SURVEY FOR NOSQL DATABASES

Jeff Xing

ABSTRACT

The bigdata technologies raise more and more attentions in recent years, as the size of data, which the internet companies usually deal with, has scale to TBs or PBs. The traditional databases meet great challenges for the bigdata demands, especially the large scale and high-concurrency applications. NoSQL databases are created for the raising demand for store and query for bigdata. This survey paper will cover three representative databases of three different NoSQL types according to CS5425 course requirement and summarize the several key points, the course may certain, of each NoSQL database.

1 Introduction

The traditional relational databases were developed in the early 1970's with it own query language model – Structured Query Language (SQL) [6]. When we use SQL, we often refer to the structured query language as well as the relational database management system (RDBMS), which the SQL is designed for. Although relational databases have been widely used since its birth, the relational databases come out a variety of limits with the rapid growth of the stored and analyzed data. For large scale-size data, the data query, storage and management of relational databases become far less efficient. Specifically, the relational databases:

- Slow in reading and writing: the relational databases have own logic complexity. With the data size increases, it is prone to bring about deadlocks and other concurrency issues, which will lose the efficiency of reading and writing.
- Limited in storage capacity: the existing relational databases cannot support for bigdata applications in search engine.
- Limited in scalability: The multi-table correlation mechanism of the relational databases limits the expansion of the databases.

In order to solve the limits of relational databases, a variety of flexible and scalable NoSQL databases were developed. According to [5], currently, there are more than 225 NoSQL databases.

The NoSQL can be classified into 12 main types, i.e., Wide Column Store, Document Store, Key Value Store, Graph Databases, Multimodel Databases, Object Databases, Grid & Cloud Database, XML Databases, Multidimensional Databases, Multivalue Databases, Event Sourcing, Time Series / Streaming Databases [5]. In this survey paper, according to CS5425 course requirement, we select three types of NoSQL, i.e., Wide Column Store, Key Value Store and Graph Databases.

From [15], we select three representative and popular databases for each type of NoSQL, i.e., Redis (Wide Column Store), Cassandra (Wide Column Store) and Neo4j (Graph Databases).All the three NoSQL databases have high ranks in the [15], which means that they are all very popular in the database community. That is the main reason we choose them from the 225 NoSQL databases. The following sections, we will give brief overviews of each database. Then, for each database, we try to answer the scale performance under extreme large dataset, the methods to handle the data dynamics, the key difference from the traditional relational databases, the key limits and comparisons with other databases in the same category.

2 Wide Column Store: Redis

First of all, we will introduce the key-value store. Key-value storage has key-value data model which means that a value corresponds to a key. The data structure is the same ad the dictionary or hash table structure. The records stored in key-value databases are stored and retrieved using a key that uniquely identifies the record and is used to quickly find the data within the database. Because the structure is simple, the query speed of key-value storage is faster than the traditional relational databases. It also supports mass storage and high concurrency.

2.1 Overview

Redis is a very popular key-value memory database [13]. Redis was mainly developed by Salvatore Sanfilippo and is currently sponsored by Redis Lab. Now, Redis is the most popular key-value databases ranked by [15]. The Redis has the following characteristics:

- Key-value memory database: The memory database means that when the database runs, all the data and operation will be in the memory and save the data to the hard disk periodically. The whole dataset is kept in-memory, and therefore the processed data cannot exceed the size of physical memory. The characteristics of pure memory operation contribute to Redis' high performance, i.e., it can handle more than 100,000 read or write operation per second [7]. From [1], where the authors do several experiments to compare several databases' performance, the Redis ranks the first two in the experiment tasks.
- Redis also supports master-slave asynchronous replication as well as very fast non-blocking first synchronization. Being very simple, it is fast for many key-value implementations as for many basic operations.
- Data structures in Redis are more generally called Redis Datatypes, which have strings, lists, sets, sorted sets and hashes. Redis also provides commands that can be used to make some modification of these types, e.g., listing supports normal list operations.

