NoSQL, the database for the Cloud

NoSQL databases, the next generation databases, are a cost-effective alternative to relational databases. They have been successfully used by Google, Amazon, Twitter and Facebook to achieve massive parallelism, unlimited scalability and high availability. For enterprises, they can be well suited for Cloud Computing, Social Networking and Web 2.0 applications. However, they are not meant to handle mission-critical and highly transactional applications, therefore they cannot totally replace Relation Database Management Systems (RDBMS). In this whitepaper, we describe various breeds, choices, and tradeoffs to enable enterprises to make informed decisions to optimize the utilization of NoSQL databases.
About the Author

R. Sasirekha
R. Sasirekha is an Architecture Consultant with TCS. Ms. Sasirekha has over 18 years of IT experience covering various areas - Mainframe, Windows, Architecture, Performance management and Consulting. She has handled large project delivery and Management for clients across geographies – the U.S., the U.K., Switzerland, Malaysia, Singapore and India. Now, as part of the Global Consulting Practice, she tracks technologies and trends and their applicability to enterprises.

She has had her articles published in various magazines – Xephon, DB2 magazine and ISSA Journal. Blogs at http://itknowledgeexchange.techtarget.com/enterprise-IT-tech-trends/. She can be reached at sasirekha.r@tcs.com
# Table of Contents

1. NoSQL Movement .................................................. 4  
2. NoSQL Breeds ..................................................... 5  
3. Typical Characteristics of NoSQL ......................... 6  
4. Adoption and appropriate usage patterns .............. 7  
5. Hybrid Database Strategy .................................... 9
NoSQL Movement

The movement toward NoSQL (or Next Generation Databases that are non-relational, distributed, open-source, horizontal and scalable), that started in early 2009 as an alternative to the use of the relational databases is gaining momentum.

However, relational databases, which have been existence for more than three decades, have a strong hold in enterprises today. The newer breed of applications and technologies – involving Web 2.0, social networking and cloud computing – require horizontal scaling, involving thousands of nodes. And this is where the relational database fails to deliver.

NoSQL originally stood for a provocative “No SQL”, and the abbreviation was debated in the industry, which also came up with and rapidly discarded alternatives like “No Relational”, and “No Join”. But considering that these NoSQL databases are not meant to handle mission-critical and highly transactional applications, they are not going to replace the relational databases anytime soon, if at all. Instead what the DBs aim to do is to give users the choice of selecting the database that is most suitable for a given application.

In this context, it would therefore be apt for NoSQL to stand for “Not only SQL”, and when this is deployed in a cloud environment, the ecosystem can be called, “Databases for the Cloud”.

NoSQL has been successfully used by high-volume web sites such as Google, Amazon, Twitter and Facebook to achieve massive parallelism, unlimited scalability and high availability. But Enterprise Adoption of NoSQL is still at a nascent stage, according to a September 2010 survey by InformationWeek, which showed that:

- A mere 5% of companies are using or piloting NoSQL
- 22% of the IT workers surveyed say they’re interested but need to learn more.
- 44% of survey respondents hadn’t even heard of NoSQL.

Alternatives to Relational (mainly Object and XML databases) have been there for more than a decade, but the factors that make sense for Enterprises to adopt NoSQL now are:

- Cloud computing, which is expected to reduce costs, improve flexibility and enhance agility. But these benefits cannot be achieved without providing “massive scalability” at incremental costs. This necessitates fault-tolerant data stores and alternates to per-processor licensing structure of RDBMS.
- Applications like business intelligence, enterprise analytics, customer relationship management, document processing, Web 2.0 based applications, social networking have different needs for data, query and index types. Relational concepts like normalization, ACID properties, are found to be constraining the natural design of such applications.
- Relational databases are found to be inadequate in distributed processing involving very large number of servers and handling Big Data applications.
- The relational database performance on data-intensive applications including indexing a large number of documents, serving pages on high-traffic websites and delivering streaming media is found to be poor.

While NoSQL definitely is not to be ignored – especially if an organization wants to exploit the benefits of cloud computing, it is important to take the following points into consideration:

- Most of NoSQL databases are young (less than 10 years) and hence the maturity is not comparable to RDBMS.
- Unlike the existing RDBMS vendors, these databases are mostly from small start-up companies – and the longevity and support level cannot be taken for granted.
A breed of structured storage comes under this umbrella and even within each breed, the products have subtle differences and the population with expertise in NoSQL is not as vast as in the case of RDBMS.

