# Table of Contents

Table of Contents .................................................................................................................. 2

Abstract ................................................................................................................................. 3
Introduction .............................................................................................................................. 3

*Oracle and Today’s Online Applications* ............................................................................. 3
  Architectural Limitations ........................................................................................................ 3
  Data Model Limitations .......................................................................................................... 3
  Scalability and Performance Limitations .............................................................................. 4
  Pricing Limitations ................................................................................................................. 4

*When NoSQL Over Oracle?* ............................................................................................... 4
  Business Considerations ........................................................................................................ 4
  Technical Considerations ..................................................................................................... 5

*Comparing Oracle and Cassandra* ...................................................................................... 5
  Overview of Cassandra .......................................................................................................... 5
  Overview of DataStax Enterprise ........................................................................................... 6
  Detailed Comparison of Oracle and Cassandra/DataStax Enterprise ................................. 6

*Oracle Replacement & Co-Existence* .................................................................................. 7
  How to Move from Oracle to Cassandra/DataStax Enterprise ............................................ 7
  Customer Examples Who Have Moved From Oracle ............................................................ 8

Conclusion .............................................................................................................................. 8

About DataStax ....................................................................................................................... 8

*Appendix A – Detailed Comparison of DataStax Enterprise / Cassandra & Oracle* .......... 9

*Appendix B – FAQ on Switching from Oracle to DataStax Enterprise/Cassandra* .......... 10
Abstract

For decades, Oracle has been the top relational database (RDBMS) used to support key business systems. However, today's always-online world has brought with it a substantive change in how IT professionals must manage data and use it to achieve maximum business impact. This paper looks at why NoSQL technology like Apache Cassandra™ / DataStax Enterprise is quickly becoming the first and best database choice over Oracle for online applications, providing guidelines for when a legacy RDBMS like Oracle should be used and when NoSQL is required.

Introduction

Literally every IT professional is acquainted with the Oracle RDBMS. According to recent numbers by Gartner Group, Oracle owns more than 48% of the database market, which puts it squarely in the lead where traditional database technology is concerned.

While Oracle remains a solid RDBMS that performs well for the use cases for which it was designed (e.g. ERP and accounting applications), even its strongest supporters admit that it is not architectured to tackle the new wave of big data, online applications developed today. The march from mainframes to client server to Web/mobile/social has resulted in the generation of countless applications and exploding data volumes that outstrip a legacy database like Oracle.

Oracle and Today’s Online Applications

Modern online applications differ from those of several decades ago in many key respects, especially where data is concerned. The following depicts the most important distinctions in regard to data management:

These and other shifts in application requirements tax an RDBMS like Oracle in ways that oftentimes prohibit its use where modern, big/fast data systems are concerned. Let's look in more detail at a few of the reasons why this is the case.

Architectural Limitations

One reason modern businesses switch from Oracle to NoSQL is because the underlying RDBMS does not often support the architectural requirements of modern online applications. Today's online systems necessitate a divide-and-conquer configuration for processing both big and fast data, which comes in from millions of user interfaces from many different locations. The scale-up, master-slave, non-distributed architecture of Oracle was never designed for such use cases and therefore falls short of what modern online applications need.

This is true no matter what type of Oracle software extension or platform is used; Oracle RAC, Exadata, Dataguard, Goldengate, and the like all miss the mark from an architecture perspective when it comes to tackling the data velocity, volume, performance, uptime, and distribution requirements of many big/fast data applications.
Data Model Limitations
A big reason why many businesses are moving to NoSQL-based solutions is because the legacy RDBMS data model is not flexible enough to handle modern online application use cases that contain a mixture of structured, semi-structured, and unstructured data. While Oracle has good datatype support for traditional RDBMS situations that deal with structured data, it lacks the dynamic data model necessary to tackle high-velocity data coming in from machine-generated systems or time series applications, as well as cases needing to manage social media data.

Scalability and Performance Limitations
Oracle’s scale-up, master-slave design limits both its scalability and performance for servicing the online elasticity and performance SLA needs of many online applications. The failure of Oracle to add capacity online in an elastic, scale-out vs. scale-up manner to service increasing user workloads, keep performance high, and easily consume fast incoming data from countless geographical locations is widely recognized.

