A Schematic Analysis on Selective-RDF Database Stores

Sharmi Sankar [1], Awny Sayed [1,2], Jihad Alkhalaf Bani-Younis[1]
[1] College of applied Sciences, Ibri, Sultanate of Oman, [2] Permanently Faculty of Science, Minia University, Egypt

ABSTRACT
RDF has gained great interest in both academia and industry as an important language to describe graph data. With the increasing amount of RDF data which is becoming available, efficient and scalable nowadays has become a challenge to achieve the semantic web vision. The RDF model has attracted the attention of the database community and researchers to propose various methods to store and query the RDF data efficiently. However, current RDF database suffer from several problems, like, poor performance behavior for querying RDF data. This paper provides a comparative analysis made on selective RDF databases storages. It provides a precise study on the various means of having a persistent storage and access of RDF graphs. Recently there has been a major development on initiatives in query processing, access protocols and triple-store technologies. In the evaluation the use of a non- memory and a non-native store Sesame, a native store Allegro graph and Jena API a main-memory based RDF storage system, specifically designed to support fast semantic association discovery. The framework and applications with the ability to store and to query RDF data are analyzed and investigated. Moreover, this paper gives an overview of the features of techniques for storing RDF data and the main purpose of study is to find suitable storage system to store RDF data.

Keywords: W3C, API, RDF, RDFS, Native Store, OWL, SPARQL, DAML, OIL, API, SDB/TDB.

1. INTRODUCTION – RDF
The Resource Description Framework (RDF) is a standard data model for describing machine-readable information in the emerging Semantic Web [24]. An RDF data set is a collection of statements, called *triples*, of the form (*S*, *P*, *O*) where *S* is a subject, *P* is a predicate (also called property) and *O* is an object. Each triple states the relation (represented by its predicate) between its subject and object. A set of triples can be represented as a labeled directed graph, with nodes representing subjects and objects and labeled edges representing predicates, connecting subject nodes to object nodes.

A triple store is a framework used for storing and querying RDF data [6]. The number of triple stores has been considerably increased from Jena and Sesame in the early 2000s to YARS2, Jena TDB, Jena SDB, Virtuoso, AllegroGraph, BigData, Mu lgara, Sesame, Kowari, 3Store and RDF Gateway. Among these some like Garlik and YARS2 are not distributed [5]. A few like AllegroGraph are commercially available. The others are open sources. Majority of the efforts were laid in to check on the freely available open source triple stores and hence the choice of Allegrograph, Sesame, and Jena API.

Triple stores are basically divided into 3 categories based on the architecture of their implementation.
* In-memory
* Native
* Non-memory and non-native.

The first category of triple stores, the in-memory triple stores does store the RDF graph in main memory. These stores also have efficient reasoners available and help to solve the issues on performing inferences on persistent RDF stores, which is complex to perform otherwise [15]. The second category of triple stored, the native store provide persistent storage with their own implementation of the databases, example: Virtuoso, Mulgara, AllegroGraph, Garlik JXT [28]. It provides support for transactions with their own SQL compilers and generally relies on their own procedure language. Recently native triple stores due to their superior load times and ability to be optimized for RDF have gained popularity. The third category of triple stores, the non -native non-memory triples stores persistent storage systems are set up to run on third party databases for eg. Jena SDB which can be coupled with almost all relational databases like MySQL, PostsgreSQL, Oracle so on and so forth [5,29].

In the evaluations a non-memory and non-native store Jena SDB, a native store –Sesame native which has an API to provide fine level access and Jena API native stores are precisely analyzed based on the architecture/design and other factors.

2. RDF DATA REPRESENTATIONS, STORAGE APPROACH
There are several approaches for storage of RDF data. The RDF data representations are in various formats as listed below [1].

