IIT Madras
BSc Degree

Module 59

Partha Pratim Das

Database Management Systems

Module 59: Non-Relational DBMS: NOSQL

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ac.in

Module Recap

- Evaluated RDBMS, especially with reference to performance and scalability, as a backbone for data-intensive application development
- Understood the role of system and database architecture in performance
- Understood the options for scaling databases

Module Objectives

- To understand issues in Big Data
- To understand the approach of NOSQL and CAP theorem viz-a-viz ACID
- To take tour of common types of NOSQL database

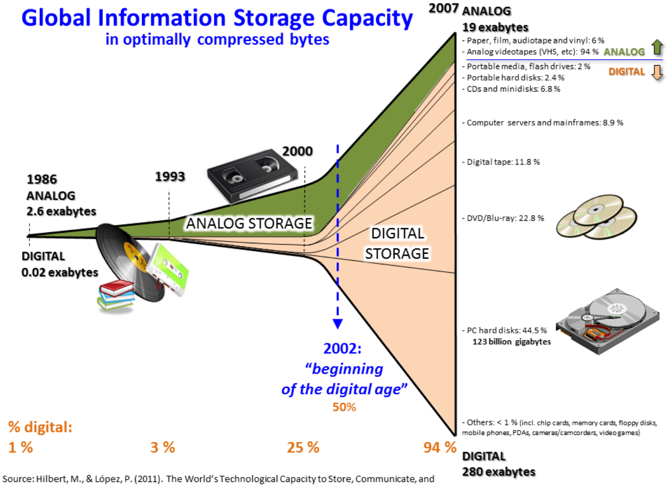
Module Outline

- What is Big Data?
- What is NOSQL?
- CAP Theorem
- Types of NOSQL Databases
- Relational vs. Non-Relational

What is Big Data?

What is BIG DATA ?

What is Big Data?



Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

What is Big Data?

- Big data is data sets that are so voluminous and complex that traditional data-processing application software are inadequate to deal with them
- Big data challenges include
  - capturing data,
  - data storage,
  - data analysis,
  - search,
  - sharing,
  - transfer,
  - visualization,
  - querying,
  - updating,
  - information privacy and
  - data source
- It refers to the use of predictive analytics, user behavior analytics, or certain other advanced data analytics methods that extract value from big data, and seldom to a particular size of data set

What is Big Data?

- **5V's (characteristics) of big data**:
  - **Volume**: The quantity of generated and stored data. The size of the data determines the value and potential insight, and whether it can be considered big data or not.
  - **Variety**: The type and nature of the data. This helps people who analyze it to effectively use the resulting insight. Big data draws from text, images, audio, video; plus it completes missing pieces through data fusion.
  - **Velocity**: In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development. Big data is often available in real-time.
  - **Variability**: Inconsistency of the data set can hamper processes to handle and manage it.
  - **Veracity**: The data quality of captured data can vary greatly, affecting the accurate analysis

What is NOSQL?

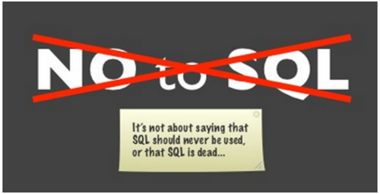
What is NOSQL?

- A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases
- NoSQL databases are increasingly used in big data and real-time web applications
- Such databases have existed since the late 1960s
  - **Network Database Model (NDBMS)** is a flexible way of representing objects and their relationships. Its distinguishing feature is that the schema, viewed as a graph in which object types are nodes and relationship types are arcs, is not restricted to being a hierarchy or lattice.
    It was introduced in 1969 and widely replaced by relational databases in the 1980s
  - **Hierarchical Database Model (HDBMS)** organizes data into a tree-like structure. The data are stored as records which are connected to one another through links. A record is a collection of fields, with each field containing only one value. The type of a record defines which fields the record contains.
    It was recognized as the first database model in the 1960s and widely replaced by relational databases in the 1980s

Source: *Introduction to NOSQL Databases, SlidePlayer*

What is NOSQL?

