Edge-labelled graphs and property graphs - to the user, more similar than different

Paul Warren[1][0000-0002-4209-1436] and Paul Mulholland[1][0000-0001-6598-0757]

[1] Knowledge Media Institute, The Open University, Milton Keynes, MK7 6AA, U.K.
{paul.warren, paul.mulholland}@open.ac.uk

Abstract. The two dominant paradigms for graph databases, edge-labelled graphs and property graphs, may appear quite different. Yet, to the user, they have strong similarities. A usability study, comparing RDF-star / SPARQL-star and Cypher, found evidence for only limited differences in preferences between the modelling paradigms. This suggests the possibility of a convergence of the two paradigms; indeed in one case (Stardog) this is already happening. We also found little difference between the paradigms in users' ability to detect valid and non-valid queries. In one specific case, the use of a reverse arrow in Cypher was interpreted significantly more accurately than the ^ symbol in the SPARQL query language; this argues, where possible, for the more intuitive Cypher notation.

Keywords: Graph databases; Cypher; SPARQL-star; Reification; Empirical study.

1 Introduction

Over the past several decades, two graph database formats have arisen: edge-labelled graphs and property graphs. The former has been standardized by the W3C as the Resource Description Framework (RDF), along with an accompanying query language (SPARQL). These have been the subject of considerable theoretical investigation. The latter has been the subject of proprietary standards and implemented and used widely.

Our study sought to:

- compare the ease of use of the two language paradigms;
- identify preferences for alternative semantically equivalent models, and determine whether these preferences differed between the two paradigms;
- determine where there were difficulties in querying models.

We found that the similarities between the paradigms' usage outweighed the differences.

For edge-labelled graphs, we have investigated an extension of RDF, known as RDF-star (RDF*), along with a parallel extension of SPARQL known as SPARQL-star (SPARQL*), using the Blazegraph (https://blazegraph.com/) implementation. These extensions have been proposed to enable reification, and are now the subject of a W3C

Community Group Draft Report) [1][1]. As an example of a property graph language, we took Cypher, developed by neo4j (https://neo4j.com/).

Section 2 describes related work and Section 3 provides an overview of the study. The study was divided into five sections, and Sections 4 to 8 describe each of these. Sections 4 and 5 are not concerned with the extended features of RDF and SPARQL, but compare the more basic features of the two paradigms. The remaining sections are concerned with how RDF* and SPARQL* permit the creation and querying of metadata, and how this compares with the use of metadata in Cypher. Section 6 compares how the two paradigms deal with predicate metadata. Sections 7 and 8 are concerned with metadata about metadata; in the case of Section 7, metadata about node metadata, in the case of Section 8, metadata about predicate metadata. Finally, Section 9 draws some conclusions and makes some recommendations for further development. Warren and Mulholland [2] provide a more detailed discussion.

2 Related work

There have been few studies looking specifically at the usability of graph database languages. Holzschuher and Peinl [3] have compared the readability of Cypher, Gremlin, SQL and a Java API. Based on a readability metric, they concluded that Cypher was the most readable of the four languages. Warren and Mulholland [4] have investigated how accurately and rapidly users can reason about the main SPARQL features. Of relevance to this current study, they found that questions with reverse predicates were answered less accurately and more slowly than analogous questions with forward predicates.

Whilst we are not aware of any usability comparison of the two graph database paradigms, Hartig [5] has produced algorithms to transform between the two paradigms. For edge-labelled graphs, he uses RDF*. For property graphs, he uses his own formalization of the property graph model. However, because of the greater power of RDF*, there are limitations on what can be transformed from RDF* to property graphs.

3 Overview of the study[2]

The study was organized as a between-participants study, i.e. each participant answered questions relating to one of the two paradigms. Participants were contacted through three routes: specialist mailing lists; internal mailing lists within our own organization; and personal contacts. Participants were provided with a tutorial document which

[1] At the time of undertaking the study, we only had available to us the echelon (<< … >>) syntax for quoted triples. The assumption at the time was that embedded triples would be asserted and this was implicit in the study questions. The current assumption is that, when using the echelon syntax, embedded triples are not asserted. There is now an additional syntax, the annotation syntax, which permits embedded triples to be asserted concurrently with their being quoted.

[2] The study was approved by the Open University Human Research Ethics Committee (HREC/3568).

specified all they needed to know for the study, to refer to before and during the study. The study was implemented using an online survey tool. Participants were initially asked to provide some basic demographic information; and also asked about their expertise in graph database languages.

There were then two practice questions; one concerned with data models, the other with queries. The results of these questions were not analyzed. The next five sections of the study were the questions proper. There were two types of questions: modelling questions and querying questions. To minimize participants' time, and for simplicity of analysis, we sought participants' responses to our prepared models and queries. For the modelling questions, participants were given an English-language description of the model, and one or more English-language queries to be answered by the model. Participants were shown several models, in either RDF* or Cypher; and asked which of these models were correct, in the sense of representing the English description and being able to answer the queries. They were also asked to rank the models, with the possibility of ranking models equally. In the following we only report the results of the model ranking. This is partly for reasons of space, but also because we found that, in some cases, participants were classifying a model as incorrect, but then making a comment that they realized the model was technically correct but that they thought it unsuitable to the task, e.g. because it makes difficult future extensions to the data model.

