being able to use the LUBM data generator, we extended the DL-Lite$_\mathcal{R}$-version of LUBM in two directions:

(1) We added 26 concept inclusions, many of which have existential restrictions on the right-hand side, to generate a more interesting anonymous part of canonical models. A complete list of these CIs can be found in Appendix B of the full version of this paper.

(2) With reasonable effort, it does not seem possible to significantly increase the size of LUBM (to hundreds or thousands of concepts) while retaining a careful modeling. One particularly unrealistic aspect of LUBM and a striking difference to more comprehensive ontologies is its limited concept hierarchy, where each concept has only very few subconcepts. To alleviate this shortcoming, we added subconcepts to each of the LUBM concepts Course, Department, Professor, and Student by introducing subject areas, such as MathCourse, BioCourse, and CSCourse for courses, MathProfessor, BioProfessor for professors, etc.

We call the resulting TBox LUBM$_n^\exists$ with n indicating the number of subconcepts introduced in Point 2 above (20 by default). For example, LUBM$_{20}^\exists$ contains 106 concept names and 27 role names.

To generate ABoxes, we use the LUBM Data Generator (UBA) version 1.7, modified so as to complement our modifications to the TBox. Specifically, the original UBA generates data that is complete w.r.t. existential restrictions in the LUBM ontology: it produces ABoxes $\mathcal{A}$ such that for every assertion $A(a) \in \mathcal{A}$ and CI $A \sqsubseteq \exists R$ (and $A \sqsubseteq \exists R.B$) in LUBM$_n^\exists$, there is already an r-successor of a in $\mathcal{A}$. Our modifications introduce a controlled amount of incompleteness: the modified data generator takes a probability p as a parameter and, in selected parts of the data, drops generated role assertions with probability p. More information can be found in Appendix C of the full version. The second modification of the data generator is linked to the subconcepts introduced in Point 2 above. Whenever the original generator produces an instance a of Student, the new generator randomly chooses a value between 1 and n and generates an assertion for the i-th subject, Subj$i$Student(a); similarly for Course, Department, and Professor.

The main aim of our experiments is to show that our approach is feasible on realistic ontologies, data, and queries. Additionally, we also provide a preliminary comparison with the query rewriting approach, using the Requiem tool for producing those rewritings [10]. We use 11 queries, six of which we have handcrafted specifically for our experiments and five originating from the evaluation of Requiem presented in [10]. The latter queries are extremely simple and do in most cases neither pose a serious challenge for the filtering approach nor for pure rewriting. The former are shown in Figure 4. Note that q_3 is very similar to the query discussed in Examples 3, 4, and 7; and is designed in such a way that spurious cycles in the anonymous part of canonical models produce spurious matches that have to be filtered out. Query q_2 is essentially the query discussed in Example 6 and is designed to stress-test the filtering approach: based on the data generation scheme, it is expected to produce a very large number of spurious answers.

```
q1(x,y) <- Student(x), takesCourse(x,z), Course(z), teacherOf(y,z),
           Faculty(y), worksFor(y,u), Department(u), memberOf(x,u)
q2(x,y) <- Subj3Student(x), Subj4Student(y),
           takesCourse(x,z), takesCourse(y,z)
q3(x)   <- Faculty(x), degreeFrom(x,y), University(y),
           subOrganizationOf(z,y), Department(z), memberOf(x,z)
q4(x,y) <- Subj3Department(x), Subj4Department(y),
           Professor(z), memberOf(z,x), publicationAuthor(u,z),
           Professor(v), memberOf(v,y), publicationAuthor(u,v)
q5(x)   <- Publication(x), publicationAuthor(x,y), Professor(y),
           publicationAuthor(x,z), Student(z)
q6(x,y) <- University(x), University(y), memberOf(z,x), Student(z),
           memberOf(u,y), Professor(u), advisor(z,u)
```

Fig. 4. Queries $\mathtt{q}_1$ to $\mathtt{q}_6$.

Universities	CA	CA (compl)	RA	RA (compl)	Inds	Inds (compl)
10	373K	636K	593K	1.3M	201K	201K
25	984K	1.6M	1.5M	3.6M	528K	528K
50	1.9M	3.3M	3.1M	7.2M	1M	1M
75	3M	5.1M	4.7M	10.9M	1.6M	1.6M
100	4M	6.8M	6.3M	14.6M	2.1M	2.1M
125	5M	8.5M	7.9M	18M	2.7M	2.7M
150	6M	10.1M	9.5M	21.8M	3.2M	3.2M
200	8M	13.5M	12.6M	29M	4.3M	4.3M

Fig. 5. Number of concepts, roles, and individuals in original and completed ABoxes.

