Security and Performance advances with Oracle Big Data SQL

Jean-Pierre Dijcks
Oracle
Redwood Shores, CA, USA

Key Words

SQL, Oracle, Database, Analytics, Object Store, Files, Big Data, Big Data SQL, Hadoop, Kafka, NoSQL, Streams, Security

Introduction

Oracle Big Data SQL enables organizations to immediately analyze data across Apache Hadoop, Apache Kafka, various NoSQL systems and Oracle Database leveraging their existing SQL skills, security policies and applications with extreme performance. From simplifying data science efforts to unlocking data lakes, Big Data SQL makes the benefits of Big Data available to the largest group of end users possible.

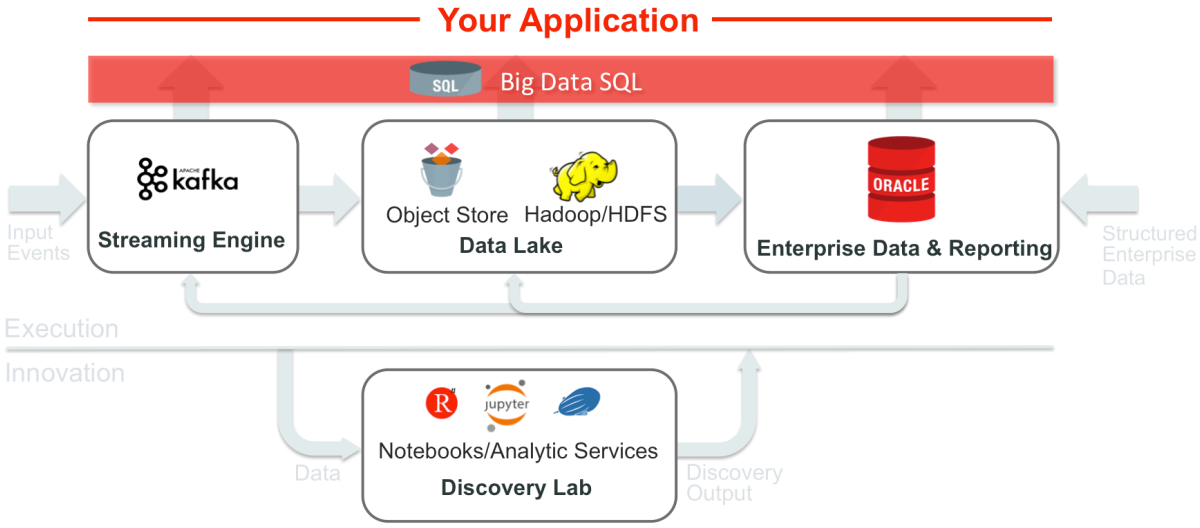

Figure 1. The Role of Big Data SQL in Oracle's Information Management Architecture

Oracle Big Data SQL is designed to support a wide range of deployment options and platforms. Big Data SQL requires 1) any Oracle Database 12c (version 12.1.0.2 or higher) running Enterprise Linux and 2) leading Apache Hadoop distributions from Cloudera and Hortonworks. Big Data SQL achieves highest performance when paired with Oracle Engineered Systems. Big Data SQL takes full advantage of the power of Oracle Exadata and Oracle Big Data Appliance to create a best-in-class Big Data Management System, unifying the power of big data and Oracle Database. For those looking to bring the power of Oracle SQL to big data in the cloud, Big Data SQL Cloud Service is available in combination with Oracle Exadata Cloud Service and Oracle Big Data Cloud Service, and will be expanded to support a wider variety of platforms going forward.

Current Feature Set in Big Data SQL

With Big Data SQL releasing it's version 3.2, a deep set of features is available. Before elaborating on the specific new elements in version 3.2, here is a quick recap of what these core features are:

Topic / Feature	Description
Enhanced External Tables	Big Data SQL dynamically tracks metadata about external data sources at runtime – including both clusters and the tables within them – without moving or copying data. External tables for Big Data SQL provide: **Seamless metadata integration and queries** which join data from Oracle Database with data from Hadoop, Kafka and NoSQL databases; **Automatic runtime mappings** from metadata stored in HCatalog (or the Hive Metastore) to Oracle Tables; **Multiple cluster support** to allow one Oracle Database to query multiple Hadoop clusters and **Enhanced access parameters** to give database administrators the flexibility to control column mapping, error handling and data access behavior
Partition Pruning & Predicate Pushdown	Big Data SQL's *predicate push down* technology allows predicates in queries issued in Oracle Database to be executed by remote systems, and to be pushed into certain file formats for optimal data filtering. Using predicate push down, Big Data SQL enables you to: P**rune partitions** from tables managed by Apache Hive; **Minimize I/O** on files stored in Apache Parquet and Apache ORC formats; **Enable remote reads** on data stored in Oracle NoSQL Database or Apache HBase
Smart Scan	Smart Scan enables Oracle SQL operations to be pushed down to the storage tiers of the Big Data system. Paired with the horizontal scalability of these storage systems, Smart Scan automatically provides parallel processing equal to your biggest data set, enabling: **Locally filtered data**; **Join optimization** via Bloom filters; **Scoring** for data mining models and **enhanced processing** for querying document data sets in for example JSON or XML and **Oracle-native operators** providing complete fidelity between queries run with Big Data SQL and Oracle Database alone