- Stands for **N**ot **O**nly **SQL**. Also referred as Non-Relational DBSM (**NDBMS**) and as **Multi-Model Databases**
- "NoSQL" was coined in the early 21st century, triggered by Web 2.0 companies
- The term NOSQL was introduced by Carl Strozzi in 1998 for his lightweight Strozzi NoSQL open-source relational database and re-introduced by Eric Evans when an event was organized to discuss open source distributed databases
- Eric states that "*... but the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for ...*"



Source: *Introduction to NOSQL Databases, SlidePlayer*

What is NOSQL?

Advantages	Disadvantages

• non-relational
• don't require schema
• data are replicated to multiple nodes (so, identical & fault-tolerant) and can be partitioned:
  ◦ down nodes easily replaced
  ◦ no single point of failure
• horizontal scalable
• cheap, easy to implement (open-source)
• massive write performance
• fast key-value access
	• Don't fully support relational features
	  ◦ no join, group by, order by operations (except within partitions)
	  ◦ no referential integrity constraints across partitions
	• No declarative query language (like SQL) → more programming
	• Relaxed ACID (CAP theorem) → fewer guarantees
	• No easy integration with other applications that support SQL

Source: *Introduction to NOSQL Databases, SlidePlayer*

What are NOSQL DBs?

Source: *Introduction to NOSQL Databases, SlidePlayer*

Who is using NOSQL?

Source: *Introduction to NOSQL Databases, SlidePlayer*

The Perfect Storm

- The Perfect Storm
  - Large datasets
  - Acceptance of alternatives, and
  - dynamically-typed data

  has come together in a "perfect storm"
- Not a backlash against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NOSQL offerings

Source: *Introduction to NOSQL Databases, SlidePlayer*

NOSQL: 3 Major Papers

- BigTable (Google): *Bigtable: A Distributed Storage System for Structured Data*, 2006
- DynamoDB (Amazon): *Amazon's Dynamo*, 2007
  - Ring partition and replication
  - Gossip protocol (discovery and error detection)
  - Distributed key-value data stores
  - Eventual consistency: *Eventually Consistent - Revisited*, 2008. *Choosing Consistency*, 2010
- CAP Theorem: *Brewer's CAP Theorem*, 2009

Source: *Introduction to NOSQL Databases, SlidePlayer*

CAP Theorem

CAP Theorem

CAP Theorem

Three properties of a distributed system (sharing data)

- **C***onsistency*
  - All copies have same value
- **A***vailability*
  - Reads and writes always succeed
  - Each client always can read and write
- **P***artition-tolerance*
  - System properties (consistency and/or availability) hold even when network failures prevent some machines from communicating with others
  - A system can continue to operate in the presence of a network partitions

Source: *Introduction to NOSQL Databases, SlidePlayer*

CAP Theorem (2)

- **Brewer's CAP Theorem**
  - *For any system sharing data, it is "impossible" to guarantee simultaneously all of these three properties*
  - You can have at most two of these three properties for any shared-data system
- Very large systems will *partition* at some point:
  - That leaves either **C** or **A** to choose from (traditional DBMS prefers **C** over **A** and **P**)
  - In almost all cases, you would choose **A** over **C** (except in specific applications such as order processing) these three properties

Source: *Introduction to NOSQL Databases, SlidePlayer*

CAP Theorem (3): Consistency

- **All client always have the same view of the data**
- **C***onsistency*: Two types:
    - *Strong Consistency*: ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability)
    - *Weak Consistency*: BASE (**B**asically **A**vailable **S**oft-state **E**ventual consistency)
- ACID: A DBMS is expected to support "ACID transactions," processes that are:
    - Atomicity: either the whole process is done or none is
    - Consistency: only valid data are written
    - Isolation: one operation at a time
    - Durability: once committed, it stays that way
- CAP
    - Consistency: all data on cluster has the same copies
    - Availability: cluster always accepts reads and writes
    - Partition tolerance: guaranteed properties are maintained even when network failures prevent some machines from communicating with others

