Deloitte.

# Choosing the right database

**Akhil Sakhardande**
**Deloitte Consulting LLP**

## Document Control

**Change History:**

| Date | Author | Version | Description |
|---|---|---|---|
| 28–Nov-2015 | Akhil Sakhardande | 1.0 | Initial version |
| 1-Dec-2015 | Akhil Sakhardande | 1.1 | More descriptiveness |
| 3-Dec-2015 | Akhil Sakhardande | 1.2 | Included a concluding paragraph |

**Reviewers:**

| Reviewed by | Designation | Data Reviewed |
|---|---|---|
| | | |
| | | |
| | | |

## Table of Contents

## Abbreviations and Key Terms

| Abbreviation | Description |
|---|---|
| DBA | Database Administrator |
| OLAP | Online Analytical Processing |
| ETL | Extract, Transform and Load |
| DDL | Data Definition Language |
| DML | Data Manipulation Language |

## 1. Overview

The market is abuzz with terms such as NoSQL, Big Data, SMAC, etc. Often, IT decision makers can get very confused with all the noise. It gets difficult to understand why one should consider a newer, alternative database when RDBMSs have been around for a considerable time. However, many leading enterprises are already using alternative databases and are saving money, innovating more quickly, and completing projects they could not pursue before as a result. This article would discuss how one can determine if NoSQL is a fit for current or future applications.

You will need a towering amount of thought processing to select the right database solution for your business or let's say even for a project. Business insight takes a crucial part in this decision making.

The first consideration that needs to be made when selecting a database is the characteristics of the data you are looking to leverage. The analysis of your data will be very important part of the decision. The banking industry would have a completely different functional data as compared to the one in Life science. If the data has a simple tabular structure like an auditing spreadsheet, then the relational model could be adequate. Data such as geo-spatial, engineering parts, or molecular modeling, on the other hand, tends to be very complex. It may have multiple levels of nesting and the complete data model can be complicated. Such data has, in the past, been modeled into relational tables, but has not fit into that two-dimensional row-column structure naturally. In similar cases today, one should consider NoSQL databases as an option. Multi-level nesting and hierarchies are very easily represented in the JavaScript Object Notation (JSON) format used by some NoSQL products.

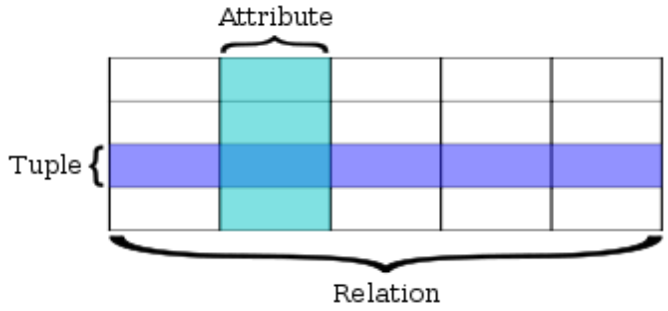
The famous Jnan Dash has focused a numerous times on the volatility of the data model. There is a need to understand if your designed data model is going to change or more likely to remain the same over a period of time. While designing the data model initially, all the facts are not known or not visualized and hence some flexibility is obviously needed. The issue of schema-rigidity still rings true today, leading to little flexibility when it comes to application development and evolution. This 'get it right first' approach may have worked in the old world of static schema, but it will not be suitable for the new world of dynamic schema, where changes need to be made daily, if not hourly, to fit the ever changing data model. The data model determines the gist of your project and focuses more on the job that one needs to accomplish. Sometimes it's the retrieval efficiency and other times its data loading and accuracy. It is not advisable to change the model later and hence need to be chosen wisely.

This paper provides guidance on what features to look for when choosing the appropriate database and database vendor. It also describes the advantages/disadvantages offered by the conventional relational databases compared to their non-relational counterparts.

## 2. RDBMS

Short for **Relational Database Management System**. This is a type of database management system that stores data in the form of relational tables. RDBMS is a product that presents a snapshot of data as a collection of rows and columns, even if it is not based strictly upon relational theory. These products typically implement some of Codd's 12 rules. These are the ongoing conventional relational systems enforcing on the ACID properties – Atomicity, Consistency, Isolation and Durability. This systems have been in the business for a considerable time now and they have been dominant while catering to the client needs.

