WIKIPEDIA

NoSQL

A **NoSQL** (originally referring to "non-SQL" or "non-relational")[1] database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century,[2] triggered by the needs of Web 2.0 companies.[3][4] NoSQL databases are increasingly used in big data and real-time web applications.[5] NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages or sit alongside SQL databases in polyglot-persistent architectures.[6][7]

Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases),[2] finer control over availability and limiting the object-relational impedance mismatch.[8] The data structures used by NoSQL databases (e.g. key–value pair, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.[9]

Many NoSQL stores compromise consistency (in the sense of the CAP theorem) in favor of availability, partition tolerance, and speed. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages (instead of SQL, for instance, the lack of ability to perform ad-hoc joins across tables), lack of standardized interfaces, and huge previous investments in existing relational databases.[10] Most NoSQL stores lack true ACID transactions, although a few databases have made them central to their designs.

Instead, most NoSQL databases offer a concept of "eventual consistency", in which database changes are propagated to all nodes "eventually" (typically within milliseconds), so queries for data might not return updated data immediately or might result in reading data that is not accurate, a problem known as stale reads.[11] Additionally, some NoSQL systems may exhibit lost writes and other forms of data loss.[12] Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss.[13] For distributed transaction processing across multiple databases, data consistency is an even bigger challenge that is difficult for both NoSQL and relational databases. Relational databases "do not allow referential integrity constraints to span databases".[14] Few systems maintain both ACID transactions and X/Open XA standards for distributed transaction processing.[15] Interactive relational databases share conformational relay analysis techniques as a common feature.[16] Limitations within the interface environment are overcome using semantic virtualization protocols, such that NoSQL services are accessible to most operating systems.[17]

Contents

History

The term *NoSQL* was used by Carlo Strozzi in 1998 to name his lightweight Strozzi NoSQL open-source relational database that did not expose the standard Structured Query Language (SQL) interface, but was still relational.[18] His NoSQL RDBMS is distinct from the around-2009 general concept of NoSQL databases. Strozzi suggests that, because the current NoSQL movement "departs from the relational model altogether, it should therefore have been called more appropriately 'NoREL'",[19] referring to "not relational".

Johan Oskarsson, then a developer at Last.fm, reintroduced the term *NoSQL* in early 2009 when he organized an event to discuss "open-source distributed, non-relational databases".[20] The name attempted to label the emergence of an increasing number of non-relational, distributed data stores, including open source clones of Google's Bigtable/MapReduce and Amazon's DynamoDB.

Types and examples

There are various ways to classify NoSQL databases, with different categories and subcategories, some of which overlap. What follows is a basic classification by data model, with examples:

- Wide column: Accumulo, Cassandra, Scylla, HBase.
- Document: Apache CouchDB, ArangoDB, BaseX, Clusterpoint, Couchbase, Cosmos DB, eXist-db, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
- 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, ZooKeeper
- Graph: AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso

A more detailed classification is the following, based on one from Stephen Yen:[21][22]

Type	Notable examples of this type
Key–value cache	Apache Ignite, Couchbase, Coherence, eXtreme Scale, Hazelcast, Infinispan, Memcached, Redis, Velocity
Key–Value store	ArangoDB, Aerospike, Couchbase, Redis
Key–Value store (eventually consistent)	Oracle NoSQL Database, Dynamo, Riak, Voldemort
Key–value store (ordered)	FoundationDB, InfinityDB, LMDB, MemcacheDB
Tuple store	Apache River, GigaSpaces
Object database	Objectivity/DB, Perst, ZopeDB
Document store	ArangoDB, BaseX, Clusterpoint, Couchbase, CouchDB, DocumentDB, eXist-db, IBM Domino, MarkLogic, MongoDB, Qizx, RethinkDB, Elasticsearch
Wide Column Store	Amazon DynamoDB, Bigtable, Cassandra, Scylla, HBase, Hypertable
Native multi-model database	ArangoDB, Cosmos DB, OrientDB, MarkLogic

Correlation databases are model-independent, and instead of row-based or column-based storage, use value-based storage.

