CHAPTER 1

Overview of Oracle NoSQL Database and Big Data
Since the invention of the transistor, the proliferation and application of computer technologies has been shaped by Moore’s Law. The growth in CPU compute capacity, high-density memory, and low-cost data storage has resulted in the invention and mass adoption of a variety of computing devices over time. These devices have become ubiquitous in our life and provide various modes of communication, computation, and intelligent sensing. As more and more of these devices are connected to the cloud, the amount of online data generated by these devices is growing tremendously. Until recently, there did not exist a very cost-effective means for businesses to store, analyze, and utilize this data to improve competitiveness and efficiency. In fact, the sheer volume and sparse nature of this data has necessitated the development of new technologies to store and analyze the data. This book covers those technologies, and focuses specifically on the role that Oracle NoSQL Database plays in that space.

Introduction to NoSQL Systems

In recent years, there has been a huge surge in the use of big data technologies to gain additional insights and benefits for business. Big data is an informal term that encompasses the analysis of a variety of data from sources such as sensors, audio and video, location information, weather data, web logs, tweets, blogs, user reviews, and SMS messages among others. This large, interactive, and rapidly growing data presents its own data management challenges. NoSQL data management refers to the broad class of data management solutions that are designed to address this space.

The idea of leveraging non-intuitive insights from big data is not new, but the work of producing these insights requires understanding and correlating interesting patterns in human behavior and aggregating the findings. Historically, such insights were largely based on the use of secret, custom-built, in-house algorithms, and systems. Only a handful of enterprises were able to do this successfully, because it was very difficult to analyze the large volume of data and the various types of data sources involved.

During the first decade of the twenty-first century, techniques and algorithms for processing large amounts of data were popularized by web enterprises such as Google and Yahoo!. Because of the sheer volume of data and the need for cost-effective solutions, such systems incorporated design choices that made them diverge significantly from traditional relational databases, leading to their characterization as NoSQL systems. Though the term suggests that these systems are the antithesis of traditional row and column relational systems, NoSQL solutions borrow many concepts from contemporary relational systems as well as earlier systems such as hierarchical and CODASYL systems. Therefore, NoSQL systems are probably better characterized as Not only SQL rather than Not SQL.
Brief Historical Perspective
It is useful to review a brief history of data management systems to understand how they have influenced modern NoSQL systems. Database systems of the early 1960s were invented to address data processing for scenarios where the amount of data was larger than the available memory on the computer. The obvious solution to this problem was to use secondary storage such as magnetic disks and tapes in order to store the additional data. Because access to secondary storage is typically a few hundred (or more) times slower than access to memory, early research in data processing was focused on addressing this performance disparity. Techniques such as efficient in-memory data structures, buffer management, sequential scanning, and batch processing and access methods (indices) for disk resident data were created in order to improve the performance of such systems.

The issue of data modeling also posed significant challenges because each application had its own view of data. The manner in which information was organized in memory as well as on disk had a huge influence on application design and processing. In the early days, data organization and modeling was largely the responsibility of the application. As a result, any changes to the methods in which data was stored or organized forced drastic changes to applications. This was hugely inefficient, and gave the impetus to decouple data storage from applications.

Early database management systems were based on the hierarchical data model. Each entity in this model has a parent record and several sub-records that are associated with the parent record organized in a hierarchy. For example, an employee entity might have a sub-record for payroll information, another sub-record for human resource (HR) information, and so on. Modeling the data in this manner improves performance because an application needs to access only the sub-records that are required, resulting in fewer disk accesses and better memory utilization. For example, a payroll application needs to reference only the payroll sub-record (and the parent record that is the “root” of the hierarchy). Application development is also simplified because applications that manage separate sub-records can be modularized and developed independently. Figure 1-1 illustrates how an employee entity might be organized in the hierarchical model.

The CODASYL model improved upon the hierarchical data model by providing indexing and links between related sub-records, resulting in further improvements in performance and simplified application development. If we use the earlier example of modeling Employee records, the CODASYL data model allows the designer to link the records of all the dependents of an employee, as shown in Figure 1-2.

Despite these improvements, the issue of record structure and schema design continued to be the dominant factor in application design. To add to the complexity, the data model was relatively inflexible; making a significant change to the organization of data often necessitated significant changes to the applications that used the data. In spite of these limitations, it is important to remember that these early systems provided excellent performance for data management problems of
the day. The overall simplicity of the system also contributed to better stability and reliability of the software. To this day, several common database applications such as airline reservation systems and banking applications are based on these architectures, a testament to their simplicity, performance, and reliability.

