Building Applications with Self-Managing Databases:
Developers and Oracle Autonomous Database

Sponsored by: Oracle
Carl W. Olofson | October 2019
IN THIS WHITE PAPER

This white paper describes the challenges associated with building a major enterprise database application, including the design of the database itself. It elaborates the challenges developers face in ensuring that data is stored using the appropriate data model, kept secure, and delivered in a high-performance manner. This document examines the limitations specific to most major database management systems that compel developers to make compromises.

It also looks at Oracle Autonomous Transaction Processing and Oracle Autonomous Data Warehouse, the two forms of Oracle Autonomous Database fully managed database service, and shows the range of options available to developers when defining their data and building their applications. This document illustrates how Oracle Autonomous Database relieves developers of the need to configure and tune the database and how it supports application development for the cloud. It also describes the experiences of two organizations that developed applications using Oracle Autonomous Database.
Typical Challenges Facing Database Application Developers

Developers who are building database applications confront a multitude of considerations: They must consider what data is needed as well as how it is to be used and whether it might be subject to reporting, query, or analysis. They need to consider what constraints, if any, are imposed by the database management system that they are using, including issues regarding data types, data consistency, data combinations, and data with dependencies. The DBMS might even limit the development languages that developers want to use.

Data Organization Issues

A relational DBMS requires that the data be organized in tables, with each column value dependent only upon one unique value, which is the primary key. Data structures related to other structures must be in separate tables and joined by a foreign key. If a developer wants to manage a data structure that fully represents a particular application object, it may be necessary to map table data from multiple tables to that structure or use some sort of relational data mapping tool.
Today’s applications frequently process unstructured and semistructured data as well as relational data, prompting the need to also store other data types such as JSON documents, XML, key values, graphs, and spatial data. For example, a document DBMS allows developers to organize data inside a self-contained, hierarchical document however they want because a document database does not impose structural rules based on a schema, unless such a function is required by the team. This freedom to organize data puts an extra onus on the developer to ensure that data structures represented as documents do not overlap in content or, if they do, to include in application operations a capability to keep redundant data synchronized. It also makes the program code the sole source of information regarding the organization and meaning (by way of comments) of the data structure.

**Issues Related to Database Performance**

Developers sometimes need to make special provisions for database performance issues. They may need to optimize indexes, SQL execution plans, data model choices, database configuration, and the application’s data access patterns. They may need to make trade-offs in the degree of database consistency, isolation, and durability for performance and scalability. Other performance considerations include designing the application for distributed systems, replication, sharding, and caching.

**What Happens When the Data Changes**

Some applications use a data cache in memory, or some sort of grid-based distributed cache, to hold active application data and guarantee performance. If any of that data is also stored in a database, it may be necessary to capture any change to that data outside of the given application and refresh the cache with the changed data. In cases where the database is memory optimized, however, such a cache is often unnecessary.

In the document database case, if the document structures of later documents vary from those of earlier documents of the same type, the application must be coded to handle both older and newer formats. By contrast, a relational database is driven by a schema, and although it is more effort to change the schema structure, adjusting applications to these structure changes is usually straightforward if the data is normalized.
In any application database, however, when the data changes, either in instance data or in data structure, there can be performance shifts that require attention. This is especially true in the relational case if a SQL statement adds a SELECT that might call for an additional index on a table or if a document field develops a coordinative relationship with a corresponding field in another document. In these cases, the database usually needs to be retuned, which can cause changes to the data structure itself, server configuration and, worst case, even storage allocation, resulting in delays in the project.

Oracle Autonomous Database

Oracle offers a database service, called the Oracle Autonomous Database, which can address many of these issues mentioned previously. It is available as a managed service in Oracle Cloud and has two variations: Oracle Autonomous Data Warehouse for analytics workloads and Oracle Autonomous Transaction Processing for workloads that are transactional or mixed in nature.

A Fully Autonomic Database System Augmented by Cloud Services

Oracle Autonomous Database is a fully managed service, Oracle manages the resource allocation, software maintenance, and routine operations, including running backups and, if necessary, restores as well as handling high-availability and disaster recovery details. It is configured in such a way that patching and software upgrades can be performed without incurring downtime. It also includes an autonomic capability.
Technology that enables a computing asset to manage itself; it is typically self-tuning and self-healing. Oracle Autonomous Database implements its autonomic capability using machine learning (ML). ML enables the system to dynamically adjust its configuration to optimize performance and tune itself, taking the burden of these tasks off the developer. ML also enables the system to detect and correct problems as they happen, without requiring human intervention. Because it is constantly learning, these ML-driven functions continually improve over time.

**Multimodel Database Support**

Oracle Autonomous Database is based on Oracle Database, which includes multimodel support. This means that the development team can choose what data structures and operations it would like to store and execute that make sense for the application. If the data is managed mainly by a single application, and extreme agility is required, the team might choose a document data structure. Oracle Database fully supports JSON documents. If the data needs to be shared among many separately developed applications or needs to be recombined in various ways for analysis and reporting, using the fully relational features of Oracle Database might make more sense.

Oracle Autonomous Database also supports XML documents, as well as graph, and spatial database constructs. In addition to data organization, Oracle Autonomous Database provides operational characteristics that support microservices and IoT data processing and can act as a key value store or distributed ledger.

**ML Also Powers Faster Database Conversion**

If one is converting data from another database, even another Oracle Database, to Oracle Autonomous Database, one cannot simply load the data. It must be converted to Oracle Database 19c format, encrypted, and properly organized and indexed as it loads. Fortunately, with its ML-powered processes, Oracle Autonomous Database enables quick and easy data conversion.
Customer Experiences

IDC interviewed two customers in preparing this document to determine how they are using Oracle Autonomous Database and what benefits they are experiencing.

