Oracle® Database

SQL Language Quick Reference

12*c* Release 1 (12.1)

E49206-12

November 2014

Oracle Database SQL Language Quick Reference, 12*c* Release 1 (12.1)

E49206-12

Primary Authors: Mary Beth Roeser, Simon Watt

Contents

Preface

This reference contains a complete description of the Structured Query Language (SQL) used to manage information in an Oracle Database. Oracle SQL is a superset of the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) SQL:2011 standard.

This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

The *Oracle Database SQL Language Quick Reference* is intended for all users of Oracle SQL.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database PL/SQL Language Reference* for information on PL/SQL, the procedural language extension to Oracle SQL
- *Pro*C/C++ Programmer's Guide*, *Oracle SQL*Module for Ada Programmer's Guide*, and the *Pro*COBOL Programmer's Guide* for detailed descriptions of Oracle embedded SQL

Many of the examples in this book use the sample schemas, which are installed by default when you select the Basic Installation option with an Oracle Database installation. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 SQL Statements

This chapter presents the syntax for Oracle SQL statements.

This chapter includes the following section:

- Syntax for SQL Statements

Syntax for SQL Statements

SQL statements are the means by which programs and users access data in an Oracle database.

The sections that follow show each SQL statement and its related syntax. Refer to Chapter 5, "Subclauses" for the syntax of the subclauses listed in the syntax for the statements.

> **See Also:** *Oracle Database SQL Language Reference* for detailed information about Oracle SQL

ADMINISTER KEY MANAGEMENT

```
ADMINISTER KEY MANAGEMENT
  { keystore_management_clauses
  | key_management_clauses
  | secret_management_clauses
  } ;
```

ALTER AUDIT POLICY

```
ALTER AUDIT POLICY policy
  [ ADD [ privilege_audit_clause ] [ action_audit_clause ] [ role_audit_clause ] ]
  [ DROP [ privilege_audit_clause ] [ action_audit_clause ] [ role_audit_clause ] ]
  [ CONDITION { DROP | 'audit_condition' EVALUATE PER { STATEMENT | SESSION | INSTANCE } } ]
;
```

ALTER CLUSTER

```
ALTER CLUSTER [ schema. ] cluster
  { physical_attributes_clause
  | SIZE size_clause
  | [ MODIFY PARTITION partition ] allocate_extent_clause
  | deallocate_unused_clause
  | { CACHE | NOCACHE }
  } ...
  [ parallel_clause ] ;
```

ALTER DATABASE

```
ALTER DATABASE [ database ]
  { startup_clauses
  | recovery_clauses
  | database_file_clauses
  | logfile_clauses
  | controlfile_clauses
  | standby_database_clauses
  | default_settings_clauses
  | instance_clauses
  | security_clause
  } ;
```

ALTER DATABASE LINK

```
ALTER DATABASE LINK dblink
  { CONNECT TO user IDENTIFIED BY password [ dblink_authentication ]
  | dblink_authentication
  };
```

ALTER DIMENSION

```
ALTER DIMENSION [ schema. ] dimension
  { ADD { level_clause
        | hierarchy_clause
        | attribute_clause
        | extended_attribute_clause
        }
  } ...
  |
  { DROP { LEVEL level [ RESTRICT | CASCADE ]
         | HIERARCHY hierarchy
         | ATTRIBUTE attribute [ LEVEL level [ COLUMN column ] ]...
         }
  } ...
  |
  COMPILE
  ;
```

ALTER DISKGROUP

```
ALTER DISKGROUP
  { diskgroup_name
      { { { add_disk_clause | drop_disk_clause }
            [, { add_disk_clause | drop_disk_clause } ]...
        | resize_disk_clause
        } [ rebalance_diskgroup_clause ]
      | replace_disk_clause
      | rename_disk_clause
      | disk_online_clause
      | disk_offline_clause
      | rebalance_diskgroup_clause
      | check_diskgroup_clause
      | diskgroup_template_clauses
      | diskgroup_directory_clauses
      | diskgroup_alias_clauses
      | diskgroup_volume_clauses
      | diskgroup_attributes
      | modify_diskgroup_file
      | drop_diskgroup_file_clause
      | usergroup_clauses
      | user_clauses
      | file_permissions_clause
      | file_owner_clause
      | scrub_clause
      }
    | { diskgroup_name [, diskgroup_name ] ...
      | ALL
      } { undrop_disk_clause
        | diskgroup_availability
        | enable_disable_volume
        }
  } ;
```

ALTER FLASHBACK ARCHIVE

```
ALTER FLASHBACK ARCHIVE flashback_archive
  { SET DEFAULT
  | { ADD | MODIFY } TABLESPACE tablespace [flashback_archive_quota]
  | REMOVE TABLESPACE tablespace_name
  | MODIFY RETENTION flashback_archive_retention
  | PURGE { ALL | BEFORE { SCN expr | TIMESTAMP expr } }
  | [NO] OPTIMIZE DATA
  };
```

ALTER FUNCTION

```
ALTER FUNCTION [ schema. ] function_name
{ function_compile_clause | { EDITIONABLE | NONEDITIONABLE } }
```

ALTER INDEX

```
ALTER INDEX [ schema. ]index
  { { deallocate_unused_clause
    | allocate_extent_clause
    | shrink_clause
    | parallel_clause
    | physical_attributes_clause
    | logging_clause
    | partial_index_clause
    } ...
  | rebuild_clause
  | PARAMETERS ( 'ODCI_parameters' )
             )
  | COMPILE
  | { ENABLE | DISABLE }
  | UNUSABLE [ ONLINE ]
  | VISIBLE | INVISIBLE
  | RENAME TO new_name
  | COALESCE [ CLEANUP ]
  | { MONITORING | NOMONITORING } USAGE
  | UPDATE BLOCK REFERENCES
  | alter_index_partitioning
  }
  ;
```

ALTER INDEXTYPE

```
ALTER INDEXTYPE [ schema. ] indextype
  { { ADD | DROP } [ schema. ] operator ( parameter_types )
      [ , { ADD | DROP } [schema. ] operator ( parameter_types ) ]... [ using_type_clause ]
  | COMPILE
  }
  [ WITH LOCAL [ RANGE ] PARTITION ] [ storage_table_clause ]
  ;
```

ALTER JAVA

```
ALTER JAVA
  { SOURCE | CLASS } [ schema. ]object_name
  [ RESOLVER
      ( ( match_string [, ] { schema_name | - } )... )
  ]
  { { COMPILE | RESOLVE }
  | invoker_rights_clause
  } ;
```

ALTER LIBRARY

```
ALTER LIBRARY [ schema. ] library_name
{ library_compile_clause | { EDITIONABLE | NONEDITIONABLE } }
```

ALTER MATERIALIZED VIEW

```
ALTER MATERIALIZED VIEW
  [ schema. ] materialized_view
  [ physical_attributes_clause
  | modify_mv_column_clause
  | table_compression
  | inmemory_alter_table_clause
  | LOB_storage_clause [, LOB_storage_clause ]...
  | modify_LOB_storage_clause [, modify_LOB_storage_clause ]...
  | alter_table_partitioning
  | parallel_clause
  | logging_clause
  | allocate_extent_clause
  | deallocate_unused_clause
  | shrink_clause
  | { CACHE | NOCACHE }
  ]
  [ alter_iot_clauses ]
  [ USING INDEX physical_attributes_clause ]
  [ MODIFY scoped_table_ref_constraint
  | alter_mv_refresh
  ]
  [ evaluation_edition_clause ]
  [ alter_query_rewrite_clause
  | COMPILE
  | CONSIDER FRESH
  ] ;
```

ALTER MATERIALIZED VIEW LOG

```
ALTER MATERIALIZED VIEW LOG [ FORCE ]
  ON [ schema. ]table
  [ physical_attributes_clause
  | add_mv_log_column_clause
  | alter_table_partitioning
  | parallel_clause
  | logging_clause
  | allocate_extent_clause
  | shrink_clause
  | move_mv_log_clause
  | { CACHE | NOCACHE }
  ] [ mv_log_augmentation ] [  mv_log_purge_clause ] [ for_refresh_clause ]
  ;
```

ALTER MATERIALIZED ZONEMAP

```
ALTER MATERIALIZED ZONEMAP [ schema. ] zonemap_name
  { alter_zonemap_attributes
  | zonemap_refresh_clause
  | { ENABLE | DISABLE } PRUNING
  | COMPILE
  | REBUILD
  | UNUSABLE
  } ;
```

ALTER OPERATOR

```
ALTER OPERATOR [ schema. ] operator
  { add_binding_clause
  | drop_binding_clause
  | COMPILE
  } ;
```

ALTER OUTLINE

```
ALTER OUTLINE [ PUBLIC | PRIVATE ] outline
  { REBUILD
  | RENAME TO new_outline_name
  | CHANGE CATEGORY TO new_category_name
  | { ENABLE | DISABLE }
  } ...
  ;
```

ALTER PACKAGE

```
ALTER PACKAGE [ schema. ] package_name
{ package_compile_clause | { EDITIONABLE | NONEDITIONABLE } }
```

ALTER PLUGGABLE DATABASE

```
ALTER PLUGGABLE DATABASE
  { pdb_unplug_clause
  | pdb_settings_clauses
  | pdb_datafile_clause
  | pdb_recovery_clauses
  | pdb_change_state
  | pdb_change_state_from_root
  } ;
```

ALTER PROCEDURE

```
ALTER PROCEDURE [ schema. ] procedure_name
{ procedure_compile_clause | { EDITIONABLE | NONEDITIONABLE } }
```

ALTER PROFILE

```
ALTER PROFILE profile LIMIT
  { resource_parameters | password_parameters } ...
  [ CONTAINER = { CURRENT | ALL } ] ;
```

ALTER RESOURCE COST

```
ALTER RESOURCE COST
  { { CPU_PER_SESSION
    | CONNECT_TIME
    | LOGICAL_READS_PER_SESSION
    | PRIVATE_SGA
    } integer
  } ...
  ;
```

ALTER ROLE

```
ALTER ROLE role
  { NOT IDENTIFIED
  | IDENTIFIED
      { BY password
      | USING [ schema. ] package
      | EXTERNALLY
      | GLOBALLY
      }
  }
  [ CONTAINER = { CURRENT | ALL } ] ;
```

ALTER ROLLBACK SEGMENT

```
ALTER ROLLBACK SEGMENT rollback_segment
  { ONLINE
  | OFFLINE
  | storage_clause
  | SHRINK [ TO size_clause ]
  };
```

ALTER SEQUENCE

```
ALTER SEQUENCE [ schema. ] sequence
  { INCREMENT BY integer
  | { MAXVALUE integer | NOMAXVALUE }
  | { MINVALUE integer | NOMINVALUE }
  | { CYCLE | NOCYCLE }
  | { CACHE integer | NOCACHE }
  | { ORDER | NOORDER }
  | { KEEP | NOKEEP }
  | { SESSION | GLOBAL }
  } ...
;
```

ALTER SESSION

```
ALTER SESSION
  { ADVISE { COMMIT | ROLLBACK | NOTHING }
  | CLOSE DATABASE LINK dblink
  | { ENABLE | DISABLE } COMMIT IN PROCEDURE
  | { ENABLE | DISABLE } GUARD
  | { ENABLE | DISABLE | FORCE } PARALLEL
    { DML | DDL | QUERY } [ PARALLEL integer ]
  | { ENABLE RESUMABLE [ TIMEOUT integer ] [ NAME string ]
    | DISABLE RESUMABLE
    }
  | SYNC WITH PRIMARY
  | alter_session_set_clause
  } ;
```

ALTER SYNONYM

```
ALTER [ PUBLIC ] SYNONYM [ schema. ] synonym
  { EDITIONABLE | NONEDITIONABLE | COMPILE } ;
```

ALTER SYSTEM

```
ALTER SYSTEM
  { archive_log_clause
  | checkpoint_clause
  | check_datafiles_clause
  | distributed_recov_clauses
  | FLUSH { SHARED_POOL | BUFFER_CACHE | REDO TO target_db_name [ [ NO ] CONFIRM APPLY ] }
  | end_session_clauses
  | SWITCH LOGFILE
  | { SUSPEND | RESUME }
  | quiesce_clauses
  | rolling_migration_clauses
  | rolling_patch_clauses
  | security_clauses
  | shutdown_dispatcher_clause
  | REGISTER
  | SET alter_system_set_clause
        [ alter_system_set_clause ]...
  | RESET alter_system_reset_clause
          [ alter_system_reset_clause ]...
  | RELOCATE CLIENT client_id
  } ;
```

ALTER TABLE

```
ALTER TABLE [ schema. ] table
  [ alter_table_properties
  | column_clauses
  | constraint_clauses
  | alter_table_partitioning
  | alter_external_table
  | move_table_clause
  | modify_opaque_type
  ]
  [ enable_disable_clause
  | { ENABLE | DISABLE } { TABLE LOCK | ALL TRIGGERS }
  ] ...
  ;
```

ALTER TABLESPACE

```
ALTER TABLESPACE tablespace
  { DEFAULT [ table_compression ] [ inmemory_clause ] [ storage_clause ]
  | MINIMUM EXTENT size_clause
  | RESIZE size_clause
  | COALESCE
  | SHRINK SPACE [ KEEP size_clause]
  | RENAME TO new_tablespace_name
  | { BEGIN | END } BACKUP
  | datafile_tempfile_clauses
  | tablespace_logging_clauses
  | tablespace_group_clause
  | tablespace_state_clauses
  | autoextend_clause
  | flashback_mode_clause
  | tablespace_retention_clause
  } ;
```

ALTER TRIGGER

```
ALTER TRIGGER [ schema. ] trigger_name
  { trigger_compile_clause
  | { ENABLE| DISABLE }
  | RENAME TO new_name
  | { EDITIONABLE | NONEDITIONABLE }
  } ;
```

ALTER TYPE

```
ALTER TYPE [ schema. ] type_name
{ alter_type_clause | { EDITIONABLE | NONEDITIONABLE } }
```

ALTER USER

```
ALTER USER
  { user
    { IDENTIFIED
      { BY password [ REPLACE old_password ]
      | EXTERNALLY [ AS 'certificate_DN' | AS 'kerberos_principal_name' ]
      | GLOBALLY [ AS '[directory_DN]' ]
      }
    | DEFAULT TABLESPACE tablespace
    | TEMPORARY TABLESPACE { tablespace | tablespace_group_name }
    | { QUOTA { size_clause
              | UNLIMITED
              } ON tablespace
      } ...
    | PROFILE profile
    | DEFAULT ROLE { role [, role ]...
                   | ALL [ EXCEPT role [, role ]... ]
                   | NONE
                   }
    | PASSWORD EXPIRE
    | ACCOUNT { LOCK | UNLOCK }
    | ENABLE EDITIONS [ FOR object_type [, object_type ]... ] [ FORCE ]
    | CONTAINER = { CURRENT | ALL }
    | container_data_clause
    } ...
  | user [, user ]... proxy_clause
  } ;
```

ALTER VIEW

```
ALTER VIEW [ schema. ] view
  { ADD out_of_line_constraint
  | MODIFY CONSTRAINT constraint
      { RELY | NORELY }
  | DROP { CONSTRAINT constraint
         | PRIMARY KEY
         | UNIQUE (column [, column ]...)
         }
  | COMPILE
  | { READ ONLY | READ WRITE }
  | { EDITIONABLE | NONEDITIONABLE }
  } ;
```

ANALYZE

```
ANALYZE
  { { TABLE [ schema. ] table
    | INDEX [ schema. ] index
    } [ partition_extension_clause ]
  | CLUSTER [ schema. ] cluster
  }
  { validation_clauses
  | LIST CHAINED ROWS [ into_clause ]
  | DELETE [ SYSTEM ] STATISTICS
  } ;
```

ASSOCIATE STATISTICS

```
ASSOCIATE STATISTICS WITH
  { column_association | function_association }
  [ storage_table_clause ] ;
```

AUDIT (Traditional Auditing)

```
AUDIT
  { audit_operation_clause [ auditing_by_clause | IN SESSION CURRENT ]
  | audit_schema_object_clause
  | NETWORK
  | DIRECT_PATH LOAD [ auditing_by_clause ]
  } [ BY { SESSION | ACCESS } ]
    [ WHENEVER [ NOT ] SUCCESSFUL ]
    [ CONTAINER = { CURRENT | ALL } ]
;
```

AUDIT (Unified Auditing)

```
AUDIT
  { POLICY policy
    [ { BY user [, user]... } | { EXCEPT user [, user]... } ]
    [ WHENEVER [ NOT ] SUCCESSFUL ]
  }
  |
  { CONTEXT NAMESPACE namespace ATTRIBUTES attribute [, attribute ]...
      [, CONTEXT NAMESPACE namespace ATTRIBUTES attribute [, attribute ]... ]...
    [ BY user [, user]... ]
  } ;
```

CALL

```
CALL
  { routine_clause
  | object_access_expression
  }
  [ INTO :host_variable
    [ [ INDICATOR ] :indicator_variable ] ] ;
```

COMMENT

```
COMMENT ON
  { AUDIT POLICY policy
  | COLUMN [ schema. ]
      { table. | view. | materialized_view. } column
  | EDITION edition_name
  | INDEXTYPE [ schema. ] indextype
  | MATERIALIZED VIEW materialized_view
  | MINING MODEL [ schema. ] model
  | OPERATOR [ schema. ] operator
  | TABLE [ schema. ] { table | view }
  }
  IS string ;
```

COMMIT

```
COMMIT [ WORK ]
  [ [ COMMENT string ]
    | [ WRITE [ WAIT | NOWAIT ] [ IMMEDIATE | BATCH ]
    ]
  | FORCE { string [, integer ]
          | CORRUPT_XID string
          | CORRUPT_XID_ALL
          }
  ] ;
```

CREATE AUDIT POLICY

```
CREATE AUDIT POLICY policy
  [ privilege_audit_clause ] [ action_audit_clause ] [ role_audit_clause ]
  [ WHEN 'audit_condition' EVALUATE PER { STATEMENT | SESSION | INSTANCE } ]
  [ CONTAINER = { ALL | CURRENT } ] ;
```

CREATE CLUSTER

```
CREATE CLUSTER [ schema. ] cluster
  (column datatype [ SORT ]
    [, column datatype [ SORT ] ]...
  )
  [ { physical_attributes_clause
    | SIZE size_clause
    | TABLESPACE tablespace
    | { INDEX
      | [ SINGLE TABLE ]
        HASHKEYS integer [ HASH IS expr ]
      }
    }...
  ]
  [ parallel_clause ]
  [ NOROWDEPENDENCIES | ROWDEPENDENCIES ]
  [ CACHE | NOCACHE ] [ cluster_range_partitions ] ;
```

CREATE CONTEXT

```
CREATE [ OR REPLACE ] CONTEXT namespace
  USING [ schema. ] package
  [ INITIALIZED { EXTERNALLY | GLOBALLY }
  | ACCESSED GLOBALLY
  ] ;
```

CREATE CONTROLFILE

```
CREATE CONTROLFILE
  [ REUSE ] [ SET ] DATABASE database
  [ logfile_clause ]
  { RESETLOGS | NORESETLOGS }
  [ DATAFILE file_specification
           [, file_specification ]... ]
  [ MAXLOGFILES integer
  | MAXLOGMEMBERS integer
  | MAXLOGHISTORY integer
  | MAXDATAFILES integer
  | MAXINSTANCES integer
  | { ARCHIVELOG | NOARCHIVELOG }
  | FORCE LOGGING
  ]...
  [ character_set_clause ] ;
```