Ted Codd’s seminal research on relational database theory in the early 1970s, the introduction of Structured Query Language (SQL) for data manipulation, and the subsequent work on relational database management systems revolutionized the data management industry. Relational database systems support logical relationships between data items and provide a clean separation between the data model and the
application. The database system assumes the responsibility of mapping logical relationships to physical data organization. This data model independence has several important benefits, including significant acceleration of application development and maintenance, ease of physical data reorganization, and evolution and use of the relational data repository in multiple ways for managing a variety of data for multiple applications. Relational data is also referred to as structured data to highlight the “row and column” organization of the data. Since the mid-1980s, the use of relational database systems has been growing exponentially; it is fair to say that present-day enterprise data management is dominated by SQL-based systems.

In addition to the advances in data modeling and application design, the last 40 years have also seen major architectural and technological innovations such as the concept of transactions, indexing, concurrency control, and high availability. Transactions embody the intuitive notion of the all-or-nothing unit of work, typically involving multiple operations on different data entities. Various indexing techniques provide fast access to specific data quickly and efficiently; concurrency control ensures proper operation when multiple operations simultaneously manipulate shared resources. Recovery and high availability ensure that the system is resilient to a variety of failures. These technologies have been adapted and used in a variety of ways in modern NoSQL solutions.

Modern NoSQL systems were developed in the early 2000s in response to demands for processing the vast amounts of data produced by increasing Internet usage and mobile and geo-location technologies. Traditional solutions were either too expensive, not scalable, or required too much time to process data. Out of necessity, companies such as Google, Yahoo!, and others were forced to invent solutions that could address big data processing challenges. These modern NoSQL systems borrowed from earlier solutions but made significant advances in horizontal scalability and the efficient processing of diverse types of data such as text, audio, video, image, and geo-location.

**Big Data and NoSQL: Characteristics and Architectural Trade-Offs**

Big data is often characterized by the three Vs—volume, variety, and velocity. Volume obviously refers to the terabytes and petabytes of data that need to be processed, often in unstructured or semi-structured form. In a relational database system, each row in a table has the same structure (same number of columns, with a well-defined data type for each column and so on). By contrast, each individual entity (row) in an unstructured or semi-structured system can be structurally very different and therefore, contains more, less, or different information from another entity in the same repository. This variety is a fundamental aspect of
big data and can pose interesting management and processing challenges, which NoSQL systems can address. Yet another aspect of big data is the velocity at which the data is generated. For data capture scenarios, NoSQL systems need to be able to ingest data at very high throughput rates (for example, hundreds of thousands to millions of entities per second). Similarly, results often need to be delivered at very high throughput as well as very low latency (milliseconds to a few seconds per recipient).

Unlike data in relational database systems, the intrinsic value of an individual entity in a big dataset may vary widely, depending on the intended use. Take the common case of capturing web log data in files for later analysis. A sentiment analysis application aggregates information from millions or billions of individual data items in order to make conclusions about trends and patterns in the data. An individual data item in the dataset provides very little insight, but contributes to the aggregate results. Conversely, in the case of an application that manages user profile data for ecommerce, each individual data item has a much higher value because it represents a customer (or potential customer). Traditionally, every row in a relational database repository is typically a “high value” row. We will refer to this variability in value as the fourth V of big data.

In addition to this “four V” characterization of big data, there are a few implicit characteristics as well. Often, the volume of data is variable and changes in unpredictable or unexpected ways. For example, it may arrive at rates of terabytes per day during some periods and gigabytes per day during others. In order to handle this variability in volume, most NoSQL solutions provide dynamic horizontal scalability, making it possible to add more hardware to the online system to gracefully adapt to the increased demand. Traditional solutions also provide some level of scalability in response to growing demand; however, NoSQL systems can scale to significantly higher levels (10 times or more) compared to these systems.

Another characteristic of most NoSQL systems is high availability. In the vast majority of usage scenarios, big data applications must remain available and process information in spite of hardware failures, software bugs, bad data, power and/or network outages, routine maintenance, and other disruptions. Again, traditional systems provide high availability; however, the massive scalability of NoSQL systems poses unique and interesting availability challenges. Unlike traditional relational database solutions, NoSQL systems permit data loss, relaxed transaction guarantees, and data inconsistency in order to provide availability and scalability over hundreds or thousands of nodes.

**Types of Big Data Processing**

Big data processing falls into two broad categories—*batch* (or analytical) processing and *interactive* (or “real-time”) processing. Batch processing of big data is targeted to derive aggregate value (data analytics) from data by combining terabytes or petabytes
Chapter 1: Overview of Oracle NoSQL Database and Big Data

of data in interesting ways. MapReduce and Hadoop are the most well-known big data batch processing technologies available today. As a crude approximation, this is similar to data warehousing applications in the sense that data warehousing also involves aggregating vast quantities of data in order to identify trends and patterns in the data.

As the term suggests, interactive big data processing is designed to serve data very quickly with minimal overhead. The most common example of interactive big data processing is managing web user profiles. Whenever an ecommerce user connects to the web application, the user profile needs to be accessed with very low latency (in a few milliseconds); otherwise the user is likely to visit a different site. A 2010 study by Amazon.com found that every 100 millisecond increase in latency results in a 1 percent reduction in sales. Oracle NoSQL Database is a great example of a database that can handle the stringent throughput and response-time requirements of an interactive big data processing solution.

