ULB UNIVERSITÉ LIBRE DE BRUXELLES

Implementing Yelp Dataset with Document-based Technology and Tools: MarkLogic DB and Arango DB

Advanced Databases (INFO-H-415)

Erasmus Mundus Joint Master's Degree
in
Big Data Management and Analytics

by

Ahmad (ahmad@ulb.be)
Rishika Gupta (rishika.gupta@ulb.be)
Mir Wise Khan (mir.khan@ulb.be)
Chidiebere Ogbuchi (chidiebere.ogbuchi@ulb.be)

under the guidance of

Prof. Esteban Zimanyi

École polytechnique de Bruxelles
Université Libre de Bruxelles
December 2022

Contents

Abstract **i**

1 Introduction **1**
1.1 Overview . . . 1
1.2 Problem Statement . . . 1
1.3 Aim and Objectives . . . 2
1.4 Tools Used . . . 2
1.5 Limitations and Justifications . . . 2

2 Technology Fundamentals **3**
2.1 Advanced Database Management Systems . . . 3
2.1.1 Document Stores Technology . . . 3
2.1.2 ArangoDB . . . 4
2.1.3 Marklogic . . . 5

3 Implementation **7**

4 Proposed Application and Dataset **8**
4.1 Proposed Application . . . 8
4.2 Dataset: Yelp . . . 8
4.3 Scale Factors . . . 10
4.4 Why DocumentDB for this application? . . . 10
4.5 Relational Model Comparison . . . 10
4.6 Database Setup . . . 11
4.6.1 General Setup . . . 11
4.6.2 Arango . . . 12
4.6.3 Marklogic . . . 12

5 Use Cases **14**
5.1 CRUD Queries . . . 14
5.2 OLAP Queries . . . 15

6 Benchmarking, Results and Analysis **16**
6.1 Benchmarking Methodology . . . 16
6.2 Results and Analysis . . . 16
6.2.1 Overall Results . . . 16
6.2.2 CRUD Results . . . 17
6.2.3 OLAP Results . . . 18

7 Conclusion **22**

Bibliography **i**

List of Figures

2.1 Components of ArangoDB 4
2.2 Schemaless Data 5
2.3 Sample Composable Query Lanaguage 5
2.4 XQuery Sample 6
2.5 XSLT Sample 6

4.1 Yelp Website Snippet 8
4.2 Yelp Website Snippet 9
4.3 Sample Document from Business Collection 11
4.4 Query for bulk importing into Arango Database 12
4.5 Arango Database Setup Flow 12
4.6 Query for bulk importing into Marklogic Database 13
4.7 Marklogic Database Setup Flow 13

6.1 Total run time in mins for all queries 17
6.2 Total run time in mins for all CRUD queries 17
6.3 Query 01 - Average Runtime (Mins) by Scale Factor (SF) 18
6.4 Query 02 - Average Runtime (Mins) by Scale Factor (SF) 18
6.5 Query 03 - Average Runtime (Mins) by Scale Factor (SF) 18
6.6 Query 04 - Average Runtime (Mins) by Scale Factor (SF) 18
6.7 Query 05 - Average Runtime (Mins) by Scale Factor (SF) 18
6.8 Total run time in mins for all OLAP queries 19
6.9 Query 06 - Average Runtime (Mins) by Scale Factor (SF) 19
6.10 Query 07 - Average Runtime (Mins) by Scale Factor (SF) 19
6.11 Query 08 - Average Runtime (Mins) by Scale Factor (SF) 19
6.12 Query 09 - Average Runtime (Mins) by Scale Factor (SF) 19
6.13 Query 10 - Average Runtime (Mins) by Scale Factor (SF) 20
6.14 Query 11 - Average Runtime (Mins) by Scale Factor (SF) 20
6.15 Query 12 - Average Runtime (Mins) by Scale Factor (SF) 20
6.16 Query 13 - Average Runtime (Mins) by Scale Factor (SF) 20
6.17 Query 14 - Average Runtime (Mins) by Scale Factor (SF) 20
6.18 Query 15 - Average Runtime (Mins) by Scale Factor (SF) 20
6.19 Query 16 - Average Runtime (Mins) by Scale Factor (SF) 20
6.20 Query 17 - Average Runtime (Mins) by Scale Factor (SF) 20
6.21 Query 18 - Average Runtime (Mins) by Scale Factor (SF) 21

List of Tables

1.1 Tools Used for Benchmarking ArangoDB and MarklogicDB 2

3.1 Local Machine Specifications 7

4.1 Collections Overview 9
4.2 Scale Factor Summary Overview 10

5.1 CRUD Queries Overview 14
5.2 OLAP Queries Overview 15

List of Abbreviations

AQL	ArangoDB Query Language
CRUD	Create Read Update Delete
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
CMD	Command Line Prompt
DB	Database
DBMS	Database Management Systems
GB	Gigabytes
JSON	JavaScript Object Notation
MS	Milli-seconds
OLAP	Online Analytical Processing
RDBMS	Relational Database Management Systems
RDF	Resource Description Framework
SF	Scale Factor
SQL	Structured Query Language
XML	Extensible Markup Language
XQuery	XML Query
XSLT	Extensible Stylesheet Language Transformations

Abstract

The document database is category of the NoSQL database. It is a non-relational database that stores and queries data from a JSON like format. In this study, we explain the foundations of the document stores technology implemented over the Arango and Marklogic database engines. The underlying concept was illustrated using the Yelp dataset to explore and assess the performance of the databases used over document stores. Furthermore for the study benchmarks, two db-engines are selected in relation to the database requirements of the application. In general, the benchmarks model was very essential aspect in comparing the Arango and Marklogic as well as gain insight on the performance of both engines in order to identify suitable application.

Chapter 1

Introduction

1.1 Overview

This project entails illustrating database technology (e.g key-value, Search engines, document stores, stream, time series databases, etc.) and its application over two selected database management systems. It addresses in a technical way, the fundamentals of the database management systems and a use case to expound the bedrock of the underlying technology. The study benchmarks the proposed tools in relation to the requirements of the database over several scale factors of a dataset. Some set of queries and updates are generated and implemented to determine the performance and behavior of the database. In this report, we employ the use of Document Stores of ArangoDb and MarkLogic Database engines as preferred choice of technology and tools respectively. We study the aforementioned technology and deliver a self-explanatory report on the crucial aspects of the implementation [INFO-H-415: Advanced Databases, 2022].

1.2 Problem Statement

There are large numbers of businesses across the globe and it is strenuous to know and select a quality restaurant that offers good services. In this study, we hope to apply document store over the Arango and Marklogic engines to design a review application that helps alleviate this issue. In our method, we utilize the Yelp dataset which presents a subset of several restaurants with users and reviews data. Yelp website offers JSON files that provides essential information regarding restaurants as well as elaborated review data that supports us in creating the review application. However, the original yelp dataset does not go further into analyzing the reasons behind restaurant performance ratings in relation to user's reviews. In order to support customer decision making in picking a choice of restaurant, this study embarks on contributing to available insights by analytically digging deeper into the yelp datasets. This application will assist in improving customer satisfaction as well and help restaurant business improve on their services. Hence, the yelp dataset will be implemented over the afore selected database engine in order to build a review application that provides more perspicaciously in user's choice of restaurant thus improving restaurant business services.