Effective use of NoSQL requires unlearning some of the traditional wisdom of the relational databases.

Two analogies that reasonably sum up what NoSQL is about

- NoSQL are like specialized tools compared to the Swiss Army knife functionality of SQL platforms.
- SQL databases are like automatic transmission and NoSQL databases are like manual transmission (where you get more control leading to more responsibility).

What is required for successful adoption of NoSQL is the mindset change in terms of accepting trade-off in consistency to achieve availability and scalability, approximate answers are “okay” and users willing to take control.

**NoSQL Breeds**

As can be expected, there are several “breeds” of NoSQL, principal among them being:

**Key-Value Stores**

The key/value stores are the simplest NoSQL databases where a key points to a value that is typically an arbitrary string. The operation of finding the value associated with a key is called a lookup (or indexing) and the relationship between a key and its value is called a mapping (or binding).

Key/Value stores have been there for a long time – and Unix’s dbm, gdbm and Berkley DB are key/value stores. Key-value pair concept has been frequently used in traditional applications for lookup tables, hash tables and configuration files.

Most NoSQL key/value stores are a bit more than a simple key/value store and have advanced features. Within the key/value stores, the in-memory variants retain their data in memory for improved performance (useful as distributed cache mechanism), and the on-disk versions save their data directly to disk (useful as data storage).

*Redis, Memcached, Amazon SimpleDB, Google’s BigTable, Tuplespace, Tokyo Cabinet and Scalaris are the key products that fall under “Key-value store” category.*

**Document Stores**

Designed for “document-oriented” applications, document stores retain documents of any length and allow for retrieval based on the document content.

For example, an invoice that contains all the pertinent information about a single transaction—the seller, the buyer, the date, and a list of the items or services sold – can be stored as a single document.

Any number of fields of any length can be added to a document. Fields can also contain multiple pieces of data. Each document can have different set of fields and there are no empty fields. The idea of “evolving, self-contained documents” is the very core of its data model. In essence, document stores are similar to XML, YAML or JSON which are referred to as “semi-structured data”.

The document store databases provide the ability of handling millions of concurrent reads as they have a simple read (as typically one document contains all the required information). Document store is a good fit for user profiles, sessions, product information and all forms of web content (blogs, wikis, comments, messages etc.)
CouchDB, MongoDB RavenDB, Apache Jackrabbit, Terrastore, ThruDB and OrientDB are the key products that fall under “Document Store” category.

Wide Column Stores
Wide Column Stores shares a lot of ideas with a column oriented database. A column-oriented database stores its content by column rather than by row. Column oriented databases tend to be a hybrid of classic relational databases and the column oriented technology. Column is the basic element composed of a name, value and timestamp. The database stores its data (physically by column families) such that it can be rapidly aggregated with less I/O activity.

As column data is of uniform type, the storage size optimization achieved by modern compression schemes such as LZW (that make use of similarity of adjacent data) is higher for column databases. Compression of data like bitmaps provides additional benefit of ease of comparison. To make effective use of column-oriented approach, changes are required at storage level as well as programming paradigm.

Wide column stores are suitable for data warehouses, data mining, business intelligence, decision support where aggregates are computed over large number of similar data items. Hadoop, Cassandra, Hypertable and Cloudera are the key products that fall under “Wide column store” category.

Tabular
Tabular are distributed structured storage systems that resemble a database and share many implementation strategies with parallel and main-memory databases. Unlike others, in this category the concept of tables does exist.

Hbase (part of Hadoop), Google Bigtable, Hypertable and Mnesia are the key products that fall under “Tabular” category.

Graph Databases
A graph database is a database that uses graph structures with nodes (things like a person, a book or a website), edges (relationship between things, a family, a related book, a hyperlink) and properties to represent and store information. Some form of a graph database is used to do things like “People who bought this also bought…” (Amazon-style), “You might know…” (LinkedIn).

A graph (or a network) is a flexible data structure and maps more directly to the structure of object-oriented applications. Graph databases can scale more naturally to large data sets and are more suitable to manage ad-hoc and changing data with evolving schemas. Neo4J, InfoGrid, HyperGraphDB, DEX, Sonex, VertexDB and AllegroGraph are the key products that fall under “Graph Database” category.

