## Deliverable 1.3.3

**Continuous Report on Performance Evaluation**

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>Month 24, 30/11/2014</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>06/01/2015</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP1 - Requirements, Design, Benchmarking, Component Integration</td>
</tr>
<tr>
<td>Task</td>
<td>T1.3 - Performance Benchmarking and Evaluation</td>
</tr>
<tr>
<td>Type</td>
<td>Report</td>
</tr>
<tr>
<td>Approval Status</td>
<td>Final</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Number of Pages</td>
<td>28</td>
</tr>
<tr>
<td>Filename</td>
<td>D1.3.3_Continuous_Report_on_Performance_Evaluation.pdf</td>
</tr>
</tbody>
</table>

**Abstract:** The purpose of this deliverable is to summarise the performance evaluations performed against the Virtuoso geospatial query and storage engine during year two of the project and comparing with the year one results where possible.

The information in this document reflects only the author’s views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/her sole risk and liability.
### History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>31/12/2014</td>
<td>Initial Draft</td>
<td>Mirko Spasić</td>
</tr>
<tr>
<td>0.2</td>
<td>02/01/2015</td>
<td>Updated Content</td>
<td>Orri Erling</td>
</tr>
<tr>
<td>0.4</td>
<td>04/01/2015</td>
<td>First Review</td>
<td>Hugh Williams</td>
</tr>
<tr>
<td>0.6</td>
<td>06/01/2015</td>
<td>Peer Review</td>
<td>Jens Lehmann</td>
</tr>
<tr>
<td>1.0</td>
<td>06/01/2015</td>
<td>Finalised</td>
<td>Hugh Williams</td>
</tr>
</tbody>
</table>

### Author List

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Name</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGL</td>
<td>Hugh Williams</td>
<td><a href="mailto:hwilliams@openlinksw.com">hwilliams@openlinksw.com</a></td>
</tr>
<tr>
<td>OGL</td>
<td>Mirko Spasić</td>
<td><a href="mailto:mspasic@openlinksw.com">mspasic@openlinksw.com</a></td>
</tr>
<tr>
<td>OGL</td>
<td>Orri Erling</td>
<td><a href="mailto:oerling@openlinksw.com">oerling@openlinksw.com</a></td>
</tr>
<tr>
<td>OGL</td>
<td>Ivan Mikhailov</td>
<td><a href="mailto:imikhailov@openlinksw.com">imikhailov@openlinksw.com</a></td>
</tr>
<tr>
<td>InfAI</td>
<td>Jens Lehmann</td>
<td><a href="mailto:lehmann@informatik.uni-liepzig.de">lehmann@informatik.uni-liepzig.de</a></td>
</tr>
</tbody>
</table>

### Time Schedule before Delivery

<table>
<thead>
<tr>
<th>Next Action</th>
<th>Deadline</th>
<th>Care of</th>
</tr>
</thead>
<tbody>
<tr>
<td>First version</td>
<td>31/12/2014</td>
<td>Mirko Spasić (OGL)</td>
</tr>
<tr>
<td>Second version</td>
<td>02/01/2015</td>
<td>Orri Erling (OGL)</td>
</tr>
<tr>
<td>Third version</td>
<td>04/01/2015</td>
<td>Hugh Williams (OGL)</td>
</tr>
<tr>
<td>Final version</td>
<td>06/01/2015</td>
<td>Hugh Williams (OGL)</td>
</tr>
</tbody>
</table>
Executive Summary

This deliverable details the benchmark results from running the FacetBench program against the Virtuoso SPARQL engine during the second year of the project and comparing with the year one results where possible. New analytical queries were developed for use with the integrated LGD and SNB dataset created during this period for running the benchmarks against. Virtuoso Characteristic Sets (CS) and Geospatial clustering will be applied to the workloads in the coming months.
### Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>Characteristic Set</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>LDBC</td>
<td>Linked Data Benchmark Council</td>
</tr>
<tr>
<td>LGD</td>
<td>Linked Geo Data</td>
</tr>
<tr>
<td>LOD</td>
<td>Linked Open Data</td>
</tr>
<tr>
<td>OSM</td>
<td>Open Street Map</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SNB</td>
<td>Social Network Benchmark</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
</tbody>
</table>
# Table of Contents