CREATE DATABASE

```
CREATE DATABASE [ database ]
  { USER SYS IDENTIFIED BY password
  | USER SYSTEM IDENTIFIED BY password
  | CONTROLFILE REUSE
  | MAXDATAFILES integer
  | MAXINSTANCES integer
  | CHARACTER SET charset
  | NATIONAL CHARACTER SET charset
  | SET DEFAULT
      { BIGFILE | SMALLFILE } TABLESPACE
  | database_logging_clauses
  | tablespace_clauses
  | set_time_zone_clause
  | [ BIGFILE | SMALLFILE ] USER_DATA TABLESPACE tablespace_name
      DATAFILE datafile_tempfile_spec [, datafile_tempfile_spec ]...
  | enable_pluggable_database
  }... ;
```

CREATE DATABASE LINK

```
CREATE [ SHARED ] [ PUBLIC ] DATABASE LINK dblink
  [ CONNECT TO
    { CURRENT_USER
    | user IDENTIFIED BY password [ dblink_authentication ]
    }
  | dblink_authentication
  ]...
  [ USING connect_string ] ;
```

CREATE DIMENSION

```
CREATE DIMENSION [ schema. ] dimension
  level_clause ...
  { hierarchy_clause
  | attribute_clause
  | extended_attribute_clause
  }...
;
```

CREATE DIRECTORY

```
CREATE [ OR REPLACE ] DIRECTORY directory
  AS 'path_name' ;
```

CREATE DISKGROUP

```
CREATE DISKGROUP diskgroup_name
  [ { HIGH | NORMAL | EXTERNAL } REDUNDANCY ]
  { [ QUORUM | REGULAR ][  FAILGROUP failgroup_name ]
  DISK qualified_disk_clause [, qualified_disk_clause]...
  } ...
  [ ATTRIBUTE { 'attribute_name' = 'attribute_value' }... ]
;
```

CREATE EDITION

```
CREATE EDITION edition
  [ AS CHILD OF parent_edition ]
;
```

CREATE FLASHBACK ARCHIVE

```
CREATE FLASHBACK ARCHIVE [DEFAULT] flashback_archive
  TABLESPACE tablespace
  [flashback_archive_quota]
  [ [NO] OPTIMIZE DATA ]
  flashback_archive_retention
;
```

CREATE FUNCTION

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
FUNCTION plsql_function_source
```

CREATE INDEX

```
CREATE [ UNIQUE | BITMAP ] INDEX [ schema. ] index
  ON { cluster_index_clause
     | table_index_clause
     | bitmap_join_index_clause
     }
[ USABLE | UNUSABLE ] ;
```

CREATE INDEXTYPE

```
CREATE [ OR REPLACE ] INDEXTYPE [ schema. ] indextype
  FOR [ schema. ] operator (paramater_type [, paramater_type ]...)
        [, [ schema. ] operator (paramater_type [, paramater_type ]...)
        ]...
  using_type_clause
  [WITH LOCAL [RANGE] PARTITION ]
  [ storage_table_clause ]
;
```

CREATE JAVA

```
CREATE [ OR REPLACE ] [ AND { RESOLVE | COMPILE } ] [ NOFORCE ]
  JAVA { { SOURCE | RESOURCE } NAMED [ schema. ] primary_name
       | CLASS [ SCHEMA schema ]
       }
  [ invoker_rights_clause ]
  [ RESOLVER ( (match_string [,] { schema_name | - })...) ]
  { USING { BFILE (directory_object_name, server_file_name)
          | { CLOB | BLOB | BFILE } subquery
          | 'key_for_BLOB'
          }
  | AS source_char
  } ;
```

CREATE LIBRARY

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
LIBRARY plsql_library_source
```

CREATE MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW [ schema. ] materialized_view
  [ OF [ schema. ] object_type ]
  [ ( { scoped_table_ref_constraint
      | column_alias [ENCRYPT [encryption_spec]]
      }
      [, { scoped_table_ref_constraint
         | column_alias [ENCRYPT [encryption_spec]]
         }
      ]...
    )
  ]
  { ON PREBUILT TABLE
    [ { WITH | WITHOUT } REDUCED PRECISION ]
  | physical_properties materialized_view_props
  }
  [ USING INDEX
    [ physical_attributes_clause
    | TABLESPACE tablespace
    ]...
  | USING NO INDEX
  ]
  [ create_mv_refresh ]
  [ FOR UPDATE ]
  [ evaluation_edition_clause ]
  [ query_rewrite_clause ]
AS subquery ;
```

CREATE MATERIALIZED VIEW LOG

```
CREATE MATERIALIZED VIEW LOG ON [ schema. ] table
  [ physical_attributes_clause
  | TABLESPACE tablespace
  | logging_clause
  | { CACHE | NOCACHE }
  ]...
  [ parallel_clause ]
  [ table_partitioning_clauses ]
  [ WITH [ { OBJECT ID
         | PRIMARY KEY
         | ROWID
         | SEQUENCE
         | COMMIT SCN
         }
           [ { , OBJECT ID
             | , PRIMARY KEY
             | , ROWID
             | , SEQUENCE
             | , COMMIT SCN
             }
           ]... ]
    (column [, column ]...)
    [ new_values_clause ]
  ] [ mv_log_purge_clause ] [ for_refresh_clause ]
;
```

CREATE MATERIALIZED ZONEMAP

```
{ create_zonemap_on_table | create_zonemap_as_subquery } ;
```

CREATE OPERATOR

```
CREATE [ OR REPLACE ] OPERATOR
   [ schema. ] operator binding_clause ;
```

CREATE OUTLINE

```
CREATE [ OR REPLACE ]
   [ PUBLIC | PRIVATE ] OUTLINE [ outline ]
   [ FROM [ PUBLIC | PRIVATE ] source_outline ]
   [ FOR CATEGORY category ]
   [ ON statement ] ;
```

CREATE PACKAGE

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
PACKAGE plsql_package_source
```

CREATE PACKAGE BODY

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
PACKAGE BODY plsql_package_body_source
```

CREATE PFILE

```
CREATE PFILE [= 'pfile_name' ]
   FROM { SPFILE [= 'spfile_name']
        | MEMORY
        } ;
```

CREATE PLUGGABLE DATABASE

```
CREATE PLUGGABLE DATABASE pdb_name
  { create_pdb_from_seed | create_pdb_clone | create_pdb_from_xml } ;
```

CREATE PROCEDURE

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
PROCEDURE plsql_procedure_source
```

CREATE PROFILE

```
CREATE PROFILE profile
  LIMIT { resource_parameters
        | password_parameters
        }...
  [ CONTAINER = { CURRENT | ALL } ] ;
```

CREATE RESTORE POINT

```
CREATE RESTORE POINT restore_point
   [ AS OF {TIMESTAMP | SCN} expr ]
   [ PRESERVE
   | GUARANTEE FLASHBACK DATABASE
   ];
```

CREATE ROLE

```
CREATE ROLE role
   [ NOT IDENTIFIED
   | IDENTIFIED { BY password
                | USING [ schema. ] package
                | EXTERNALLY
                | GLOBALLY
                }
   ] [ CONTAINER = { CURRENT | ALL } ];
```

CREATE ROLLBACK SEGMENT

```
CREATE [ PUBLIC ] ROLLBACK SEGMENT rollback_segment
  [ TABLESPACE tablespace | storage_clause ]...];
```

CREATE SCHEMA

```
CREATE SCHEMA AUTHORIZATION schema
   { create_table_statement
   | create_view_statement
   | grant_statement
   }...
;
```

CREATE SEQUENCE

```
CREATE SEQUENCE [ schema. ] sequence
   [ { INCREMENT BY | START WITH } integer
   | { MAXVALUE integer | NOMAXVALUE }
   | { MINVALUE integer | NOMINVALUE }
   | { CYCLE | NOCYCLE }
   | { CACHE integer | NOCACHE }
   | { ORDER | NOORDER }
   | { KEEP | NOKEEP }
   | { SESSION | GLOBAL }
   ]...
;
```

CREATE SPFILE

```
CREATE SPFILE [= 'spfile_name' ]
  FROM { PFILE [= 'pfile_name' ]
       | MEMORY
       } ;
```

CREATE SYNONYM

```
CREATE [ OR REPLACE ] [ EDITIONABLE | NONEDITIONABLE ]
  [ PUBLIC ] SYNONYM
  [ schema. ] synonym
  FOR [ schema. ] object [ @ dblink ] ;
```

CREATE TABLE

```
CREATE [ GLOBAL TEMPORARY ] TABLE [ schema. ] table
  { relational_table | object_table | XMLType_table } ;
```

CREATE TABLESPACE

```
CREATE
   [ BIGFILE | SMALLFILE ]
   { permanent_tablespace_clause
   | temporary_tablespace_clause
   | undo_tablespace_clause
   } ;
```

CREATE TRIGGER

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
TRIGGER plsql_trigger_source
```

CREATE TYPE

```
CREATE [OR REPLACE]
[ EDITIONABLE | NONEDITIONABLE ]
TYPE plsql_type_source
```

CREATE TYPE BODY

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
TYPE BODY plsql_type_body_source
```

CREATE USER

```
CREATE USER user
   IDENTIFIED { BY password
              | EXTERNALLY [ AS 'certificate_DN' | AS 'kerberos_principal_name' ]
              | GLOBALLY [ AS '[ directory_DN ]' ]
              }
   [ DEFAULT TABLESPACE tablespace
   | TEMPORARY TABLESPACE
        { tablespace | tablespace_group_name }
   | { QUOTA { size_clause | UNLIMITED } ON tablespace }...
   | PROFILE profile
   | PASSWORD EXPIRE
   | ACCOUNT { LOCK | UNLOCK }
     [ DEFAULT TABLESPACE tablespace
     | TEMPORARY TABLESPACE
          { tablespace | tablespace_group_name }
     | { QUOTA { size_clause | UNLIMITED } ON tablespace }...
     | PROFILE profile
     | PASSWORD EXPIRE
     | ACCOUNT { LOCK | UNLOCK }
     | ENABLE EDITIONS
     | CONTAINER = { CURRENT | ALL }
     ]...
  ] ;
```

CREATE VIEW

```
CREATE [OR REPLACE]
  [[NO] FORCE]
  [ EDITIONING | EDITIONABLE [ EDITIONING ] | NONEDITIONABLE ]
  VIEW [schema.] view
   [ ( { alias [ VISIBLE | INVISIBLE ] [ inline_constraint... ]
       | out_of_line_constraint
       }
         [, { alias [ VISIBLE | INVISIBLE ] [ inline_constraint...]
            | out_of_line_constraint
     }
  ]
     )
   | object_view_clause
   | XMLType_view_clause
   ]
   [ BEQUEATH { CURRENT_USER | DEFINER } ]
   AS subquery [ subquery_restriction_clause ] ;
```

DELETE

```
DELETE [ hint ]
   [ FROM ]
   { dml_table_expression_clause
   | ONLY (dml_table_expression_clause)
   } [ t_alias ]
     [ where_clause ]
     [ returning_clause ]
     [error_logging_clause];
```

DISASSOCIATE STATISTICS

```
DISASSOCIATE STATISTICS FROM
   { COLUMNS [ schema. ]table.column
               [, [ schema. ]table.column ]...
   | FUNCTIONS [ schema. ]function
                 [, [ schema. ]function ]...
   | PACKAGES [ schema. ]package
                [, [ schema. ]package ]...
   | TYPES [ schema. ]type
             [, [ schema. ]type ]...
   | INDEXES [ schema. ]index
               [, [ schema. ]index ]...
   | INDEXTYPES [ schema. ]indextype
                  [, [ schema. ]indextype ]...
   }
   [ FORCE ] ;
```

DROP AUDIT POLICY

```
DROP AUDIT POLICY policy ;
```

DROP CLUSTER

```
DROP CLUSTER [ schema. ] cluster
   [ INCLUDING TABLES [ CASCADE CONSTRAINTS ] ] ;
```

DROP CONTEXT

```
DROP CONTEXT namespace ;
```

DROP DATABASE

```
DROP DATABASE ;
```

DROP DATABASE LINK

```
DROP [ PUBLIC ] DATABASE LINK dblink ;
```

DROP DIMENSION

```
DROP DIMENSION [ schema. ] dimension ;
```

DROP DIRECTORY

```
DROP DIRECTORY directory_name ;
```

DROP DISKGROUP

```
DROP DISKGROUP diskgroup_name
   [  FORCE INCLUDING CONTENTS
   | { INCLUDING | EXCLUDING } CONTENTS
   ];
```

DROP EDITION

```
DROP EDITION edition [CASCADE];
```

DROP FLASHBACK ARCHIVE

```
DROP FLASHBACK ARCHIVE flashback_archive;
```

DROP FUNCTION

```
DROP FUNCTION [ schema. ] function_name ;
```

DROP INDEX

```
DROP INDEX [ schema. ] index [ ONLINE ] [ FORCE ] ;
```

DROP INDEXTYPE

```
DROP INDEXTYPE [ schema. ] indextype [ FORCE ] ;
```

DROP JAVA

```
DROP JAVA { SOURCE | CLASS | RESOURCE }
  [ schema. ] object_name ;
```

DROP LIBRARY

```
DROP LIBRARY library_name ;
```

DROP MATERIALIZED VIEW

```
DROP MATERIALIZED VIEW [ schema. ] materialized_view
   [ PRESERVE TABLE ] ;
```

DROP MATERIALIZED VIEW LOG

```
DROP MATERIALIZED VIEW LOG ON [ schema. ] table ;
```

DROP MATERIALIZED ZONEMAP

```
DROP MATERIALIZED ZONEMAP [ schema. ] zonemap_name ;
```

DROP OPERATOR

```
DROP OPERATOR [ schema. ] operator [ FORCE ] ;
```

DROP OUTLINE

```
DROP OUTLINE outline ;
```

DROP PACKAGE

```
DROP PACKAGE [ BODY ] [ schema. ] package ;
```

DROP PLUGGABLE DATABASE

```
DROP PLUGGABLE DATABASE pdb_name
  [ { KEEP | INCLUDING } DATAFILES ] ;
```

DROP PROCEDURE

```
DROP PROCEDURE [ schema. ] procedure ;
```

DROP PROFILE

```
DROP PROFILE profile [ CASCADE ] ;
```

DROP RESTORE POINT

```
DROP RESTORE POINT restore_point ;
```

DROP ROLE

```
DROP ROLE role ;
```

DROP ROLLBACK SEGMENT

```
DROP ROLLBACK SEGMENT rollback_segment ;
```

DROP SEQUENCE

```
DROP SEQUENCE [ schema. ] sequence_name ;
```

DROP SYNONYM

```
DROP [PUBLIC] SYNONYM [ schema. ] synonym [FORCE] ;
```

DROP TABLE

```
DROP TABLE [ schema. ] table
  [ CASCADE CONSTRAINTS ] [ PURGE ] ;
```

DROP TABLESPACE

```
DROP TABLESPACE tablespace
```

```
   [ INCLUDING CONTENTS [ {AND | KEEP} DATAFILES ]
     [ CASCADE CONSTRAINTS ]
   ] ;
```

DROP TRIGGER

```
DROP TRIGGER [ schema. ] trigger ;
```

DROP TYPE

```
DROP TYPE [ schema. ] type_name [ FORCE | VALIDATE ] ;
```

DROP TYPE BODY

```
DROP TYPE BODY [ schema. ] type_name ;
```

DROP USER

```
DROP USER user [ CASCADE ] ;
```

DROP VIEW

```
DROP VIEW [ schema. ] view [ CASCADE CONSTRAINTS ] ;
```

EXPLAIN PLAN

```
EXPLAIN PLAN
   [ SET STATEMENT_ID = string ]
   [ INTO [ schema. ] table [ @ dblink ] ]
FOR statement ;
```

FLASHBACK DATABASE

```
FLASHBACK [ STANDBY ] DATABASE [ database ]
   { TO { { SCN | TIMESTAMP } expr
        | RESTORE POINT restore_point
        }
   }
   | { TO BEFORE { { SCN | TIMESTAMP } expr
                 | RESETLOGS
                 }
   } ;
```

FLASHBACK TABLE

```
FLASHBACK TABLE
   [ schema. ] table
     [, [ schema. ] table ]...
   TO { { { SCN | TIMESTAMP } expr
        | RESTORE POINT restore_point
        } [ { ENABLE | DISABLE } TRIGGERS ]
      | BEFORE DROP [ RENAME TO table ]
      } ;
```

GRANT

```
GRANT
  { { { grant_system_privileges | grant_object_privileges }
      [ CONTAINER = { CURRENT | ALL } ] }
  | grant_roles_to_programs
  } ;
```

INSERT

```
INSERT [ hint ]
   { single_table_insert | multi_table_insert } ;
```

LOCK TABLE

```
LOCK TABLE [ schema. ] { table | view }
   [ partition_extension_clause
   | @ dblink
   ] [, [ schema. ] { table | view }
      [ partition_extension_clause
      | @ dblink
      ]
     ]...
   IN lockmode MODE
   [ NOWAIT
   | WAIT integer
   ] ;
```

MERGE

```
MERGE [ hint ]
   INTO [ schema. ] { table | view } [ t_alias ]
   USING { [ schema. ] { table | view }
         | subquery
         } [ t_alias ]
   ON ( condition )
   [ merge_update_clause ]
   [ merge_insert_clause ]
   [ error_logging_clause ] ;
```

NOAUDIT (Traditional Auditing)

```
NOAUDIT
   { audit_operation_clause [ auditing_by_clause ]
   | audit_schema_object_clause
   | NETWORK
   | DIRECT_PATH LOAD [ auditing_by_clause ]
   }
   [ WHENEVER [ NOT ] SUCCESSFUL ]
   [ CONTAINER = { CURRENT | ALL } ] ;
```

NOAUDIT (Unified Auditing)

```
NOAUDIT
  { POLICY policy
  | CONTEXT NAMESPACE namespace ATTRIBUTES attribute [, attribute ]...
    [, CONTEXT NAMESPACE namespace ATTRIBUTES attribute [, attribute ]... ]...
  }
  [ BY user [, user]... ] ;
```

PURGE

```
PURGE { { TABLE table | INDEX index }
      | { RECYCLEBIN | DBA_RECYCLEBIN }
      | TABLESPACE tablespace [ USER username ]
      } ;
```

RENAME

```
RENAME old_name TO new_name ;
```

REVOKE

```
REVOKE
  { { revoke_system_privileges | revoke_object_privileges }
    [ CONTAINER = { CURRENT | ALL } ] }
  | revoke_roles_from_programs ;
```

ROLLBACK

```
ROLLBACK [ WORK ]
```

```
   [ TO [ SAVEPOINT ] savepoint
   | FORCE string
   ] ;
```

SAVEPOINT

```
SAVEPOINT savepoint ;
```

SELECT

```
subquery [ for_update_clause ] ;
```

SET CONSTRAINT[S]

```
SET { CONSTRAINT | CONSTRAINTS }
    { constraint [, constraint ]...
    | ALL
    }
    { IMMEDIATE | DEFERRED } ;
```

SET ROLE

```
SET ROLE
   { role [ IDENTIFIED BY password ]
     [, role [ IDENTIFIED BY password ] ]...
   | ALL [ EXCEPT role [, role ]... ]
   | NONE
   } ;
```

SET TRANSACTION

```
SET TRANSACTION
   { { READ { ONLY | WRITE }
     | ISOLATION LEVEL
       { SERIALIZABLE | READ COMMITTED }
     | USE ROLLBACK SEGMENT rollback_segment
     } [ NAME string ]
   | NAME string
   } ;
```

TRUNCATE_CLUSTER

```
TRUNCATE CLUSTER [schema.] cluster
  [ {DROP | REUSE} STORAGE ] ;
```

TRUNCATE_TABLE

```
TRUNCATE TABLE [schema.] table
  [ {PRESERVE | PURGE} MATERIALIZED VIEW LOG ]
  [ {DROP [ ALL ] | REUSE} STORAGE ] [ CASCADE ] ;
```

UPDATE

```
UPDATE [ hint ]
   { dml_table_expression_clause
   | ONLY (dml_table_expression_clause)
   } [ t_alias ]
   update_set_clause
   [ where_clause ]
   [ returning_clause ]
   [error_logging_clause] ;
```

2 SQL Functions

This chapter presents the syntax for SQL functions.