Typical Characteristics of NoSQL

Key-value pair
Most NOSQL databases – BigTable, Hadoop, CouchDB, SimpleDB, memcached - use key-value pair (KVP) concept. A key-value pair (KVP) is a set of two linked data items:

1. a key (say account number or part number), which is a unique identifier for some item of data, and
2. the value, which is either the data that is identified or a pointer to the location of that data.

The databases using KVP can be considered as powerful B-tree storage engines. A B-tree is a sorted data structure that allows for searches, insertions, and deletions in logarithmic time.
Key-value pairs are frequently used in lookup tables, hash tables and configuration files. KVP has an important restriction, namely being able to access results by key alone. This restriction results in huge performance gains, massive speed improvements enabling partition of data over multiple nodes without impacting the ability to query each node in isolation.

Using KVPs with key based access only restriction compromises on the rapid retrieval capability provided by relational databases and makes reporting (especially ad hoc ones) difficult.

While it is recommended to avoid reporting against big heaps of KVPs, options like having additional lookup keys that allows possible queries or having a background ETL process to transform KVP data into entity based schema are used.

**Schemaless**

Relational database typically required modeling of data up front and expects all columns to be filled (at least as “not available” or “NULL”).

NoSQL databases are schema-free and hence can store unstructured data and allow aggregating the data after the fact (evolving data model), as humans tend to do. Schemaless data store gives the flexibility to design a domain model with objects first and then mapping the domain to the underlying datastore.

The downside is that as there are no constraints, theoretically different forms of the same information and ad hoc design could lead to a “murky” data store. Also there could be infinite number of instances of the same data (with only the key changing) and also lack of cascade delete (no foreign keys) could result in stale children.

In effect, schemaless gives flexibility – at the expense of consistency and integrity - and to be effective needs discipline.

**Adoption and appropriate usage patterns**

The most common forms of web applications – blogs, forums, CMS, wikis – are all examples of structured document storage, and have a natural fit with NoSQL and an impedance mismatch with SQL. The following are some of the real life examples of NoSQL usage:

- When Youtube grew from 30 million pages per day to 100 million pages per day, in a 4 month period, they zeroed in on database sharding which resulted in large performance boost, better cache locality and reduced their total DB hardware by 30%
- BBC chose CouchDB to create a multi-master, multi-datacenter failover configuration to support their website that has terabytes of data serving billions of requests. They have 32 nodes split between two data centers, with half of them designated as backup nodes. The backup nodes are at times used to meet the additional capacity requirements. The CouchDB solution allows them to do a hot backup while running.
- The New York Times is using MongoDB in a form-building application for photo submission. Mongo's lack of schema gives the ability to define any combination of custom form fields and the solution scales well.

The NoSQL movement is more about choosing the appropriate data store that is most suited for the applications. Whatever the choices, trade-offs exist and designing applications with these in mind and making informed decisions is required for effective utilization.

- For transaction oriented applications, with large number of updates from multiple users, and consistency cannot be compromised (locking is critical), relational databases are still the best.
For BI-oriented data-warehousing, the most important requirement is the efficient support of star-schemas. Column-based databases – such as Netezza and Vertica – are the ideal fits here. For typical ETL tasks, MapReduce would be a good fit.

For insert-only (in most cases at least) situations, involving crawling and indexing (both in the web and in Enterprises), blogs/wikis, Facebook-like applications, search-based retrieval (as against query-based), batch-oriented or in-memory aggregations and computations, Wide column stores like Hadoop with KVP support would be relevant.

For same user update (the creator updates and others only do read/query) situations, like shopping-cart type scenarios, product reviews, social network Q&A, e-Commerce, eAuctions etc, - simple key-value pair databases would be well-suited.

