Analysis of Resource Description Framework and Web Semantic in Big Data

V.Shanmugapriya [1*], Dr. D. Maruthanayagam[2]
[1]Research Scholar, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India.
[2] Head/Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India.

Abstract: Big Data Movement is considered to convert unstructured and semi-structured data on the files to semantically connected documents known as the "web of data". Resource Description Framework **(RDF)** represents the semantic data and a collection of such statements shapes an RDF graph. **SPARQL** is considered as a query language designed to query **RDF** data. Managing large amount of data scalable is a matter of concern for a long time. Same process is true for semantic web data. Current semantic web frameworks lack this ability. ***Big Data identify huge amounts of data in a connected form. Certainly, utilizing Big Data continue to be in high demand***. Hence, a scalable and accessible query system requires for the availability and reusability of existing web data. RDF model is the database community using several techniques to store and query the data (RDF) in an efficient manner. But, current RDF database bear from many issues, such as poor routine behavior for querying RDF data. ***In this paper, a comparative analysis provides the selective RDF databases storages and a persistent access and storage of RDF graphs.***

Keywords: Resource Description Framework, Corporate Semantic Web, RDF Query Languages, Sesame, Jena, Semantic Web Database, Big Data

I. INTRODUCTION

Scalability is a major trouble in Information Technology environment. Basically what it indicates is that a system can handle addition of large amount of data, tasks, and users and also without affecting its performance significantly. A trivial task is not easy to design a scalable system. It can also apply to systems handling huge data sets. There is no exception to semantic web data repositories. Saving massive set of RDF triples and the ability to query RDF data is a challenging issue that cannot be solved yet. Trillions of triples requiring peta bytes of disk space are not possible a distant to any further extent. Researchers are processing on millions of triples [1]. Competitions are being organized to encourage researchers to make efficient repositories [2]. ***Hadoop is a distributed file system*** in which files can be stored with replication (duplication). It offers high fault tolerance and reliability. Additionally, it provides an implementation of MapReduce programming model. ***MapReduce is considered as a functional programming model*** that has apt for processing huge amount of data in parallel way. MapReduce processes run on independent amounts of data making parallelization much easier. MapReduce is an evolving technology and it has been well received by the community that handles huge amount of data. Google uses it for social networking, data storage, and web indexing [3]. It is used to scale up classifiers for mining peta-bytes of data [4]. In different forms, data mining techniques are being rewritten to take the benefit of MapReduce technology [5]. Biological data is being analyzed by using MapReduce technology [6].

In recent times, the Semantic Web [7, 8] technologies have been increasingly used for provenance management because of their semantics support and flexibility performance [9], such that provenance metadata is represented and captured through RDF (Resource Description Framework) [10], OWL (Web Ontology Language) [12], and RDFS (RDF Schema) [11], and queried using the SPARQL [13] query language. RDF is a W3C (World Wide Web Consortium) suggestion for the notation of meta-data on the WWW. A semantic extension of RDF is that a RDF Schema by providing developers to model object structures and to specify vocabulary. The enrichment of the Web will allow the machine-processable semantics, so that giving rise to the preference what has been dubbed in the Semantic Web. An RDF data set is a group of statements which is known as *triples* in the form (*S*, *P*, *O*), in which *S* represents a subject, *P* considers a predicate and *O* indicates an object. Each triple states the relation (represented by its predicate) between its object and subject. A set of triples represent a labeled directed graph, with nodes represents subjects and objects and labeled edges representing predicates, to establish the communication between the subject nodes to object nodes. A triple store is considered as a framework used for querying and storing RDF data [6]. Semantic web databases (a.k.a triple stores) are designed to hold huge semantic web data in such a manner that the data encoded and retrieved in an efficient manner. Semantic web databases have different architectures that result in varying performance levels. Depending on their storage structure and medium, semantic web databases can be partitioned into three broad kinds: main memory databases (i.e., process and store data in main memory), ***non-native non-memory databases*** (i.e., save data on a disk by employing some RDBMS), ***and native databases*** (i.e., permanently data stored on a disk in file format). ***The set of triple stores has been enhanced considerably from Sesame and Jena*** in the early 2000s to BigData, Jena SDB, YARS2, Virtuoso, Jena TDB, and AllegroGraph, Sesame, Mu lgara, 3Store, Kowari, and RDF Gateway, where Garlik and YARS2 are not distributed [14]. AllegroGraph are commercially available and the others are available on open sources. To check on the freely available open source triple stores use and the choice of Allegrograph, Sesame, and Jena API. ***The main challenges of semantic databases (web) are* (i) performance,** i.e., to execute various data management tasks ***i***n a user interactive time by utilizing lowest amount of system resources, and **(ii) scalability**, i.e., to handle the expansion of data, as the number of triples maximizes quickly to model a piece of information in semantic web. It is significant to evaluate the pre-existing semantic web databases because developers of each one of them claim to perform the best. The evaluation (a.k.a benchmark) exhibits the shortcomings of the pre-existing semantic web databases. In this paper, to compare and present two techniques of realizing a mapping between a database and ontology: Sesame [14] and Jena [7]. The evaluation of existing Semantic Web databases are mainly used to their comparative behavior and scalability trends for our proposed evaluation methodology and to analyze their storage schemas.

II. RDF (Resource Description Framework)

RDF is a formal W3C (World Wide Web Consortium) recommendation that it has initially designed as a metadata data model and also used for modeling of information or conceptual description which is implemented in web resources, using a choice of data serialization formats, syntax notations, and knowledge management applications. The RDF description (it refers to "data about data" or metadata) can involve information that describes content in terms of content rating, the authors of the resource, date of updating or creation, the organization of the sitemap (the pages on a site), and key words for search engine data collection. RDF will make it possible to share Web site and other descriptions more without problems and for software developers to make products that can employ the metadata to give better directories and search engines, to act as intelligent agents, and to offer Web users more organize of what they are viewing. An **XML (Extensible Markup Language)** is widen under the auspices of the W3C using the RDF.
An Internet resource is described as any resource with a Uniform Resource Identifier **(URI)**. It consists of the Uniform Resource Locators **(URL)** which recognizes the entire Web sites and specific Web pages. A Web page consists of the RDF description statements, **HTML META** tags, encased as part of XML (Extensible Markup Language) section. RDF is a formal W3C Recommendation, meaning that it is ready for general utilization. A second **W3C** recommendation, it proposes a system wherein the descriptions associated to a particular purpose (for instance, all descriptions related to privacy and security) and it constitute a class of such that descriptions (using class here much as it is used in object-oriented programming data modeling and programming) still at the Proposal phase.