Key–value store

Key–value (KV) stores use the associative array (also called a map or dictionary) as their fundamental data model. In this model, data is represented as a collection of key–value pairs, such that each possible key appears at most once in the collection.[23][24]

The key–value model is one of the simplest non-trivial data models, and richer data models are often implemented as an extension of it. The key–value model can be extended to a discretely ordered model that maintains keys in lexicographic order. This extension is computationally powerful, in that it can efficiently retrieve selective key *ranges*.[25]

Key–value stores can use consistency models ranging from eventual consistency to serializability. Some databases support ordering of keys. There are various hardware implementations, and some users store data in memory (RAM), while others on solid-state drives (SSD) or rotating disks (aka hard disk drive (HDD)).

Document store

The central concept of a document store is that of a "document". While the details of this definition differ among document-oriented databases, they all assume that documents encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, and JSON and binary forms like BSON. Documents are addressed in the database via a unique *key* that represents that document. Another defining characteristic of a document-oriented database is an API or query language to retrieve documents based on their contents.

Different implementations offer different ways of organizing and/or grouping documents:

- Collections
- Tags
- Non-visible metadata

- Directory hierarchies

Compared to relational databases, collections could be considered analogous to tables and documents analogous to records. But they are different: every record in a table has the same sequence of fields, while documents in a collection may have fields that are completely different.

Graph

Graph databases are designed for data whose relations are well represented as a graph consisting of elements connected by a finite number of relations. Examples of data include social relations, public transport links, road maps, network topologies, etc.

Graph databases and their query language

Name	Language(s)	Notes
AllegroGraph	SPARQL	RDF triple store
Amazon Neptune	Gremlin, SPARQL	Graph database
ArangoDB	AQL, JavaScript, GraphQL	Multi-model DBMS Document, Graph database and Key-value store
DEX/Sparksee	C++, Java, C#, Python	Graph database
FlockDB	Scala	Graph database
IBM DB2	SPARQL	RDF triple store added in DB2 10
InfiniteGraph	Java	Graph database
MarkLogic	Java, JavaScript, SPARQL, XQuery	Multi-model document database and RDF triple store
Neo4j	Cypher	Graph database
OpenLink Virtuoso	C++, C#, Java, SPARQL	Middleware and database engine hybrid
Oracle	SPARQL 1.1	RDF triple store added in 11g
OrientDB	Java, SQL	Multi-model document and graph database
OWLIM	Java, SPARQL 1.1	RDF triple store
Profium Sense	Java, SPARQL	RDF triple store
Sqrrl Enterprise	Java	Graph database

Object database

- db4o
- GemStone/S
- InterSystems Caché
- JADE
- ObjectDatabase++
- ObjectDB
- Objectivity/DB
- ObjectStore

- ODABA
- Perst
- OpenLink Virtuoso
- Versant Object Database
- ZODB

Tabular

- Apache Accumulo
- Bigtable
- Apache Hbase
- Hypertable
- Mnesia
- OpenLink Virtuoso

Tuple store

- Apache River
- GigaSpaces
- Tarantool
- TIBCO ActiveSpaces
- OpenLink Virtuoso

Triple/quad store (RDF) database

- AllegroGraph
- Apache JENA (It is a framework, not a database)
- MarkLogic
- Ontotext-OWLIM
- Oracle NoSQL database
- Profium Sense
- Virtuoso Universal Server

Hosted

- Amazon DynamoDB
- Amazon DocumentDB
- Amazon SimpleDB
- Clusterpoint database
- Cloudant Data Layer (CouchDB)
- Datastore on Google Appengine
- Freebase
- Microsoft Azure Storage services
- OpenLink Virtuoso

Multivalue databases

- D3 Pick database
- Extensible Storage Engine (ESE/NT)
- InfinityDB
- InterSystems Caché
- jBASE Pick database
- mvBase Rocket Software
- mvEnterprise Rocket Software
- Northgate Information Solutions Reality, the original Pick/MV Database
- OpenQM
- Revelation Software's OpenInsight (Windows) and Advanced Revelation (DOS)
- UniData Rocket U2
- UniVerse Rocket U2