The variations in the key characteristics of NoSQL that would help in choosing a NoSQL product can be summarized as:

<table>
<thead>
<tr>
<th>Logical Data Model</th>
<th>Voldemort, Dynomite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-value oriented</td>
<td>BigTable, Hadoop, Hypertable</td>
</tr>
<tr>
<td>Column family oriented</td>
<td>CouchDB, MongoDB</td>
</tr>
<tr>
<td>Consistency and Availability</td>
<td>Hadoop, Hypertable, MongoDB</td>
</tr>
<tr>
<td>Availability and Partitionality</td>
<td>Cassandra, CouchDB</td>
</tr>
<tr>
<td>Memory Based</td>
<td>MemCaches, Scalaris</td>
</tr>
<tr>
<td>Disk Based</td>
<td>MongoDB, Riak</td>
</tr>
<tr>
<td>Memory and Disk</td>
<td>Hadoop, Cassandra, Hypertable</td>
</tr>
</tbody>
</table>

Hadoop is the only NoSQL implementation that has the most adoption today. IBM uses as well as distributes Hadoop.

MongoDB is the option typically chosen when an enterprise first experiments with NoSQL as it is closer to MySQL (than other NoSQL databases). It has a query optimizer, handles ad hoc queries and a custom network layer. Similar to MySQL, it allows organizing documents into collections for speed, efficiency and organization.

Most other NoSQL databases are built by the users (Google, Amazon, Facebook etc.) to meet their specific requirements and later open-sourced for external use. This naturally resulted in data models, product capabilities and query languages that are specific to each product / implementation. Portability and Interoperability are the trade-offs and it may be sometime before these are taken into consideration.

Graph databases – embracing RDF and SPARQL of W3C and built on a simple, uniform data model and a powerful declarative query language – can be said to the only standardized NoSQL solutions available today.

As expected these NoSQL databases are available in Cloud – Cloudant takes the reliability, simplicity, and power of CouchDB and adds distribution, scalability, and ‘cloud readiness’; and offers BigCouch, CouchOne (earlier CouchIO) in an effort to commercialize CouchDB is targeting Mobile and Clouds, MongoDB is the cloud-based hosted database solution for MongoDB,. heavy users of Cassandra deploy in the cloud, e.g. CloudKick on Rackspace Cloud Servers and SimpleGeo on Amazon EC2, Cloudera’s Distribution for Hadoop (CDH) provides additional features like job scheduling, workflow sequencing and the ability to control streaming data sources and Neo4j is available in Amazon AMI cloud.
Foreseeing Enterprise Adoption in the near future, third party products from vendors like Quest Software, Embarcadero Technologies are extending their relational database tools to include NoSQL systems.

NoSQL is easy to use as it is simple and is not as restricting (especially in the data model and ACID) as the relational databases.

**Hybrid Database Strategy**

Using single data store would be appropriate for a whole suite of applications like business intelligence, aggregation, semantic web, websites with high availability requirements, highly transactional mission critical application and financial transactions. But it is not uncommon to find applications where different areas have different storage, querying and consistency requirements.

Naturally, using more than one data store for such applications is the solution. CAP Theorem also clarifies that applications are expected to be a careful mixture of ACID and BASE subsystems.

Database vendors (both SQL and NoSQL) are also taking a hybrid approach:

- Drizzle, a fork of open source MySQL RDBMS database, designed for the Cloud aims at providing massive horizontal scalability while maintaining ACID compliance. The Drizzle team is actually removing non-essential code, refactoring the remaining code, switching to microkernel architecture with well-defined APIs making it a lean, mean query running machine.
- OrientDB is a NoSQL hybrid document-graph database with special operators for graph operations. It allows querying using SQL language with some extension like:

  ```sql
  Select from People where friends TRAVERSE(1,7) (name = 'Ayende' and surname = 'Rahien')
  <get all the people that have any relationship of friends up to the 7th level of separation>
  ```

- NoSQL datastores like Voldemort offer relational MySQL as one of the storage backends.
- VoltDB is a relational database that handles millions of transactions a second and still meets the ACID test by distributing both the database and the data across a server cluster.
- Oracle Table Functions provide a robust scalable way to implement Map-Reduce within the Oracle database and leverage the scalability of the Oracle Parallel Execution framework.
- IBM is expected to improve Netezza by including Hadoop MapReduce distribution for the parallel processing of large amounts of information or complex data types on hardware clusters.
- GreenPlum and AsterData, the well known Data warehousing vendors, have included MapReduce support in their products.

Hybrid models - with multiple data stores - are expected to be the trend in the future, letting each database do what it is best at.