2.2 Questions to Answer

2.2.1 Why choose Redis?

First, Redis is one of most popular key-value NoSQL databases. In the database ranking list [15], the Redis rank 8th currently, as well as the top among all the key-value databases. Beside the high rank, Redis can offer high performance, replication, and a unique data model, which can produce a platform for undertaking different tasks. The writing speed of Redis can be more than 81000 times/sec and the reading speed can be more than 110000 times/sec. Because Redis can support for five different datatypes, Redis can solve varieties of problems by mapping them to what Redis can deal with. In addition, Redis is suitable for high performance bigdata system. The features, e.g., replication, persistence, and client-side sharding, enable Redis to set up scalable system, which can scale up to hundreds of gigabytes of data and millions of requests per second.

2.2.2 How Redis scale to handle extremely large dataset?

As Redis is memory database. The main drawback of Redis is that the capacity is limited by physical memory, so the scalability of Redis is usually poor. Consequently, if the data size Redis handles with is extreme, i.e., far larger than the available memory, the Redis will run out of memory. According to the documents in [13], Redis will either be killed by OS, crash with an error, or will start to slow down. If some swap space is configured, the server will utilize the swap space. However, the performance if Redis will fall sharply. Another way to handle the extremely large dataset is to split your data set into multiple Redis instances. According to [13], Redis Lab are planning to develop a "Redis on flash" solution, which enable the database to use a mixed RAM/flash approach for larger datasets. However, this feature is not available in current Redis database.

2.2.3 How well does Redis adapt to data dynamics?

Usually, we use Redis to cache the data in memory while using another database to store the data in the disk. Consequently, Redis need to keep consistency with the data store in the other database. There are a frameworks to adapt the data dynamics. The workflow diagrams shown in Figure 1 is a way for Redis to adapt to the data changes. This way is to use the Redis directly. We will check if the updated data is in dataset. If the data is available in the Redis, then the data in Redis will be updated. Then updated date will be returned from Redis. All the updated data in Redis will be synchronized to the data in hard disk periodically. If the data is not in Redis, then the data in hard disk will updated and the updated data will be cached in Redis. The updated data will be return from Redis. This way is suitable for large amount of data changes. The speed is also very fast.

2.2.4 What are the key differences between Redis and relational databases?

In relational databases, we use the SQL query to relate the data in different tables. Redis is a typical NoSQL database. There is no tables nor no data-basedefined or -enforced way of relating data with other data in Redis [2]. In a word,

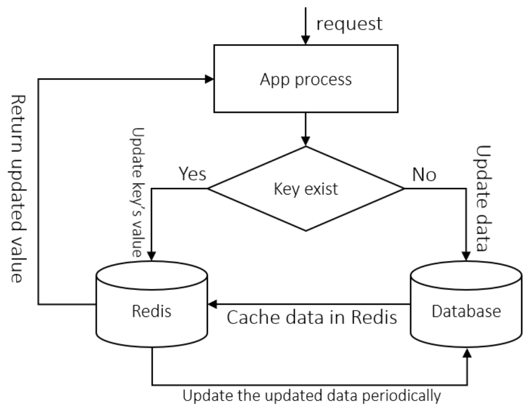

Figure 1: A frameworks to adapt the data dynamics

Redis is fit for non-structured data, improving performance for scalability issues and cache layer, while the relational databases are fit for structured data, heavy transactions and complex lookups.

2.2.5 What are limitations of Redis?

- First, as Redis is a in-memory database., which means that all the data is in memory. Consequently, the data cannot be greater than the memory capacity. In a word, the main drawback is that capacity of Redis is limited by memory size, so Redis cannot be alone used as large data storage. Usually, we use Redis with other type of databases.
- Second, there is no query language (only commands) and no support for a relational algebra. You cannot use SQL-like query. All data query need the participation of developers, and proper data access paths also need to be designed.
- Third, A unique Redis instance is not scalable. It only runs on one CPU core in single-threaded mode. To get more scalability, several Redis instances should be deployed. In a word, the scalability of Redis is poor.

2.2.6 How is Redis compared with other key-value database?

There are mainly two differences [16]:

- First, Redis has a different evolution path compared with the other key-value databases. The values in Redis can contain more complex datatypes. The atomic operations can be defined on those data types. Without any additional abstraction layers, the programmer can directly modify the datatype and the operations in them.
- Second, Redis is an in-memory database and the data is persisted in hard disk. There is a tradeoff about where very high write and read speed is achieved, because data sets cannot be larger than memory size. Another advantage of in memory databases is that the more complex data structures is easier to operated on in memory. Thus, Redis can do a lot of operations with little internal complexity. At the same time, the data format of Redis are more compact and always generated in an append-only fashion.

