Format for the cover page

### SECT ZG629 T: Dissertation

***RDBMS to NoSQL Migration Tool***

# Dissertation work carried out at

***(Cognizant Technology Solutions Ltd***

***by***

**Name : Sampath Kumar KK**

**BITS ID\_No: 2013HT78285**

**Dissertation submitted for the partial fulfillment of**

***M.Tech. Software Engineering* Degree**

**under the Supervision of**

***Ashutosh Gupta***

***Senior Architect***

**Cognizant Technology Solutions, London**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

**Dec., 2015**

#### CERTIFICATE

**This is to certify that the Dissertation entitled *“RDBMS to NoSQL Migration Tool”* submitted by Sampath Kumar .K.K., BITS *ID-No.* 2013HT78285 for the partial fulfillment of the requirements of M.Tech. Software Engineering degree of BITS, Pilani embodies the work done by him/her under my supervision.**

Ashutosh Gupta

**Signature of the Guide/Supervisor**

**Name : Ashutosh Gupta**

**Designation : Senior Architect**

**Location: London**

**Date: 21-Nov-15**

**ABSTRACT**

**Dissertation Title: RDBMS to NoSQL Migration Tool**

**Name of Supervisor/Guide: Ashutosh Gupta**

**Name of the student : Sampath Kumar KK**

**BITS ID. No. : 2013HT78285**

**Semester : IV semester 2015-2016**

The NoSQL databases are used to store the unstructured data and semi-structured data whereas RDBMS is used to store structured data. The RDBMS databases have been popular for many decades now. However, the NoSQL database is gaining popularity as the data structure changed due to the introduction of Internet and social media. The applications that are very much social media centric, handling unstructured data and growing volume is most likely to use NoSQL databases. As the business appreciates the need of NoSQL databases, there is a requirement to migrate the data from existing RDBMS to NoSQL databases.

The goal of the project is to build a tool to migrate the data from RDBMS to NoSQL databases. The thesis illustrates the main advantages of NoSQL databases.

##### Signature of student Signature of supervisor

Name: Sampath Kumar KK Name of supervisor: Ashutosh Gupta

Date: 21-Nov-2015 Designation: Senior Architect

**Location: London, UK**

# Objective

The introduction of internet and social media has challenged the structured form of RDBMS database as the ultimate choice of the database management system. The factors such as the data can be stored in the way the front end application needs the data while retrieving and storing the data also plays an important role in choosing the NoSQL database. Basically, storing the data as it is required for the application simplifies the application and enhances the response time of the application.

# Different types of NoSQL

There are many different types of NoSQL databases. They are as follows:-

|  |  |
| --- | --- |
| **Types** | **Examples** |
| Document Store | CouchDB, MongoDB |
| Column Store | Cassandra, HBase |
| Key-Value Store | Redis, Riak |
| Graph Store | InfiniteGraph, Neo4j |

* 1. **Document Store:-**

A document-oriented database or document store is designed for storing, retrieving, and managing document-oriented information also known as semi-structured data. Document-oriented databases are inherently a subclass of the key-value store. The documents typically stored in the structure that is like JSON (JavaScript Object Notation). Documents contain one or more fields, where each field contains a typed value, such as a string, date, binary, or array. Rather than spreading out a record across multiple columns and tables connected with foreign keys, each record and its associated (i.e., related) data are typically stored together in a single document. This simplifies data access and, in many cases, eliminates the need for expensive JOIN operations and complex, multi-record transactions.

In a document database, the notion of a schema is dynamic: each document can contain different fields. This flexibility can be particularly helpful for modeling unstructured data. It also makes it easier to evolve an application during development, such as adding new fields. Additionally, document databases generally provide the query robustness that developers have come to expect from relational databases. In particular, data can be queried based on any combination of fields in a document.

**Advantages of document store are as follows:-**

* Fast, easy scalability
* Schema-less
* Highly fault tolerant
* Programmer friendly data types like JSON

**Disadvantages of document store are as follows:-**

* Query model limited to keys and indexes
* Row based systems are not efficient at performing operations that apply to the entire data set as opposed to specific records
  1. **Column Store:-**

A column-oriented DBMS stores data tables as sections of columns of data rather than as rows of data. Column families are groups of related data that is often accessed together. Column-oriented store would be selected where queries are likely to look at similar data items on many different records. For example, find all employees with last name “Kumar” can be retrieved in a single operation. The column-oriented store is most suitable for OLAP applications.