This chapter includes the following section:

- Syntax for SQL Functions

Syntax for SQL Functions

A function is a command that manipulates data items and returns a single value.

The sections that follow show each SQL function and its related syntax. Refer to Chapter 5, "Subclauses" for the syntax of the subclauses.

> **See Also:** *Oracle Database SQL Language Reference* for detailed information about SQL functions

ABS

```
ABS(n)
```

ACOS

```
ACOS(n)
```

ADD_MONTHS

```
ADD_MONTHS(date, integer)
```

aggregate_function

Aggregate functions return a single result row based on groups of rows, rather than on single rows.

analytic_function

```
analytic_function([ arguments ]) OVER (analytic_clause)
```

APPENDCHILDXML

```
APPENDCHILDXML
  ( XMLType_instance, XPath_string, value_expr [, namespace_string ])
```

APPROX_COUNT_DISTINCT

```
APPROX_COUNT_DISTINCT(expr)
```

ASCII

```
ASCII(char)
```

ASCIISTR

```
ASCIISTR(char)
```

ASIN

```
ASIN(n)
```

ATAN

```
ATAN(n)
```

ATAN2

```
ATAN2(n1 , n2)
```

AVG

```
AVG([ DISTINCT | ALL ] expr) [ OVER(analytic_clause) ]
```

BFILENAME

```
BFILENAME('directory', 'filename')
```

BIN_TO_NUM

```
BIN_TO_NUM(expr [, expr ]... )
```

BITAND

```
BITAND(expr1, expr2)
```

CARDINALITY

```
CARDINALITY(nested_table)
```

CAST

```
CAST({ expr | MULTISET (subquery) } AS type_name)
```

CEIL

```
CEIL(n)
```

CHARTOROWID

```
CHARTOROWID(char)
```

CHR

```
CHR(n [ USING NCHAR_CS ])
```

CLUSTER_DETAILS

```
CLUSTER_DETAILS ( [ schema . ] model
                  [ , cluster_id [ , topN ] ] [ DESC | ASC | ABS ]
                  mining_attribute_clause )
```

CLUSTER_DETAILS (analytic)

```
CLUSTER_DETAILS ( INTO n
                  [ , cluster_id [ , topN ] ] [ DESC | ASC | ABS ]
                  mining_attribute_clause )
                OVER ( mining_analytic_clause )
```

CLUSTER_DISTANCE

```
CLUSTER_DISTANCE ( [ schema . ] model [ , cluster_id ] mining_attribute_clause )
```

CLUSTER_DISTANCE (analytic)

```
CLUSTER_DISTANCE ( INTO n [, cluster_id] mining_attribute_clause )
                 OVER ( mining_analytic_clause )
```

CLUSTER_ID

```
CLUSTER_ID ( [ schema . ] model mining_attribute_clause )
```

CLUSTER_ID (analytic)

```
CLUSTER_ID ( INTO n mining_attribute_clause )
           OVER ( mining_analytic_clause )
```

CLUSTER_PROBABILITY

```
CLUSTER_PROBABILITY ( [ schema . ] model [, cluster_id ] mining_attribute_clause )
```

CLUSTER_PROBABILITY (analytic)

```
CLUSTER_PROBABILITY ( INTO n [, cluster_id] mining_attribute_clause )
                    OVER ( mining_analytic_clause )
```

CLUSTER_SET

```
CLUSTER_SET ( [ schema . ] model [ , topN [ , cutoff ] ] mining_attribute_clause )
```

CLUSTER_SET (analytic)

```
CLUSTER_SET ( INTO n [, topN [, cutoff]] mining_attribute_clause )
            OVER ( mining_analytic_clause )
```

COALESCE

```
COALESCE(expr [, expr ]...)
```

COLLECT

```
COLLECT( [ DISTINCT | UNIQUE ] column [ ORDER BY expr ] )
```

COMPOSE

```
COMPOSE(char)
```

CON_DBID_TO_ID

```
CON_DBID_TO_ID(container_dbid)
```

CON_GUID_TO_ID

```
CON_GUID_TO_ID(container_guid)
```

CON_NAME_TO_ID

```
CON_NAME_TO_ID(container_name)
```

CON_UID_TO_ID

```
CON_UID_TO_ID(container_uid)
```

CONCAT

```
CONCAT(char1, char2)
```

CONVERT

```
CONVERT(char, dest_char_set[, source_char_set ])
```

CORR

```
CORR(expr1, expr2) [ OVER (analytic_clause) ]
```

CORR_K, CORR_S

```
{ CORR_K | CORR_S }
   (expr1, expr2
     [, { COEFFICIENT
        | ONE_SIDED_SIG
        | ONE_SIDED_SIG_POS
        | ONE_SIDED_SIG_NEG
        | TWO_SIDED_SIG
        }
     ]
   )
```

COS

```
COS(n)
```

COSH

```
COSH(n)
```

COUNT

```
COUNT({ * | [ DISTINCT | ALL ] expr }) [ OVER (analytic_clause) ]
```

COVAR_POP

```
COVAR_POP(expr1, expr2)
   [ OVER (analytic_clause) ]
```

COVAR_SAMP

```
COVAR_SAMP(expr1, expr2) [ OVER (analytic_clause) ]
```

CUBE_TABLE

```
CUBE_TABLE
( ' { schema.cube [ {HIERARCHY | HRR} dimension hierarchy ]...
    | schema.dimension [ {HIERARCHY | HRR} [dimension] hierarchy ]
    }
  '
)
```

CUME_DIST (aggregate)

```
CUME_DIST(expr[,expr ]...) WITHIN GROUP
  (ORDER BY expr [ DESC | ASC ]
                 [ NULLS { FIRST | LAST } ]
            [, expr [ DESC | ASC ]
                    [ NULLS { FIRST | LAST } ]
            ]...
  )
```

CUME_DIST (analytic)

```
CUME_DIST() OVER ([ query_partition_clause ] order_by_clause)
```

CURRENT_DATE

```
CURRENT_DATE
```

CURRENT_TIMESTAMP

```
CURRENT_TIMESTAMP [ (precision) ]
```

CV

```
CV([ dimension_column ])
```

DATAOBJ_TO_PARTITION

```
DATAOBJ_TO_PARTITION( table, partition_id )
```

DBTIMEZONE

```
DBTIMEZONE
```

DECODE

```
DECODE(expr, search, result [, search, result ]... [, default ])
```

DECOMPOSE

```
DECOMPOSE( string [, { 'CANONICAL' | 'COMPATIBILITY' } ] )
```

DELETEXML

```
DELETEXML( XMLType_instance, XPath_string [, namespace_string ])
```

DENSE_RANK (aggregate)

```
DENSE_RANK(expr [, expr ]...) WITHIN GROUP
  (ORDER BY expr [ DESC | ASC ]
                 [ NULLS { FIRST | LAST } ]
            [,expr [ DESC | ASC ]
                   [ NULLS { FIRST | LAST } ]
            ]...
  )
```

DENSE_RANK (analytic)

```
DENSE_RANK( ) OVER([ query_partition_clause ] order_by_clause)
```

DEPTH

```
DEPTH(correlation_integer)
```

DEREF

```
DEREF(expr)
```

DUMP

```
DUMP(expr[, return_fmt [, start_position [, length ] ]])
```

EMPTY_BLOB, EMPTY_CLOB

```
{ EMPTY_BLOB | EMPTY_CLOB }( )
```

EXISTSNODE

```
EXISTSNODE(XMLType_instance, XPath_string [, namespace_string ])
```

EXP

```
EXP(n)
```

EXTRACT (datetime)

```
EXTRACT( { YEAR
         | MONTH
         | DAY
         | HOUR
         | MINUTE
         | SECOND
         | TIMEZONE_HOUR
         | TIMEZONE_MINUTE
         | TIMEZONE_REGION
         | TIMEZONE_ABBR
         }
         FROM { expr }
       )
```

EXTRACT (XML)

```
EXTRACT(XMLType_instance, XPath_string [, namespace_string ])
```

EXTRACTVALUE

```
EXTRACTVALUE(XMLType_instance, XPath_string [, namespace_string ])
```

FEATURE_DETAILS

```
FEATURE_DETAILS ( [ schema . ] model
                  [ , feature_id [ , topN ] ] [ DESC | ASC | ABS ]
                  mining_attribute_clause )
```

FEATURE_DETAILS (analytic)

```
FEATURE_DETAILS ( INTO n
                  [ , feature_id [ , topN ] ] [ DESC | ASC | ABS ]
                  mining_attribute_clause )
                OVER ( mining_analytic_clause )
```

FEATURE_ID

```
FEATURE_ID( [ schema . ] model mining_attribute_clause )
```

FEATURE_ID (analytic)

```
FEATURE_ID ( INTO n mining_attribute_clause )
           OVER ( mining_analytic_clause )
```

FEATURE_SET

```
FEATURE_SET ( [ schema . ] model [, topN [, cutoff ]] mining_attribute_clause )
```

FEATURE_SET (analytic)

```
FEATURE_SET ( INTO n [, topN [, cutoff ] ] mining_attribute_clause )
            OVER ( mining_analytic_clause )
```

FEATURE_VALUE

```
FEATURE_VALUE ( [ schema . ] model [, feature_id ] mining_attribute_clause )
```

FEATURE_VALUE (analytic)

```
FEATURE_VALUE ( INTO n [ , feature_id ] mining_attribute_clause )
              OVER ( mining_analytic_clause )
```

FIRST

```
aggregate_function
   KEEP
   (DENSE_RANK FIRST ORDER BY
    expr [ DESC | ASC ]
         [ NULLS { FIRST | LAST } ]
    [, expr [ DESC | ASC ]
            [ NULLS { FIRST | LAST } ]
    ]...
   )
   [ OVER ( [query_partition_clause] ) ]
```

FIRST_VALUE

```
FIRST_VALUE
  { (expr) [ {RESPECT | IGNORE} NULLS ]
  | (expr [ {RESPECT | IGNORE} NULLS ])
  }
  OVER (analytic_clause)
```

FLOOR

```
FLOOR(n)
```

FROM_TZ

```
FROM_TZ (timestamp_value, time_zone_value)
```

GREATEST

```
GREATEST(expr [, expr ]...)
```

GROUP_ID

```
GROUP_ID( )
```

GROUPING

```
GROUPING(expr)
```

GROUPING_ID

```
GROUPING_ID(expr [, expr ]...)
```

HEXTORAW

```
HEXTORAW(char)
```

INITCAP

```
INITCAP(char)
```

INSERTCHILDXML

```
INSERTCHILDXML
  ( XMLType_instance, XPath_string, child_expr, value_expr [, namespace_string ] )
```

INSERTCHILDXMLAFTER

```
INSERTCHILDXMLAFTER
  ( XMLType_instance, XPath_string, child_expr, value_expr [, namespace_string ] )
```

INSERTCHILDXMLBEFORE

```
INSERTCHILDXMLBEFORE
  ( XMLType_instance, XPath_string, child_expr, value_expr [, namespace_string ] )
```

INSERTXMLAFTER

```
INSERTXMLAFTER
  ( XMLType_instance, XPath_string, value_expr [, namespace_string ] )
```

INSERTXMLBEFORE

```
INSERTXMLBEFORE
  ( XMLType_instance, XPath_string, value_expr [, namespace_string ] )
```

INSTR

```
{ INSTR
| INSTRB
| INSTRC
| INSTR2
| INSTR4
}
(string , substring [, position [, occurrence ] ])
```

ITERATION_NUMBER

```
ITERATION_NUMBER
```

JSON_QUERY

```
JSON_QUERY
  ( expr [ FORMAT JSON ], JSON_path_expression
    [ JSON_query_returning_clause ] [ JSON_query_wrapper_clause ]
    [ JSON_query_on_error_clause ]
  )
```

JSON_TABLE

```
JSON_TABLE
  ( expr [ FORMAT JSON ], JSON_path_expression
    [ JSON_table_on_error_clause ] JSON_columns_clause )
```

JSON_VALUE

```
JSON_VALUE
  ( expr [ FORMAT JSON ], JSON_path_expression
    [ JSON_value_returning_clause ] [ JSON_value_on_error_clause ]
  )
```

LAG

```
LAG
  { ( value_expr [, offset [, default]]) [ { RESPECT | IGNORE } NULLS ]
  | ( value_expr [ { RESPECT | IGNORE } NULLS ] [, offset [, default]] )
  }
  OVER ([ query_partition_clause ] order_by_clause)
```

LAST

```
aggregate_function KEEP
  (DENSE_RANK LAST ORDER BY
    expr [ DESC | ASC ]
         [ NULLS { FIRST | LAST } ]
    [, expr [ DESC | ASC ]
            [ NULLS { FIRST | LAST } ]
    ]...
  )
  [ OVER ( [query_partition_clause] ) ]
```

LAST_DAY

```
LAST_DAY(date)
```

LAST_VALUE

```
LAST_VALUE
  { (expr) [ { RESPECT | IGNORE } NULLS ]
  | (expr [ { RESPECT | IGNORE } NULLS ])
  OVER (analytic_clause)
```

LEAD

```
LEAD
  { ( value_expr [, offset [, default]] ) [ { RESPECT | IGNORE } NULLS ]
  | ( value_expr [ { RESPECT | IGNORE } NULLS ] [, offset [, default]] )
  }
  OVER ([ query_partition_clause ] order_by_clause)
```

LEAST

```
LEAST(expr [, expr ]...)
```

LENGTH

```
{ LENGTH
| LENGTHB
| LENGTHC
| LENGTH2
| LENGTH4
}
(char)
```

LISTAGG

```
LISTAGG(measure_expr [, 'delimiter'])
  WITHIN GROUP (order_by_clause) [OVER query_partition_clause]
```

LN

```
LN(n)
```

LNNVL

```
LNNVL(condition)
```

LOCALTIMESTAMP

```
LOCALTIMESTAMP [ (timestamp_precision) ]
```

LOG

```
LOG(n2, n1)
```

LOWER

```
LOWER(char)
```

LPAD

```
LPAD(expr1, n [, expr2 ])
```

LTRIM

```
LTRIM(char [, set ])
```

MAKE_REF

```
MAKE_REF({ table | view } , key [, key ]...)
```

MAX

```
MAX([ DISTINCT | ALL ] expr) [ OVER (analytic_clause) ]
```

MEDIAN

```
MEDIAN(expr) [ OVER (query_partition_clause) ]
```

MIN

```
MIN([ DISTINCT | ALL ] expr) [ OVER (analytic_clause) ]
```

MOD

```
MOD(n2, n1)
```

MONTHS_BETWEEN

```
MONTHS_BETWEEN(date1, date2)
```

NANVL

```
NANVL(n2, n1)
```

NCHR

```
NCHR(number)
```

NEW_TIME

```
NEW_TIME(date, timezone1, timezone2)
```

NEXT_DAY

```
NEXT_DAY(date, char)
```

NLS_CHARSET_DECL_LEN

```
NLS_CHARSET_DECL_LEN(byte_count, 'char_set_id')
```

NLS_CHARSET_ID

```
NLS_CHARSET_ID(string)
```

NLS_CHARSET_NAME

```
NLS_CHARSET_NAME(number)
```

NLS_INITCAP

```
NLS_INITCAP(char [, 'nlsparam' ])
```

NLS_LOWER

```
NLS_LOWER(char [, 'nlsparam' ])
```

NLS_UPPER

```
NLS_UPPER(char [, 'nlsparam' ])
```

NLSSORT

```
NLSSORT(char [, 'nlsparam' ])
```

NTH_VALUE

```
NTH_VALUE(measure_expr, n)
  [ FROM { FIRST | LAST } ][ { RESPECT | IGNORE } NULLS ]
  OVER (analytic_clause)
```

NTILE

```
NTILE(expr) OVER ([ query_partition_clause ] order_by_clause)
```

NULLIF

```
NULLIF(expr1, expr2)
```

NUMTODSINTERVAL

```
NUMTODSINTERVAL(n, 'interval_unit')
```

NUMTOYMINTERVAL

```
NUMTOYMINTERVAL(n, 'interval_unit')
```

NVL

```
NVL(expr1, expr2)
```

NVL2

```
NVL2(expr1, expr2, expr3)
```

ORA_DST_AFFECTED

```
ORA_DST_AFFECTED(datetime_expr)
```

ORA_DST_CONVERT

```
ORA_DST_CONVERT(datetime_expr [, integer [, integer ]])
```

ORA_DST_ERROR

```
ORA_DST_ERROR(datetime_expr)
```

ORA_HASH

```
ORA_HASH(expr [, max_bucket [, seed_value ] ])
```

ORA_INVOKING_USER

```
ORA_INVOKING_USER
```

ORA_INVOKING_USERID

```
ORA_INVOKING_USERID
```

PATH

```
PATH(correlation_integer)
```

PERCENT_RANK (aggregate)

```
PERCENT_RANK(expr [, expr ]...) WITHIN GROUP
  (ORDER BY
   expr [ DESC | ASC ]
        [NULLS { FIRST | LAST } ]
   [, expr [ DESC | ASC ]
           [NULLS { FIRST | LAST } ]
   ]...
  )
```

PERCENT_RANK (analytic)

```
PERCENT_RANK( )
   OVER ([ query_partition_clause ] order_by_clause)
```

PERCENTILE_CONT

```
PERCENTILE_CONT(expr) WITHIN GROUP
  (ORDER BY expr [ DESC | ASC ])
  [ OVER (query_partition_clause) ]
```

PERCENTILE_DISC

```
PERCENTILE_DISC(expr) WITHIN GROUP
  (ORDER BY expr [ DESC | ASC ])
  [ OVER (query_partition_clause) ]
```

POWER

```
POWER(n2, n1)
```

POWERMULTISET

```
POWERMULTISET(expr)
```

POWERMULTISET_BY_CARDINALITY

```
POWERMULTISET_BY_CARDINALITY(expr, cardinality)
```

PREDICTION

```
PREDICTION ( [ schema . ] model [ cost_matrix_clause ] mining_attribute_clause )
```

PREDICTION (analytic)

```
PREDICTION ( ( OF ANOMALY | FOR expr ) [ cost_matrix_clause ] mining_attribute_clause )
           OVER ( mining_analytic_clause )
```

PREDICTION_BOUNDS

```
PREDICTION_BOUNDS ( [schema.] model [, confidence_level [, class_value]]
                    mining_attribute_clause )
```

PREDICTION_COST

```
PREDICTION_COST ( [ schema . ] model [ , class ] cost_matrix_clause mining_attribute_clause )
```

PREDICTION_COST (analytic)

```
PREDICTION_COST ( ( OF ANOMALY | FOR expr ) [ , class ]
                  cost_matrix_clause mining_attribute_clause )
                OVER (mining_analytic_clause)
```

