35
Table DDL Change Notification

This chapter explains how to use Table DDL Change Notification for receiving notifications about DDL changes in database tables.

Topics:

35.1 Overview of Table DDL Change Notification

Table DDL Change Notification provides Oracle Call Interface (OCI) clients an efficient mechanism to subscribe for DDL notifications on tables of interest. Table DDL Change Notification ensures that applications are notified when DDL statements make changes to a table. Changes are captured as DDL events and these events are processed asynchronously without blocking the user DML activity.

Applications seeking table metadata can use Table DDL Change Notification instead of continuously polling for DDL changes, and benefit from reduced network round trips. Table DDL Change Notification is also useful for OCI clients of the database that cache table metadata in the middle tier to avoid round trips to the database.

An OCI client can register to receive table DDL change notification for any of the following:

- A list of tables
- A list of schemas
- All tables in a database

Table DDL Change Notification is included in Oracle Database 23ai, and later releases.

35.2 Table DDL Change Notification Terminology

OCI Client

An OCI client (client, client application) is a client program that calls the APIs in Oracle Call Interface (OCI).

EMON

Notification systems use the Event monitor (EMON) server pool to publish notifications to OCI clients.

Notification

A notification is a message describing the table DDL event that is sent to an OCI client. A notification does not contain the entire DDL statement but provides the table name and type of DDL operation.

Subscription

A subscription defines a channel that the clients can use to receive a particular notification.

Event

An event is a message published on a subscription, describing the DDL action on a table.

Registration

A registration represents a client that wants notification for a particular subscription topic.

35.3 Benefits of Table DDL Change Notification

Table DDL Change Notification works without affecting the application activity in the database.

The following are some of the benefits of using Table DDL Change Notification:

- Notifications can be enabled on highly active tables.
- Registering and event processing do not block the DDL activity on a table.
- Registrations and notifications are processed with minimal overhead on DML activity and without invalidating table cursors.
- Registrations are processed quickly while allowing concurrent registrations of other client applications.
- Client applications can dynamically subscribe to any number of additional tables or schemas.

35.4 Features of Table DDL Change Notification

The following are the features of Table DDL Change Notification:

- DDL events are processed asynchronously without blocking user DML queries.
- The client application can choose to include optional events, such as Partition Maintenance Operations (PMOP) or truncate, which are not available for event registration by default.
- DDL notifications are staged in the System Global Area (SGA). If an instance restarts, unprocessed notifications in SGA are lost. Persistent events are not supported.
- On failover of the database, any undelivered events in SGA are lost. The client caches must be invalidated on failover to avoid reconciliation issues in the table metadata due to lost notifications.

- For logical standby and physical standby, the OCI client must register again over the new primary database to resume notification.

35.5 Using Table DDL Change Notification

Here is how the notification process works with Table DDL Change Notification:

- A client application registers for receiving notification about the DDL events on a table as a part of a subscription.

  > **See Also:**
  >
  > Registering for Table DDL Change Notification for more information about registrations.

- A DDL event is generated when any DDL transaction commits to the table.
- The OCI EMON processes the DDL event and notifies the event to the client application in a native OCI format (`OCI_DTYPE_DDL_EVENT`).

  See DDL Event Payload in the following section for the information contained in a DDL event.
- The client application receives the DDL event and invokes a user callback to process the event.

  See Registering Client Callback for DDL Notification in the following section for more information about the user callback process.

DDL Event Payload

The event payload describes the DDL event that is notified to the client application. The attributes of the event include:

- Operation type: `CREATE`, `ALTER`, `DROP`, `TRUNCATE`, `RENAME`, or `FLASHBACK`
- Object name: The base table name
- Object type: The object affected by the DDL transaction, for instance, partition, index, or table
- Database name
- SCN: The commit SCN of the DDL transaction (to order events within the database).
- UTC Modification time: Useful to correlate events across databases

Registering Client Callback for DDL Notification

- A client uses the following API to register a user callback and process events:

  `OCISubscriptionRegister(subhp)`: With the `OCI_SUBSCR_CQ_QOS_DDL_NTFN` DDL notification QoS and the `DBCHANGE` namespace.
- This starts a client thread to asynchronously invoke the user callback on an event.
- The subscription is assigned a unique (system generated) registration ID (`regid`) and subscription name `DDNF<regid>`.

> **See Also:**
>
> OCISubscriptionRegister() for more information about the user callback API.

Use Case: GoldenGate Replicat

Here is a use case that has Oracle GoldenGate Replicat as the client:

Oracle GoldenGate Replicat applies source DML and DDL changes at the target database during logical replication. During an application upgrade, in many instances, the target table structure is modified prior to making the corresponding changes to the source database. In such cases, DDL replication cannot be used to synchronize and reconcile the metadata between the source and the target database.

For example; suppose a source table called `FA1.AR_SALES_TAX` is being replicated to the target database. Replicat loads the metadata of `FA1.AR_SALES_TAX` to reconstruct the SQL statement `INSERT into FA1.AR_SALES_TAX values(..)`. The relevant metadata includes information, such as column names, column types, and whether the table is compressed. Since the table metadata rarely changes, Replicat caches the data instead of querying the target database repeatedly. When the table metadata is first cached, Replicat registers for table DDL events of `FA1.AR_SALES_TAX` on the target database.

Take a scenario where the `FA1.AR_SALES_TAX` table is modified on the target database directly (and not on the source). For example, additional columns are added to the target table without being added to the source database. Replicat is notified with the table name and the operation (for instance, `ALTER`). When Replicat receives a new transaction to apply, it can refresh its stale metadata based on the DDL events received and can replicate inserts into the newly added columns successfully.