Storage Indexes	As data is accessed, Oracle Big Data SQL automatically builds data **local, in-memory indexes** that capture where relevant data is stored. On subsequent queries of the same data, Storage Index technology ensures that data blocks that are not relevant to the query are not read. Because data blocks in Big Data systems can be very large (up to hundreds of megabytes), this "I/O skipping" strategy can improve performance on some queries by orders of magnitude.
Information Life Cycle Management	Various techniques are possible to optimize data placement: Oracle Big Data SQL includes the Oracle **Copy to Hadoop** utility. This utility simplifies copying Oracle data to the Hadoop Distributed File System (HDFS). Data is stored in Oracle Data Pump format. This format optimizes queries thru Big Data SQL: 1) the data is stored as Oracle data types – eliminating data type conversions and 2) the data is queried directly – without requiring the overhead associated with Java SerDes. Other formats include Parquet and ORC. An additional technique is based on **Oracle Partitioning** where a single table's data partitions can be stored on the various tiers. Database queries seamlessly access this archive data as they would any other data – exploiting the optimized access and storage structures (like indexes) for fast query performance.
Database Security on Big Data	Oracle Big Data SQL's unique approach to integrating data enables Oracle Database Security features on Hadoop and NoSQL data. Using standard Oracle security mechanisms, you can secure Big Data using standard Oracle **Database roles and privileges** to govern access to data; **data redaction**, to ensure that sensitive information is obscured when accessed by unauthorized users and **Virtual Private Databases** to better enforce governance policies

More information can be found here:

- www.oracle.com/database/big-data-sql
- blogs.oracle.com/datawarehousing/

New in Big Data SQL 3.2

The new features that are debuting in this latest version include:

- Expanded source support, now covering Apache Kafka through Oracle SQL queries
- Enhanced security, enables multi-user authorization to be used for data access in the remote system
- Dedicated Parquet driver, delivering up to 8x performance increases when compared with the standard Open Source Parquet driver used previously

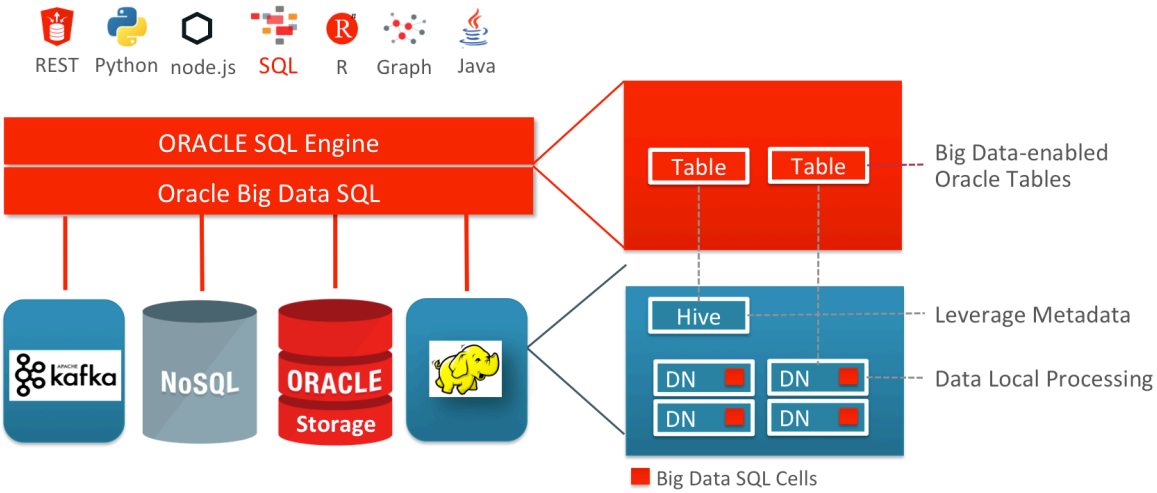

Figure 2. Big Data SQL Cell Architecture

Kafka Integration

One of the major changes in the big data ecosystem is the rapid advance and uptake of Apache Kafka. Many customers are looking at Kafka as an integral part of their data lake, where Kafka plays the role of integration engine and consolidation area for a large number of data sets. Many customers use Kafka to integrate both streaming data as well as (micro) batches of data. The main reason for using Kafka to simplify the ingest patterns by standardizing on a single "API" and ingest infrastructure. Downstream applications can then subscribe to streams.