- Notation 3 (N3) is a very complex language in order to store RDF-Triples, which was issued in 1998.
- N-Triples a recommendation of W3C, was launched in the year 2004.It is a subset of N3 in order to reduce the complexity involved with it.
- RDF Triple Language (Turtle) enlarges the expressiveness of N-Triples.
- RDF/XML a XML syntax for representing RDF-Triples [13].

This paper focuses on three fundamental storage approaches that are taken into consideration at present for the comparative analysis and they are:

- **In-memory storage:** It allocates a certain amount of the available main memory to store the given RDF data. This approach is restricted for storage and can only store a few RDF data [12].
- **Native storage:** It saves RDF data permanently on the file system [13].

- **Non-Native and Non-Memory storage:** Relational database storage makes use of relational database systems to store RDF data permanently. Unlike the native storage, this approach relies on research results in the database domain (e.g., indices or efficient processing) [12, 13].

3. FEATURES OF PROPOSED RDF STORAGE

Efficient storage of RDF data is plausible only when the appropriate physical organization techniques are applied such as triple table, property table to store it [25]. This may lead to at most efficiency, Scalability and Robustness. The characteristics of the selected RDF storage are tabulated below in Table 1.

Table 1. Features of proposed RDF storage

Store	Storage scheme		Storage support			Query language
	TT	PT	D	F	M	
Jena API	X				X	SPARQL / RDQL
Allegro graph	X			X		SPARQL
Jena		X	X		X	SPARQL / RDQL

TT: TripleTable, PT: PropertyTable, D:Database, F:File, M: Main Memory.

Triple Table : The collections of triples are stored in one single RDF table. The table approach is perhaps the most straightforward mapping of RDF into a RDBMS [26]. Each RDF statement of the form (subject, property, object) is stored as a triple in one large table with a three-column schema.

Limitations: when the number of triples rises, the RDF table may exceed main memory size. RDF triples store scales poorly because complex queries with multiple triple patterns require many self-joins on the single large table [16, 26].

Property Table: RDF tables are physically stored in a representation closer to traditional relational schemas in order to speed up the queries over the triple store [1].

4. RDF DATABASES WITH CONSTRAINT EVALUATION

The framework design and applications with the ability to store and to query RDF data are analyzed and investigated. Further, all the databases shall have the ability to interpret SPARQL queries.

4.1 Evaluation constraints:

Extensibility: It is a very important constraint for the integration of new features, e.g., it optimizes the current working process. One of these features may implement new indices, which accelerate the performance and advance the efficiency of the entire system.

Architectural overview: It offers the basic awareness on the structure of the framework and the used programming language.

Ontology Web Language (OWL): OWL should be provisioned by the databases, as it broadens the semantic expressiveness of RDF and helps for a better inference.

Query languages (supported): It is an additional support of interest, to be aware of other supportive RDF addressing query languages in addition to SPARQL as shown in table 1.

Interpretable RDF data formats: The most interpreted formats should be covered by the frameworks to ensure completeness [13].

4.2 Evaluation of selective RDF databases

This section covers the evaluation of, Jena API, Allegrograph and Sesame following the investigation made upon the above specified evaluation conditions.

4.2.1 Jena API:

The Jena API is capable of storing, accessing and querying large ontologies. It does not use any database backend [18]. The features that lead to select this RDF data store is listed below

- It is easily extensible as it is possible to plug external inference engines into Jena.
- Simple indexing scheme based on elements (Subject, Predicate, Object) are precise.
- RDFS and OWL resoners are available.
- DARPA Agent Markup Language (DAML) support [27, 2].

Jena API is faster as it has a bulk loader. The bulk loader is a faster way to load data into an empty dataset than just using the Jena update operations. It also supports transactions, which is the preferred way to work. It is possible to act directly on the dataset without transaction with a Multiple Reader or Single Writer (MRSW) policy for concurrency access. It also employs caching at various levels, from RDF terms to disk blocks. It is important to flush all caches to make the file state consistent with the cached states because some caches are write-behind so unwritten changes may be held in-memory.