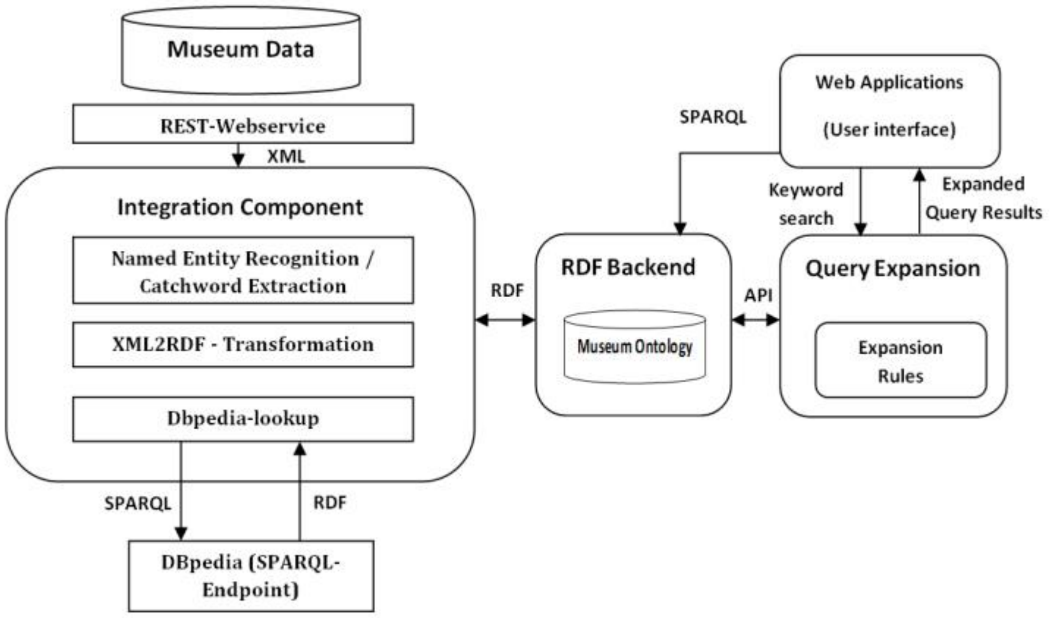


Figure 1: Corporate Semantic Web

The main growth of World Wide Web and its usage development, it will keep on produce ever more structure, content, and usage data and the value of Web mining will continue growing. Research requires to be done their measurement procedures, the right set of Web metrics, understanding how different parts of the process model impact various Web metrics of interest, how the process models change in response to various changes that are made changing stimuli to the user, developing Web mining techniques to get better performances in various other features of Web services, techniques to know intrusion detection and frauds.
It is considered as a simple language description model which makes statements about a resource by either defining its attribute or by defining its relationships with other resources. The relationship is the predicate, the resource is the subject, and the attribute value is the object [15]. If this is visualized from an object oriented perspective, an RDF “Subject - Predicate - Object” can be mapped to an “Object - Property - Value” relationship of an Object. But, the RDF not depends on the object oriented paradigm. It does not afford encapsulation of attributes. This provides flexibility to add statements about a resource without having to change any encapsulating entity associated with it. For instance, “:bob :drives:car” is an RDF statement, in which :bob is the Subject, :drives is a predicate and the :car is an object. The RDF can be imagined as a graph in which the Subject and the Object are main two nodes, and the predicate defines an edge between the two nodes in figure 2. These are known as Triples. By convention, the subject and the object values are always exposed as a node in the RDF graph. ***RDF follows URI based vocabulary and node identification***. Thus the resources can be identified by URIs or by Blank nodes. The concrete values can be literals along with the data-type specification. The Subject, Predicate and the Object (S,O,P) can all be Blank nodes or URIs, however only the Object can be a literal [16].

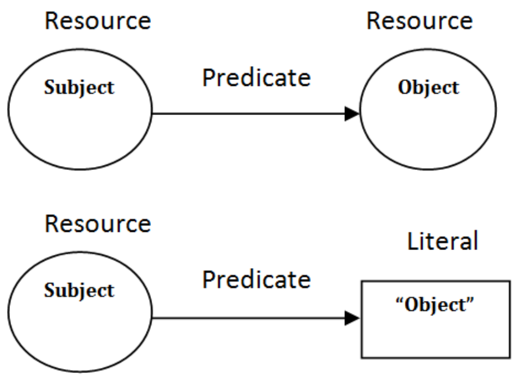


Figure 2: "Subject - Predicate - Object" Graph

The literals are always indicated by a rectangular box in an RDF graph. Therefore, the Object can be represented by a Box or a Node in an RDF graph. The literal value is labeled by the box. A complex RDF dataset and its corresponding graph visualization illustrate in **Table 1 and Figure 3.**

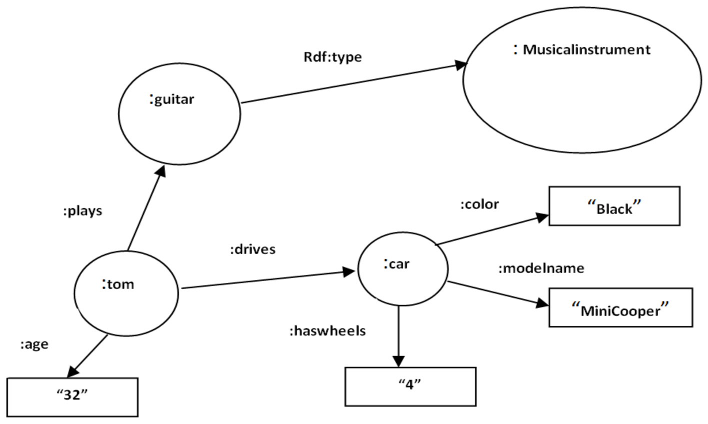


Figure 3: A complex RDF graph

The RDF triples are defined by various serialization formats. RDF/XML, Notation-3, Terse RDF Triple language (Turtle), N-triples are some of the popular formats. Website owners eager to make their web pages with machine-readable form should be provide an RDF version of the web pages with the HTML data. The specifications such as

Table 1: An RDF Dataset

RDF DataSet

```
-----------------------------
:tom :drives :car.
:tom :age "32".
:tom :plays :guitar.
:guitar rdf:type :Musicalinstrument.
:car :haswheels 4.
:car :modelname "minicooper".
:car :color "black".
-----------------------------
```

The RDF [17] and the Micro formats must be followed to embed the RDF data into web documents. RDF parsers (or scrapers) handle and pullout the required RDF relationships from the pre-existing HTML pages. The RDF data know how to be put in the meta-data part of an HTML page, or can be embedded as an attribute in the HTML tags. Therefore, Micro formats tags and the RDF co-exist with the HTML tags and it does not modify the presentation of web document.

