13 New Features in 23ai Release Updates

This section describes the new features for 23ai release updates.

Release Update 23.5 Features

Archiving and Unarchiving of Gold Images

You can archive and unarchive Oracle FPP gold images that are not currently used but cannot be deleted for future access. Archiving and unarchiving gold images allow you to store those images on external storage devices of your choice in a space-efficient, compressed fashion, thereby freeing up storage space on high-end storage devices hosting the central FPP gold image repository.

Archiving gold images that are not currently used, but need to be retained, saves space and costs by allowing you to flexibly and efficiently store them on external storage devices of your choice.

View Documentation

BINARY Vector Dimension Format

BINARY is a new dimension format that can be used with the `VECTOR` data type. Each dimension of a BINARY vector can be represented with a single bit (0 or 1). A BINARY vector itself is represented as a packed UINT8 array, for example, a single UINT8 value represents 8 dimensions of the BINARY vector. BINARY vectors can be generated using embedding models provided by Cohere (for example, embed v3), Hugging Face Sentence Transformers, and so on.

BINARY vectors offer two key benefits compared to FLOAT32 vectors:

1. The storage footprint of BINARY vectors is 32X lesser, and
2. Distance computations on BINARY vectors can be up to 40X faster, which accelerates Vector Search

BINARY vectors can provided reduced accuracy compared to FLOAT32 vectors. But, evaluations on various datasets have shown that they can still achieve 90% or higher accuracy of FLOAT32 vectors.

View Documentation

Backup, Restore, and Relocation for FPP Server

You can create a backup of the Oracle Fleet Patching and Provisioning (Oracle FPP) server, restore data from the backup, and relocate the server from backup to new hardware. Relocate the Oracle FPP Server to new hardware and re-point Oracle FPP targets whenever needed.

Ensure data safety by backing up Oracle FPP Server and restoring data in case of failure.

View Documentation

Custom Certificates for Oracle FPP Server Authentication

Starting with Oracle Grid Infrastructure 23ai, you can specify custom security certificates for authentication between Oracle Fleet Patching and Provisioning (Oracle FPP) server and clients. By default, Oracle FPP uses SSL/TLS certificates. You can use any security certificate assigned by a Certificate Authority of your choice.

Choosing custom security certificates enables you to meet stringent security policies and regulations set by your organization.

View Documentation

Duplicated HNSW Vector Indexes on RAC

HNSW Vector Indexes are now supported on RAC environments through full duplication on all instances of the cluster that have sufficient memory in the Vector Pool. On Autonomous Database Serverless deployments, the Vector Pool is autonomously managed.

All copies of the HNSW index across different RAC instances share the same ROWID-to-VID mapping table on disk. However, each instance builds its in-memory neighbor graph independently, and hence, its possible to get different results for approximate searches depending on which RAC instance is used to serve the query.

Enterprise customers often deploy Oracle Database in RAC environments. This feature enables creation of HNSW vector indexes for RAC through full duplication across all instances of the cluster. Queries directed at any instance of the RAC cluster can take advantage of HNSW vector index plans resulting in ultra-fast similarity searches.

View Documentation

FPP Metadata on External Databases

Starting with Oracle Grid Infrastructure 23ai, Oracle Fleet Patching and Provisioning (FPP) stores metadata in a local Single Instance Enterprise Edition Database or an external Oracle metadata repository. New installations will store the metadata by default in a Single Instance Enterprise Edition Database.

Allowing to choose an external metadata repository database or a local Single Instance Enterprise Edition Database database simplifies the deployment of the Fleet Patching and Provisioning server.

View Documentation

General Improvements for Oracle FPP Job Scheduler

The Oracle Fleet Patching and Provisioning (Oracle FPP) job scheduler now supports pausing and resuming jobs, forcing jobs to start in a paused state, and pausing jobs between batches in a chain execution and allows to tag jobs for easy querying.

Additional job scheduler options provide you with more control over the patching execution. You can pause jobs to verify executed jobs and resolve dependencies before resuming jobs.

View Documentation

General Purpose Cluster Configuration

The General Purpose Cluster is a new cluster deployment that requires minimum network and storage configuration and supports applications that benefit from managed server restarts and failover capabilities. Oracle Grid Infrastructure installer provides a streamlined option to configure a general-purpose cluster in either interactive or silent mode.

This feature increases productivity and reduces the required deployment time by simplifying deployment requirements.

View Documentation

Gold Image Based Out of Place Patching

Oracle DBCA can apply Release Updates (RUs) out-of-place using Gold Images for both Oracle Grid Infrastructure homes and Oracle Database homes.

Oracle DBCA can apply the new Gold Image-based quarterly Release Update patches to Oracle Grid Infrastructure home and Oracle Database homes.

View Documentation

JSON Collections

JSON collections are special tables or views that store (or represent) only JSON documents in a document-store-compatible format, such as the Oracle Database API for MongoDB. JSON collections are integrated into the database and fully operable with SQL, from creation to manipulation and query processing. For example, it is possible to do a simple INSERT AS SELECT into a JSON collection table.