1.3 Aim and Objectives

The main aim of this study is to implement a document store database technology and illustrate its application (review site) in ArangoDb and MarkLogic. The specific objectives include:

- Generate CRUD and OLAP queries for database maintenance and decision support
- Evaluate the bench marking performance and analyse the results
- Assess the advantages and disadvantages of both DB Engines tested in the application of document store technology

1.4 Tools Used

The tools installed and utilized to perform the benchmark operation are summarized in table 1.1.

Tool	**Version**	**Description**
ArangoDb	3.10	Native multi-model DBMS for graph, document, key/-value and search. All in one engine and accessible with one query language.
Marklogic	10.0	Operational and transactional Enterprise NoSQL database to implement document stores technology.
JupyterLab	3.3.2	IDE for Python and iPython notebook.
Python	3.10.5	Python was used as the main programming language for running our scripts.
MS PowerBI	2.110	Power BI was utilized to create visualizations of bench-marking results for further analysis.
GitHub	2.38.1	GitHub Desktop was used to share code files as well as images conveniently with the team members.

Table 1.1: Tools Used for Benchmarking ArangoDB and MarklogicDB

1.5 Limitations and Justifications

Given that project was implemented on a local machine, there is a usual restriction on availability of memory and computational resources. Also, the DB-engines required use of their respective query languages to generate queries which was limited by the flexibility and expression of the language as compared to standard sql language. In implementing the benchmarks, there is also limit of scaling as it would have been preferable to expand to higher scale factor with larger tuples. Despite these complexities, the report has provided good foundational knowledge on document stores and it's application over the proposed Db-engines.

Chapter 2

Technology Fundamentals

The project work focused on implementing a document stores technology for use in a review site over yelp, while utilizing ArangoDb and MarkLogic.as database engines. This chapter introduces the fundamentals behind the technology and methods used in completing the project.

2.1 Advanced Database Management Systems

Databases systems are generally used for storing, querying, manipulating, processing, maintaining and retrieving data. They can be interacted with through several software enterprise and web based environments called database management system (DBMS).. Advanced database systems attempts to satisfy the needs of modern database application by providing advanced functionalities in terms of data modeling, multimedia data type support, data integration capabilities, query languages, system features, and interfaces to other environments. [Kubiak et al., 2012]. In other words, a DBMS usually comprises of a database and a database system.

Furthermore, a DBMS that supports the use of relational data modelling is usually known as a Relational Database Management System (RDBMS). An RDBMS (such as oracle, mySQL, Postgresql, Microsoft SQL Server etc) stores data in tuple based set of uniform structure and consists of table schema which has defines columns names called attributes. At the other hand, a DBMS that usually supports other data models is denoted as NoSQL-System. Unlike RDBMS, they usually don't use relational data modellinga and come in different heterogenous classification such as RDF Stores, Content stores, Wide Column Stores, Key-Value Stores, Document Stores, Content Stores, Search Engines, Native XML DBMS, Graph DBMS amongst other[DB-Engines, 2022]. In this report, we focus on Documents stores technology and its application using ArangoDb and Marklogic.

2.1.1 Document Stores Technology

Document stores is a database system that allows storage, retrieval, manipulation and management of document-oriented information. It is sometimes known as document-oriented database systems and are typically characterized by their schema-less arrangement of data. In other words, data structure is usually not uniform on each columns as well as contain records with nested structures and dif-

ferent attributes. Document stores utilizes semi-structured data that use internal explications which can be managed categorically in application often in JSON. They are inherently a sub-concept of the key-value store with negligible differences only that in key value stores, data is considered to be inherently non-transparent to the database, whereas the document stores depends on the document's internal structure for it to extricate metadata used by the database system for manipulation and optimization.

2.1.2 ArangoDB

ArangoDb is an open source and commercial DBMS written in C++, usually used for graphs, document stores, key-value stores, search engines and machine learning, with a wide array of functions and rich features. It is scalable and multi-model database that supports a range of data access patterns with its flexible composable query language. Also, it can be built as an on-premise deployment basis as well as incorporated managed cloud service.

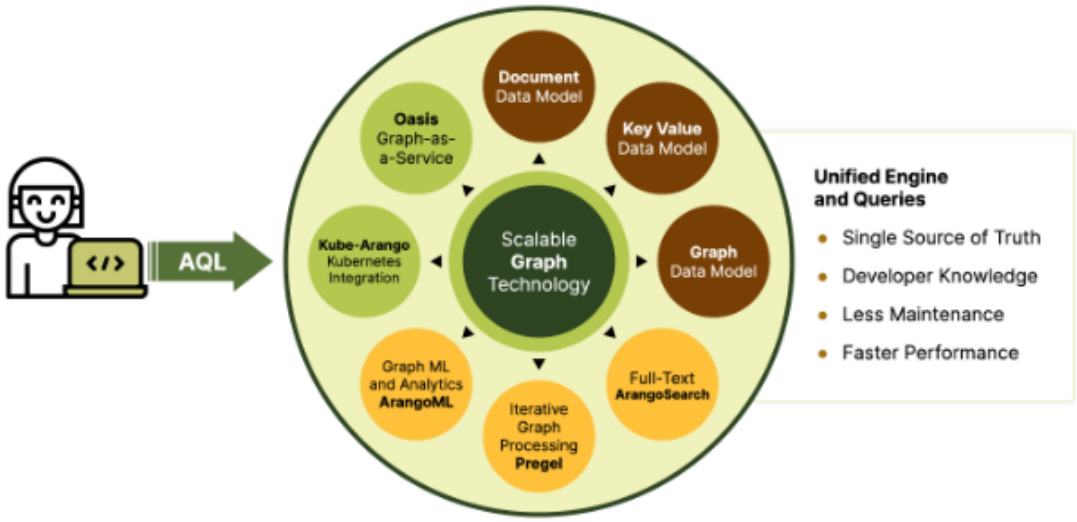

Figure 2.1: Components of ArangoDB
[Introduction to ArangoDB's Technical Documentation and Ecosystem, 2022]

Beyond it's use for graph cases, ArangoDB equally allows manipulation of structured semi-structured and unstructured data in the form of JSON objects without schema and having to connect them to form a relationship graph. However, depending on the requirement, both graphs and unstructured schema-less data can be mixed as it's query language support multi-data models.

Figure 2.2: Schemaless Data
[Introduction to ArangoDB's Technical Documentation and Ecosystem, 2022]

```
FOR book IN Books
  FILTER book.title == "ArangoDB"
  FOR person IN 2..2 INBOUND book Sales, OUTBOUND People
    RETURN person.name
```

Figure 2.3: Sample Composable Query Lanaguage
[Introduction to ArangoDB's Technical Documentation and Ecosystem, 2022]

The query language also called the ArangoDB Query Language (AQL) which can be used to retrieve and manipulate data stored in the database is mainly declarative, i.e., it asseverates what result should be accomplished but not how it should be performed. Just as in fig 2.3, the language is human-readable as it uses English vocabularies as keywords. Some keywords overlaps Structured Query Language (SQL) even though the syntax is different. Moreover, similar to SQL, AQL supports data manipulation language (DML) however it does not support data definition language (DDL) and data control language (DCL).