3 Wide Column: Cassandra

Wide column stores are also called extensible record stores. It stores data in form of records which can hold enormous numbers of dynamic columns. The wide column stores can be treated as two-dimensional key-value stores, because the column names and the keys are not constant, and a record if wide column stores can hold billions of columns.

The wide column databases is also known as column family databases. The main storage architecture of the wide column database is shown in Figure 2. In a typical wide column database, the data is stored in cells of columns, which is also grouped into the column families. A cell can be mapped to a certain column which in turn can be mapped to a certain column family. A column family set can be identified by using row key. By using the row keys on a column family set, we can do the reading and writing operations. Note that the performance can be enhanced by storing the columns as continuous entry on the disk.

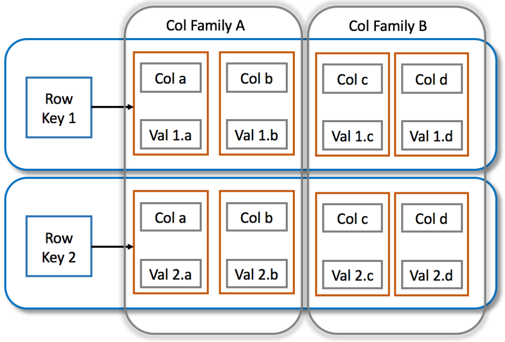

Figure 2: Storage architecture of the wide column database

3.1 Overview

Cassandra is a wide-column database based on ideas of BigTable [4], which is the original wide column store. It is designed to deal with big data across many servers, providing high availability with high successful guarantee. As Cassandra is an open source Apache project available in [3], varieties of documents and communities can be found in [3]. Cassandra is originally developed by Facebook to inbox search problem, where they need to deal with large amount of data in a way that was difficult to scale with traditional methods [8]. The detailed introduction information can be found the original release paper [11]. From varieties of literature [3, 8, 11], we summarize several main features of Cassandra.

- Distributed and decentralized: Cassandra is distributed which means Cassandra is running on multiple servers while can be treated as a unified whole to users. Most of a Cassandra design work is on make it run across many servers and optimize the performance across multiple datacenter racks. Cassandra is decentralized which means that every node in Cassandra is identical. No node in Cassandra run a distinct operation which is different from others. In another word, Cassandra is unlike the other master and slaves models, e.g., spark or Hadoop.
- Elastic Scalability: Scalability means that a system can continue serving a greater number of requests with little loss in performance. The Cassandra can seamless add or remove new nodes and start or shut down these nodes without major disruption or reconfiguration of the entire cluster, which means that you need not to restart your process. Cassandra will also do the data rebalance automatically.
- High Availability and Fault Tolerance: Availability is the ability to fulfill requests. For Cassandra, the failed datacenter can be replaced without downtime. The data can also be replicated to multiple datacenter to offer higher local performance and prevent the potential downtime.
- Tuneable Consistency: Consistency means that the query can always get the latest written value. Actually, Cassandra can offer tuneable consistent, which mean that it can allows you to decide which level of consistency you require, while in balance with the level of availability.

3.2 Questions to Answer

3.2.1 Why choose Cassandra?

First, Cassandra is one of most popular wide column NoSQL databases. In the database ranking list [15], the Cassandra ranks 8th currently, as well as the top among all the key-value databases. Beside the high rank, Cassandra can offer elastic scalability, high availability and tuneable consistency, which have been explained in the overview part. In addition, it can also provide high performance. It consistently shows very fast throughput for writes per second no matter whether the computer is physical hardware or virtual machines. As you can add more servers, you can maintain all of Cassandra's desirable properties without performance sacrifices.