JSON collection tables complement JSON Duality Views, the marquee JSON collection views that provide the benefits of relational storage and JSON document processing with a single database structure.

Native JSON collections simplify working with JSON data stored in collections within the Oracle Database ecosystem. For example, with collections, you easily analyze your JSON documents with SQL while concurrently using those operationally with document-centric APIs, such as the Oracle Database API for MongoDB.

View Documentation

JSON_ID SQL Operator

SQL operator `JSON_ID` generates a unique document-identifier value, for unique access to JSON documents in a collection. The argument to `JSON_ID` determines whether the value is a 12-byte `OID` or a 16-byte `UUID`. In Oracle JSON collections, `JSON_ID` is used to create (automatically or explicitly) the values for document-identifier field `_id`.

`JSON_ID` simplifies the generation of ID values to uniquely identify JSON documents.

View Documentation

Optimized Oracle Native Access to NVMe Devices Over Fabric

Starting with Oracle Database 23ai, you can use TCP/IP network connections to access the remote NVMe storage devices using NVMe over Fabrics (NVMe-oF). The Oracle Grid Infrastructure server works as an initiator that connects to an NVMe-oF storage target created using Linux Kernel nvmet_tcp module, providing optimized user mode access to remote NVMe devices.

NVMe-oF provides a low-latency and secure way to access remote NVMe devices exported using NVMe Over Fabrics target. Oracle provides an optimized way to access these NVMe-oF devices directly from an Oracle process. This Oracle-native method of accessing NVMe-oF devices reduces latency while Oracle ASM makes storage manageability easier.

View Documentation

Oracle DBCA Support for PMEM Storage

Oracle Database Configuration Assistant (Oracle DBCA) enables you to select persistent memory database (PMEM) as your storage option when creating a single-instance database.

This feature automates the process of assigning a PMEM device for your database storage enabling you to place database files in a PMEM storage device.

View Documentation

Oracle DBCA Support for Standard Edition High Availability

Using the Oracle Database Configuration Assistant (Oracle DBCA) and facilitating Oracle's Automatic Storage Management or Oracle's Advanced Cluster File System, you can now quickly create a Standard Edition High Availability Oracle Database fully configured for automatic failover.

Oracle Standard Edition High Availability Database can now be created very easily with more automation, eliminating manual steps and the associated complexity.

View Documentation

Oracle Database Installer Command-Line Support

Oracle Database Installer now supports specifying commands and input parameters for those commands using the command-line interface.

Easier and simpler Oracle Database deployments are supported using the command-line interface in addition to the graphical user interface.

[View Documentation]

Oracle FPP Local Mode Without Java Container

Oracle Fleet Patching and Provisioning Local Mode does not require the Grid Infrastructure Management Repository (GIMR) and the Java container.

Oracle Grid Infrastructure and Oracle Database administrators can use Oracle FPP Local Mode without setting up any additional components on a cluster, making the patching process simpler and faster.

[View Documentation]

Oracle Grid Infrastructure Installer Command-Line Support

Oracle Grid Infrastructure Installer now supports specifying life-cycle management operations and input parameters for those operations on the command line.

Easier and simpler Oracle Grid Infrastructure deployments are supported using the command line, in addition to the graphical user interface.

[View Documentation]

Oracle Grid Infrastructure Installer Improvements

The Oracle Grid Infrastructure installer has been upgraded with options to create and manage gold images and perform out-of-place patching while reducing the inventory metadata to effectively manage installation and patching.

Out-of-place patching using the Oracle Grid Infrastructure Installer directly makes patching more manageable and reliable.

[View Documentation]

Single-Server Rolling Database Maintenance

Single-Server Rolling Database Maintenance creates a new local database home and starts a second instance of the same database from the new home on the same server, allowing you to perform rolling patching and maintenance operations on a single server hosting an Oracle RAC One Node or Real Application Clusters (Oracle RAC) database.

Single-Server Rolling Database Maintenance provides database availability during maintenance activities (such as patching) on a single server hosting an Oracle RAC or Oracle RAC One Node database. This feature significantly improves the availability of your single-node databases without expanding them to a multi-node cluster and adding support for shared storage.

View Documentation

Store Images and Transfer Working Copies as ZIP Files

Store Oracle Fleet Patching and Provisioning (Oracle FPP) gold images as ZIP files, create ZIP files from existing Oracle homes, and transfer these ZIP files.

Save significant storage, bandwidth, and transfer time by storing and transferring gold images as ZIP files.

View Documentation

Vector Memory Pool Automatic Management

When using Autonomous Database services (ADB), the Vector Pool dynamically grows and shrinks when HNSW indexes are created or dropped respectively.

Because you cannot explicitly set any SGA-related memory parameters when using Autonomous Database services, this feature automatically maintains the necessary amount of Vector Pool memory needed for your HNSW indexes.

View Documentation

Verifying Digital Signature and Integrity of Installation Archive Files

Starting with Oracle Database 23ai, Oracle digitally signs the installation archive files with Oracle certificates.

