Changes in This Release for Oracle Database SQL Language Reference

This preface contains:

Changes in Oracle Database Release 23ai

New Features

The following features are new in Release 23ai:

Vector Utility API

The Vector Utility API provides a SQL function `VECTOR_CHUNKS` which processes text into pieces (chunks) in preparation for the generation of embeddings to be used with a vector index. The API is configurable in terms of size of chunks and rules for splitting chunks.

Support for ONNX-Format Models as First-Class Database Objects

The Open Neural Network Exchange (ONNX) is an open format to represent machine learning models. It faciliates the exchange of models between systems and is supported by an ONNX runtime environment that enables using models for scoring/inference.

You can import ONNX-format models to Oracle Database for the machine learning techniques classification, regression, clustering, and embeddings.

The models will be imported as first-class `MINING MODEL` objects in your schema. Inference can be done using the family of OML scoring operators, including `PREDICTION`, `CLUSTER`, and `VECTOR_EMBEDDING`.

Vector Data Type

This feature provides a built-in `VECTOR` data type that enables vector similarity searches within the database.

With a built-in `VECTOR` data type, you can run run AI-powered vector similarity searches within the database instead of having to move business data to a separate vector database. Avoiding data movement reduces complexity, improves security, and enables searches on current data. You also can run far more powerful searches with Oracle AI Vector Search by combining sophisticated business data searches with AI vector similarity search using simple, intuitive SQL and the full power of the converged database - JSON, Graph, Text, Spatial, Relational and Vector - all within a single query.

Support of Vector Data type in JSON Type

This functionality extends the standard JSON scalar types, to include the new Vector data type. It is fully supported by all Oracle JSON constructs, and a vector scalar JSON value is convertible to/from a JSON array of numbers.

Embedding vector values in JSON-type data is important for interoperability between SQL values and JSON values. For example, a table with a `VECTOR` column can be exposed in JSON data without a loss of data-type information allowing developers to create the next generation of AI applications.

Vector Indexes

SQL Support for Boolean Data Type

Oracle Database now supports the `BOOLEAN` data type in compliance with the ISO SQL standard.

With the `BOOLEAN` data type you can store `TRUE` and `FALSE` values inside tables use boolean expressions in SQL statements.

Native Representation of Graphs in Oracle Database

Oracle Database now has native support for property graph data structures and graph queries.

Property graphs provide an intuitive way to find direct or indirect dependencies in data elements and extract insights from these relationships. The enterprise-grade manageability, security features, and performance features of Oracle Database are extended to property graphs. Developers can easily build graph applications using existing tools, languages, and development frameworks. They can use graphs in conjunction with transactional data, JSON, Spatial, and other data types.

Support for the ISO/IEC SQL Property Graph Queries (SQL/PGQ) Standard

The ISO SQL standard has been extended to include comprehensive support for property graph queries and creating property graphs in SQL. Oracle is among the first commercial software products to support this standard.

Developers can easily build graph applications with SQL using existing SQL development tools and frameworks. Support of the ISO SQL standard allows for greater code portability and reduces the risk of application lock-in.

Direct Joins for UPDATE and DELETE Statements

Join the target table in `UPDATE` and `DELETE` statements to other tables using the `FROM` clause. These other tables can limit the rows changed or be the source of new values. Direct joins make it easier to write SQL to change and delete data.

Multilingual Engine Module Calls

Multilingual Engine (MLE) Module Calls allow you to invoke JavaScript functions stored in modules from SQL and PL/SQL. Call Specifications written in PL/SQL link JavaScript to PL/SQL code units.

DEFAULT ON NULL for UPDATE Statements

You can define columns as `DEFAULT ON NULL` for update operations, which was previously only possible for insert operations. Columns specified as `DEFAULT ON NULL` are automatically updated to the specific default value when an update operation tries to update a value to `NULL`.

GROUP BY Column Alias or Position

You can now use column alias or `SELECT` item position in `GROUP BY`, `GROUP BY CUBE`, `GROUP BY ROLLUP`, and `GROUP BY GROUPING SETS` clauses. Additionally, the `HAVING` clause supports column aliases. These enhancements make it easier to write `GROUP BY` and `HAVING` clauses. It can make SQL queries much more readable and maintainable while providing better SQL code portability.

SELECT Without FROM Clause

You can now run `SELECT` expression-only queries without a `FROM` clause. This new feature improves SQL code portability and ease of use.

SQL UPDATE RETURN Clause Enhancements

The `RETURNING INTO` clause for `INSERT`, `UPDATE`, and `DELETE` statements are enhanced to report old and new values affected by the respective statement. This allows developers to use the same logic for each of these DML types to obtain values before and after statement execution. Old and new values are valid only for `UPDATE` statements. `INSERT` statements do not report old values and `DELETE` statements do not report new values.

Data Use Case Domains

A data use case domain is a dictionary object that belongs to a schema and encapsulates a set of optional properties and constraints for common values, such as credit card numbers or email addresses. After you define a data use case domain, you can define table columns to be associated with that domain, thereby explicitly applying the domain's optional properties and constraints to those columns.

With data use case domains, you can define how you intend to use data centrally. This make it easier to ensure you handle values consistently across applications and improve data quality.

DBMS Blockchain Versions

The blockchain table row version feature allows you to have multiple historical versions of a row that is maintained within a blockchain table corresponding to a set of user-defined columns. A view `bctable_last$` on top of the blockchain table allows you to see just the latest version of a row. This feature allows you to guarantee row versioning when using tamper-resistant blockchain tables in your application.

CEIL FLOOR for DATE, TIMESTAMP, and INTERVAL Types

You can now pass `DATE`, `TIMESTAMP`, and `INTERVAL` values to the `CEIL` and `FLOOR` functions. These functions include an optional second argument to specify a rounding unit. You can also pass `INTERVAL` values to `ROUND` and `TRUNC` functions.

These functions make it easy to find the upper and lower bounds for date and time values for a specified unit.

IF [NOT] EXISTS Syntax Support

DDL object creation, modification, and deletion now support the `IF EXISTS` and `IF NOT EXISTS` syntax modifiers. This enables you to control whether an error should be raised if a given object exists or does not exist.

Schema Annotations

Annotations help you use database objects in the same way across all applications. This simplifies development and improves data quality. Annotations enable you to store and retrieve metadata about database objects. These are name-value pairs or simply a name. These are freeform text fields applications can use to customize business logic or user interfaces.

JSON-Relational Duality View

JSON Relational Duality Views are fully updatable JSON views over relational data. Data is still stored in relational tables in a highly efficient normalized format but can be accessed by applications in the form of JSON documents.

Deprecated Features

The following features are deprecated since Release 23, and may be desupported in a future release:

Starting from Oracle Database Release 23, the `GOST256` and `SEED128` encryption algorithms are deprecated and no longer available for new encryption keys. Oracle recommends that you use the stronger `AES256` or `ARIA256` encryption algorithms.

Desupported Features

The following features are desupported in Oracle Database Release 23:

-

For a full list of desupported features for Release 23, please see the *Oracle Database Upgrade Guide*.