2.1 Overview of RDF Query Languages

RDF querying considers on three levels of abstraction:

- Syntactic level (XML documents).
- Structure level (a set of triples).
- Semantic level (graphs with partially predefined semantics).

- In syntactic level, it seems possible to query RDF using XML query languages such as XPath [18] and XQuery [19]. The XML tree structure becomes very hard to query in the RDF data model. Other than, the RDF data model is a graph, and moreover, both its edges (properties) and its nodes (subjects/ objects) are labeled. Thus, there are several different techniques to serialize the same content using RDF syntax [20].
- In structure level, RDF documents denotes as a set of triples1 (Subject, Object, and Predicate). The benefit of such a query is that it directly addresses the RDF data model, and that it is therefore independent of the specific syntax that has been chosen to represent the data. But, a disadvantage of any query language at this level is that it interprets any **RDF model** only as a set of triples, comprising those elements that have been specified a special semantics in **RDFS** [21]. For instance, information of class and property hierarchies described in RDFS is not exploited in such query languages (e.g., SquishQL [6] and RDQL [22]).
- Finally, at the semantic level, semantics of RDFS descriptions is also used for querying. For instance, a query that retrieves all instances of some class, will also retrieve instances of that class subclasses. An example of an RDF query language at this level is RQL [20].

There is no a standard language to query annotations and ontologies. In recent times, there have been proposed many RDF query languages, like RQL [23], SeRQL [24], Versa [25], N3 [23], TRIPLE [26], RDQL [19], SquishQL [24], etc. As a result of space constraints, it provides a very brief description of these languages, however provide better introduction to RQL. **RQL** is an **OQL**-like typed functional language, and that describes a set of basic functions (queries) and iterates and relies on functional composition to construct more complex queries. One of the RQL distinguishing characteristics is internal support of schema (RDFS) queries and smooth combination of schema and data querying. RQL depends on interpretation of the RDF graph. SeRQL is an attempt to design more powerful and easy-to-use querying and transformation language based on existing ideas of RDQL, RQL, N3, etc. Syntactically, it is similar to RQL; but it is based on interpretation of the RDF Model Theory. Versa acquires an interesting approach in the main building block of the language is a list of RDF resources. RDF triples play a role in the so-called traversal operations [25]. N3 gives s an optional notation (text-based syntax) to express RDF triples, as well as capability to state rules and queries. N3 does not differentiate between queries and rules. N3 query language depends on the RDF data model. TRIPLE derives from F-Logic, such that RDF triples (*Subject, Object, Predicate*) are represented as F-Logic expressions *Subject* [*Predicate* → *Object*]. TRIPLE is close to N3, as they both "rule-oriented". RDQL has a SQL-like syntax similar to RQL and SeRQL. RDQL does not give support for schema (RDFS) queries. RDQL depends on RDF graph. SquishQL is close to RDQL in its SQL-like syntax, interpretation of data model and capabilities. Research on SquishQL is discontinued. More research on standardization and improvement of query languages must be done, as even the leading language, RQL, was not suitable for some query types. As a final point, authors [27] include aggregation, grouping, sorting, etc. to the "wish list" of RDF query language features. In [28], similarly provide detailed theoretical evaluation of expressive power for seven RDF query languages including TRIPLE, SquishQL, RQL, Versa, etc.

2.2 Retrieval and Storage of Large RDF Data

The development of the Semantic Web, there is a need for a specialized system. This system must be capable of quickly processing massive amounts of data, and must also generate more knowledge from the existing data. This knowledge base will be useful only if it can answer user queries in a fast manner. Other than, present platforms and techniques are providing centralized data retrieval and data storage. These data stores provide web interfaces with the intention that available data know how to be queried from anywhere on the web. Other than, centralized approaches cannot scale above millions of triples and instead need systems that adopt a decentralized processing architecture. In parallel processing, a system that utilizes over parts of the data would afford faster response. The data must be accurately shared to different processors to make this feasible, those can be multiple nodes or multiple processors in a cluster. Multiprocessors usually work on a single node and thus data transportation is not required.

Other than, a smart way to distribute the data amongst the processors is needed. However, if the data is distributed over multiple nodes, data transfer across the network should be minimized, since this usually becomes the bottleneck. Distributed storage is used for the data that is geographically spread over the web in a natural fit. To distribute data over the cluster is crucial in designing effective strategies. Compression, Partitioning/ Sharding, Data transportation and Filtering are considered as the four main categories related to distributed data and also carefully addressed while working on any distributed system. While designing a distributed query engine for RDF data, these points must be carefully revisited:

- **Partitioning**: Data sharding / partitioning must be performed in a way which is most suitable for processing and retrieval. Based on the data characteristics it can be partitioned in different ways. Range partitioning partitions data into groups that represent different ranges. For example, a file consists of numbers in range of 0 to 100 that can be partitioned into groups of 0-9, 10-19, 20-29 and so on. Hash or Key partitioning utilizes a hash function that returns a group id to which the data should belong. The data can be partitioned on size by splitting into several files of equal size. In different geographical locations is naturally distribute when the data that is gathered from sources.
- **Compression:** To compress the data to reduce the overall size of the corpus the system is working on. Data is uncompressed while reading. There are techniques like Dictionary encoding, in which the semantics of data is not lost. This process is used in a lossless compression technique. To make use of this technique to compress RDF data.
- **Filtering**: When a query is imposed on a shared data and requires a good filtering system must be employed, part of the complete dataset for query processing. The filtering technique must retain the data which is useful and ignore the rest. The filtering must be as fast as possible. Building indexes which point to the required data is one of the most popular ways. The indexes are built with the intention to filter data in constant time.
- **Data transportation**: Usually computation is low cost even if it takes place near the data. Data transportation on the network does not pay well in terms of latency. Therefore, a distributed system usually performs better if it exploits data locality.
- **Pre-computation**: Computation is costly and storage is low-priced, as a result it is preferable to store pre-computed data, if it assists to minimize computation during runtime.