35.6 Registering for Table DDL Change Notification

A client can register for DDL change notifications at the table level or the schema level (for schema-wide table DDL events).

To register for notifications, you must fulfill the following conditions:

- You must be a non-`SYS` user.
- You must have `SELECT ANY TABLE` privileges.
- You must have `CHANGE NOTIFICATION` privileges.
- Set up a TCP or IPC listener for OCI DDL Notification client connections, and set the `local_listener` parameter of the ROOT container to this listener. The `local_listener` parameter can be set in the initialization parameter file (`init.ora`), or as follows:

```
ALTER SYSTEM SET LOCAL_LISTENER=listener_name;
ALTER SYSTEM REGISTER;
```

> **Note:**
>
> If a user loses the required privileges, the user's table or schema-level registrations are implicitly unregistered.

35.6.1 Table-level Registration

To register for notifications at the table level, use the `OCIDdlEventRegister()` function.

> **See Also:**
>
> OCIDdlEventRegister() in *Oracle Call Interface Programmer's Guide* for more information about the `OCIDdlEventRegister()` function and examples.

Here are some important points to note while registering for table-level notifications:

- You can register to receive notifications on any number of tables.
- You can dynamically add or remove tables from the existing subscription.
- Adding or removing tables from an existing subscription is an idempotent operation.
- A table is implicitly unregistered when the table is dropped or renamed.
- Only existing tables can be registered.

35.6.2 Schema-level Registration

To register for table DDL notifications at the schema level, use the `OCIDdlEventRegister()` function.

> **See Also:**
>
> OCIDdlEventRegister() in *Oracle Call Interface Programmer's Guide* for more information about the `OCIDdlEventRegister()` function and examples.

Here are some important points to note while registering for schema-level notifications:

- You can register to receive notifications on any number of schema names.
- You can dynamically add or remove schemas from the existing subscription.
- Adding or removing schemas from an existing subscription is an idempotent operation.
- A schema is implicitly unregistered when the schema is dropped.
- Only existing schemas can be registered.

35.7 Unregistering for Table DDL Change Notifications

A client can unregister previously registered tables or schema using the `OCIDdlEventUnregister()` function. The client can select the tables and schemas to remove from registration.

To unregister DDL notifications at the table or schema level, use the `OCIDdlEventUnregister()` function.

> **See Also:**
>
> OCIDdlEventUnregister() in *Oracle Call Interface Programmer's Guide* for more information about the `OCIDdlEventUnregister()` function and examples.

In the following cases, a table is automatically unregistered:

- The table or schema is dropped.
- A client process exits without unregistering.
- The table is renamed.
- The user who registered for notification loses the required privileges (`SELECT ANY TABLE` or `CHANGE NOTIFICATION`).

When a client application exits without unregistering, its unprocessed notifications are dropped when notification delivery fails. If the OCI client restarts, it needs to re-register to resume notifications.

35.8 Supported DDL Events and Commands

In general, space management operations on the table, partition or index are not supported.

Supported Events

The following DDL events are included for DDL notifications:

- Events on user tables and global temporary tables
- Events on special tables, such as AQ queues, blockchain tables, and materialized views
- Events on DDL statements that affect dependent objects, such as indexes and constraints of the base table.

The following DDL events have conditional support for DDL notifications:

- DDL can include filters and the client can include optional events, such as PMOP or `TRUNCATE` operations, which are not delivered by default.
- For multi-table DDL statements, such as foreign key updates and online redefinition, only tables that are registered for notification generate events.

The following DDL events are excluded from DDL notifications:

- Events on tables in Oracle-maintained schema and private or temporary tables
- DDL statements on triggers and packages
- Schema create and drop events

Supported commands

At the table level, the following commands are supported:

- `CREATE TABLE`
- `DROP TABLE`
- `TRUNCATE TABLE`
- `RENAME TABLE`

- `ALTER TABLE`

At the PMOP level, the following commands are supported:

- `CREATE TABLE PARTITION [SUBPARTITION]`
- `ALTER TABLE operation [PARTITION | SUBPARTITION]`

  Operation = `[ ADD | DROP | MODIFY | EXCHANGE | MERGE | RENAME | SPLIT | TRUNCATE ]`

At the constraint level, the following commands are supported:

`ALTER TABLE operation CONSTRAINT`

Operation = `[ADD | MODIFY | RENAME | DROP]`

At the Flashback level, the following commands are supported: `FLASHBACK TABLE`

At the index level, the following commands are supported:

- `CREATE INDEX`
- `ALTER INDEX`
- `DROP INDEX`

Unsupported Commands

At the table level, the following commands are not supported:

- `COMMENT`
- `SHRINK SEGMENT`
- `MOVE`

At the PMOP level, `MOVE` operations are not supported.

At the index level, space management operations like `REBUILD`, `COALESCE` and `SPLIT PARTITION` are not supported.

35.9 Monitoring Table DDL Change Notification

The registration identifier (ID) uniquely identifies a subscription. A client can register on mutliple tables or schemas in the database using the same subscription (meaning, registration ID). The following views can be queried by the registration ID to monitor table DDL events received for a subscription.

- DBA_DDL_REGS for information on all table, schema or database-level registrations.
- DBA_SUBSCR_REGISTRATIONS for more information about subscriptions created in the database.
- `V$SUBSCR_REGISTRATION_STATS` (V_SUBSCR_REGISTRATION_STATS) for notification statistics and diagnostic information about a subscription.