Oracle digitally signs the installation archive files to allow customers to ensure the integrity of the packages before deploying them in their environments.

View Documentation

Release Update 23.6 Features

AI Vector Search: New Vector Distance Metric

AI Vector Search was extended in 23.6 to include a new vector distance metric called Jaccard distance.

This feature allows users the option to use another distance metric between vectors.

View Documentation

Hybrid Vector Index

The Hybrid Vector Index (HVI) is a new index that allows users to easily index and query their documents using a combination of full text search and semantic vector search to achieve higher quality search results.

The Hybrid Vector Index (HVI) simplifies the process of transforming documents into forms that are amenable both for vector similarity search and for textual search through a single index DDL.

The HVI provides a unified query API that allows users to run textual queries, vector similarity queries, or hybrid queries that leverage both of these approaches. This lets users easily customize the search experience and enhances search results.

View Documentation

Partition-Local Neighbor Partition Vector Index

This feature enables LOCAL indexing for Neighbor Partition Vector Indexes, optimizing search performance for partitioned tables. This feature conceptually creates a dedicated vector index for each partition, allowing queries with partition key filters to search only the relevant index partitions. As a result, vector searches are more efficient, leading to significantly lower response times when querying large partitioned datasets.

Large enterprise datasets are frequently partitioned by relational attributes to optimize performance. By enabling LOCAL Neighbor Partition Vector Indexes, users benefit from enhanced scalability and accelerated query performance through partition pruning. This approach also ensures more efficient data lifecycle management, making it ideal for handling large-scale enterprise workloads.

View Documentation

Persistent Neighbor Graph Vector Indexes

The HNSW vector index is an inmemory resident multi-layered graph index. The time taken to recreate the inmemory graph on a restart can be improved by having a disk checkpoint image of the graph. This feature adds the checkpoint format as well as the framework to take a disk checkpoint and then use it to recreate the inmemory resident graph structure.

Getting an index access plan after a restart can take a long time. A higher priority disk checkpoint based reload execution improves the time taken to get an index access plan after a restart.

View Documentation

Sparse Vectors

Sparse Vectors are vectors that typically have large number of dimensions, but only a few dimensions have non-zero values. These vectors are often produced by sparse encoding models such as SPLADE and BM25. Conceptually, each dimension in a sparse vector represents a keyword from a specific vocabulary. For a given document, the non-zero dimension values in the vector correspond to the keywords (and their variations) that appear in that document.

Sparse vectors, such as those generated by models like SPLADE and BM25, often outperform dense vectors from models such as BERT, in terms of keyword sensitivity and effectiveness in out-of-domain searches. This superior performance is especially valuable in applications where precise keyword matching is crucial, like in legal or academic research. Additionally, sparse vectors are often used for Hybrid Vector Search, where they can be used alongside dense vectors to combine semantic searches and keyword searches, and provide more relevant search results.

View Documentation

Transactional Support for Neighbor Graph Vector Indexes

HNSW Index is an in-memory hierarchical graph index for vector data. In 23.4, and 23.5, DMLs were not allowed on tables that have HNSW index built on their vector column(s). This feature enables transactions to be executed on such tables. Moreover, vector search queries that use the HNSW Index will see transactionally consistent results, based on their read snapshot. Transactional consistency is guaranteed even on Oracle RAC where the HNSW Index is duplicated on all instances in the Cluster, DMLs occur on one or more instances in the Cluster, and search queries can be executed on any instance in the Cluster.

HNSW Index is the fastest vector search index that Oracle offers in 23ai. Thus, customers want to use HNSW Index for search queries, while also issuing DML modifications on relational or vector columns in the underlying table. Since DMLs may render the in-memory HNSW Index structures stale, special protocols are added in this project to guarantee transactionally consistent results for customers.

View Documentation

Vector Format Output for Feature Extraction Algorithm

Feature extraction algorithms produce a set of features that represent projections in a lower dimensional latent space. The output is typically numerical and dense. VECTOR type representation is the natural choice.

Feature extraction algorithms represent a principled approach to vectorizing relational data. The vectorized representation can be used for similarity search.

View Documentation

GoldenGate Replication of JSON-Relational Duality Views

This feature allows developers to use Oracle GoldenGate technology to replicate JSON-relational duality view data **as JSON documents**, instead of relational tables, from an Oracle Database to a target Oracle or non-Oracle database.

Replication of JSON data is an important feature for high availability, fail-over, and real-time migration from a non-Oracle database to Oracle Database.

Oracle GoldenGate Replication to non-Oracle databases such as MongoDB (a document database) or Redis (a NoSQL key/value store) is simple and performant with the ability to replicate JSON documents from JSON-relational duality views.

Developers need not write complex JSON and SQL transformations to shape data for relational and document-based updates, or pay the cost of reconstructing application objects on the target database.

View Documentation

JSON-Relational Duality Views: Hidden and Generated Fields

You can map duality view data to *hidden fields*, absent from the view's documents. A *generated field* can use the value of hidden fields.