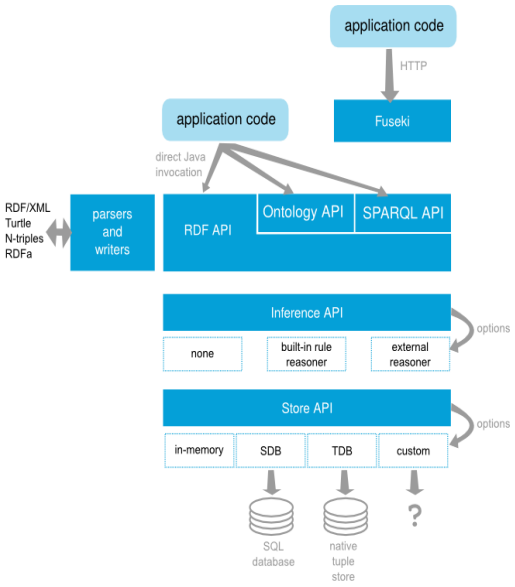

Figure 1 JENA architecture [3].

4.2.2 Allegrograph

The software producers of Allegrograph are Franz's Semantic Technology solutions Company. It is a persistent storage system with their own implementation of databases. Figure 2 illustrates an architectural overview of Allegrograph [4] and the features are as follows.

- It supports transactions such as Commit, Rollback, check pointing.
- It has ability to recover data fast.
- It provides 100% read Concurrency, and full write concurrency
- Ensures dynamic/auto indexing
- Powerful and expressive reasoning/querying.

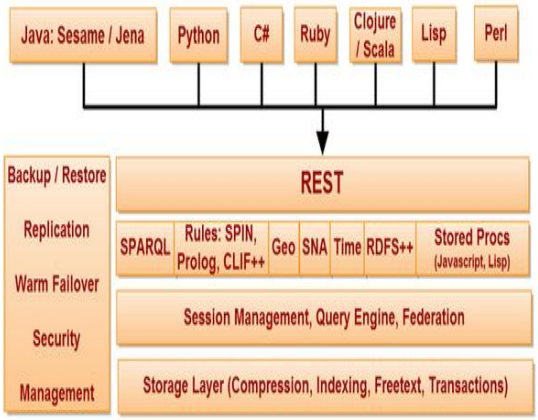

Figure 2 Allegrograph Architecture [4].

AllegroGraph provides a REST protocol architecture, essentially a superset of the Sesame HTTP Client. Franz's staff directly supports adapters for various languages, Sesame Java, Sesame Jena, Python using the Sesame signatures, and Lisp. AllegroGraph is a modern, high-performance, persistent graph database. AllegroGraph uses efficient memory utilization in combination with disk-based storage, enabling it to scale to billions of quads while maintaining superior performance [4]. AllegroGraph's SPARQL, one of the W3C's "interoperable implementations", includes a query optimizer, and has full support for named graphs. It also provides ACID transaction support, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably.

4.2.3 Sesame

The software producer of Sesame37 is Aduna. This company sets the focus of their work in revealing the meaning of information. Sesame was started as a prototype of the EU project On-To-Knowledge39 and is now developed by Aduna in cooperation with NLNet Foundation. Like Jena, Sesames associated license is open source underlying the BSD (Berkeley Software Distribution) license.