Multimodel database

- Apache Ignite[26][27]
- ArangoDB
- Couchbase
- FoundationDB
- MarkLogic
- OrientDB
- Cosmos DB
- Oracle Database

Performance

Ben Scofield rated different categories of NoSQL databases as follows:[28]

Data model	Performance	Scalability	Flexibility	Complexity	Functionality
Key–value store	high	high	high	none	variable (none)
Column-oriented store	high	high	moderate	low	minimal
Document-oriented store	high	variable (high)	high	low	variable (low)
Graph database	variable	variable	high	high	graph theory
Relational database	variable	variable	low	moderate	relational algebra

Performance and scalability comparisons are sometimes done with the YCSB benchmark.

Handling relational data

Since most NoSQL databases lack ability for joins in queries, the database schema generally needs to be designed differently. There are three main techniques for handling relational data in a NoSQL database. (See table Join and ACID Support for NoSQL databases that support joins.)

Multiple queries

Instead of retrieving all the data with one query, it is common to do several queries to get the desired data. NoSQL queries are often faster than traditional SQL queries so the cost of additional queries may be acceptable. If an excessive number of queries would be necessary, one of the other two approaches is more appropriate.

Caching, replication and non-normalized data

Instead of only storing foreign keys, it is common to store actual foreign values along with the model's data. For example, each blog comment might include the username in addition to a user id, thus providing easy access to the username without requiring another lookup. When a username changes however, this will now need to be changed in many places in the database. Thus this approach works better when reads are much more common than writes.[29]

Nesting data

With document databases like MongoDB it is common to put more data in a smaller number of collections. For example, in a blogging application, one might choose to store comments within the blog post document so that with a single retrieval one gets all the comments. Thus in this approach a single document contains all the data you need for a specific task.

ACID and join support

A database is marked as supporting ACID properties (Atomicity, Consistency, Isolation, Durability) or join operations if the documentation for the database makes that claim. The degree to which the capability is fully supported in a manner similar to most SQL databases is sufficiently characterized through simple dialogue.

Database	ACID	Joins
Aerospike	Yes	No
Apache Ignite	Yes	Yes
ArangoDB	Yes	Yes
Couchbase	Yes	Yes
CouchDB	Yes	Yes
Db2	Yes	Yes
InfinityDB	Yes	No
LMDB	Yes	No
MarkLogic	Yes	Yes[nb 1]
MongoDB	Yes	Yes[nb 2]
OrientDB	Yes	Yes[nb 3]

1. Joins do not necessarily apply to document databases, but MarkLogic can do joins using semantics.[30]
2. MongoDB does not support joining from a shared collection.[31]

3. OrientDB can resolve 1:1 joins using links by storing direct links to foreign records.[32]

See also

- CAP theorem
- Comparison of object database management systems
- Comparison of structured storage software
- Correlation database
- Database scalability
- Distributed cache
- Faceted search
- MultiValue database
- Multi-model database
- Triplestore
- Schema-agnostic databases

