IDC LINK
Real-time IDC Research® opinion on industry news, trends and events

Oracle Announces Oracle Autonomous Database for JSON

August 13, 2020

By: Carl W. Olofson

IDC's Quick Take

During a keynote emphasizing the multimodel nature of Oracle Autonomous Database, its support for a wide range of data management models, and the benefits that such a range delivers to developers, Oracle's Juan Loaiza announced that Oracle is now offering Oracle Autonomous Database for JSON. This is a packaging of Oracle Autonomous Database with a lower price for JSON-only workloads. It supports a range of open standards API options for accessing JSON, and technical optimizations that provide faster, cheaper support for JSON on Oracle than was possible previously. This offering is obviously aimed squarely at JSON document database competitors MongoDB and Couchbase, among others, but also is meant to give Oracle developers the ability to embrace JSON without leaving the Oracle Database domain, and to connect JSON data with relational table data in an operationally effective manner.

Product Announcement Highlights

On August 13, 2020, Oracle held a session of Oracle Developer Live on Zoom, presenting a series of sessions on Oracle Database development for database and database application developers, along with a series of "hands-on labs". The day was kicked off with a keynote address by Juan Loaiza, Executive Vice President, Mission-Critical Database Technologies at Oracle. He discussed a range of capabilities offered by Oracle Autonomous Database, including declarative data security, native SQL support for JSON, blockchain table support, memory optimization enhancements for OLTP, and easy to configure fault tolerance.

His major announcement, however, was immediate availability of the Autonomous JSON Database, which is a low-cost configuration of Oracle Database focused on JSON document support. It includes a form of optimized storage for JSON called "Native JSON storage" that delivers improved store and retrieve performance, with advanced indexing, transparent scale-out, and full ACID transaction support. It also includes support for open interfaces, including driver APIs, a command line interface, and REST API support. Users can use SQL, but they can also build their applications with no SQL at all.

He emphasized as benefits a capability for immediate provisioning and setup, low cost, and operations made simple because Oracle Autonomous Database provides ML-driven self-tuning. Also, the user of the Oracle Autonomous JSON Database can, by exercising a simple option, turn on relational support at any time, since the underlying technology is the same. There is a technical blog entry that describes the capability in detail.

IDC's Point of View

Oracle is putting heavy emphasis on the openness of its interface support, which includes REST, a variant of SQL for JSON called SQLcl, and a command interface called SODA (Simple Oracle Document Access) that resembles that of MongoDB. It can be used from the command line, or through language-native statements in Node.js, Python, Java, and of course, PL/SQL. Oracle also claims that Oracle Autonomous

JSON Database is cheaper and easier to use than the alternatives, and provided some supporting data from their own testing.

This is the latest salvo in Oracle's war on "purpose built" database systems; that is, database systems that are each designed to address one class of problem: relational OLTP, relational analytics, document, graph, timeseries, etc. Oracle is representing Oracle Database as a single database system that can address all those workloads, and do so efficiently. The alternative, from Oracle's point of view, is banding together a disparate set of database systems using ETL and other data connectivity software, introducing unnecessary operational and administrative complexity. A competing argument is that each specialized database system ought to be the absolute best at solving the problems it was designed to solve, and that using a multi-model DBMS represents an unacceptable compromise.

The reality is that both arguments have some merit, and neither is the total answer. In cases where an application is totally isolated from any other operational or analytic context and is focused on a specific set of managed data, the single use DBMS may make perfect sense, because it makes the development and management of the application and its data very simple, and can take advantage of performance optimizations in the DBMS. In other cases, where the data needs to be integrated with other data on a regular and timely basis, the multi-model approach may be preferred, since, as is the case with Oracle, the document data (in this case) can be kept in a consistent and integral relationship with other data, such as relational table data, by the DBMS. Of course, the fact that this is deployed on Oracle Autonomous Database means that the database is self-tuning, self-correcting, and has all the other automated features that reduce to near zero the amount of human deployment management that needs to be done.

The idea that all enterprise data can be kept in a single database instance is unrealistic, because multiple application development and analytics projects working on the same database will inevitably butt heads, create excessive schema complexity, and produce an operational nightmare. This is not to say, however, that managing all or most data on the same technology base is a mistake; rather, it offers considerable benefits in terms of integration and operational simplicity, especially when the database is self-managing. All in all, the idea of keeping all or most enterprise data in database instances that have the same underlying technology, and can be linked together, has considerable merit, and Oracle seems to be positioned to offer that possibility.

Subscriptions Covered:
Data Management Software