III. Semantic Web Databases

Semantic databases are considered as RDF databases in which semantic data can be conveniently stored, operated and retrieved [29]. A triple store can be termed as "**A system to provide a mechanism** for persistent storage and access of Semantic Web graphs." Its main functions include reasoning, storing, and querying Semantic Web data. They employ index structures, algorithms for buffering, and concurrency control for optimal query processing and reasoning. An intelligent query optimizer in a triple store strives to store resources in terms of memory space and time consumption in query processing and reasoning.

3.1 Design goals of Semantic Web databases

The eminent design goals of Semantic Web databases are given below:

- **Scalability:** Resources are described on the Semantic Web in terms of triples. A resource may require several triples for its perfect description. It is necessary for Semantic Web databases to deal with a large number of triples in a graceful manner.
- **Dynamism and Network Distribution:** Data on the Semantic Web is dynamic as it belongs to different sources due to network distribution. Semantic Web databases can be used as a client or a server to handle timeouts, bandwidth utilization, network failure and deal with denial-of-services (DoS). Semantic Web databases must be able to handle the network resources in both a client or a server and deal with dynamic data in an elegant way.
- **Unpredictability:** Semantic data is highly unpredictable in its nature. A Large number of triples, dissimilar terms used to describe resources, rate of triples exchange over the network and effect of network provide a high degree of unpredictability to the data. Semantic Web databases need to handle this unpredictable nature of data in an efficient and accurate manner.
- **Provenance:** The Semantic Web data comes across different sources that can necessitate keeping track of original location or context of the data. This part of information is known as ***provenance***. Semantic Web databases may need to store the context, along with the original data.
- **Data Processing:** In the Semantic Web databases, data processing requires to be processed. This necessitates Semantic Web databases to afford some mechanism for accessing the RDF graphs, storing triples in data stores, identifying triples, merging data from multiple sources into single store, and querying and administering the data stores. These operations are performed by applications several times so that require Semantic Web databases to give fast, lightweight, easy to use and understandable APIs to perform these tasks. Many of the existing Semantic Web applications interact with human. So, it can perform as accurate and fast as possible so as to provide shield against frustration. Interactive level performance is one of the key requirements of these stores for human friendliness.
- **Reasoning:** A final clear problem is support for inference. Semantic Web databases support different degree of reasoning (RDF/RDFS/OWL) while RDFS support is common. A very few stores support OWL features, and they do not provide the performance measures while using these features whereas simpler systems do.

3.2 Types of Semantic Web databases

Different Semantic Web databases have different architectures thus result in varying performance levels. Semantic Web databases can be separated into three broad categories, ***In-memory, Native, Non-native Non-memory***, depending on their storage medium and structure. List of existing large triple stores consist of Garlik, 4store, BigOWLIM, Bigdata(R), Virtuoso, YARS2, AllegroGraph, Jena TDB, Mulgara, Jena SDB, Jena with PostgreSQL, RDF gateway, Kowari, 3store with MySQL, Sesame [30].

A. Jena

The software producers of Jena [31] are the HP Labs [32], which are a part of the Hewlett-Packard Development Company. This department was founded by Bill Hewlett and Dave Packard in 1966. Jena was developed in the terms of the HP Labs Semantic Web Research. The associated license of the Jena project is completely open source. This implies that re-distribution and use in binary forms and source with or without alteration are permitted [33]. The Jena product documentation can be establish on the project page and is broadly complete. The documentation covers the central parts of Jena providing basic information about the framework, Javadocs and many tutorials respectively How Tos. Moreover, the downloadable version of Jena consists of code instances, and that underline the basic steps in the working process of Jena. The support focuses on a newsgroup [34], and that is founded in the Yahoo! Groups [35]. It can be considered unsatisfactory which support is primarily limited to a newsgroup. Because of the fact that there is a large amount of registered members the activity of the newsgroup and therefore the delivered support is excellent [36]. The Jena download package consists of the source files of the entire Jena project implemented in Java. A basis for implementations extends the framework, for instance with new indices. An architectural overview of Jena illustrates in figure 4. The framework offers methods to load RDF data into a memory based triple store, a native storage or into a persistent triple store. In order to build a persistent triple store a variety of relational databases, for instance MySQL, Oracle or PostgreSQL, can be used. The stored data may be retrieved through SPARQL queries. A standard implementation of the SPARQL query language is encapsulated in the ARQ package of Jena. SPARQL queries can be executed using Java applications or using of the graphical frontend Joseki [37]. The Ontology API provides methods to work on ontologies of different formats, such as RDFS or OWL. Jena's Core RDF Model API provides techniques to manipulate, create, read, navigate, write or query RDF data. The remaining major factors are provided on the Inference API, and that enables the integration of inference engines or reasoners into the system. Alternatively, the Reification API is a proposal to optimize the representation of reification.

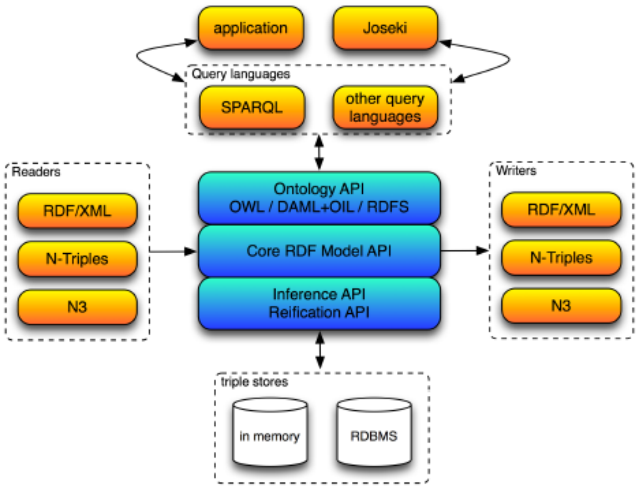


Figure 4: Architectural overview of Jena

OWL support is provided in form of the Ontology API. The inference subsystem [26] enables the use of inference engines or reasoners in Jena. Besides SPARQL, RDQL is a supported query language. In a tutorial about RDQL it is suggested that new users of Jena must use SPARQL instead. Jena makes use of writers and readers for RDF/XML, N3 and N-Triples, which are known as RDF data formats.