PREDICTION_DETAILS

```
PREDICTION_DETAILS ( [ schema . ] model
                     [ , class_value [ , topN ] ] [ DESC | ASC | ABS ]
                     mining_attribute_clause )
```

PREDICTION_DETAILS (analytic)

```
PREDICTION_DETAILS ( ( OF ANOMALY | FOR expr ) [ , class_value [ , topN ] ]
                     [ DESC | ASC | ABS ] mining_attribute_clause )
                   OVER ( mining_analytic_clause )
```

PREDICTION_PROBABILITY

```
PREDICTION_PROBABILITY ( [ schema . ] model [ , class ] mining_attribute_clause )
```

PREDICTION_PROBABILITY (analytic)

```
PREDICTION_PROBABILITY ( ( OF ANOMALY | FOR expr ) [ , class ]
                         mining_attribute_clause )
                       OVER (mining_analytic_clause )
```

PREDICTION_SET

```
PREDICTION_SET ( [ schema . ] model [ , bestN [ , cutoff ] ]
                 [ cost_matrix_clause ] mining_attribute_clause )
```

PREDICTION_SET (analytic)

```
PREDICTION_SET ( ( OF ANOMALY | FOR "expr" ) [ , bestN [ , cutoff ] ]
                 [ cost_matrix_clause ] mining_attribute_clause )
               OVER ( mining_analytic_clause )
```

PRESENTNNV

```
PRESENTNNV(cell_reference, expr1, expr2)
```

PRESENTV

```
PRESENTV(cell_reference, expr1, expr2)
```

PREVIOUS

```
PREVIOUS(cell_reference)
```

RANK (aggregate)

```
RANK(expr [, expr ]...) WITHIN GROUP
   (ORDER BY
    expr [ DESC | ASC ]
         [ NULLS { FIRST | LAST } ]
    [, expr [ DESC | ASC ]
            [ NULLS { FIRST | LAST } ]
    ]...
   )
```

RANK (analytic)

```
RANK( )
   OVER ([ query_partition_clause ] order_by_clause)
```

RATIO_TO_REPORT

```
RATIO_TO_REPORT(expr)
   OVER ([ query_partition_clause ])
```

RAWTOHEX

```
RAWTOHEX(raw)
```

RAWTONHEX

```
RAWTONHEX(raw)
```

REF

```
REF (correlation_variable)
```

REFTOHEX

```
REFTOHEX (expr)
```

REGEXP_COUNT

```
REGEXP_COUNT (source_char, pattern [, position [, match_param]])
```

REGEXP_INSTR

```
REGEXP_INSTR ( source_char, pattern
               [, position
                  [, occurrence
                     [, return_opt
                        [, match_param
                           [, subexpr ]
                        ]
                     ]
                  ]
               ]
             )
```

REGEXP_REPLACE

```
REGEXP_REPLACE ( source_char, pattern
                 [, replace_string
                    [, position
                       [, occurrence
                          [, match_param ]
                       ]
                    ]
                 ]
```

```
               )
```

REGEXP_SUBSTR

```
REGEXP_SUBSTR ( source_char, pattern
                [, position
                   [, occurrence
                      [, match_param
                         [, subexpr ]
                      ]
                   ]
                ]
              )
```

REGR_AVGX, REGR_AVGY, REGR_COUNT, REGR_INTERCEPT, REGR_R2, REGR_SLOPE, REGR_SXX, REGR_SXY, REGR_SYY

```
{ REGR_SLOPE
| REGR_INTERCEPT
| REGR_COUNT
| REGR_R2
| REGR_AVGX
| REGR_AVGY
| REGR_SXX
| REGR_SYY
| REGR_SXY
}
(expr1 , expr2)
[ OVER (analytic_clause) ]
```

REMAINDER

```
REMAINDER(n2, n1)
```

REPLACE

```
REPLACE(char, search_string
        [, replacement_string ]
       )
```

ROUND (date)

```
ROUND(date [, fmt ])
```

ROUND (number)

```
ROUND(n [, integer ])
```

ROW_NUMBER

```
ROW_NUMBER( )
   OVER ([ query_partition_clause ] order_by_clause)
```

ROWIDTOCHAR

```
ROWIDTOCHAR(rowid)
```

ROWIDTONCHAR

```
ROWIDTONCHAR(rowid)
```

RPAD

```
RPAD(expr1 , n [, expr2 ])
```

RTRIM

```
RTRIM(char [, set ])
```

SCN_TO_TIMESTAMP

```
SCN_TO_TIMESTAMP(number)
```

SESSIONTIMEZONE

```
SESSIONTIMEZONE
```

SET

```
SET (nested_table)
```

SIGN

```
SIGN(n)
```

SIN

```
SIN(n)
```

SINH

```
SINH(n)
```

SOUNDEX

```
SOUNDEX(char)
```

SQRT

```
SQRT(n)
```

STANDARD_HASH

```
STANDARD_HASH(expr [, 'method' ])
```

STATS_BINOMIAL_TEST

```
STATS_BINOMIAL_TEST(expr1, expr2, p
                     [, { TWO_SIDED_PROB
                        | EXACT_PROB
                        | ONE_SIDED_PROB_OR_MORE
                        | ONE_SIDED_PROB_OR_LESS
                        }
                     ]
                    )
```

STATS_CROSSTAB

```
STATS_CROSSTAB(expr1, expr2
               [, { CHISQ_OBS
                  | CHISQ_SIG
                  | CHISQ_DF
                  | PHI_COEFFICIENT
                  | CRAMERS_V
                  | CONT_COEFFICIENT
                  | COHENS_K
                  }
               ]
              )
```

STATS_F_TEST

```
STATS_F_TEST(expr1, expr2
             [, { { STATISTIC
                  | DF_NUM
                  | DF_DEN
                  | ONE_SIDED_SIG
    } , expr3
```

```
                 | TWO_SIDED_SIG
                 }
              ]
             )
```

STATS_KS_TEST

```
STATS_KS_TEST(expr1, expr2
              [, { STATISTIC | SIG } ]
             )
```

STATS_MODE

```
STATS_MODE(expr)
```

STATS_MW_TEST

```
STATS_MW_TEST(expr1, expr2
              [, { STATISTIC
                 | U_STATISTIC
                 | ONE_SIDED_SIG , expr3
                 | TWO_SIDED_SIG
                 }
              ]
             )
```

STATS_ONE_WAY_ANOVA

```
STATS_ONE_WAY_ANOVA(expr1, expr2
                    [, { SUM_SQUARES_BETWEEN
                       | SUM_SQUARES_WITHIN
                       | DF_BETWEEN
                       | DF_WITHIN
                       | MEAN_SQUARES_BETWEEN
                       | MEAN_SQUARES_WITHIN
                       | F_RATIO
                       | SIG
                       }
                    ]
                   )
```

STATS_T_TEST_INDEP, STATS_T_TEST_INDEPU, STATS_T_TEST_ONE, STATS_T_TEST_PAIRED

```
{
  STATS_T_TEST_ONE ( expr1 [, expr2 ]
|
  { { STATS_T_TEST_PAIRED
    | STATS_T_TEST_INDEP
    | STATS_T_TEST_INDEPU
    } ( expr1, expr2
  }
}
[, { { STATISTIC | ONE_SIDED_SIG } , expr3 | TWO_SIDED_SIG | DF } ] )
```

STATS_WSR_TEST

```
STATS_WSR_TEST(expr1, expr2
               [, { STATISTIC
                  | ONE_SIDED_SIG
                  | TWO_SIDED_SIG
                  }
               ]
              )
```

STDDEV

```
STDDEV([ DISTINCT | ALL ] expr)
```

```
   [ OVER (analytic_clause) ]
```

STDDEV_POP

```
STDDEV_POP(expr)
   [ OVER (analytic_clause) ]
```

STDDEV_SAMP

```
STDDEV_SAMP(expr)
   [ OVER (analytic_clause) ]
```

SUBSTR

```
{ SUBSTR
| SUBSTRB
| SUBSTRC
| SUBSTR2
| SUBSTR4
}
(char, position [, substring_length ])
```

SUM

```
SUM([ DISTINCT | ALL ] expr)
   [ OVER (analytic_clause) ]
```

SYS_CONNECT_BY_PATH

```
SYS_CONNECT_BY_PATH(column, char)
```

SYS_CONTEXT

```
SYS_CONTEXT('namespace', 'parameter' [, length ])
```

SYS_DBURIGEN

```
SYS_DBURIGEN({ column | attribute }
             [ rowid ]
               [, { column | attribute }
                  [ rowid ]
               ]...
             [, 'text ( )' ]
            )
```

SYS_EXTRACT_UTC

```
SYS_EXTRACT_UTC(datetime_with_timezone)
```

SYS_GUID

```
SYS_GUID( )
```

SYS_OP_ZONE_ID

```
SYS_OP_ZONE_ID( [ [ schema. ] table. | t_alias. ] rowid [, scale ] )
```

SYS_TYPEID

```
SYS_TYPEID(object_type_value)
```

SYS_XMLAGG

```
SYS_XMLAGG(expr [, fmt ])
```

SYS_XMLGEN

```
SYS_XMLGEN(expr [, fmt ])
```

SYSDATE

```
SYSDATE
```

SYSTIMESTAMP

```
SYSTIMESTAMP
```

TAN

```
TAN(n)
```

TANH

```
TANH(n)
```

TIMESTAMP_TO_SCN

```
TIMESTAMP_TO_SCN(timestamp)
```

TO_BINARY_DOUBLE

```
TO_BINARY_DOUBLE(expr [, fmt [, 'nlsparam' ] ])
```

TO_BINARY_FLOAT

```
TO_BINARY_FLOAT(expr [, fmt [, 'nlsparam' ] ])
```

TO_BLOB

```
TO_BLOB ( raw_value )
```

TO_CHAR (character)

```
TO_CHAR(nchar | clob | nclob)
```

TO_CHAR (datetime)

```
TO_CHAR({ datetime | interval } [, fmt [, 'nlsparam' ] ])
```

TO_CHAR (number)

```
TO_CHAR(n [, fmt [, 'nlsparam' ] ])
```

TO_CLOB

```
TO_CLOB(lob_column | char)
```

TO_DATE

```
TO_DATE(char [, fmt [, 'nlsparam' ] ])
```

TO_DSINTERVAL

```
TO_DSINTERVAL ( ' { sql_format | ds_iso_format } ' )
```

TO_LOB

```
TO_LOB(long_column)
```

TO_MULTI_BYTE

```
TO_MULTI_BYTE(char)
```

TO_NCHAR (character)

```
TO_NCHAR({char | clob | nclob})
```

TO_NCHAR (datetime)

```
TO_NCHAR({ datetime | interval }
         [, fmt [, 'nlsparam' ] ]
        )
```

TO_NCHAR (number)

```
TO_NCHAR(n [, fmt [, 'nlsparam' ] ])
```

TO_NCLOB

```
TO_NCLOB(lob_column | char)
```

TO_NUMBER

```
TO_NUMBER(expr [, fmt [, 'nlsparam' ] ])
```

TO_SINGLE_BYTE

```
TO_SINGLE_BYTE(char)
```

TO_TIMESTAMP

```
TO_TIMESTAMP(char [, fmt [, 'nlsparam' ] ])
```

TO_TIMESTAMP_TZ

```
TO_TIMESTAMP_TZ(char [, fmt [, 'nlsparam' ] ])
```

TO_YMINTERVAL

```
TO_YMINTERVAL
  ( '  { [+|-] years - months
       | ym_iso_format
       } ' )
```

TRANSLATE

```
TRANSLATE(expr, from_string, to_string)
```

TRANSLATE ... USING

```
TRANSLATE ( char USING
          { CHAR_CS | NCHAR_CS }
          )
```

TREAT

```
TREAT(expr AS [ REF ] [ schema. ]type)
```

TRIM

```
TRIM([ { { LEADING | TRAILING | BOTH }
         [ trim_character ]
       | trim_character
       }
       FROM
     ]
     trim_source
    )
```

TRUNC (date)

```
TRUNC(date [, fmt ])
```

TRUNC (number)

```
TRUNC(n1 [, n2 ])
```

TZ_OFFSET

```
TZ_OFFSET({ 'time_zone_name'
          | '{ + | - } hh : mi'
          | SESSIONTIMEZONE
          | DBTIMEZONE
          }
         )
```

UID

```
UID
```

UNISTR

```
UNISTR( string )
```

UPDATEXML

```
UPDATEXML
      (XMLType_instance,
        XPath_string, value_expr
          [, XPath_string, value_expr ]...
        [, namespace_string ]
      )
```

UPPER

```
UPPER(char)
```

USER

```
USER
```

user-defined function

```
[ schema. ]
{ [ package. ]function | user_defined_operator }
[ @ dblink. ]
[ ( [ [ DISTINCT | ALL ] expr [, expr ]... ] ) ]
```

USERENV

```
USERENV('parameter')
```

VALUE

```
VALUE(correlation_variable)
```

VAR_POP

```
VAR_POP(expr) [ OVER (analytic_clause) ]
```

VAR_SAMP

```
VAR_SAMP(expr) [ OVER (analytic_clause) ]
```

VARIANCE

```
VARIANCE([ DISTINCT | ALL ] expr)
        [ OVER (analytic_clause) ]
```

VSIZE

```
VSIZE(expr)
```

WIDTH_BUCKET

```
WIDTH_BUCKET
   (expr, min_value, max_value, num_buckets)
```

XMLAGG

```
XMLAGG(XMLType_instance [ order_by_clause ])
```

XMLCAST

```
XMLCAST ( value_expression AS datatype )
```

XMLCDATA

```
XMLCDATA ( value_expr )
```

XMLCOLATTVAL

```
XMLCOLATTVAL
  (value_expr [ AS { c_alias  | EVALNAME value_expr } ]
    [, value_expr [ AS { c_alias  | EVALNAME value_expr } ]
      ]...
  )
```

XMLCOMMENT

```
XMLCOMMENT ( value_expr )
```

XMLCONCAT

```
XMLCONCAT(XMLType_instance [, XMLType_instance ]...)
```

XMLDIFF

```
XMLDIFF ( XMLType_document, XMLType_document [ , integer, string ] )
```

XMLELEMENT

```
XMLELEMENT
 ( [ ENTITYESCAPING | NOENTITYESCAPING ]
   [ NAME ]
     { identifier
     | EVALNAME value_expr
     }
   [, XML_attributes_clause ]
   [, value_expr [ [AS] c_alias ]]...
 )
```

XMLEXISTS

```
XMLEXISTS ( XQuery_string [ XML_passing_clause ] )
```

XMLFOREST

```
XMLFOREST
  ( value_expr [ AS { c_alias | EVALNAME value_expr } ]
    [, value_expr [ AS { c_alias | EVALNAME value_expr } ]
      ]...
  )
```

XMLISVALID

```
XMLISVALID ( XMLType_instance [, XMLSchema_URL [, element ]] )
```

XMLPARSE

```
XMLPARSE
  ({ DOCUMENT | CONTENT } value_expr [ WELLFORMED ]
  )
```

XMLPATCH

```
XMLPATCH ( XMLType_document, XMLType_document )
```

XMLPI

```
XMLPI
 ( { [ NAME ] identifier
   | EVALNAME value_expr
   } [, value_expr ]
 )
```

XMLQUERY

```
XMLQUERY
 ( XQuery_string
   [ XML_passing_clause ]
   RETURNING CONTENT [NULL ON EMPTY]
 )
```

XMLROOT

```
XMLROOT
  ( value_expr, VERSION
  { value_expr | NO VALUE }
  [, STANDALONE { YES | NO | NO VALUE } ]
  )
```

XMLSEQUENCE

```
XMLSEQUENCE( XMLType_instance
           | sys_refcursor_instance [, fmt ]
           )
```

XMLSERIALIZE

```
XMLSERIALIZE
  ( { DOCUMENT | CONTENT } value_expr [ AS datatype ]
    [ ENCODING xml_encoding_spec ]
    [ VERSION string_literal ]
    [ NO INDENT | { INDENT [SIZE = number] } ]
    [ { HIDE | SHOW } DEFAULTS ]
  )
```

XMLTABLE

```
XMLTABLE
 (
  [ XMLnamespaces_clause , ] XQuery_string XMLTABLE_options
 )
```

XMLTRANSFORM

```
XMLTRANSFORM(XMLType_instance, { XMLType_instance
                               | string
                               }
            )
```

3 SQL Expressions

This chapter presents the syntax for combining values, operators, and functions into expressions.

This chapter includes the following section:

- Syntax for SQL Expression Types

Syntax for SQL Expression Types

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value. An expression generally assumes the data type of its components.

Expressions have several forms. The sections that follow show the syntax for each form of expression. Refer to Chapter 5, "Subclauses" for the syntax of the subclauses.

> **See Also:** *Oracle Database SQL Language Reference* for detailed information about SQL expressions

CASE expressions

```
CASE { simple_case_expression
     | searched_case_expression
     }
     [ else_clause ]
     END
```

Column expressions

A column expression can be a simple expression, compound expression, function expression, or expression list, containing only columns of the subject table, constants, and deterministic functions.

Compound expressions

```
{ (expr)
| { + | - | PRIOR } expr
| expr { * | / | + | - | || } expr
}

Note: The double vertical bars are part of the syntax
      (indicating concatenation) rather than BNF notation.
```

CURSOR expressions

```
CURSOR (subquery)
```

Datetime expressions

```
expr AT
   { LOCAL
   | TIME ZONE { ' [ + | - ] hh:mi'
               | DBTIMEZONE
               | 'time_zone_name'
               | expr
               }
   }
```

Function expressions

You can use any built-in SQL function or user-defined function as an expression.

Interval expressions

```
( expr1 - expr2 )
   { DAY [ (leading_field_precision) ] TO
     SECOND [ (fractional_second_precision) ]
   | YEAR [ (leading_field_precision) ] TO
     MONTH
   }
```

JSON object access expressions

```
table_alias.JSON_column [.JSON_object_key ]...
```

Model expressions

```
{ measure_column [ { condition | expr } [, { condition | expr } ]... ]
| aggregate_function
     { [ { condition | expr } [, { condition | expr } ]... ]
     | [ single_column_for_loop [, single_column_for_loop ]... ]
     | [ multi_column_for_loop ]
     }
| analytic_function
}

Note: The outside square brackets shown in boldface type are part of the syntax.
      In this case, they do not represent optionality.
```

Object access expressions

```
{ table_alias.column.
| object_table_alias.
| (expr).
}
{ attribute [.attribute ]...
  [.method ([ argument [, argument ]... ]) ]
| method ([ argument [, argument ]... ])
}
```

Placeholder expressions

```
:host_variable
   [ [ INDICATOR ]
     :indicator_variable
   ]
```

Scalar subquery expressions

A scalar subquery expression is a subquery that returns exactly one column value from one row.

Simple expressions

```
{ [ query_name.
```

```
  | [schema.]
    { table. | view. | materialized view. }
  ] { column | ROWID }
| ROWNUM
| string
| number
| sequence. { CURRVAL | NEXTVAL }
| NULL
}
```

Type constructor expressions

```
[ NEW ] [ schema. ]type_name
   ([ expr [, expr ]... ])
```

4 SQL Conditions

This chapter presents the syntax for combining one or more expressions and logical (Boolean) operators to specify a condition.