Henry Ford Health System is one of the leading healthcare providers in the United States. It includes hospitals, medical centers, and one of the country’s largest group practices, with more than 1,200 physicians practicing in more than 40 specialties. With more than 30,000 employees, Henry Ford Health System is the fifth-largest employer in the metro area of Detroit, Michigan.

A development manager was tasked with replacing a frontline system that interacted with patients. The system used a non-Oracle RDBMS and was mostly written in Java. He decided to move to Oracle Database, but was concerned about database complexity because as he put it, “I am a developer,” not a DBA. He moved the application to Oracle WebLogic Server and was persuaded to try Oracle Autonomous Transaction Processing. He praised the DBMS, which was easy to set up and performs very well with no tuning required.

Formerly, he had used a plethora of open source tools, but they required so much effort to manage and configure that, in his words, “that free open source stuff was costing me much more” than the toolset that came with the Oracle platform. He had been concerned that he had too many tasks for one person to perform, but now he is looking at more applications he can build with Oracle Autonomous Transaction Processing.
ImpulseLogic provides software to its retail customers that enables them to manage on-the-shelf inventory. Although the application they had been using to manage inventory worked well, customers were experiencing too much distortion caused by a mismatch between what the system said was on the shelf and what was actually there. The reason was performance.

The product offered a forecasting capability, so the store knew when to restock. Forecasting requires floor inventory (and uses sales to predict demand), tracking sales in real time, and decrementing the on-the-shelf count based on sales. The system could predict which SKU would go to minimum first and knew when it was time to replenish, eliminating the need for visual gap checking. It used a variety of factors to calculate a “demand curve.”

ImpulseLogic had been supporting any configuration and RDBMS that customers requested, but the lack of control over the environment in which the software operated was a key part of the problem in managing performance and scalability. The system was transaction oriented and high volume. One customer had 84 million SKUs to track, with state changes at the rate of 50 to 60 million items per hour, an average of roughly 15,000 per second. ImpulseLogic had been offering a cloud service based on a leading cloud provider with the user’s preferred RDBMS but decided to move to the Oracle Cloud with Oracle Autonomous Transaction Processing. Now, it can scale as demand grows with steady, reliable performance. ImpulseLogic praised the system’s ability to scale on demand, self-tune, and self-heal. The continuous availability of the system throughout patches and updates is another important advantage because many of its customers are open 24 hours. Most importantly, the transition to Oracle Autonomous Transaction Processing provided certainty in the inventory forecasts delivered to customers.

ImpulseLogic moved from its prior cloud platform and RDBMSs to Oracle Cloud in just six weeks and it was able to demonstrate the new system to customers after just four weeks. ImpulseLogic deploys a separate cloud service instance for each customer, most instances being very large.

In short, ImpulseLogic found that managing databases it had configured and deployed itself had become a huge challenge and that Oracle Autonomous Database addresses that challenge quite effectively. In addition, the development manager with whom we spoke said, “The technical team at Oracle we found to be outstanding, not just in expertise but in their willingness to work with developers.”
Self-managing systems are the future. Oracle has provided a comprehensive capability in the area of database management, that sets the standard for others to emulate. In time, users will come to expect self-managing databases, self-managing application systems, self-managing expert systems, and so on, available in the cloud.
There can be no doubt that all vendors offering managed database services in the cloud will move toward autonomic operation, using some form of ML, in addition to staff management to provide a full, hands-off database experience from the user’s perspective. With its lead in this area, Oracle has an excellent opportunity to stay ahead of the pack.

Oracle Autonomous Database streamlines and simplifies a multitude of operational challenges specific to managing databases for application development. For example, the machine learning–powered, autonomic capability of Oracle Autonomous Database enables it to dynamically modify its configuration parameters to optimize performance, automatically create indexes, and adjust SQL execution plans. The autonomic qualities of Oracle Autonomous Database enable it to automate scaling, self-tuning, and self-healing to ensure the delivery of reliable performance, security, availability, as well as accurate analytics. In addition, its multimodel database support empowers developers to build applications that leverage a diverse range of data types and objects. This support for a diversity of data types is particularly well suited for microservices-based application development that might require developers to use one data store for one microservice and a qualitatively different data store for another microservice.

Meanwhile, the fully managed quality of Oracle Autonomous Database empowers developers to focus on developing applications without worrying about operational considerations such as patches, updates, security, backups, disaster recovery, performance tuning, or high availability. Powered by machine learning, Oracle Autonomous Database is illustrative of the future of databases that are self-managing and demonstrative of intelligence and the capability to automatically adjust to changing input parameters and workloads. Developers will appreciate the ability of Oracle Autonomous Database to dynamically self-manage its operational behavior because they will subsequently be able to focus more of their attention on writing code.
About IDC

International Data Corporation (IDC) is the premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications and consumer technology markets. IDC helps IT professionals, business executives, and the investment community make fact-based decisions on technology purchases and business strategy. More than 1,100 IDC analysts provide global, regional, and local expertise on technology and industry opportunities and trends in over 110 countries worldwide. For 50 years, IDC has provided strategic insights to help our clients achieve their key business objectives. IDC is a subsidiary of IDG, the world’s leading technology media, research, and events company.

Global Headquarters

5 Speen Street
Framingham, MA  01701
USA
508.872.8200
Twitter: @IDC
idc-community.com
www.idc.com

Copyright Notice

External Publication of IDC Information and Data — Any IDC information that is to be used in advertising, press releases, or promotional materials requires prior written approval from the appropriate IDC Vice President or Country Manager. A draft of the proposed document should accompany any such request. IDC reserves the right to deny approval of external usage for any reason.

Copyright 2019 IDC. Reproduction without written permission is completely forbidden