



---

CSE 681- SOFTWARE MODELING AND ANALYSIS

---

Project #1



KEY-VALUE DATABASE  
OPERATIONAL CONCEPT DOCUMENT  
VER 1.0

INSTRUCTOR- DR. JIM FAWCETT

RAVICHANDRA MALAPATI

SUID: 22375-2155

Date: 16-SEP-2015

## Contents

|   |    |
|---|----|
| <b>1. Executive Summary</b> .....               | 3  |
| <b>2. Introduction</b> .....                    | 5  |
| 2.1 Obligations .....                           | 5  |
| 2.2 Organizing Principles.....                  | 5  |
| 2.3 Key architectural designs.....              | 6  |
| <b>3. Use Cases</b> .....                       | 7  |
| 3.1 Software developer.....                     | 7  |
| 3.2 Software architect.....                     | 7  |
| 3.3 Testing team.....                           | 7  |
| 3.4 Quality analyst/Quality Auditor .....       | 7  |
| 3.5 End User.....                               | 8  |
| <b>4. Application Activities</b> .....          | 9  |
| 4.1 Parsing the input file .....                | 9  |
| 4.2 Generate the Key-Value pair.....            | 9  |
| 4.3 Extracting the Key information.....         | 9  |
| 4.4 Persistence of Data.....                    | 9  |
| 4.4 Saving Data to Database .....               | 9  |
| 4.5 Retrieving Data from Database.....          | 10 |
| 4.6 Heterogeneous Database.....                 | 10 |
| Technical Heterogeneity.....                    | 10 |
| Data model Heterogeneity .....                  | 10 |
| Semantic heterogeneity .....                    | 10 |
| 4.7 Logging and Debugging .....                 | 10 |
| 4.8 Dynamic Schema.....                         | 11 |
| 4.9 Sharding of files .....                     | 11 |
| 4.10 Scheduling .....                           | 11 |
| 4.11 Immutable Database from query results..... | 11 |
| 4.12 Zero configuration DB Engine .....         | 11 |
| 4.13 Error reporting.....                       | 11 |
| 4.14 Display Results.....                       | 12 |
| <b>5. Packages or Modules</b> .....             | 13 |
| 5.1 Executive .....                             | 13 |

|           |  |           |
|-----------|--|-----------|
| 5.2       | Query Formatter .....  | 14        |
| 5.3       | Query Editor .....   | 15        |
| 5.4       | Logger.....  | 16        |
| 5.4       | Key Value package relationship index .....                         | 17        |
| 5.5       | Query Engine .....   | 18        |
| 5.5       | DB Factory .....   | 18        |
| 5.6       | Memory Management.....   | 19        |
| 5.7       | XML to DB.....   | 19        |
| 5.8       | Data Sharder .....   | 19        |
| 5.9       | Scheduler.....   | 20        |
| 5.10      | Display .....  | 20        |
| <b>6.</b> | <b>Critical Issues .....</b>                                       | <b>21</b> |
| 6.1       | Pluggable sharding strategy.....                                   | 21        |
| 6.2       | Multi Thread .....   | 21        |
| 6.3       | Distributed Architecture.....                                      | 21        |
| 6.4       | Performance for Big Data files .....                               | 21        |
| 6.5       | Without GUI and Console how does user provide input .....          | 21        |
| 6.6       | Maintaining relationships with old packages and new packages ..... | 22        |
| 6.7       | Data loss .....  | 22        |
| <b>7.</b> | <b>Extended Applications .....</b>                                 | <b>23</b> |
| 7.1       | Extended application as Code Repository.....                       | 23        |
| 7.2       | Extended application in social media .....                         | 23        |
| 7.3       | Extended applications in Internet of Things .....                  | 23        |
| <b>8.</b> | <b>References .....</b>  | <b>24</b> |
| <b>9.</b> | <b>Appendix .....</b>  | <b>25</b> |
| 9.1       | DBEngine .....   | 25        |
| 9.2       | DBElement Package testing .....                                    | 27        |
| 9.3       | Display .....  | 29        |

## 1. Executive Summary

With the advent of IoT (Internet of things) the number of devices on the internet are increasing tremendously. The number of connected devices on the internet doubled in the last three years which means if there were x number of devices in the market till 2012 now the count has become 2x in just a span of three years. The data generated by these device is increasing rapidly. Take the example of social network, the data generated by the social network in the last three years is almost equal to the data that has been already existed before three years. (The values presented above are just estimates from my reading in internet\*)

In the technology market there is always demand to process the large amount of data that is present in the database, the existing RDBMS databases are not efficient enough to process this large volume of data. SQL has not been efficient in handling this large volume of data. Hence there are new database that have come in the market which does not use the conventional SQL but proven to be efficient handling large volume of the data. Since these databases doesn't use not only the conventional SQL but also other query methods, these databases are named as NoSQL Databases which means not only SQL.

In this project to solve the industry problems of accessing the large databases we will work on developing one of the NoSQL databaset. In the market there is long list of the databases, however they are not suitable for all applications, Hence we are developing a new database with which we can come over all the problems and can be used as single for any type of applications. For example mongoDB is used for large document storage whereas Cassandra is used for key-value database. Hence the existing NoSQL databases have their own merits and demerits. But we will try overcoming all these drawbacks and develop one single database using C# language and .Net framework. While doing so we will also try to retain some of the good features of the traditions SQL database. The following are the list of NoSQL databases that are available in the market with their drawbacks[4].

1. Key / Value Based  
e.g. Redis, MemcacheDB, etc
2. Column Based  
e.g. Cassandra, HBase, etc
3. Document Based  
e.g. MongoDB, Couchbase, etc
4. Graph Based  
e.g. OrientDB, Neo4J, etc

As we talked earlier our goal in this project is to develop a database which can perform better in all the above scenarios and better than the above databases.

We will also use C# and .Net frame work in this project to develop the database. Visual studio 2015 will be our IDE to write C# programs with .Net framework.

We will also use the famous 3Tier architecture to implement the requirements. The above all implementations will be explained in detail in this document.

The tool that we are going to develop will serve the following purposes

1. Read XML data file with instructions from user and save it in key value database
2. Up on Query from user in the form of XML file the tool will read from the database and provide XML file to the user
3. Read the individual key value data(Hardcoded) from user and save it in database
4. Add the key values to the database up on request from user
5. Remove the key values from the database un on request from user
6. Modify the key values from the database up on request from the user
7. Create the child relationships between the key values when requested from user in metadata

The main users of the tool will be the developers, programmers, architects, Quality analysts, Quality Auditors, Project Management Office (PMO), Testing Team, Client testing team and other stake holders of the software project etc.

The database will also be accessed by the other applications online and offline from different locations. Hence the database should be able to communicate with other applications using APIs error free.

The main stakeholders of this project would be Dr. Jim Fawcett as guide and requirements elicitation, Ravi as software developer, software Architect and project manager, Microsoft for providing .Net framework.