Furthermore, the syntax is usually same for all clients and support large and complex queries. ArangoDb parses queries, executes it and compiles the results. However, If the query is invalid, the server outputs a message that the user client can react to and vice versa returns query results if valid. Asides AQL, it can be accessed through other methods like foxx Framework, HTTP API, java, JSON style queries, velocyStream, Graph APis etc and supports programming languages such as C, PHP, python, Go, clojure, R, Rust, Node.js amongst others. Like in this project study, ArangoDB can be deployed as the backend for several applications such content management, Internet of Things, e-commerce systems and more conventionally as a persistence layer for a wide range of services that benefit from an agile and scalable data store [Introduction to ArangoDB's Technical Documentation and Ecosystem, 2022].

2.1.3 Marklogic

MarkLogic is a NoSQL Database that incorporates search-style indexing, database internals and application server features in a a unified enterprise system. It is a multi-model database that utilizes XML and JSON as one of its core data forms due to it use of non-relational model structure. It stores documents within a transactional repository and can support binary text, document stores, native xml, search engines and Resource Description Framework (RDFs). Marklogic sometimes called

Marklogic server also has extensible and programmable application server capabilities that be deployed and scaled into massive architectures.

Furthermore, Marklogic servers are usually accessed through and interacted through with XML Query (XQuery) and Extensible Stylesheet Language Transformations (XSLT) as in Figure 2.4 and Figure 2.5. XQuery is a programmatic functional language that can query, manipulate, process and retrieve XML where as XSLT is a styie sheet language that allows easy transformation of XML during processing [Hunter, 2013].

```
let $me := doc("/passenger.xml")/passenger
for $match in cts:search(
   /driver,
   cts:and-query((
      cts:query($me/preferences/*),
      cts:reverse-query($me)
   ))
)
return base-uri($match)
```

Figure 2.4: XQuery Sample [Hunter, 2013]

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Figure 2.5: XSLT Sample [Hunter, 2013]

Other methods of accessing Marklogic includes exposing a REST API and a SQL interface over ODBC, XDBC, WebDAV, Java API, Node.js Client API, proprietary optic API etc. Similar to Arango, It supports C, C, PHP, python, ruby, Perl, Java, C++ amongst other notable programming languages and can be applied to NoSQL database for documents like user manuals, web pages, metadata, sensor data, finance, message traffic etc.

Chapter 3

Implementation

This chapter depicts a brief rundown of the project implementation. Some of the major processes undertaken are summarized as follows:

- Identifying the tools under DocumentDB.
- Downloading, installing and setting up the tools on local environment.
- Identifying a potential use-case (real-world example) that could illustrate the tools
- Downloading yelp dataset
- Cleaning, wrangling and transforming the data using suitable python commands to prepare and load the data into the database
- Choose scale factors
- Generating dataset with respect to the chosen scale factors
- Create indexes
- Compile queries and then run using python scripts, perform tests on the queries and compile all results
- Visualize all outputs and provide detailed reports.

The benchmark was implemented on a local machine with specifications illustrated in table: 3.1. The implementation details can be found on our Github link: ADB_Project.

CPU (AMD Ryzen 7 6800HS)	**RAM (DDR5 SODIMM)**	**GPU (NVIDIA GeoForceRTX3060)**
• 8 cores, 16 threads • Base clocking speed at 3.2GHz and can overclock up to 4.7GHz • 16MB L3 Cache	• 16 GB memory • 4800MHz speed	• Dedicated graphics • 6GB VRAM

Table 3.1: Local Machine Specifications

Chapter 4

Proposed Application and Dataset

4.1 Proposed Application

Yelp is an American company [Yelp, 2022] founded in 2004 by former PayPal employees (Russel Simmons and Jeremy Stoppelman). The application aims to connect several businesses with their customers through crowd-funding reviews. Figure 4.1 shows the same snippet from the Yelp application.

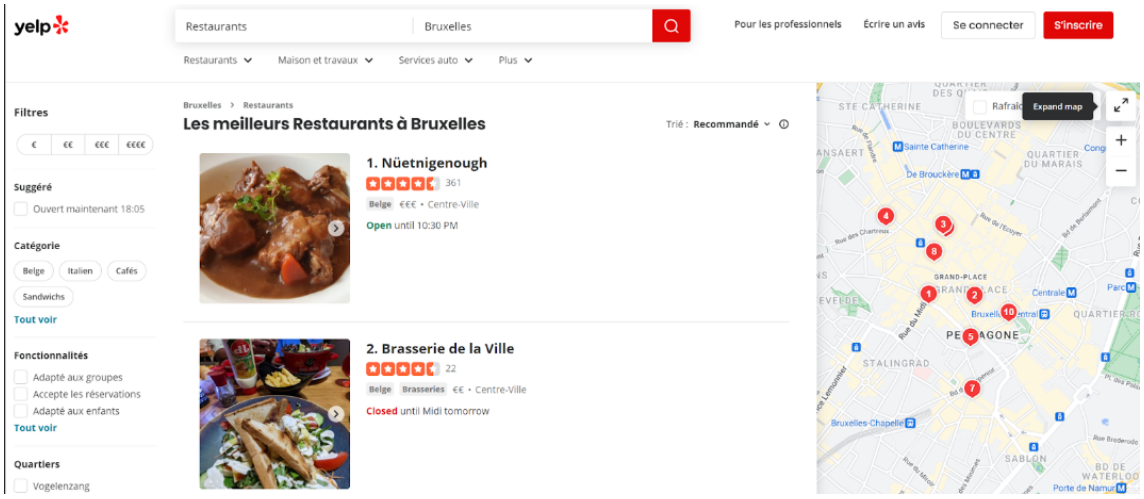

Figure 4.1: Yelp Website Snippet

4.2 Dataset: Yelp

In this project, we will be using the data set provided by Yelp Inc. [Yelp Dataset, 2022] themselves, which is retrieved from Kaggle.

The dataset comprised multiple collections namely - business, review, user, checking, tips and photos in json files. This project's focus is only on three main collections out of 6: business, review and user. The table 4.1 describes the selected collections briefly.

Collection Name	Description
Business	This collection comprises business-related details which include the location of the business, the total number of reviews that the business has, attributes, categories, etc.
User	This collection has data about the various users associated with Yelp. Attributes of the collection include a user name, total reviews they have shared, user friends, etc.
Review	This collection maps a user's review to the specific business. It displays the rating given by the user, their review, and some more attributes.

Table 4.1: Collections Overview

The relationship between the collections is depicted in figure 4.2, wherein reviews is the main collection and maps two other collections businesses (using business_id) and users (using users_id).

