1touch.io
Data Aware Security

NOSQL DATABASE DISCOVERY

WHITE PAPER

APRIL 2021

TABLE OF CONTENTS

*INTRODUCTION TO NOSQL

What are NoSQL Databases?

Data models and schemata define the way a database organizes its data. Data modeling is an activity to represent the flow of data by documenting the system design with the help of text and symbols. However, the activity of data modeling leads to a defined **database schema** - a logical organization or structure of data which provides grouping of data entities such as fields, tables, relations, functions, and documents. In traditional databases (relational/tabular), the schema is defined before the insertion of data: i.e. a schema is inflexible and cannot be changed easily with the evolution of requirements.

As storage costs have rapidly decreased, and the amount of data applications needed to store and query increased the structure and type of data that databases were required to store and manage has greatly expanded to include structured, semi-structured, and polymorphic data. Consequently, defining a database structure in advance has become almost impossible.

NoSQL databases have evolved to address this inflexibility. In the landscape of databases, NoSQL refers to the family of databases that are non-tabular in nature.

Traditional relational databases were not designed to cope with the scalability and flexibility challenges facing modern applications, including:

- The need to store large volumes of data about users, objects, and products
- The need to provide near-real-time transaction processing
- The need to provide high-speed analytics on large volumes of data (big data)
- The need to support flexible data schemas for evolving business environments
- The need to scale applications horizontally with predictable HW & licensing cost

NoSQL database technologies are used by a variety of NoSQL vendors, each tailored to serve specific functions, with many competing vendors and systems in each segment of the market. Each NoSQL solution seeks to address one or more of the above issues, often by providing alternative approaches to long-established relational database norms.

As a variety of NoSQL categories exist, there is significant feature variability both across and within categories. This variance generally occurs with respect to each vendor's implementation of database consistency and transaction management guarantees, query language implementation, and other strategies to improve database performance.

NoSQL databases are designed to easily scale out as and when they grow. Most NOSQL systems have made tradeoffs like removing the multi-platform support and some extra unnecessary features of RDBMS, making them much more lightweight and efficient than their RDMS counterparts.

NoSQL systems tend to be adopted by companies after traditional SQL systems fall short of their requirements for performance, flexibility, or scalability. Social networking, big data and business intelligence applications, all pushed traditional relational databases to their limits, helping spur the creation of nonrelational, horizontally scalable, distributed databases.

NoSQL databases were created to allow developers to store data of all types, regardless of structure, providing more flexibility than traditional, tabular, rigid-schema databases.

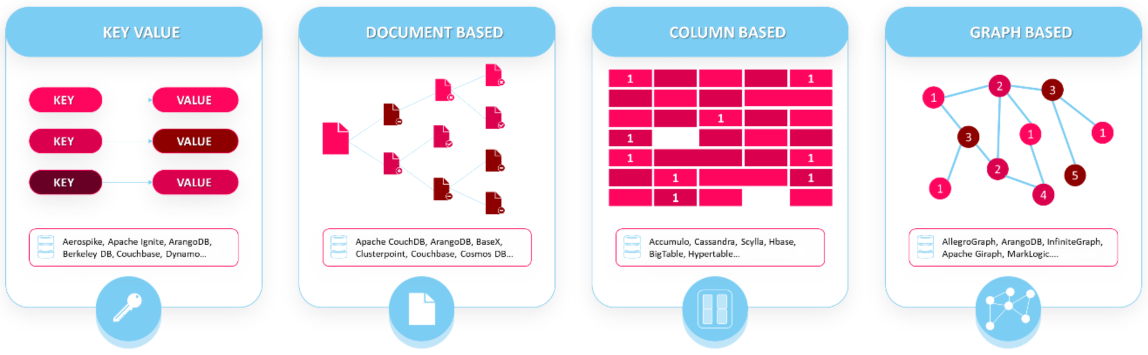
Comparing NoSQL VS. SQL Databases

The following table provides a high-level comparison of the parameters affecting SQL vs. NoSQL databases.