**Advantages of column store are as follows:-**

* Good horizontal scaling
* High availability
* Span across multiple data centers
* Faster query performances than RDBMS to retrieve similar data
* More suitable for reference data store

**Disadvantages of column-oriented database are as follows:-**

* Not optimum for transactional applications
* Doesn’t support complex data model
  1. **Key-Value Store:-**

Key-Value store is the simplest form of NoSQL database consisting of unique key and value can contain any data you wish to store. The value can be used to store large blocks of unstructured data. Simply, it is the hash table to store data in a schema free environment. The data type of stored data can be any data type supported by any programming language. So the data type depends on the programming language used by the application. The data stored in a schema free fashion alleviates the need to have fixed data model and structured data.

Amazon was the first to implement the key value structure to resolve their data availability and scalability issues using the Dynamo storage system.

**Advantages of Key-Value stores are as follows:-**

* High fault tolerant
* Very simple data model
* Very fast to set up and deploy
* Schema-less offers easier upgrade path for changing data requirements
* Good horizontal scaling

**Disadvantages of key-value database are as follows:-**

* Not suitable for complex applications
* Not efficient at updating portion of the records
  1. **Graph Store:-**

Graph stores consist of nodes, relationships among nodes and their properties. Graph models are more easily scalable over many servers than SQL DBs. The main feature of the graph database is its improved ability to handle relationships compared to a relational model. The Graph DB traverses through huge numbers of edges in a fraction of the time of a relational join due to direct links between nodes. The graph DBs uses the concept of vertex (node), edge (relationship) and property. Graph DBs stores data with nodes and edges using a graph algorithm. A node is a single record that has at least one and potentially many named properties. Edges define the relationship among nodes and both the nodes and their relationships have some predefined properties. Nodes can have multiple edges defining many different kinds of relationships they have with other nodes. Both nodes and relationships (edges) can be addressed with key values. Search or query with Graph DB is called “traversal”.

**Advantages of Graph store are as follows:-**

* Extremely fast for connected data
* Can handle complex queries involving multiple levels of related data

**Disadvantages of Graph store are as follows:-**

* Can’t scale horizontally
* Requires some learning curve to understand the database

# MongoDB:-

MongoDB is a document store NoSQL database. MongoDB is an open-source document database that provides high performance, high availability and automatic scaling. A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

**3.1 Collections:-**

MongoDB stores documents in collections. Collections are analogous to tables in relational databases. Unlike a table, however, a collection does not require its documents to have the same schema.

In MongoDB, documents stored in a collection must have a unique \_id field that acts as a primary key.

**3.2 Data types:-**

* String - This is most commonly used datatype to store the data. String in mongodb must be UTF-8 valid.
* Integer - This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
* Boolean - This type is used to store a boolean (true/ false) value.
* Double - This type is used to store floating point values.
* Min/ Max keys - This type is used to compare a value against the lowest and highest BSON elements.
* Arrays - This type is used to store arrays or list or multiple values into one key.
* Timestamp - ctimestamp. This can be handy for recording when a document has been modified or added.
* Object - This datatype is used for embedded documents.
* Null - This type is used to store a Null value.
* Symbol - This datatype is used identically to a string however, it's generally reserved for languages that use a specific symbol type.
* Date - This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
* Object ID - This datatype is used to store the document’s ID.
* Binary data - This datatype is used to store binay data.
* Code - This datatype is used to store JavaScript code into document.
* Regular expression - This datatype is used to store regular expression

**3.2 Indexes in MongoDB:-**

Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a *collection scan*, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

Fundamentally, indexes in MongoDB are similar to indexes in other database systems. MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents in a MongoDB collection. MongoDB indexes use a B-tree data structure.

**Default \_id:-**

All MongoDB collections have an index on the \_id field that exists by default. If applications do not specify a value for \_id, the driver will create an \_id field with an ObjectId value.

The \_id index is unique and prevents clients from inserting two documents with the same value for the \_id field.

