Changes in This Release for Oracle Database Development Guide

This is a summary of important changes in Oracle Database 23ai for *Oracle Database Development Guide*.

New Features in 23ai

The following are the new features in *Oracle Database Development Guide* for Oracle Database Release 23ai.

Blockchain Table Log History

This feature allows changes to one or more regular user tables to be tracked in a blockchain table, which is maintained by Oracle Database as part of the Flashback Data Archive.

See Protecting Flashback Archive Data.

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.

See Data Use Case Domains.

Flashback Time Travel Enhancements

Flashback Time Travel can automatically track and archive transactional changes to tables. Flashback Time Travel creates archives of the changes made to the rows of a table and stores the changes in history tables. It also maintains a history of the evolution of the table's schema. By maintaining the history of the transactional changes to a table and its schema, Flashback Time Travel enables you to perform operations, such as Flashback Query `AS OF` and `VERSIONS`), on the table to view the history of the changes made during transaction time.

See Using Flashback Time Travel.

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.

The `IF [NOT] EXISTS` syntax can simplify error handling in scripts and by applications.

See Using `IF EXISTS` and `IF NOT EXISTS`.

Implicit Connection Pooling for Database Resident Connection Pooling

This feature enables the automatic assignment of Database Resident Connection Pooling (DRCP) servers to and from an application connection at runtime when the application starts and finishes database operations, even if the application does not explicitly close the connection.

This feature can provide better scalability and efficient usage of database resources for applications that do not use application connection pooling.

See Implicit Connection Pooling.

Lock-Free Reservation

Lock-free Reservation enables concurrent transactions to proceed without being blocked on updates of heavily updated rows. Lock-free reservations are held on the rows instead of locking them. Lock-free Reservation verifies if the updates can succeed and defers the updates until the transaction commit time.

Lock-free Reservation improves the end user experience and concurrency in transactions.

See Lock-Free Reservation.

Multilingual Engine JavaScript Modules and Environments

Multilingual Engine (MLE) Modules and Environments allow JavaScript code to persist and be managed in the database. Call specifications provide a means to call JavaScript functions from an MLE module, anywhere that you can call PL/SQL functions.

The introduction of JavaScript Modules and Environments as schema objects in Oracle Database allows developers to follow established and well-known workflows used in client-side JavaScript development. Complex projects can be broken down into smaller, more manageable pieces that can be worked on independently by team members.

See Overview of JavaScript.

Multiple Named Pools for DRCP

Database Resident Connection Pooling (DRCP) now supports multiple named pools. The `DBMS_CONNECTION_POOL.ADD_POOL()` and `DBMS_CONNECTION_POOL.REMOVE_POOL()` procedures are added to the `DBMS_CONNECTION_POOL` package. Oracle Net connection string syntax is enhanced, so you can specify a pool name for each connection. Existing procedures can be used to start, stop, or configure named pools.

Having multiple pools allows finer control on the DRCP pool usage. It helps prevent situations where some applications dominate the use of a single pool.

See Using Multi-pool DRCP.

Precheckable Constraints using JSON Schema

A check constraint can be checked outside the database. For this, you mark the check constraint with the `PRECHECK` keyword. You can create a JSON Schema document from a precheckable check constraint. This means that data can be checked for validity outside of the database using an external JSON Schema validator.

A JSON schema can describe rules that define valid data (rules that correspond to the check constraint). This lets you use the database as a central repository of business rules.

A JSON schema can be used to check data (in application code, for instance) prior to sending it to the database, which can provide earlier detection of invalid data. This provides more resilient applications and reduces potential system downtime and the need to clean up 'bad' data.

See Using PRECHECK to Pre-validate a CHECK Constraint.

Reset Database Session State

The reset session state feature clears the session state set by the application when the request ends. The `RESET_STATE` database service attribute cleans up dirty sessions so that the applications cannot see the state of these sessions. This feature applies to all applications that connect to the database using database services.

The `RESET_STATE` feature enables you to clean the session state at the end of each request so that the database developers do not have to clean the session state manually. By using this feature, you can ensure that there are no data leaks from a previous session.

See Reset Database Session State to Prevent Application State Leaks, Use RESET_STATE.

Saga APIs Using Oracle Saga Framework

Oracle Saga APIs are implemented in the database and provide a framework to implement transactional semantics for microservices built with Oracle Database. The orchestrator Saga framework provides a way to maintain atomic data consistency across microservices.

Sagas are concurrent and execute local transactions in each participant database making it more efficient than distributed ACID transactions, thereby simplifying application code and increasing developer productivity.

See Developing Applications with Sagas.

Schema Annotations

Schema 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.

Schema annotations help you use database objects in the same way across all applications. This simplifies development and improves data quality.

See Schema Annotations.

Sessionless Transactions

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, allowing it to be reused by other transactions, and therefore effectively multiplexing transactions across sessions or connections.

See Developing Applications with Sessionless Transactions

Shut Down Connection Draining for DRCP

A new, optional `DRAINTIME` argument to `DBMS_CONNECTION_POOL.STOP_POOL()` allows active DRCP pools to be closed after a specified connection drain time, or be closed immediately without waiting for connections to be idle. This feature gives DBAs better control over DRCP usage and configuration.

See Shut Down Connection Draining for DRCP.

Table DDL Change Notification

Applications can now be notified when DDL changes occur on tables.

Applications that need or want to be aware of table metadata can now be notified of DDL changes rather than having to continuously poll for them.

See Table DDL Change Notification.

Deprecated Features

This section lists the deprecated features in Oracle Database Release 23ai for *Database Development Guide*.

Oracle recommends that you do not use deprecated features/values in new applications. Support for deprecated features is for backward compatibility only.

Deprecation of Traditional Audit Initialization Parameters

With the desupport of creating and modifying traditional audit policies with Oracle Database 23, the traditional audit parameters are deprecated.
Traditional audit policies are available after upgrading, but the initialization parameters associated with those policies are deprecated. These deprecated parameters include the following:

- `AUDIT_TRAIL`
- `AUDIT_SYS_OPERATIONS`
- `AUDIT_FILE_DEST`
- `AUDIT_SYSLOG_LEVEL`

Oracle recommends that you migrate to unified auditing as soon as possible, because these traditional audit parameters can be removed in a future database release.

Deprecation of Oracle JDBC-OCI (Thick) Driver

The Oracle ODBC Oracle Call Interface (OCI) Type 2 client driver (also known as a "thick" driver) is deprecated in Oracle Database 23ai.
Most Java applications that use Open Database Connectivity (ODBC) with Oracle JDBC drivers use the Thin driver. To enable Oracle to allocate resources to better address customer requirements, Oracle is deprecating the JDBC-OCI driver.

Deprecation of SQLJ

Starting with Oracle Database 23ai, the SQLJ method of embedding SQL statements in Java code is deprecated.
In place of SQLJ, Oracle recommends that you directly use the Java Database Connectivity (JDBC) APIs (dynamic SQL, prepared statements or PL/SQL blocks).