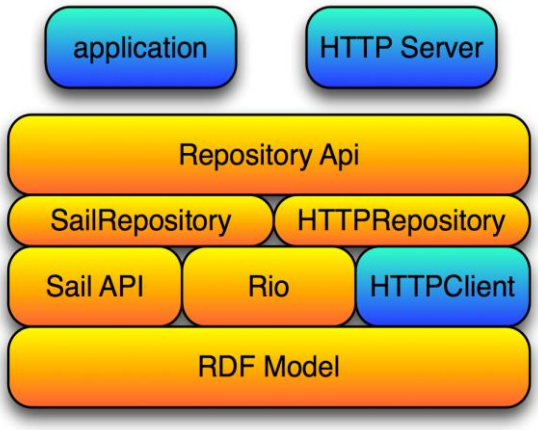

Figure 3 Sesame Architecture

Sesame is able to handle all three in section 2.1 discussed approaches to store RDF data. The RDF Model implements basic concepts about RDF data. The component RDF I/O (Rio) consists of a set of parser and writer for the handling of RDF data. This is for instance used by the Storage and Inference Layer (Sail) API for initializing, querying, modifying and the shutdown of RDF stores. On the topmost layer constitutes the Repository API the main entrance to address repositories. Compared to Sail, which is rather a low level API, the Repository API is the associated high level API with a larger amount of methods for managing RDF data. The HTTP repository is an implementation that acts like a proxy in order to connect to a remote Sesame server via the HTTP protocol. In order to achieve OWL support a Plug-In is available called (Ontology Web Language In Memory) BigOWLIM. It is implemented as a high performance semantic repository for Sesame and packaged as a Sail. Alternatively to SPARQL Sesame is able to interpret the Sesame RDF Query Language (SeRQL) integrated for enhancing the functionality of RQL and RDQL. Sesame offers parsers for various well known RDF formats N3, N-Triples, RDF/XML, Turtle and two new formats TriG43 and TriX [11].

5. PERFORMANCE METRICS

The Lehigh University Benchmark (LUBM) as the first in an eventual suite of benchmarks that would standardize and facilitate such evaluations. The individual metrics initially used by the (Lehigh University Benchmark) LUBM were used as a starting point for the data collection in this evaluation study. Specifically, we collected data on as follows.

Cumulative load time: The time, measured in hours, to load the OWL files describing university departments into the triple-store for a given number of triples. This includes any time spent processing the ontology and source files.

Query response time: Time calculated to respond to the queries. Query response time is measured based on the mean response time of executing each query a number of times.

Query completeness and soundness: A triple-store is complete if it returns all of the correct responses to a query, while a triple-store is sound if it only returns correct responses to a query.

Performance Metrics for the selected RDF data store Jena TDB, Allegrograph version 3.1 and 3.3, Sesame/BigOWLIM with respect to the load and response time parameters are discussed below.

5.1 Experimental results:

5.1.1 Data set (LUBM):

Our work borrows and shares the LUBM database benchmark. The ontology used in the benchmark is called Univ-Bench. Univ-Bench describes universities and departments and the activities that occur at them. The LUBM benchmark was downloaded from [21] . The benchmark is intended to evaluate the performance of those repositories with respect to extensional queries over a large data set that commits to a single realistic ontology [22]. It consists of university domain ontology, customizable and repeatable synthetic data, a set of test queries, and several performance metrics.

5.1.2 Query evaluation:

The LUBM offers 14 queries and readers are referred to appendix for a list of these queries written in SPARQL [9]. Fourteen test queries are chosen to represent a variety of properties including input size, selectivity, and complexity, assumed hierarchy information, assumed logical inference, amongst others. The total number of files read in is 1000. The total number of triples after running the queries is 6,875,705. In the LUBM results below, AllegroGraph's dynamic materialization befell as it's necessary to answer each query [15]. For AllegroGraph version 3.3, loading, indexing and merging required a total of 7 minutes and 50 seconds.

5.1.3 Results and discussion:

Table 2 shows the results of running the LUBM 50 queries with both version 3.1 and 3.3 of AllegroGraph. The total query time for the 14 queries on went from 275.379 seconds in version 3.1 to 3.798 seconds in version 3.3. The results are reported in seconds.