1. INTRODUCTION .............................................................................................................................................. 8
   1.1 OUTLINE ............................................................................................................................................. 8

2. DATASET DESCRIPTION .................................................................................................................................... 10
   2.1 LGD MAPPINGS ...................................................................................................................................... 10

3. BENCHMARK RESULTS ...................................................................................................................................... 12
   3.1 VIRTUOSO SPARQL RESULTS ............................................................................................................. 12
   3.2 COMPARISON OF RESULTS FROM THIS AND THE PREVIOUS YEAR ..................................................... 13

4. INTEGRATION OF LGD WITH LDBC SNB ....................................................................................................... 22
   4.1 QUERY EXAMPLES ............................................................................................................................... 24

5. CONCLUSION AND FUTURE WORK .................................................................................................................. 28

6. APPENDIX ...................................................................................................................................................... 29

7. BIBLIOGRAPHY ............................................................................................................................................... 30
List of Figures

Figure 1: Linked Geodata Browser........................................................................................................................................8
Figure 2: Comparison with last year ........................................................................................................................................14
Figure 3: Query execution times exploring Rotterdam...........................................................................................................15
Figure 4: Query execution times exploring Munich..................................................................................................................15
Figure 5: Query execution times exploring Budapest..............................................................................................................16
Figure 6: Query execution times exploring Lisbon..................................................................................................................16
Figure 7: Query execution times exploring Berlin....................................................................................................................17
Figure 8: Query execution times exploring Manchester........................................................................................................17
Figure 9: Query execution times exploring Prague..................................................................................................................18
Figure 10: Query execution times exploring Copenhagen.....................................................................................................18
Figure 11: Query execution times exploring Birmingham......................................................................................................19
Figure 12: Query execution times exploring Warsaw..............................................................................................................19
Figure 13: Deceleration grouped by cities...............................................................................................................................20
Figure 14: Deceleration grouped by queries...........................................................................................................................21

List of Tables

Table 1: Distribution of Geometries ..........................................................................................................................................10
Table 2: Distribution of Predicates ..........................................................................................................................................10
Table 3: Virtuoso SPARQL results .........................................................................................................................................13
Table 4: Number of nodes per city ..........................................................................................................................................14
Table 5: UML model of the SNB schema ............................................................................................................................23
1. Introduction

The primary goal of this report is the summarization of the performance evaluation of the GeoKnow components (mainly Virtuoso) as developed within the second project year, and comparison with the evaluation from the end of the first year.

In the previous deliverables D1.3.1 (GeoKnow Consortium, 2013) and D1.3.2 (GeoKnow Consortium, 2014) we specified the setup and configuration of the GeoKnow Benchmarking laboratory, gave the query descriptions in detail, and presented benchmark results running the FacetBench program against Virtuoso SQL, Virtuoso SPARQL and PostGIS. This benchmark is focused on addressing practical challenges in the Geo Browsing components, as developed by the University of Leipzig (browser.linkedgeodata.org) and emulates heavy drill-down style online access patterns and accessing large volumes of thematic data.

With Virtuoso SPARQL, we will compare scale-out and single server versions, as last year, and compare this year’s results, with previous ones.

1.1 Outline

In Section 2 we describe the dataset used in these experiments, and changes in LGD mappings.

In Section 3 the new benchmarking results are presented, together with comparison against results from previous year.

Integration of LGD with SNB dataset is analyzed in Section 4, as well as analytical queries against this merged dataset.
Section 5 contains some conclusions and future work, while Section 6 is an appendix, providing a link where the benchmark programs can be downloaded from.

References are found in the Section 7.
2. Dataset description

The experiments discussed in this set of deliverables are all on the September 2014 Open Street Map dataset converted to RDF with Sparqlify using the Linked Geodata mappings. This OSM dataset cover the entire planet. The zipped xml file from OSM had 36GB of data, while the size of zipped LGD files in turtle format is 177GB. The detailed description of the dataset can be found in the D1.3.2 (GeoKnow Consortium, 2014), while the current counts are presented in the following table:

<table>
<thead>
<tr>
<th>Type of geometry</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>2424342788</td>
</tr>
<tr>
<td>Polygon</td>
<td>242305128</td>
</tr>
</tbody>
</table>