NoSQL Database vs. Relational Database

Relational database management systems (RDBMS) have been very effective in managing transactional data. The Atomicity, Consistency, Isolation, and Durability (ACID) properties of relational databases have made them a staple for enterprises looking to manage data that spans various critical business functions. Examples include Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), data warehouse, and a multitude of similar applications.

The Oracle Database has a 30-year legacy of high performance, scalability, and fault tolerance. Enterprise customers demand a high level of security, disaster recovery capabilities, and rich application development functionality. Relational databases, like the Oracle Database, provide a very comprehensive functionality to manage a multitude of data types and deployment options. These capabilities result in a rich and complex database engine.

NoSQL databases were created at the other end of this spectrum; their primary goal was to provide a very quick and dirty mechanism to retrieve information without all the varied capabilities of the RDMBS that we have highlighted in the preceding paragraph. NoSQL databases are highly distributed, run on commodity hardware, and provide minimal or no transactional support; they also have a very flexible or nonexistent schema definition requirement, and this makes them very suitable for fast storage, retrieval, and update of unstructured data. NoSQL databases have developed into a very lightweight, agile, developer-centric, API-driven database engine. NoSQL database developers are comfortable using low-level APIs to interact with the database, and don’t rely on higher-level languages such as SQL (Structured Query Language), which is a standard for an RDBMS.

It is recommended that NoSQL databases be used for high volume, rapidly evolving datasets, with low latency requirements, and where you need the complete flexibility of its APIs to develop a very specialized data store. An RDBMS has
enterprise-grade features for high availability and disaster recovery, which are essential for transactional systems. When availability requirements are more flexible and the possibility of data loss or consistency can be tolerated, NoSQL databases prove to be a cost-effective solution. Also, applications that require a very efficient mechanism to retrieve individual records without the need for operations such as complex joins will also benefit from the use of the Oracle NoSQL Database. NoSQL databases make efficient use of commodity servers and storage; they do not rely on specialized hardware and can scale to thousands of servers and hence can manage petabytes of data with very good scalability characteristics.

Both RDBMS and NoSQL databases provide significant benefits in their individual use case scenarios. It is therefore very important to choose the appropriate technology based on the need, and it is also critical to realize that the two can complement each other, to provide a very comprehensive solution for big data.

While it is critical to choose a NoSQL technology that meets your specific use case scenario, may it be key-value pair, graph, or document store (terms explained in the next section), it is also important to realize that like any other data management technology, NoSQL databases do not operate in a vacuum. Choose a NoSQL database implementation that integrates very well with data ingestion tools, RDBMS, Business Intelligence tools, and enterprise management utilities. Such an integrated NoSQL database will allow you to combine information across different database types, and data types (structured and unstructured), resulting in a big data deployment that brings tremendous value to your enterprise.

**Types of NoSQL Databases**

In a highly distributed database management system, it is important to realize that Consistency, Availability, and Partition Tolerance come at a price. The CAP Theorem states that it is impossible to provide all three capabilities simultaneously. Different NoSQL systems provide varying degrees of Consistency, Availability, and Partition Tolerance, and it is important to choose the right implementation based on your application needs.

In addition to the distributed system properties that are mentioned in the preceding paragraph, you can also classify NoSQL database implementations based on the mechanisms they use to store and retrieve data. These are important for the application developer to consider before choosing the appropriate implementation. There are four broad implementation types: key-value store, document store, columnar, and graph.

**Key-Value Stores**

The key-value implementation stores data with unique keys, and the system is opaque to the contents of the data. It is the responsibility of the client to introspect the contents. This architecture allows for a highly optimized key-based lookup.
Scalability is achieved through the *sharding* (a.k.a. partitioning) of data across nodes. To protect against data loss, key-value store implementations replicate data over nodes, and this can potentially lead to consistency issues when you have network failures and inaccessible nodes. Many systems therefore leave it up to the client to handle and resolve any consistency issues.

Key-value stores are very useful for applications such as user profile lookup, storing and retrieving online shopping carts, and catalog lookups. These applications have a unique user ID or an item ID associated with the data, and the key-value store provides a clean and efficient API to retrieve this information.

**Document Stores**

The document stores at their foundation are very similar to key-value implementation. An important distinction, however, is their capability to introspect the data that is associated with the key. This is possible because the document store understands the format of the data stored. This opens up the possibility to carry out aggregates and searches across elements of the document itself. Also, bulk update of the data is possible. Document stores work with multiple formats including XML and JSON. This allows for storage and retrieval of data without an impedance match.

The scalability, replication, and consistency characteristics of document stores are very similar to those of KV stores. Typical use cases for document stores include the storage and retrieval of catalogs, blog posts, news articles, and data analysis.