In all but one case, a modelling question was followed by one or more querying questions. Each querying question made use of a correct model, in RDF* or in Cypher, from the previous modelling question. One of the English-language queries from the modelling question was repeated, along with a number of queries in the target query language. Participants were asked to indicate which of these queries corresponded to the English-language query; in some cases there were more than one correct query.

In the reporting of results, model rankings have been normalized to a scale of 0 (lowest) to 1 (highest). The accuracy of assessing queries is also shown on a scale of 0 to 1. In the figures in Section 4 to 8, we present a 95% confidence interval for each statistic. We also use the conventional 95% level when discussing statistical significance. The R language was used for all statistical calculations reported in this paper [6].

Table 1. Expertise of participants in the two conditions

Condition	Language	No knowledge at all	A little knowledge	Some knowledge	Expert knowledge
RDF* / SPARQL*	SPARQL	0%	11.5%	19.2%	69.2%
	Cypher	76.9%	19.2%	3.8%	0%
Cypher	SPARQL	61.1%	11.1%	16.7%	11.1%
	Cypher	38.9%	11.1%	33.3%	16.7%

There were 26 participants for RDF* / SPARQL*, and 18 participants for Cypher; in both cases they were chiefly drawn from Europe and the Americas. Because participants were anonymous, it is not possible to be sure that there was no overlap between the two groups. However, the targeting of participants was designed to reduce the likelihood of this. Table 1 shows the level of expertise in SPARQL and Cypher for

participants in the two conditions, indicating that the level of relevant expertise was greater in the RDF* / SPARQL* condition. Where appropriate we have controlled for this in our statistical analysis.

4 Nodes, literals and reverse predicates

This section describes five questions. The first is a modelling question. The remaining four questions are querying questions which compare the ^ notation in SPARQL with the arrow notation in Cypher. The questions in this section do not make use of the extended features of RDF* or SPARQL*.

Figure 1 shows the modelling question. In model 1, the RDF represented a location as a literal, whilst Cypher used a property value. In model 2, the RDF represented the location with an IRI, whilst Cypher used a node.

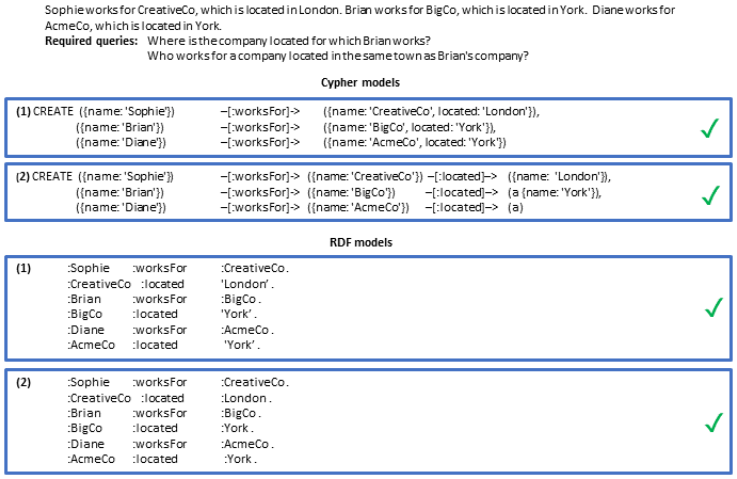

Fig. 1. Modelling question with alternative representations of location information

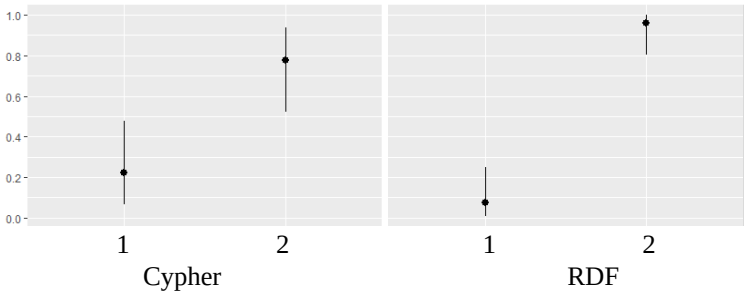

Fig. 2. Preference ratings for modelling question of Figure 1

Figure 2 shows the participants' mean preference ratings with 95% confidence intervals. In both conditions there was a strong preference for model 2, in which the

locations are represented by IRIs or nodes. A two factor analysis of deviance showed a significant difference in preference between the models ($\chi^2(1) = 55.741$, $p < 0.001$) and a significant interaction effect between models and languages ($\chi^2(1) = 5.418$, $p = 0.020$), indicating that the difference in rankings was significantly less for Cypher than for RDF. This is consistent with the greater use of strings to denote entities in property graphs, see a participant's comment below.

The participants' comments mostly confirmed the preference for model 2. A participant commented on RDF model 1 that it "hinders future extensions of the database, as no attributes or relations can be added to strings". Another participant commented "using strings to represent cities is such a bad modelling decision that this needs to be answered as an incorrect model". Similarly, in the Cypher condition a participant commented that "locations are best treated as entities - and thus, nodes - in their own right". On the other hand, in response to RDF model 1, a participant did observe "in Property Graphs it is not uncommon to denote entities by simple strings".