## 2. Introduction

The existing NoSQL Databases and RDBMS lack some flexibility and most of the NoSQL databases are not serving all the purpose of the applications. For example: most of the applications are presently using two databases one for accessing the critical data using the RDBSM and NoSQL database for storing the large amount of the data for Data analytics purpose which is later used by Hadoop, Apache, HBase and pig for data analytics purpose.

If a database is fast it cannot handle large amount of the data and if the database is able to handle the large amount of the data then it will not be speed, because of this most of the BigData Analytics applications are using the proxy backup servers or replicas of the actual real time server and using them to do Data analytics, Hence most of the Data analytics are not real time and relayed with delay.

In this project we will develop a database which can solve the problem of the speed, reliability and big data streams inflow and outflow. The database we develop can handle the document and the key-value dictionaries with the same speed.

The new database can also be used as repository with parent child relationships. This feature will be explained later in this document.

The presently available databases don't provide the dynamic schema and in our database will implement the dynamic schema feature. The dynamic schema feature will provide the feature

### 2.1 Obligations

The main objective of key-value database project is import XML files and save them in the database using key-value relationship and export the data that is present in the database to the XML when requested by the user. The project also should be able to scale the database size based on the requirement. The explanation of obligations of this project is provided clearly in the below points.

1. To read input from the user and persist them in the database as key-value relationship with date and time stamp
2. To write output to the user when requested using the key-value database with date and time stamp
3. To read XML file from the user and enter into the database with key value relationship with date and time stamp
4. To write the XML file to the user when requested by the user with date and time stamp
5. The data base also should be able to read the meta data in program files and act as a code repository.

### 2.2 Organizing Principles

- We will be using the existing packages like DBEngine package, DBElement package, DBExtensions package, Display and Utility extensions.
- Based on the requirement we may use the JSON and BSON format to store the stream of the data
- Visual studio 2015 community version is used as IDE for this project
- We will also use parser package provided by Dr. Fawcett

### 2.3 Key architectural designs

Key-Value database tool uses the DB Engine package, DBElement package, DBExtensions package, Display package and Utility extensions package developed by Dr. Facwcett.

**DBEngine:** DBEngine takes input of key and value pair. The DB engine consists of Insert API. The query generated by the Query formatter is forwarded to the Executive module and the executive module calls DB Engine.

**DBelement:** DB element package provides the interface to create a new key value pair which is passed to the DB Engine. Executive package uses the DB element to create a instance of the key value pair before inserting data into the DBEngine.

**DBExtensions:** DBExtensions package is dependent on DB Engine and DBElement package. DBExtesnsion package is used to extend the functionality of the DBEngine and DBElement.

**UtilityExtensions:** UtilityExtensions package is not dedicated to single package and it includes all the APIs that are for all the other packages.

**Display:** The display package is an API for DBExtensions, DBEngine, DBElement, DBExtensions and UtilityExtensions package. The Display package is developed in such a way that it formats the output of DBExtensions, DBEngine, DBelement, DBExtensions and UtilityExtensions.

### 3. Use Cases

All the stakeholders of the project access the key-value database tool on a daily basis, the following are the project stakeholders that are identified as of now.

1. Software developer
2. Software Architect
3. Testing team
4. Integration team
5. Project Management Office(PMO)
6. Technical team at Client
7. Client-End user
8. Quality Analyst
9. Quality Auditor
10. Any other project stakeholder who uses the tool
11. Other tools that use API to interact with the Database

#### 3.1 Software developer

As a Developer: On a daily basis the software developer writes multiple applications to access the database and will use the database interface that is created by this project. As a developer the software developer can integrate the database as part of a product or project and provide it to a third party customer who will be the actual end user.

As end User:

The software developer also will be an end user when the key-value database is used on a daily basis to store the programming files, to share the programming files with other developers etc.

The software developer also can develop other applications to interact with the key value database using APIs.

#### 3.2 Software architect

The software Architect uses the key-value database to set up the initial project setup before the actual project development started. When the database is used as a repository the software architect uses it to access the program files. If the database is used for developing another product or project then the project architect will use the key-value database to evaluate the performance of the product or project.

#### 3.3 Testing team

Testing team can use the database as a repository to keep track of the test cases and test results.

Testing team may use this project for testing such as to check if all the requirements are implemented as per the requirements document that is provided during the project initiation.

#### 3.4 Quality analyst/Quality Auditor

The quality analyst and the Quality Auditor will use the key-value database to check if the database



### 3.5 End User

The end user may use the database to save different data type of the data such as photos, videos, code files etc. as part of their daily usage.

## 4. Application Activities

Key-Value data base analyze the metadata of the file to know some important information about the data present in the file. The metadata provides the information about the author of the file, the software package that the file is part of, other relations with other files and packages. In order to get the metadata of the file, the XML file should be parsed for the required information.

### 4.1 Parsing the input file

The user provides the input file while starting the application. The user also provides in the metadata of the file if he needs to modify the existing file in the repository or he wants to append to the existing file in the database or he want to enter the new file in to the database.

The above information is extracted using the file parser by the Executive package.

### 4.2 Generate the Key-Value pair

After parsing the file the key-value pair is generated along with other important child keys. The key value pair is used to identify if the key is already existing in the database. If the key is not available in the database then a new key is created and the value is saved in the database. If the key is available based on the information provided in the metadata the data is modified or appended.

### 4.3 Extracting the Key information

If the user wants to delete the record then he will provide only the key information in the metadata and mark for delete. The executive package will extract the key by parsing the file and use the key to delete the existing records. Incase if the user does not provide any information about the operation to be performed then the database considers it as new key-value pair and saves in the database.

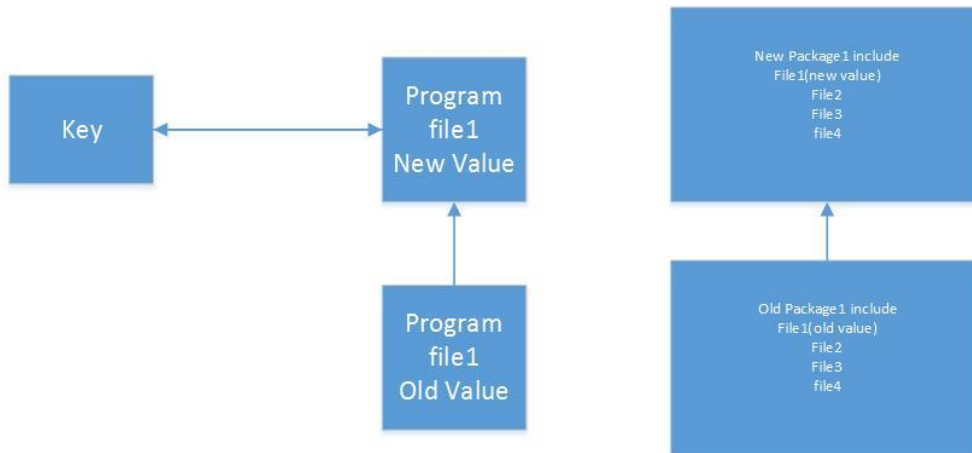
### 4.4 Persistence of Data

The database should also make sure that the data persists when modified and not ephemeral. The persistence database will always preserve the earlier versions of the data that is modified and can be accessible at any time by querying. As we know in code repositories and in social network when data is changed it is very much required to retain the previous versions.