This chapter includes the following section:

- Syntax for SQL Condition Types

Syntax for SQL Condition Types

A condition specifies a combination of one or more expressions and logical (Boolean) operators and returns a value of `TRUE`, `FALSE`, or unknown.

Conditions have several forms. The sections that follow show the syntax for each form of condition. Refer to Chapter 5, "Subclauses" for the syntax of the subclauses.

> **See Also:** *Oracle Database SQL Language Reference* for detailed information about SQL conditions

BETWEEN condition

```
expr1 [ NOT ] BETWEEN expr2 AND expr3
```

Compound conditions

```
{ (condition)
| NOT condition
| condition { AND | OR } condition
}
```

EQUALS_PATH condition

```
EQUALS_PATH
    (column, path_string [, correlation_integer ])
```

EXISTS condition

```
EXISTS (subquery)
```

Floating-point conditions

```
expr IS [ NOT ] { NAN | INFINITE }
```

Group comparison conditions

```
{ expr
     { = | != | ^= | <> | > | < | >= | <= }
     { ANY | SOME | ALL }
     ({ expression_list | subquery })
| ( expr [, expr ]... )
```

```
  { = | != | ^= | <> }
  { ANY | SOME | ALL }
  ({ expression_list
     [, expression_list ]...
   | subquery
   }
  )
}
```

where !=, ^=, and <> test for inequality

IN condition

```
{ expr [ NOT ] IN ({ expression_list | subquery })
| ( expr [, expr ]... )
    [ NOT ] IN ({ expression_list [, expression_list ]...
                | subquery
                }
               )
}
```

IS A SET condition

```
nested_table IS [ NOT ] A SET
```

IS ANY condition

```
[ dimension_column IS ] ANY
```

IS EMPTY condition

```
nested_table IS [ NOT ] EMPTY
```

IS JSON condition

```
expr IS [ NOT ] JSON [ FORMAT JSON ] [ STRICT | LAX ]
[ { WITH | WITHOUT } UNIQUE KEYS ]
```

IS OF *type* condition

```
expr IS [ NOT ] OF [ TYPE ]
   ([ ONLY ] [ schema. ] type
      [, [ ONLY ] [ schema. ] type ]...
   )
```

IS PRESENT condition

```
cell_reference IS PRESENT
```

JSON_EXISTS condition

```
JSON_EXISTS( expr [ FORMAT JSON ], JSON_path_expression
             [ JSON_exists_on_error_clause ] )
```

JSON_TEXTCONTAINS condition

```
JSON_TEXTCONTAINS( column, JSON_path_expression, string )
```

LIKE condition

```
char1 [ NOT ] { LIKE | LIKEC | LIKE2 | LIKE4 }
  char2 [ ESCAPE esc_char ]
```

Logical conditions

```
{ NOT | AND | OR }
```

MEMBER condition

```
expr [ NOT ] MEMBER [ OF ] nested_table
```

Null conditions

```
expr IS [ NOT ] NULL
```

REGEXP_LIKE condition

```
REGEXP_LIKE(source_char, pattern
            [, match_param ]
           )
```

Simple comparison conditions

```
{ expr
  { = | != | ^= | <> | > | < | >= | <= }
  expr
| (expr [, expr ]...)
  { = | != | ^= | <> }
  ( expression_list | subquery )
}
```

where `!=`, `^=`, and <> test for inequality

SUBMULTISET condition

```
nested_table1
[ NOT ] SUBMULTISET [ OF ]
nested_table2
```

UNDER_PATH condition

```
UNDER_PATH (column [, levels ], path_string
             [, correlation_integer ]
           )
```

5 Subclauses

This chapter presents the syntax for the subclauses found in the syntax for SQL statements, functions, expressions and conditions.

This chapter includes the following section:

- Syntax for Subclauses

Syntax for Subclauses

The sections that follow show the syntax for each subclause found in:

- Chapter 1, "SQL Statements"
- Chapter 2, "SQL Functions"
- Chapter 3, "SQL Expressions"
- Chapter 4, "SQL Conditions"

> **See Also:** *Oracle Database SQL Language Reference* for detailed information about Oracle SQL

action_audit_clause

```
{ standard_actions | component_actions }...
```

activate_standby_db_clause

```
ACTIVATE
     [ PHYSICAL | LOGICAL ]
     STANDBY DATABASE
     [ FINISH APPLY ]
```

add_binding_clause

```
ADD BINDING
  (parameter_type [, parameter_type ]...)
  RETURN (return_type)
  [ implementation_clause ]
  using_function_clause
```

add_column_clause

```
ADD
   {column_definition | virtual_column_definition
      [, column_definition | virtual_column_definition] ...
   }
   [ column_properties ]
   [ out_of_line_part_storage [, out_of_line_part_storage]...]
```

add_disk_clause

```
ADD
  { [ QUORUM | REGULAR ] [ FAILGROUP failgroup_name ]
    DISK qualified_disk_clause [, qualified_disk_clause ]...
  }...
```

add_hash_index_partition

```
ADD PARTITION
   [ partition_name ]
   [ TABLESPACE tablespace_name ]
   [ index_compression ]
   [ parallel_clause ]
```

add_hash_partition_clause

```
partitioning_storage_clause
[ update_index_clauses ]
[ parallel_clause ]
[ indexing_clause ]
```

add_hash_subpartition

```
ADD individual_hash_subparts
   [ dependent_tables_clause ]
   [ update_index_clauses ]
   [ parallel_clause ]
```

add_list_partition_clause

```
list_values_clause
[ table_partition_description ]
[ ( { range_subpartition_desc [, range_subpartition_desc] ...
    | list_subpartition_desc [, list_subpartition_desc] ...
    | individual_hash_subparts [, individual_hash_subparts] ...
    }
  ) | hash_subparts_by_quantity ]
[ update_index_clauses ]
```

add_list_subpartition

```
ADD list_subpartition_desc [, list_subpartition_desc ]...
[ dependent_tables_clause ] [ update_index_clauses ]
```

add_logfile_clauses

```
ADD [ STANDBY ] LOGFILE
   {
     { [ INSTANCE 'instance_name' ] | [ THREAD 'integer' ] }
     [ GROUP integer ] redo_log_file_spec
       [, [ GROUP integer ] redo_log_file_spec ]...
   | MEMBER 'filename' [ REUSE ] [, 'filename' [ REUSE ] ]...
        TO logfile_descriptor [, logfile_descriptor ]...
   }
```

add_mv_log_column_clause

```
ADD (column)
```

add_overflow_clause

```
ADD OVERFLOW [ segment_attributes_clause ]
  [ ( PARTITION [ segment_attributes_clause ]
    [, PARTITION [ segment_attributes_clause ] ]...
    )
  ]
```

add_period_clause

```
ADD ( period_definition )
```

add_range_partition_clause

```
range_values_clause
[ table_partition_description ]
[ ( { range_subpartition_desc [, range_subpartition_desc] ...
    | list_subpartition_desc [, list_subpartition_desc] ...
    | individual_hash_subparts [, individual_hash_subparts] ...
    }
  ) | hash_subparts_by_quantity ]
[ update_index_clauses ]
```

add_range_subpartition

```
ADD range_subpartition_desc [, range_subpartition_desc ]...
[ dependent_tables_clause ] [ update_index_clauses ]
```

add_system_partition_clause

```
[table_partition_description]
[update_index_clauses]
```

add_table_partition

```
ADD {
PARTITION [ partition ] add_range_partition_clause
  [, PARTITION [ partition ] add_range_partition_clause ]...
| PARTITION [ partition ] add_list_partition_clause
  [, PARTITION [ partition ] add_list_partition_clause ]...
| PARTITION [ partition ] add_system_partition_clause
  [, PARTITION [ partition ] add_system_partition_clause ]...
  [ BEFORE { partition_name | partition_number } ]
| PARTITION [ partition ] add_hash_partition_clause
} [ dependent_tables_clause ]
```

add_update_secret

```
{ ADD | UPDATE } SECRET 'secret' FOR CLIENT 'client_identifier'
  [ USING TAG 'tag' ]
  IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

add_volume_clause

```
ADD VOLUME asm_volume SIZE size_clause [redundancy_clause]
  [ STRIPE_WIDTH integer {K | M} ]
  [ STRIPE_COLUMNS integer ]
  [ ATTRIBUTE (disk_region_clause) ]
```

advanced_index_compression

```
COMPRESS ADVANCED LOW | NOCOMPRESS
```

alias_file_name

```
+diskgroup_name [ (template_name) ] /alias_name
```

allocate_extent_clause

```
ALLOCATE EXTENT
  [ ( { SIZE size_clause
      | DATAFILE 'filename'
      | INSTANCE integer
      } ...
    )
  ]
```

allow_disallow_clustering

```
{ ALLOW | DISALLOW } CLUSTERING
```

alter_datafile_clause

```
DATAFILE
   { 'filename' | filenumber }
     [, 'filename' | filenumber ]...
   }
   { ONLINE
   | OFFLINE [ FOR DROP ]
   | RESIZE size_clause
   | autoextend_clause
   | END BACKUP
   }
```

alter_external_table

```
{ add_column_clause
| modify_column_clauses
| drop_column_clause
| parallel_clause
| external_data_properties
| REJECT LIMIT { integer | UNLIMITED }
| PROJECT COLUMN { ALL | REFERENCED }
}
  [ add_column_clause
  | modify_column_clauses
  | drop_column_clause
  | parallel_clause
  | external_data_properties
  | REJECT LIMIT { integer | UNLIMITED }
  | PROJECT COLUMN { ALL | REFERENCED }
  ]...
```

alter_index_partitioning

```
{ modify_index_default_attrs
| add_hash_index_partition
| modify_index_partition
| rename_index_partition
| drop_index_partition
| split_index_partition
| coalesce_index_partition
| modify_index_subpartition
}
```

alter_interval_partitioning

```
{ SET INTERVAL ( [ expr ] )
| SET STORE IN ( tablespace [, tablespace]... )
}
```

alter_iot_clauses

```
{ index_org_table_clause
| alter_overflow_clause
| alter_mapping_table_clauses
| COALESCE
}
```

alter_keystore_password

```
ALTER KEYSTORE PASSWORD
  IDENTIFIED BY old_keystore_password
  SET new_keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

alter_mapping_table_clauses

```
MAPPING TABLE
  { allocate_extent_clause
  | deallocate_unused_clause
  }
```

alter_mv_refresh

```
REFRESH
   { { FAST | COMPLETE | FORCE }
   | ON { DEMAND | COMMIT }
   | { START WITH | NEXT } date
   | WITH PRIMARY KEY
   | USING
        { DEFAULT MASTER ROLLBACK SEGMENT
        | MASTER ROLLBACK SEGMENT rollback_segment
        }
   | USING { ENFORCED | TRUSTED } CONSTRAINTS
   }
```

alter_overflow_clause

```
{ add_overflow_clause
| OVERFLOW
     { segment_attributes_clause
     | allocate_extent_clause
     | shrink_clause
     | deallocate_unused_clause
     }...
}
```

alter_query_rewrite_clause

```
[ ENABLE | DISABLE ] QUERY REWRITE [ unusable_editions_clause ]
```

alter_session_set_clause

```
SET { { parameter_name = parameter_value }...
    | EDITION = edition_name
    | CONTAINER = container_name
    | ROW ARCHIVAL VISIBILITY = { ACTIVE | ALL }
    }
```

alter_system_reset_clause

```
parameter_name
   [ { SCOPE = SPFILE
     | SID = { 'sid' | '*' }
     }...
   ]
```

alter_system_set_clause

```
{ set_parameter_clause
| USE_STORED_OUTLINES = (TRUE | FALSE | category_name)
| GLOBAL_TOPIC_ENABLED = (TRUE | FALSE)
}
```

alter_table_partitioning

```
{ modify_table_default_attrs
| alter_interval_partitioning
| set_subpartition_template
| modify_table_partition
| modify_table_subpartition
| move_table_partition
| move_table_subpartition
| add_table_partition
| coalesce_table_partition
| drop_table_partition
| drop_table_subpartition
| rename_partition_subpart
| truncate_partition_subpart
| split_table_partition
| split_table_subpartition
| merge_table_partitions
| merge_table_subpartitions
| exchange_partition_subpart
}
```

alter_table_properties

```
{ { { physical_attributes_clause
    | logging_clause
    | table_compression
    | inmemory_alter_table_clause
    | ilm_clause
    | supplemental_table_logging
    | allocate_extent_clause
    | deallocate_unused_clause
    | { CACHE | NOCACHE }
    | RESULT_CACHE ( MODE {DEFAULT | FORCE} )
    | upgrade_table_clause
    | records_per_block_clause
    | parallel_clause
    | row_movement_clause
    | flashback_archive_clause
    }...
  | RENAME TO new_table_name
  } [ alter_iot_clauses ] [ alter_XMLSchema_clause ]
| { shrink_clause
  | READ ONLY
  | READ WRITE
  | REKEY encryption_spec
  | [NO] ROW ARCHIVAL
  | ADD attribute_clustering_clause
  | MODIFY CLUSTERING [ clustering_when ] [ zonemap_clause ]
  | DROP CLUSTERING
  }
}
```

alter_tempfile_clause

```
TEMPFILE
   { 'filename' [, 'filename' ]...
   | filenumber [, filenumber ]...
   }
   { RESIZE size_clause
   | autoextend_clause
   | DROP [ INCLUDING DATAFILES ]
   | ONLINE
   | OFFLINE
   }
```

alter_varray_col_properties

```
MODIFY VARRAY varray_item
   ( modify_LOB_parameters )
```

alter_XMLSchema_clause

```
{ ALLOW ANYSCHEMA
| ALLOW NONSCHEMA
| DISALLOW NONSCHEMA
}
```

alter_zonemap_attributes

```
{ PCTFREE integer
| PCTUSED integer
| { CACHE | NOCACHE }
}...
```

analytic_clause

```
[ query_partition_clause ] [ order_by_clause [ windowing_clause ] ]
```

archive_log_clause

```
ARCHIVE LOG
   [  INSTANCE 'instance_name' ]
   { { SEQUENCE integer
     | CHANGE integer
     | CURRENT [ NOSWITCH ]
     | GROUP integer
     | LOGFILE 'filename'
          [ USING BACKUP CONTROLFILE ]
     | NEXT
     | ALL
     }
     [ TO 'location' ]
   }
```

array_DML_clause

```
[ WITH | WITHOUT ]
ARRAY DML
[ ([ schema. ]type
   [, [ schema. ]varray_type ])
    [, ([ schema. ]type
        [, [ schema. ]varray_type ])...
]
```

array_step

```
[ { integer | integer TO integer [, integer | integer TO integer ]... } | * ]

Note: The outside square brackets shown in boldface type are part of the syntax.
      In this case, they do not represent optionality.
```

ASM_filename

```
{ fully_qualified_file_name
| numeric_file_name
| incomplete_file_name
| alias_file_name
}
```

attribute_clause

```
ATTRIBUTE level DETERMINES
   { dependent_column
   | ( dependent_column
       [, dependent_column ]... )
   }
```

attribute_clustering_clause

```
CLUSTERING [ clustering_join ] cluster_clause
           [ clustering_when ] [ zonemap_clause ]
```

audit_operation_clause

```
{ { sql_statement_shortcut
  | ALL
  | ALL STATEMENTS
  } [, { sql_statement_shortcut
       | ALL
       }
    ]
| { system_privilege
  | ALL PRIVILEGES
  } [, { system_privilege
       | ALL PRIVILEGES
       }
    ]
}
```

audit_schema_object_clause

```
{ sql_operation [, object_option]
| ALL
} auditing_on_clause
```

auditing_by_clause

```
BY user [, user ]...
```

auditing_on_clause

```
ON { [ schema. ] object
   | DIRECTORY directory_name
   | MINING MODEL [ schema. ] model
   | SQL TRANSLATION PROFILE [ schema. ] profile
   | DEFAULT
   }
```

autoextend_clause

```
AUTOEXTEND
   { OFF
   | ON [ NEXT size_clause ]
        [ maxsize_clause ]
   }
```

backup_keystore

```
BACKUP KEYSTORE [ USING 'backup_identifier' ]
  IDENTIFIED BY keystore_password
  [ TO 'keystore_location' ]
```

binding_clause

```
BINDING
   (parameter_type [, parameter_type ]...)
   RETURN return_type
   [ implementation_clause ]
   using_function_clause
    [, (parameter_type [, parameter_type ]...)
       RETURN return_type
       [ implementation_clause ]
       using_function_clause
    ]...
```

bitmap_join_index_clause

```
[ schema.]table
   ( [ [ schema. ]table. | t_alias. ]column
     [ ASC | DESC  ]
```

```
       [, [ [ schema. ]table. | t_alias. ]column
          [ ASC | DESC ]
       ]...
   )
   FROM [ schema. ]table [ t_alias ]
          [, [ schema. ]table [ t_alias ]
        ]...
   WHERE condition
      [ local_partitioned_index ] index_attributes
```

build_clause

```
BUILD { IMMEDIATE | DEFERRED }
```

cell_assignment

```
measure_column [ { { condition
                   | expr
                   | single_column_for_loop
                   }
                     [, { condition
                        | expr
                        | single_column_for_loop
                        }
                     ]...
                 | multi_column_for_loop
                 }
               ]

Note: The outer square brackets are part of the syntax.
      In this case, they do not indicate optionality.
```

cell_reference_options

```
[ { IGNORE | KEEP } NAV ]
[ UNIQUE { DIMENSION | SINGLE REFERENCE } ]
```

character_set_clause

```
CHARACTER SET character_set
```

check_datafiles_clause

```
CHECK DATAFILES [ GLOBAL | LOCAL ]
```

check_diskgroup_clause

```
CHECK [ REPAIR | NOREPAIR ]
```

checkpoint_clause

```
CHECKPOINT [ GLOBAL | LOCAL ]
```

close_keystore

```
SET KEYSTORE CLOSE
  [ IDENTIFIED BY keystore_password ]
  [ CONTAINER = { ALL | CURRENT } ]
```

cluster_clause

```
BY [ LINEAR | INTERLEAVED ] ORDER clustering_columns
```

cluster_index_clause

```
CLUSTER [ schema. ] cluster index_attributes
```

cluster_range_partitions

```
PARTITION BY RANGE (column[, column ]...)
( PARTITION [ partition ]
    range_values_clause table_partition_description
      [, PARTITION [ partition ]
        range_values_clause table_partition_description
      ]...
)
```

clustering_column_group

```
( column [, column ]... )
```

clustering_columns

```
clustering_column_group
| ( clustering_column_group [, clustering_column_group ]... )
```

clustering_join

```
[ schema. ] table JOIN [ schema. ] table ON ( equijoin_condition )
                    [, JOIN [ schema. ] table ON ( equijoin_condition ) ]...
```

clustering_when

```
[ { YES | NO } ON LOAD ] [ { YES | NO } ON DATA MOVEMENT ]
```

coalesce_index_partition

```
COALESCE PARTITION [ parallel_clause ]
```

coalesce_table_partition

```
COALESCE PARTITION
  [ update_index_clauses ]
  [ parallel_clause ]
  [ allow_disallow_clustering ]
```

coalesce_table_subpartition

```
COALESCE SUBPARTITION subpartition
  [update_index_clauses]
  [parallel_clause]
  [allow_disallow_clustering]
```

column_association

```
COLUMNS [ schema. ]table.column
          [, [ schema. ]table.column ]...
   using_statistics_type
```

column_clauses

```
{ { add_column_clause
  | modify_column_clauses
  | drop_column_clause
  | add_period_clause
  | drop_period_clause
  }...
| rename_column_clause
| { modify_collection_retrieval }...
| { modify_LOB_storage_clause }...
| { alter_varray_col_properties }...
}
```