There were four querying questions, all using model 2 from the previous modelling question, and with analogous SPARQL and Cypher queries. Two of the questions used the first, relatively simple English-language query from Figure 1; two used the second, more complex query from Figure 1. At each level of complexity, one of the questions began with a constant ('Brian') and the other with a variable. Figure 3 shows the correct queries for the four questions. For each question, there were also three incorrect proposed queries. For the low complexity questions, these were generated by switching the directionality of the first predicate or edge, the second predicate or edge, or both. For the high complexity questions, they were generated by switching the directionality of the innermost predicates or edges, the outermost, or both innermost and outermost. Figure 4 shows the accuracy of response to each question. Since each question presents four queries, a score of 4 would mean all queries were accurately identified as correct or incorrect. For each of the conditions, the total number of correct responses over the four questions, i.e. out of 16, was computed for each participant. An ANOVA, after controlling for the participants' expertise in the language being studied, showed no significant difference between the languages ($F(1, 37) = 0.0095$, $p = 0.923$).

Where is the company located for which Brian works?

Cypher queries

(LC) MATCH ({name: 'Brian'})–[:worksFor]–>()–[:located]–>(m) RETURN m.name

(LV) MATCH (m)<–[:located]–()<–[:worksFor]–({name: 'Brian'}) RETURN m.name

SPARQL queries

(LC)
```
SELECT ?town
WHERE { :Brian :worksFor / :located  ?town}
```

(LV)
```
SELECT ?town
WHERE { ?town  ^:located / ^:worksFor :Brian}
```

Who works for a company located in the same town as Brian's company?

Cypher queries

(HC) MATCH ({name: 'Brian'})-[:worksFor]->()-[:located]->()<-[:located]-()<-[:worksFor]-(m) RETURN m.name

(HV) MATCH (m)-[:worksFor]->()-[:located]->()<-[:located]-()<-[:worksFor]-({name: 'Brian'}) RETURN m.name

SPARQL queries

(HC)
```
SELECT ?person
WHERE { :Brian :worksFor / :located / ^:located / ^:worksFor  ?person}
```

(HV)
```
SELECT ?person
WHERE { ?person :worksFor / :located / ^:located / ^:worksFor :Brian}
```

Fig. 3. Correct queries based on model 2 of Figure 1

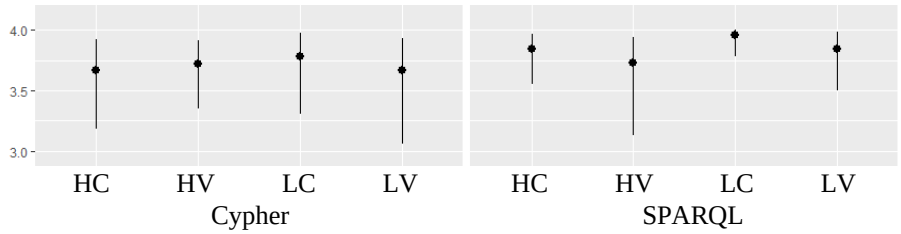

Fig. 4. Accuracy of responses to querying questions shown in Figure 3

LC = low complexity, constant first; LV = low complexity, variable first;
HC =high complexity, constant first; HV = high complexity, variable first.

5 Class hierarchies

The questions in this section were concerned with modelling and querying class hierarchies. They were partially motivated by what seems to be a limitation in the property graph paradigm. As in the previous section, the questions did not make use of the extended features of RDF* and SPARQL*. For the modelling question, we presented two models. The natural way to model hierarchies in RDF is using *rdfs:subClassOf*. Cypher has node labels, which are generally used to indicate class membership. However, there is no natural way to represent hierarchies. One way to create hierarchies in Cypher is to use the *labels()* function to extract the strings representing the labels of a node. Another way is to emulate the use of *rdfs:subClassOf* in RDF. These are the approaches of Cypher models 1 and 2 respectively. For the RDF model 1, we created a model which mixed IRIs and strings, comparable to the use of the Cypher *labels()* function. The corresponding query uses the *STR()* function to extract the string representation of an IRI. For RDF model 2, we followed the *rdfs:subClassOf* approach. However, we did not use *rdfs:subClassOf* but rather a predicate *:subGroupOf*, because participants might not necessarily be familiar with RDFS. The models are shown in Figure 5.

Figure 6 shows the participants' mean preference ratings. For both languages, the majority of participants preferred model 2. This was the case in Cypher, even though the use of labels is the natural way to indicate class membership. An analysis of deviance showed a significant difference between the models ($\chi^2(1) = 59.027$, $p < 0.001$) and a significant interaction effect between models and language ($\chi^2(1) = 6.158$, $p = 0.013$), indicating that the preference was significantly stronger for RDF than for Cypher. This is consistent with *subClassOf* being the accepted approach in RDF. A number of participants criticized RDF model 1 for mixing "resources and literal values for the same concept". For Cypher model 1, one participant noted "Neo4j generally pushes the use of labels for "types", but the path query for recursive subgroups is going to be awkward, requiring an equality condition on a node label and a node's property value."

Fido is a dog. Fred is a baboon. Chirpie is a grasshopper. Dog is a subgroup of mammal. Baboon is a subgroup of primate. Primate is a subgroup of mammal.
Required query:
What are the names of the individuals which are mammals. By this we mean, e.g. *'Fido'*, **not** the names of the groups, e.g. *'Dog'*?