Various thumb rules that can be applied in splitting the data between multiple data stores are:

- For static, historical data, NoSQL offers speed and efficiency. For dynamic, data entry with multiple concurrent users and data integrity is the key, relational databases are more suitable.
- Keep the data that needs to be queried and reported in the traditional SQL database and the data that needs best performance in a fast, distributed NoSQL database.
- Use the relational database for your transactions and pass the data to NoSQL for aggregation, business intelligence and decision support.
- Place the high-read data needing high-availability in a NoSQL and the historical, reporting data into a relational database.
- NoSQL data stores – especially the ones that relies on memory caching – is more appropriate for “right now” kind of data whereas the counterpart of past data (yesterday’s) can be kept in a relational data store.
Replication using messaging solutions, JMS or products like RabbitMQ supporting AMQP standards, provide the basis for SQL-NoSQL hybrid solutions. Drizzle is an example of this being supported at a product level. Using RabbitMQ as the transport, Drizzle can replicate data as serialized Java objects to Voldemort, as JSON marshalled objects to Memcached or as a hashmap to column family based Cassandra.

Custom implementation of such hybrid models include:

- Oracle as the main database for securities trading back office solution and MongoDB used for processing semi-structured data from external feeds. Relevant information from MongoDB pushed to JMS based queues from which the data is copied asynchronously to Oracle servers.
- Log application where Redis is used to keep a few days of data in memory and the past data archived to MySQL.
- A website that hold current documents in MongoDB and archives them to MySQL using DataMapper.
- Implementing database in the cloud using multiple SimpleDB replicas while retaining the Oracle database as the primary source for access within the data center – for transitioning to cloud.
- Crowdsourcing Website asking public to help dig through and categorize the stack of documents – forms, scanned receipts and hand-written letters using Django, MySQL, Redis and Memcached.
- A gaming company moved to cloud – load currently at 50-50 split between Amazon EC2 and internal data centers - to handle the increasing load with millions of users getting added every month. Uses Apache PHP on the front end, memcached to store key value pairs for active user play and MySQL on the back end.

Using a hybrid approach is the way forward as NoSQL movement is about having choice and specialization, moving away from a one-size-fits-all approach of using RDBMS everywhere and choosing a data store that is the best fit for the purpose.

References

2. As Enterprise start adoption NoSQL, these requirements would be given priority and handled.
About TCS’ Global Consulting Practice
TCS’ Global Consulting Practice (GCP) is a key component in how TCS delivers additional value to clients. Using our collective industry insight, technology expertise, and consulting know-how, we partner with enterprises worldwide to deliver integrated end-to-end IT enabled business transformation services.

By tapping our worldwide pool of resources - onsite, offshore and nearshore, our high caliber consultants leverage solution accelerators and practice capabilities, balanced with our knowledge of local market demands, to enable enterprises to effectively meet their business goals.

GCP spearheads TCS’s consulting capacity with consultants located in North America, UK, Europe, Asia Pacific, India, Ibero-America and Australia.

Contact
For more information about TCS’ consulting services, contact global.consulting@tcs.com or visit www.tcs.com/consulting

Subscribe to TCS White Papers
TCS.com RSS: http://wwwsoieth sl/fl/d t/lg hgrtrtuwu.aspx?f=w
Feedburner: http://riut,kghgsrh.:.rugfh tepapers

About Tata Consultancy Services (TCS)
Tata Consultancy Services is an IT services, business solutions and outsourcing organization that delivers real results to global businesses, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled services delivered through its unique Global Network Delivery Model™, recognized as the benchmark of excellence in software development. A part of the Tata Group, India’s largest industrial conglomerate, TCS has over 160,000 of the world’s best trained IT consultants in 42 countries. The Company generated consolidated revenues of over US $6.3 billion for fiscal year ended 31 March 2010 and is listed on the National Stock Exchange and Bombay Stock Exchange in India.

For more information, visit us at www.tcs.com

IT Services
Business Solutions
Outsourcing

All content / information present here is the exclusive property of Tata Consultancy Services Limited (TCS). The content / information contained here is correct at the time of publishing. No material from here may be copied, modified, reproduced, republished, uploaded, transmitted, posted or distributed in any form without prior written permission from TCS. Unauthorized use of the content / information appearing here may violate copyrights, trademarks and other applicable laws, and could result in criminal or civil penalties.

Copyright © 2011 Tata Consultancy Services Limited