column_definition

```
column datatype [ SORT ] [ VISIBLE | INVISIBLE ]
```

```
  [ DEFAULT [ ON NULL ] expr | identity_clause ]
  [ ENCRYPT encryption_spec ]
  [ ( { inline_constraint }... )
  | inline_ref_constraint
  ]
```

column_properties

```
{ object_type_col_properties
| nested_table_col_properties
| { varray_col_properties | LOB_storage_clause }
    [ (LOB_partition_storage [, LOB_partition_storage ]...) ]
| XMLType_column_properties
}...
```

commit_switchover_clause

```
{ PREPARE | COMMIT } TO SWITCHOVER
[ TO { { [ PHYSICAL | LOGICAL ] PRIMARY
     | [ PHYSICAL ] STANDBY
     } [ { WITH | WITHOUT } SESSION SHUTDOWN
         { WAIT | NOWAIT }
       ]
     | LOGICAL STANDBY
     }
| CANCEL
]
```

component_actions

```
ACTIONS COMPONENT =
  { DATAPUMP | DIRECT_LOAD | OLS | XS } component_action [, component_action ]...
  |
  DV component_action ON object_name [, component_action ON object_name ]...
```

composite_hash_partitions

```
PARTITION BY HASH (column [, column ] ...)
  { subpartition_by_range
  | subpartition_by_list
  | subpartition_by_hash
  }
  { individual_hash_partitions
  | hash_partitions_by_quantity
  }
```

composite_list_partitions

```
PARTITION BY LIST ( column )
  { subpartition_by_range
  | subpartition_by_list
  | subpartition_by_hash
  }
( list_partition_desc [, list_partition_desc]... )
```

composite_range_partitions

```
PARTITION BY RANGE ( column [, column]... )
  [ INTERVAL ( expr ) [ STORE IN ( tablespace [, tablespace]... ) ]]
  { subpartition_by_range
  | subpartition_by_list
  | subpartition_by_hash
  }
( range_partition_desc [, range_partition_desc]... )
```

conditional_insert_clause

```
[ ALL | FIRST ]
```

```
WHEN condition
THEN insert_into_clause
  [ values_clause ]
  [ error_logging_clause ]
  [ insert_into_clause [ values_clause ] [ error_logging_clause ] ]...
[ WHEN condition
  THEN insert_into_clause
    [ values_clause ]
    [ error_logging_clause ]
    [ insert_into_clause [ values_clause ] [ error_logging_clause ] ]...
]...
[ ELSE insert_into_clause
  [ values_clause ]
  [ error_logging_clause ]
   [ insert_into_clause [ values_clause ] [ error_logging_clause ] ]...
]
```

constraint

```
{ inline_constraint
| out_of_line_constraint
| inline_ref_constraint
| out_of_line_ref_constraint
}
```

constraint_clauses

```
{ ADD { { out_of_line_constraint }...
      | out_of_line_REF_constraint
      }
| MODIFY { CONSTRAINT constraint_name
         | PRIMARY KEY
         | UNIQUE (column [, column ]...)
         } constraint_state [ CASCADE ]
| RENAME CONSTRAINT old_name TO new_name
| drop_constraint_clause
}
```

constraint_state

```
[ [ [ NOT ] DEFERRABLE ]
  [ INITIALLY { IMMEDIATE | DEFERRED } ]
| [ INITIALLY { IMMEDIATE | DEFERRED } ]
  [ [ NOT ] DEFERRABLE ]
]
[ RELY | NORELY ]
[ using_index_clause ]
[ ENABLE | DISABLE ]
[ VALIDATE | NOVALIDATE ]
[ exceptions_clause ]
```

container_data_clause

```
{
SET CONTAINER_DATA = { ALL | DEFAULT | ( container_name [, container_name ]... ) }
|
ADD CONTAINER_DATA = ( container_name [, container_name ]... )
|
REMOVE CONTAINER_DATA = ( container_name [, container_name ]... )
}
[ FOR [ schema. ] container_data_object ]
```

containers_clause

```
CONTAINERS( [schema.] { table | view } )
```

context_clause

```
[ WITH INDEX CONTEXT,
  SCAN CONTEXT implementation_type
  [ COMPUTE ANCILLARY DATA ]
]
[ WITH COLUMN CONTEXT ]
```

controlfile_clauses

```
CREATE { [ LOGICAL | PHYSICAL ] STANDBY | FAR SYNC INSTANCE }
  CONTROLFILE AS
  'filename' [ REUSE ]
| BACKUP CONTROLFILE TO
  { 'filename' [ REUSE ]
  | trace_file_clause
  }
```

convert_database_clause

```
CONVERT TO ( PHYSICAL | SNAPSHOT ) STANDBY
```

cost_matrix_clause

```
COST
  { MODEL [AUTO]
  | ( class_value [, class_value]... )
        VALUES ( ( cost_value [, cost_value]...)
                 [ , (cost_value [, cost_value]... ) ]...
               )
  }
```

create_datafile_clause

```
CREATE DATAFILE
   { 'filename' | filenumber }
     [, 'filename' | filenumber ]...
   }
   [ AS { file_specification
          [, file_specification ]...
        | NEW
        }
   ]
```

create_file_dest_clause

```
CREATE_FILE_DEST = { NONE | 'directory_path_name' | diskgroup_name }
```

create_key

```
CREATE [ ENCRYPTION ] KEY [ USING TAG 'tag' ]
  IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
  [ CONTAINER = { ALL | CURRENT } ]
```

create_keystore

```
CREATE
  { KEYSTORE 'keystore_location'
  | [ LOCAL ] AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location'
  }
  IDENTIFIED BY keystore_password
```

create_mv_refresh

```
{ REFRESH
  { { FAST | COMPLETE | FORCE }
  | { ON DEMAND
    | ON COMMIT
    }
  | { START WITH date |
      NEXT date
    }...
  | WITH { PRIMARY KEY | ROWID }
  | USING
     { DEFAULT [ MASTER | LOCAL ] ROLLBACK SEGMENT
     | [ MASTER | LOCAL ] ROLLBACK SEGMENT rollback_segment
     }...
  | USING
     { ENFORCED | TRUSTED } CONSTRAINTS
  }...
| NEVER REFRESH
}
```

create_pdb_clone

```
FROM { src_pdb_name [ @ dblink ] } | { NON$CDB @ dblink }
  [ pdb_storage_clause ]
  [ file_name_convert ]
  [ path_prefix_clause ]
  [ tempfile_reuse_clause ]
  [ SNAPSHOT COPY ]
  [ user_tablespaces_clause ]
  [ standbys_clause ]
  [ logging_clause ]
  [ create_file_dest_clause ]
  [ NO DATA ]
```

create_pdb_from_seed

```
ADMIN USER admin_user_name IDENTIFIED BY password
  [ pdb_dba_roles ]
  [ default_tablespace ]
  [ file_name_convert ]
  [ pdb_storage_clause ]
  [ path_prefix_clause ]
  [ tempfile_reuse_clause ]
  [ user_tablespaces_clause ]
  [ standbys_clause ]
  [ logging_clause ]
  [ create_file_dest_clause ]
```

create_pdb_from_xml

```
[ AS CLONE ] USING filename [ source_file_name_convert ]
  [ { [ COPY | MOVE ] file_name_convert } | NOCOPY ]
  [ pdb_storage_clause ]
  [ path_prefix_clause ]
  [ tempfile_reuse_clause ]
  [ user_tablespaces_clause ]
  [ standbys_clause ]
  [ logging_clause ]
  [ create_file_dest_clause ]
```

create_zonemap_as_subquery

```
CREATE MATERIALIZED ZONEMAP
  [ schema. ] zonemap_name
  [ ( column_alias [, column_alias ]... ) ]
  [ zonemap_attributes ]
  [ zonemap_refresh_clause ]
  [ { ENABLE | DISABLE } PRUNING ]
  AS query_block
```

create_zonemap_on_table

```
CREATE MATERIALIZED ZONEMAP
  [ schema. ] zonemap_name
  [ zonemap_attributes ]
  [ zonemap_refresh_clause ]
  [ { ENABLE | DISABLE } PRUNING ]
  ON [ schema. ] { table | materialized_view } ( column [, column]... )
```

cross_outer_apply_clause

```
{ CROSS | OUTER } APPLY { table_reference | collection_expression }
```

cycle_clause

```
{CYCLE c_alias [, c_alias]...
    SET cycle_mark_c_alias TO cycle_value
    DEFAULT no_cycle_value
}
```

database_file_clauses

```
{ RENAME FILE  'filename' [, 'filename' ]...
   TO 'filename'
| create_datafile_clause
| alter_datafile_clause
| alter_tempfile_clause
| move_datafile_clause
}
```

database_logging_clauses

```
{ LOGFILE
    [ GROUP integer ] file_specification
      [, [ GROUP integer ] file_specification ]...
| MAXLOGFILES integer
| MAXLOGMEMBERS integer
| MAXLOGHISTORY integer
| { ARCHIVELOG | NOARCHIVELOG }
| FORCE LOGGING
}
```

datafile_tempfile_clauses

```
{ ADD { DATAFILE | TEMPFILE }
   [ file_specification [, file_specification ]... ]
| DROP {DATAFILE | TEMPFILE } { 'filename' | file_number }
| SHRINK TEMPFILE { 'filename' | file_number } [KEEP size_clause]
| RENAME DATAFILE 'filename' [, 'filename' ]...
    TO 'filename' [, 'filename' ]...
| { DATAFILE | TEMPFILE } { ONLINE | OFFLINE }
}
```

datafile_tempfile_spec

```
[ 'filename' | 'ASM_filename' ]
[ SIZE size_clause ]
[ REUSE ]
[ autoextend_clause ]
```

db_user_proxy_clauses

```
[ WITH
  { ROLE { role_name [, role_name]...
         | ALL EXCEPT role_name [, role_name]...
         }
  | NO ROLES
  }
]
[ AUTHENTICATION REQUIRED ]
```

dblink

```
database[.domain [.domain ]... ] [ @ connection_qualifier ]
```

dblink_authentication

```
AUTHENTICATED BY user IDENTIFIED BY password
```

deallocate_unused_clause

```
DEALLOCATE UNUSED [ KEEP size_clause ]
```

default_cost_clause

```
DEFAULT COST (cpu_cost, io_cost, network_cost)
```

default_selectivity_clause

```
DEFAULT SELECTIVITY default_selectivity
```

default_settings_clauses

```
{ DEFAULT EDITION = edition_name
| SET DEFAULT { BIGFILE | SMALLFILE } TABLESPACE
| DEFAULT TABLESPACE tablespace
| DEFAULT TEMPORARY TABLESPACE { tablespace | tablespace_group_name }
| RENAME GLOBAL_NAME TO database.domain [.domain ]...
| ENABLE BLOCK CHANGE TRACKING [ USING FILE 'filename' [ REUSE ] ]
| DISABLE BLOCK CHANGE TRACKING
| [NO] FORCE FULL DATABASE CACHING
| flashback_mode_clause
| set_time_zone_clause
}
```

default_tablespace

```
DEFAULT TABLESPACE tablespace
[ DATAFILE datafile_tempfile_spec ]
[ extent_management_clause ]
```

default_temp_tablespace

```
  [ BIGFILE | SMALLFILE ]
DEFAULT TEMPORARY TABLESPACE tablespace
  [ TEMPFILE file_specification [, file_specification ]...]
  [ extent_management_clause ]
```

deferred_segment_creation

```
SEGMENT CREATION { IMMEDIATE | DEFERRED }
```

delete_secret

```
DELETE SECRET FOR CLIENT 'client_identifier'
  IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

dependent_tables_clause

```
DEPENDENT TABLES
( table ( partition_spec [, partition_spec]...
          [, table ( partition_spec [, partition_spec]... ]
        )
)
```

dimension_join_clause

```
{ JOIN KEY
   { child_key_column
   | (child_key_column [, child_key_column ]...)
   }
  REFERENCES parent_level
}...
```

disk_offline_clause

```
OFFLINE
  { [QUORUM | REGULAR] DISK disk_name [, disk_name ] ...
  | DISKS IN [QUORUM | REGULAR] FAILGROUP failgroup_name [, failgroup_name ]...
  } ... [timeout_clause]
```

disk_online_clause

```
ONLINE
  { { [QUORUM | REGULAR] DISK disk_name [, disk_name]...
    | DISKS IN [QUORUM | REGULAR] FAILGROUP failgroup_name [, failgroup_name]...
    } ...
  | ALL
  } [ POWER integer ] [ WAIT | NOWAIT ]
```

disk_region_clause

```
[ HOT | COLD ] [ MIRRORHOT | MIRRORCOLD ]
```

diskgroup_alias_clauses

```
{ ADD ALIAS
    'alias_name' FOR 'filename'
    [, 'alias_name' FOR 'filename' ]...
| DROP ALIAS 'alias_name' [, 'alias_name' ]...
| RENAME ALIAS
    'old_alias_name' TO 'new_alias_name'
    [, 'old_alias_name' TO 'new_alias_name' ]...
}
```

diskgroup_attributes

```
SET ATTRIBUTE 'attribute_name' = 'attribute_value'
```

diskgroup_availability

```
{ MOUNT [ RESTRICTED | NORMAL ]
          [ FORCE | NOFORCE ]
| DISMOUNT [ FORCE | NOFORCE ]
}
```

diskgroup_directory_clauses

```
{ ADD DIRECTORY 'filename' [, 'filename' ]...
| DROP DIRECTORY
    'filename' [ FORCE | NOFORCE ]
    [, 'filename' [ FORCE | NOFORCE ] ]...
| RENAME DIRECTORY
    'old_dir_name' TO 'new_dir_name'
    [, 'old_dir_name' TO 'new_dir_name' ]...
}
```

diskgroup_template_clauses

```
{ { ADD | MODIFY } TEMPLATE template_name qualified_template_clause
      [, template_name qualified_template_clause ]...
| DROP TEMPLATE template_name [, template_name ]...
}
```

diskgroup_volume_clauses

```
{ add_volume_clause
| modify_volume_clause
| RESIZE VOLUME asm_volume SIZE size_clause
| DROP VOLUME asm_volume
}
```

distributed_recov_clauses

```
{ ENABLE | DISABLE } DISTRIBUTED RECOVERY
```

dml_table_expression_clause

```
{ [ schema. ]
  { table
    [ partition_extension_clause
    | @ dblink
    ]
  | { view | materialized view } [ @ dblink ]
  }
| ( subquery [ subquery_restriction_clause ] )
| table_collection_expression
}
```

domain_index_clause

```
indextype
   [ local_domain_index_clause ]
   [ parallel_clause ]
   [ PARAMETERS ('ODCI_parameters') ]
```

drop_binding_clause

```
DROP BINDING (parameter_type [, parameter_type ]...)
  [ FORCE ]
```

drop_column_clause

```
{ SET UNUSED { COLUMN column
             | (column [, column ]...)
             }
  [ { CASCADE CONSTRAINTS | INVALIDATE }... ]
  [ ONLINE ]
| DROP { COLUMN column
       | (column [, column ]...)
       }
  [ { CASCADE CONSTRAINTS | INVALIDATE }... ]
  [ CHECKPOINT integer ]
| DROP { UNUSED COLUMNS
       | COLUMNS CONTINUE
       }
  [ CHECKPOINT integer ]
}
```

drop_constraint_clause

```
DROP
   { { PRIMARY KEY
     | UNIQUE (column [, column ]...)
     }
     [ CASCADE ]
     [ { KEEP | DROP } INDEX ]
   | CONSTRAINT constraint_name
     [ CASCADE ]
   } [ ONLINE ]
```

drop_disk_clause

```
DROP
{ [QUORUM | REGULAR] DISK
    disk_name [ FORCE | NOFORCE ]
    [, disk_name [ FORCE | NOFORCE ] ]...
| DISKS IN [QUORUM | REGULAR] FAILGROUP
    failgroup_name [ FORCE | NOFORCE ]
    [, failgroup_name [ FORCE | NOFORCE ] ]...
}
```

drop_diskgroup_file_clause

```
DROP FILE 'filename' [, 'filename' ]...
```

drop_index_partition

```
DROP PARTITION partition_name
```

drop_logfile_clauses

```
DROP [ STANDBY ] LOGFILE
   { logfile_descriptor
     [, logfile_descriptor ]...
   | MEMBER 'filename'
           [, 'filename' ]...
   }
```

drop_period_clause

```
DROP ( PERIOD FOR valid_time_column )
```

drop_table_partition

```
DROP partition_extended_names
  [ update_index_clauses [ parallel_clause ] ]
```

drop_table_subpartition

```
DROP subpartition_extended_names
  [ update_index_clauses [ parallel_clause ] ]
```

ds_iso_format

```
[-] P [days D]
  [T [hours H] [minutes M] [seconds [. frac_secs] S ] ]
```

else_clause

```
ELSE else_expr
```

enable_disable_clause

```
{ ENABLE | DISABLE }
[ VALIDATE | NOVALIDATE ]
{ UNIQUE (column [, column ]...)
| PRIMARY KEY
| CONSTRAINT constraint_name
}
[ using_index_clause ]
[ exceptions_clause ]
[ CASCADE ]
[ { KEEP | DROP } INDEX ]
```

enable_disable_volume

```
{ ENABLE | DISABLE } VOLUME
  { asm_volume [, asm_volume]...
  | ALL
  }
```

enable_pluggable_database

```
ENABLE PLUGGABLE DATABASE
  [ SEED
  [ file_name_convert ]
  [ SYSTEM tablespace_datafile_clauses ]
  [ SYSAUX tablespace_datafile_clauses ]
  ]
```

encryption_spec

```
  [ USING 'encrypt_algorithm' ]
  [ IDENTIFIED BY password ]
  [ 'integrity_algorithm' ]
  [ [ NO ] SALT ]
```

end_session_clauses

```
{ DISCONNECT SESSION 'integer1, integer2'
     [ POST_TRANSACTION ]
| KILL SESSION 'integer1, integer2 [, @integer3]'
}
[ IMMEDIATE | NOREPLAY ]
```

error_logging_clause

```
LOG ERRORS
  [ INTO [schema.] table ]
  [ (simple_expression) ]
  [ REJECT LIMIT { integer | UNLIMITED } ]
```

evaluation_edition_clause

```
EVALUATE USING { CURRENT EDITION | EDITION edition | NULL EDITION }
```

exceptions_clause

```
EXCEPTIONS INTO [ schema. ] table
```

exchange_partition_subpart

```
EXCHANGE { partition_extended_name
         | subpartition_extended_name
         }
   WITH TABLE [ schema. ] table
   [ { INCLUDING | EXCLUDING } INDEXES ]
   [ { WITH | WITHOUT } VALIDATION ]
   [ exceptions_clause ]
   [ update_index_clauses [ parallel_clause ] ]
   [ CASCADE ]
```

export_keys

```
EXPORT [ ENCRYPTION ] KEYS WITH SECRET secret
  TO 'filename'
  IDENTIFIED BY keystore_password
  [ WITH IDENTIFIER IN { 'key_id' [, 'key_id' ]... | ( subquery ) } ]
```

expr

```
{ simple_expression
| compound_expression
| case_expression
| cursor_expression
| datetime_expression
| function_expression
| interval_expression
| JSON_object_access_expr
| model_expression
| object_access_expression
| scalar_subquery_expression
| type_constructor_expression
| variable_expression
}
```

expression_list