References

1. http://nosql-database.org/ "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points : being non-relational, distributed, open-source and horizontally scalable".
2. Leavitt, Neal (2010). "Will NoSQL Databases Live Up to Their Promise?" (http://www.leavcom.com/pdf/NoSQL.pdf) (PDF). *IEEE Computer*.
3. Mohan, C. (2013). *History Repeats Itself: Sensible and NonsenSQL Aspects of the NoSQL Hoopla* (http://openproceedings.eu/2013/conf/edbt/Mohan13.pdf) (PDF). Proc. 16th Int'l Conf. on Extending Database Technology.
4. "Amazon Goes Back to the Future With 'NoSQL' Database" (https://www.wired.com/2012/01/amazon-dynamodb/). WIRED. 19 January 2012. Retrieved 6 March 2017.
5. "RDBMS dominate the database market, but NoSQL systems are catching up" (http://db-engines.com/en/blog_post/23). DB-Engines.com. 21 November 2013. Retrieved 24 November 2013.
6. "NoSQL (Not Only SQL)" (http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL). "NoSQL database, also called Not Only SQL"
7. Fowler, Martin. "NosqlDefinition" (http://martinfowler.com/bliki/NosqlDefinition.html). "many advocates of NoSQL say that it does not mean a "no" to SQL, rather it means Not Only SQL"
8. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Educational Publishers Inc, 2009, ISBN 978-0321826626.
9. Vogels, Werner (18 January 2012). "Amazon DynamoDB – a Fast and Scalable NoSQL Database Service Designed for Internet Scale Applications" (http://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html). All Things Distributed. Retrieved 6 March 2017.
10. Grolinger, K.; Higashino, W. A.; Tiwari, A.; Capretz, M. A. M. (2013). "Data management in cloud environments: NoSQL and NewSQL data stores" (http://www.journalofcloudcomputing.com/content/pdf/2192-113X-2-22.pdf) (PDF). Aira, Springer. Retrieved 8 January 2014.
11. "Jepsen: MongoDB stale reads" (https://aphyr.com/posts/322-call-me-maybe-mongodb-stale-reads). *Aphyr.com*. 20 April 2015. Retrieved 6 March 2017.
12. "Large volume data analysis on the Typesafe Reactive Platform" (http://www.slideshare.net/MartinZapletal/zapletal-martinlargevolumedataanalytics). *Slideshare.net*. Retrieved 6 March 2017.
13. Fowler, Adam. "10 NoSQL Misconceptions" (http://www.dummies.com/how-to/content/10-nosql-misconceptions.html). *Dummies.com*. Retrieved 6 March 2017.

14. "No! to SQL and No! to NoSQL | So Many Oracle Manuals, So Little Time" (https://iggyfernandez.wordpress.com/2013/07/28/no-to-sql-and-no-to-nosql/). *Iggyfernandez.wordpress.com*. Retrieved 6 March 2017.
15. Chapple, Mike. "The ACID Model" (http://databases.about.com/od/specificproducts/a/acid.htm). *about.com*.
16. Fiore, S. (2011). *Grid and cloud database management*. Springer Science & Business Media. p. 210.
17. Lawrence, Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB (2014). "Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB". *International Conference on Computational Science and Computational Intelligence 1*.
18. Lith, Adam; Mattson, Jakob (2010). "Investigating storage solutions for large data: A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data" (http://publications.lib.chalmers.se/records/fulltext/123839.pdf) (PDF). Göteborg: Department of Computer Science and Engineering, Chalmers University of Technology. p. 70. Retrieved 12 May 2011. "Carlo Strozzi first used the term NoSQL in 1998 as a name for his open source relational database that did not offer a SQL interface[...]"
19. "NoSQL Relational Database Management System: Home Page" (http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page). Strozzi.it. 2 October 2007. Retrieved 29 March 2010.
20. "NoSQL 2009" (https://web.archive.org/web/20110716174012/http://blog.sym-link.com/2009/05/12/nosql_2009.html). Blog.sym-link.com. 12 May 2009. Archived from the original (http://blog.sym-link.com/2009/05/12/nosql_2009.html) on 16 July 2011. Retrieved 29 March 2010.
21. Yen, Stephen. "NoSQL is a Horseless Carriage" (https://dl.dropboxusercontent.com/u/2075876/nosql-steve-yen.pdf) (PDF). NorthScale. Retrieved 26 June 2014.
22. Strauch, Christof. "NoSQL Databases" (http://www.christof-strauch.de/nosqldbs.pdf) (PDF). pp. 23–24. Retrieved 27 August 2017.
23. Sandy (14 January 2011). "Key Value stores and the NoSQL movement" (http://dba.stackexchange.com/a/619). http://dba.stackexchange.com/questions/607/what-is-a-key-value-store-database: Stackexchange. Retrieved 1 January 2012. "Key–value stores allow the application developer to store schema-less data. This data usually consists of a string that represents the key, and the actual data that is considered the value in the "key–value" relationship. The data itself is usually some kind of primitive of the programming language (a string, an integer, or an array) or an object that is being marshaled by the programming language's bindings to the key-value store. This structure replaces the need for a fixed data model and allows proper formatting."
24. Seeger, Marc (21 September 2009). "Key-Value Stores: a practical overview" (http://blog.marc-seeger.de/assets/papers/Ultra_Large_Sites_SS09-Seeger_Key_Value_Stores.pdf) (PDF). http://blog.marc-seeger.de/2009/09/21/key-value-stores-a-practical-overview/: Marc Seeger. Retrieved 1 January 2012. "Key–value stores provide a high-performance alternative to relational database systems with respect to storing and accessing data. This paper provides a short overview of some of the currently available key–value stores and their interface to the Ruby programming language."
25. Katsov, Ilya (1 March 2012). "NoSQL Data Modeling Techniques" (http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/). Ilya Katsov. Retrieved 8 May 2014.
26. https://apacheignite.readme.io/docs Ignite Documentation
27. https://www.infoworld.com/article/3135070/data-center/fire-up-big-data-processing-with-apache-ignite.html fire-up-big-data-processing-with-apache-ignite
28. Scofield, Ben (14 January 2010). "NoSQL - Death to Relational Databases(?)" (http://www.slideshare.net/bscofield/nosql-codemash-2010). Retrieved 26 June 2014.
29. "Moving From Relational to NoSQL: How to Get Started" (https://resources.couchbase.com/c/relational-no-sql-wp?x=3-FqHm). *Couchbase.com*. Retrieved 11 November 2019.