**Graph Stores**

Graph stores are different from the other methods in that they have the capability not only to capture information about objects, but can also record the relationships between these objects. Within each graph store, there are objects and relationships, which have specific *properties* attached to them. At the application level, these properties can be used to create specific subsets of relationships or objects best suited to a specific enterprise purpose. For example, the developer of a social network gaming application may wish to target a promotion of free in-game currency to those users who are friends of a gamer who ranks amongst the top 10 percentile of the highest scorers. Such data would be difficult to retrieve in other NoSQL database implementations, but the capability to traverse relationships in graph databases makes such queries very intuitive. For social networks, this analytical capability of graph stores allows for quick analysis and monetization of relationships that have been captured in their application. Graph databases can be used to analyze customer interactions, social media, and scientific application where it is crucial to traverse long relationship graphs to better understand data.

**Column Stores**

Column stores are the final type of NoSQL database that we will review. These store data in a columnar fashion; the result is a table where each row can have one or
more columns, and the number of columns in each row can vary from row to row. This provides a very flexible data model to store your data, and a clear demarcation of similar attributes, which also acts as an index to quickly retrieve data. To further demarcate by columns, you can combine similar columns to build column families. This concept of grouping helps with more complex queries as well. At the core, each column and its associated data is essentially a key-value pair. As data is organized into columns, you have better indexing (and therefore visibility) compared to other key-value stores. Also, when it comes to updates, multiple column block updates can be aggregated. Column store databases were born when Google open sourced its implementation of a Column store NoSQL database called Big Table. Apparently, the data for the well-known Google e-mail service, Gmail, is stored in the Google Big Table NoSQL Database.

Based on the discussion of the four different types of NoSQL databases, it is evident that this family of products provides a rich set of functionality for storing and retrieving data in a very cost-effective, fault-tolerant, and scalable manner.

**Big Data Use Cases**

The initial use of NoSQL technology began with the social media sites as they were looking at ways to deal with large sets of data generated by their user communities. For example, in 2010 Twitter saw data arriving at the rates of 12TB/day, and that resulted in a 4PB dataset in a year. These numbers have grown significantly as Twitter usage has expanded globally.

While the social media sites such as Twitter gave users an option to share their thoughts, ideas, and pictures, there was no easy way to make sense of such a large tsunami of information as it arrived from millions of users. HDFS is used to store such data in a distributed and fault-tolerant manner, and MapReduce technology, with its batch processing capability, is used to analyze the data. However, this wasn’t the right technology for answering real-time analytics on the data. Each tweet is stored with a unique identifier, and Twitter also saves the user ID. This key-value store could potentially take advantage of the capability of NoSQL databases. NoSQL database technologies could be used to run queries such as user searches, tweets from a specific user, and graph database capabilities could be used to find friends and followers.

Present-day enterprises have come to value the insight that social media provides into customer behavior, opinions, and market trends. Combining social media data with CRM data can provide a holistic view about the customer, something that was not possible just a few years ago. Customer data is no longer just limited to the past interactions; it can now include images, recordings, Likes (as in Facebook likes), web pages visited, preferences, loyalty programs, and an evolving
set of artifacts. This requires a system that can handle both structured and unstructured data. As more channels of communication and collaboration come and go, the data format keeps constantly changing, requiring that developers and data management systems know how to operate in a schema-less fashion. While each record in a transactional system is very critical for the operation of the business, the new customer data is high volume and sparse. This requires a distributed storage and computing environment.

Customer profile data is predominantly a read-only lookup and requires a simple key-based access. NoSQL databases, with their support of unstructured and semi-structured data, key-value store, and distributed deployments, are ideal candidates. When it comes to operational analysis, you might want to combine the customer profile data with that in your OLTP or DW systems. The tight integration between Oracle NoSQL Database and the Oracle Database makes it possible for you to join data across both of these systems. Therefore, enterprises now deploy NoSQL databases alongside RDBMS, and MapReduce technologies.

Another use case that will illustrate how the different data management and analysis technologies work together is that of online advertisers. Advertisers are always in search of a new set of eyes, and the fast growth of mobile devices has made that a key focus.

Usage patterns on mobile devices are characterized by short intermittent access, as compared to that of a desktop interface, and this puts stringent constraints on the time publishers have to make the decision about which ad to display. Typically, this is of the order of 75 milliseconds, and a medium-sized publisher might have more than 500 million ad impressions in a day. The short time intervals, the large number of events, and the huge amount of associated data that gets generated require a multifaceted data management system. This system needs to be highly responsive, be able to support high throughput, and be able to respond to varying loads and system fault conditions. There is no single technology that can fulfill these requirements.

To be effective, the publisher needs to be able to quickly analyze the user so as to decide which ad to display. A user lookup is carried out on a NoSQL database and the profile is loaded. The profile might include details on demographics, behavioral segments, recency, location, and a user rating, which might have been arrived at behind the scenes through a scoring engine.