Table 2. Summary (LUBM results Allegrograph)

LUBM Query	# Triples	3.1 Time	3.3 Time
Query 1	4	0	0.007
Query 2	130	2.634	0.33
Query 3	6	0.002	0.006
Query 4	34	0.046	0.03
Query 5	719	3.899	0.055
Query 6	519,842	5.42	1.363
Query 7	67	0.027	0.013
Query 8	7,790	3.371	0.303
Query 9	13,639	254.107	1.245
Query 10	4	0.002	0.01
Query 11	224	0.075	0.01
Query 12	15	3.47	0.025
Query 13	228	0.091	0.014
Query 14	383,730	2.235	0.387

Table 3: Summary (LUBM Queries)

	Sesame/Big OWLIM msec. (results)	Jena	Allegro
Loading time (sec.)	200	260	239
Query 1	2(4)	160	4
Query 2	1 873 (130)	timeout	130
Query 3	1 (6)	215	6
Query 4	4 (34)	51	34
Query 5	6 (719)	585	719
Query 6	257 (519 842)	215	519,842
Query 7	2 (67)	272951	67
Query 8	85 (7 790)	timeout	7,790
Query 9	3 256 (13 639)	timeout	13,639
Query 10	1 (4)	209	4
Query 11	1 (224)	14	224

Query 12	5 (15)	4	15
Query 13	8 (228)	203	228
Query 14	193 (393 730)	220	383,730

Table 3 shows the response on the listed 14 queries and the cumulative load time for all the 3 selective RDF data stores are listed in the above table 2 [23]. The results when compared with respect to the queries reveal that sesame/BigOWLIM outperforms the others. The outcome is visualized as shown in figure 4 with respect to the load time for LUBM benchmark queries and figure 5 that depicts the results on test queries respectively.

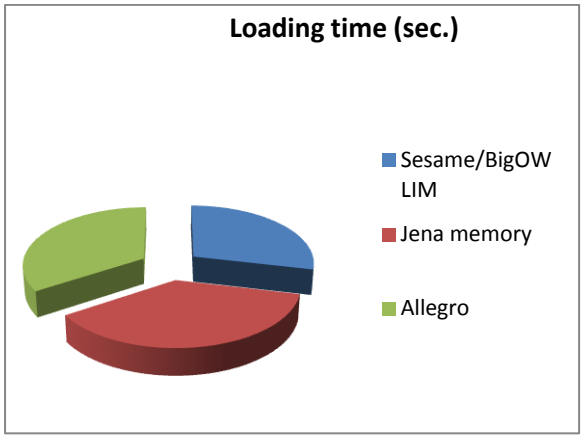

Figure 4 LUBM dataset Load time for Sesame, Jena and Allegrograph.

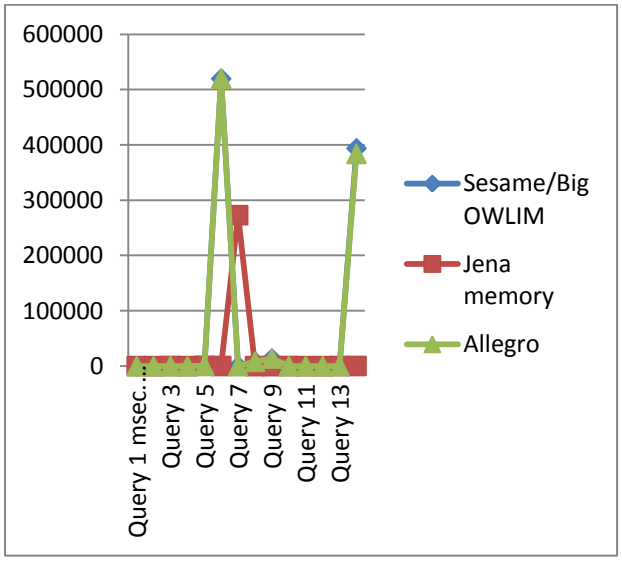

Figure 5 Response on LUBM test queries.

As we observe the above visual representation the sesame/bigOWLIM and allegrograph version 3.3 performance matches very closely despite of the performance with respect to query 14 where they vary slightly.