A relational database is a database that has a collection of tables of data items, all of which is properly described and organized according to the relational model. Data in a single table represents a relation, from which the name of the database type comes. A relation is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints.




The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as select to identify tuples, project to identify attributes, and join to combine relations. Relations can be modified using the insert, delete, and update operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting.

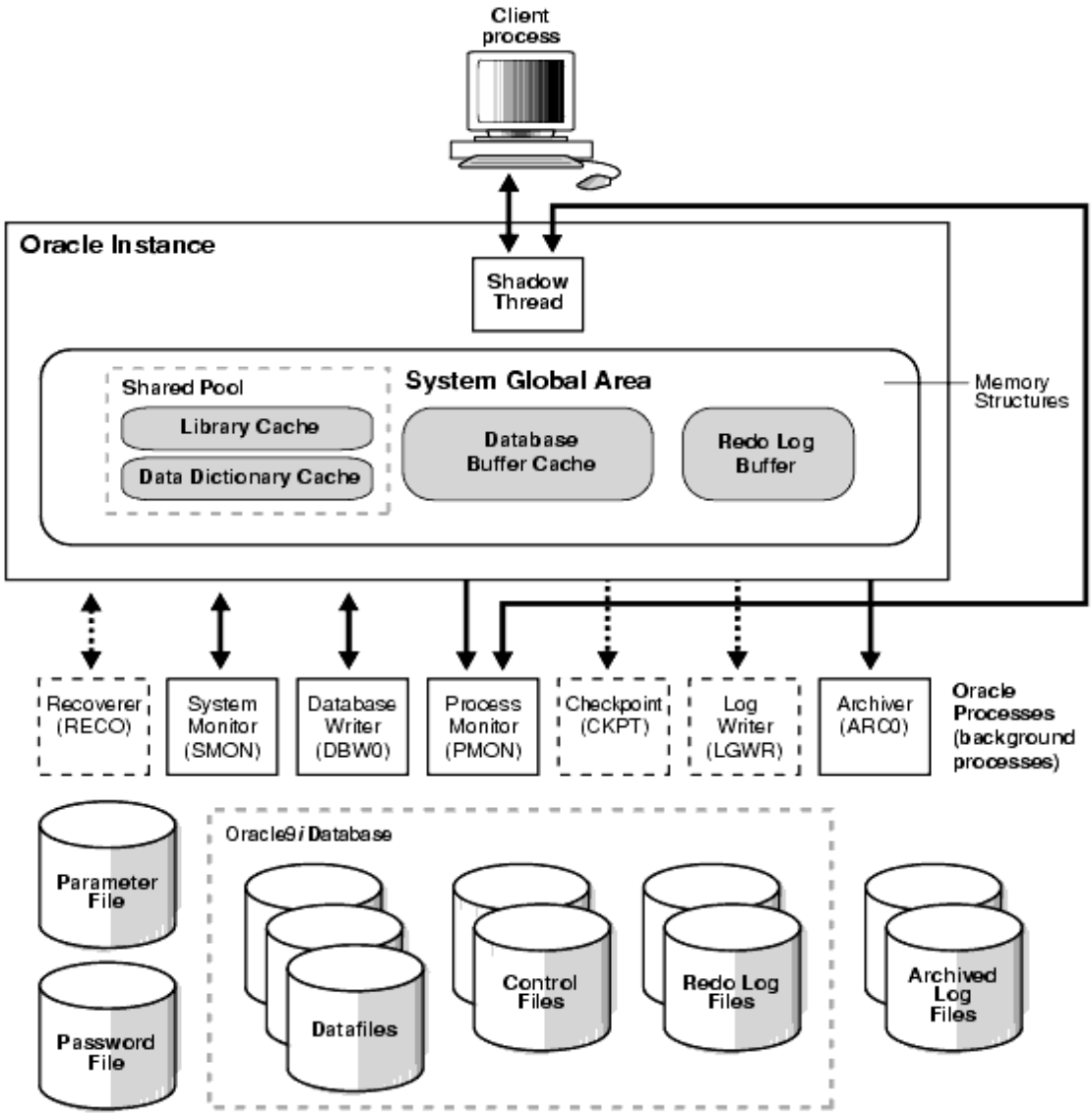
Tuples usually contains a candidate or primary key then obviously it is unique; however, a primary key need not be defined for a row or record to be a tuple. The definition of a tuple requires that it be unique, but does not require a primary key to be defined. Thus its attributes by definition constitute a super key.