In addition to displaying the ad, there are campaign budgets to manage, client financial transactions to track, and campaign effectiveness to analyze. NoSQL database technologies, in conjunction with MapReduce and relational databases, are used in such a deployment, as shown in Figure 1-3.
Oracle’s Approach to Big Data

The amount of data being generated is on the verge of an explosion, and according to an International Data Corporation (IDC) 2012 report, the total amount of data stored by corporations globally would surpass a zettabyte (1 zettabyte = 1 billion terabytes) by the end of 2012. Therefore, it is critical for the data companies to be prepared with an infrastructure that can store and analyze extremely large datasets, and be able to generate actionable intelligence that in turn can drive business decisions. Oracle offers a broad portfolio of products to help enterprises acquire, manage, and integrate big data with existing corporate data, and perform rich and intelligent analytics.

Implementing big data solutions with tools and techniques that are not tested or integrated is too risky and problematic. The approach to solve big data problems should follow best practice methodologies and toolsets that are proven in real-world deployments. The typical best practices for processing big data can be categorized by the flow of data in the processing stream, mainly the data acquisition, data organization, and data analysis. Oracle’s big data technology stack includes hardware and software components that can process big data during all the critical phases of its lifecycle, from acquisition to storage to organization to analysis.
Oracle engineered systems such as Oracle Big Data Appliance, Oracle Exadata, and Oracle Exalytics, along with the Oracle’s proprietary and open source software, are able to acquire, organize, and analyze all enterprise data, including structured and unstructured data, to help make informed business decisions.

Acquire

The acquire phase refers to the acquisition of incoming big data streams from a variety of sources such as social media, mobile devices, machine data, and sensor data. The data often has flexible structures, and comes in with high velocity and in large volumes. The infrastructure needed to ingest and persist these big datasets needs to provide low and predictable latencies when writing data, high throughput on scans, and very fast and quick lookups, and it needs to support dynamic schemas. Some of the popular technologies that support the requirements of storing big data are NoSQL databases, Hadoop Distributed File System (HDFS), and Hive.

NoSQL databases are designed to support high performance and dynamic schema requirements; in fact, they are considered the real-time databases of big data. They are able to provide fast throughput on writes because they use a simple data model in which the data is stored as-is with its original structure, along with a single identifying key, rather than interpreting and converting the data into a well-defined schema. The reads also become very simple: You supply a key and the database quickly returns the value by performing a key-based index lookup. The NoSQL databases are also distributed and replicated to provide high availability and reliability, and can linearly scale in performance and capacity just by adding more Storage Nodes to the cluster. With this lightweight and distributed architecture, NoSQL databases can rapidly store a large number of transactions and provide extremely fast lookups.

NoSQL databases are well suited for storing data with dynamic structures. NoSQL databases simply capture the incoming data without parsing or making sense of its structure. This provides low latencies at write time, which is a great benefit, but the complexity is shifted to the application at read time because it needs to interpret the structure of stored data, which is often a great trade-off because when the underlying data structures change, the effect is only noticed by the application querying the data. Modifying application logic to support schema evolution is considered more cost-effective than reorganizing the data, which is resource-intensive and time-consuming, especially when multi-terabytes of data are involved. Project planners already assume that change is part of an application lifecycle, but not so much for reorganization of data.

Hadoop Distributed File System (HDFS) is another option to store big data. HDFS is the storage engine behind the Apache Hadoop project, which is the software framework built to handle storage and processing of big data. Typical use of HDFS is for storing data warehouse–oriented datasets whose needs are store-once and scan-many-times, with the scans being directed at most of the stored data.
HDFS works by splitting the file into small chunks called blocks, and then storing the blocks across a cluster of HDFS servers. As with NoSQL, HDFS also provides high scalability, availability, and reliability by replicating the blocks multiple times, and providing the capability to grow the cluster by simply adding more nodes.

Apache Hive is another option for storing data warehouse–like big data. It is a SQL-based infrastructure originally built at Facebook for storing and processing data residing in HDFS. Hive simply imposes a structure on HDFS files by defining a table with columns and rows—which means it is ideal for supporting structured big datasets. HiveQL is the SQL interface into Hive in which users query data using the popular SQL language.

HDFS and Hive are both not designed for OLTP workloads and do not offer update or real-time query capabilities, for which NoSQL databases are best suited. On the flip side, HDFS and Hive are best suited for batch jobs over big datasets that need to scan large amounts of data, a capability that NoSQL databases currently lack.

**Organize**

Once the data is acquired and stored in a persistent store such as a NoSQL database or HDFS, it needs to be organized further in order to extract any meaningful information on which further analysis could be performed. You could think of data organization as a combination of knowledge discovery and data integration, in which large volumes of big data undergo multiple phases of data crunching, at the end of which the data takes a form suitable to perform meaningful business analysis. It is only after the organization phase that you begin to see a business value from the otherwise yet-to-be-valued big data.