*Table 1: Distribution of Geometries*

The counts of the top properties in the dataset are as follows (in millions):

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td>8659</td>
</tr>
<tr>
<td><a href="http://linkedgeodata.org/triplify/hasseq">http://linkedgeodata.org/triplify/hasseq</a></td>
<td>2907</td>
</tr>
<tr>
<td><a href="http://linkedgeodata.org/triplify/hasnode">http://linkedgeodata.org/triplify/hasnode</a></td>
<td>2907</td>
</tr>
<tr>
<td><a href="http://linkedgeodata.org/triplify/hasway">http://linkedgeodata.org/triplify/hasway</a></td>
<td>2907</td>
</tr>
<tr>
<td><a href="http://linkedgeodata.org/ontology/changeset">http://linkedgeodata.org/ontology/changeset</a></td>
<td>2669</td>
</tr>
<tr>
<td><a href="http://purl.org/dc/terms/modified">http://purl.org/dc/terms/modified</a></td>
<td>2669</td>
</tr>
<tr>
<td><a href="http://purl.org/dc/terms/contributor">http://purl.org/dc/terms/contributor</a></td>
<td>2669</td>
</tr>
<tr>
<td><a href="http://linkedgeodata.org/ontology/version">http://linkedgeodata.org/ontology/version</a></td>
<td>2669</td>
</tr>
<tr>
<td><a href="http://geovocab.org/geometry#geometry">http://geovocab.org/geometry#geometry</a></td>
<td>2666</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2003/01/geo/wgs84_pos#long">http://www.w3.org/2003/01/geo/wgs84_pos#long</a></td>
<td>2434</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2003/01/geo/wgs84_pos#lat">http://www.w3.org/2003/01/geo/wgs84_pos#lat</a></td>
<td>2434</td>
</tr>
</tbody>
</table>

*Table 2: Distribution of Predicates*

The total count of the dataset is 37bn triples. The dataset is approximately 1.8x the size of the dataset in the previous evaluation, which corresponds to the natural growth of the OSM (OpenStreetMap Project) corpus.

2.1 LGD mappings

The LGD mappings (LinkedGeoData Project, 2011) were modified as follows:

- No intermediate node was present between the subject with the OSM properties and the geometry. In other words, the geometry was a direct property of the OSM node object, as it is in OSM itself.
• Table waynodes from OSM, that contains data on all of those nodeIDs that belong to a given wayID, was exported to RDF using the Sparqlification Mapping Language (SML) in the following way:

```
Create View view_waynodes As
Construct {
  ?wn a lgdm:WayNode .
}
With
  ?wn = uri(lgd:way_node_, ?way_id, '_', ?sequence_id)
  ?p1 = uri(lgd:hasway)
  ?w1 = uri(concat(lgd:way, ?way_id))
  ?p2 = uri(lgd:hasnode)
  ?n = uri(concat(lgd:node, ?node_id))
  ?p3 = uri(lgd:hasseq)
  ?seq = typedLiteral(?sequence_id, xsd:int)
From [[SELECT * FROM way_nodes]];
```

That means that every entity from this table is an instance of WayNode, and has its own wayID, nodeID, and sequence id.

• The Sparqlify export procedure would take too long to export the whole planet, and therefore we made it parallel, in order to get RDF much faster. Also, this procedure only works with pgsimple schema of OSM, but not with the snapshot schema (GeoKnow Consortium, 2013). We loaded the data into PostGIS using wrong schema, so we had to create derived tables:

```sql
CREATE TABLE node_tags AS SELECT a.id node_id, (each(a.tags)).key k, (each(a.tags)).value v FROM nodes a;
CREATE TABLE way_tags AS SELECT a.id way_id, (each(a.tags)).key k, (each(a.tags)).value v FROM ways a;
CREATE TABLE relation_tags AS SELECT a.id relation_id, (each(a.tags)).key k, (each(a.tags)).value v FROM relations a;
CREATE INDEX idx_node_tags_node_id ON node_tags USING btree (node_id);
CREATE INDEX idx_way_tags_way_id ON way_tags USING btree (way_id);
CREATE INDEX idx_relation_tags_relation_id ON relation_tags USING btree (relation_id);
```
3. Benchmark results

In the first deliverable D1.3.1 (GeoKnow Consortium, 2013), we presented the benchmark metrics, and the reporting template containing all the metrics. In this section, we give the benchmark results obtained at the end of second year. The tested system is Virtuoso SPARQL.