As we are planning to use the database for repository purpose and if possible we can also use the database for social media and other web applications. Hence according to government's rule in many countries it is mandatory according to law to persist the data along with all earlier versions after modification.

### 4.4 Saving Data to Database

The database saves the data once received from user, the key value table is created after saving the data. If there is an update request then the data is updated at new address by persisting the previous versions of the data. Now the new data exists at one address and the old data exists at another address. Hence a new relationship between the old data, new data and the primary key to be created along with the package information.



Once the value is update the database also should update the new package with the latest file. During the retrieval of data the database should be able to refer to the old package as well as new package. This is the main purpose of the data persistence and code repository. This feature is mainly useful during implementation of social media projects and code repositories which we are planning to implement in project 5.

#### 4.5 Retrieving Data from Database

The database will return the package of program files, a single file and an individual value on query from the user. The modification of data will be accurate and each version of the file will be linked to respective packages. From the above figure we can understand that the obsolete versions of the file or the key are retained even after updating the key-value pair. This feature makes the database robust and can be used in any critical applications.

#### 4.6 Heterogeneous Database

Unlike all other NoSQL databases the key-value database that is being implemented in this project will perform better with all type of data types. This database will support the following types of heterogeneity.

##### Technical Heterogeneity

This database will support multiple file formats, multiple communication protocols (to access remotely).

##### Data model Heterogeneity

##### Semantic heterogeneity

#### 4.7 Logging and Debugging

The database will support full activity logging and debugging. For every actions that is performed on the database will be logged in the system.

The log report typically consist of the changed key values and the changed child relationships with the package. For example if a key is changed from one package to another package then this complete activity will be logged with user details. This feature of logging and debugging will help users to keep track of the changes that happen on the database.

## 4.8 Dynamic Schema

The database will support dynamic schema which gives flexibility to the user. Since most of the projects that are run in the real time industry are following agile process of development, it is very much required that the schema of the database should be dynamic.

For example while the customer providing the requirements he is not sure how many users will use the database and how much data inflow or outflow will be required in the future. Hence by implementing the dynamic schema we can make sure that the growing data will not affect the existing data.

The dynamic schema is also best at supporting unstructured data. In real time unstructured data is very common and static schema cannot handle the unstructured data.

## 4.9 Sharding of files

The database in this project deals with big data and the size of the file or value of the key can be as small as 1KB to 10petabytes. Since single system cannot handle such a huge data we implement Sharding of huge files. Even though there are many types of sharding models and algorithms we implement automatic integrated sharding.

In automatic integrated sharding we shard the file whichever is above some predefined size. However the user will be notified before sharding and no additional learning or understanding of sharding is required from user side as the database system can handle in such a way that the speed will remain same.

## 4.10 Scheduling

The database scheduler schedules periodical data backup to make sure that the data is not lost in case of system failure. The scheduler is also used to read the stream of input from the executive package. As we know in big data the data can be as bigger such that the data keeps on coming and waiting in the queue it is the duty of the scheduler to take care and handle multiple requests at the same time.

The scheduler also should make sure that all data queries and data operations are attended in equal time. The scheduler will keep database stable and healthy.

## 4.11 Immutable Database from query results

The database on command saves the results of the query by creating a new immutable database. The DBEngine creates immutable database on request from user. The immutable database holds the child parent relationship of key-value pairs.

## 4.12 Zero configuration DB Engine

The database will be zero configuration database which means there is no configuration or installation required by the user. The database will be folder with the executable file.

## 4.13 Error reporting

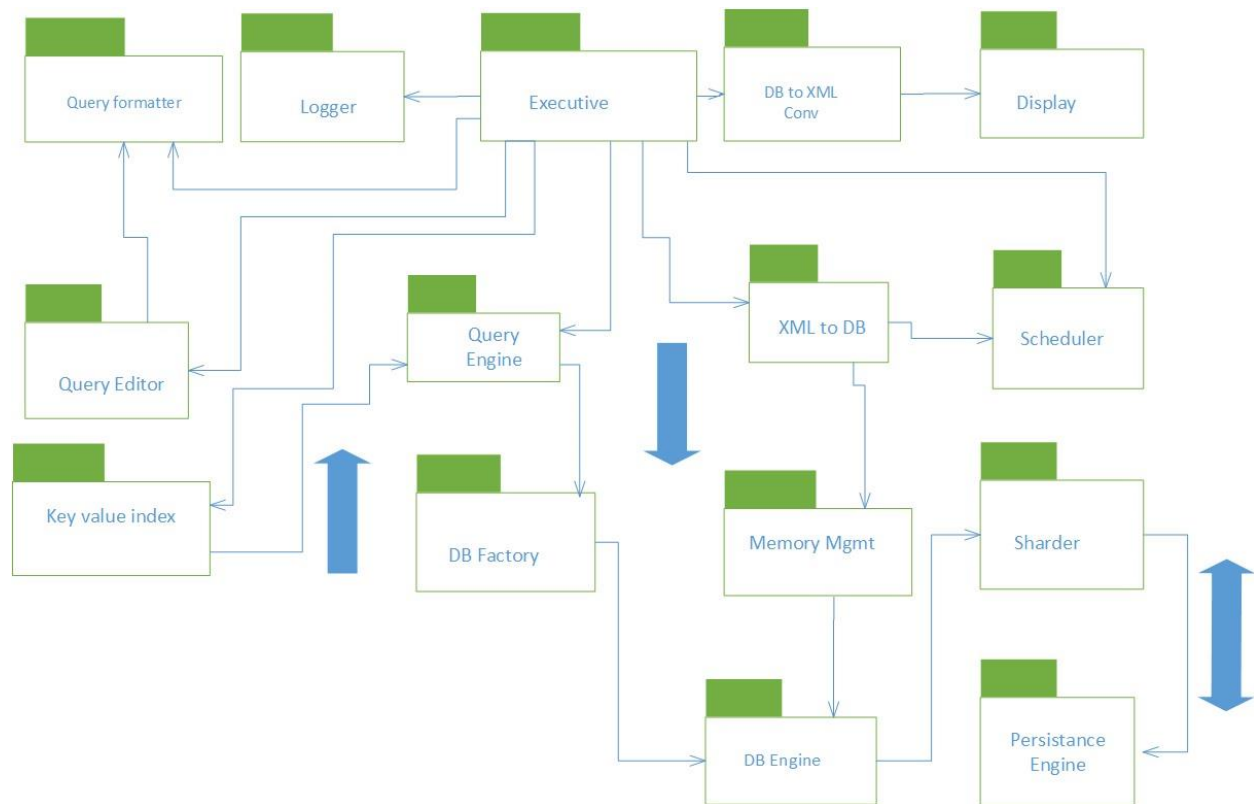
The database system will report error if the input XML file is not according to the format specified by the database manual. The error reporting scenarios include if the XML file missing some mandatory information or if the file is not according to the format specified by the database such improper XML metadata tags.