Multiple technologies exist for organizing big data, the popular ones being Apache Hadoop MapReduce Framework, Oracle Database In-Database Analytics, R Analytics, Oracle R Enterprise, and Oracle Big Data Connectors.

The MapReduce framework is a programming model, originally developed at Google, to assist in building distributed applications that work with big data. MapReduce allows the programmer to focus on writing the business logic, rather than focusing on the management and control of the distributed tasks, such as task parallelization, inter-task communication, and data transfers, and handling restarts upon failures.

As you can imagine, MapReduce can be used to code any business logic to analyze large datasets residing in HDFS. MapReduce is a programmer’s paradise for analyzing big data, along with the help of several other Apache projects such as Mahout, an open source machine learning framework. However, MapReduce requires the end user to know programming language such as Java, which needs quite a few lines of code even for programming a simple scenario. Hive, on the other hand, translates the SQL-like statements (HiveQL) into MapReduce programs.
behind the scenes, a nice alternative to coding in Java since SQL is a language that most data analysts are already familiar with.

Open source R along with its add-on packages can also be used to perform MapReduce-like statistical functions on the HDFS cluster without using Java. R is a statistical programming language and an integrated graphical environment for performing statistical analysis. R language is a product of a community of statisticians, analysts, and programmers who are not only working on improvising and extending R, but also are able to strategically steer its development, by providing open source packages that extend the capability of R.

The results of R scripts and MapReduce programs can be loaded into the Oracle Database where further analytics can be performed (see the next section on the analyze phase). This leads to an interesting topic—integration of big data with transactional data resident in a relational database management system such as the Oracle Database. Transactional data of an enterprise has extreme value in itself, whether it is the data about enterprise sales, or customers, or even business performance. The big data residing in HDFS or NoSQL databases can be combined with the transactional data in order to achieve a complete and integrated view of business performance.

Oracle Big Data Connectors is a suite of optimized software packages to help enterprises integrate data stored in Hadoop or Oracle NoSQL Database with Oracle Database. It enables very fast data movements between these two environments using Oracle Loader for Hadoop and Oracle Direct Connector for Hadoop Distributed File System (HDFS), while Oracle Data Integrator Application Adapter for Hadoop and Oracle R Connector for Hadoop provide non-Hadoop experts with easier access to HDFS data and MapReduce functionality.

Oracle NoSQL Database also has the capability to expose the key-value store data to the Oracle Database by combining the powerful integration capabilities of the Oracle NoSQL Database with the Oracle Database external table feature. The external table feature allows users to access data (read-only) from sources that are external to the database such as flat files, HDFS, and Oracle NoSQL Database. External tables act like regular database tables for the application developer. The database creates a link that just points to the source of the data, and the data continues to reside in its original location. This feature is quite useful for data analysts who are accustomed to using SQL for analysis. Chapter 9 has further details on this feature.

Analyze
The infrastructure required for analyzing big data must be able to support deeper analytics such as data mining, predictive analytics, and statistical analysis. It should support a variety of data types and scale to extreme data volumes, while at the same time deliver fast response times. Also, supporting the ability to combine big data with traditional enterprise data is important because new insight comes not just
from analyzing new data or existing data, but by combining and analyzing together to provide new perspectives on old problems.

Oracle Database supports the **organize** and **analyze** phases of big data through the in-database analytics functionality that is embedded within the database. Some of the useful in-database analytics features of the Oracle Database are Oracle R Enterprise, Data Mining and Predictive Analytics, and in-database MapReduce. The point here is that further organization and analysis on big data can still be performed even after the data lands in Oracle Database. If you do not need further analysis, you can still leverage SQL or business intelligence tools to expose the results of these analytics to end users.

**Oracle R Enterprise** (ORE) allows the execution of R scripts on datasets residing inside the Oracle Database. The ORE engine interacts with datasets residing inside the database in a transparent fashion using standard R constructs, thus providing a rich end-user experience. ORE also enables embedded execution of R scripts, and utilizes the underlying Oracle Database parallelism to run R on a cluster of nodes.

**In-Database Data Mining** offers the capability to create complex data mining models for performing predictive analytics. Data mining models can be built by data scientists, and business analysts can leverage the results of these predictive models using standard BI tools. In this way the knowledge of building the models is abstracted from the analysis process. **In-Database MapReduce** provides the capability to write procedural logic conforming to the popular MapReduce model, and seamlessly leverage Oracle Database parallel execution. In-database MapReduce allows data scientists to create high-performance routines with complex logic, using PL/SQL, C, or Java.

Each one of the analytical components in Oracle Database is quite powerful by itself, and combining them creates even more value to the business. Once the data is fully analyzed, tools such as Oracle Business Intelligence Enterprise Edition and Oracle Endeca Information Discovery help assist the business analyst in the final decision-making process.

