Facilitating the Storage, Retrieval and Analysis of Heterogeneous Spatio-temporal Data

Michael Stewart - 20947715

Supervisors: Professor Wei Liu and Professor Rachel Cardell-Oliver

INTRODUCTION

Techniques for **visualising** and **analysing spatio-temporal data** are currently in demand. This demand is rising with the increase of large geo-spatial datasets, sensor data, and trackable moving objects.

One common solution for analysing spatio-temporal data is to import it into a NoSQL database that supports spatial and temporal queries, such as OrientDB. However, in some cases, the structure of the dataset does not enable these types of queries to be performed.

This project addresses this problem by contributing the following:

- a comprehensive literature review that covers spatial, temporal, and spatio-temporal data, as well as NoSQL databases;
- a generic model for representing heterogeneous spatio-temporal data within a NoSQL graph database; and
- a pre-processing framework that may be used to adapt heterogeneous data to our model.

LITERATURE REVIEW

This review briefly covers spatial, temporal and spatio-temporal databases, as well as NoSQL databases.

Glossary

- **Spatial database:** holds information relating to the location of objects in space.
- **Temporal database:** stores data that contains timestamps.
- **Spatio-temporal database:** combines the theory of spatial and temporal databases.
- **NoSQL**: in comparison to relational databases, NoSQL databases handle unstructured data, which does not adhere to a fixed schema.

Spatio-temporal data types

- **Spatial** [1]:
  - **Point:** a point in space, e.g. a house.
  - **Line:** connects two points, e.g. a road.
  - **Region:** a two-dimensional area containing many points or lines, e.g. a city.
- **Temporal** [2]:
  - **Valid time:** when a fact was true in reality.
  - **Transaction time:** when a record was saved in the database.
  - **User-defined time:** semantic, non-queryable time, such as 'birthday'.

NoSQL models [3]

- **Document store:** data is stored in JSON or XML format, allowing nested data.
- **Key-value store:** each record contains a key, mapped to one or more values.
- **Column family store:** data is stored in columns, rather than in rows.
- **Graph database:** data is stored in a graph model as vertices and edges.

OrientDB[1]

OrientDB was chosen for this project because it supports both the **document** and **graph** models, allowing storage of nested data and improved performance on data containing associations.

PROJECT SCOPE

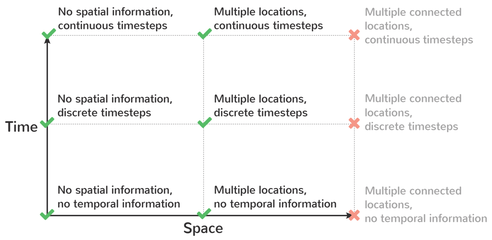

Figure 1. The relationships between spatial and temporal data, in three levels of granularity.

It was decided that all spatio-temporal relationships, except for those that involve multiple connected locations, would be managed in this project. This is due to the complexity that results from connected locations, as well as OrientDB's inability to handle line and region spatial data types.

SYSTEM ARCHITECTURE

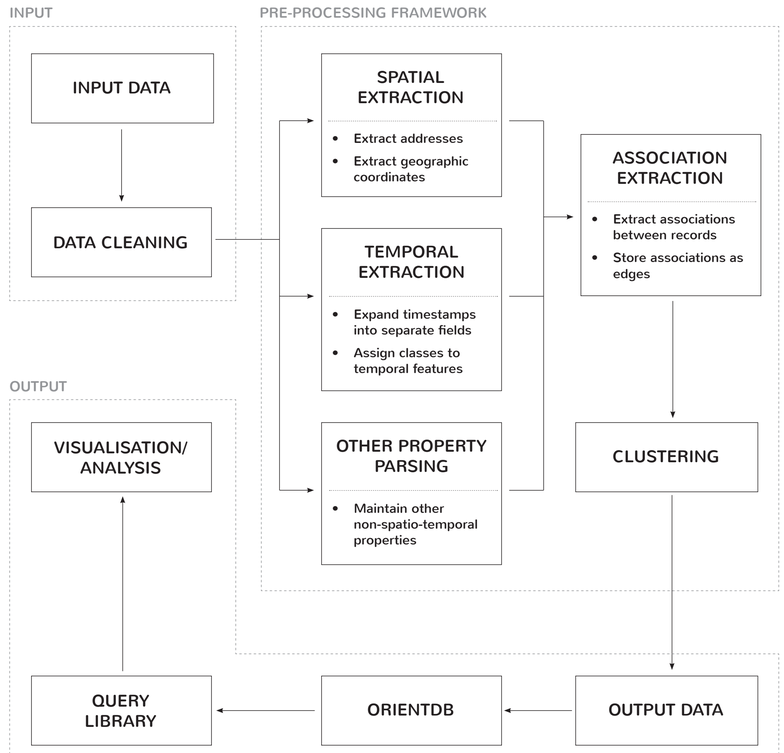

Figure 2 (above). An outline of the pre-processing framework introduced in this project.