#### 4.14 Display Results

The final step after doing the operation that is required by the user is to display the result. When user request for a new key-value entry the result displayed on the console will be the state of the database before the operation and the state of the database after the operation. The display function also will display the records that are modified and the state of the database to the console after user operation is performed successfully. The display function also provides the error information such as no key found in the file that is input, no author information found, no child key information is found, the file is not part of any package etc. the error information will be discussed deeply in another document in project 2.

## 5. Packages or Modules

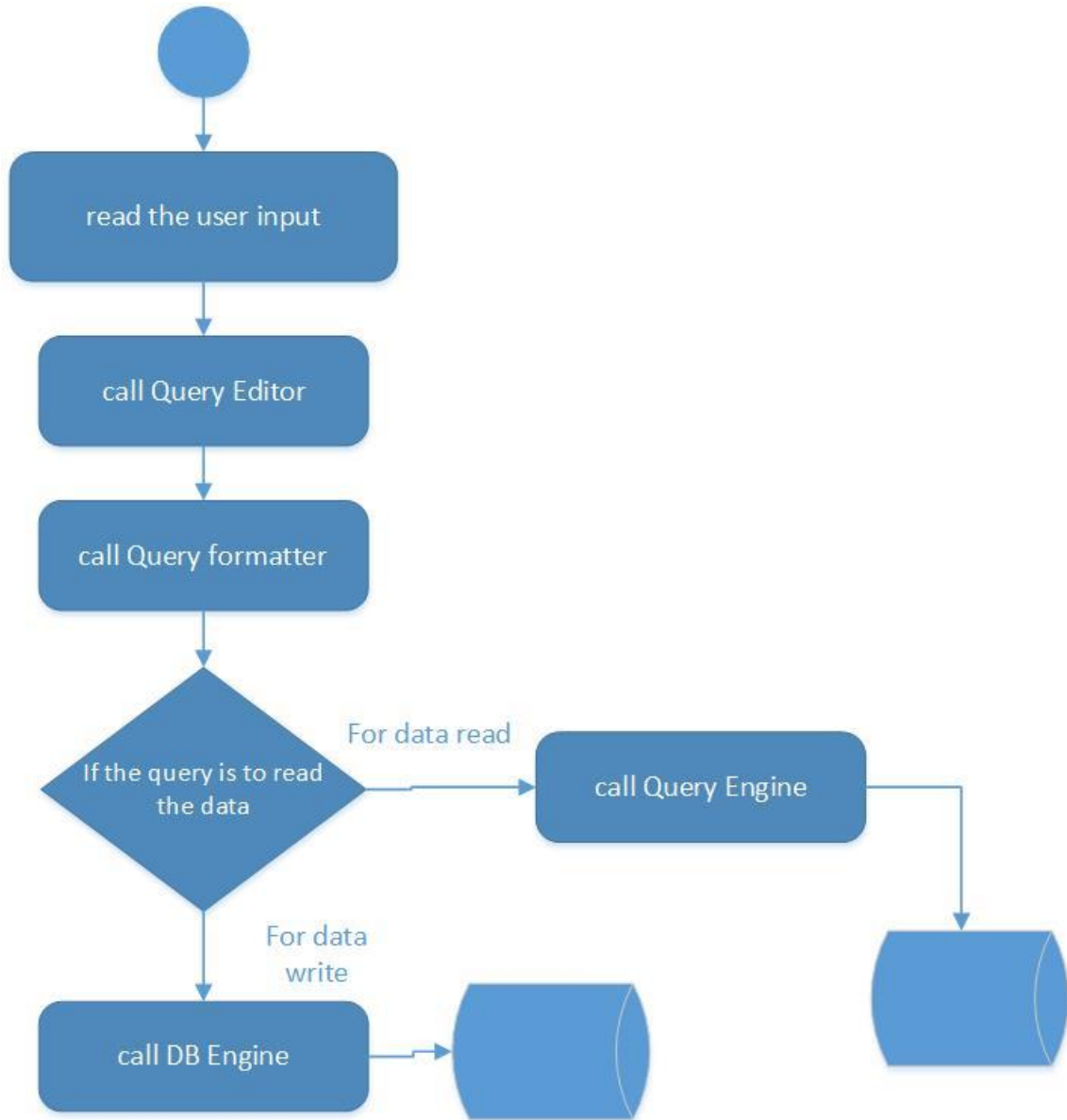
The design of this project will result in the following packages or modules. Each module or package is explained in detail.



### 5.1 Executive

This module acts like a controller module, this module controls and calls all other modules in this project. The module receives the file and sends the instructions to Query Editor which analyses the input file from the user and sends the valid data to the Query formatter. The Query formatter formats a query with the input data. The Query formatter also generates the key value pair and send back to the executive package. The executive package receives the response from the Query formatter and checks if the query is to read the data or write the data. If the query formatted is read then the query engine is invoked or if the query formatted is write then the DBEngine is invoked.

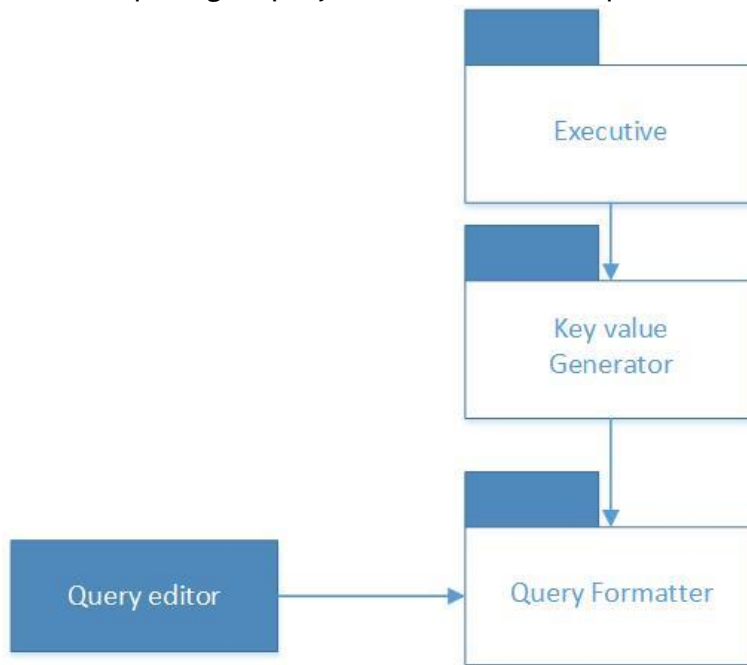
After receiving the inputs from the Query engine or DBEngine the executive module calls the display module. The display module displays the result to either to console or XML file or GUI depending on the user request.



