution time of the system has improved drastically. Moreover, here we consider a real urban dataset and constraints, as exemplified in Section 2.

We report experimental results comparing the BNS and FKAC systems. As pointed out in Section 3, one important difference between the FKAC and BNS systems concerns the ability to generate and store null values. As the FKAC system *does not consider nulls*, when an insertion imposes the existence of an unknown required data, it may arbitrary choose an entity to play the required part. As an example, consider again the insertion of *CI(LineA, ExpressLine)*. Since r_2 requires *LineA* to have start and end stops, the system may arbitrarily choose any *ExpressLineStop*s s_1, s_2, and generate as side-effects *PI(s_1, LineA, StartStop)*, and *PI(s_2, LineA, EndStop)*. Worse still, when in a later time, the real start and end stops were inserted, it is up the user to remove the previous arbitrary ones. In contrast, BNS stores null values which can *automatically* replaced by facts introduced by the user in a later time. This strategy helps the user validation process and produce operations that are semantically more meaningful. Besides, our experiments results, illustrated in Figure 6 show that BNS reduces the number of side-effects.

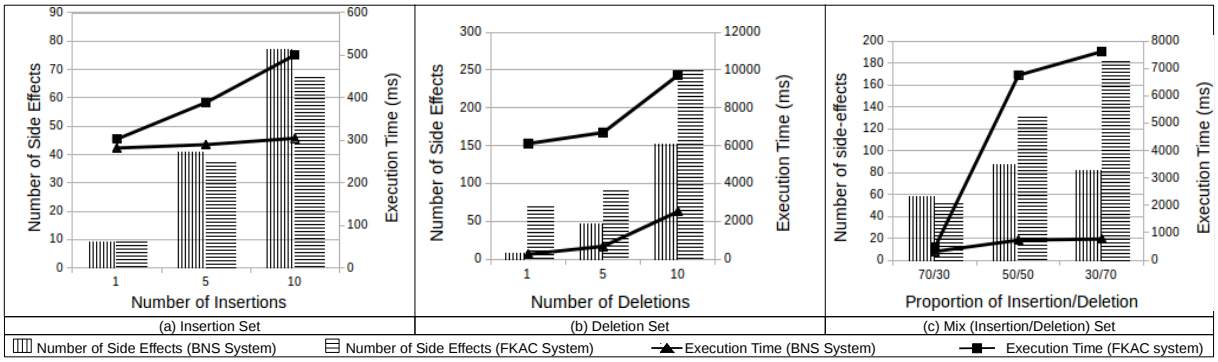

Figure 6. Results for URBS Database.

In the graph, the bars show the final update size with side-effects, while the lines present the execution time. For each experiment, we conducted 5 executions. The first two were to warm up and the reported results consist of the average time of the following 3 executions. Figure 6(a) show results when considering only insertion operations. In this setting, the number of side-effects in the BNS system is larger than in the FKAC system. This is caused by the intrinsic semantic constraints concerning nulls in the BNS system. Consider again the insertion of *CI(LineA, ExpressLine)*. While FKAC generates as side-effects *PI(s_1, LineA, StartStop)*, and *PI(s_2, LineA, EndStop)*, BNS generates *PI($Null_1$, LineA, StartStop)*, and *PI($Null_2$, LineA, EndStop)*, and also *CI($Null_1$,rdfs:resource)*, *BN($Null1$)*, *CI($Null_2$,rdfs:resource)*, and *BN($Null2$)* to satisfy the constraints depicted in Table 2. Even producing a larger set of side-effects, the BNS system is still faster because FKAC has an extra task for searching the database for a subject (or object) of the property domain (or range). Figure 6(b) considers only deletions operations. In this case the BNS System is faster and produces smaller sets of side-effects. Figure 6(c) shows the results when we mix insertion and deletions operations. When there are a larger number of insertions, both the execution time and amount of side-effects are similar for both systems. When deletion operations rate increases, both parameters grow faster for the FKAC system.

5. Related Work

Mechanisms to control frequent updates on RDF are desirable [Magiridou et al. 2005] and have been proposed in several works such as [Flouris et al. 2013, Frommhold et al. 2016]. RDF/S update and consistency maintenance approaches in [Flouris et al. 2013, Magiridou et al. 2005, Solimando and Guerrini 2013] do not consider blank nodes. In [Muñoz 2016] integrity constraints are used to extract information about the RDF database. Blank nodes have been used in different contexts such as for *version control* in [Völkel and Groza , Frommhold et al. 2016] or for *ontology evolution* as in [Faisal et al. 2016]. In [Mallea et al. 2011] blank node existential semantics in RDF/S and query languages is explored. When related to urban transportation, several works deal with constraints, but in different context, such that location modeling [Li and Tong 2017], optimization [Yang et al. 2000], using complex network metrics [da Silva et al. 2016, De Bona et al. 2016] or exploratory analysis [Kozievitch et al. 2017].