3.2.2 How Cassandra scale to handle extremely large dataset?

Cassandra is very easy to deal with extremely large dataset, Cassandra can offer elastic scalability. Specifically, Cassandra can offer outstanding horizontal scalability. For extremely large dataset which may exceed the capacity of the existing system, Cassandra can add more machines to have parts of the data on them so that only a few numbers of

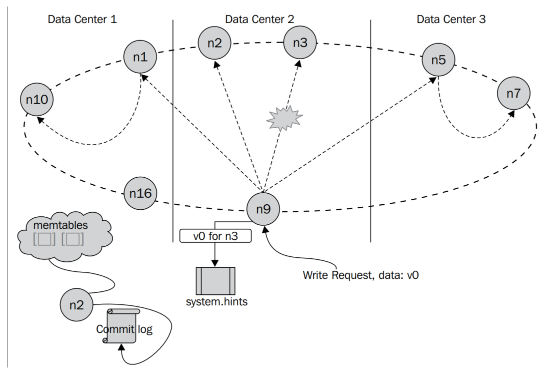

Figure 3: Writing process in Cassandra [12]

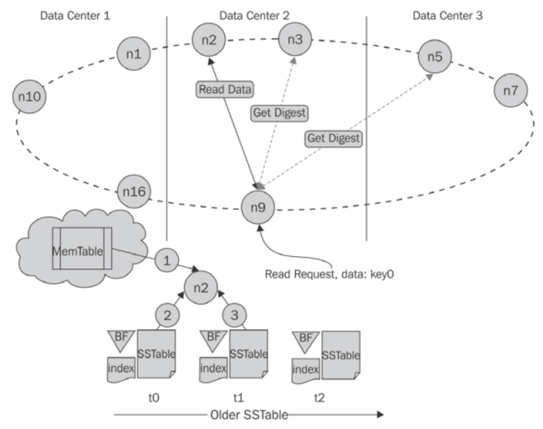

Figure 4: Reading process in Cassandra [12]

machines have to bear the entire burden of large dataset. In a word, As dataset and demands grow, just simply launching new instances to increase the size of the Cassandra cluster is enough for the large dataset challenge.

3.2.3 How well does Cassandra adapt to data dynamics?

For the data change, it may concern with the writing part and the reading part.

In writing part, the user may connect to any of the Cassandra nodes and make a request for change. This connected node is called coordinate node [12]. When a node in Cassandra cluster receives such a request, it will first start a service call `StorageProxy`. Because the node may be not the target node where the change is made. The task of `StorageProxy` is to look for the nodes which has the data to be changed or is the right node to insert a record. If the target nodes are identified, the coordinate node will send a `RowMutaion` message to those node and wait for the replies from those nodes. When the number of the replies is large than `ConsistencyLevel`, then the operations can be made in those replying nodes. Figure 3 shows what happens during a writing process.

For reading part, the `StorageProxy` of a node will return a list of nodes which have the requested data. The connected coordinator node will send a command to the closest nodel to do reading operation and return the data. Based on `ConsistencyLevel`, other nodes will send a command to perform a read operation and send just the digest of the result. In this process, it can update all the nodes with the latest value, making all replicas consistent. With correct read and write consistency levels, the system can guarantee an latest response all the time. Figure 4 shows the above reading process.

3.2.4 What are the key differences between Cassandra and relational databases?

There several main differences between Cassandra and relational databases [10]:

- Cassandra is usually used to handle the unstructured data while the relational databases is used to handle the structured data.

- Cassandra has more flexible schema while the relational databases has fixed schema.
- The components of the table are different, although Cassandra and relational databases share table as the basic data structure. The table of Cassandra is a list of “nest key-value pairs” while the table of relational databases is an array of arrays.
- The outermost container of Cassandra is the keyspace while the outermost container of relational databases is database.
- In Cassandra, tables or column families are the entity of a keyspace while tables are the entities of a database for relational databases.
- Row is the unit of replication for Cassandra while the row is an individual record for relational databases.
- The column is a unit of storage for Cassandra while the column represent the attributes of a relation for relational databases.
- Collections are used to represent the relationship for Cassandra while foreign key and joins are used to get the relationship for relational databases.

3.2.5 What are limitations of Cassandra?

Although Cassandra has perfect features listed in Section 3.1, there still exist some drawbacks of Cassandra.