Pricing Limitations
Sometimes pushing a square peg into a round hole can be done, but usually it’s done only at great cost. To overcome the architectural, data model, and scale/performance limitations of Oracle, some companies attempt to spend their way to success with a technology not suited to their application requirements only to realize that such a strategy typically fails in the long (and sometimes short) term.

Oracle’s expensive core software licensing and maintenance costs are well known. What oftentimes are overlooked, though, are the costs associated with trying to extend Oracle with its software add-ons and push its square peg into today’s online application round hole. Again, the result is not only disappointing from a requirements-met perspective, but also from a dollars-spent standpoint.

Viewing the costs for Oracle and its various add-ons are eye-opening indeed, especially when one realizes that NoSQL technology, oftentimes a perfect fit for today’s online applications, normally costs 80-90+% less than just Oracle’s enterprise edition alone.

<table>
<thead>
<tr>
<th>Oracle Product</th>
<th>List Price/Processor¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Enterprise Edition</td>
<td>$47,500</td>
</tr>
<tr>
<td>Oracle Partitioning (assists with large data volume management)</td>
<td>$11,500</td>
</tr>
<tr>
<td>Oracle RAC add-on to Enterprise Edition</td>
<td>$23,000</td>
</tr>
<tr>
<td>Oracle Dataguard (active, assists with disaster recovery/high availability)</td>
<td>$10,000</td>
</tr>
<tr>
<td>Oracle GoldenGate (transactional replication/data integration)</td>
<td>$17,500</td>
</tr>
</tbody>
</table>

Such costs, coupled with Oracle’s lack of meeting online application needs, leads analysts such as Peter Goldmacher with Cowen and Company to say:

“We believe that we are in a major technology transition as IT buyers are increasingly turning to newer apps and data mgmt technologies that offer more robust and flexible functionality at dramatically lower prices.”

When NoSQL Over Oracle?
With an understanding of how Oracle falls short in satisfying the requirements of today’s online applications now completed, let’s turn attention to questions that both business and tech leaders can ask to help determine whether NoSQL technologies like Apache Cassandra and DataStax Enterprise are better suited than Oracle for a particular system.

Business Considerations
The questions business leaders should ask when it comes to deciding whether a NoSQL database like Cassandra is suited for a particular application/use case over a traditional database like Oracle are the following:

- Do you need to keep the application always online and serving customers?
- Do you need to serve customers with multiple interfaces ad in multiple locations?

¹ Oracle Technology Global Price List, January 2013.
• Do you need to consume and deliver lots of data very quickly?
• Do you need to easily add database capacity to handle increasing customer demand?
• Do you need to manage many different types of data (e.g. social media, etc.)?
• Do you need to easily run analysis on your line of business data?
• Do you need to easily search your line of business data?
• Do you need to receive strong payback for IT investments?

If the application being considered delivers multiple affirmatives, then NoSQL should be considered for all/part of the solution.

Technical Considerations
The technical considerations for determining whether Oracle or NoSQL should be used for an application reflect the business questions:
• Do you need continuous availability with redundancy in both data and function across one or more locations vs. simple failover for the database?
• Do you need a database that runs over multiple data centers / cloud availability zones?
• Do you need to handle high velocity data coming in via sensors, mobile devices, and the like, and have extreme right speed and low latency query speed?
• Do you need to go beyond single machine limits for scale-up and instead go to a scale-out architecture to support the easy addition of more processing power and storage capacity?
• Do you need to manage data that goes beyond a rigid RDBMS table/row data structure and instead includes a combination of structured, semi-structured, and unstructured data?
• Do you need to run different workloads (e.g. online, analytics, search) on the same data without needing to manually ETL the data to separate systems/machines?
• Do you need to manage a widely distributed system with minimal staff?

Again, if you answered yes to several of the questions above, NoSQL is your best choice.

Comparing Oracle and Cassandra
Once a general understanding has been reached that NoSQL technology should be considered over Oracle for a particular application, a more detailed feature/benefit comparison is often needed. The following sections provide an overview of both Cassandra and DataStax Enterprise along with a more detailed feature comparison with Oracle.

Overview of Cassandra
Apache Cassandra is an open source massively scalable NoSQL database the offers continuous availability, linear scale performance, and easy data distribution across one or more data centers.