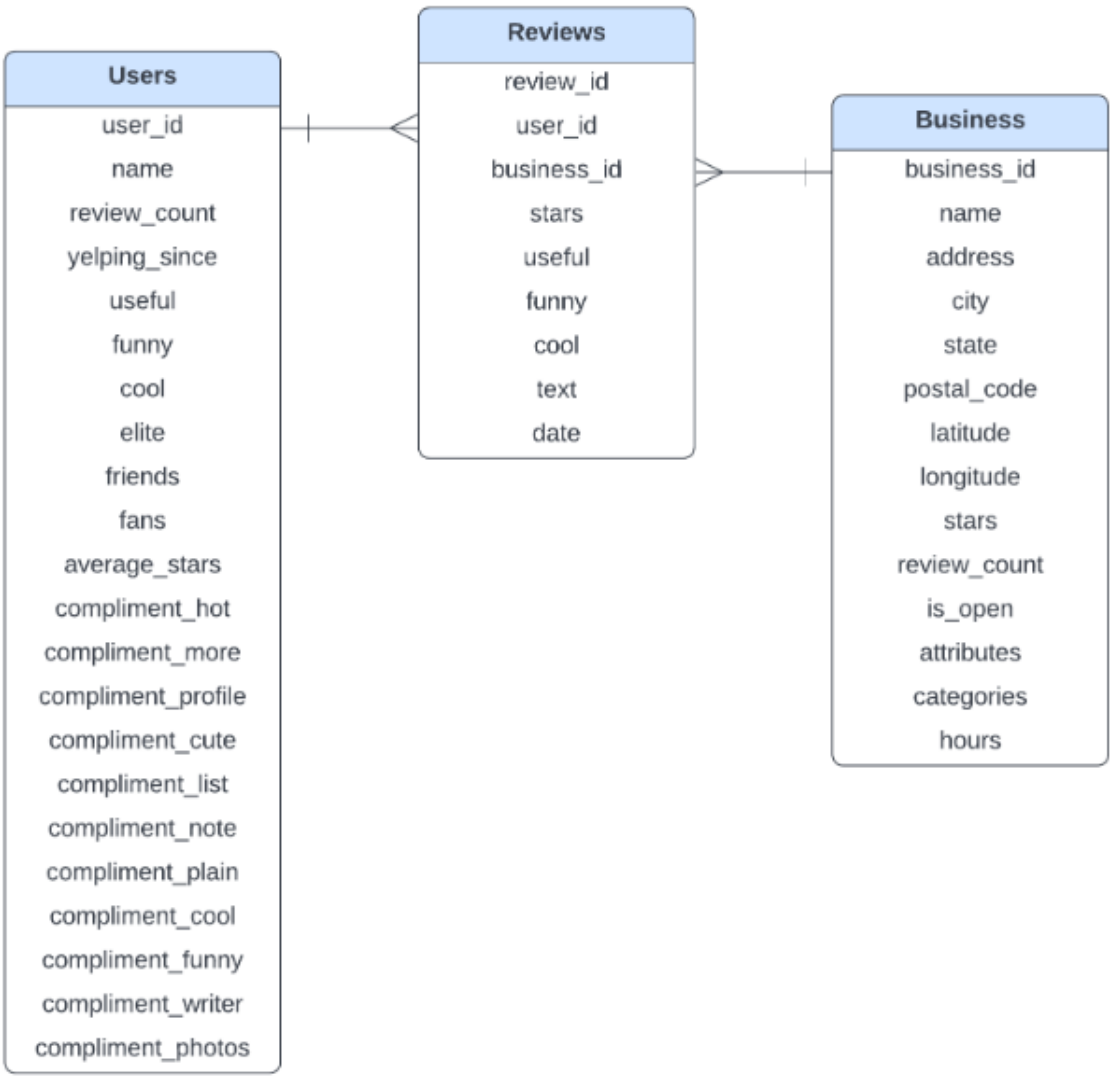

Figure 4.2: Yelp Website Snippet

4.3 Scale Factors

To determine how the chosen technology and tools perform with the application, various scale factors are chosen. The following method was used for the same:

- Identify 6 Scale Factors (SF) for Benchmarking
- Creating subsets of review collections for each SF. For example: SF1 has 1k review documents, SF2 has 10k review documents and so on
- Creating subsets of businesses collections by extracting information with repsect to the reviews data present in that SF
- Creating subsets of users collections by extracting information with repsect to the reviews data present in that SF

This shall result in 3 collections - business, reviews and users with variation in the total number of documents. This variation is depicted in the table 4.2

Collections Count	**yelp_sf_1**	**yelp_sf_2**	**yelp_sf_3**	**yelp_sf_4**	**yelp_sf_5**	**yelp_sf_6**
reviews	1000	10000	100000	1000000	2000000	6990280
businesses	823	3930	9973	27095	45055	150346
users	994	9472	79345	541999	895924	1987897

Table 4.2: Scale Factor Summary Overview

4.4 Why DocumentDB for this application?

Document databases store documents that comprise of different attributes and values. It is perfectly suited to this use case of Yelp Application as different users can provide various types of information. This concept applies to businesses as well. The schema-less feature of document stores makes this application much easier.

Let's consider a sample use case wherein a user wants to add or remove some information from the review they provided. For this example, the document of this particular user can be easily replaced with an updated version of the document that consists of the modified attributes or values (if any) or removes any attributes or values. It is because of this reason, document DBs make it easy to manage data on individual as well as fluid scales.

4.5 Relational Model Comparison

As we know that, relational model entails a database that typically follows a specific schema rather than regular dynamic changes to it.

```
{
 "business_id": "EjzWyITI75CKSKb6RGCG_Q",
 "name": "Sonny's BBQ",
 "address": "8106 W Hillsborough Avenue",
 "city": "Tampa",
 "state": "FL",
 "postal_code": "33615",
 "latitude": 27.9958415,
 "longitude": -82.5722699,
 "stars": 3,
 "review_count": 11,
 "is_open": 0,
 "attributes": {
  "GoodForKids": "False",
  "HasTV": "None",
  "NoiseLevel": "u'average'",
  "RestaurantsDelivery": "False",
  "WiFi": "u'no'",
  "RestaurantsPriceRange2": "2",
  "RestaurantsReservations": "False",
  "OutdoorSeating": "False",
  "RestaurantsTakeOut": "True",
  "RestaurantsGoodForGroups": "False",
  "RestaurantsAttire": "u'casual'",
  "Ambience": "{'romantic': False, 'intimate': False, 'touristy': False, 'hipster': False, 'divey': False, 'classy': False, 'trendy': False, 'upscale': False, 'casual': True}",
  "BusinessParking": "{'garage': False, 'street': False, 'validated': False, 'lot': False, 'valet': False}",
  "BusinessAcceptsCreditCards": "True",
  "GoodForMeal": "{'dessert': False, 'latenight': False, 'lunch': False, 'dinner': False, 'brunch': False, 'breakfast': False}"
 },
 "categories": "Barbeque, Restaurants, Event Planning & Services, Caterers",
 "hours": {
  "Monday": "11:0-21:0",
  "Tuesday": "11:0-21:0",
  "Wednesday": "11:0-21:0",
  "Thursday": "11:0-21:0",
  "Friday": "11:0-21:30",
  "Saturday": "11:0-21:30",
  "Sunday": "11:0-21:0"
 }
}
```


Figure 4.3: Sample Document from Business Collection

The following points refer to the figure 4.3, highlighting the reason why relational model database is not the best choice for this application.

- For element 'attributes', it has a nested object with many children and nested children, which would require several hundred columns to transform.
- Element 'categories' sometimes have the data type as a list of strings, while in other cases it is in a nested json object form, which adds more complexity for conversion.
- Furthermore, we cannot directly add a child {'CuisineType' : 'True'} to 'attributes', if new data is added (without changing schema).
- Given that the schema is not fixed, it is very likely that over the year Yelp's dataset would expand regularly requiring schema maintenance several times, resulting in more overhead.

In summary, due to these issues, data maintenance will have more overhead cost if stored as relational model.

4.6 Database Setup

4.6.1 General Setup

This sub-section elaborates the generic idea of any document DB set-up for the Yelp application.

- Databases are set-up in each of the tool - Arango and Marklogic for the 6 chosen scale factors, namely 'yelp_sf_1', 'yelp_sf_2', 'yelp_sf_3', 'yelp_sf_4', 'yelp_sf_5' and 'yelp_sf_6'.
- 3 Collections: Reviews, Users and Businesses are created in each of the databases.
- Document data is loaded into each of the collections for every SF.