B. Sesame

The software producer of Sesame is Aduna [39] [40]. This company sets the focus of their work in revealing the meaning of information. Sesame was initiated as a prototype of the EU project. Aduna is developed On-To-Knowledge in the company of NLNet Foundation [41] [42]. As open-source license underlying the BSD-style is refer by Sesames associated license. The product documentation of Sesame is managed well. There is a large user guide available for each version of Sesame. Javadocs can also refer by the users and tutorials done with example code. Aduna provides support in the type of an active forum accessible on the project page and a mailing list depends on SourceForge [43]. Commercial consulting services are also provided. The Java source files are shipped by Sesame's download package. Hence, a basis for extending the framework is provided related to Jena.

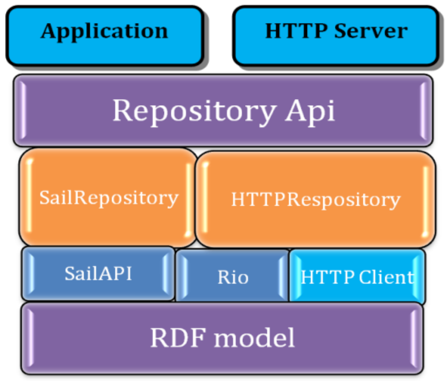


Figure 5: Architectural overview of Sesame

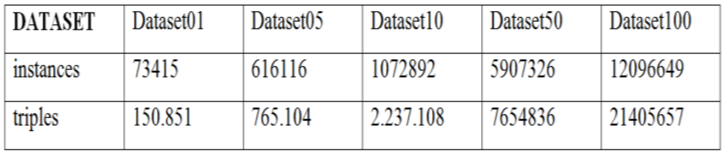
An architectural overview of Sesame shows in Figure 5. With the purpose of use Sesame, Apache Tomcat is suggested. The Sesame package also contains two web applications, the Sesame server which stores the RDF data and the OpenRDF Workbench as a graphical frontend for the server. This workbench can handle repositories, execute queries and load RDF data. Sesame is capable to manage all three N3. Triple, RDF/XML discussed approaches to save RDF data. The RDF Model implements basic concepts about RDF data. The component RDF I/O (Rio) contains a set of writer and parser for the managing of RDF data. For example, this is used by the SAIL (Storage And Inference Layer) API for initializing, modifying, querying, and the shutdown of RDF stores. On the topmost layer represents the Repository API the main entrance to address repositories. The Repository API is the associated high level API with a larger amount of techniques for handling RDF data, Compared to Sail, which is rather a low level API. The HTTPRepository is an implementation that acts like a proxy in order to establish connection to a remote Sesame server through the HTTP protocol. In order to attain OWL support a Plug-In is available known as BigOWLIM [44]. As a high performance semantic repository for Sesame and packaged as a Sail are implemented using BigOWLIM. Alternatively to SPARQL Sesame is able to interpret the Sesame RDF Query Language (SeRQL) integrated for boosting the functionality of RQL and RDQL. Sesame offers parsers for various well known RDF formats N3, RDF/XML, Turtle, N-Triples and two new formats TriX and TriG43.

IV. EXPREMENTAL RESULTS

A single machine is performed all experiments under 64-bit Enterprise edition of Microsoft Windows Server (2003) on the top of ACPI Multiprocessor (16 processors) X5550@ 2.67 GHz CPU. To perform this evaluation with the support of using 8 GB RAM and 120 GB PERC 6/I SCSI Disk Device with 8 GB virtual memory. All Java engines were executed with the support of Net beans IDE (6.8) in which run on the top of JRE 1.6.0_18 64-bit version. RAM Rush 1.0.5.817 was used as a RAM cleaner.

The LUBM database benchmark works borrows and shares. The ontology used in the benchmark is known as Univ-Bench. It defines departments and universities and their activities that arise at them. The LUBM benchmark was downloaded. The benchmark performance is evaluated with those repositories to extensional queries over a huge data set which commits to a single realistic ontology. It comprises numerous performance metrics, repeatable and customizable synthetic data, university domain ontology, and a set of test queries. ***To use the benchmark of Lehigh University (represented as LUBM) is to generate five datasets with respectively* 1, 5, 10, 50** and **100** universities (denoted Dataset01, Dataset05, Dataset10, Dataset50 and Dataset100 respectively).

Table 2: Generated Datasets



DATASET	Dataset01	Dataset05	Dataset10	Dataset50	Dataset100
instances	73415	616116	1072892	5907326	12096649
triples	150.851	765.104	2.237.108	7654836	21405657

Test Cases

Database community widely divides all database operations into ***four major Database communities*** broadly divides all database operations into four major categories, as given below;

- Load data to a data store **(Create)**
- Obtain existing data from a data store **(Read)**
- Modify existing data in a data store **(Update)**
- Remove data from a data store **(Delete)**

Using these four operations are collectively known as **CRUD** operations. To optimize a data store for some operations of a category can result in performance degradation of another category's operations. For example, multiple indexes can considerably decrease the data access time; but, can degrade the performance of create, update and delete operations. Each deletion or addition needs updating all the indexes so does each update operation also requires modifying an index key of indexes. ***Therefore, the performance testing results of a data store, for single category,*** are not reliable. All these operations need to be tested for any data storage system in order to obtain a clear vision of data store performance. On the basis of this argument, ***to divide our test cases into four different categories representing the CRUD operations.***

Metrics

A set of performance and scalability metrics are proposed that captures a variety of aspects of the space scalability and space time scalability evaluation of Semantic Web databases. While evaluating the performance of a Semantic Web database, consider **two** significant parameters as a measure of its execution cost for each proposed test case, i.e. ***resource utilization and execution time.*** Depending on the scenarios will consider ***CPU time, main memory, and secondary disk space*** as cost primitives of resource usage to an in-depth study of performance. The cost execution of operation on a Semantic Web database is manipulated by a number of aspects, including type of operation, number of triples in dataset, number of nodes in dataset, and size 68. Thus, record the execution cost for dataset sizes and different operations.