- No Support for Aggregates and not suitable for Ad-Hoc Queries: Alough newer versions of Cassandra will have limited support for aggregations with a single partition, it is still limited. Because Cassandra is a key-value store, the aggregate operations are resource intensive. Consequently, the ad-hoc query and analysis do not have good performance for Cassandra.
- Unpredictable Performance: Because Cassandra has many different asynchronous jobs and background tasks that are not scheduled by the user, the performance can be unpredictable.
- JVM-based: If it is required to store huge amount of data, JVM is required to manage the memory which itself is a language. Consequently, garbage collection is not done by application but by a language in Cassandra, which means that it is required expert knowledge of the language the database.

3.2.6 How is Cassandra compared with other wide column database?

- Cassandra is decentralized. All nodes in Cassandra cluster are identical and there is no master-slave paradigm in Cassandra, which makes Cassandra highly available, fault tolerant and no-single point of failure.
- Cassandra is elastic scalability. The Cassandra has the ability seamless hire or fire new nodes and start or shut down these nodes without restart of the entire cluster, which means that you need not to restart your process. This make Cassandra easy to scale.

4 Graphic Database: Neo4j

The graphic databases use the graph structures to represent and store data. The key idea of a graphic database is graph relating the data items. The relationship between the items allows the data item in the database linked together. The relationship itself is also store in the database. Consequently, query the relationships in the graphic database is very fast. In addition, the relationships can be visualized easily in graphic databases. There are two types of graph model used in graphic databases [14].

- Labeled-property graph: This graphic model is composed of nodes, relationships, properties, and labels. The nodes and relationships can have name and properties which are represented by key-value pairs. The nodes are grouped by their labels. Each edge have the start node and end node, which represent the relationship between two nodes. The relationships can have properties, which can provide additional information about the relationship. The most popular labeled-property graph database is the Neo4j, which we will discuss in detail.
- Resource Description Framework: A resource description framework consists of nodes and arcs. In the resource description framework, the additional information is represented with a separated node. In resource description framework, if we want to add a property of a node, we need to create a node which corresponds to the property and then link the two nodes together.

4.1 Overview

As mentioned above, Neo4j is a labeled-property graphic database, which is developed by Neo4j Inc. Now, Neo4j is becoming the most popular graphic database overall, ranking 22th [15]. Neo4j is developed by Java and accessible for the software of other languages, like Cypher Query Language. Plenty of resources about Neo4j is available in [9]. Because of the distinguish storage framework of labeled-property graph, the platform can have performance of up to four million hops per second and core [9]. Neo4j is suitable for bigdata and scalable. Neo4j can store trillion of entities while keeping sensitive to compact storage. Neo4j can also be deployed into machine clusters. Neo4j also support for hot backup and extensive monitoring.

4.2 Questions to Answer

4.2.1 Why choose Neo4j?

There are several reasons to choose Neo4j for survey:

- Firstly, Neo4j is the most popular graphic database rank 22th [1]. Neo4j also has a helpful and contributing community surrounding it, which make it closer to the developers.
- Second, Neo4j offer flexible data model. Based on labeled-property graph model, Neo4j offers a flexible but powerful data model, which is adaptive to the varieties of demand of applications and industry.
- Third, Neo4j has CQL. Neo4j provides a declarative query language to represent the graph visually. The commands of CQL are readable and very easy to learn.
- Fourth, there is no joins. As the relationships (edges) are also stored directly. In Neo4j, it does not require joins to relate different data as it is easy to retrieve the adjacent node or relationship edges without joins or indexes.
- Fifth, Neo4j is scalability and reliability. We can scale the database by adding the number of reads/writes. What's more, this does not effect the query processing speed and data integrity. Neo4j also provides support for replication for data safety and reliability.

4.2.2 How Neo4j scale to handle extremely large dataset?

- Neo4j is a scalable JVM-based graphic database. As it stores data in graph not in table, it can provide high scalability for large data set. Within one machine, it can handle graph billions of nodes. In addition, it can run on multiple machines in parallel. Neo4j can handle the high complex, low structured data easily. Because the relationships have been sored in the database, Neo4j is very suitable for the queries which need plenty of join operation in the relational database.
- Neo4j is based graph model. It can search each node and edge with the same speed. The speed of traversal for Neo4j not influenced by the size of the data which constructs the graph. Consequently, Neo4j has high scalability for extremely large dataset.

4.2.3 How well does Neo4j adapt to data dynamics?

For data dynamics, Neo4j perform differently for update and query.