Cypher models

(1)
```
CREATE (:Dog {name: 'Fido'}),
       (:Baboon {name: 'Fred'}),
       (:Grasshopper {name: 'Chirpie'}),
       ({group: 'Dog'})      -[:subGroupOf]-> (m {group: 'Mammal'}),
       ({group: 'Baboon'})   -[:subGroupOf]-> (p {group: 'Primate'}),
       (p)                   -[:subGroupOf]-> (m)
```
✓

(2)
```
CREATE ({name: 'Fido'})      -[:typeOf]->      (d {group: 'Dog'}),
       ({name: 'Fred'})      -[:typeOf]->      (b {group: 'Baboon'}),
       ({name: 'Chirpie'})   -[:typeOf]->      ({group: 'Grasshopper'}),
       (d)                   -[:subGroupOf]-> (m {group: 'Mammal'}),
       (b)                   -[:subGroupOf]-> (p {group: 'Primate'}),
       (p)                   -[:subGroupOf]-> (m)
```
✓

RDF models

(1)
```
:Fido     :typeOf      'Dog' .
:Fred     :typeOf      'Baboon'.
:Chirpie  :typeOf      'Grasshopper' .
:Dog      :subGroupOf  :Mammal.
:Baboon   :subGroupOf  :Primate.
:Primate  :subGroupOf  :Mammal.
```
✓

(2)
```
:Fido     :typeOf      :Dog .
:Fred     :typeOf      :Baboon.
:Chirpie  :typeOf      :Grasshopper .
:Dog      :subGroupOf  :Mammal.
:Baboon   :subGroupOf  :Primate.
:Primate  :subGroupOf  :Mammal.
```
✓

Fig. 5. Modelling question illustrating class hierarchies

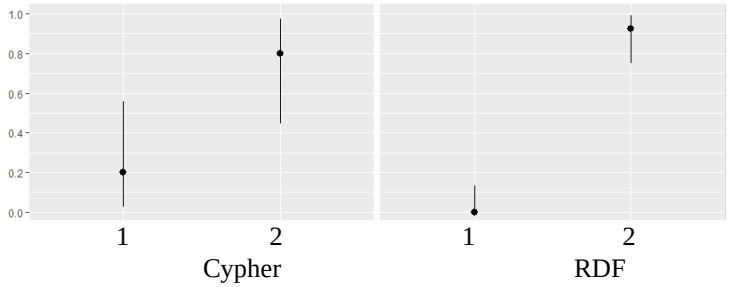

Fig. 6. Preference ratings for modelling question of Figure 5

There were two querying questions, making use of each of the two models, and the English-language query from Figure 5. Figure 7 shows the queries appropriate to model 1. For both conditions, only query 1 is correct.

Cypher queries

(1)
```
MATCH (x), (y)-[:subGroupOf *]->({group: 'Mammal'}) WHERE y.group IN labels(x) RETURN x.name
```
✓

(2)
```
MATCH (x)-[:subGroupOf *]->({group: 'Mammal'}) RETURN x.name
```
X

(3)
```
MATCH (x) WHERE 'Mammal' IN labels(x) RETURN x.name
```
X

SPARQL queries

(1)
```
SELECT ?name
WHERE { ?name :typeOf ?groupname . ?group :subGroupOf+ :Mammal . FILTER (CONTAINS(STR(?group), ?groupname))
```
✓

(2)
```
SELECT ?name
WHERE { ?name :subGroupOf+ :Mammal }
```
X

(3)
```
SELECT ?name
WHERE { ?name :typeOf+ ?groupname . FILTER (CONTAINS(STR(:Mammal), ?groupname)) }
```
X

Fig. 7. Queries based on model 1 of Figure 5

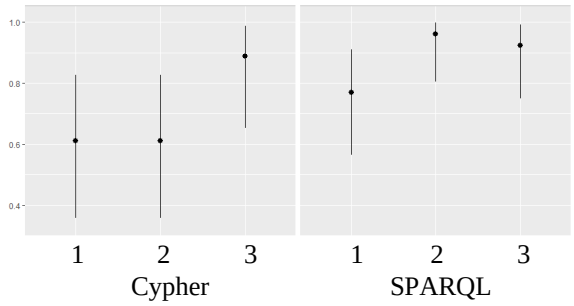

Fig. 8. Accuracy of responses to querying questions shown in Figure 7

Figure 8 shows the accuracy of response to each question. For queries 1 and 2, accuracy of response for Cypher appears appreciably less than for SPARQL; for both these Cypher queries, accuracy was no greater than chance ($p = 0.240$). To compare the two languages, we calculated the number of accurate responses for each participant for each question. An ANOVA, after controlling for expertise in the language being studied, shown no significant difference in performance between the languages ($F(1,37) = 2.4872$, $p = 0.123$).

Figure 9 shows the queries appropriate to model 2. For both conditions, only query 3 is correct. Figure 10 shows the accuracy of response to each question. Based on the number of correct responses for each participant, an ANOVA, after controlling for expertise, showed no significant difference in performance between the languages ($F(1,37) = 0.2662$, $p = 0.609$). This question was answered much more accurately than the previous question: a mean of 2.89 correct compared with 2.43. A two-sided paired t-test indicated that this difference was significant ($t(43) = 3.3463$, $p = 0.002$).