- ***Load Time:*** This metric provides loading time *T* for datasets of different sizes. Load time is evaluated as a cumulative time to construct a repository structure, build initial index structures and generate statistics about a dataset for query optimization.
- ***Query Response Time:*** This metric affords query execution time *T* for every test carried on datasets of different sizes. Query execution time consists of time to connect to a repository, execute the query, print its result set and then close the connection.
- ***Main Memory:*** This metric computes memory needed for execution of a particular test case. This is the amount of memory required to hold working buffers. Small size of main memory is a limit on size of the semantic data that can reside or process in main memory. To analyze and consider the maximum committed memory for a task as memory used to determine each task.
- ***CPU Time:*** This metric describes CPU time consumed for processing a particular test case. Each time a test case is executed by a user, it keeps some of system resources among which CPU time is significant, because the higher the percentage of a CPU used by a semantic store, the lesser the power the CPU can devote to other tasks.
- ***Repository Size:*** This metric provides storage size *S* occupied by a dataset after loading it into persistent storage. Repository size is a composite figure of the total size of the entire files present in the repository, including index files and data files.
- ***Success Ratio:*** Success ratio explains the fraction of successfully executed test cases either for a dataset or for a semantic store. For success ratio computation, to considered one bulk load test case and six read test cases to develop a better effect of failed tests.

Load Time:

With large RDF files generated, the import test data in triple store will provide a particular view of data loading performances. Each and every system has a particular mode of organizing data indexing that impacts the data loading mechanism. A few triple stores permit users to set various configurations like peak memory used, the order of priority for the index, field's indexes, and so on. In such case, the systems cannot directly load large files (for example: with Sesame, Virtuoso). In that case, a system has been set up particularly to divide the files in smaller chunks. Other systems such as Stardog, Fuseki, and GraphDB provide tools to capable load large files. ***As results of benchmark data on load time are achieved with the best time(less than Jena) to Sesame at the same time as the Jena system is the slowest***. In Jena the import is performed in RDBMS tables using ODBC protocol, while in the case of Sesame import does not require processing for the storage because of its tree structure.

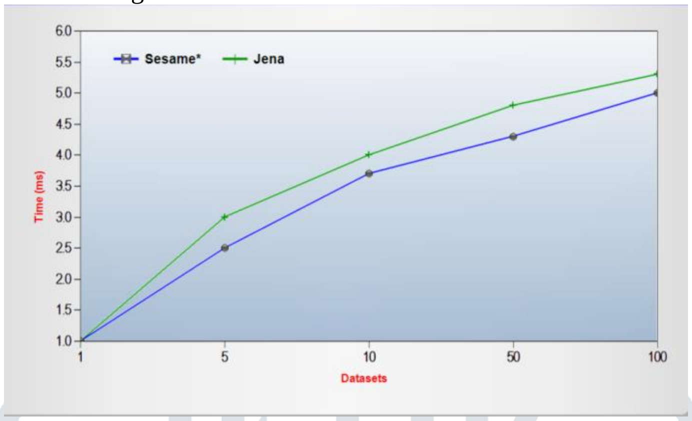


Figure 6: Comparison of loading times on triple stores

Query Response Time:

On the topic of the queries response times, ***Sesame responds better than the other Jena for most queries executed on LUBM illustrates in Figure 7.*** Certainly, Sesame uses the horizontal layout and that queries involve many property, it performs less join operations than other Jena. Indeed, all queries on properties of a same class can be reduced to selection operations on table corresponding to this class, which is not the case when use another layout. Dataset10 of the benchmark LUBM is a better illustration. This Dataset is made of 2.237.108 triples patterns having all the same domain. This Dataset does not need a join operation, but it needs at least four join operations in other systems. If the query involves less property, the query response time is close to the Sesame that utilizes a binary layout as this layout also needs a single scan of a unique property table.

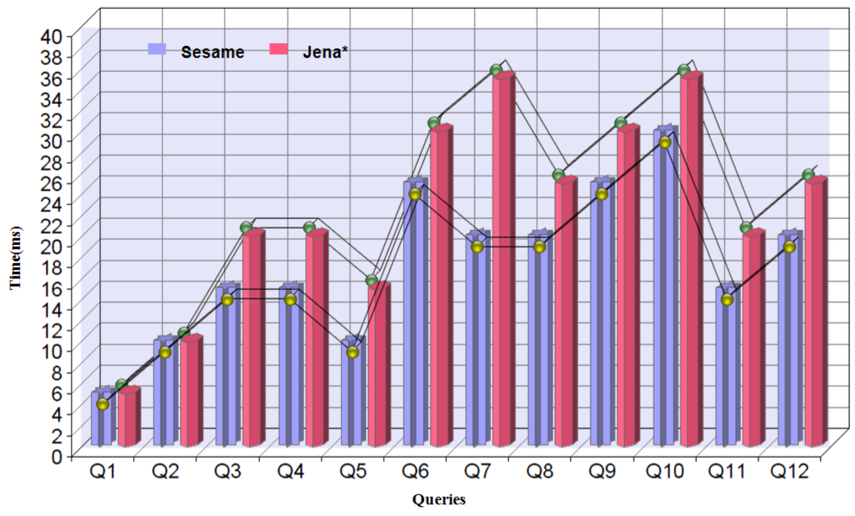


Figure 7: Queries response time on LUBM

Main memory

Usage of main memory during bulk load is improved almost linearly for in-memory stores. But, for all native and non-memory non-native stores, it is raised in polynomial times as illustrated in Figure 8. Here, y-axis and x-axis signify our datasets and main memory (i.e., in MB), respectively. **Sesame has performed better utilization of memory compare than Jena.**