PARAMETER	SQL DATABASES	NOSQL DATABASES & VENDORS
Examples	Oracle, MySQL, MS SQL Server, PostgreSQL	**Key Value**: Aerospike, Apache Ignite, ArangoDB, Berkeley DB, Couchbase, Dynamo, FoundationDB, InfinityDB, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, SciDB, SDBM/Flat File dbm **Document Based**: Apache CouchDB, ArangoDB, BaseX, Clusterpoint, Couchbase, Cosmos DB, eXist-db, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB **Column Based**: Accumulo, Cassandra, Scylla, HBase. **Graph Based**: AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso
Storage	Tables with fixed rows and columns	**Key Value**: key-value pairs **Document Based**: JSON documents **Column Based**: tables, rows, dynamic columns **Graph Based**: nodes & edges
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server, scale-out with limitations)	Horizontal (scale-out across commodity servers)
Joins	Typically required	Typically not required

NoSQL Database Types

Broadly, NoSQL databases can be categorized into 4 types: key-value stores, document databases, column-based stores, and graph-based databases.


KEY VALUE
KEY
VALUE
KEY
VALUE
KEY
VALUE
Aerospike, Apache Ignite, ArangoDB, Berkeley DB, Couchbase, Dynamo...
DOCUMENT BASED
Apache CouchDB, ArangoDB, BaseX, Clusterpoint, Couchbase, Cosmos DB...
COLUMN BASED
Accumulo, Cassandra, Scylla, Hbase, BigTable, Hypertable...
GRAPH BASED
AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic....

KEY-VALUE NOSQL DATABASES

Key-value data stores allow the user to store data in a schema-less manner. The data consists of two parts: a string which represents the **key**, and the actual data - which is referred to as **value**, creating the eponymous "key-value" pair.

The data model is a map/dictionary that allows the user to request the values according to the key specified. Key value data stores prioritize scalability over consistency, hence ad-hoc querying and analytics features like joins and aggregate operations are omitted.

Key value data stores are used to store user sessions or shopping carts, or to get details like favorite products - they are usually utilized in forums, online shopping sites, etc.

The main characteristics of key-value databases include:

- Keys can have a dynamic set of attributes in the key value databases
- Data stored in the database is stored in the alphabetical order
- All activities can be performed on the data i.e. Create, Read, Update, Delete
- All data relationships are stored in the application code (not explicitly spread)
- Key-value can handle very large data loads, and upscale to large data volumes

DOCUMENT BASED NOSQL DATABASES

Document Based Databases store unstructured (text) or semi-structured (XML) documents that are usually hierarchical in nature. Each document consists of a set of **keys** and **values** which are almost the same as the Key Value databases. Documents in the database are addressed using a unique **key** that represents that specific document. These keys may be a simple string or a string that refers to a URL/path.

Document stores are slightly more complex than key-value stores as they allow encasing of the key-value pairs in documents known as **key-document pairs**. Each document in the collection stores **pointers** to its fields. Document based DBs are schema free and are not fixed in nature.

Document oriented databases are used for applications in which data need not be stored in a table with uniform sized fields and when the domain model can be split and partitioned across documents.

The main characteristics of document-based databases include:

- Documents are addressed in the database using unique keys
- Data can be organized via: collections, tags, metadata, directory hierarchies
- A key-value lookup is used to retrieve documents
- Document stores offer great performance and horizontal scalability options.

COLUMN BASED NOSQL DATABASES

Column stores in NoSQL are actually **hybrid row/column stores** - unlike pure relational column databases such as MySQL. Although these databases share the storage concept of columnar databases and columnar extensions to row-based databases, column stores do not store data in tables - but in massively distributed architectures. In column stores, each **key** is associated with one or more **attributes** (columns).

Column store data can be aggregated rapidly with less I/O activity; the data which is stored in the database is based on the sort order of the column family.

Column oriented databases are suitable for data mining and analytic applications, where the storage method is ideal for the common operations performed on the data.

The main characteristics of column-based databases include:

- Faster than row-based databases while querying
- Assignment of storage unit is per column.
- Only required columns are read, so reading is faster
- Offer high scalability in data storage, and rapid querying performance

GRAPH BASED NOSQL DATABASES

Graph databases store data in the form of a graph, consisting of **nodes** and **edges**, where nodes act as the objects and edges act as the relationship between the objects. The database also assigns and tracks **properties**, which are the relevant information related to individual nodes.

Graph databases provide schema-less and efficient storage of semi structured data. The queries are expressed as traversals, thus making graph databases faster than relational databases - Millions of records can be traversed using this technique.

Graph databases can be used for a variety of applications like social networking applications, recommendation software, bioinformatics, content management, security and access control, network and cloud management etc.

The main characteristics of graph-based databases include:

- Graph traversals are executed with constant speed independent of graph size
- There are no set operations involved that decrease performance as seen with join operations in RDBMS
- Graph DBs are easy to scale and whiteboard friendly, ACID compliant, and offer rollback support.