## 5.2 Query Formatter

The Query formatter plays important role of making a query by taking inputs from the executer. The input is the metadata provided by the user in the file. By using the meta data the query formatter forms a query which can be understand by the Query engine and passes it to the

Executive package. Query formatter is also responsible for generating unique key value pair.

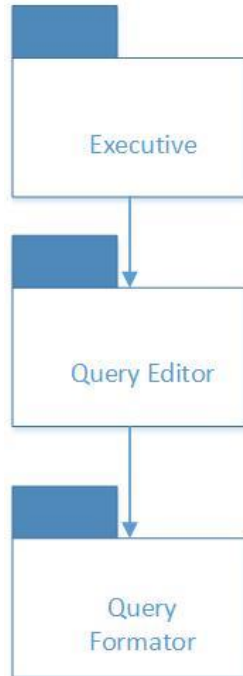


### 5.3 Query Editor

The responsibility of the Query editor is to take inputs from the executive package and read the key value index and provide them as input to the Query formatter.

Executive package will request query editor in case the input is not understandable by Query formatter. The query editor converts the raw form of the metadata and handovers to executive package later the query formatter will be called.

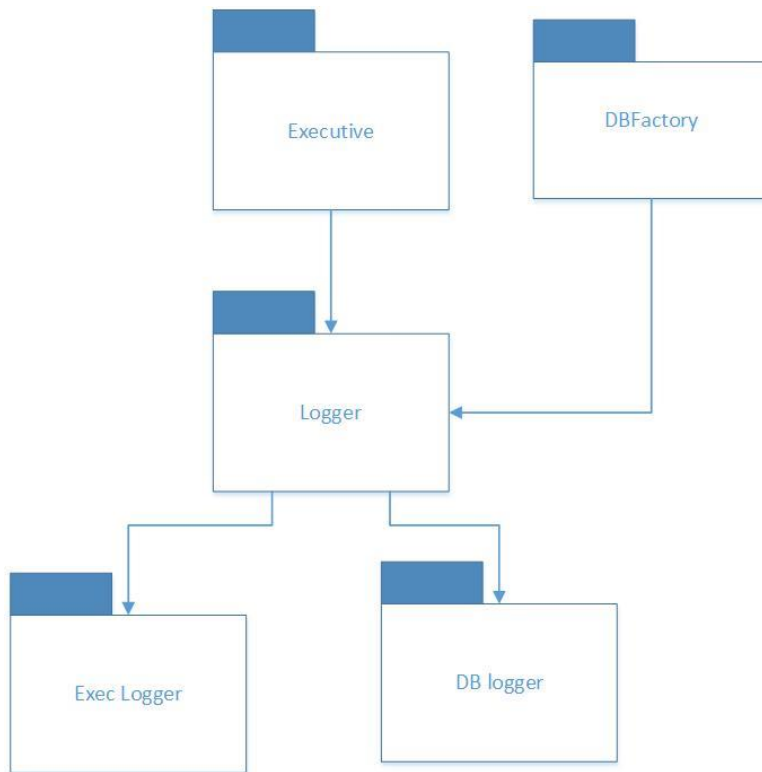




#### 5.4 Logger

The logger is very important package, the responsibility of the logger is to keep track of all the changes that are done on the database for every operation.

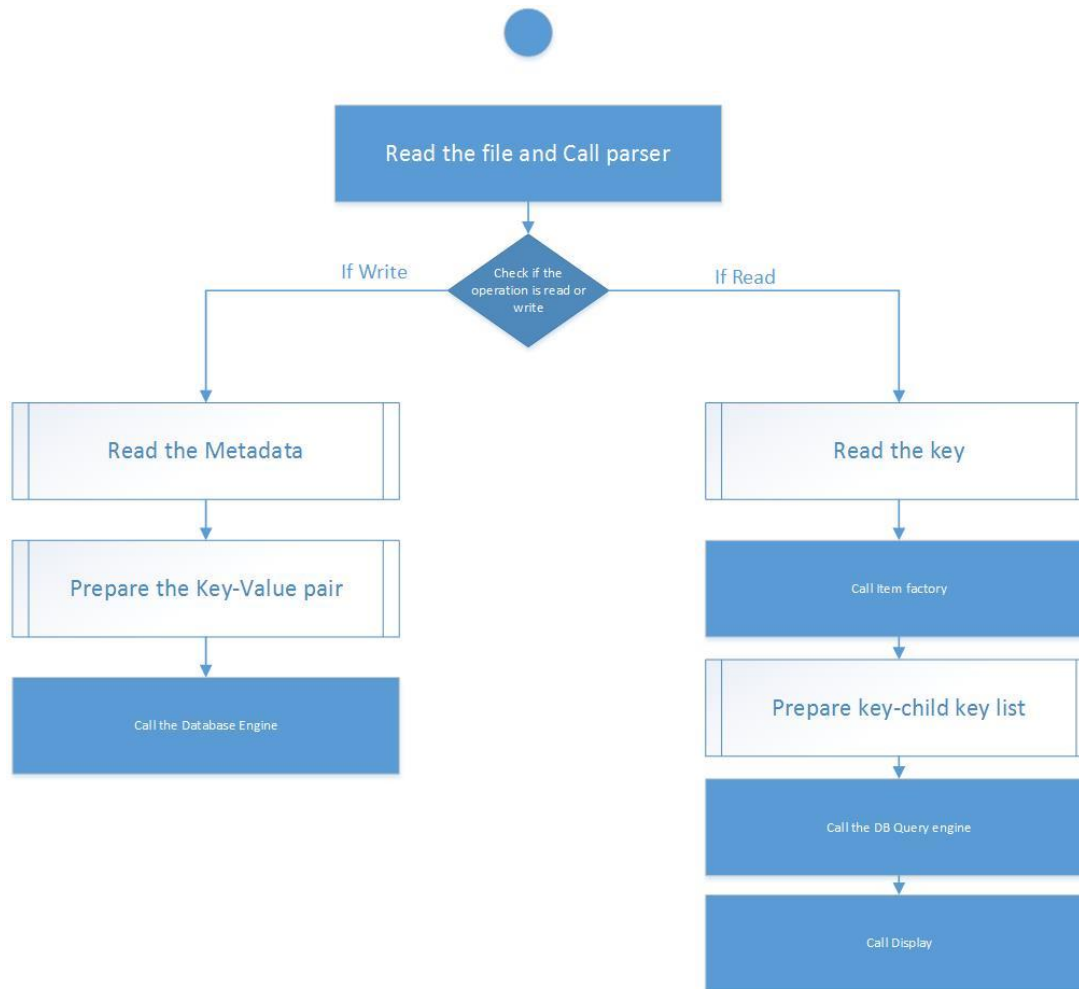
The logger is controlled by executive package and it will save each change in the data base with the time stamp.



#### 5.4 Key Value package relationship index

The key value index package is called by the executive whenever there is change in the key value relationship. The key value index will also have the details of the packages that the key is assigned to.