```
{ expr [, expr ]...
| ( [expr [, expr ]] ...)
}
```

extended_attribute_clause

```
ATTRIBUTE attribute
  { LEVEL level
    DETERMINES { dependent_column
               | (dependent_column [, dependent_column ]... )
               }
  }...
```

extent_management_clause

```
EXTENT MANAGEMENT LOCAL
  [ AUTOALLOCATE
  | UNIFORM [ SIZE size_clause ]
  ]
```

external_data_properties

```
DEFAULT DIRECTORY directory
[ ACCESS PARAMETERS
  { (opaque_format_spec)
  | USING CLOB subquery
  }
]
LOCATION
   ([ directory: ] 'location_specifier'
      [, [ directory: ] 'location_specifier' ]...
   )
```

external_table_clause

```
([ TYPE access_driver_type ]
 external_data_properties
)
[ REJECT LIMIT { integer | UNLIMITED } ]
```

failover_clause

```
FAILOVER TO target_db_name [ FORCE ]
```

file_name_convert

```
FILE_NAME_CONVERT =
  { ( 'filename_pattern', 'replacement_filename_pattern'
      [, 'filename_pattern', 'replacement_filename_pattern' ]... )
    |
    NONE
  }
```

file_owner_clause

```
SET OWNERSHIP { OWNER = user | GROUP = usergroup
                  [, OWNER = user | GROUP = usergroup ]...
```

```
              } FOR FILE 'filename' [, 'filename']...
```

file_permissions_clause

```
SET PERMISSION { OWNER | GROUP | OTHER }
  = { NONE | READ ONLY | READ WRITE }
  [, { OWNER | GROUP | OTHER | ALL }
    = { NONE | READ ONLY | READ WRITE } ]...
    FOR FILE 'filename' [, 'filename']...
```

file_specification

```
{ datafile_tempfile_spec
| redo_log_file_spec
}
```

flashback_archive_clause

```
FLASHBACK ARCHIVE [flashback_archive] | NO FLASHBACK ARCHIVE
```

flashback_archive_quota

```
QUOTA integer { M | G | T | P | E }
```

flashback_archive_retention

```
RETENTION integer {YEAR | MONTH | DAY}
```

flashback_mode_clause

```
FLASHBACK { ON | OFF }
```

flashback_query_clause

```
{ VERSIONS BETWEEN { SCN | TIMESTAMP }
    { expr | MINVALUE } AND { expr | MAXVALUE }
| VERSIONS PERIOD FOR valid_time_column BETWEEN
    { expr | MINVALUE } AND { expr | MAXVALUE }
| AS OF { SCN | TIMESTAMP } expr
| AS OF PERIOD FOR valid_time_column expr
}
```

for_refresh_clause

```
{ FOR SYNCHRONOUS REFRESH USING staging_log_name
| FOR FAST REFRESH
}
```

for_update_clause

```
FOR UPDATE
  [ OF [ [ schema. ] { table | view } . ] column
         [, [ [ schema. ] { table | view } . ] column
         ]...
  ]
  [ { NOWAIT | WAIT integer
    |  SKIP LOCKED
    }
  ]
```

full_database_recovery

```
[ STANDBY ] DATABASE
[ { UNTIL { CANCEL
          | TIME date
          | CHANGE integer
          | CONSISTENT
          }
```

```
  | USING BACKUP CONTROLFILE
  | SNAPSHOT TIME date
  }...
]
```

fully_qualified_file_name

```
+diskgroup_name/db_name/file_type/
   file_type_tag.filenumber.incarnation_number
```

function_association

```
{ FUNCTIONS
     [ schema. ]function [, [ schema. ]function ]...
| PACKAGES
     [ schema. ]package [, [ schema. ]package ]...
| TYPES
     [ schema. ]type [, [ schema. ]type ]...
| INDEXES
     [ schema. ]index [, [ schema. ]index ]...
| INDEXTYPES
     [ schema. ]indextype [, [ schema. ]indextype ]...
}
{ using_statistics_type
| { default_cost_clause [, default_selectivity_clause ]
  | default_selectivity_clause [, default_cost_clause ]
  }
}
```

general_recovery

```
RECOVER
[ AUTOMATIC ]
[ FROM 'location' ]
{ { full_database_recovery
  | partial_database_recovery
  | LOGFILE 'filename'
  }
  [ { TEST
    | ALLOW integer CORRUPTION
    | parallel_clause
    }...
  ]
| CONTINUE [ DEFAULT ]
| CANCEL
}
```

global_partitioned_index

```
GLOBAL PARTITION BY
   { RANGE (column_list)
        (index_partitioning_clause)
   | HASH (column_list)
        { individual_hash_partitions
        | hash_partitions_by_quantity
        }
   }
```

grant_object_privileges

```
{ object_privilege | ALL [ PRIVILEGES ] }
  [ (column [, column ]...) ]
    [, { object_privilege | ALL [ PRIVILEGES ] }
       [ (column [, column ]...) ]
    ]...
on_object_clause
TO grantee_clause
```

```
  [ WITH HIERARCHY OPTION ]
  [ WITH GRANT OPTION ]
```

grant_roles_to_programs

```
role [, role ]... TO program_unit [, program_unit ]...
```

grant_system_privileges

```
{ system_privilege | role | ALL PRIVILEGES }
  [, { system_privilege | role | ALL PRIVILEGES } ]...
TO { grantee_clause | grantee_identified_by } [ WITH { ADMIN | DELEGATE } OPTION ]
```

grantee_clause

```
{ user | role | PUBLIC }
  [, { user | role | PUBLIC } ]...
```

grantee_identified_by

```
user [, user ]... IDENTIFIED BY password [, password ]...
```

group_by_clause

```
GROUP BY
   { expr
   | rollup_cube_clause
   | grouping_sets_clause
   }
     [, { expr
        | rollup_cube_clause
        | grouping_sets_clause
        }
     ]...
   [ HAVING condition ]
```

grouping_expression_list

```
expression_list [, expression_list ]...
```

grouping_sets_clause

```
GROUPING SETS
({ rollup_cube_clause | grouping_expression_list })
```

hash_partitions

```
PARTITION BY HASH (column [, column ] ...)
{ individual_hash_partitions
| hash_partitions_by_quantity
}
```

hash_partitions_by_quantity

```
PARTITIONS hash_partition_quantity
[ STORE IN (tablespace [, tablespace ]...) ]
[ table_compression | index_compression ]
[ OVERFLOW STORE IN (tablespace [, tablespace ]...) ]
```

hash_subparts_by_quantity

```
SUBPARTITIONS integer [STORE IN ( tablespace [, tablespace]... )]
```

heap_org_table_clause

```
[ table_compression ] [ inmemory_table_clause ] [ ilm_clause ]
```

hierarchical_query_clause

```
{ CONNECT BY [ NOCYCLE ] condition [ START WITH condition ]
| START WITH condition CONNECT BY [ NOCYCLE ] condition
}
```

hierarchy_clause

```
HIERARCHY hierarchy
(child_level { CHILD OF parent_level }...
  [ dimension_join_clause ]
)
```

identity_clause

```
GENERATED
[ ALWAYS | BY DEFAULT [ ON NULL ] ]
AS IDENTITY [ ( identity_options ) ]
```

identity_options

```
{ START WITH ( integer | LIMIT VALUE )
| INCREMENT BY integer
| ( MAXVALUE integer | NOMAXVALUE )
| ( MINVALUE integer | NOMINVALUE )
| ( CYCLE | NOCYCLE )
| ( CACHE integer | NOCACHE )
| ( ORDER | NOORDER ) }...
```

ilm_clause

```
ILM
{ ADD POLICY ilm_policy_clause
| { DELETE | ENABLE | DISABLE } POLICY ilm_policy_name
| DELETE_ALL | ENABLE_ALL | DISABLE_ALL
}
```

ilm_policy_clause

```
{ table_compression | tiering_clause }
{ SEGMENT | GROUP | ROW }
{{ AFTER number { { DAY | DAYS } | { MONTH | MONTHS } | { YEAR | YEARS } }
   OF { { NO ACCESS } | { NO MODIFICATION } | CREATION } }
 | ON function_name }
```

implementation_clause

```
{ ANCILLARY TO primary_operator
    ( parameter_type [, parameter_type ]...)
      [, primary_operator
         ( parameter_type [, parameter_type ]...)
      ]...
| context_clause
}
```

import_keys

```
IMPORT [ ENCRYPTION ] KEYS WITH SECRET secret
  FROM 'filename'
  IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

incomplete_file_name

```
+diskgroup_name [ (template_name) ]
```

index_attributes

```
[ { physical_attributes_clause
```

```
  | logging_clause
  | ONLINE
  | TABLESPACE { tablespace | DEFAULT }
  | index_compression
  | { SORT | NOSORT }
  | REVERSE
  | VISIBLE | INVISIBLE
  | partial_index_clause
  | parallel_clause
  }...
]
```

index_compression

```
{ prefix_compression
| advanced_index_compression
}
```

index_expr

```
{ column | column_expression }
```

index_org_overflow_clause

```
  [ INCLUDING column_name ]
OVERFLOW [ segment_attributes_clause ]
```

index_org_table_clause

```
[ { mapping_table_clause
  | PCTTHRESHOLD integer
  | prefix_compression
  }...
]
[ index_org_overflow_clause ]
```

index_partition_description

```
PARTITION
[ partition
   [ { segment_attributes_clause
     | index_compression
     }...
   | PARAMETERS ( 'ODCI_parameters' )
   ]
   [ USABLE | UNUSABLE ]
]
```

index_partitioning_clause

```
PARTITION [ partition ]
   VALUES LESS THAN (literal[, literal]... )
   [ segment_attributes_clause ]
```

index_properties

```
[ { { global_partitioned_index
    | local_partitioned_index
    }
  | index_attributes
  }...
| INDEXTYPE IS { domain_index_clause
               | XMLTable_index_clause
               | XMLIndex_clause
               }
]
```

index_subpartition_clause

```
{ STORE IN (tablespace[, tablespace ]...)
| (SUBPARTITION
      [ subpartition ][ TABLESPACE tablespace ] [ index_compression ] [ USABLE | UNUSABLE ]
   [, SUBPARTITION
         [ subpartition ][ TABLESPACE tablespace ] [ index_compression ] [ USABLE | UNUSABLE
]
   ]...
  )
}
```

indexing_clause

```
INDEXING { ON | OFF }
```

individual_hash_partitions

```
PARTITION [partition] [indexing_clause] [partitioning_storage_clause]
  [, PARTITION [partition] [indexing_clause] [partitioning_storage_clause]]...
```

individual_hash_subparts

```
SUBPARTITION [subpartition] [indexing_clause] [partitioning_storage_clause]
```

inline_constraint

```
[ CONSTRAINT constraint_name ]
{ [ NOT ] NULL
| UNIQUE
| PRIMARY KEY
| references_clause
| CHECK (condition)
}
[ constraint_state ]
```

inline_ref_constraint

```
{ SCOPE  IS [ schema. ] scope_table
| WITH ROWID
| [ CONSTRAINT constraint_name ]
  references_clause
  [ constraint_state ]
}
```

inmemory_alter_table_clause

```
[ INMEMORY [ inmemory_parameters ] ] [ inmemory_column_clause ]
| NO INMEMORY
```

inmemory_clause

```
INMEMORY [ inmemory_parameters ]
| NO INMEMORY
```

inmemory_column_clause

```
{ INMEMORY [ inmemory_memcompress ] | NO INMEMORY } ( column [, column ]... )
 [ { INMEMORY [ inmemory_memcompress ] | NO INMEMORY } ( column [, column ]... ) ]...
```

inmemory_distribute

```
DISTRIBUTE [ AUTO | BY { ROWID RANGE | PARTITION | SUBPARTITION } ]
```

inmemory_duplicate

```
DUPLICATE | DUPLICATE ALL | NO DUPLICATE
```

inmemory_memcompress

```
MEMCOMPRESS FOR { DML | QUERY [ LOW | HIGH ] | CAPACITY [ LOW | HIGH ] }
| NO MEMCOMPRESS
```

inmemory_parameters

```
[ inmemory_memcompress ] [ inmemory_priority ] [ inmemory_distribute ] [ inmemory_duplicate ]
```

inmemory_priority

```
PRIORITY { NONE | LOW | MEDIUM | HIGH | CRITICAL }
```

inmemory_table_clause

```
INMEMORY [ inmemory_parameters ] [ inmemory_column_clause ]
| NO INMEMORY
```

inner_cross_join_clause

```
{ [ INNER ] JOIN table_reference
    { ON condition
    | USING (column [, column ]...)
    }
| { CROSS
  | NATURAL [ INNER ]
  }
  JOIN table_reference
}
```

insert_into_clause

```
INTO dml_table_expression_clause [ t_alias ]
[ (column [, column ]...) ]
```

instance_clauses

```
{ ENABLE | DISABLE } INSTANCE 'instance_name'
```

instances_clause

```
INSTANCES = { ( 'instance_name' [, 'instance_name' ]... )
            | ALL [ EXCEPT ( 'instance_name' [, 'instance_name' ]... ) ] }
```

integer

```
[ + | - ] digit [ digit ]...
```

interval_day_to_second

```
INTERVAL '{ integer | integer time_expr | time_expr }'
{ { DAY | HOUR | MINUTE } [ (leading_precision) ]
| SECOND [ (leading_precision [, fractional_seconds_precision ]) ]
}
[ TO { DAY | HOUR | MINUTE | SECOND [ (fractional_seconds_precision) ] } ]
```

interval_year_to_month

```
INTERVAL 'integer [- integer ]'
{ YEAR | MONTH } [ (precision) ] [ TO { YEAR | MONTH } ]
```

into_clause

```
INTO [ schema. ] table
```

invoker_rights_clause

```
AUTHID { CURRENT_USER | DEFINER }
```

join_clause

```
table_reference
  { inner_cross_join_clause | outer_join_clause | cross_outer_apply_clause }...
```

JSON_column_definition

```
JSON_exists_column
| JSON_query_column
| JSON_value_column
| JSON_nested_path
| ordinality_column
```

JSON_columns_clause

```
COLUMNS ( JSON_column_definition [, JSON_column_definition ]... )
```

JSON_exists_column

```
column_name JSON_value_return_type
  EXISTS PATH JSON_path_expression [ JSON_exists_on_error_clause ]
```

JSON_exists_on_error_clause

```
{ ERROR | TRUE | FALSE } ON ERROR
```

JSON_nested_path

```
NESTED PATH JSON_path_expression JSON_columns_clause
```

JSON_path_expression

```
$[ object_step | array_step ]...
```

JSON_query_column

```
column_name JSON_query_return_type
  FORMAT JSON [ JSON_query_wrapper_clause ]
  PATH JSON_path_expression [ JSON_query_on_error_clause ]
```

JSON_query_on_error_clause

```
{ ERROR | NULL | EMPTY } ON ERROR
```

JSON_query_return_type

```
VARCHAR2 [ ( size [BYTE | CHAR] ) ]
```

JSON_query_returning_clause

```
[ RETURNING JSON_query_return_type ] [ PRETTY ] [ ASCII ]
```

JSON_query_wrapper_clause

```
WITHOUT [ ARRAY ] WRAPPER
| WITH [ UNCONDITIONAL | CONDITIONAL ] [ ARRAY ] WRAPPER
```

JSON_table_on_error_clause

```
{ ERROR | NULL | DEFAULT literal } ON ERROR
```

JSON_value_column

```
column_name JSON_value_return_type PATH JSON_path_expression
  [ JSON_value_on_error_clause ]
```

JSON_value_on_error_clause

```
{ ERROR | NULL | DEFAULT literal } ON ERROR
```

JSON_value_return_type

```
{ VARCHAR2 [ ( size [BYTE | CHAR] ) ]
| NUMBER [ ( precision [, scale] ) ]
}
```

JSON_value_returning_clause

```
[ RETURNING JSON_value_return_type ] [ ASCII ]
```

key_management_clauses

```
{ set_key
| create_key
| use_key
| set_key_tag
| export_keys
| import_keys
| migrate_key
| reverse_migrate_key
}
```

keystore_management_clauses

```
{ create_keystore
| open_keystore
| close_keystore
| backup_keystore
| alter_keystore_password
| merge_into_new_keystore
| merge_into_exist_keystore
}
```

level_clause

```
LEVEL level IS
   { level_table.level_column
   | (level_table.level_column
      [, level_table.level_column ]...
     )
   }
```

list_partition_desc

```
PARTITION [partition]
list_values_clause
table_partition_description
  [ ( range_subpartition_desc [, range_subpartition_desc]...
      | list_subpartition_desc, [, list_subpartition_desc]...
      | individual_hash_subparts [, individual_hash_subparts]...
    )
    | hash_subparts_by_quantity
  ]
```

list_partitions

```
PARTITION BY LIST (column)
(PARTITION [ partition ]
    list_values_clause table_partition_description
  [, PARTITION [ partition ]
        list_values_clause table_partition_description
  ]...
)
```

list_subpartition_desc

```
SUBPARTITION [subpartition]
  list_values_clause
  [indexing_clause] [partitioning_storage_clause]
```

list_values_clause

```
VALUES ({ literal | NULL }
        [, { literal | NULL }]...
       | DEFAULT
       )
```

LOB_compression_clause

```
{ COMPRESS [HIGH | MEDIUM | LOW ]
| NOCOMPRESS
}
```

LOB_deduplicate_clause

```
{ DEDUPLICATE
| KEEP_DUPLICATES
}
```

LOB_parameters

```
{ { ENABLE | DISABLE } STORAGE IN ROW
  | CHUNK integer
  | PCTVERSION integer
  | FREEPOOLS integer
  | LOB_retention_clause
  | LOB_deduplicate_clause
  | LOB_compression_clause
  | { ENCRYPT encryption_spec | DECRYPT }
  | { CACHE | NOCACHE | CACHE READS } [ logging_clause ]
}...
```

LOB_partition_storage

```
PARTITION partition
{ LOB_storage_clause | varray_col_properties }...
  [ (SUBPARTITION subpartition
     { LOB_partitioning_storage | varray_col_properties }...
    )
]
```

LOB_partitioning_storage

```
LOB (LOB_item) STORE AS [BASICFILE | SECUREFILE]
  [ LOB_segname [ (TABLESPACE tablespace) ]
  | (TABLESPACE tablespace)
  ]
```

LOB_retention_storage

```
RETENTION [ MAX | MIN integer | AUTO | NONE ]
```

LOB_storage_clause

```
LOB
{ (LOB_item [, LOB_item ]...)
     STORE AS { {SECUREFILE | BASICFILE}
              | (LOB_storage_parameters)
              }...
| (LOB_item)
     STORE AS { {SECUREFILE | BASICFILE}
              | LOB_segname
              | (LOB_storage_parameters)
              }...
}
```