Benefits of Cassandra include:
• Massively scalable architecture – a masterless design where all nodes are the same.
• Linear scale performance – online node additions produce predictable increases in performance.
• Continuous availability – redundancy of both data and function mean no single point of failure.
• Transparent fault detection and recovery – easy failed node recovery.
• Flexible, dynamic schema data modeling – easily supports structured, semi-structured, and unstructured data.
• Guaranteed data safety – commit log design ensures no data loss.
• Active everywhere design – all nodes may be written to and read from.
• Tunable data consistency – support for strong or eventual data consistency.
• Multi-data center replication – cross data center and multi-cloud availability zone support for writes/reads built in.
• Data compression – data compressed up to 80% without performance overhead.
• CQL (Cassandra Query Language) – an SQL – like language that makes moving from an RDBMS very easy.

From a performance standpoint, Cassandra stands out among other NoSQL databases. Independent tests conducted by academic institutions, technical service providers, and modern enterprises confirm that Cassandra excels over its rivals in many workloads and use cases.

For example, after running a VLDB benchmark, experts at the University of Toronto concluded: “In terms of scalability, there is a clear winner throughout our experiments. Cassandra achieves the highest throughput for the maximum number of nodes in all experiments with a linear increasing throughput.”

---

A YCSB benchmark by End Point Solutions found Cassandra beating rival NoSQL solutions MongoDB and HBase by a wide margin in literally every test:

![Benchmark of Cassandra, HBase, and MongoDB](image)

**Figure 3 – Benchmark of Cassandra, HBase, and MongoDB.**

Overview of DataStax Enterprise

DataStax is the leading provider of enterprise NoSQL database software products and services based on Apache Cassandra. DataStax drives the open source development of Cassandra by employing the Apache chair of Cassandra as well as most of the project’s committers.

DataStax also provides DataStax Enterprise for businesses wishing to deploy enterprise-class NoSQL technology in production environments. DataStax Enterprise is comprised of three components:

- **The DataStax Enterprise Server** – built on Apache Cassandra, the server manages online/real-time data with Cassandra, and provides for built-in analytics and enterprise search on Cassandra data as well. It also offers the most comprehensive set of security features of any NoSQL provider.

- **OpsCenter** – a visual, browser-based solution for managing and monitoring Cassandra and the DataStax Enterprise server.

- **Production Support** – full 24x7x365 support from the big data experts at DataStax.

DataStax Enterprise provides a number of benefits over open source Cassandra, including:

- A version of Cassandra certified for production environments – one that has received extensive QA testing, benchmarking, third-party software validation, and defect resolution.
- Built-in analytics functionality for Cassandra data via integration with a number of Hadoop components (e.g. MapReduce, Hive, Pig, Mahout, etc.)
- Enterprise search capability on Cassandra data via Solr integration.
- Enterprise security including external/internal authentication and object permission management, transparent data encryption, client-to-node and node-to-node encryption, and data auditing.
- Visual cluster management for all administration tasks including backup/restore operations, performance monitoring, alerting, and more.
- Built-in support for migrating data from RDBMS’s straight into Cassandra.
- Certified software updates and formal software end-of-life policies.
- Hot fixes.
- Around-the-clock expert support.

DataStax Enterprise excels in handling many different modern online application use cases including:

- Time-series data management (e.g. financial, sensor data, web click stream, etc.)
- Online web retail
- Web buyer behavior and personalization management
- Recommendation engines
- Social media input and analysis
- Online gaming
- Fraud detection and analysis
- Risk analysis and management
- Supply chain analytics
- Web product searches
- Write intensive transactional systems

Conference, 2013.  
Appendix A in this document contains a detailed feature and function comparison between Oracle and those of Cassandra/DataStax Enterprise.

Oracle Replacement & Co-Existence

Once online businesses realize they need to move from Oracle, they experience two basic implementation scenarios:

1. New online applications – these involve completely new applications whose requirements dictate the use of NoSQL over a legacy RDBMS like Oracle. They may or may not involve a polyglot persistence approach (i.e. one that uses a combination of relational and NoSQL technology).

2. Existing online applications – these are legacy applications that are morphing into big/fast data systems and require NoSQL in either a singular or polyglot persistence manner.