A document supported by a duality view can be simpler if it need not have a field for every underlying column, in particular for columns needed only for calculating other field values.

View Documentation

JSON-Relational Duality Views: Add Fields With Calculated Values

Duality views generate JSON documents containing objects whose fields map to columns of relational tables. You can add object fields whose values are calculated automatically.

When the database automatically augments objects by adding calculated fields, such calculation need not be done by multiple, separate client applications. And field calculations can refer to stored data that is not exposed in the resulting objects - a feature similar to redaction.

View Documentation

JSON Collection Views

JSON collection views are special, read-only database views that expose JSON objects in a JSON-type column named DATA.

JSON collection views are conceptually close to JSON-relational duality views, but they have fewer restrictions because they are read only.

View Documentation

JSON Replication

The `logical_replication_clause` of the `CREATE/ALTER TABLE` statement is extended to allow disabling and enabling of partial JSON updating under supplemental logging.

Partial JSON updating makes replication more efficient because less data needs to be replicated or modified. The change can be replicated on the remote side, instead of sending all updated data. Using this functionality, you will experience better performance on the same hardware, thus reducing hardware costs.

View Documentation

JSON Search Index Path Subsetting

When creating a JSON search index you can specify the fields to include or exclude from indexing: path subsetting.

Path subsetting can reduce the size of a search index and improve its performance.

View Documentation

Replication Support for JSON Collection Tables

JSON Collection Tables can be enabled for logical replication using GoldenGate. Replication is supported to and from JSON Relational Duality Views as well to and from third-party products, such as MongoDB.

Replication is a basic database functionality that works between Oracle Databases. It is also used to facilitate online migration to Oracle Database from third-party databases, such as MongoDB.

View Documentation

CONSTRAINT_NOVALIDATE, a Data Pump Import TRANSFORM Parameter

The Oracle Data Pump `TRANSFORM` parameter option `CONSTRAINT_NOVALIDATE` enables you to set validation preferences. It has two options: `Y` or `N`. When set to `Y`, constraints are not validated during import.

Validating constraints during import that were valid on the source can be unnecessary and slow the migration process. Validation can be done after import.

View Documentation

Disable Statistics Gathering During Autonomous Database Import Operations

The `DATA_OPTIONS=DISABLE_STATS_GATHERING` parameter option disables statistics gathering during an import job.

Statistics gathering can have a performance impact on a large import operation. This parameter disables statistics gathering during an import job. The parameter is also supported for on-premises and user-managed cloud services import operations. Statistics gathering is not automatic in non-autonomous database environments, but this parameter can be useful. It controls whether Oracle Autonomous Database gathers environment statistics automatically for DML operations, such as data loading during an Oracle Data Pump Import job.

View Documentation

Enhancements to Oracle Data Redaction

This release includes many enhancements to Oracle Data Redaction, such as the optimization of existing capabilities and the removal of previous limitations.

These enhancements to Oracle Data Redaction improve the overall experience.

View Documentation

Parallel Index Creation Parameters for Data Pump Import

The Oracle Data Pump Import command-line mode `ONESTEP_INDEX` parameter optimizes index creation concurrency and balances it with overall job parallelism. It does this in conjunction with the `INDEX_THRESHOLD` parameter.

Index creation is an essential step in the migration process. Enabling parallelism for large index creation and balancing the needs of large index creation with overrall import job parallelism makes more efficient use of parallel resources assigned to a Data Pump import job. It makes the logical migration process more effective by substantially reducing the time to import. Finally, it removes the effort associated with the extra step of extracting the large index definitions from the source dumpfiles and creating these large indexes outside the import job.

View Documentation

Sessionless Transactions

Managing a transaction requires the connection and session resources to be tied to the transaction throughout its lifecycle. Therefore, the session or connection can be released only after the transaction has ended. This often results in underutilization of sessions/connections. In Sessionless Transactions, after you start a transaction, you have the flexibility to suspend and resume the transaction during its lifecycle. The session or connection can be released back to the pool and can be reused by other transactions, therefore effectively being able to multiplex transactions and sessions/connections.

Sessionless Transactions provide ability for applications to suspend and resume transactions across sessions/connections (single instance or RAC) without the need for an external transaction manager, and without the application having to coordinate the commit and recovery protocols. The database manages transaction lifecycle, including commit and recovery. Application performance, and throughput, benefit from reduced commit latency since fewer client-server roundtrips are needed. Since external coordination is not required, using Sessionless Transactions results in vastly simplified mid-tier or app-tier infrastructure, and significantly decreases downtimes when compared with externally coordinating transactions (such as with XA).

View Documentation

XMLTYPE_STORAGE_CLAUSE, a Data Pump Import TRANSFORM Parameter

Transportable Binary XML simplifies the XML data storage and makes it easier to transport. It does not store the metadata used to encode or decode XML data in a central table.