Sometimes if the Query editor requires any information about the relationships then key value package interface will provide with the required information.



## 5.5 Query Engine

The responsibility of the Query Engine is to collect the list of files that are required for a package from the executive task and provide the list to the DB Factory, the DB Factory will have the data of all the file and it will revert back with the data that is requested by Query Engine.

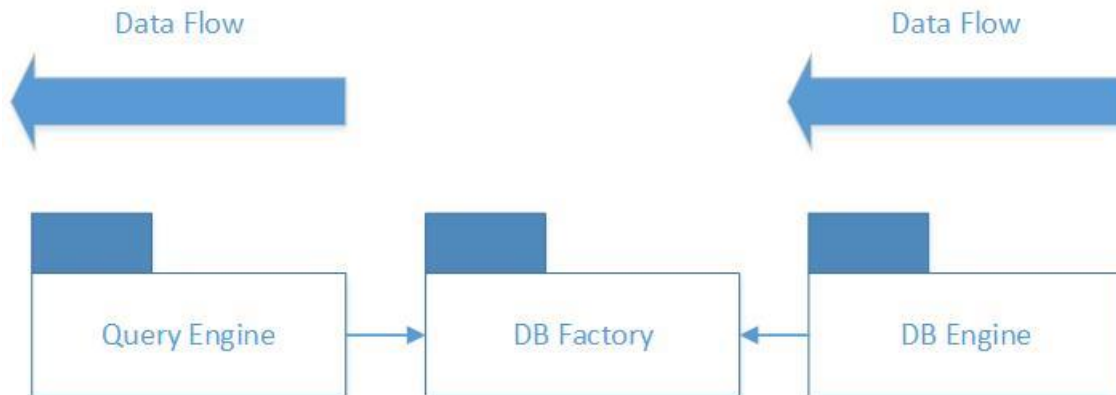
The query engine evaluates the query that is requested by the executive package and provides it file by file to the DB Factory package. The read operations happen through the query Engine.

## 5.5 DB Factory

DB factory is responsible for storing the data and maintaining in the memory. DB factory communicates mainly with two packages one is DB Engine and another one is Query Engine. The Query Engine queries the data elements that it want to read and provide the result to the user via executive and display packages which can be understand from the package diagram. Query engine mainly collects the data from the DB Factory and provides it to user. Whereas the DB Engine collects the data from the executive package and provides it to the DB Factory which keep the data and persists the data later when the scheduler runs.

The database factory will create a new database collection if required based on the user command.

DB Engine writes the data on command from the executive package whereas the Query Engine reads the data from the DBFactory on command from the Executive package.



### 5.6 Memory Management

Memory management is the responsibility of this package. As there will be huge amount of data that is writing into the database on daily bases, the memory management system will make sure that the closely related data will be kept at nearby addresses in the memory. In future the memory management package will also defragment the data. Defragmentation of the data will increase the speed of the data access.

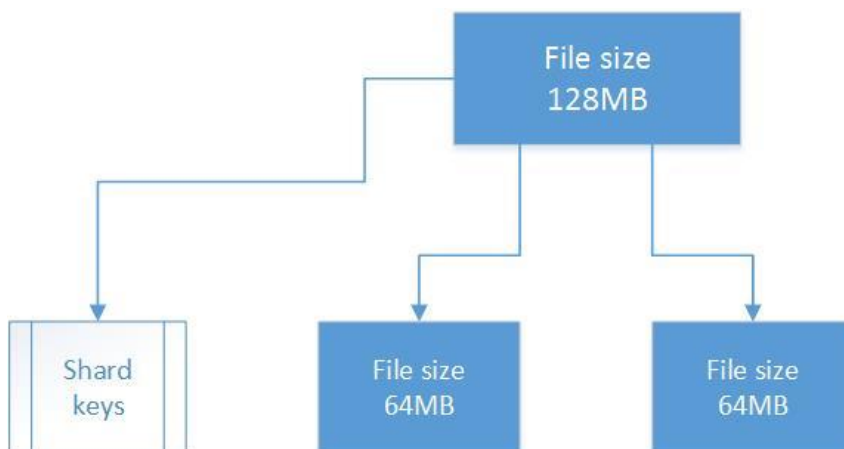
### 5.7 XML to DB

This module is responsible for converting the XML data to the JSON format or other feasible formats to save in the memory. This is heard of the DB system which accesses the system memory.

### 5.8 Data Sharder

Data Sharder's responsibility it to identify if the collection size is above certain size and split the file into small files and persist it. The sharder is also responsible for keeping track of the addresses of the data storage relevant to a collection.

Afeter sharding, the sharder generate the shard keys and share the shard keys across all the sharded servers.



## 5.9 Scheduler

The scheduler receives the input time interval or number of writes from the executive package. Using this trigger the scheduler will persist the changes.

The scheduler is capable of scheduling timed or event triggered backups of the database, timed or event triggered persistence of database. The scheduler also will have flexibility of grouping similar jobs and running them together. The scheduler package provides the log information to the logger to keep in records. The log information contains the list of tasks that are run and the time stamp along with the status of the jobs. The scheduler package can also be configured by the user.