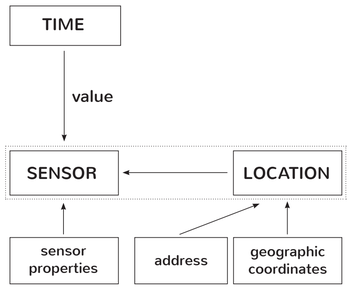

Figure 3. Our **high-level graph model**, which describes how data is represented after pre-processing.

The **pre-processing framework** (Figure 2) adapts a dataset to the **high-level graph model** (Figure 3), allowing it to support more complex spatial and temporal queries.

- **Spatial extraction:** a field is generated for each component of an address. Spatial indexes are generated for records with geographic coordinates, allowing spatial search.
- **Temporal extraction:** timestamps are parsed and fields are generated for each unit of time.
- **Association extraction:** edges are generated to represent associations between records.
- **Clustering:** Edges are clustered based on value (sensor reading) using K-Means clustering.
- **Output data:** The framework outputs a SQL file that may be **imported into OrientDB**.
- **Queries**, **visualisation** and **analysis** may then be performed.

EVALUATION

The evaluation was conducted on **Water Corporation sensor data**[2], which was inserted into OrientDB using the pre-processing framework (Figure 1). This dataset contains water usage values for 500 houses, recorded each hour for a year. The performance for a temporal, spatio-temporal, and water reading query (using a cluster generated during pre-processing) are shown below.

Query	SQL statement	Results	Time
Water usages of all houses at a particular time	`SELECT in.@rid AS House, Value FROM WaterUsage House WHERE out.timestamp = "20/01/2015 0:00"`	497	28.1s
Water usages of all houses in a particular suburb across one particular week day	`SELECT in.@rid as House, out.timestamp AS Timestamp, Value FROM WaterUsage House WHERE out.WeekDay = "Monday" AND in.Suburb = "WEST LAMINGTON"`	28,615	47.6s
Water usages of all houses that used between 1,000 and 1,050 units of water in an hour during June	`SELECT in.@rid AS House, out.timestamp AS Timestamp, Value FROM CLUSTER: waterusage_house_997_1065 WHERE Value > 1000 AND Value < 1050 AND out.Month = 6`	73	0.1s

The four evaluation criteria were based on the advantages OrientDB claims[1] over relational databases. The results were as follows:

- **Performance:** The queries complete within a reasonable time on a modern desktop computer, and clustering was very effective.
- **Scalability:** Linear with search space (see Figure 4), but some large-scale queries that return hundreds of thousands of records cause OrientDB to run out of memory.
- **Footprint:** Suboptimal when handling document-based data, as OrientDB doubles the size of the JSON data it contains.
- **Developer productivity:** A number of software bugs within OrientDB hamper developer productivity.

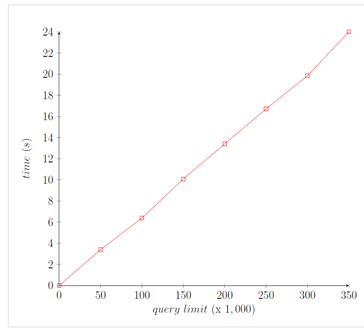

Figure 4. The scalability of OrientDB. Query time scales linearly with the search space, even when handling queries that return 350,000 results.

CONCLUSION

We have introduced a generic model for representing **spatio-temporal data** within a **NoSQL graph database**. Our goals have been achieved by adapting heterogeneous data to this model using a **pre-processing framework**. We have identied **ten spatial, temporal, and spatio-temporal queries** that would not have been possible prior to pre-processing. This framework is also capable of clustering data in order to **increase performance**, and has the potential to greatly speed up queries.

REFERENCES

[1] Max J Egenhofer and John Herring. Categorizing binary topological relations between regions, lines, and points in geographic databases. 1990.

[2] Richard Thomas Snodgrass. Temporal databases. In *IEEE computer*. Citeseer, 1986.

[3] Adam Lith and Jakob Mattsson. Investigating storage solutions for large data-a comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data. 2010.

[1] *OrientDB*. http://orientdb.com/docs/2.0/orientdb.wiki/Tutorial-Introduction-to-the-NoSQL-world.html

[2] *The University of Western Australia*. http://staffhome.ecm.uwa.edu.au/~00047104/waterusesignatures-june2015/