The first situation normally doesn’t require a rip/replace approach of both existing Oracle code and data to NoSQL; however the second oftentimes, if not always, does. In that case, how does an IT shop practically go about making the move from Oracle to something like Cassandra and DataStax Enterprise?

How to Move from Oracle to Cassandra/DataStax Enterprise

The two basic steps in moving from Oracle to Cassandra are, first, understanding the data modeling differences between an RDBMS and a NoSQL engine like Cassandra, and second, porting the legacy data from Oracle over to Cassandra.

Understand the Data Model Differences

The first step in migrating from Oracle to Cassandra is to understand that data modeling is handled differently in NoSQL solutions vs. RDBMSs. In traditional databases such as Oracle, data is modeled in a standard “third normal form” design without the need to know what questions will be asked of the data.

By contrast, in NoSQL, the questions asked of the data are what drive the data model design and the data is highly denormalized.

To help in making the mental design shift needed from relational to NoSQL, DataStax has put together a number of visual data modeling presentations with examples that can serve as how-to guides. They can be found on PlanetCassandra.org as well as http://www.datastax.com/resources/data-modeling.

Once an understanding of the data model differences is reached and new database objects have been created in Cassandra, the next step is to move the existing data from Oracle to Cassandra. Depending on the situation, there are three possible routes a developer or administrator can take.

Using Cassandra’s High-Speed Loader

Data from Oracle can be extracted into flat files that are delimited in some way (e.g. comma, tab, etc.) and then loaded into Cassandra tables via the CQL COPY command. The user specifies that Cassandra table and the Oracle flat file name and location, and the data is then quickly loaded into the target Cassandra object.

Using Sqoop

DataStax Enterprise supports Sqoop, which is a utility designed to transfer data directly from an RDBMS like Oracle into Cassandra. The only prerequisite for use is that the JDBC driver for Oracle must be downloaded from the Oracle website and placed in a directory where Sqoop has access to it (the /sqoop subdirectory of the main DataStax Enterprise installation is recommended).

The DataStax Enterprise installation package includes a sample/demo of how to move MySQL schema and data into Cassandra, which can easily be adapted for Oracle. Each Oracle table is mapped to a Cassandra table. The migration is done via a command line utility that accepts a number of different parameters.

Using Pentaho Kettle

Another way to migrate Oracle tables and data to Cassandra is by using a number of ETL tools on the market such as Pentaho’s Data Integration product, also known as Kettle. Pentaho makes two editions of their ETL tool available: a free community edition and a paid enterprise edition. For core ETL tasks such as moving Oracle schema and data to Cassandra, the community edition should provide everything that is needed.
Pentaho’s Kettle product provides an easy-to-use graphical user interface (GUI) that allows a developer to visually design their Oracle migration tasks. Unlike the Sqoop utility, which just does extract-load, Pentaho’s product allows a developer to create simple to sophisticated transformation routines to customize how an Oracle schema and data are moved to Cassandra. In addition, the data movement engine of Kettle is quite efficient, so medium to semi-large data volumes can be moved in a high-performance manner.

More information about Kettle and free downloads can be found at: [http://kettle.pentaho.com/](http://kettle.pentaho.com/).

**Customer Examples Who Have Moved From Oracle**

DataStax has numerous customers who have either moved existing applications from Oracle to DataStax Enterprise or who use DataStax Enterprise in conjunction with Oracle. Some of these customers include the following.

**Netflix**

Netflix stores 95% of their data on Cassandra in the cloud (AWS), having moved much of it from Oracle. Please see the Netflix case study for more information.

**eBay**

eBay uses DataStax Enterprise for many use cases including fraud detection, time series data management, messaging and more, with a number of systems being moved from Oracle. eBay stores more than 250TB of data in DataStax Enterprise across three data centers, and sees 9 billion writes and 5 billion reads per day flow through their DataStax Enterprise clusters. Please see the eBay case study for more information.

**OpenWave**

OpenWave moved its messaging store architecture from Oracle to DataStax Enterprise to save money and so its platform could scale and perform better. Please see the OpenWave case study for more information.

**Conclusion**

There is no argument that Oracle is a strong RDBMS that well serves the use cases for which it was originally designed. But for IT professionals who are either planning new big/fast data applications or have existing Oracle systems that have begun to break down under big data workloads, a move to DataStax Enterprise and Cassandra makes both business and technical sense.