**Monitoring jobs:** The scheduler is also responsible for monitoring the jobs until the jobs are completed.

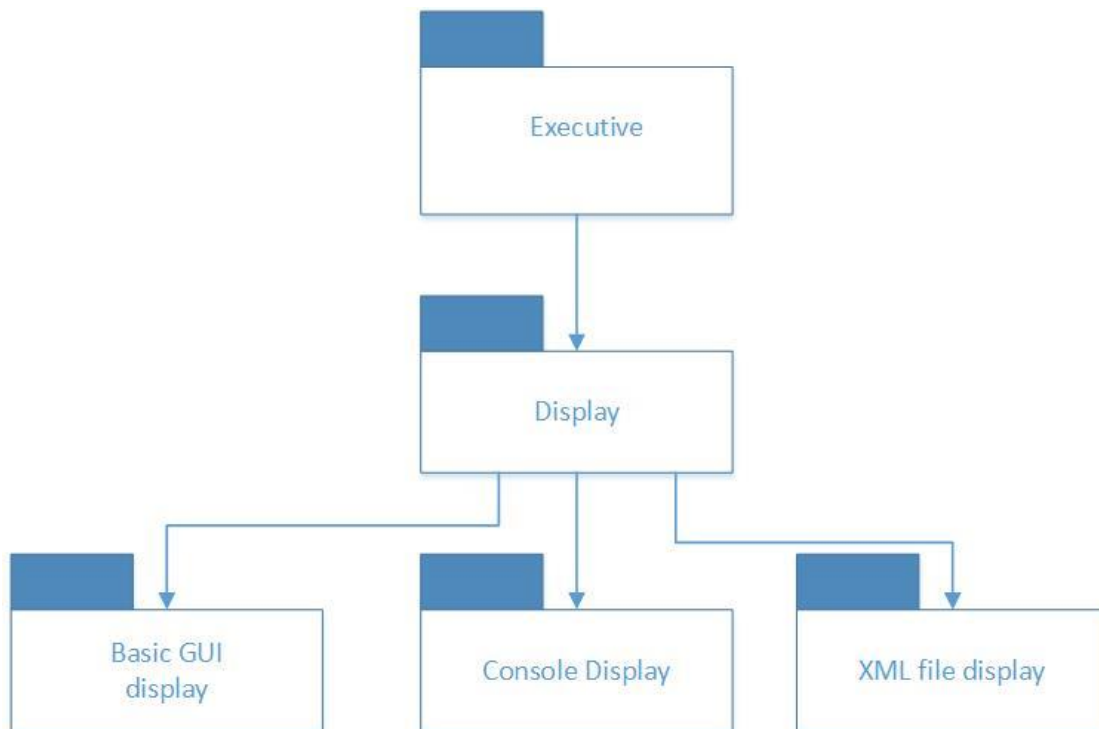
## 5.10 Display

Display package mainly receives input from the executive package. Based on the requirement the executive package may request for console display or XML file output, or GUI display.

The Basic GUI display package receives input from the display package. The basic GUI display package uses the .Net frame work to provide proper GUI display.

The console display package receives input from the display package, based on the input from the display package the Console Display package displays the value content of a particular key. The console package also displays the other information requested by the display.

The project also contains the XML generator which will generate the XML file when requested by the user. This XML generator is embedded in the XML display package.



## 6. Critical Issues

### 6.1 Pluggable sharding strategy

The pluggable sharding strategy would be critical. As of now most of the databases are implementing fixed or count based sharding strategy. In key value database since we are using the database for real time big data applications we require to implement pluggable data sharding strategy which is complex in nature and chances of losing the data.

**Solution:** In key value database the following pluggable strategies will be followed. The user can configure and run the hash based sharding. While the user providing the input file to the key value database a new XML tag will be provided to the user where the user can mention that for a particular file the sharding should happen with predefined condition.

We can also design the server in such a way that the user cannot leave the sharding option blank when providing the configuration to the server.

### 6.2 Multi Thread

As the database system is accessed from different locations and by multiple users at a single time, serving all the users concurrently is challenging task. To solve this issue we will implement a multi thread service system where all the requests are handled with equal priority and are served on constant time.

**Solution:** A new task scheduler will be implemented to solve the issue with the multi thread operations. When the database server serves multiple priority requests.

### 6.3 Distributed Architecture

Since the database will be handling huge files which are of peta byte data sometimes, we need to have multiple servers for storing these data. Implementing distributed architecture where the data is saved on different servers and maintained by multiple database systems can solve this problem but at the same time it is challenging and critical in nature.

### 6.4 Performance for Big Data files

If the file input is of big size, the time taken by the database to process it and save it will increase. And also if the output file requested by the user is of big data size then also the time taken by the database to process will increase. How the tool ensures the time taken by the database for big files and small files is same.

**Solution:** The database uses the technique data sharding. The data sharding will split the files which are huge and saves them in different data collections using multi thread. To read the big data files when queried by the user, using multi thread technique each file is processed independently and then finally put together and provided as output to the user. Hence using the multi thread we will make sure that the time taken to read and write the big data files is same.

### 6.5 Without GUI and Console how does user provide input

It is mentioned in the requirements that the tool will not have any GUI and Console interface. Without any interface it will be difficult for the users to interact with the database.

**Solution:** The XML file that is read by the database will be designed with the tags which says what operation the user want to perform. Using the tag <operation> the user can tell the database if he want to read the record or delete the record etc. In case if no tag is present the data base treats it as new record and generate new key-value pair and save it in the database.

### 6.6 Maintaining relationships with old packages and new packages

If the child parent relationships are to be maintained in the database it can become complex to relate the child parent relationship after few routines.

**Solution:** to make sure that the child parent relationships are maintained even if the database is growing, the key-value database provides a feature which will allow the user to create a separate index database up on request.

### 6.7 Data loss

Since the database is a complex software system there can be failures either due to the hardware or software failure.

**Solution:** to avoid data failure we implement the data replication feature. Data replication is the process of creating multiple copies of data at different server since the database support the distributed architecture the data replication can be implemented.

## 7. Extended Applications

### 7.1 Extended application as Code Repository

The developed NoSQL database can be used for code repository by software companies to store the code and access from multiple locations. Since the key value database has feature of maintaining the child parent index the package structure of the code can be easily retained and accessed without much hassle.

All of the above features makes the key value database a suitable database for implementing code repository.

### 7.2 Extended application in social media

Since the key value database will be very fast and responsive, this database can be used for the real time analysis of data (data analytics).

### 7.3 Extended applications in Internet of Things

The key value database is also well suited for storing the internet of things data as the database is heterogeneous. The sensors provide different kinds of data logging, hence the Internet of things applications require Heterogeneous database.



## 8. References

[1] Key Value Database requirements provided by Dr. Fawcett

<http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/lectures/Project1-F2015.htm>

[2] Key value database project #2 requirement provided by Dr. Fawcett

<http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/lectures/Project2-F2015.htm>

[3] NoSQL blog written by Dr. Fawcett

<http://ecs.syr.edu/faculty/fawcett/handouts/webpages/BlogNoSql.htm>

[4] Comparison of database management systems

<https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models>

[5] DBEngine, DBElement, UtilityExtensions, Display code provided by Dr. Fawcett

<http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project2HelpF15/>

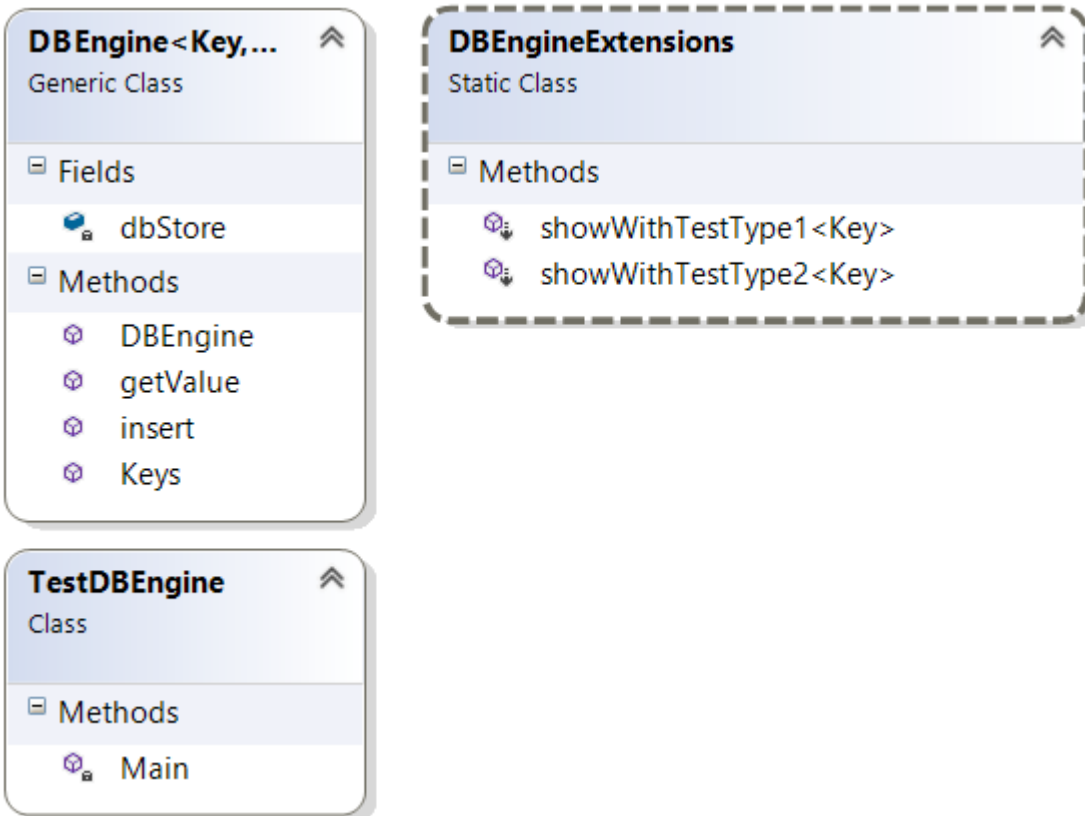
[6] Heterogeneous Database information provided by wikipedia