**Oracle Business Intelligence Enterprise Edition** (OBI EE) is a comprehensive platform that delivers full business intelligence capabilities, including BI dashboards, ad-hoc queries, notifications and alerts, enterprise and financial reporting, scorecard and strategy management, business process invocation, search and collaboration, mobile, integrated systems management, and more.

OBI EE includes the BI Server that integrates a variety of data sources into a Common Enterprise Information Model and provides a centralized view of the business model. The BI Server also comprises an advanced calculation and integration engine, and provides native database support for a variety of databases, including Oracle. Front-end components in OBI EE provide ad-hoc query and analysis, high precision reporting (BI Publisher), strategy and balanced scorecards, dashboards, and linkage to an action framework for automated detection and
business processes. Additional integration is also provided to Microsoft Office, mobile devices, and other Oracle middleware products such as WebCenter.

*Oracle Endeca Information Discovery* is a platform designed to provide rapid and intuitive exploration and analysis of both structured and unstructured data sources. Oracle Endeca enables enterprises to extend the analytical capabilities to unstructured data, such as social media, websites, e-mail, and other big data. Endeca indexes all types of incoming data so the search and the discovery process can be fast, thereby saving time and cost, and leading to better business decisions. The information can also be enriched further by integrating with other analytical capabilities such as sentiment and lexical analysis, and presented in a single user interface that can be utilized to discover new insights.

**Oracle Engineered Systems for Big Data**

Over the last few years, Oracle has been focused on purpose-built systems that are engineered to have hardware and software work together, and are designed to deliver extreme performance and high availability, while at the same time making them easy to install, configure, and maintain. The Oracle engineered systems that assist with big data processing through its various phases are the Oracle Big Data Appliance, Oracle Exadata Database Machine, and Oracle Exalytics In-Memory Machine. Figure 1-4 shows the best practice architecture of processing big data using Oracle engineered systems. As the figure depicts, each appliance plays a special role in the overall processing of big data by participating in the acquisition, organization, and analysis phases.

![Oracle engineered systems supporting acquire, organize, analyze, and decide phases of big data](image)

**FIGURE 1-4.** Oracle engineered systems supporting acquire, organize, analyze, and decide phases of big data
Oracle Big Data Appliance

The Oracle Big Data Appliance is an engineered system built with optimized hardware and a comprehensive set of software designed to provide big data solutions in a complete, easy-to-deploy offering for acquiring, organizing, and analyzing big data. Oracle Big Data Appliance delivers an affordable, scalable, and fully optimized big data infrastructure in-a-box, as compared to building a custom system from scratch, which could be time-consuming, inefficient, and prone to failures. Oracle Big Data Appliance, along with Oracle Exadata Database Machine and Oracle Exalytics In-Memory Machine, creates a complete set of technologies for leveraging and integrating big data, and helps enterprises quickly and efficiently turn information into insight.

The Oracle Big Data Appliance provides the following benefits:

- Rapid provisioning of large and highly available big data clusters that can linearly scale and process massive amounts of data
- Cost control benefits of deploying a pre-integrated, engineered system that can be installed and managed easily
- High performance by engineering state-of-the-art hardware and pre-optimized software to assist with acquiring, organizing, and analyzing big data

The Oracle Big Data Appliance comes in multiple configurations of different-sized racks: the full rack, two-thirds rack, and one-third rack. The full-rack configuration comprises 18 Sun servers and provides a total raw storage capacity of 648TB. Every server in the rack has 2 CPUs, each with 8 cores for a total of 288 cores, and 64GB memory that can be expanded to 512GB, for a total of 1152GB expandable to over 9TB of total memory for all 18 servers. The two-thirds rack and one-third rack configurations have the hardware specs that are basically two-thirds and one-third of the respective full-rack configuration. These racks can be easily cabled together using the high-speed InfiniBand network in order to provide rapid scalability and incremental growth, thereby enabling the cluster to handle extreme data volumes and storage capacity.

As shown in Figure 1-5, the software preinstalled on the Oracle Big Data Appliance includes a combination of open source software and specialized software developed by Oracle to address enterprise big data needs. The Oracle Big Data Appliance integrated software includes:

- Cloudera’s distribution including Apache Hadoop (CDH)
- Cloudera Manager
Chapter 1: Overview of Oracle NoSQL Database and Big Data

- Oracle NoSQL Database Community Edition (CE)
- Oracle Big Data Connectors
- Oracle R Distribution (Oracle’s redistribution of Open Source R)

Oracle NoSQL Database Community Edition (CE) comes preinstalled on the Oracle Big Data Appliance by default, and configured upon the customer’s request at install time. You have the capability to run Oracle NoSQL Database on all the 18 nodes in the cluster, with each node having a dedicated space of 3TB or 6TB (one disk or two disks, other custom configurations are also possible). Oracle NoSQL Database is rack aware and its block placement algorithms minimize data loss when multiple racks are interconnected by placing mirrored blocks on different racks to enhance availability. The customer can purchase the Enterprise Edition (EE) license of Oracle NoSQL Database and get enterprise-level features (see Chapter 2 for more details).