- We know that indexes optimise the performance of databases by helping in fast retrieval of documents. For this project, a specific set of attributes is identified for indexing. Indexes are added on attributes: user_id and business_id present in all the collections.
- Once database is set-up, queries are run on each of the tools.

The further sub-sections elaborate more on the details of setting up Arango and Marklogic.

4.6.2 Arango

For Arango, command line prompt (cmd) is used mainly for importing the data into collection. The command in figure 4.4 illustrates how import is performed for SF 1. Similar imports were performed for SFs 2 till 6.

```
arangoimport --server.database "yelp_sf_1" --server.username "test_adb" --server.password "adb" --file "sf_1/reviews_1.json"
--type json --collection reviews

arangoimport --server.database "yelp_sf_1" --server.username "test_adb" --server.password "adb" --file "sf_1/users_1.json"
--type json --collection users

arangoimport --server.database "yelp_sf_1" --server.username "test_adb" --server.password "adb" --file "sf_1/
businesses_1.json" --type json --collection businesses
```

Figure 4.4: Query for bulk importing into Arango Database

Arango supports multiple index types like persistent index, inverted index, time-to-live index, etc. As for our application, persistent indexes are implemented.

Persistent indexes are general purpose indexes that are used for equality look ups based on the leftmost prefix of the index attribute, range queries or for sorting. Furthermore, operations or queries that use persistent indexes follow the path of a logarithmic time complexity. The entire set-up for this tool is also summarised in the figure 4.5.

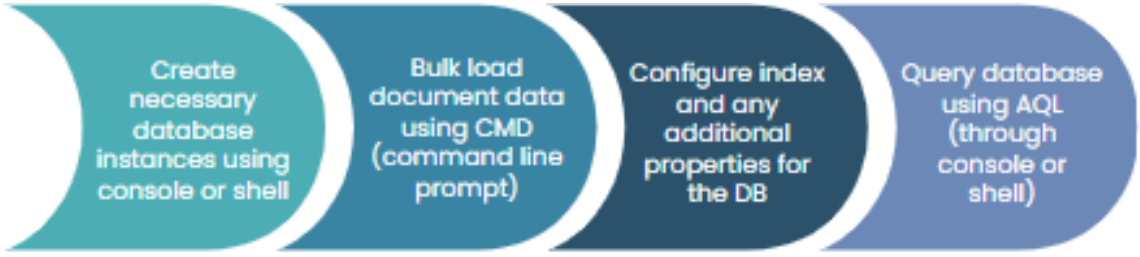

Figure 4.5: Arango Database Setup Flow

4.6.3 Marklogic

For Marklogic, MLCP pump is used for importing documents in a bulk into the collections. The command in figure 4.6 illustrates how import is performed for SF 1. Similar imports were performed for SFs 2 till 6.

```
mlcp.bat import -host localhost -port 8021 -username test_adb -password adb -input_file_path ^
"C:\sf_data\businesses_1.json" -mode local -input_file_type delimited_json -output_collections "businesses"

mlcp.bat import -host localhost -port 8021 -username test_adb -password adb -input_file_path ^
"C:\sf_data\reviews_1.json" -mode local -input_file_type delimited_json -output_collections "reviews"

mlcp.bat import -host localhost -port 8021 -username test_adb -password adb -input_file_path ^
"C:\sf_data\users_1.json" -mode local -input_file_type delimited_json -output_collections "users"
```

Figure 4.6: Query for bulk importing into Marklogic Database

As like Arango, Marklogic also supports several indexes like range-query index, element-range index, etc. but for our application only element range index is used.

Element range index accelerates queries that involve comparisons between elements. It also keeps a tab on different values that appear within the XML elements or JSON properties using a given name, type and a collation set. The entire set-up for this tool is also summarised in the figure 4.7.

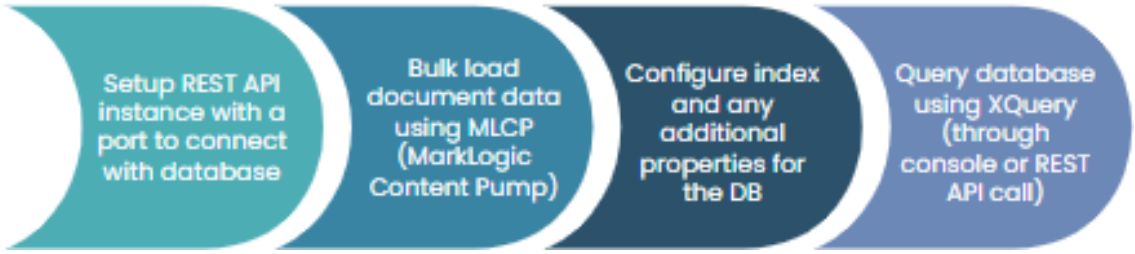

Figure 4.7: Marklogic Database Setup Flow

Chapter 5

Use Cases

This chapter covers several use cases that can help test the two databases. The chapter is split into two sections: CRUD and OLAP. In general, several query types that include simple select, insert, delete, update, joins, aggregates and word-searches are demonstrated in this project.

5.1 CRUD Queries

CRUD essentially means Create, Read, Update and Delete. For this project 5 queries are used to illustrate CRUD - 1 for each create, read and delete and 2 for update.

- Create: This command inserts documents into the collection.
- Read: This command reads the documents from the collection.
- Update: This command updates existing attribute(s) of the existing document(s).
- Delete: This command deletes documents from a collection.

Table 5.1 elaborates the queries used for CRUD with respect to the Yelp Application.

Query	**Query Description**
Q01	Insert an additional 1 million documents into the reviews collections for all SFs.
Q02	Adding an additional attribute called 'sentiment_score' which is the sum of useful, funny and cool attributes from the review collection.
Q03	Displaying the reviews from the collection.
Q04	Removing the additional attribute created in Q02.
Q05	Querying the additional 1 million documents (added in the Q01) for each SF and deleting them.

Table 5.1: CRUD Queries Overview

5.2 OLAP Queries

OLAP basically covers analytical business queries. For this project 13 queries are used to illustrate OLAP that include filters, joins, aggregates and word searches.

Table 5.2 elaborates the queries used for CRUD with respect to the Yelp Application.

Query	**Query Description**
Q06	Find all businesses with city location in 'Saint Louis'.
Q07	Find all users who have completed at least 200 reviews.
Q08	Find the reviews that were posted in the December 2017.
Q09	Find the reviews of users who have at least 100 fans.
Q10	Find the total number of businesses that currently have a rating of at least 4.5.
Q11	Find the reviews of users who have been yelping for at least a year prior to that review.
Q12	Find the reviews of businesses that are open on both days of the weekend until 10pm and were rated by a user with at least 25 fans.
Q13	Find top rated restaurants in the state of Florida (FL) with a rating of 4+.
Q14	Find the reviews of restaurants that have free wifi.
Q15	Find reviews that include the word 'love' (case insensitive) and were reviewed by a user who had elite status on the year of the review.
Q16	Identify the negative reviews (rating of 2.0 or below) from the year 2018 and only for businesses that have received at least 10 reviews overall so far.
Q17	Identify users who were active in 2018 (gave a review), but only for those that started yelping prior to that year.
Q18	Get the reviews of gyms that allow credit card payments and were rated by users who have atleast 10 reviews so far.