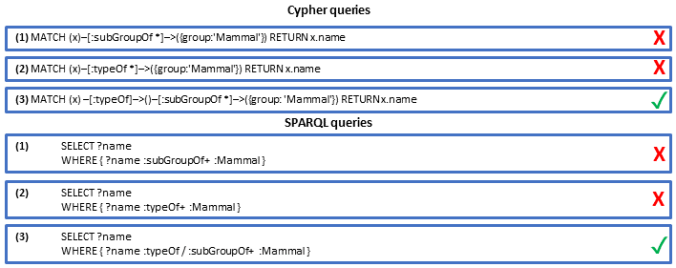

Fig. 9. Queries based on model 2 of Figure 5

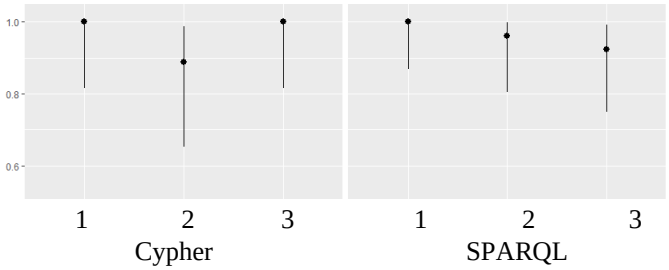

Fig. 10. Accuracy of responses to querying questions shown in Figure 9

6 Metadata about predicates

There were two questions in this section; a modelling question and a querying question. Figure 11 shows the modelling question, with the three alternative models used in the Cypher and the RDF* / SPARQL* conditions. For both conditions, model 1 associates the individuals' roles with the edge or predicate; model 2 associates roles with the individual; model 3 associates roles with the company. Cypher model 2 is incorrect because the model associates multiple roles with the node representing 'Adrian'. Cypher model 3 is similarly incorrect because it associates multiple roles with the node 'TransportCo'.

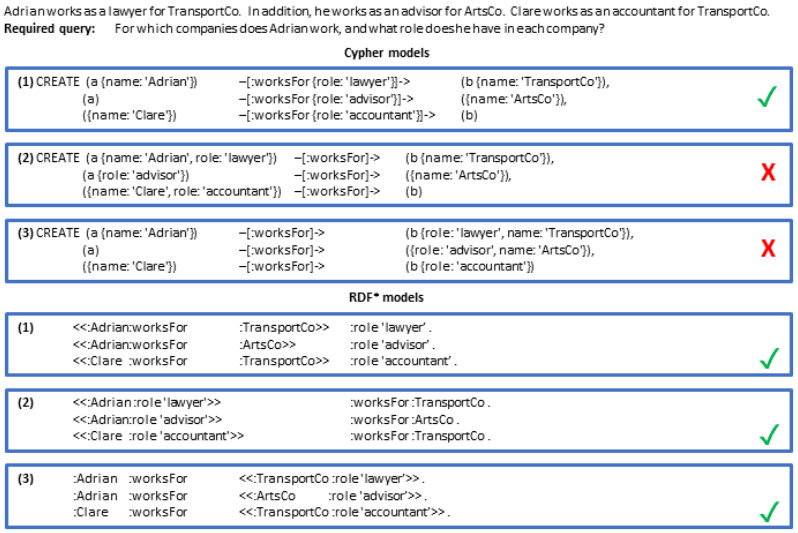

Fig. 11. Modelling question using metadata about predicates

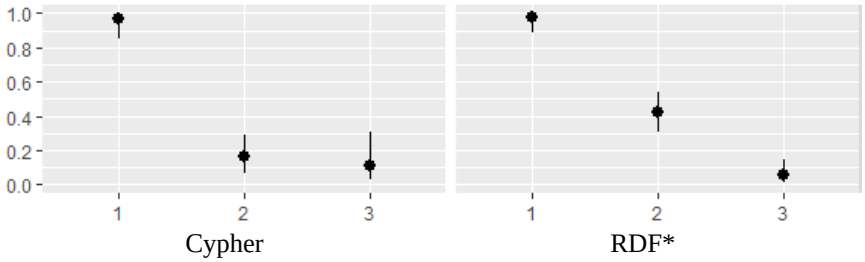

Fig. 12. Preference ratings for modelling question of Figure 11

Figure 12 shows the participants' mean preference ratings. Participants' comments regarding the Cypher models generally confirmed their awareness that models 2 and 3 were incorrect. For the RDF* models, the responses confirmed the preference for model 1; one participant noting that "this one seems most intuitive". Regarding model 3, a participant commented "the model fulfils the query but the model is anyways not a

good representation of the reality, therefore I say it is incorrect". Three participants were unhappy about the use of string literals for roles. A few commented on the relative closeness of each of the models to the original English. In response to model 3, one participant noted "if you were to change the name of :worksFor to :worksAs, this would match the English almost perfectly".

For the querying question, we presented participants with the English-language query from the previous question, along with model 1 from that question and three proposed formulations of the query. These are illustrated in Figure 13. For both conditions, query 1 is incorrect, whilst query 3 has been created by reversing query 2.

Figure 14 shows the accuracy with which the participants responded to the proposed queries. The only appreciable difference between the languages occurs with query 3, where directionality was reversed. For Cypher, 94% of the participants realised this was a correct query, against 85% for SPARQL*. An analysis of deviance, after controlling for expertise in the language being studied, showed a difference in performance between the two languages for this query ($\chi^2(1) = 5.2024$, $p = 0.023$).