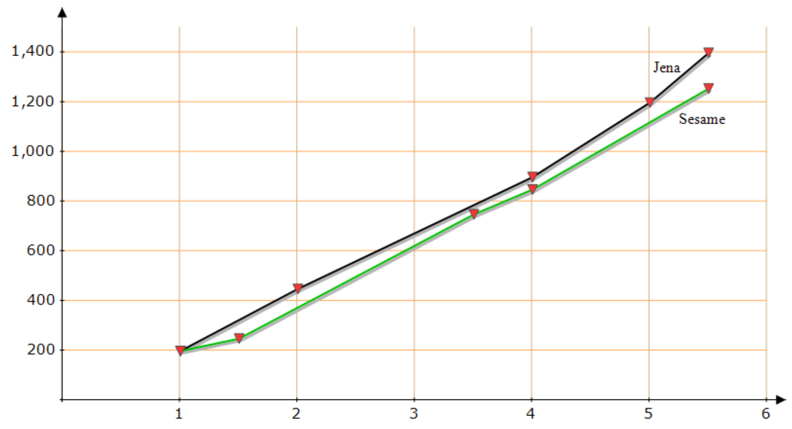


Figure 8: Comparison of memory utilization

Resource utilization

Resource utilization of in-memory, native and non-memory non-native stores for ***all read test cases is exposed in figure 9***, respectively. In that figure 8 illustrates memory utilization and figure 10 depicts CPU time utilization. To conclude that **Sesame is performed as a better resource utilization system to compare than Jena** as shown in Figure 9 in- memory category results.

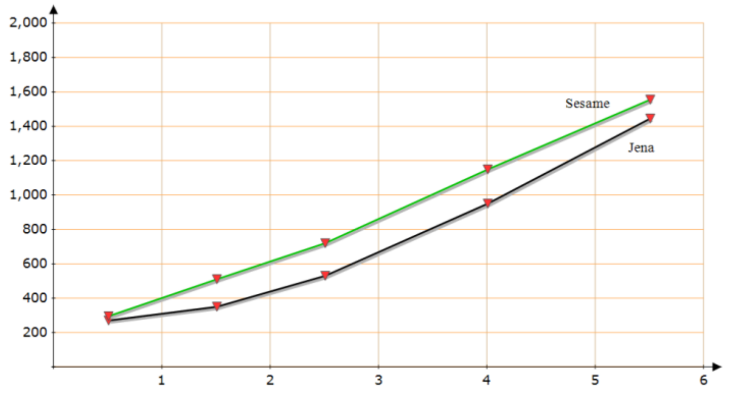


Figure 9: Comparison of Resource utilization

CPU time

CPU time for main memory stores is maximized in polynomial time; but, for native and non-memory non-native stores, ***it is increased exponentially at some stage in read tests as shown in Figure 10.*** Here *x*-axis corresponds to our datasets and *y*-axis represents the CPU time (in seconds variation). Figure 10 concluded that **Sesame has performed better CPU utilization time than Jena.**

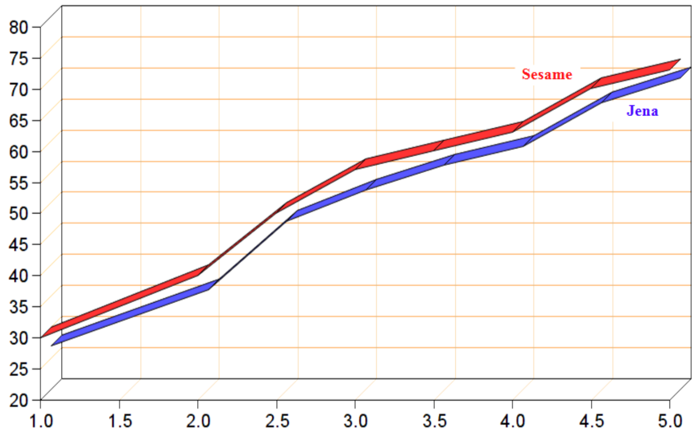


Figure 10: Comparison of CPU Utilization

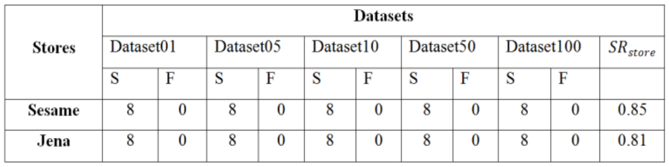
Success Ratio

The success ratio is evaluated using the formulas for each store and dataset size given below equation respectively,

$$SR_{store} = \frac{\sum_{i=1}^{4} ST_{dataset_i}^{store}}{\sum_{i=1}^{4} TT_{dataset_i}^{store}}$$

where SR_{store} and $SR_{dataset}$ are success ratio for store and dataset, respectively, $ST_{dataset_i}^{store}$ represents a number of successful test on dataset *i* for store, $TT_{dataset_i}^{store}$ shows a number of total tests on dataset *i* for store, $ST_{dataset_i}^{store}$ represents a number of successful tests on store *i* for dataset, and $TT_{dataset_i}^{store}$represents a number of total tests on store *i* for dataset. ***Table 3 enlists the success ratio for each store and each dataset.*** Success ratio *S* of both in-memory stores: **Sesame = 0.85 and Jena = 0.81** proves that they are less successful than the other stores, for the reason that *dataset4*, i.e., 50 million triples, fails to load in main memory for both these stores.

Table 3: Dataset Success Ratio



Stores	Datasets										
	Dataset01		Dataset05		Dataset10		Dataset50		Dataset100		SR_{store}
	S	F	S	F	S	F	S	F	S	F	
Sesame	8	0	8	0	8	0	8	0	8	0	0.85
Jena	8	0	8	0	8	0	8	0	8	0	0.81

V. CONCLUSION

In this paper, a comparative analysis of both ***Sesame and Jena were presented, a flexible architecture for querying and storing both RDF*** Schema information and RDF data. With LUBM test queries, ***Sesame achieves better than Jena***. Sesame is a significant step that could be the currently available storage and query devices for RDF data, because it is the first publicly available implementation of a query language that is aware of the RDF Schema semantics. An important feature of the Sesame architecture is its abstraction from the details of any particular repository used for the actual storage. This will be possible to port Sesame to a huge variety of different repositories, and that includes RDF triple stores, relational databases, and remote storage services on the Web. In the ontology strategy is obtained in the case of Sesame, followed by OWL API, to conclude that the smallest loading time of the individuals from the relational data base and finally the approaches which use Jena. Finally, this paper explained, in the case of the ***retrieval of the information, the best performance is also specified by Sesame compared to Jena***.