NoSQL Database Types Comparison

The following table provides a high-level comparison of the capabilities and strengths of the 4 main NoSQL database types.

MODEL	QUERIES	DATA	SCHEMA	STRUCTURE	VALUES
Key Value	High performance	High scalability	High flexibility	Primary key with some value	No complexity
Column Based	High performance	High scalability	Moderate flexibility	Row consisting of multiple columns	Low complexity
Document Based	High performance	Variable scalability	High flexibility	JSON in form of tree	Low complexity
Graph Based	Variable performance	Variable scalability	High flexibility	Graph entities & relations	High complexity

•

NOSQL CHALLENGES & 1TOUCH.IO SOLUTIONS

The structural and operational paradigms of the NoSQL databases present a series of challenges to data discovery. Some of these challenges are relevant to specific vendors - while others are generic and inherent to the core NoSQL paradigm.

The 1touch.io approach to data discovery is based on the flexibility of its infrastructure, which is ideal for adapting to the technical requirements presented by a wide variety of NoSQL types and vendors.

NoSQL & Big Data

One of the main drivers behind the development of NoSQL databases Is the need for management and processing of large volumes of data. NoSQL allows for high-performance, agile processing of information at massive scale. It stores unstructured data across multiple processing nodes, as well as across multiple servers. As such, the NoSQL infrastructure has been the solution of choice for some of the largest data warehouses.

This, of course, presents data discovery solutions with a challenge: analyzing ever-growing volumes of data of all types, and formats, across distributed networks and repositories.

Inventa is ideally suited to support NoSQL big data challenges: with a scale-out, distributed architecture that supports as many Analytic Cores as needed to aggregate data into a single, constantly updated inventory.

The platform also utilizes hardware infrastructure to cope with big data discovery: leveraging processing power by utilizing GPU resources among other hardware optimization schemes

Schema-Less Structure

Data discovery is based on identifying specific data structures: NoSQL databases utilize a variety of data structures, with no pre-determined schemas, rendering the discovery process much more challenging.

Other solutions either attempt to force NoSQL data into SQL-like schemas to enable discovery or limit their analysis to a given number of data hierarchies in NoSQL databases.

The 1touch.io approach is not constrained by SQL-based thinking or predetermined schemas and hierarchies. Our solution approaches all data, structured and unstructured, by seeking out the internal data organization that point to relevant strings, utilizing logic generated by Natural Language Processing and machine learning.

By seeking out the natural logic of the language that generates personal Information structures, and building data lineage and relationships using that logic, Inventa is not dependent on schemas, and Is not restricted to any particular level of data hierarchy.

Unlike some solutions, 1touch.io does not attempt to force NoSQL unstructured data into relational schema: since our basic approach is not dependent on identifying schemas, we go directly to the data, and work on discovering personal information by Identifying the structure and relationships between data entities.

Querying Language

NoSQL does not use SQL (Structured Query Language) which is the most common query language used by relational databases. NoSQL does not have a standard query language at all - consequently, it is difficult to switch from one NoSQL database provider to another.

Most NoSQL database providers have created their own query language, for example Cassandra supports CQL (Cassandra query language), MongoDB uses mongo query language etc.

The 1touch.Io data discovery solution is not constrained by the use of SQL queries, or SQL-like query languages: our algorithms utilize a proprietary query layer to optimize discovery results delivered by our platform, which investigates data element structure in any format (table/hierarchy/free text/etc.) This query layer is implemented for both SQL and NoSQL databases.