The test platform is two machines with dual Xeon E5-2630 and 192GB RAM each, QDR InfiniBand interconnect. The Virtuoso data is on 2x4 7200 rpm commodity disks.

3.1 Virtuoso SPARQL results

In this section, we give the results of benchmark run over RDF data loaded in Virtuoso on a cluster of computers (Table 3). Virtuoso was run in cluster mode where one logical database (with Linked Geodata) is served by a collection of server processes (in our case, there was 4 of them) spread over a cluster of machines (2 machines).

Here, we repeat the description of query workload from D1.3.1 (GeoKnow Consortium, 2013). The selection of the queries in the workload is based on the common requests of a user dealing with the Linked Geodata browser (http://browser.linkedgeodata.org/). The workload mimics a browsing user in a query run. A query run, based on a random seed, deterministically picks 10 center points, and executes 12 steps (12 different zoom levels), each step consisting of two queries: the facet count query and one instance (aggregation) queries. Thus the workload in total consists of (at least) 240 queries.

The benchmark tested how the system behaves when it handles one user at a time (power run - 1 stream row), of 16 users for the throughput run (16 streams row). It calculates how many queries can be finish per second (PagePerSec), and it reports this number divided by the dollar cost of the system being tested (PagePerSec/K$). These metrics are measured separately for each step (1-12) from the query workload for different zoom levels. The geometric mean of metrics stored in columns from step01 to step06 is written in the column Low zoom score, providing us information of the liability of the system to cope with low zoom level queries. Similarly, the high zoom score is reported, as well as the total score.
FacetBench 1.0 (SF=1)

<table>
<thead>
<tr>
<th>hardware</th>
<th>2x (dual Xeon E5-2630, 2.33GHz, 192GB RAM, 8 disks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Virtuoso v7</td>
</tr>
<tr>
<td></td>
<td>Linux 2.6</td>
</tr>
<tr>
<td>Price</td>
<td>$13,000 @ December 31, 2014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>database size</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bulk load time</td>
<td>500Kt/s</td>
</tr>
</tbody>
</table>

| Metrics:                |                                                     |
| PagePerSec              |                                                     |
| PagePerSec/KS           |                                                     |

<table>
<thead>
<tr>
<th></th>
<th>step01</th>
<th>step02</th>
<th>step03</th>
<th>step04</th>
<th>step05</th>
<th>step06</th>
<th>step07</th>
<th>step08</th>
<th>step09</th>
<th>step10</th>
<th>step11</th>
<th>step12</th>
<th>LGB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 stream</td>
<td>0.0573598</td>
<td>0.158469</td>
<td>0.203244</td>
<td>0.453782</td>
<td>0.536222</td>
<td>0.560915</td>
<td>0.781739</td>
<td>0.749681</td>
<td>1.62048</td>
<td>1.44404</td>
<td>2.51572</td>
<td>2.0284</td>
<td>1.1822</td>
</tr>
<tr>
<td></td>
<td>0.251487</td>
<td>0.02/KS</td>
<td>0.25167</td>
<td>0.02/KS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 streams</td>
<td>0.0561634</td>
<td>0.16984</td>
<td>0.251216</td>
<td>0.312697</td>
<td>0.386664</td>
<td>0.360722</td>
<td>0.527591</td>
<td>0.570452</td>
<td>1.54365</td>
<td>1.15935</td>
<td>2.8056</td>
<td>1.8072</td>
<td>0.506689</td>
</tr>
<tr>
<td></td>
<td>0.381476</td>
<td>0.02/KS</td>
<td>0.381476</td>
<td>0.02/KS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Virtuoso SPARQL results

From Table 3, we can conclude that the average execution time of low zoom level queries is 2.88s, while the average of the high zoom levels is 0.40s, giving us the total average of 1.64s. These values are for the power run, and the following are for the throughput run with 16 users: low zoom level – 48.31s; high zoom level – 8.09s; total – 28.2s.