Listed are the most commonly used relational databases

- Oracle – The most popular system
- MySQL
- DB2
- Sybase
- Teradata

An Oracle database system - identified by an alphanumeric system identifier - comprises at least one instance of the application, along with data storage. An instance, identified persistently by an instantiation number comprises a set of operating system processes and memory structures that interact with the storage.

Figure below depicts the architecture of Oracle database




## Choosing the right database

## 3. NoSQL

**NoSQL** is increasingly considered a viable alternative to relational databases. This is true as more organizations recognize that operating at scale is better achieved on clusters of commodity servers, and a schema-less data model is often better for the variety and type of data captured and processed today.

A NoSQL database provides a mechanism for storage and retrieval of data that is designed in means other than the tabular relations used in relational databases. They are finding significant and growing industry use in big data and real-time web applications. NoSQL systems are also referred to as "Not only SQL" to emphasize that they may in fact allow SQL-like query languages to be used. Many NoSQL stores compromise consistency in favor of availability and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, the lack of standardized interfaces, and the huge investments already made in SQL by enterprises.

There have been various approaches to classify NoSQL databases, each with different categories and subcategories. Because of the variety of approaches and overlaps it is difficult to maintain an overview of non-relational databases. Nevertheless, the basic classification that most would agree on is based on data model.

A few examples in each category include

- Column – Cassandra, HBase
- Document – MongoDB, Clusterpoint
- Key-value – DynamoDB, FoundationDB, MemcacheDB, Redis, Riak, FairCom c-treeACE
- Graph – Neo4J, OrientDB, Virtuoso

The data structure (e.g. key-value, graph, or document) differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS.

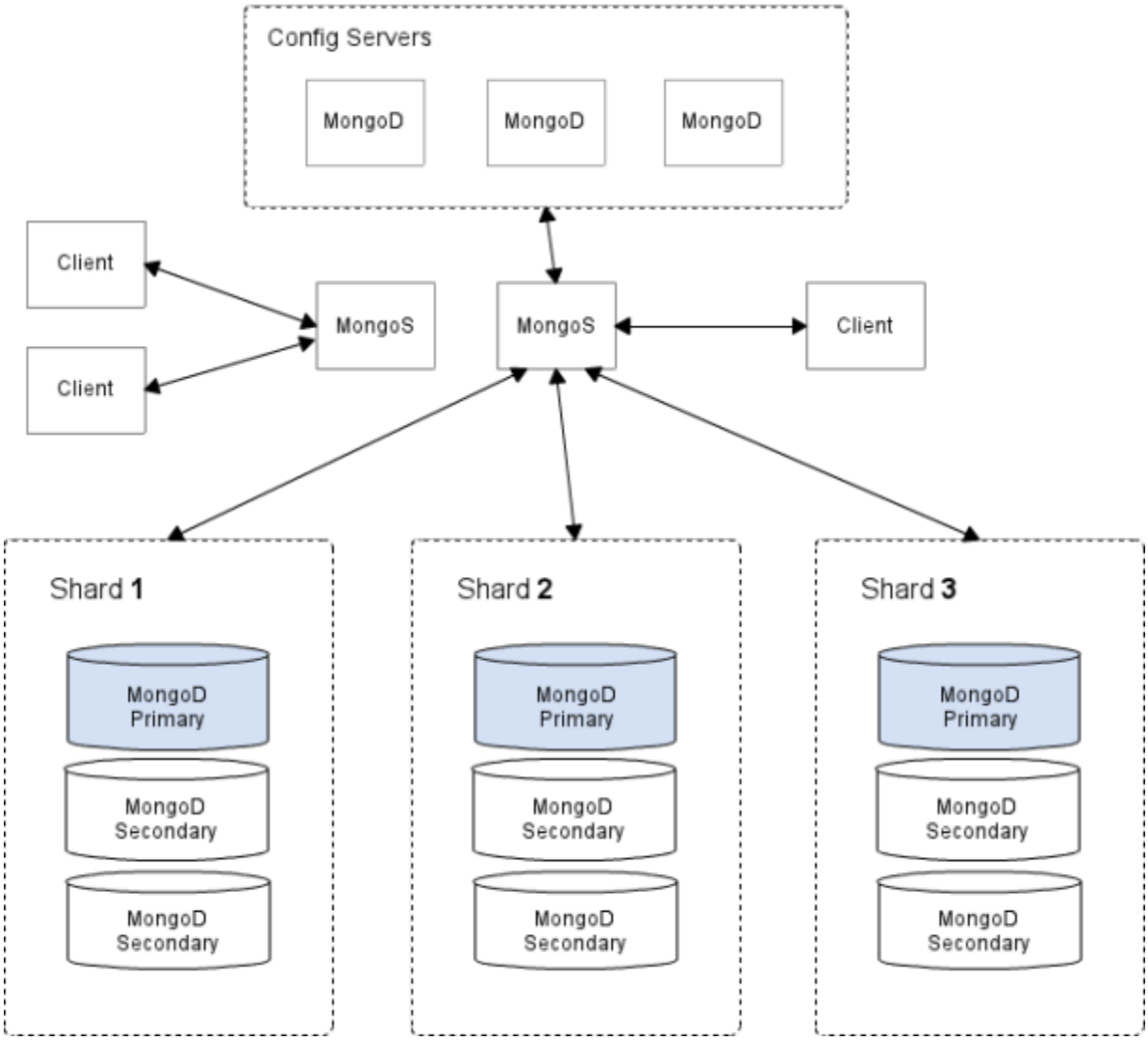
Relational and NoSQL data models are very different. The relational model takes data and separates it into many interrelated tables that contain rows and columns. Tables reference each other through foreign keys that are stored in columns as well. When looking up data, the desired information needs to be collected from many tables and combined before it can be provided to the application. Similarly, when writing data, the write needs to be coordinated and performed on many tables.