Table 5.2: OLAP Queries Overview

Chapter 6

Benchmarking, Results and Analysis

6.1 Benchmarking Methodology

Benchmarking involves comparing performance indicators and processes to industry best practices usually in relation to time, quality and cost metrics. It is generally used to estimate similarities and contrast between a specific performance metric.

In this project, the following approach was followed:

- Python scripts were developed to run all the queries against each database tool.
  - Arango was connected via Python library of pyArango.
  - Marklogic was connected with Python using REST API.
- All the 18 queries were run in sequence for each scale factor. This was iterated 3 times and the average run time of the queries was considered as the actual run-time for that query.
- Results were then integrated together and performance analysis was performed for both DB tools, which is detailed in the further section.

6.2 Results and Analysis

This section discusses the results and analysis for all queries, CRUD Queries as well as OLAP queries.

6.2.1 Overall Results

As per the figure 6.1, interestingly Arango starts off much better with the smaller SFs, performing better, but the gap closes in from SF 5, and Arango shows poorer performance in the last SF, which may suggest that MarkLogic works better with larger volumes of data.

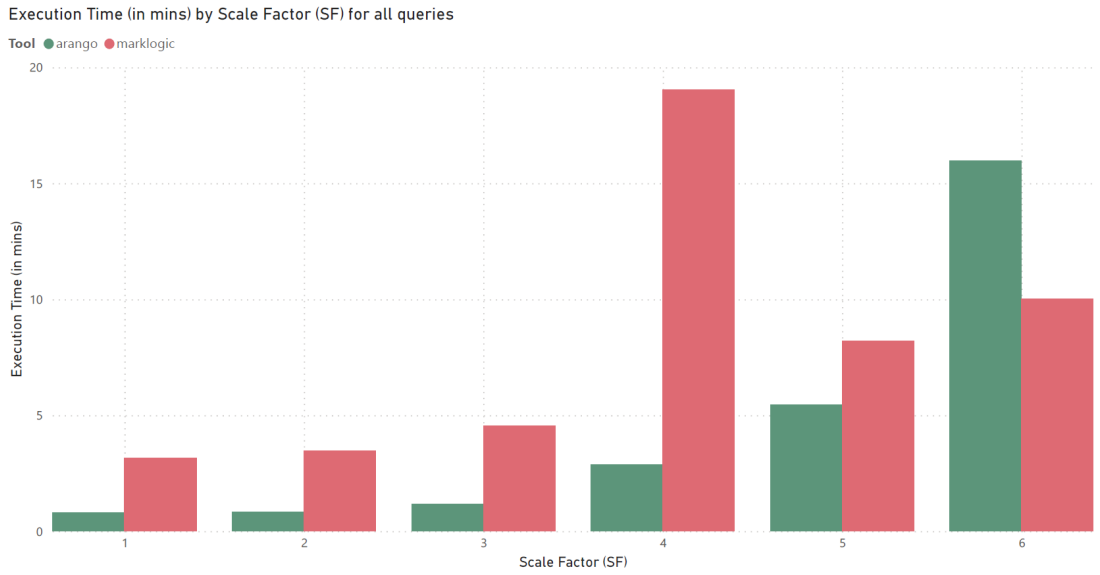

Figure 6.1: Total run time in mins for all queries

6.2.2 CRUD Results

The total runtime of all CRUD queries is shown in the figure 6.2.

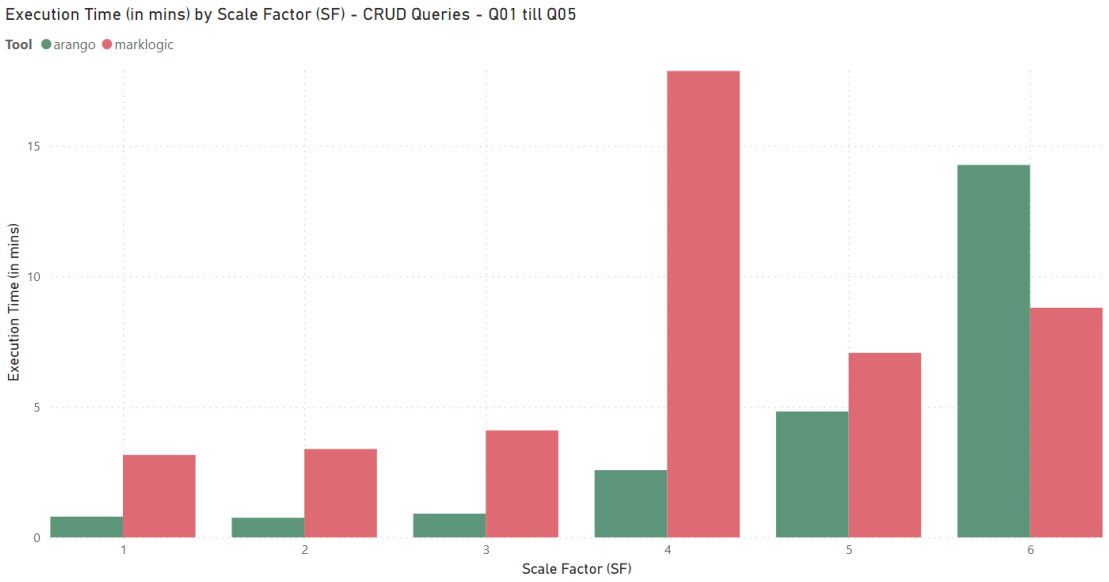

Figure 6.2: Total run time in mins for all CRUD queries

This runtime comparisons is very similar to the total run time of all 18 queries (similar pattern). To understand it a bit further, let's explore each query individually for its run-time against each database tool.

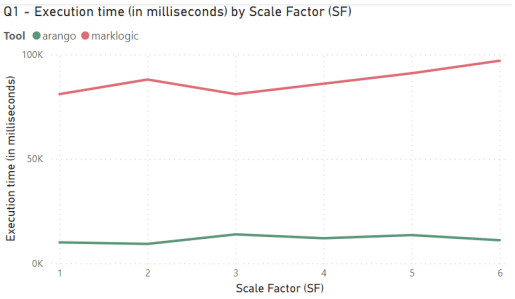

Figure 6.3: Query 01 - Average Runtime (Mins) by Scale Factor (SF)

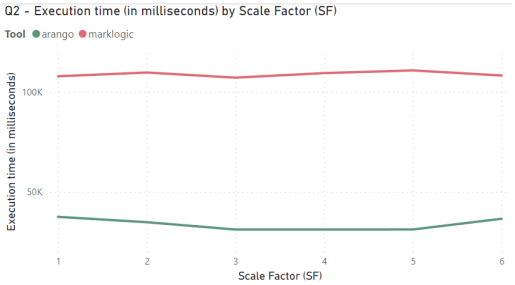

Figure 6.4: Query 02 - Average Runtime (Mins) by Scale Factor (SF)

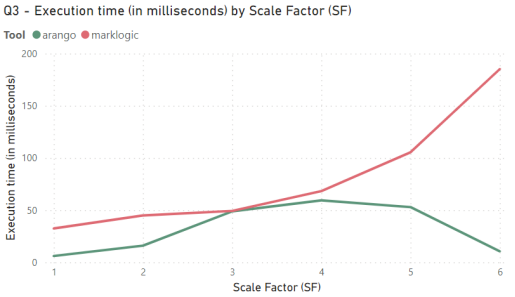

Figure 6.5: Query 03 - Average Runtime (Mins) by Scale Factor (SF)