5.2 Results

We report on three experiments, in each experiment varying a different parameter: in experiment one, we vary the size of the ABox via the number of universities generated by the data generator; in experiment two, we vary the degree of incompleteness of the data; and in experiment three, we vary the number of subclasses, i.e., the parameter n of the ontology $\text{LUBM}_n^{\exists}$. All experiments were carried out on a Linux (3.2.0) machine with a 3.5Ghz quad-core processor and 8GB of RAM, using IBM DB2 version 9.7.5.

The size of the test data is detailed in Figure 5 and query execution times for the first experiment are reported in Figure 7. Here we use 5% incompleteness of the data and $n = 20$ subclasses in $\text{LUBM}_n^{\exists}$. The orange curves are for the combined approach with filtering while the blue ones indicate the pure rewriting approach for those cases in which Requiem succeeded generating a rewriting. Using the combined approach with filtering, all queries were answered within very reasonable time. To better understand the results, it is interesting to consider the number of spurious and valid answers for each query shown in Figure 6 for the 200 universities experiment. Query $\mathtt{q}_2$, designed specifically to stress-test the filter, as expected produces a huge number of spurious answers. Indeed, the

	q_1	q_2	q_3	q_4	q_5	q_6	req_1	req_2	req_3	req_4	req_5
spurious answers	2	28M	2	24K	0	0	0	22K	0	163K	0
valid answers	4.6M	0	0	0	8.3M	0	0	410K	48K	137K	0

Fig. 6. Number of answers for 200 universities, 5% incompleteness, 20 subclasses.

comparably long execution time of this query appears to be mainly due to the fact that DB2 has to handle a large number of spurious answers before the filtering takes place, and not to a poor performance of the filter itself. The execution times of q_1 and q_5 can also be explained by a large number of answers. Note that the number of filter calls is actually the sum of the numbers of answers, both spurious and valid. Also note that, in principle, it is possible to avoid an extremely large number of spurious answers in q_2 (and any other query) at the cost of slightly increasing the size of the canonical model: duplicate the anonymous part of the canonical model so that no two individuals in the original ABox 'share' an anonymous part of the canonical model. Analyzing this further in experiments is left for future work.

Experiments two and three are reported about in Figures 8 and 9. Here, we only tested the filtering approach. In both cases, we use 100 universities. In experiment two, the number of subclasses is fixed to 20 while in experiment three, the degree of incompleteness is fixed to 5%. In general, the degree of incompleteness has virtually no effect on the execution time of queries. Again, q_2 is an exception as the number of spurious answers increases dramatically from 3M for 1% incompleteness to 125M for 20% incompleteness. The number of subclasses also has essentially no effect on query execution times (in contrast to the pure query rewriting approach for which a non-trivial number subclasses can dramatically increase the size of the rewritten query). Note that the execution time of q_2 becomes shorter with an increasing number of subclasses because the number of spurious matches decreases from 6.5M for 5 subclasses to 211K for 100 subclasses: this is due to the atoms Subj3Student and Subj4Student in q_2 and the fact that the number of assertions for these two concepts decreases as the number of subject areas increases.

6 Conclusion

We have modified the combined approach to OBDA by replacing the query rewriting part with a filtering technique. This step is natural from an implementation perspective and allows to circumvent an exponential blowup of the query. Based on experiments with an improved version of the LUBM ontology, we have demonstrated the scalability of our approach.

As future work, we plan to extend the combined approach with filtering to other description logics for which, until now, it is unknown how to avoid an exponential blowup. For example, we believe that polytime filtering is possible for the extension of $\mathcal{EL}$ with transitive roles, as found in the OWL2 EL profile. It would also be interesting to better understand the impact of modifying the canonical model on query answering, both from a theoretical and from a

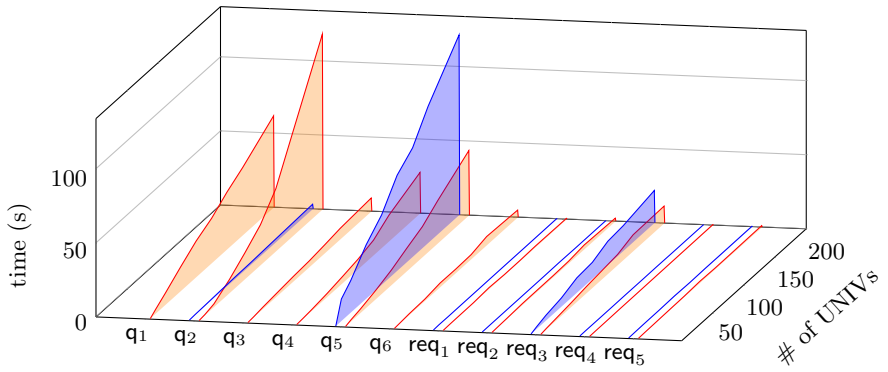

Fig. 7. Query run times for varying numbers of Universities.