`XMLTYPE_STORAGE_CLAUSE` now takes the options `TRANSPORTABLE BINARY XML` or `BINARY XML`. Oracle recommends that you use the `TRANSPORTABLE BINARY XML` XMLType, the new and recommended storage type for Release 23ai to store data in a self-contained binary format.

Use the `BINARY XML` (Non-Transportable) storage XMLType to store data in a post-parse, binary format designed specifically for XML data. Binary XML is compact, post-parse, XML schema-aware XML data. Binary XML is non-transportable and stores the metadata used to encode or decode XML data efficiently in a central table.

Data Pump can export and import data of type XMLType regardless of the source database XMLType storage format (object-relational, binary XML or CLOB). Oracle Data Pump exports and imports XML data as Transportable Binary XML so the the source and target databases can use different XMLType storage models for the data.

View Documentation

Release Update 23.7 Features

AI Vector Search: Arithmetic and Aggregate Operations

Just as you can add (+), subtract (-), or multiply (*) dates, timestamps, intervals, and numbers, you can now apply these arithmetic operators to vectors. The arithmetic operation is performed at each dimensional element of the vectors. It's also possible to calculate the SUM or AVG of a set of vectors.

Arithmetic operations on vectors allow AI systems to manipulate and combine abstract concepts, which enhances their ability to understand and process language or data in more sophisticated ways.

View Documentation

Additional Flexibility Defining JSON-Relational Duality Views

You can use `field _id` in document subobjects to identify a column that selects document, and use an identity column as an identifying column.

These additional possibilities when defining a duality view provide more kinds of documents that can be supported, and thus allow for more application use cases.

View Documentation

Change Compatible to 23.6.0 to Use New AI Vector Search Features in 23.6 or Later Releases

Starting with Oracle Database 23ai (23.6), the ability to use the new AI Vector Search features is introduced in Release Update (RU) 23.6 or later releases.

Regarding the `COMPATIBLE` parameter in Release Update 23.6:

1. New customers installing Oracle Database 23ai Release Update 23.6 directly:
   - The 23.6 RU fresh install and using db create comes with the `COMPATIBLE` parameter set to 23.6.0 by default.
   - After this installation, you will be unable to downgrade `COMPATIBLE` parameter to a lower value later.
2. Customers on Oracle Database 23ai Release Update 23.4 and 23.5:
   - Before patching, the `COMPATIBLE` parameter can be 23.0.0 or 23.4.0 or 23.5.0.

    - When you install the 23.6 release update (that is, when you patch to RU 23.6), to access the new Vector DB features, you must manually set the `COMPATIBLE` parameter to 23.6.0.
    - Note: In RU 23.6.0. updating the `COMPATIBLE` parameter requires downtime. It is not automatically done as part of patching. You must choose to update the database `COMPATIBLE` setting.
3. Customers on Oracle Database 19c or 21c:
    - When you upgrade the database to Oracle Database 23ai, the `COMPATIBLE` parameter remains as previously set in the source database.
    - If you want to use the new Vector DB features, then you must manually set the `COMPATIBLE` parameter to 23.6.0.
    - Note: The minimum `COMPATIBLE` setting permitted for upgrading to 23ai is 19.0.0. In other words, you cannot upgrade directly from Oracle Database 12c or 18c to Oracle Database 23ai.

View Documentation

Cloud Developer Packages

The Cloud Developer packages provide built-in tools to connect, process, and integrate with cloud services seamlessly. These packages, namely DBMS_CLOUD, DBMS_CLOUD_PIPELINE, DBMS_CLOUD_REPO, DBMS_CLOUD_NOTIFICATION, and DBMS_CLOUD_AI collectively provide the following advantages:

- Enable easy data access to cloud storage
- Automate workflows through pipelines
- Trigger event notifications
- Integrate AI capabilities for use directly from the database

DBMS_CLOUD_AI, which supports the Select AI feature, provides a SQL and PL/SQL interface to leverage Large Language Models (LLMs) and transformers. Select AI enables natural language-to-SQL generation and retrieval augmented generation (RAG), among other AI-based features.

The Cloud Developer packages enable businesses to work with data stored in the cloud, process real-time data, and use advanced tools like AI, benefiting from cloud capabilities. This approach helps simplify operations and supports specifically a mix of on-premises and cloud systems.

View Documentation

DBMS_DEVELOPER Package

The `DBMS_DEVELOPER` package provides an efficient way for developers to retrieve metadata for database objects, such as tables, views, and indexes.

In previous releases, object metadata had to be retrieved using the `DBMS_METADATA` package, and then queries had to be performed on various data dictionary tables and views. This method returned metadata in XML format.

The `DBMS_DEVELOPER` package returns metadata in JSON format, which can be easily integrated with applications and services, such as developer tools. You can adjust the level of detail to control the amount of metadata returned, tailoring it to your application's needs. Additionally, metadata retrieval time is significantly improved, enhancing integration with applications and services.

View Documentation

Dimension-Wise Arithmetic Support in PL/SQL

Addition (+), subtraction (-), and multiplication (*) can now be applied to vectors in PL/SQL. The arithmetic operation is performed at each dimensional element of the vectors.