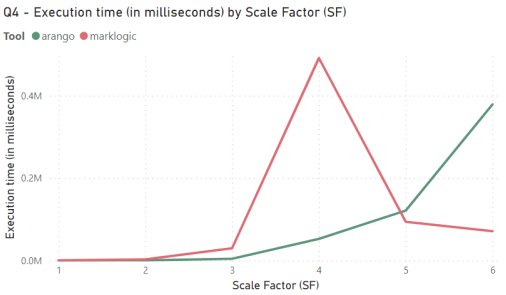

Figure 6.6: Query 04 - Average Runtime (Mins) by Scale Factor (SF)

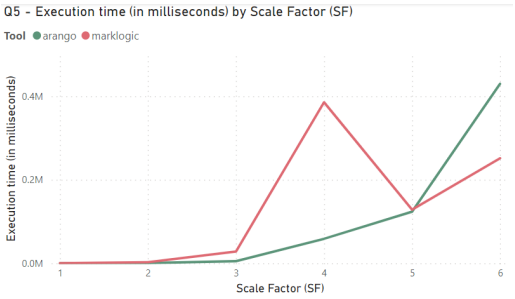

Figure 6.7: Query 05 - Average Runtime (Mins) by Scale Factor (SF)

Figures - 6.3 till 6.7 represent the run-times of all 5 CRUD queries in Marklogic and Arango. Some key insights obtained from these results are:

- Arango manages to perform both large inserts and deletes at a much faster pace as compared to MarkLogic, and similar observation can be seen with read operations.
- Interestingly, with update operations (Q4-5), MarkLogic starts to perform better from SF5 onwards.

6.2.3 OLAP Results

The total runtime of all OLAP queries is shown in the figure 6.8.

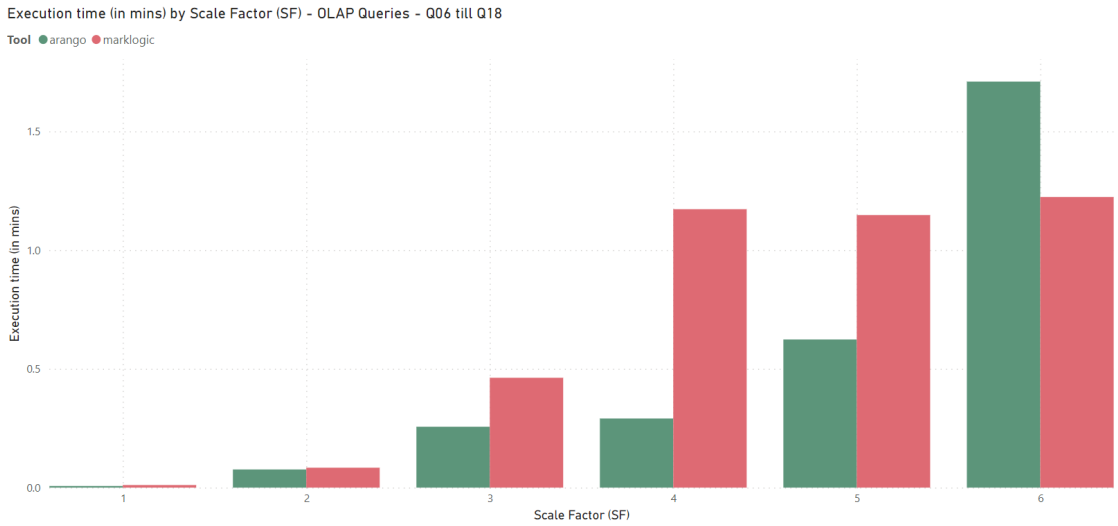

Figure 6.8: Total run time in mins for all OLAP queries

To understand the pattern of this runtimes, let's explore each query individually for its run-time against each database tool.

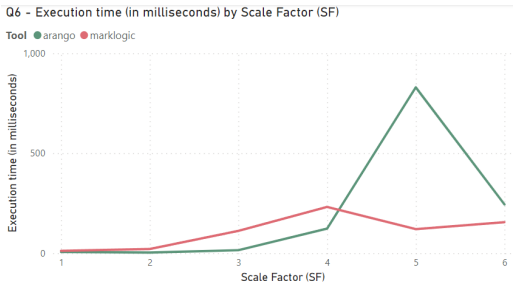

Figure 6.9: Query 06 - Average Runtime (Mins) by Scale Factor (SF)

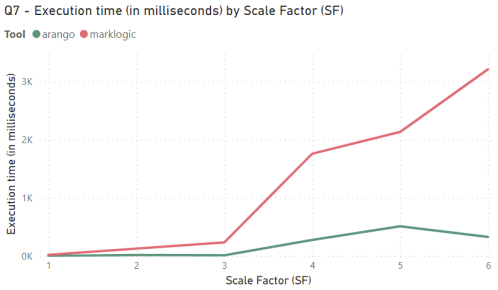

Figure 6.10: Query 07 - Average Runtime (Mins) by Scale Factor (SF)

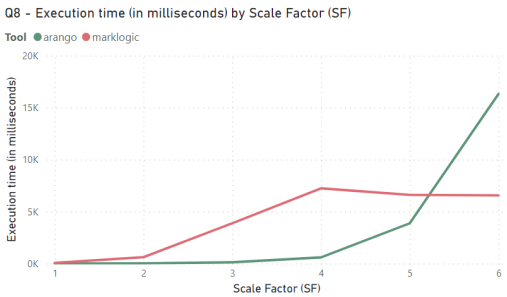

Figure 6.11: Query 08 - Average Runtime (Mins) by Scale Factor (SF)

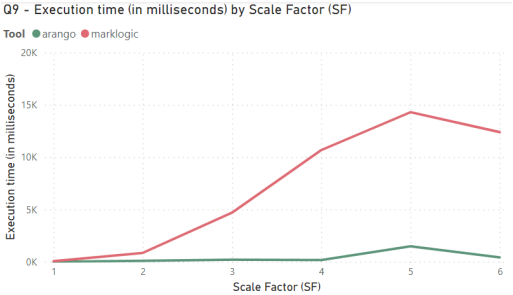

Figure 6.12: Query 09 - Average Runtime (Mins) by Scale Factor (SF)

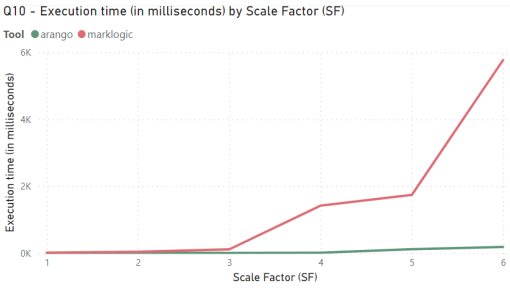

Figure 6.13: Query 10 - Average Runtime (Mins) by Scale Factor (SF)

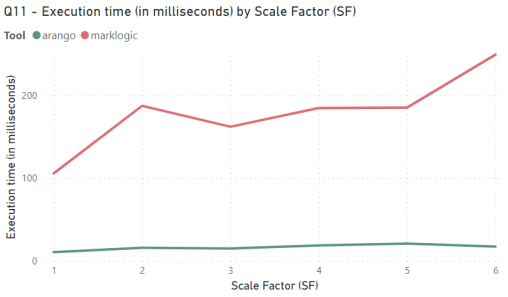

Figure 6.14: Query 11 - Average Runtime (Mins) by Scale Factor (SF)