NoSQL databases have a very different model. For example, a document-oriented NoSQL database takes the data you want to store and aggregates it into documents using the JSON format. Each JSON document can be thought of as an object to be used by your application. A JSON document might take all the data stored in a row that spans 20 tables of a relational database and aggregate it into a single object. Aggregating this information may lead to duplication of information, but since storage is no longer cost prohibitive, the resulting data model flexibility, ease of efficiently distributing the resulting documents and read and write performance improvements make it an easy an trade-off.

One of the widely used document-oriented NoSQL database is **MongoDB**.

MongoDB is a cross-platform document-oriented database. It eschews the traditional table based relational database structure in favor of JSON-like documents with dynamic schemas , making the integration of data in applications easier and faster. This perfectly scales horizontally using a concept called sharding. The user has to choose a shard key that determines how the data in a collection will be distributed. The data would be split into ranges and distributed across multiple shards. This database can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

Figure below depicts the architecture of MongoDB database




## Choosing the right database

## 4. Advantages over one another

SQL and NoSQL databases share some unique relationships among themselves. Both of them have its respective domains, where they perform their best. We will look into some of the key features and how do they outperform each other in these areas.

### 4.1 Application development

This area majorly focuses on the high coding velocity and nimbleness required by the developers to gain insights on the technology related to the domain.

The key constituency of the DBMS is the application developer community. In the past, the industry always delineated the DBA from the application developer. Now, we need to blur such distinctions and demands very little dependency on dedicated DBAs. The software developer becomes the most important user. The developer requires great agility in the application building process. NoSQL databases have proven to be a better choice in that regard, using object-focused technologies such as JSON. Even if you are a SQL shop, the incremental time to learn emerging database technologies will save lots of development cost over time.

The learning curve on JSON (JavaScript Object Notation) is quite fast and programmers are able to build a prototype in relatively less amount of time. Since many NoSQL offerings include an open system, the community provides many productivity tools, another big advantage over single-vendor proprietary products. Some organizations even offer free courses online that train employees and interested users in how to use the technology.

### 4.2 Operational issues

These issues primarily relate to scaling, performance and high availability. We know from experience that as a database grows in size or the number of users multiplies over the years, many RDBMS-based sites suffer serious performance issues.

Next, we need consultants to provide solutions. Vertical scaling usually comes at a high cost. As processors are added, linear scaling occurs, up to a point where other bottlenecks can appear. Many commercial RDBMS products offer horizontal scaling (clustering) as well, but these are bolted-on solutions and can be very expensive and complex. If an organization is facing such issues, then it should consider NoSQL technologies, as many of them were designed specifically to address these scales (horizontal scaling or scale-out using commodity servers) and performance issues. These new NoSQL technologies are built to host distributed databases for online systems. Redundancy (in triplicate) is implemented here for high availability.