6. CONCLUSION

With LUBM test queries, Sesame out performs really better than Jena and Allegrograph. The test queries are listed in appendix 1. It is considered that sesame performs better than others only with small datasets. However benchmark has been found comparing Sesame, Allegrograph and Jena using LUBM datasets. Testing systems that expose SPARQL endpoints with realistic workloads of use case motivated queries are listed below in table 2 [8, 9].

This experimentation on these selective RDF data stores to compare the loading and querying performance. First, we discovered that the performance of Jena and Allegrograph is extremely poor as far as the cumulative loading time is concerned. Second, Sesame's load time increase exponentially with the size of data loaded and has the least cumulative load time. Third, as expected we showed that persistent storage systems could handle larger data sizes than memory-based systems, but we were surprised to discover that Sesame memory could handle up to 110 MB of input data. Finally, we found Sesame response was better at different queries [9]. The results of this investigation to make universal pronouncements may not be relative as quality of different benchmarks and selective RDF stores may vary. This investigation has driven us further into a deeper research in practical, scalable reasoning systems and that by examining the relative strengths and weaknesses of different systems and had guided us on how to build a better OWL KBS.

7. FUTURE WORK

In future, a standard benchmark for Ministry of Higher Education(MoHE), (Sultanate of Oman) will be shortly deployed and similar performance measurement with queries against large amounts of RDF data is to be made to ensure the performance on complex critical datasets on the very same selected RDF Stores. The benchmark would aid the evaluation of Semantic Web repositories (MoHE) in an efficient way. The evolution of such benchmarks enables the Omani community to find, share, and combine information more easily on the web.

8. REFERENCES:

[1] A survey of RDF storage approaches David C. FAYE, Olivier CURE,Guillaume BLIN ARIMA Journal, vol. 15 (2012), pp. 11-35.

[2] http://jena.sourceforge.net/inference/index.html

[3] http://jena.apache.org/about_jena/architecture.html

[4] http://www.franz.com/agraph/allegrograph

[5]http://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf

[6] An Evaluation of Triple-Store Technologies for Large Data Stores, Kurt Rohloff, Mike Dean, Ian Emmons, Dorene Ryder and John Sumner.

[7]http://www.franz.com/agraph/allegrograph3.3/agraph3.3_bench_lubm.lhtml

[8] LUBM: A Benchmark for OWL knowledge base systems. Yuanbo Guo, Zheng xiang pan, Jeff Heflin presented on ISWC2004.

[9] Guo, Yuanbo, Pan, Zhengxiang and Heflin, Jeff. LUBM: A Benchmark for OWL Knowledge Base Systems. Web Semantics. 3(2) July 2005. pp.158-182.

[10] Kevin Wilkinson, Carig Sayers, Harumi Kuno , Dave Reynolds, "Efficient RDF storage and retrieve in Jena2 ", Enterprise system and data management laboratory

[11] Jeen Broekstra, Arjohn Kampman, Frank van Harmelen, " Sesame: An architecture for storing and quering RDF data and schema information", Vrije university Amsterdam

[12] David C.Faye, Oliver crue, Guillaume BLIN, "a survey of RDF storage approaches",Arima Journal

[13] Florian Stegmier, Udo Grobner, Mario Doller, Harald Kosch, Gero Baese, "Evaluation of current RDF database solutions", Chair of distributed information systems University of Passau

[14] YingHong Liao, ChuenTsai Sun, "An educational genetic algorithms learning tool", http://www.ewh.ieee.org/soc/es/May2001/14/Begin.htm.

[15] Performance of native SPARQL query processors , UPPSALA University, Shridevika Maharajan.

[16] Resource Description Framework (RDF). http://www.w3.org/RDF/

[17] Y. Guo, Z. Pan, and J. Heflin. Choosing the Best Knowledge Base System for Large Semantic Web Applications. In Proc. of the 13th International World Wide Web Conference (WWW2004) - Alternate Track Papers & Posters, 2004.