**Single field:-**

In addition to the default index, you can define user defined ascending/descending indexes on a single field of a document. The ascending or descending order doesn’t matter as MongoDB can transverse in both directions.

**Compound Index:-**

More than one field can be defined as user defined indexes. This is called compound index. The order of fields listed in a compound index has significance. The data is sorted and stored based in the order of the fields.

**Multi-key Index:-**

MongoDB uses multi key indexes to index the content stored in arrays. If you index a field that holds an array value, MongoDB creates separate index entries for every element of the array. These multi key indexes allow queries to select documents that contain arrays by matching on element or elements of the arrays. MongoDB automatically determines whether to create a multi key index if the indexed field contains an array value; you do not need to explicitly specify the multi key type.

**Geospatial Index:-**

To support efficient queries of geospatial coordinate data, MongoDB provides two special indexes: 2d indexes that use planar geometry when returning results and 2sphere indexes that use spherical geometry to return results.

**Text Index:-**

MongoDB provides a text index type that supports searching for string content in a collection. These text indexes do not store language-specific stop words (e.g. “the”, “a”, “or”) and stem the words in a collection to only store root words.

**Hash Index:-**

To support hash based sharding, MongoDB provides a hashed index type, which indexes the hash of the value of a field. These indexes have a more random distribution of values along their range, but only support equality matches and cannot support range-based queries.

**3.3 Sharding:-**

Sharding is a method for storing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

To address the scaling issues, database systems have two basic approaches:-

* Vertical Scaling
* Sharding or Horizontal Scaling

**3.3.1 Vertical Scaling:-**

Vertical scaling adds more CPU and storage resources to increase capacity. Scaling by adding capacity has limitations: high performance systems with large numbers of CPUs and large amount of RAM are disproportionately more expensive than smaller systems. Additionally, cloud-based providers may only allow users to provision smaller instances. As a result there is a practical maximum capability for vertical scaling.

**3.3.2 Horizontal Scaling:-**

Sharding, or horizontal scaling, by contrast, divides the data set and distributes the data over multiple servers, or shards. Each shard is an independent database, and collectively, the shards make up a single logical database.

**3.4 Data Partitioning:-**

MongoDB distributes data, or shards, at the collection level. Sharding partitions a collection’s data by the shard key.

**Shard Keys:-**

To shard a collection, you need to select a shard key. A shard key is either an indexed field or an indexed compound field that exists in every document in the collection. MongoDB divides the shard key values into chunks and distributes the chunks evenly across the shards. To divide the shard key values into chunks, MongoDB uses either range based partitioning or hash based partitioning.

# Cassandra DB:-

Apache Cassandra is an open source distributed DBMS designed to handle large amount of data across many servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous master less replication allowing low latency operations for all clients.

Apache Cassandra is a massively scalable open source NoSQL database. Cassandra is perfect for managing large amounts of structured, semi-structured, and unstructured data across multiple data centers and the cloud. Cassandra delivers continuous availability, linear scalability, and operational simplicity across many commodity servers with no single point of failure, along with a powerful dynamic data model designed for maximum flexibility and fast response times.

Cassandra also places a high value on performance. Cassandra meets the performance and availability demands of Internet-of-Things (IoT), web and mobile applications. It provides enterprises a secure, fast, always-on database that remains operationally simple when scaled in a single datacenter or across multiple datacenters and clouds.

**4.1 How Cassandra is different from RDBMS:-**

Cassandra is designed from the ground up as a distributed database with peer-to-peer communication. As a best practice, queries should be one per table. Data is denormalized to make this possible. For this reason, the concept of JOINs between tables does not exist, although client-side joins can be used in applications.

**4.2 Why Cassandra avoids read-before write:-**

Cassandra uses a storage structure similar to a Log-Structured Merge Tree, unlike a typical relational database that uses a B-Tree. Cassandra avoids reading before writing. Read-before-write, especially in a large distributed system, can produce stall in read performance and other problems.

For example, two clients read at the same time, one overwrites the row to make update A, and the other overwrites the row to make update B, removing update A. Reading before writing also corrupts caches and increases IO requirements. To avoid a read-before-write condition, the storage engine groups inserts/updates to be made, and sequentially writes only the updated parts of a row in append mode. Cassandra never re-writes or re-reads existing data, and never overwrites the rows in place.