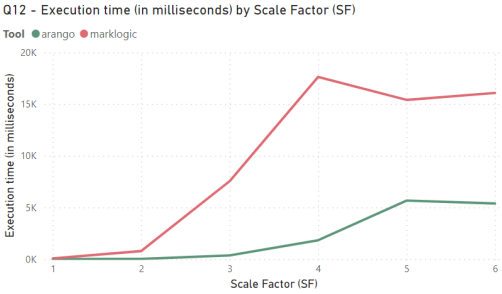

Figure 6.15: Query 12 - Average Runtime (Mins) by Scale Factor (SF)

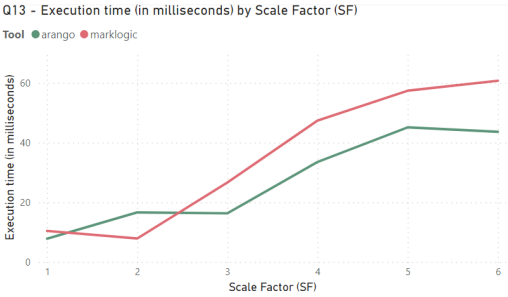

Figure 6.16: Query 13 - Average Runtime (Mins) by Scale Factor (SF)

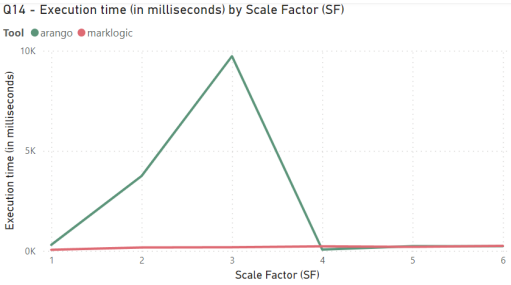

Figure 6.17: Query 14 - Average Runtime (Mins) by Scale Factor (SF)

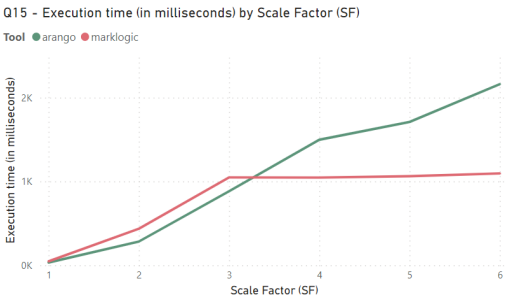

Figure 6.18: Query 15 - Average Runtime (Mins) by Scale Factor (SF)

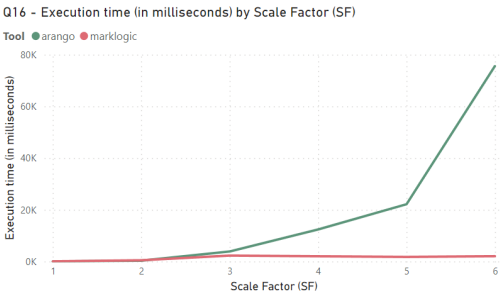

Figure 6.19: Query 16 - Average Runtime (Mins) by Scale Factor (SF)

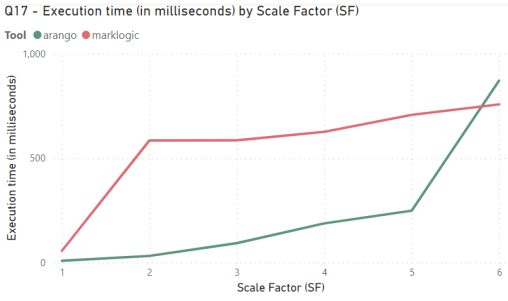

Figure 6.20: Query 17 - Average Runtime (Mins) by Scale Factor (SF)

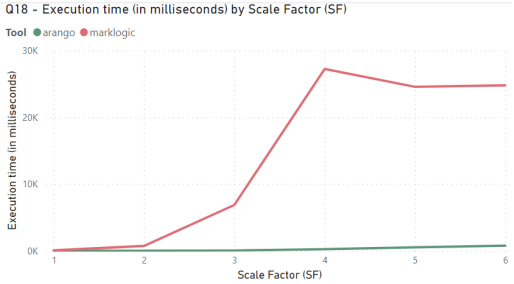

Figure 6.21: Query 18 - Average Runtime (Mins) by Scale Factor (SF)

Figures - 6.9 till 6.21 represent the run-times of all 13 OLAP queries in Marklogic and Arango. Some key insights from the results are:

- Arango outperforms MarkLogic by a large margin, with queries that involve normal filters, joins and aggregation.
- MarkLogic starts to shine when it comes to queries that involve word searches. For example Q15: identifying key words from the review text.
- Furthermore, Arango does seem to struggle a bit when it comes to nested sub-queries (Q16)

Chapter 7

Conclusion

In conclusion, this study has given us well-rounded insights into the document oriented technology using Arangodb and Marklogic. This technology was applied over the Yelp dataset which is a JSON-like format document suitable for the aforementioned tools. Some CRUD and OLAP queries were executed at different scale factors to help perform benchmark. In comparison, arango seems to outshine MarkLogic in majority of the scenarios within the use cases we have tested in terms of CRUD and OLAP queries, however, MarkLogic does show better performance in some specific cases, such as word searches and update operations on large data volumes.

In general, document database are usually schema-lees which make it easier and more efficient to maintain since the models do not change and the documents only need to be updated. Also, arangodb and marklogic mininimizes complexity of the technology stack as it allows a multi-model approach. However, it is essential to understand the performance scaling aspect when considering a suitable engine as this study shows the difference in performance between arangodb and markogic as the scale factors increased.

Bibliography

[DB-Engines, 2022] DB-engines. (2022). DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. Retrieved December 7, 2022, from https://db-engines.com/en/

[Hunter, 2013] Hunter, J. (2013). Inside MarkLogic Server. MarkLogic Corporation. https://www.yumpu.com/en/document/read/55746900/marklogic-server

[INFO-H-415: Advanced Databases, 2022] INFO-H-415: Advanced Databases. (2022, November 15). INFO-H-415: Advanced Databases [Université Libre de Bruxelles - Service CoDE - Laboratoire WIT]. Retrieved December 5, 2022, from https://cs.ulb.ac.be/public/teaching/infoh415

[Kubiak et al., 2012] Kubiak, W., Rusinkiewicz, M., Blazewicz, J., & Morzy, T. (Eds.). (2012). Handbook on Data Management in Information Systems. Springer Berlin Heidelberg. 10.1007

[Yelp, 2022] Yelp. (2022, February 5th). Yelp. Retrieved December 5, 2022, from https://www.yelp.com

[Yelp Dataset, 2022] Yelp Dataset. (2022, February 5th). Yelp. Retrieved December 5, 2022, from https://www.yelp.com/dataset

[Zimanyi, n.d.] Zimanyi, E. (n.d.). INFO-H-419: Data Warehouses. INFO-H-419: Data Warehouses [Université Libre de Bruxelles - Service CoDE - Laboratoire WIT]. Retrieved October 27, 2022, from https://cs.ulb.ac.be/public/teaching/infoh419

[Introduction to ArangoDB's Technical Documentation and Ecosystem, 2022] Introduction to ArangoDB's Technical Documentation and Ecosystem. (2022, January). ArangoDB. Retrieved December 8, 2022, from https://www.arangodb.com/docs/stable/