Kafka is disrupting architectures in one more, very significant, way: Lambda architectures, with different data-at-rest repositories are starting to look obsolete. Instead of ingesting into a NoSQL store, running a Kafka cluster and HDFS cluster seem to become the more desirable architecture. One major reason is simplicity. Kafka can play both the roll of an integration platform, can retain data for a period of time and can be queried directly with SQL. Why continue to use HDFS? The primary reasons are the cost of the infrastructure and the fact that Kafka is not a database. Data is stored in "log form" and this makes queries and analytics less efficient. Hence the "requirement" to keep HDFS around to enable long term, dedicated research with adequate performance and agile data access.

With that in mind, it seems obvious to expand the reach of Big Data SQL to include direct SQL access to Kafka. To enable this, Oracle developed a separate and specialized storage handler for Kafka. This new storage handler leverages all the Hadoop extensibility infrastructure and now enables the definition of SQL queries against topics in Kafka. The initial version of the storage handler pushes down predicates to the Kafka system, using the following structures in Kafka:

- Topic
- Partionid
- Offset

These initial pushdowns ensure Big Data SQL does IO elimination and tailors the queries to the system for optimal performance. Various other optimizations are being worked on, like leveraging the recent timestamp API in Kafka.

Dedicated Parquet Driver

Big Data SQL has supported both ORC and Parquet from the early days of the product, however the initial support for Apache Parquet was based on the open source, included Parquet Hive driver. Over time that driver showed performance issues for customers and Oracle has decided to create a dedicated Parquet driver to optimize performance for an increasingly heavily used data source.

The new Parquet driver delivers the following optimizations:

- Lazy Materialization – optimizing IO for Parquet queries by intelligently processing columns where rows satisfy the filter applied. This gives the new driver both columnar IO benefits, and within that the optimization to skip parts of the columns. This optimization leverages the full metadata of the Parquet file to enhance performance
- Dictionary based filtering – optimizing CPU consumption by leveraging the dictionary – instead of the data – to do filtering for queries
- Streamlined data conversion – which optimizes the CPU utilization by optimizing the internal code used to convert between Parquet data types and Oracle data types

Across various queries and tests, the dedicated parquet driver has shown much improved scalability and a big bump in performance, of up to 8x over the original driver used.

Multi-User Security

Big Data SQL has enabled Oracle Security on any data and Big Data SQL has used tools like Sentry to control access to SQL data, as well as adhered to Access Control Lists on files in HDFS.

The new feature that is introduced with version 3.2 is the ability to run queries as the original application user. Today, queries access data in HDFS as the oracle user as they are originated from the database and run in that scope. This leads to some issues:

- Cannot authorize data access based on the user that actually runs the query. Authorization rules must be replicated within the Oracle Database to ensure the correct users access the acceptable data elements
- Hadoop auditing shows that all Big Data SQL queries are run by oracle – not the user running the query.

The new multi-user functionality removes these issues; data access is performed using the identity of actual application user. Because there are many ways to connect and authenticate to the Oracle

Database (LDAP, Kerberos, database users, application users, etc.) – a critical element of the new capability is to provide a declarative approach to identifying the actual query user. The feature leverages capabilities like the SYS_CONTEXT function to map the current database user to the query user. This mapping information is captured in a new metadata table (bdsql_user_map) and populated by security administrators using the dbms_bdsql.add_user_map procedure.

The end result of this mapping and this new feature is an end-to-end secure and auditable trace of data access; underlying file access privileges are automatically leveraged. Oracle advanced security features – like Virtual Private Databases and Data Redaction can then be layered on top of the basic Hadoop access controls. No SQL tools on the market enable this deep integration between two disparate data repositories.

Summary

Big Data SQL is a strategic component to Oracle's Information Management strategy and as such is continually updated to handle the changing demands of customers. Big Data SQL 3.2 introduces integration with Kafka, a high performance Parquet driver and an innovative approach to enhancing security in a distributed data environment.

Contact:

Jean-Pierre Dijcks
Oracle
500 Oracle Parkway
MS 4op720
Redwood Shores, CA 94065

Phone: +1 650 607 5394
E-Mail: jean-pierre.dijcks@oracle.com
Internet: www.oracle.com, cloud.oracle.com/bigdata