[18] S. Alexaki et al. The RDFSuite: Managing Voluminous RDF Description Bases. In Proc. of the 2nd International Workshop on the Semantic Web (SemWeb' 01), in conjunction with the Tenth International World Wide Web Conference (WWW10), 2001.

[19] S. Alexaki et al. On Storing Voluminous RDF Description: The case of Web Portal Catalogs. In Proc. of the 4th International Workshop on the Web and Databases, 2001.

[20] J.J. Carroll and J.D. Roo ed. OWL Web Ontology Test Cases, W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-owl-test-20040210/

[21] http://swat.cse.lehigh.edu/projects/lubm/

[22] http://www.w3.org/wiki/RdfStoreBenchmarking

[23] Tu Ngoc Nguyen, Wolf Siberski ,SLUBM: An Extended LUBM Benchmark for Stream Reasoning.

[24] Eric Miller, An Introduction to the Resource Description Framework, D-Lib Magazine, ISSN 1082-9873, May 1998.

[25] Alisdair Owens, An Investigation into Improving RDF Store Performance,March 2009.

[26] Olivier Cure,David Faye,Guillaume Blin, Towards a better insight of RDF triples Ontology-guided Storage system abilities, Jun 2013.

[27] http://jena.apache.org/documentation/inference/#api

[28] http://www.w3.org/wiki/LargeTripleStores

[29]Vaibhav Khadilkar, Jyothsna Rachapalli,Bhavani Thuraisingham,The University of Texas at Dallas, Semantic Web Implementation Scheme for National Vulnerability Database,2009.

APPENDIX - 1

LUBM benchmark test queries:

Query:1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:GraduateStudent .
?X ub:takesCourse
http://www.Department0.University0.edu/GraduateCourse0}
```

Query:2

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:GraduateStudent .
?Y rdf:type ub:University .
?Z rdf:type ub:Department .
?X ub:memberOf ?Z .
?Z ub:subOrganizationOf ?Y .
?X ub:undergraduateDegreeFrom ?Y}
```

Query:3

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
```

{?X rdf:type ub:Publication .
?X ub:publicationAuthor
http://www.Department0.University0.edu/AssistantProfessor0}

Query:4

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y1, ?Y2, ?Y3
WHERE
{?X rdf:type ub:Professor .
?X ub:worksFor <http://www.Department0.University0.edu> .
?X ub:name ?Y1 .
?X ub:emailAddress ?Y2 .
?X ub:telephone ?Y3}

Query:5

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Person .
?X ub:memberOf <http://www.Department0.University0.edu>}

Query:6

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X WHERE {?X rdf:type ub:Student}

Query:7

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Course .
?X ub:takesCourse ?Y .
<http://www.Department0.University0.edu/AssociateProfessor0>,
ub:teacherOf, ?Y}

Query:8

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Department .
?X ub:memberOf ?Y .
?Y ub:subOrganizationOf <http://www.University0.edu> .
?X ub:emailAddress ?Z}

Query:9

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Faculty .
?Z rdf:type ub:Course .
?X ub:advisor ?Y .
?Y ub:teacherOf ?Z .
?X ub:takesCourse ?Z}

Query:10

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Student .
?X ub:takesCourse
<http://www.Department0.University0.edu/GraduateCourse0>}
```

Query:11

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:ResearchGroup .
?X ub:subOrganizationOf <http://www.University0.edu>}
```

Query:12

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y
WHERE
{?X rdf:type ub:Chair .
?Y rdf:type ub:Department .
?X ub:worksFor ?Y .
?Y ub:subOrganizationOf <http://www.University0.edu>}
```

Query:13

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Person .
<http://www.University0.edu> ub:hasAlumnus ?X}
```

Query:14

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE {?X rdf:type ub:UndergraduateStudent}
```