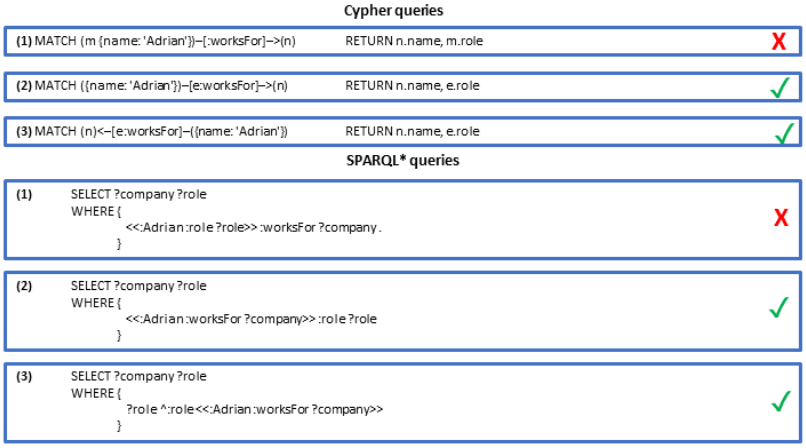

Fig. 13. Proposed queries based on model 1 of Figure 11

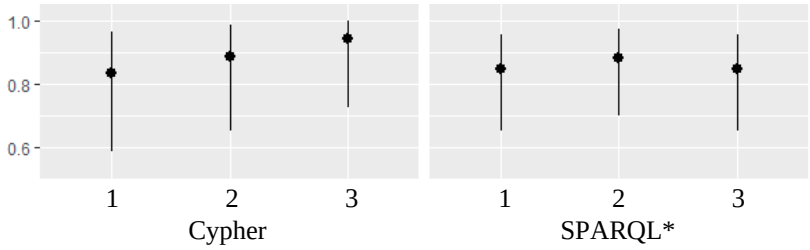

Fig. 14. Accuracy of responses to querying question shown in Figure 13

Warren and Mulholland have previously noted that questions with reverse predicates in SPARQL were answered less accurately and more slowly than analogous questions [4]. We suggest that the difference between SPARQL and Cypher may be due to the more intuitive method of representing predicate directionality in Cypher.

7 Metadata about node metadata

This section contained only one question, a modelling question concerned with metadata about node metadata, specifically attaching date-stamped population values to a node representing a city. The models are shown in Figure 15. For the RDF* model 1, the population is associated with the triple, and analogously for Cypher model 1, the population is associated with the edge. For model 2, it is the date which is associated with the RDF* triple and the Cypher edge. For both RDF* models 1 and 2, the population and date are represented as literals. For RDF* models 3 and 4, the date is an IRI; the difference between the two being that model 3 prepends 'year' before the numeric date. Cypher model 3 illustrates how an unlimited number of population-date pairs can be associated with a node, without creating any additional nodes.

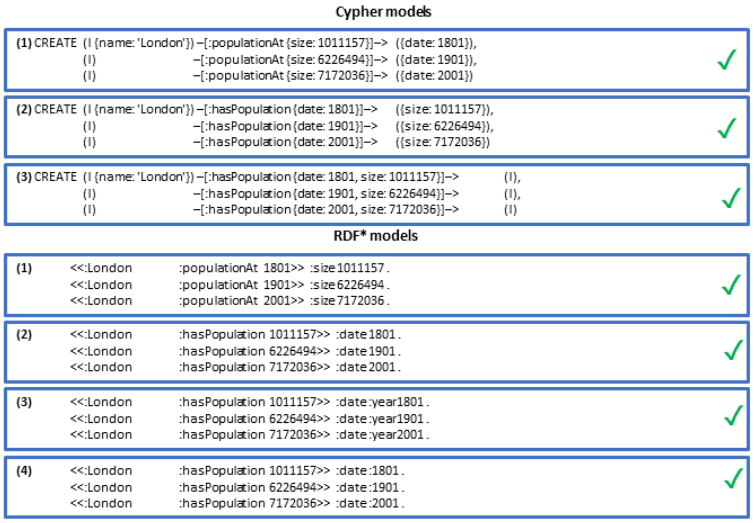

Fig. 15. Modelling question illustrating metadata about node metadata

Figure 16 shows the participants' mean preference ratings. For the RDF* models an ANOVA indicated a significant difference between the rankings ($F(3,100) = 14.527$, $p < 0.001$). A subsequent Tukey HSD analysis indicated a significant pairwise difference between all models, except 2 and 3, and 3 and 4.

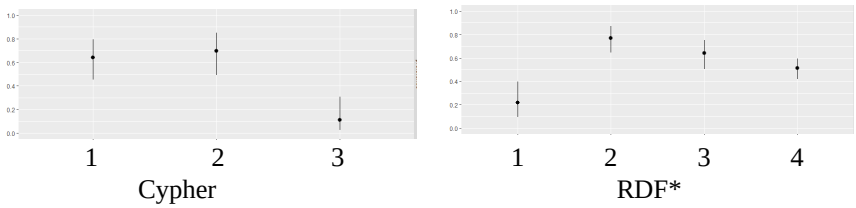

Fig. 16. Preference ratings for modelling question of Figure 15

Comments about the RDF* models varied. Consistent with the low ranking for model 1, one participant said of this model "A date is more logical to be a qualifying statement than the population. I would not use this." On the other hand, two other participants viewed it favourably, commenting: "this seems the natural order" and "This model allows for the simplest query." Two participants explicitly complained about the use of a literal for a date. On the other hand, one commented on model 4: "I think this should be just an integer". The particular choice of predicate names may influence responses. Commenting on model 1, one of the participants said: "but I would not mind experimenting with something like <<London :existsAt :year1801>> :hasPopulation XXXX".