VII. REFERENCES

[1] R. Bose, J. Frew, Lineage retrieval for scientific data processing: a survey, ACM Computing Surveys 37 (1) (2005) 1–28.

[2] S. Bowers, T.M. McPhillips, B. Ludäscher, S. Cohen, S.B. Davidson, A Model for User-oriented Data Provenance in Pipelined Scientific Workflows, Proc. of the International Provenance and Annotation Workshop (IPAW), 2006.

[3] J. Broekstra and A. Kampman. SeRQL: a second generation RDF query language. Technical report, 2003, Available from http://www.w3.org/2001/sw/Europe/ events/20031113-storage/positions/aduna.pdf.

[4] J. Broekstra, A. Kampman, F. van Harmelen, Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, Proc. of the International Semantic Web Conference (ISWC), 2002, pp. 54–68.

[5] P. Buneman, A. Chapman, J. Cheney, Provenance Management in Curated Databases, Proc. of the SIGMOD International Conference on Management of Data, 2006, pp. 539–550.

[6] P. Buneman, S. Khanna, W.C. Tan, Why and Where: A Characterization of Data Provenance, Proc. of the International Conference on Database Theory (ICDT), 2001, pp. 316–330.

[7] Berners Lee, T.; Hendler, J.; Lassila, O.: The semantic web. Scientific American. **284**(5), 35–43 (2001).

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

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

[10] Guo, Y., Pan, Z., Heflin, J.: An evaluation of knowledge base systems for large OWL datasets. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 274–288. Springer, Heidelberg (2004)

[11] Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2), 158–182 (2005)

[12] Lee, R.: Scalability report on triple store applications. Technical report, Massachusetts Institute of Technology (July 2004), http://simile.mit.edu/reports/stores/

[13] Beckett, D.: SWAD-Europe deliverable 10.1: Scalability and storage: Survey of free software / open source RDF storage systems. Technical Report IST-2001-34732, EU (July 2002), http://www.w3.org/2001/sw/Europe/reports/rdf scalable storage report

[14] Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)

[15] W3C. Resource description framework (RDF). http://www.w3.org/RDF/, November 2006.

[16] W3C. RDFa. http://www.w3.org/TR/xhtml-rdfa-primer/, 2010.

[17] W3C. URI. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ #section-URIspaces, 2010.

[18] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0, November 1999.

[19] http://www.w3.org/TR/1999/REC-xpath-19991116.

[20] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML Query Language, February 2005. http://www.w3.org/TR/2005/WD-xquery- 20050211/.

[21] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In I. Horrocks and J. Hendler, editors, Proceedings of the First Internation Semantic Web Conference, number 2342 in Lecture Notes in Computer Science, pages 54–68. Springer Verlag, July 2002.
[22] RDQL - RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm.
[23] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A declarative query language for RDF. In The Eleventh International World Wide Web Conference (WWW'02), 2002.
[24] J. Broekstra and A. Kampman. SeRQL: An RDF query and transformation language. DRAFT. http://www.cs.vu.nl/ jbroeks/papers/SeRQL.pdf.
[25] M. Olson and U. Ogbuji. Versa. http://uche.ogbuji.net/tech/rdf/versa/.
[26] T. Berners-Lee. Notation 3. 1998. http://www.w3.org/DesignIssues/Notation3.html.
[27] TRIPLE homepage. http://www.dfki.uni-kl.de/frodo/triple/.
[28] RDF query using SquishQL. http://swordfish.rdfweb.org/rdfquery/.
[29] P. Haase, J. Broekstra, A. Eberhart, and R. Volz. A comparison of RDF query languages. In The Semantic Web - ISWC2004. Proceedings of the Third International Semantic Web Conference, 2004.
[30] A. Magkanaraki, G. Karvounarakis, T. T. Anh, V. Christophides, and D. Plexousakis. Ontology storage and querying. Technical Report No 308. April 2002. http://139.91.183.30:9090/RDF/publications/tr308.pdf.
[31] A. Owens, "An Investigation into Improving RDF Store Performance," [Online]: eprints.ecs.soton.ac.uk/17917/1/MiniThesis.pdf
[32] Powered by MediaWiki, A List of large Triple Stores.[online]: http://esw.w3.org/topic/LargeTripleStores
[33] http://jena.sourceforge.net/
[34] http://www.hpl.hp.com/
[35] http://jena.sourceforge.net/license.html
[36] http://tech.groups.yahoo.com/group/jena-dev/
[37] http://groups.yahoo.com/
[38] http://www.joseki.org/
[39] http://jena.sourceforge.net/inference/
[40] http://www.openrdf.org/
[41] http://www.aduna-software.com/
[42] http://www.ontoknowledge.org/
[43] http://www.nlnet.nl/
[44] http://www.sourceforge.net
[45] http://www4.wiwiss.fu-berlin.de/bizer/TriG/

ABOUT THE AUTHORS

V.Shanmugapriya received her **M.Phil** Degree from Periyar University, Salem in the year 2007. She has received her **M.C.A** Degree from Madurai Kamaraj University, Madurai in the year 2002. She is working as Assistant Professor, Department of Computer Science, **PGP** College of Arts & Science, Namakkal, Tamilnadu, India. She is pursuing her Ph.D (Part-Time) Degree at Sri Vijay Vidyalaya College of Arts & Science. Salem, Tamilnadu, India. Her areas of interest include Data Mining, Big Data and Wireless Networks.

.

Dr.D.Maruthanayagam received his **Ph.D** Degree from Manonmaniam Sundaranar University, Tirunelveli in the year 2014. He received his **M.Phil** Degree from Bharathidasan University, Trichy in the year 2005. He received his **M.C.A** Degree from Madras University, Chennai in the year 2000. He is working as **Head and Professor**, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India. He has above **18** years of experience in academic field. He has published **5** books, **32** papers in International Journals and **30** papers in National & International Conferences so far. His areas of interest include Computer Networks, Grid Computing, Cloud Computing and Mobile Computing.