Transaction Layers

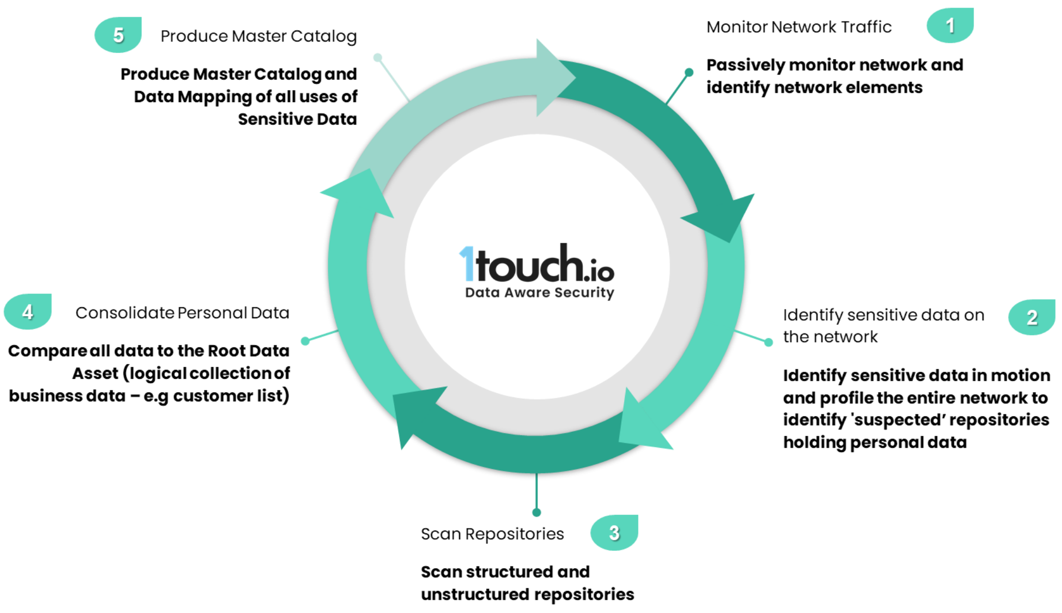
Due to the operational challenges and differences between various vendors - some vendors created a "SQL-like" layer on top of the NoSQL repository to support SQL-based access to the data; these layers are not always standard and limited to subset of the NoSQL features: e.g. **Calvin** is a *transaction scheduling and data replication layer* that provides a layer of ACID-compliant, transactional support on top of a NoSQL database.

Data discovery products face the challenge of accessing NoSQL databases both directly - and through various interface layers.

As with the Inventa approach to querying languages, our discovery plugins Implement an abstraction layer driver to standardize analysis and data retrieval: In SQL databases Inventa discovers metadata schemas/tables/columns etc. and with NoSQL the same process applies an abstraction layer to the data based on the natural language logic inherent to personal data records.

INVENTA 1TOUCH.IO

1touch.io Inventa™ uses a proprietary passive network packet capture process to discover personal sensitive Information stored and moving throughout the organizational network. This allows Inventa to identify repositories (databases, applications, file systems, log files, etc.) where sensitive data resides, and scan them to get full visibility into the depth and breadth of the data. Inventa™ then analyzes and consolidates the data identified by those scans into a structure that allows the user to access, view, and export this data to support a variety of business cases: responding to data access requests, identifying unauthorized data migration, implementing data minimization, alerting to exposure of sensitive data in unprotected locations, and more.


5 Produce Master Catalog
Produce Master Catalog and Data Mapping of all uses of Sensitive Data
Monitor Network Traffic 1
Passively monitor network and identify network elements
1touch.io Data Aware Security
4 Consolidate Personal Data
Compare all data to the Root Data Asset (logical collection of business data – e.g customer list)
Identify sensitive data on the network 2
Identify sensitive data in motion and profile the entire network to identify 'suspected' repositories holding personal data
Scan Repositories 3
Scan structured and unstructured repositories

Inventa™ is the only network-based data privacy solution to implement data-in-motion and data-at-rest techniques to automatically discover information related to identities across a wide variety of structured and unstructured data sources.

Inventa™ creates a data subject-centric picture by correlating all discovered records back to the identity to which the information belongs.

This process allows Inventa™ to discover any sensitive data, whether it can be found on-premise or in the cloud, whether the data is structured or unstructured, and whether it is in motion or at rest, to create a master catalog.

Inventa™ leverages artificial intelligence and machine learning to consolidate and normalize identities to provide a unified view of sensitive personal data.

WORKS CITED

Nayak, A. & Poriya, A. & Poojary, Dikshay. **Type of nosql databases and its comparison with relational databases**. *International Journal of Applied Information Systems*.

Dave, Meenu. **SQL and NoSQL Databases**. International Journal of Advanced Research in Computer Science and Software Engineering.

Madison, Michael; Barnhill, Mark; Napier, Cassie; and Godin, Joy. **NoSQL Database Technologies**. *Journal of International Technology and Information Management*

Farooq, Hina & Mahmood, Azka & Ferzund, Javed. **Do NoSQL Databases Cope with Current Data Challenges**. *International Journal of Computer Science and Information Security*

1touch.io uniquely uses network analytics to help your company discover both sensitive data and its use, even the data you didn't know existed.

1touch.io's Inventa is a data privacy platform with unprecedented data lineage techniques for data discovery and classification. Inventa gives companies complete visibility into their unknown usage of customer data by automating the discovery process and providing them with a comprehensive, accurate, and up-to-date master catalog. This visibility enables you to easily meet regulatory, compliance, and security requirements.