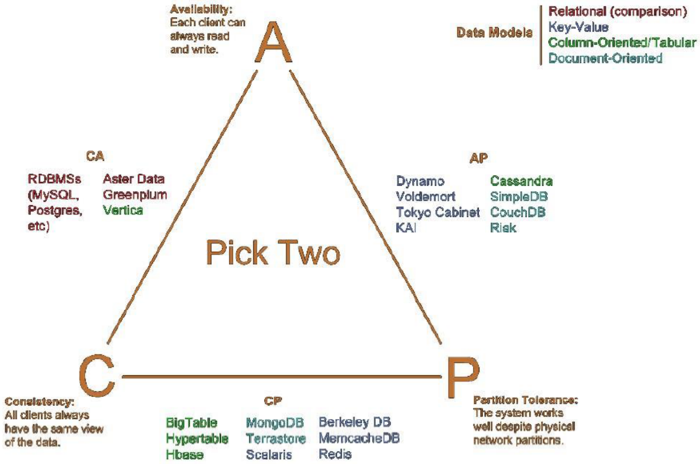
Source: *Introduction to NOSQL Databases, SlidePlayer*

CAP Theorem (4): Consistency

- A consistency model determines rules for visibility and apparent order of updates
- Example:
  - Row X is replicated on nodes M and N
  - Client A writes row X to node N
  - Some period of time t elapses
  - Client B reads row X from node M
  - **Does client B see the write from client A?**
  - Consistency is a continuum with tradeoffs
  - **For NOSQL, the answer would be: "maybe"**
  - CAP theorem states: "*strong consistency can't be achieved at the same time as availability and partition-tolerance*"
- Eventual consistency
  - When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent
- Cloud computing
  - ACID is hard to achieve, moreover, it is not always required, for example, for blogs, status updates, product listings, etc.

Source: *Introduction to NOSQL Databases, SlidePlayer*

CAP Theorem (5)



Source: *Introduction to NOSQL Databases, SlidePlayer*

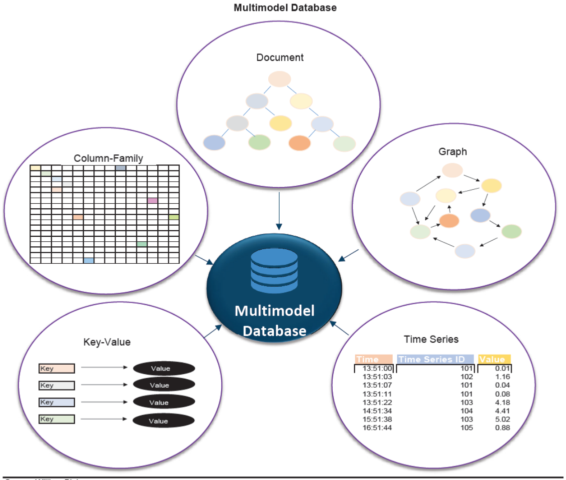
Types of NOSQL Databases

Types of NoSQL Databases

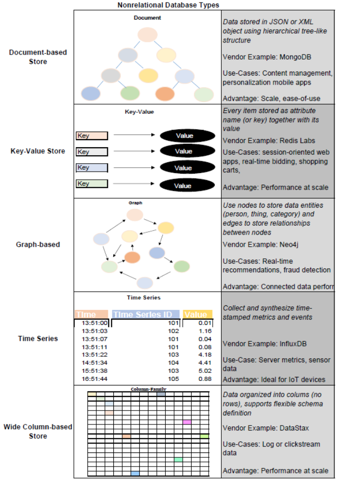
- **Key-value Stores**: **DynamoDB, Voldermort, Scalaris, Redis, MemcacheDB**
  - Work by matching keys with values, similar to a dictionary. There is no structure nor relation
- **Document Stores**: **MongoDB, Couchbase/CouchDB**
  - Work similarly to column-based ones; however, they allow much deeper nesting and complex structures to be achieved (for example, a document, within a document, within a document)
    - Documents overcome the constraints of 1 / 2 levels of key / value nesting of columnar databases
- **Column Stores**: **BigTable, Cassandra, Hbased**
  - Column-based NoSQL databases are two dimensional arrays whereby each key (that is, row / record) has one or more key / value pairs attached to it and these management systems allow very large and un-structured data to be kept and used (for example, a record with tons of information)
- **Graph Stores**: **OrientDB, Neo4J, InfoGrid**
  - These use tree-like structures (graphs) with nodes and edges connecting each other through relations
- **Time Series** (Discussed in **Module 30**): **InfluxDB, Kdb+, Prometheus, Graphite**
  - A time series database (TSDB) is a database optimized for time-stamped or time series data
  - Measurements or events that are tracked, monitored, downsampled, and aggregated over time
- *No-schema* and support for *flexible data types* are common characteristics of most NOSQL systems