Switching to a modern, big data platform like DataStax Enterprise will future-proof any application, and provide confidence that the system will scale and perform well now and into a demanding future. For more information about DataStax Enterprise and Cassandra, visit [www.datastax.com](http://www.datastax.com).

For downloads of DataStax Enterprise – which may be freely used for development evaluation purposes – visit [http://www.datastax.com/download/enterprise](http://www.datastax.com/download/enterprise).

**About DataStax**

DataStax delivers Apache Cassandra™ in a database platform purpose built for the performance and availability demands of web, mobile and IOT applications, giving enterprises a secure always-on database that remains operationally simple when scaled in a single datacenter or across multiple data centers and clouds.

DataStax has more than 500 customers in 38 countries including leaders such as Netflix, Rackspace, Pearson Education, and Constant Contact, and spans verticals including web, financial services, telecommunications, logistics, and government. Based in San Mateo, Calif., DataStax is backed by industry-leading investors including Lightspeed Venture Partners, Meritech Capital, and Crosslink Capital.
Appendix A – Detailed Comparison of DataStax Enterprise / Cassandra & Oracle

This appendix provides a detailed feature comparison between Oracle and Cassandra / DataStax Enterprise.

<table>
<thead>
<tr>
<th>Feature/Function</th>
<th>DataStax Enterprise/Cassandra</th>
<th>Oracle RDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core architecture</td>
<td>&quot;Masterless&quot;; peer-to-peer with all nodes being the same</td>
<td>Traditional standalone</td>
</tr>
<tr>
<td>High availability</td>
<td>Continuous availability with built in redundancy and hardware rack awareness in both single and multiple data centers</td>
<td>General replication; Oracle Dataguard (for failover) and Oracle RAC (single point of failure with storage) both of which are expensive add-ons. GoldenGate also offered for certain use cases.</td>
</tr>
<tr>
<td>Scalability model</td>
<td>Linear performance gains via node additions</td>
<td>Scale up via adding CPU's, RAM, or Oracle RAC or Exadata</td>
</tr>
<tr>
<td>Replication model</td>
<td>Peer-to-peer; number of copies configurable across cluster and each datacenter</td>
<td>Peer-to-peer; number of copies configurable across cluster and each datacenter</td>
</tr>
<tr>
<td>Multi-data center/geography/cloud capabilities</td>
<td>Multi-directional, 1-many data center support built in, with true read/write anywhere capability</td>
<td>Nothing specific for multi-data center</td>
</tr>
<tr>
<td>Data partitioning/sharding model</td>
<td>Automatic; done via primary key; random or ordered</td>
<td>Table partitioning option to enterprise edition; manual server sharding</td>
</tr>
<tr>
<td>Data volume support</td>
<td>TB-PB capable</td>
<td>TB capable; PB with Exadata</td>
</tr>
<tr>
<td>Analytic support</td>
<td>Analytics on Cassandra data via Hadoop integration(MapReduce, Hive, Pig, Mahout)</td>
<td>Analytic functions in Oracle RDBMS via SQL MapReduce. Hadoop support done in NoSQL appliance</td>
</tr>
<tr>
<td>Enterprise search support</td>
<td>Built into DataStax Enterprise via Solr integration</td>
<td>Handled via Oracle search (cost add-on)</td>
</tr>
<tr>
<td>Mixed workload support</td>
<td>All handled in one cluster with built-in workload isolation; no workload competes for resources with another</td>
<td>Handled via Oracle Exadata</td>
</tr>
<tr>
<td>Data model</td>
<td>Google Bigtable like; a wide column store</td>
<td>Relational/tabular</td>
</tr>
<tr>
<td>Flexibility of data model</td>
<td>Flexible. Designed for structured, semi-structured, and unstructured data</td>
<td>Rigid; primarily structured data</td>
</tr>
<tr>
<td>Data consistency model</td>
<td>Tunable consistency (CAP theorem consistency per operation (e.g. per insert, delete, etc.) across cluster.)</td>
<td>Traditional ACID</td>
</tr>
<tr>
<td>Transaction support</td>
<td>Provides full Atomic, Isolated, and Durable (AID) transactions including batch transactions and &quot;lightweight&quot; transactions with Cassandra 2.0 and higher.</td>
<td>Traditional ACID</td>
</tr>
<tr>
<td>Security</td>
<td>Support for all key security needs: Login ID/passwords, external security support; object permission management; transparent data encryption; client to node, node to node encryption; data auditing</td>
<td>Full security support</td>
</tr>
</tbody>
</table>
### Appendix B – FAQ on Switching from Oracle to DataStax Enterprise/Cassandra

This appendix supplies answers to frequently asked questions about migrating from Oracle to DataStax Enterprise/Cassandra.