[https://en.wikipedia.org/wiki/Heterogeneous\\_database\\_system](https://en.wikipedia.org/wiki/Heterogeneous_database_system)

## 9. Appendix

### 9.1 DBEngine

The following class diagram will help to understand to the DBEngine package.



The DBEngine has insert function, getValue function and keys function.

Sample output of the DBEngine is provided for reference below.

```
Testing DBEngine Package
=====
--- Test DBElement<int,string> ---
name: unnamed
desc: undescribed
desc: 9/15/2015 10:57:30 PM
payload: a payload

name: Darth Vader
desc: Evil Overlord
desc: 9/15/2015 10:57:30 PM
payload: The Empire strikes back!

name: Luke Skywalker
desc: Young HotShot
desc: 9/15/2015 10:57:30 PM
payload: X-Wing fighter in swamp - Oh oh!

--- Test DBEngine<int,DBElement<int,string>> ---
all inserts succeeded

-- key = 1 --
name: unnamed
desc: undescribed
desc: 9/15/2015 10:57:30 PM
payload: a payload

-- key = 2 --
name: Darth Vader
desc: Evil Overlord
desc: 9/15/2015 10:57:30 PM
payload: The Empire strikes back!

-- key = 3 --
name: Luke Skywalker
desc: Young HotShot
desc: 9/15/2015 10:57:30 PM
payload: X-Wing fighter in swamp - Oh oh!

--- Test DBElement<string,List<string>> ---
name: newelem1
desc: test new type
desc: 9/15/2015 10:57:30 PM
payload:
one, two, three
```

## 9.2 DBElement Package testing

The DB element package class diagram and test output are shown below for understanding purpose. The DBElement package has all metadata properties such as payload, time stamp, file description, file name, children key details.

```
Testing DBElement Package
=====


--- Test TestType1 = DBElement<int,string> ---
-- use showElementWithTestType1<int>() --
name: unnamed
desc: undescribed
desc: 9/15/2015 10:58:44 PM

name: Darth Vader
desc: Evil Overlord
desc: 9/15/2015 10:58:44 PM
payload: The Empire strikes back!






name: Luke Skywalker
desc: Young HotShot
desc: 9/15/2015 10:58:44 PM
Children: 1, 2, 7
payload: X-Wing fighter in swamp - Oh oh!

--- Test TestType2 = DBElement<string,List<string>> ---
-- use showElementWithTestType2<string>() --
name: newelem1
desc: test new type
desc: 9/15/2015 10:58:44 PM
payload:
one, two, three


name: newerelem1
desc: same stuff
desc: 9/15/2015 10:58:44 PM
Children: first_key, second_key
payload:
alpha, beta, gamma, delta, epsilon
```


**DBElement<Key, Data>**   
Generic Class

[-] Properties




-  children
-  descr
-  name
-  payload
-  timeStamp


[-] Methods

-  DBElement


**LocalExtensions**   
Static Class

[-] Methods

-  showElementWithTestType1<Key>
-  showElementWithTestType2<Key>
-  showElemMetaData<Key, Data>

**TestDBElement**   
Class

[-] Methods

-  Main

### 9.3 Display

The following are class diagram and the test execution of the display package.

```
Testing DBEngine Package
=====
--- Test DBEngine<int,DBElement<int,string>> ---
all inserts succeeded

-- key = 1 --
name: unnamed
desc: undescribed
desc: 9/15/2015 11:13:33 PM
payload: a payload

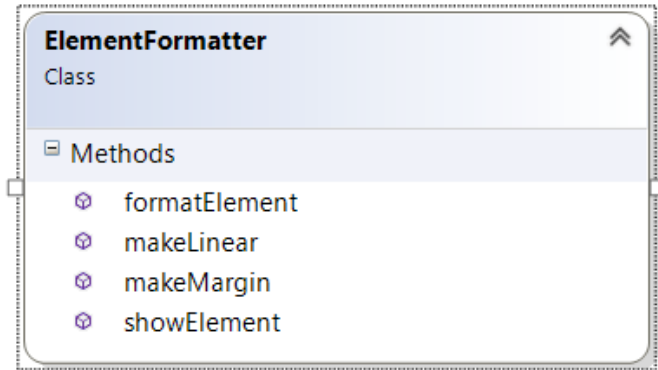
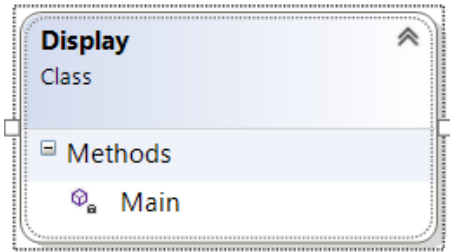
-- key = 2 --
name: Darth Vader
desc: Evil Overlord
desc: 9/15/2015 11:13:33 PM
payload: The Empire strikes back!

-- key = 3 --
name: Luke Skywalker
desc: Young HotShot
desc: 9/15/2015 11:13:33 PM
payload: X-Wing fighter in swamp - Oh oh!

--- Test DBEngine<string,DBElement<string,List<string>>> ---

-- key = -1587306074 --
name: newerelem1
desc: better formatting
desc: 9/15/2015 11:13:33 PM
Children: first, second
payload:
alpha, beta, gamma, delta, epsilon

-- key = 1141577281 --
name: newerelem2
desc: better formatting
desc: 9/15/2015 11:13:33 PM
payload:
a, b, c, d, e
```



The display package has the formatElement, makeLinear, makeMargin and showElement functions and the display package collects input from the executive package.