Source: *Introduction to NOSQL Databases, SlidePlayer*

Multi-Model Databases



Source: William Blair



Source: William Blair

Source: *Database Software Market: The Long-Awaited Shake-up by William Blair, 2019*

NoSQL Databases: Key-value Stores

- Focus on scaling to huge amounts of data
- Designed to handle massive load
- Based on Amazon's dynamo paper
- Data model: (global) collection of Key-value pairs
- Dynamo ring partitioning and replication
- Example: (DynamoDB)
  - *items* having one or more attributes (name, value)
  - An *attribute* can be single-valued or multi-valued like set
  - Items are combined into a *table*
- Basic API access:
  - `get(key)`: extract the value given a key
  - `put(key, value)`: create or update the value given its key
  - `delete(key)`: remove the key and its associated value
  - `execute(key, operation, parameters)`: invoke an operation to the value (given its key) which is a special data structure (e.g. List, Set, Map .... etc)

Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Key-value Stores

- Pros:
  - very fast
  - very scalable (horizontally distributed to nodes based on key)
  - simple data model
  - eventual consistency
  - fault-tolerance
- Cons:
  - Can't model more complex data structure such as objects

Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Key-value Stores

Name	Producer	Data model	Querying
SimpleDB	Amazon	set of couples (key, {attribute}), where attribute is a couple (name, value)	restricted SQL; select, delete, GetAttributes, and PutAttributes operations
Redis	Salvatore Sanfilippo	set of couples (key, value), where value is simple typed value, list, ordered (according to ranking) or unordered set, hash value	primitive operations for each value type
Dynamo	Amazon	like SimpleDB	simple get operation and put in a context
Voldemort	LinkeId	like SimpleDB	similar to Dynamo

Source: *Introduction to NOSQL Databases, SlidePlayer*

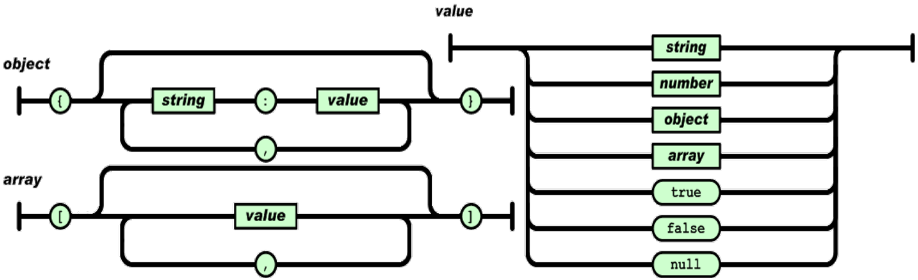
NoSQL Databases: Document Stores

- Inspired by Lotus Notes, Can model more complex objects
- Data model: Collection of documents
  - JSON (JavaScript Object Notation) is a data model, key-value pairs, which supports objects, records, structs, lists, array, maps, dates, Boolean with nesting
  - XML and other semi-structured formats
- Example: (MongoDB) document

```
{
    Name:"Jaroslav",
    Address:"Malostranske nám. 25, 118 00 Praha 1",
    Grandchildren: {
        Claire: "7", Barbara: "6", "Magda: "3",
        "Kirsten: "1", "Otis: "3", Richard: "1"
    }
    Phones: [ "123-456-7890", "234-567-8963" ]
}
```

Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Document Stores



Source: *Introduction to NOSQL Databases, SlidePlayer*

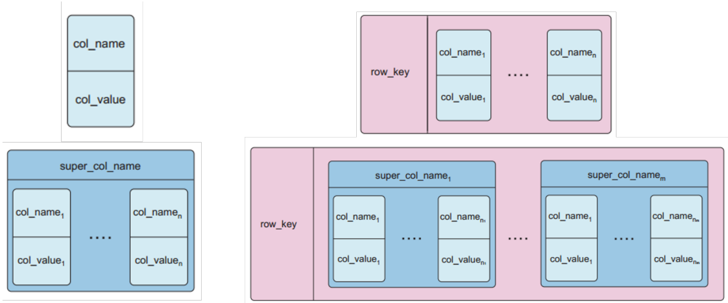
NoSQL Databases: Document Stores

Name	Producer	Data model	Querying
MongoDB	10gen	object-structured documents stored in collections; each object has a primary key called ObjectId	manipulations with objects in collections (find object or objects via simple selections and logical expressions, delete, update,)
Couchbase	Couchbase[1]	document as a list of named (structured) items (JSON document)	by key and key range, views via Javascript and MapReduce

Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Column Stores

- Based on BigTable paper
- Like column oriented RDBMS (store data in column order) but with a twist
- Tables similarly to RDBMS, but handle semi-structured
- Data model:
  - Collection of Column Families
  - Column family = (key, value) where value = set of **related** columns (standard, super)
  - indexed by *row key*, *column key* and *timestamp*



Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Column Stores

- One column family can have variable numbers of columns
- Cells within a column family are sorted "physically"
- Very sparse, most cells have null values
- Comparison: RDBMS vs column-based NOSQL
  - Query on multiple tables
    - ▷ RDBMS: must fetch data from several places on disk and glue together
    - ▷ Column-based NOSQL: only fetch column families of those columns that are required by a query (all columns in a column family are stored together on the disk, so multiple rows can be retrieved in one read operation → data locality)
- Example: (Cassandra column family–timestamps removed for simplicity)

```
UserProfile = {
    Cassandra = { emailAddress:"casandra@apache.org", age:"20" }
    TerryCho = { emailAddress:"terry.cho@apache.org", gender:"male" }
    Cath = { emailAddress:"cath@apache.org", age:"20", gender:"female",
        address:"Seoul"
    }
}
```

Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Column Stores

Name	Producer	Data model	Querying
BigTable	Google	set of couples (key, {value})	selection (by combination of row, column, and time stamp ranges)
HBase	Apache	groups of columns (a BigTable clone)	JRUBY IRB-based shell (similar to SQL)
Hypertable	Hypertable	like BigTable	HQL (Hypertext Query Language)
CASSANDRA	Apache (originally Facebook)	columns, groups of columns corresponding to a key (supercolumns)	simple selections on key, range queries, column or columns ranges
PNUTS	Yahoo	(hashed or ordered) tables, typed arrays, flexible schema	selection and projection from a single table (retrieve an arbitrary single record by primary key, range queries, complex predicates, ordering, top-k)

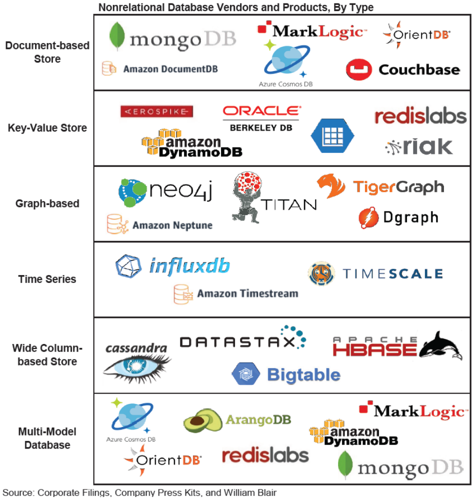
Source: *Introduction to NOSQL Databases, SlidePlayer*

NoSQL Databases: Graph Stores

- Focus on modeling the structure of data (*interconnectivity*)
- Scales to the complexity of data
- Inspired by mathematical Graph Theory (G=(E,V))
- Data model:
  - (Property Graph) nodes and edges
    - ▷ Nodes may have properties (including ID)
    - ▷ Edges may have labels or roles
  - Key-value pairs on both
- Interfaces and query languages vary
- *Single-step* vs *path expressions* vs *full recursion*
- Example:
  - Neo4j, FlockDB, Pregel, InfoGrid

Source: *Introduction to NOSQL Databases, SlidePlayer*

NOSQL Database Vendors



Source: *Database Software Market: The Long-Awaited Shake-up by William Blair, 2019*