- In the traversal of Neo4j, the speed is not influenced by the size of the data which constructs the graph. Consequently, for the frequent changing data and data queries, Neo4j is quite suitable for this scenario. Since the speed of the query stay the constant.
- Comparing the high speed of queries, the speed of updating data is usually slow. For large dataset, it is usually to utilize to parameter mechanism and `UNWIND` to update the data in batch. In such operation, it can avoid frequently connection to the database and sessions.

4.2.4 What are the key differences between Neo4j and relational databases?

There are two main differences between Neo4j and the traditional relational databases.

- The key difference between Neo4j and the traditional relational databases is that the relationships of the graph database are stored at the individual record level. However, for the traditional relational databases, these relationship structures are gotten from the join or some other operations, which means that the relationships are not directly stored in the relational databases.

Redis	Cassandra	Neo4j
session cache	transaction logging	fraud detection & analytics solution
full page cache	storing time series data	knowledge graph
real time analysis	internet of things status and event history	recommendation engine and product recommendation system
Queues	tracking pretty much anything	social media and social network graphs

Table 1: Suitable Scenarios

- The storage structures of relational databases and Neo4j are different. The relational databases store all the records as rows while the Neo4j stores the records as nodes. The attributes of each table is corresponding to the columns of the table while the attributes of each node is stored as the properties of the nodes.

4.2.5 What are limitations of Neo4j?

- The Neo4j does not support for sophisticated index mechanisms. Consequently, it is impossible to search for ranges or something in SQL, e.g. `LIKE`.
- Writing operations are always done on the master machine as the Neo4j work in master-slaves structures. Consequently, even if the user wants to write on the a slave machine, the write command is still forwarded to the master machines internally.
- If the user wants to avoid the expensive disk seeks, all the data in each machine need to be in the memory of each server. If the dataset size is larger that the available memory capacity, the performance of the system will become slow.

4.2.6 How is Neo4j compared with other graphic database?

- Neo4j is also a native graph database, with a native format for storage and processing. Consequently, Neo4j is specially optimized for storing graph data. Not all graph databases on the market have native storage and processing, it is easy for research activities and figure out which set up is best for user's own cases.
- Neo4j is very easy to learn. Many documents and courses can be easily found. Besides, Neo4j user CQL, which is a easy-use graph query language.
- Neo4j also has complete communities, where the users can get file and help quickly, comparing with other graphic databases.

5 Application Scenarios

In this section, considering the features of Redis, Cassandra and Neo4j, we will give a summary about the scenarios where each database is suitable to use. The summary is shown in Table 1.

6 Conclusions

In this survey paper, we survey for three types of NoSQL databases and pick one popular database for each type to an brief introduction of each of them. Then, we provide how these databases deal with bigdata scenarios, comparison with the traditional relational databases and other databases in the same category according to CS5425 course requirement.

References

[1] Veronika Abramova, Jorge Bernardino, and Pedro Furtado. Experimental evaluation of nosql databases. *International Journal of Database Management Systems*, 6(3):1, 2014.

[2] Josiah L Carlson. *Redis in action*. Manning Publications Co., 2013.

[3] Cassandra®. Cassandra documents. `http://cassandra.apache.org`.

[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[5] Stefan Edlich. List of nosql databases. `http://nosql-database.org/`.

[6] Ines Fayech and Habib Ounalli. Towards a flexible database interrogation. *International Journal of Database Management Systems*, 4(3):21, 2012.

[7] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *2011 6th international conference on pervasive computing and applications*, pages 363–366. IEEE, 2011.

[8] Eben Hewitt. *Cassandra: the definitive guide*. " O'Reilly Media, Inc.", 2010.

[9] Neo4j Inc.®. Redis tutorial. `https://neo4j.com/`.

[10] javatpoint®. Redis tutorial. `https://www.javatpoint.com/rdbms-vs-cassandra`.

[11] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[12] Nishant Neeraj. *Mastering Apache Cassandra*. Packt Publishing Ltd, 2013.

[13] Redis®. Introduction to redis. `https://redis.io/topics/introduction`.

[14] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

[15] solid IT®. Db-engines ranking. `https://db-engines.com`.

[16] w3resource®. Redis tutorial. `https://www.w3resource.com/redis/`.