For Cypher, an ANOVA showed a significant difference between the rankings ($F(2,51) = 15.224$, $p < 0.001$). A subsequent Tukey HSD analysis indicated a significant difference between model 3 and the other two, but not between models 1 and 2. One respondent, in criticizing Cypher model 1, commented: "It seems more intuitive that the value of population should be the size, not the year". On the other hand, another participant said of Cypher model 2: "But horrible... don't want a separate node for each number." There is a tension here. It seems more natural to make the object node the population, and to make the year a property. Yet, from a practical viewpoint, making the object node the year avoids arbitrarily creating nodes for particular numbers, and instead creates nodes for years which might be reused. A resolution of this tension was suggested by one participant, who commented "Allowing for composite properties - the design of which is under way - would be the best option." The third Cypher model was clearly less popular. One participant looked favourably on it, but made a suggestion which had some similarity to the comment above about composite properties: "The most correct and the best model, but really hard to read. I'd be arguing for nodes representing the measurement itself (e.g. node of type 'census report' or something similar, containing the date, size and information source.)".

The most striking result from this question is the significant preference amongst the RDF* models for the three which placed the date in the outer triple; compared with the lack of a significant preference between Cypher models 1 and 2. For the RDF* models, it seems that the participants saw the date as "more logical to be a qualifying statement than the population". This did not seem such a concern for the Cypher models.

8 Metadata about predicate metadata

This section contained a modelling question and a querying question. The models made use of doubly-nested RDF* statements, and their analogues in Cypher. Figure 17 shows the modelling question. In the RDF* condition, the question allowed comparing the effect of placing the timing information in the middle and outer triple, and also comparing the positioning of the role and company information. In the Cypher condition, considering the two valid models, the question allowed comparing the effect of placing the role and timing information as properties in the edge and as properties in the object.

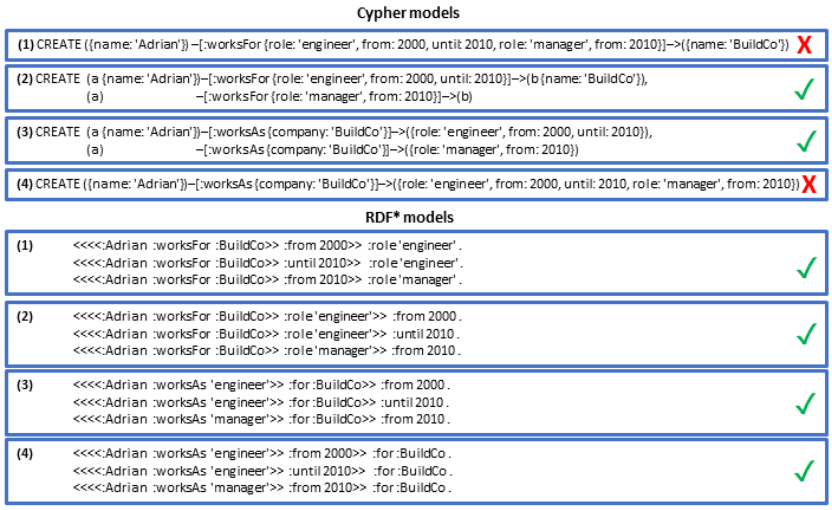

Fig. 17. Modelling question illustrating metadata about predicate metadata

Figure 18 shows the participants' mean preference ratings. For RDF*, the two most preferred models had the timing information in the outer triple. An ANOVA of the rankings for the RDF* models shows a significant difference between the models ($F(3,100) = 21.59$, $p < 0.001$). A subsequent Tukey HSD indicated significant pairwise differences between all models, except models 1 and 3, and 1 and 4.

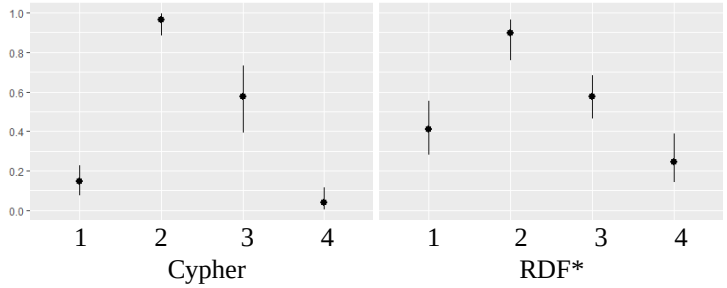

Fig. 18. Preference ratings for modelling question in Figure 17

For Cypher, models 1 and 4 are ranked very low, reflecting the fact that these models are incapable of answering the query. Of the two correct Cypher models, there is a clear preference for model 2, i.e. in which the role and timing metadata are included as edge properties. An ANOVA of rankings for the Cypher models shows a significant difference between the models ($F(3, 68) = 66.8$, $p < 0.001$). Moreover, a subsequent Tukey HSD analysis shows a significant difference in ranking between all the models, except models 1 and 4, i.e. the two incorrect models. The significant difference between the two correct models indicates a firm preference for placing the metadata as properties in the predicate.

The querying question used model 2 from the previous question, along with the English-language query shown in Figure 17. The proposed queries are shown in Figure 19. For both conditions, question 3 was correct and questions 1 and 2 are distractors. In this case, it was not possible to create analogous distractors for the two conditions.