3.2 Comparison of results from this and the previous year

From the following graph, we can see that the results are worse than last year. In average, all the queries on low zoom levels are slower by factor of 3, and the situation is similar on high zoom levels. For the throughput run, the factor is almost 4.
The reason for that is the following: dataset from this year was considerably larger. It is already said that the total size of newer dataset was 1.8 times bigger than last year. But, all the queries of this benchmark explore the highly populated areas of Europe, major cities where there are plenty of instances provided by OSM, even on very high zoom levels. These areas are enriched much more than all the others, so there are a couple of hundreds times more nodes in these areas. In the Deliverable 1.3.1, we described the query workload, and it is said that a query run, based on a random seed, deterministically picks 10 centre points (big cities), and executes 24 queries on 12 zoom levels. For the sake of fair comparison, we didn't use a random seed, but the same seed as last year, and we got the same city centres. So, the cities that are chosen are presented in the following table, together with the number of instances that are covered on the lowest zoom level. More precisely, the numbers do not present the total number of nodes, but we considered only the instances that belong to the 50 most numerous facets, so the vast majority of them are counted (for Facet Count Query). The numbers in the third column represent the numbers of nodes of the four selected facets (for Instance Aggregation Query). The numbers in second and fourth column have the same meaning as the previous columns, but for the previous year's dataset.

<table>
<thead>
<tr>
<th>City</th>
<th>Total (this year)</th>
<th>4 facets (this year)</th>
<th>Total (last year)</th>
<th>4 facets (last year)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotterdam</td>
<td>472944490</td>
<td>1714299</td>
<td>39128</td>
<td>20208</td>
</tr>
<tr>
<td>Munich</td>
<td>384592219</td>
<td>2689342</td>
<td>182594</td>
<td>127420</td>
</tr>
<tr>
<td>Budapest</td>
<td>139571587</td>
<td>894782</td>
<td>125193</td>
<td>77013</td>
</tr>
<tr>
<td>Lisbon</td>
<td>23233799</td>
<td>136433</td>
<td>4359</td>
<td>0</td>
</tr>
<tr>
<td>Berlin</td>
<td>188618025</td>
<td>1524786</td>
<td>65884</td>
<td>16181</td>
</tr>
<tr>
<td>Manchester</td>
<td>114100591</td>
<td>975928</td>
<td>60637</td>
<td>0</td>
</tr>
<tr>
<td>Prague</td>
<td>300799622</td>
<td>1786212</td>
<td>84232</td>
<td>16929</td>
</tr>
<tr>
<td>Copenhagen</td>
<td>87690948</td>
<td>549922</td>
<td>11470</td>
<td>0</td>
</tr>
<tr>
<td>Birmingham</td>
<td>136836063</td>
<td>1434415</td>
<td>36941</td>
<td>0</td>
</tr>
<tr>
<td>Warsaw</td>
<td>87249957</td>
<td>774194</td>
<td>62040</td>
<td>53429</td>
</tr>
</tbody>
</table>

Table 4: Number of nodes per city

On the following pages, one graph per city is presented. On the x-axis there are query numbers (24 of them). Query 1 and Query 2 examine map on the lowest zoom level (the rectangle 500km x
750km), and so on, while Query 23 and Query 24 deal with the highest zoom level (4km x 6km). All odd numbers represent Facet Count Query, even numbers less than or equal to 12 are Instance Aggregation Query, while all the others are instances of Instance Query. The execution times in milliseconds are shown on the y-axis.

Figure 3: Query execution times exploring Rotterdam

Figure 4: Query execution times exploring Munich
Figure 5: Query execution times exploring Budapest

Figure 6: Query execution times exploring Lisbon
Figure 7: Query execution times exploring Berlin

Figure 8: Query execution times exploring Manchester
Figure 9: Query execution times exploring Prague

Figure 10: Query execution times exploring Copenhagen
In the following chart, we present how the average times are longer this year than last, grouped by cities.
Figure 13: Deceleration grouped by cities

We notice that the greatest decelerations are in the case of Birmingham, Manchester, Copenhagen and Lisbon. The cause of that can be found in the previous table (Table 4: Number of nodes per city). These cities had 0 nodes of the specified facet in the previous dataset, while now, there are lot of them. Due to this, last year the optimizer produced a query plan that would first filter everything on these 4 selected facets, and that way, all the Instance Aggregation Queries and Instance Queries were very fast. Taking this into account, we can note that curves from graphs representing comparison of execution times for Birmingham, Manchester, Copenhagen and Lisbon, do not follow each other, as in the case of all other cities. The red dots with even x-coordinates are quite low. The next city in which the queries perform more slowly is Rotterdam, and it corresponds to the fact that Rotterdam has grown the most since last year in the number of nodes (276 times more nodes than last year). In case of all the other cities, the average execution time of queries is less than 3.5 times slower than last year, and that is reasonable in terms of mentioned table (Table 4: Number of nodes per city).