30. "Can't do joins with MarkLogic? It's just a matter of Semantics! - General Networks" (http://www.gennet.com/big-data/cant-joins-marklogic-just-matter-semantics/). *Gennet.com*. Retrieved 6 March 2017.
31. "Sharded Collection Restrictions" (https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#sharded-collection-restrictions). *docs.mongodb.com*. Retrieved 24 January 2020.
32. "SQL Reference · OrientDB Manual" (http://orientdb.com/docs/2.2.x/SQL.html#joins). *OrientDB.com*. Retrieved 24 January 2020.

Further reading

- Sadalage, Pramod; Fowler, Martin (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley. ISBN 978-0-321-82662-6.
- McCreary, Dan; Kelly, Ann (2013). *Making Sense of NoSQL: A guide for managers and the rest of us*. ISBN 9781617291074.
- Wiese, Lena (2015). *Advanced Data Management for SQL, NoSQL, Cloud and Distributed Databases*. DeGruyter/Oldenbourg. ISBN 978-3-11-044140-6.
- Strauch, Christof (2012). "NoSQL Databases" (http://www.christof-strauch.de/nosqldbs.pdf) (PDF).
- Moniruzzaman, A. B.; Hossain, S. A. (2013). "NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison". arXiv:1307.0191 (https://arxiv.org/abs/1307.0191). Bibcode:2013arXiv1307.0191M (https://ui.adsabs.harvard.edu/abs/2013arXiv1307.0191M).
- Orend, Kai (2013). "Analysis and Classification of NoSQL Databases and Evaluation of their Ability to Replace an Object-relational Persistence Layer". CiteSeerX 10.1.1.184.483 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.184.483).
- Krishnan, Ganesh; Kulkarni, Sarang; Dadbhawala, Dharmesh Kirit. "Method and system for versioned sharing, consolidating and reporting information" (https://www.google.com/patents/US7383272?pg=PA1&dq=ganesh+krishnan&hl=en&sa=X).

External links

- Strauch, Christoph. "NoSQL whitepaper" (http://www.christof-strauch.de/nosqldbs.pdf) (PDF). Stuttgart: Hochschule der Medien.
- Edlich, Stefan. "NoSQL database List" (http://nosql-database.org/).
- Neubauer, Peter (2010). "Graph Databases, NOSQL and Neo4j" (http://www.infoq.com/articles/graph-nosql-neo4j).
- Bushik, Sergey (2012). "A vendor-independent comparison of NoSQL databases: Cassandra, HBase, MongoDB, Riak" (http://www.networkworld.com/article/2160905/tech-primers/a-vendor-independent-comparison-of-nosql-databases--cassandra--hbase--mongodb--riak.html). NetworkWorld.
- Zicari, Roberto V. (2014). "NoSQL Data Stores – Articles, Papers, Presentations" (http://www.odbms.org/category/downloads/nosql-data-stores/nosql-data-stores-articles/). *odbms.org*.