Relational vs. Non-Relational

Relational vs. Non-Relational

	Relational	**Non-Relational**
Flexible Data Model	Works on only structured data (relational tables)	Can handle unstructured and semi structured data (xml, json), along with dynamic modification of schema. Suited for BigData
Cost, Complexity, Speed	Faster but less capable operations, Cheaper and less complex	Much more database operations supported, Highly complex internally and costlier
Performance and Scalability	Incurs issues, as data integrity needs to be maintained at all levels, in case of distributed architecture	Better Scalability and performance by exploring distributed systems using sharding and partitioning
Consistency	Enforces strong consistent systems. Less overhead for developers	Enforces eventual consistent systems. More overhead for developers
Enterprise Management and Integrations	Fits into the Enterprise IT stack, much more secure and robust	Designed to cope with agility of modern cloud based applications

Database Types and Usecases

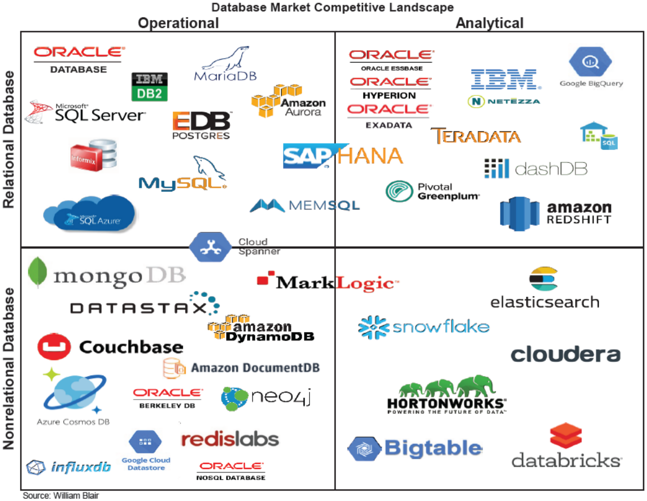
Primary Database Types and Use Cases

	Operational	Analytical
Relational Database	Key Applications: ERP, CRM, credit card processing, e-commerce, and other data of record applications	Key Applications: Data warehousing, business intelligence, and data science
	How is Data Stored: Tables (rows and columns)	How is Data Stored: Tables (rows and columns)
	Popular Products: Oracle Database, Microsoft SQL Server, IBM DB2, SAP Hana, Amazon Aurora, Azure SQL Database, EnterpriseDB (PostgreSQL), MySQL, MemSQL	Popular Products: Oracle Exadata, Oracle Hyperion, Teradata, IBM Netezza, IBM dashDB, Amazon Redshift, Microsoft SQL Data Warehouse, Google BigQuery
	Pros: Transactional guarantees/data consistency, limitless indexing, large and mature ecosystem	Pros: Consistency of information and calculations
	Cons: Rigid schema definitions, cost, mainly vertical scaling, difficult to use with unstructured/semi-structured data	Cons: IT professionals need to maintain; data response in minutes instead of milliseconds like operational databases
Nonrelational Database	Key Applications: Web, mobile, and IoT applications, social networking, user recommendations, shopping carts	Key Applications: Indexing millions of data points, predictive analytics, fraud detection
	How is Data Stored: Multiple data structures (document, graph, column, key-value, time series)	How is Data Stored: Hadoop needs no inherent data structure; data can be stored across numerous servers
	Popular Products: MongoDB, Amazon DynamoDB, Amazon DocumentDB, Azure CosmosDB, DataStax, Neo4j, Couchbase, MarkLogic, Redis	Popular Products: Cloudera, Hortonworks, MapR, MarkLogic, Snowflake, DataBricks, ElasticSearch
	Pros: Ease of use, flexibility (no need for pre-defined schema), horizontal scaling (to accommodate massive data volumes), generally low-cost (open source)	Pros: Good for batch processing, large files, and parallel scans; mainly open-source, so cost efficient
	Cons: Lack of transactional guarantees, limited querying features, relative immaturity	Cons: Slow response times; not good for fast lookups or quick updates

Source: William Blair

Source: *Database Software Market: The Long-Awaited Shake-up by William Blair, 2019*

Database Market Competitive Landscape



Source: *Database Software Market: The Long-Awaited Shake-up by William Blair, 2019*

Module Summary

- Understood the issues in Big Data
- Understood the approach of NOSQL and CAP theorem viz-a-viz ACID
- Took a tour of common types of NOSQL database
- Compared Relational with Non-relational