Cloudera’s Distribution including Apache Hadoop (CDH) consists of open source Apache Hadoop and a comprehensive set of open source software components needed to use Hadoop, with Cloudera’s branding and support. Cloudera Manager is a proprietary product from Cloudera that provides an end-to-end management application that provides monitoring and administration capabilities of CDH clusters. It also incorporates a full range of reporting and diagnostic tools to help optimize cluster performance and utilization.

FIGURE 1-5. Oracle Big Data Appliance software overview
Oracle Exadata Database Machine
The Oracle Exadata Database Machine is an engineered system built to support all types of database workloads, ranging from data warehouse applications that scan large amounts of data, to OLTP applications supporting highly concurrent and real-time transactions. It has an award-winning combination of smart software that runs in the storage layers called Exadata Storage Server Software, the intelligent Oracle Database 11g software, and the latest industry hardware components from Oracle, all combined to deliver extreme performance in a highly available, reliable, and highly secure environment out-of-the-box.

The Database Machine has large amounts of memory and PCIe-based Flash storage, which allows caching and storage of frequently accessed data into entities that are hundreds of times faster than the hard disks, which helps boost OLTP-like workload performance. The smart features of the Exadata Storage Server Software offloads processing to run near the disks where the data resides, thereby eliminating a lot of unnecessary data movement between the database CPUs and disks, a feature that can provide ten- or twenty-fold speed-up for data warehousing workloads.

The Database Machine is also well-suited for consolidating multiple databases onto a single grid by utilizing the resource management, clustering, workload management, and the pluggable database features of the Oracle Database. Also, the award-winning Exadata Hybrid Columnar Compression feature allows you to achieve 10- to 50-times compression of data on disk, thereby offering cost savings and performance improvements because you store and scan less data.

The Oracle Exadata Database Machine has the capability to perform the organize and analyze stages of big data processing. The In-Database Analytics offers powerful features for knowledge discovery and data mining, which helps extract hidden intelligence and allows the organization of data in a manner suitable for making business decisions. The Oracle business intelligence tools, such as Oracle BI EE and Oracle Endeca, rely on the data residing in a relational system, for which the Exadata Database Machine is the ideal platform of choice. Connections between Oracle Big Data Appliance, Oracle Exadata, and Oracle Exalytics are via InfiniBand, enabling high-speed data transfer for batch or query workloads.

Oracle Exalytics In-Memory Machine
In the world of rapidly evolving economy and business dynamics, it has become even more important for organizations to perform real-time, visual analysis, and enable new types of analytic applications in order to assist with speed-of-thought decision process, in order to help them stand out from the rest. Static reports and dashboards have become passé; enterprises are now utilizing tools and techniques such as business modeling, planning, forecasting, and predictive analytics, and using rich and interactive visualizations to assist with actionable intelligence and real-time decisions.
Chapter 1: Overview of Oracle NoSQL Database and Big Data

Oracle Exalytics In-Memory Machine is an engineered system built to deliver high-performance business intelligence (BI) and enterprise planning applications. The hardware consists of a single server that is optimally configured for business intelligence workloads and includes powerful compute capacity and abundant memory to assist with in-memory analytics. The InfiniBand network connectivity provides an extremely fast option to connect Exalytics to other Exalytics or Oracle engineered systems such as Exadata. For example, this option can augment the business intelligence capabilities of Exalytics with powerful embedded in-database analytics capabilities of Exadata.

The software included in the Oracle Exalytics In-Memory Machine is the optimized Oracle BI Foundation Suite (Oracle BI Foundation) and Oracle TimesTen In-Memory Database. Business Intelligence Foundation takes advantage of the Exalytics hardware and system configuration to deliver rich and actionable intelligence. Exalytics also provides better query responsiveness and higher user scalability compared to standalone installation of Oracle BI Foundation. The TimesTen In-Memory Database for Exalytics is an optimized in-memory database that offers some exclusive features especially enabled for Exalytics, such as columnar compression to reduce the footprint for in-memory data.

Summary

NoSQL databases provide a simple and lightweight mechanism for storing new and diverse sets of digital data streams, which oftentimes would not be appropriate to store in a traditional RDBMS. NoSQL databases are optimized to handle quick reads and writes of large datasets by allowing the application to define loose durability and consistency models in order to favor read and write performance, which is a key factor for a big data application with real-time needs.

Oracle NoSQL Database is a distributed key-value database designed to provide highly reliable, scalable, and available data storage across a configurable set of systems. Oracle NoSQL Database plays a key role in the overall portfolio of Oracle’s big data offerings, to assist in analyzing enterprise big data. The rest of the chapters cover Oracle NoSQL Database in much greater depth.