The absence of nodes of specified facet in these 4 cities has the impact on the following chart, where we present how the average times are longer this year than last, grouped by queries, and zoom levels. The execution times are less than 3 times as long for Facet Count Queries (odd x-coordinate), while the other types of queries are slowed down by a significantly larger factor.
Figure 14: Deceleration grouped by queries
4. Integration of LGD with LDBC SNB

In this section, we will present the integration of LGD with the dataset obtained from the efforts of Linked Data Benchmark Council (LDBC) which aims to establish industry cooperation between vendors of RDF and Graph database technologies in developing, endorsing, and publishing reliable and insightful benchmark results.

As part of this project two official benchmarks for industry are being developed: Social Network Benchmark (SNB) and Semantic Publishing Benchmark (SPB). The first one consists of a data generator that generates a synthetic social network, used in this integration. The LDBC-SNB Data Generator (DATAGEN) is software responsible of providing the data sets used by all the LDBC benchmarks. This data generator is designed to produce directed labelled graphs that mimic the characteristics of those graphs of real data. It exploits parallelism via Hadoop, so you can address huge-sized problems using cluster hardware. Even if a user does not posses a cluster, he can use a local install of Hadoop in pseudo-distributed mode to take advantage of multi-core parallelism in a single computer. The SNB data generator can generate data at scales 1GB, 3GB, 10GB, 30GB, etc... These are the official scale factors, where the data size is measured as the total uncompressed size of the CSV textual data. The scaling is done by varying the size of the network in terms of users. The network always contains three years of user interaction. The output of the DATAGEN can be serialized in ttl, n3 of csv formats, therefore there is no need for a rewriter tool in this integration.

The generated dataset represents a snapshot of the activity of a social network for a period of time. The following figure describes the schema of the data generated using UML. The Data includes entities such as Persons, Organizations, and Places. The schema also models the way persons interact, by means of the friendship relations established with other persons, and the sharing of content such as messages (both textual and images), replies to messages and likes to messages. People form groups to talk about specific topics, which are represented as tags.
We enrich this dataset with geo-locations from LGD in the following way. For every entity that has a place assigned to it, we allocate geometry. For example, each person has its own location, e.g. home city:

```
sn:pers00000005497558391012 snvoc:isLocatedIn <http://dbpedia.org/resource/Beijing> .
```

In this case we assign a random LGD node representing a residential building, preferably in that city. In the SNB dataset, every university has the city where it is located, so we enrich this with geo location from LGD dataset, preferably real location. We deal in the same way with locations of companies people work for. For all messages (textual and photos) that persons posted, the geometries representing the exact location of a post are assigned as follows: 70% of posted content is made from home location of a person, while the rest is created outside of home, mostly on some popular locations, tourist regions and other kind of well known places.
4.1 Query examples

This integration provides us tremendous opportunities in terms of getting answers to variety of business questions. Some of them follow:

- We are interested in the most popular tags per places. In such a way, we will find out what people usually talk about, grouped by places. In the following query, we divide the map of Bremen into 40x20 conceptual square tiles, and count tags of posts per tile, so we can eventually conclude that persons from the north have different interests than persons from the south.

```sparql
select ?x ?y ?tag ?cnt
where {
  {
    select ?x ?y ?tag count(*) as ?cnt
    where {
      select ?post bif:st_x(?place) bif:st_y(?place) xsd:integer(40* (bif:st_x(?place) - 8.507008900000001)/(0.982870700000001 - 8.507008900000001)) as ?x
      xsd:integer(20*(bif:st_y(?place) - 53.01837158203125)/(53.6066316 - 53.01837158203125)) as ?y
      ?tag
    }
    group by ?x ?y ?tag
    order by ?x ?y desc(?cnt)
  },
  filter (?cnt > 500)
}
```

The places can be parted in many other ways. We divided Bremen by longitude and latitude, but we can use administrative borders or natural boundaries (such as river flows, or coast line) as well.