Arithmetic operations on vectors allow AI systems to manipulate and combine abstract concepts, enhancing their ability to understand and process language or data in more sophisticated ways. PL/SQL support of vector arithmetic provides developers a means to apply these operations within PL/SQL blocks and functions without calling out to SQL.

View Documentation

Foreign Function Interface for JavaScript to Call PL/SQL Code Units

The Foreign Function Interface (FFI) allows JavaScript developers to use a more familiar syntax to call code units written in PL/SQL.

Rather than using PL/SQL blocks, it is possible to use native JavaScript constructs to interact with most code written in PL/SQL. Using the FFI provides JavaScript developers with a much improved experience by streamlining the process of integrating PL/SQL code into JavaScript functions.

View Documentation

Hybrid Vector Index for JSON

Hybrid Vector Indexes allow document retrieval by integrating full-text search capabilities with semantic vector search techniques, resulting in higher-quality search results. This powerful feature has now been extended to support JSON columns, offering greater flexibility in data indexing and querying.

Creating a Hybrid Vector Index on a JSON column offers a unified query API that enables users to execute various types of searches:

- Textual queries
- Vector similarity queries
- Hybrid queries that leverage both approaches

This versatile functionality allows users to:

- Easily customize the search experience
- Significantly enhance the quality and relevance of search results

View Documentation

In-Database Algorithms Support for VECTOR Data Type Predictors

This feature enables users to include one or more columns of `VECTOR` data type as predictors along with structured enterprise data to in-database machine learning algorithms.

Vector representations of unstructured data can be a powerful input to traditional machine learning algorithms. They enable efficient data processing on text and image data, helping to speed data-driven decision making. Providing vectors as input to machine learning models enables handling a broader class of use cases.

View Documentation

Included Columns in Neighbor Partition Vector Indexes

Included columns in vector indexes facilitate faster searches with attribute filters by incorporating non-vector columns within a Neighbor Partition Vector Index. This feature optimizes query execution by removing the need to access these columns from the base table.

Sophisticated workloads often combine business data search on relational columns with vector similarity search. Having included columns in Neighbor Partition Vector Indexes significantly enhances enterprise search capabilities by integrating attribute filters with vector-based similarity searches.

This integration facilitates efficient execution of complex queries by:

1. Directly evaluating attribute filters in tandem with vector searches
2. Eliminating the need for base table access through expensive join operations.

Moreover, when the index includes all columns required for a query as covering columns, data can be retrieved directly from the index, thereby accelerating query performance.

View Documentation

JSON to Duality Migrator: Multi-Collection Import API

`DBMS_JSON_DUALITY.IMPORT_ALL` is a PL/SQL procedure that is designed to import multiple document collections into a JSON-relational duality view.

While the existing `DBMS_JSON_DUALITY.IMPORT` procedure is limited to importing a single collection, the ability to import multiple collections in a single PL/SQL call simplifies the process and helps prevent constraint violation errors. The `IMPORT_ALL` procedure enables the efficient import of multiple document collections into a JSON-relational duality view.

View Documentation

JSON to Duality Migrator: Validation of Schema and Data

The PL/SQL functions `DBMS_JSON_DUALITY.VALIDATE_SCHEMA_REPORT` and `VALIDATE_IMPORT_REPORT` are provided to validate the relational schema and data that are created and imported by the JSON-To-Duality Migrator.

There is a growing need for functions that would validate the recommended relational schema. To address these needs, the validation APIs have been developed that:

- Help users verify accuracy of the recommended relational schema
- Confirms that no data is lost when they migrate their document collections to duality views.

View Documentation

Materialized Expression Columns

Expression columns, also known as virtual columns, are additional columns derived (computed) from existing columns. They can be persisted (materialized) on disk, complementing the existing default functionality of computing the results at runtime only.

Choosing between computation at runtime and computation at DML time for expression columns provides more flexibility in choosing the right approach for an application.

Materializing expression columns trades disk storage for the need to compute the same expression over and over again.

View Documentation

Move Data Chunks Between Shardspaces

With the Oracle Globally Distributed Database composite sharding method, data is organized into different shardspaces, allowing you to differentiate subsets of data; however, any automatic chunk moves for load balancing occurred within a shardspace. With this release, Oracle Globally Distributed Database provides you with the ability to move data chunks from an existing shardspace to another shardspace.

This feature makes it possible to move data between existing shardspaces, or move some data to a newly added shardspace. You can arrange sharded data for your new business needs, such as providing new levels of services or resources for certain customers, or move customers from one class of service to another while maintaining regional data sovereignty.

View Documentation

PL/SQL BINARY Vector Support

PL/SQL supports `BINARY` as a new dimension format for the vector type, in line with SQL.

`BINARY` vectors are frequently used to represent whether some entity, such as a textual document, does or does not include certain features, list words, or terms. The advantages of the `BINARY` format are two-fold. The storage footprint of vectors can be reduced 32X compared to the default `FLOAT32` vectors and distance computations on `BINARY` vectors are up to 40X faster.