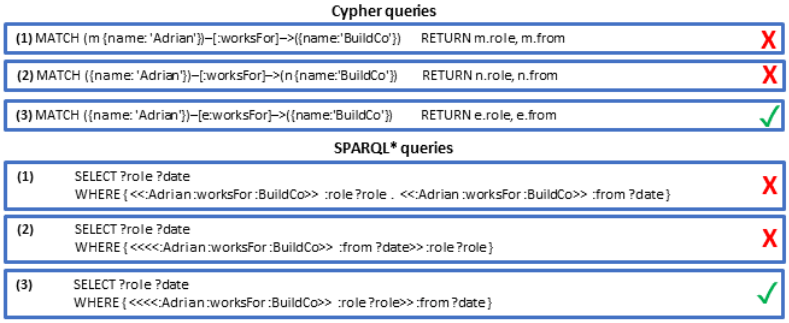

Fig. 19. Queries based on model 2 of Figure 17

Figure 20 shows the accuracy of response to each question. The lack of equivalence between the distractors makes an overall comparison between the languages difficult. It is safer to compare the two correct queries, query 3. A logistic analysis of deviance, after controlling for the participants' expertise, showed no significant difference in performance between the languages ($\chi^2(1) = 1.2953$, $p = 0.255$).

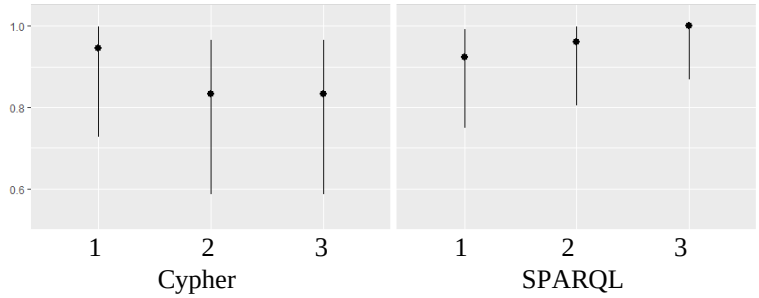

Fig. 20. Accuracy of responses to querying questions shown in Figure 19

9 Conclusions and future directions

In Section 1 we set out three objectives: compare ease of use of the two paradigms; identify modelling preferences; and identify any particular difficulties with queries.

With one exception, described in Section 6, there was no evidence that one of the language paradigms was significantly easier than the other. The one exception suggests that, for the inversion of a predicate, the approach of Cypher may be preferable to the approach of SPARQL. However, we found no such effect for the questions in Section 4. Possibly, the benefits of Cypher's approach become apparent in complex situations.

Generally, where models were analogous between the two paradigms, there was an agreement in ranking, underlining our thesis that the differences between the paradigms are not so great. In Section 4, there was agreement that cities should preferably be

treated as nodes, rather than literals; perhaps to allow the possibility of making further statements about these cities, or representing an aversion to string comparisons in the required queries. A desire to avoid string manipulations showed itself in Section 5, where both sets of participants showed the same preference for constructing class hierarchies, despite the label in Cypher being a natural way to represent classes. This argues for support for basic query-time reasoning, e.g. with class hierarchies. In Section 8, there was agreement about the way the metadata should be ordered, perhaps corresponding to a natural way of thinking. The one exception to this agreement was in Section 7, where there was a significant difference between two RDF* models, but not between the analogous Cypher models.

When we consider the identification of valid and non-valid queries, we find that both sets of participants generally responded with a high degree of accuracy. The exception to this was in Section 5, where the query required string manipulation.

Turning to consider future directions, we suggest that, whilst the worlds of open Web and closed commercial applications may continue to have different needs, there is scope for bringing the paradigms closer together. Stardog (https://www.stardog.com/) has illustrated this, by implementing RDF* and SPARQL* with the syntax used in this paper, but also with a semantically equivalent syntax which is similar to Cypher. Thus, the following two lines are equivalent:

<<:Pete a :Engineer>> :since 2010 .

:Pete a { :since 2010 } :Engineer .

There may also be benefit from experimenting with more radical approaches. For example, Vaticle (https://vaticle.com/) use the hypergraph data model. Whereas an edge in an ordinary graph connects two nodes, a hyperedge in a hypergraph can connect any number of nodes. Thus, n-ary relations are a native feature. Hyperedges can also connect other hyperedges, extending the capability in RDF to use a predicate as the subject or object of another predicate. Vaticle's approach also supports query-time reasoning, through the ability to create hierarchies of entities, relations and attributes.

References

[1] O. Hartig, P. Champin, G. Kellogg, and A. Seaborne, 'RDF-star and SPARQL-star Draft Community Group Report'. Accessed: Aug. 07, 2022. [Online]. Available: https://w3c.github.io/rdf-star/cg-spec/editors_draft.html

[2] P. Warren and P. Mulholland, 'Edge Labelled Graphs and Property Graphs; a comparison from the user perspective', *ArXiv Prepr. ArXiv220406277*, 2022.

[3] F. Holzschuher and R. Peinl, 'Querying a graph database–language selection and performance considerations', *J. Comput. Syst. Sci.*, vol. 82, no. 1, pp. 45–68, 2016.

[4] P. Warren and P. Mulholland, 'A comparison of the cognitive difficulties posed by SPARQL query constructs', presented at the EKAW2020, 2020.

[5] O. Hartig, 'Reconciliation of RDF* and property graphs', *ArXiv Prepr. ArXiv14093288*, 2014.

[6] R Core Team, *R:* A language and environment for statistical computing. *R Foundation for Statistical Computing*, Vienna, Austria. 2013.