**Do I lose transaction support when moving from Oracle to Cassandra?**

Oracle supplies ACID transaction support, whereas Cassandra provides AID transaction support. The “C” or consistency part of transaction support does not apply to Cassandra, as there is no concept of referential integrity or foreign keys in a NoSQL database. There is also no concept of commit/rollback in Cassandra. Batch operations are supported in Cassandra via the `BATCH` option in CQL and “lightweight transaction” support is available in Cassandra 2.0 and higher.

**What type of data consistency does DataStax Enterprise/Cassandra support?**

DataStax Enterprise and Cassandra support “tunable data consistency”. This type of consistency is the kind represented by the “C” in the CAP theorem, which concerns distributed systems.

Cassandra extends the concept of “eventual consistency” in NoSQL databases by offering tunable consistency. For any given read or write operation, the client application decides how consistent the requested data should be.

Consistency levels in Cassandra can be set on any read or write query. This allows application developers to tune consistency on a per-query/operation basis depending on their requirements for response time versus data accuracy. Cassandra offers a number of consistency levels for both reads and writes.
What parts of an Oracle database cannot be migrated to DataStax Enterprise/Cassandra?

Schema, data, and general indexes may be migrated, but objects that currently cannot be migrated include:

- Stored procedures
- Views
- Triggers
- Functions
- Security privileges
- Referential integrity constraints
- Rules
- Partitioned table definitions

Do I need to use an Oracle caching layer (like TimesTen) with Cassandra?

No. Cassandra negates the need for extra software caching layers like memcached through its distributed architecture, fast write throughput capabilities, and internal memory caching structures. When you want more memory cache available to your cluster, you simply add more nodes and it will handle the rest for you.

Is data absolutely safe in Cassandra?

Yes. First, data durability is fully supported in Cassandra, so any data written to a database cluster is first written to a commit log in the same fashion that nearly every popular RDBMS does.

Second, Cassandra offers tunable data consistency. This means a developer or administrator can choose how strong they wish consistency across nodes to be. The strongest form of consistency is to mandate that any data modifications be made to all nodes, with any unsuccessful attempt on a node resulting in a failed data operation. Cassandra provides consistency in the CAP sense, in that all readers will see the same values.

How is data written and stored in Cassandra?

Cassandra has been architected for consuming large amounts of data as fast as possible. To accomplish this, Cassandra first writes new data to a commit log to ensure it is safe. After that, the data is written to an in-memory structure called a memtable. Cassandra deems the write successful once it is stored on both the commit log and a memtable, which provides the durability required for mission-critical systems.

Once a memtable’s memory limit is reached, all writes are then written to disk in the form of an SSTable (sorted strings table). An SSTable is immutable, meaning it is not written to ever again. If the data contained in the SSTable is modified, the data is written to Cassandra in an upsert fashion and the previous data automatically removed.

Because SSTables are immutable and only written once the corresponding memtable is full, Cassandra avoids random seeks and instead only performs sequential I/O in large batches, resulting in high write throughput.

A related factor is that Cassandra doesn’t have to do a read as part of a write (i.e., check index to see where current data is). This means that insert performance remains high as data size grows, while with b-tree based engines (e.g., MongoDB) it deteriorates.

What kind of query language is provided in Cassandra? Is it like SQL in Oracle?

Cassandra supplies the Cassandra Query Language (CQL), which is very SQL-like. Queries are done via the standard `SELECT` command, while DML operations are accomplished via the familiar `INSERT`, `UPDATE`, `DELETE`, and `TRUNCATE` commands. DDL commands such as `CREATE` are used to create new keyspaces and column families.

Although CQL has many similarities to SQL, it does not change the underlying Cassandra data model. There is no support for JOINs, for example.