Support for `BINARY` vectors means that PL/SQL is able to handle a binary vector in the same way that it supports a vector of any other dimension format.

View Documentation

PL/SQL JACCARD Distance Support

The `JACCARD` distance metric is now supported in PL/SQL, which provides a similarity measure as a value from 0 to 1 between two `BINARY` vectors.

Jaccard distance is a common similarity measure for binary vectors. Support for `JACCARD` as a new metric in the PL/SQL `VECTOR_DISTANCE` operator means that PL/SQL code can make use of the `JACCARD` distance metric from within PL/SQL instead of calling out into SQL.

View Documentation

SQL Time Bucketing

Time bucketing is a common operation when processing time series or event streaming data where a series of data points within an arbitrarily defined time window need to be mapped to a specific fixed time interval (bucket) for aggregated analysis.

With the new SQL operator `TIME_BUCKET`, Oracle provides native and performant support for time bucketing of time-based data for `DATETIMES`.

Providing a native SQL operator for common fixed-time interval bucketing for time series data significantly simplifies application development and data analysis of such information. In addition to simpler and less error prone code, the native operator increases the performance of time series analysis.

View Documentation

Sharding Support for AI Vector Search

With Globally Distributed Database support for Vector Search, tables containing vectors are automatically distributed and replicated across a pool of Oracle databases that share no hardware. Similarity searches are automatically parallelized across shards or directed to a specific shard if the sharding key is provided.

Globally Distributed Database AI Vector Search offers several benefits, including greater scalability by allowing vectors to be distributed across multiple machines, improved performance by parallelizing vector searches across shards, and improved data resilience because if one shard goes down, the other shards can continue to operate. It also allows vector search to be deployed as part of a distributed database, where a single logical database is distributed over multiple geographies.

View Documentation

Smallfile Tablespace Shrink

This feature supplies the capability to reliably shrink a smallfile tablespace.

In earlier releases, organizations may find the datafiles of a smallfile tablespace grow larger despite the actual used space being much smaller. This can happen after a user drops segments or objects in the tablespace, but depending on where data was located in the datafiles, users were not always able to use datafile resize to recover the freed space.

By using Smallfile Tablespace Shrink, you can now expect your smallfile tablespace size to be close to the sum of the size of all segments,and objects in that tablespace. You can now reliably shrink a smallfile tablespace, which means improved storage optimization and reduced storage costs.

View Documentation

Support for Image Transformer Models with AI Vector Search Using the In-Database ONNX Runtime

This feature enables importing and using image transformer models with the in-database ONNX Runtime engine available in Oracle Database 23ai. Image transformer models must be in ONNX format and include the required image decoding and preprocessing as part of the ONNX pipeline.

The in-database ONNX Runtime engine with imported transformers enables you to vectorize text and image data directly in the database, eliminating the need to provision, configure, and maintain a separate environment to generate embeddings. Further, it eliminates the need to move data from the database to a separate environment and return vectors to the database. This enhancement expands the set of use cases that you can develop using Oracle Database to include image data. This feature is integrated for use with Oracle AI Vector Search for semantic similarity search.

View Documentation

Vector Data Type Support in External Tables

Vector data stored outside the database (on a file system or cloud object store) can be easily accessed to perform similarity searches using external tables.

Vector embeddings created outside the database can be loaded quickly and easily using an external table and standard SQL. It is also possible to run similarity searches on vector embeddings stored outside the database and seamlessly combine those searches with data that is stored inside the database.

View Documentation

Release Update 23.8 Features

Additional Predicate Support with Hybrid Vector Search

Hybrid vector search queries can now have additional `WHERE` clause predicates on columns other than the indexed columns.

Hybrid vector search combines vector-distance and text-based search in a single query. There are situations where it is beneficial to add an additional filter predicate on columns that are not covered by the vector or text-based indexes. The `FILTER_BY` field provides a method to supply additional filter predicates using standard SQL operators.

View Documentation

Client Support for SPARSE Vectors

Sparse vectors are now supported by JDBC and OPD.NET client drivers.

Sparse vectors are vectors that typically have a large number of dimensions, but only a few of those dimensions have non-zero values. Because sparse vectors only store non-zero values, their use can improve efficiency and save storage space. ODP.NET and JDBC now supports sparse vector types enabling efficient interaction with sparse vector data through these client drivers.

View Documentation

Dynamic Statistics for PL/SQL Functions

Dynamic statistics support has been enhanced for PL/SQL functions used in SQL `WHERE` clauses and `TABLE` functions. Long-running PL/SQL functions can increase SQL parse times if they are used with dynamic statistics, so global-level and fine-grained controls are provided to configure which functions should be included or excluded.

Cardinality can be difficult to estimate when optimizing SQL statements containing PL/SQL functions. This can lead to poor SQL execution plans and poor SQL performance. The ability to control and use dynamic statistics with PL/SQL functions enables the optimizer to find better execution plans, which leads to improved overall database performance.