6. Conclusion

This paper presented a strategy to apply constraints in Curitiba urban transportation database in RDF format. These constraints assist the user during insertion and deletion operations, generating side effects to keep the database consistency. An experimental study demonstrates a shorter execution time and smaller set of side effects when compared with similar strategies. Moreover, the resulting operations are semantically meaningful, since our strategy does not introduce arbitrary facts in order to satisfy the given constraints. In the case of Curitiba urban transportation, our strategy supports the update process by allowing the user to validate the final update set, and decide to either accept of reject it, reducing the chance of introducing inconsistencies in the database.

Acknowledgments. We would like to thank the Municipality of Curitiba, IPPUC and CNPq.

References

[da Silva et al. 2016] da Silva, E. L. C., d. O. Rosa, M., Fonseca, K. V. O., Luders, R., and Kozievitch, N. P. (2016). Combining k-means method and complex network analysis to evaluate city mobility. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1666–1671.

[De Bona et al. 2016] De Bona, A., Fonseca, K., Rosa, M., Lüders, R., and Delgado, M. (2016). Analysis of public bus transportation of a brazilian city based on the theory of complex networks using the p-space. *Mathematical Problems in Engineering*, 2016:1–12.

[Faisal et al. 2016] Faisal, S., Endris, K. M., Shekarpour, S., Auer, S., and Vidal, M.-E. (2016). Co-evolution of rdf datasets. In *Proc. of the 16th Int. Conf. on Web Engineering (ICWE)*, pages 225–243.

[Flouris et al. 2013] Flouris, G., Konstantinidis, G., Antoniou, G., and Christophides, V. (2013). Formal foundations for RDF/S KB evolution. *Knowl. Inf. Syst.*, 35(1):153–191.

[Frommhold et al. 2016] Frommhold, M., Piris, R. N., Arndt, N., Tramp, S., Petersen, N., and Martin, M. (2016). Towards Versioning of Arbitrary RDF Data. In *Proc of the 12th Int. Conf. on Semantic Systems*.

[Halfeld-Ferrari et al. 2017] Halfeld-Ferrari, M., Hara, C. S., and Uber, F. R. (2017). RDF updates with constraints. In *Knowledge Engineering and Semantic Web*, pages 229–245.

[Kozievitch et al. 2016] Kozievitch, N. P., Gadda, T. M. C., Fonseca, K. V. O., Rosa, M. O., Gomes-Jr, L. C., and Akbar, M. (2016). Exploratory analysis of public transportation data in curitiba. In *XXXVI CSBC*, pages 1656–1666. Sociedade Brasileira de Computação.

[Kozievitch et al. 2017] Kozievitch, N. P., Silva, T. H., Ziviani, A., Costa, G., and Lugo, G. (2017). Three decades of business activity evolution in curitiba: A case study. *Annals of Data Science*, 4(3):307–327.

[Li and Tong 2017] Li, R. and Tong, D. (2017). Incorporating activity space and trip chaining into facility siting for accessibility maximization. *Socio-Economic Planning Sciences*, 60:1 – 14.

[Magiridou et al. 2005] Magiridou, M., Sahtouris, S., Christophides, V., and Koubarakis, M. (2005). RUL: A declarative update language for RDF. In *Proc of the 4th Int Semantic Web Conference*, pages 506–521.

[Mallea et al. 2011] Mallea, A., Arenas, M., Hogan, A., and Polleres, A. (2011). On blank nodes. In *Proc of the 10th Int Semantic Web Conference*, pages 421–437.

[Muñoz 2016] Muñoz, E. (2016). On learnability of constraints from RDF data. In *Proc. of the 13th Extended Semantic Web Conference*, pages 834–844.

[Ohori and Ueno 2011] Ohori, A. and Ueno, K. (2011). Making standard ml a practical database programming language. *SIGPLAN Not.*, 46(9):307–319.

[Rabinovitch and Leitman 1996] Rabinovitch, J. and Leitman, J. (1996). Urban planning in curitiba. *Scientific American*, 274(3):46–53.

[Solimando and Guerrini 2013] Solimando, A. and Guerrini, G. (2013). Ontology adaptation upon updates. In *ESWC Satellite Events*, pages 34–45.

[Völkel and Groza] Völkel, M. and Groza, T. In *Proc of the IADIS International Conf on WWW/Internet*.

[Yang et al. 2000] Yang, H., Bell, M. G., and Meng, Q. (2000). Modeling the capacity and level of service of urban transportation networks. *Transportation Research Part B: Methodological*, 34(4):255 – 275.