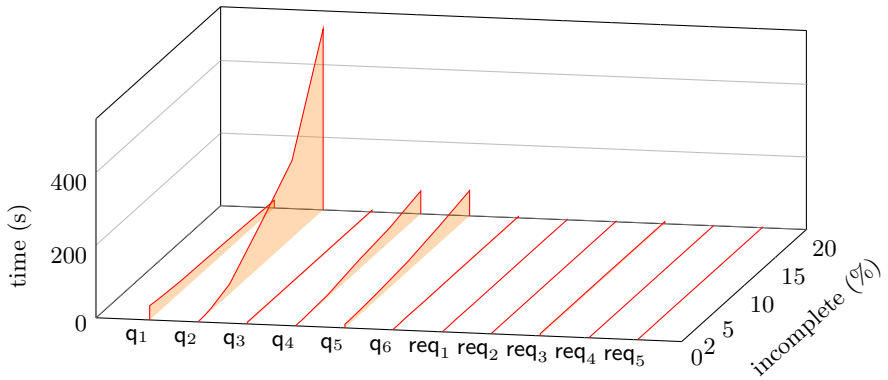

Fig. 8. Query run times for varying incompleteness (in %).

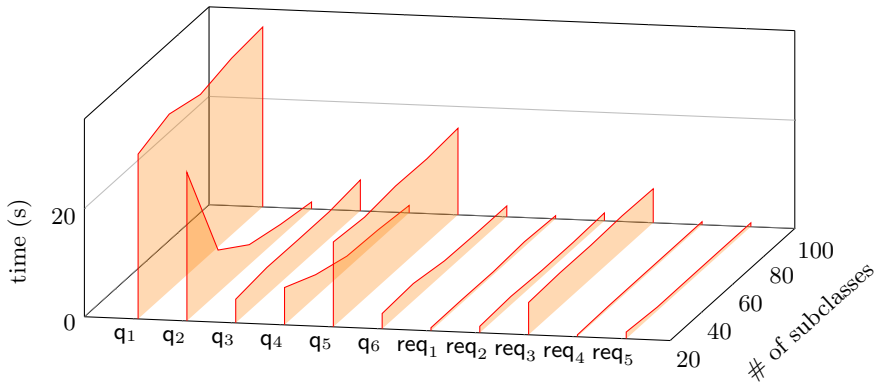

Fig. 9. Query run times for varying number of subclasses.

practical perspective. For example, we do not know whether the more natural canonical model obtained by identifying all individuals $c_{R,0}$ and $c_{R,1}$ admits polytime filtering. Moreover, as discussed above it is conceivable that versions of the canonical model that are *less* economic regarding individual reuse result in better runtime in practice.

We also plan to compare the performance of our approach more thoroughly with the performance of pure query rewriting, using other state-of-the-art query rewriting tools such as Quest [12], Presto [13], OWLgres [14], CLIPPER [4]. In this context, it is interesting to note that promising new optimization techniques have recently been developed in [11] and implemented in the Quest system. While some of them (such as the exploitation of ABox integrity constraints) aim specifically at the query rewriting approach, others (such as semantic indexing) can easily be combined with the filtering approach proposed in this paper.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
2. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: AAAI (2011)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
4. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In: AAAI (2012)
5. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Towards practical query answering for Horn-$\mathcal{SHIQ}$. In: Description Logics (2012)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
7. Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: ICALP (2). pp. 263–274 (2012)
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR (2010)
9. Lutz, C., Wolter, F., Toman, D.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database systems. In: IJCAI. pp. 2070–2075 (2009)
10. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: International Semantic Web Conference. pp. 489–504 (2009)
11. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: KR (2012)
12. Rodriguez-Muro, M., Calvanese, D.: Quest, an OWL 2 QL reasoner for ontology-based data access. In: OWLED (2012)
13. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR (2010)
14. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: OWLED (2008)
15. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR (2012)