View Documentation

Elastic Vector Memory Management

The vector memory pool can be configured to dynamically grow or shrink to accomodate new or changing HNSW indexes.

HNSW vector indexes reside in the Vector Memory Pool in the Database's SGA. Creating an HNSW Index is dependent upon the pool being adequately sized to accommodate the index, and subsequent changes to the underlying base-table can result in the size of the HNSW index changing. With Elastic Vector Memory Management, the pool automatically resizes to accommodate all HNSW Indexes dynamically.

View Documentation

Included Columns Support for JSON, BLOB and CLOB Data Types

Included Columns in IVF (Neighbor Partition) vector indexes can now be of type `JSON`, `BLOB` or `CLOB`.

Included Columns permit additional non-vector columns in a base table to be stored in an IVF (Neighbor Partition) vector index. By storing the extra columns in the index, the query execution no longer needs an additional table access to retrieve the underlying columns from the base table.

View Documentation

JSON Type Modifier Enhancements: Data Size and Array Specifications

You can specify the maximum size in bytes for a JSON-type column.

For type modifier ARRAY, you can specify:

- A (single) scalar type for all of the array elements
- Whether array elements can be JSON null
- The maximum number of array elements
- Whether to store array elements sorted in ascending order

Specifying a size limit for a JSON-type column and constraining array size, element type, and storage order can lead to more efficient handling of the stored JSON data.

View Documentation

JSON to Duality Migrator: Hints Configuration Field

The hints configuration field for duality views can map primary-key columns to subobject fields, map SQL data types to fields, and disable sharing of data underlying a subobject.

You can obtain more control over the mapping between the documents supported by duality view and their underlying relational data:

1. Which document fields should correspond to primary-key columns
2. Which SQL data types should underlie particular fields
3. Whether to normalize particular objects in a document

This feature enhances the ability of developers to leverage JSON documents for data access using the highly efficient relational data storage model, without compromising simplicity or efficiency.

View Documentation

JSON_TEXTCONTAINS and JSON_EXISTS Support in DBMS_HYBRID_VECTOR.SEARCH

The `DBMS_HYBRID_VECTOR.SEARCH` API now supports a way for users to specify optional `JSON_EXISTS` and `JSON_TEXTCONTAINS` clauses as part of their hybrid search.

Writing complex queries requires the ability to further constrain hybrid (semantic and textual) search by `JSON_EXISTS` and `JSON_TEXTCONTAINS` when working with complex JSON data. This feature lets you design sophisticated applications that better satisfy users' search criteria.

View Documentation

Oracle Database Cloud Backup Module for Azure Blob Storage

The Oracle Database Cloud Backup Module for Azure Blob Storage enables Oracle Database to send backups to and recover from Microsoft Azure Blob Storage. This backup module is compatible with Oracle Databases deployed on-premises or on Azure cloud. Support for writing backups to Azure Blob Storage for Oracle-managed database services is not available. Database administrators can use RMAN commands, RMAN scripts, and Oracle Enterprise Manager to perform backup and recovery operations with Azure Blob Storage.

You can run Oracle Databases in many different locations, and the backup module for Azure Blob Storage provides on-premises and Azure users with more flexibility for backup storage locations.

View Documentation

Restricted Execution Contexts for In-Database JavaScript

The `PURE` option can be specified on Multilingual Engine (MLE) environments and JavaScript inline call specifications to create restricted JavaScript contexts in which interaction with database functionality is not possible.

Creating restricted JavaScript execution contexts using the `PURE` option gives you a convenient way to limit capabilities of JavaScript code running in the database. A JavaScript program executing in a restricted context is guaranteed not to modify database tables or use PL/SQL packages, regardless of database privileges currently in effect. Common data processing use cases with user-defined functions require only computations on function inputs. In these scenarios, restricted contexts provide a safety net, prohibiting unwanted database modifications—for example, when using third-party or open-source JavaScript libraries.

In the context of AI Vector Search, JavaScript functions can be used as user-defined vector distance metrics when creating an HNSW index. Only JavaScript functions marked with the `PURE` option are allowed on HNSW indexes.

View Documentation

Sparse Vector Support in PL/SQL

Sparse vectors are now supported in PL/SQL.

Sparse vectors are vectors that typically have a large number of dimensions, but only a few of those dimensions have non-zero values. Because sparse vectors only store non-zero values, their use can improve efficiency and save storage space. Native support in PL/SQL allows sparse vectors to be created and used directly from within PL/SQL.

View Documentation

User-Defined Vector Distance Functions

AI vector search supports custom, user-defined distance metrics. Proprietary and/or domain-specific distance metrics can be used in addition to the standard built-in distance metrics.

Vector search operations are often based on standard distance metrics such as Euclidean, Cosine, and Dot Product. However, there are situations where domain-specific or proprietary metrics may be required. User-defined vector distance functions allow users to create their own custom metrics using JavaScript functions.

View Documentation