A common complaint about NoSQL databases is that they forfeit consistency in favor of high availability. This is not true for all NoSQL databases. In general, one should consider an RDBMS if one has multi-row transactions and complex joins. In a NoSQL database such as MongoDB – a document can be the equivalent of rows joined across multiple tables, and consistency is guaranteed within that object.

NoSQL databases, in general, avoid RDBMS functions like multi-table joins that can be the cause of high latency. In the new world of big data, NoSQL offers choices of strict to relaxed consistency that need to be looked at on a case-by-case basis. The issues that have been encountered in the relational databases like Oracle, DB2 are mainly related to scaling. Multi-table joins and write to disk operations in relational

databases restricts themselves in scaling. Vertical scaling as everyone is aware about is costly and hence most of the organizations find their options in horizontal scaling.

## 4.3 Data warehousing & Analytics

RDBMS is ideally suited for complex query and analysis. Originally DB2 and Oracle were mostly used for query-intensive workloads. Data from production systems were extracted and transformed using the ETL methodologies and loaded into a warehouse for slicing and dicing. Even in today's world, Hadoop data is sometimes loaded back to RDBMS for reporting purposes. So an RDBMS is a good choice if the query and reporting needs are very critical.

Real time analytics for operational data is better suited to a NoSQL setting. Further, in cases where data is brought together from many upstream systems to build an application, NoSQL is a must. BI tool support for NoSQL databases are new as of now but growing.

## 4.4 Co-existence of RDBMS and NoSQL databases

IBM has announced the implementation of the MongoDB API, data representation, query language and wire protocol, thus establishing a way for mobile and other next-generation applications to connect with enterprise database systems such as IBM's DB2 relational database and its WebSphere extreme Scale data grid. This could usher in a new wave of flexible applications that add significant value by spanning multiple data systems. Data exchange and interoperability will continue to evolve as other industry leaders follow in IBM's footsteps and the functionality of NoSQL databases will continue to evolve over time. Fortune 1000 companies will be well-advised to look at NoSQL database solutions to meet their needs in a data-intensive business world. The rapid adoption of these alternative databases in just a few years is a testament to their attractiveness to the new world of Big Data, where agility, performance, and scalability reign supreme.

RDBMS has several advantages which have grown over the years

- **Strong mathematical basis** – Many researchers have expressed that the RDBMS basis in set theory and the mathematical concept of the relation attributes majorly to its success and dominance. This mathematical aspect is an attraction for RDBMS because it provides an accepted logic and rigor.
- **Declarative syntax** – SQL operations include data insert, query, update and delete, schema creation and modification, and data access control. Even though SQL is described as a declarative language, it also includes procedural elements.
- **A well-known language in Structured Query Language (SQL)**

And these still exists. It would be a mistake to think about this as either/or argument. NoSQL is an alternative that we need to consider when it fits. If using NoSQL gives your project model a better shape in terms of enhancibility, maintenance, performance, scalability then one should definitely go ahead and make its better use. Not all data in this world would be considered relational. For those situations, NoSQL can be helpful. It's not intended to knock SQL or supplant it.

### Choosing the right database

NoSQL address certain limitations of current SQL DBMSs but it doesn't imply any fundamentally new capabilities over previous data models. NoSQL means only no SQL but that doesn't mean the same as no relational. A relational database in principle would make a very good NOSQL solution - it's just that none of the current set of NOSQL products uses the relational model. These databases are designed to excel in speed and volume. To pull this off, NoSQL software will use techniques that can scare the crap out of relational database users — such as not promising that all data is consistent within a system all of the time.

NoSQL has its own set of advantages which have turned to be very promising

- **Higher Scalability** – If your company suddenly finds itself deluged by overnight success, for instance, with customers coming to your Web site by the droves, a relational database would have to be painstakingly replicated and re-partitioned in order to scale up to meet the new demand. This can be handled pretty smoothly with non-relational databases.
- **Performance** – If a relational database had tens or even hundreds of thousands of tables, data processing would generate far more locks on that data, and greatly degrade the performance of the database. Because NoSQL databases have weaker data consistency models, they can trade off consistency for efficiency.

**The clash of RDBMS against NOSQL is reliability and consistency, against scalability and performance. The choice is yours.**