- We would like to know if there are differences between the places where Americans and Chinese take photos the most frequently. The possible results can show us that the first ones are interested in culture and history, while the second ones prefer entertainments, or vice versa. The map is split as in the above example.

```sparql
select ?x ?y ?country
((?cnt * 100.0)/ (select count(*) where
```
• If we compare two regions (north and south of Bremen in our case), and want to know from which area people more often come to visit the other ones, we will use the following query.

```sparql
select ?s_to_n ?n_to_s
where
{ 
  { 
    select count(*) as ?s_to_n
    where
    { 
      ?post snvoc:hasCreator ?person .
      ?person snvoc:isLocatedIn ?city .
      filter (bif:st_intersects (?post, ?place)) .
    } 
} . 
{ 
  select count(*) as ?n_to_s
  where
  { 
    ?post snvoc:hasCreator ?person .
    ?person snvoc:isLocatedIn ?city .
    filter (bif:st_intersects (?post, ?home)) .
  } 
} . 
}
```
We would like to know how often American country music fans go to visit the origin of that music, e.g., Southern United States (in this example, south of Bremen). We consider somebody a fan, if he is interested in Bob Dylan, Johnny Cash, Ray Charles or Willie Nelson (this was present in the dataset). We count only persons that are not located in the place of interest, and calculate percentage of their post made right there.

```
sparql
select sum(?x) * 100.0 / count(*)
where
{
  ?post snvoc:hasCreator ?person .
  filter (bif:st_intersects (bif:st_geomfromtext("BOX(53.01837158203125, 53.136023586)", ?place)) .
}
}
```

The following query can be used if we are looking for cities that are isolated, e.g. the high percentage of friends of persons from that city are located in radius of 10km.

```
sparql
select ?cityname sum(?has_friend_in_neighbourhood) * 100.0 / count(*)
where
{
  select ?person ?cityname
  (if (bif:st_distance(?home, ?frhome) < 10, 1, 0)) as
?has_friend_in_neighbourhood
  where
    ?person snvoc:hasInterest ?artist .
    ?post snvoc:hasCreator ?person .
    filter (?artist = <http://dbpedia.org/resource/Bob_Dylan> or
      ?artist = <http://dbpedia.org/resource/Johnny_Cash> or
      ?artist = <http://dbpedia.org/resource/Ray_Charles> or
    filter (bif:st_intersects (bif:st_geomfromtext("BOX(8.507008900000001 53.01837158203125, 8.982870700000001 53.136023586)", ?home) = 0) .
}
```
\begin{verbatim}
{  ?person snvoc:isLocatedIn ?city .
}
group by ?cityname
order by desc(?)
limit 100
\end{verbatim}

- The last query gives us the answer to the question, in which country people travel furthest from home to their work place.

\begin{verbatim}
sparql
select ?country avg(bif:st_distance(?home, ?work))
where
{
  {  
    ?person snvoc:isLocatedIn ?city .
    ?person snvoc:workAt ?company .
    ?company snvoc:hasOrganisation ?o .
  }
} 
group by ?country
order by desc(?)
\end{verbatim}

In the following months of project, we plan to use these queries to make performance tests against this additional workload, and compare Virtuoso SPARQL results with and without characteristic sets.
5. Conclusion and Future Work

In this deliverable, we used the Benchmarking System presented in the previous deliverables to evaluate the performance of Virtuoso SPARQL with the newer dataset from this year, and compared them with the results from the previous year in as much as possible. The integration of LGD with SNB datasets was performed, and queries were devised for exploiting these new dimensions of the resultant dataset.

In the remaining months of a project, we shall repeat the whole benchmark against the new Virtuoso Structure-Aware RDF Store feature, using Characteristic Set (CS). CS and Geospatial clustering will be applied to the workloads in the coming months.
6. Appendix

All the scripts and programs from the GeoKnow Benchmark are available as a git project:

https://github.com/GeoKnow/GeoBenchLab
7. Bibliography