LOB_storage_parameters

```
{ { TABLESPACE tablespace
  | LOB_parameters [storage_clause]
  }...
| storage_clauase
}
```

local_domain_index_clause

```
LOCAL
  [ ( PARTITION partition [ PARAMETERS ( 'ODCI_parameters' ) ]
      [,  PARTITION partition [ PARAMETERS ('ODCI_parameters') ]]...
    )
  ]
```

local_partitioned_index

```
LOCAL
[ on_range_partitioned_table
| on_list_partitioned_table
| on_hash_partitioned_table
| on_comp_partitioned_table
]
```

local_XMLIndex_clause

```
LOCAL
  [ ( PARTITION partition [ XMLIndex_parameters_clause ]
      [, PARTITION partition [ XMLIndex_parameters_clause ] ]...
    )
  ]
```

logfile_clause

```
LOGFILE
[ GROUP integer ] file_specification
  [, [ GROUP integer ] file_specification ]...
```

logfile_clauses

```
{ { ARCHIVELOG [ MANUAL ]
  | NOARCHIVELOG
  }
| [ NO ] FORCE LOGGING
| RENAME FILE 'filename' [, 'filename' ]...
    TO 'filename'
| CLEAR [ UNARCHIVED ]
    LOGFILE logfile_descriptor [, logfile_descriptor ]...
    [ UNRECOVERABLE DATAFILE ]
| add_logfile_clauses
| drop_logfile_clauses
| switch_logfile_clause
| supplemental_db_logging
}
```

logfile_descriptor

```
{ GROUP integer
| ('filename' [, 'filename' ]...)
| 'filename'
}
```

logging_clause

```
{ LOGGING | NOLOGGING |  FILESYSTEM_LIKE_LOGGING }
```

main_model

```
[ MAIN main_model_name ]
model_column_clauses
[ cell_reference_options ]
model_rules_clause
```

managed_standby_recovery

```
RECOVER
{ MANAGED STANDBY DATABASE
    [ { USING ARCHIVED LOGFILE
      | DISCONNECT [FROM SESSION]
      | NODELAY
      | UNTIL CHANGE integer
      | UNTIL CONSISTENT
      | parallel_clause
      }...
    | FINISH
    | CANCEL
    ]
| TO LOGICAL STANDBY { db_name | KEEP IDENTITY }
}
```

mapping_table_clauses

```
{ MAPPING TABLE | NOMAPPING }
```

materialized_view_props

```
[ column_properties ]
[ table_partitioning_clauses ]
[ CACHE | NOCACHE ]
[ parallel_clause ]
[ build_clause ]
```

maximize_standby_db_clause

```
SET STANDBY DATABASE TO MAXIMIZE
{ PROTECTION | AVAILABILITY | PERFORMANCE }
```

maxsize_clause

```
MAXSIZE { UNLIMITED | size_clause }
```

merge_insert_clause

```
WHEN NOT MATCHED THEN
INSERT [ (column [, column ]...) ]
VALUES ({ expr | DEFAULT }
          [, { expr | DEFAULT } ]...
       )
[ where_clause ]
```

merge_into_exist_keystore

```
MERGE KEYSTORE 'keystore1_location' [ IDENTIFIED BY keystore1_password ]
  INTO EXISTING KEYSTORE 'keystore2_location' IDENTIFIED BY keystore2_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

merge_into_new_keystore

```
MERGE KEYSTORE 'keystore1_location' [ IDENTIFIED BY keystore1_password ]
  AND KEYSTORE 'keystore2_location' [ IDENTIFIED BY keystore2_password ]
  INTO NEW KEYSTORE 'keystore3_location' IDENTIFIED BY keystore3_password
```

merge_table_partitions

```
MERGE PARTITIONS partition_or_key_value
```

```
   { , partition_or_key_value [, partition_or_key_value ]...
   | TO partition_or_key_value }
   [ INTO partition_spec ]
   [ dependent_tables_clause ]
   [ update_index_clauses ]
   [ parallel_clause ]
   [ allow_disallow_clustering ]
```

merge_table_subpartitions

```
MERGE SUBPARTITIONS subpartition_or_key_value
   { , subpartition_or_key_value [, subpartition_or_key_value ]...
   | TO subpartition_or_key_value }
   [ INTO { range_subpartition_desc
          | list_subpartition_desc
          }
   ]
   [ dependent_tables_clause ]
   [ update_index_clauses ]
   [ parallel_clause ]
   [ allow_disallow_clustering ]
```

merge_update_clause

```
WHEN MATCHED THEN
UPDATE SET column = { expr | DEFAULT }
           [, column = { expr | DEFAULT } ]...
[ where_clause ]
[ DELETE where_clause ]
```

migrate_key

```
SET [ ENCRYPTION ] KEY IDENTIFIED BY HSM_auth_string
  MIGRATE USING software_keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

mining_analytic_clause

```
[ query_partition_clause ] [ order_by_clause ]
```

mining_attribute_clause

```
USING
{ *
| { [ schema . ] table . *
  | expr [ AS alias ]
  }
    [, { [ schema . ] table . *
       | expr [ AS alias ]
       }
    ]...
}
```

model_clause

```
MODEL
   [ cell_reference_options ]
   [ return_rows_clause ]
   [ reference_model ]...
main_model
```

model_column

```
expr [ [ AS ] c_alias ]
```

model_column_clauses

```
[ PARTITION BY (expr [ c_alias ] [, expr [c_alias] ]...) ]
```

```
DIMENSION BY (expr [c_alias] [, expr [c_alias] ]...)
MEASURES (expr [c_alias] [, expr [c_alias] ]...)
```

model_iterate_clause

```
ITERATE ( number ) [ UNTIL ( condition ) ]
```

model_rules_clause

```
[ RULES
  [ { UPDATE | UPSERT [ ALL ] } ]
  [ { AUTOMATIC | SEQUENTIAL } ORDER ]
  [ model_iterate_clause ]
]
( [ { UPDATE | UPSERT [ ALL ] } ]
cell_assignment [ order_by_clause ] = expr
  [,  [ { UPDATE | UPSERT [ ALL ] } ]
    cell_assignment [ order_by_clause ] = expr
  ]...
)
```

modify_col_properties

```
column [ datatype ]
       [ DEFAULT [ ON NULL ] expr | identity_clause | DROP IDENTITY ]
       [ { ENCRYPT encryption_spec } | DECRYPT ]
       [ inline_constraint ... ]
       [ LOB_storage_clause ]
       [ alter_XMLSchema_clause ]
```

modify_col_substitutable

```
COLUMN column
[ NOT ] SUBSTITUTABLE AT ALL LEVELS
[ FORCE ]
```

modify_col_visibility

```
column { VISIBLE | INVISIBLE }
```

modify_collection_retrieval

```
MODIFY NESTED TABLE collection_item
RETURN AS { LOCATOR | VALUE }
```

modify_column_clauses

```
MODIFY
{ ( modify_col_properties | modify_virtcol_properties
    [, modify_col_properties | modify_virtcol_properties ]... )
| ( modify_col_visibility [, modify_col_visibility ]... )
| modify_col_substitutable
}
```

modify_diskgroup_file

```
MODIFY FILE 'filename' ATTRIBUTE ( disk_region_clause )
  [, 'filename' ATTRIBUTE ( disk_region_clause ) ]...
```

modify_hash_partition

```
MODIFY partition_extended_name
  { partition_attributes
  | coalesce_table_subpartition
  | alter_mapping_table_clause
  | [ REBUILD ] UNUSABLE LOCAL INDEXES
  | indexing_clause
  }
```

modify_index_default_attrs

```
MODIFY DEFAULT ATTRIBUTES
   [ FOR PARTITION partition ]
   { physical_attributes_clause
   | TABLESPACE { tablespace | DEFAULT }
   | logging_clause
   }...
```

modify_index_partition

```
MODIFY PARTITION partition
{ { deallocate_unused_clause
  | allocate_extent_clause
  | physical_attributes_clause
  | logging_clause
  | index_compression
  }...
| PARAMETERS ('ODCI_parameters')
| COALESCE [ CLEANUP ]
| UPDATE BLOCK REFERENCES
| UNUSABLE
}
```

modify_index_subpartition

```
MODIFY SUBPARTITION subpartition
{ UNUSABLE
| allocate_extent_clause
| deallocate_unused_clause
}
```

modify_list_partition

```
MODIFY partition_extended_name
  { partition_attributes
  | { ADD | DROP } VALUES (literal[ , literal ]...)
  | { add_range_subpartition
    | add_list_subpartition
    | add_hash_subpartition
    }
  | coalesce_table_subpartition
  | [ REBUILD ] UNUSABLE LOCAL INDEXES
  | indexing_clause
  }
```

modify_LOB_parameters

```
{ storage_clause
| PCTVERSION integer
| FREEPOOLS integer
| REBUILD FREEPOOLS
| LOB_retention_clause
| LOB_deduplicate_clause
| LOB_compression_clause
| { ENCRYPT encryption_spec | DECRYPT }
| { CACHE
  | { NOCACHE | CACHE READS } [ logging_clause ]
  }
| allocate_extent_clause
| shrink_clause
| deallocate_unused_clause
} ...
```

modify_LOB_storage_clause

```
MODIFY LOB (LOB_item)
   (modify_LOB_parameters)
```

modify_mv_column_clause

```
MODIFY ( column [ ENCRYPT encryption_spec
       | DECRYPT ]
       )
```

modify_opaque_type

```
MODIFY OPAQUE TYPE anydata_column
STORE ( type_name [, type_name ]... ) UNPACKED
```

modify_range_partition

```
MODIFY partition_extended_name
   { partition_attributes
   | { add_range_subpartition
     | add_hash_subpartition
     | add_list_subpartition
     }
   | coalesce_table_subpartition
   | alter_mapping_table_clause
   | [ REBUILD ] UNUSABLE LOCAL INDEXES
   | indexing_clause
   }
```

modify_table_default_attrs

```
MODIFY DEFAULT ATTRIBUTES
   [ FOR partition_extended_name ]
   [ deferred_segment_creation ]
   [ indexing_clause ]
   [ segment_attributes_clause ]
   [ table_compression ]
   [ inmemory_clause ]
   [ PCTTHRESHOLD integer ]
   [ prefix_compression ]
   [ alter_overflow_clause ]
   [ { LOB (LOB_item) | VARRAY varray } (LOB_parameters) ]...
```

modify_table_partition

```
{ modify_range_partition
| modify_hash_partition
| modify_list_partition
}
```

modify_table_subpartition

```
MODIFY subpartition_extended_name
{ allocate_extent_clause
| deallocate_unused_cluse
| shrink_clause
| { { LOB LOB_item | VARRAY varray } (modify_LOB_parameters) }...
| [ REBUILD ] UNUSABLE LOCAL INDEXES
| { ADD | DROP } VALUES ( literal [, literal]... )
| indexing_clause
}
```

modify_virtcol_properties

```
column [ datatype ]
[ GENERATED ALWAYS ] AS (column_expression) [ VIRTUAL ]
evaluation_edition_clause [ unusable_editions_clause ]
```

modify_volume_clause

```
MODIFY VOLUME asm_volume
  [ ATTRIBUTE (disk_region_clause) ]
  [ MOUNTPATH 'mountpath_name' ]
  [ USAGE 'usage_name' ]
```

move_datafile_clause

```
MOVE DATAFILE ( 'filename' | 'ASM_filename' | file_number )
 [ TO ( 'filename' | 'ASM_filename' ) ]
 [ REUSE ] [ KEEP ]
```

move_mv_log_clause

```
MOVE segment_attributes_clause [parallel_clause]
```

move_table_clause

```
MOVE [ ONLINE ]
   [ segment_attributes_clause ]
   [ table_compression ]
   [ index_org_table_clause ]
   [ { LOB_storage_clause | varray_col_properties }... ]
   [ parallel_clause ]
   [ allow_disallow_clustering ]
```

move_table_partition

```
MOVE partition_extended_name
   [ MAPPING TABLE ]
   [ table_partition_description ]
   [ update_index_clauses ]
   [ parallel_clause ]
   [ allow_disallow_clustering ]
   [ ONLINE ]
```

move_table_subpartition

```
MOVE subpartition_extended_name [ indexing_clause ]
     [ partitioning_storage_clause ] [ update_index_clauses ]
     [ parallel_clause ] [ allow_disallow_clustering ] [ ONLINE ]
```

multi_column_for_loop

```
FOR (dimension_column
      [, dimension_column ]...)
IN ( { (literal [, literal ]...)
       [ (literal [, literal ]...) ]...
     | subquery
     }
   )
```

multi_table_insert

```
{ ALL
  { insert_into_clause [ values_clause ] [error_logging_clause] }...
| conditional_insert_clause
} subquery
```

multiset_except

```
nested_table1
MULTISET EXCEPT [ ALL | DISTINCT ]
nested_table2
```

multiset_intersect

```
nested_table1
MULTISET INTERSECT [ ALL | DISTINCT ]
nested_table2
```

multiset_union

```
nested_table1
MULTISET UNION [ ALL | DISTINCT ]
nested_table2
```

mv_log_augmentation

```
ADD { { OBJECT ID
      | PRIMARY KEY
      | ROWID
      | SEQUENCE
      } [ (column [, column ]...) ]
    | (column [, column ]... )
    } [, { { OBJECT ID
           | PRIMARY KEY
           | ROWID
           | SEQUENCE
           }
           [ (column [, column ]...) ]
         | (column [, column ]...)
         }
      ]...
    [ new_values_clause ]
```

mv_log_purge_clause

```
PURGE { IMMEDIATE [ SYNCHRONOUS | ASYNCHRONOUS ]  )
      | START WITH datetime_expr
          [ NEXT datetime_expr
          | REPEAT INTERVAL interval_expr
          ]
      | [ START WITH datetime_expr ] { NEXT datetime_expr
                                     | REPEAT INTERVAL interval_expr
                                     }
      }
```

nested_table_col_properties

```
NESTED TABLE
{ nested_item | COLUMN_VALUE }
[ substitutable_column_clause ]
[ LOCAL | GLOBAL ]
STORE AS storage_table
[ ( { (object_properties)
    | [ physical_properties ]
    | [ column_properties ]
    }...
  )
]
[ RETURN [ AS ] { LOCATOR | VALUE } ]
```

nested_table_partition_spec

```
PARTITION partition [segment_attributes_clause]
```

new_values_clause

```
{ INCLUDING | EXCLUDING } NEW VALUES
```

number

```
[ + | - ]
{ digit [ digit ]... [ . ] [ digit [ digit ]... ]
| . digit [ digit ]...
}
[ [ e | E ] [ + | - ] digit [ digit ]... ] [ f | F | d | D ]
```

numeric_file_name

```
+diskgroup_name.filenumber.incarnation_number
```

object_properties

```
{ { column | attribute }
    [ DEFAULT expr ]
    [ { inline_constraint }...  | inline_ref_constraint ]
| { out_of_line_constraint
  | out_of_line_ref_constraint
  | supplemental_logging_props
  }
}
```

object_step

```
.{ simple_name | "complex_name"  | * }
```

object_table

```
OF
   [ schema. ] object_type
   [ object_table_substitution ]
   [ (object_properties) ]
   [ ON COMMIT { DELETE | PRESERVE } ROWS ]
   [ OID_clause ]
   [ OID_index_clause ]
   [ physical_properties ]
   [ table_properties ]
```

object_table_substitution

```
[ NOT ] SUBSTITUTABLE AT ALL LEVELS
```

object_type_col_properties

```
COLUMN column substitutable_column_clause
```

object_view_clause

```
OF [ schema. ] type_name
{ WITH OBJECT { IDENTIFIER | ID }
  { DEFAULT | ( attribute [, attribute ]... ) }
| UNDER [ schema. ] superview
}
[ ( { out_of_line_constraint
    | attribute { inline_constraint }...
    }  [, { out_of_line_constraint
          | attribute { inline_constraint }...
          }
       ]...
  )
]
```

OID_clause

```
OBJECT IDENTIFIER IS
{ SYSTEM GENERATED | PRIMARY KEY }
```

OID_index_clause

```
OIDINDEX [ index ]
({ physical_attributes_clause
 | TABLESPACE tablespace
 }...
)
```

on_comp_partitioned_table

```
[ STORE IN ( tablespace [, tablespace ]... ) ]
( PARTITION
    [ partition ]
    [ { segment_attributes_clause
      | index_compression
      }...
    ] [ USABLE | UNUSABLE ] [ index_subpartition_clause ]
      [, PARTITION
           [ partition ]
           [ { segment_attributes_clause
             | index_compression
             }...
           ] [ USABLE | UNUSABLE ] [ index_subpartition_clause ]
       ]...
)
```

on_hash_partitioned_table

```
{ STORE IN (tablespace[, tablespace ]...)
| (PARTITION [ partition ] [ TABLESPACE tablespace ]
    [ index_compression ] [ USABLE | UNUSABLE ]
  [, PARTITION [ partition ] [ TABLESPACE tablespace ]
    [ index_compression ] [ USABLE | UNUSABLE ]] ...
  )
}
```

on_list_partitioned_table

```
( PARTITION
    [ partition ]
    [ { segment_attributes_clause
      | index_compression
      }...
    ] [ USABLE | UNUSABLE ]
      [, PARTITION
           [ partition ]
           [ { segment_attributes_clause
             | index_compression
             }...
           ] [ USABLE | UNUSABLE ]
      ]...
)
```

on_object_clause

```
ON { [ schema. ] object
   | USER user [, user]...
   | DIRECTORY directory_name
   | EDITION edition_name
   | MINING MODEL [ schema. ] mining_model_name
   | JAVA { SOURCE | RESOURCE } [ schema. ] object
   | SQL TRANSLATION PROFILE [ schema. ] profile
   }
```

on_range_partitioned_table

```
( PARTITION
    [ partition ]
    [ { segment_attributes_clause
      | index_compression
      }...
    ] [ USABLE | UNUSABLE ]
      [, PARTITION
          [ partition ]
          [ { segment_attributes_clause
            | index_compression
          }...
        ] [ USABLE | UNUSABLE ]
     ]...
)
```

open_keystore

```
SET KEYSTORE OPEN
  IDENTIFIED BY keystore_password
  [ CONTAINER = { ALL | CURRENT } ]
```

order_by_clause

```
ORDER [ SIBLINGS ] BY
{ expr | position | c_alias }
[ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
  [, { expr | position | c_alias }
     [ ASC | DESC ]
     [ NULLS FIRST | NULLS LAST ]
  ]...
```

ordinality_column

```
column_name FOR ORDINALITY
```

out_of_line_constraint

```
  [ CONSTRAINT constraint_name ]
{ UNIQUE (column [, column ]...)
| PRIMARY KEY (column [, column ]...)
| FOREIGN KEY (column [, column ]...) references_clause
| CHECK (condition)
} [ constraint_state ]
```

out_of_line_part_storage

```
PARTITION partition
  { nested_table_col_properties | LOB_storage_clause | varray_col_properties }
    [ nested_table_col_properties | LOB_storage_clause | varray_col_properties ]...
[ (SUBPARTITION subpartition
   { nested_table_col_properties | LOB_storage_clause | varray_col_properties }
     [ nested_table_col_properties | LOB_storage_clause | varray_col_properties
     ]...
  )
]
```

out_of_line_ref_constraint

```
{ SCOPE FOR ({ ref_col | ref_attr })
    IS [ schema. ] scope_table
| REF ({ ref_col | ref_attr }) WITH ROWID
| [ CONSTRAINT constraint_name ] FOREIGN KEY
    ( { ref_col [, ref_col ] | ref_attr [, ref_attr ] } ) references_clause
    [ constraint_state ]
}
```

outer_join_clause

```
  [ query_partition_clause ] [ NATURAL ]
outer_join_type JOIN table_reference
  [ query_partition_clause ]
  [ ON condition
  | USING ( column [, column ]...)
  ]
```

outer_join_type

```
{ FULL | LEFT | RIGHT } [ OUTER ]
```

parallel_clause

```
{ NOPARALLEL | PARALLEL [ integer ] }
```

partial_database_recovery

```
{ TABLESPACE tablespace [, tablespace ]...
| DATAFILE { 'filename' | filenumber }
             [, 'filename' | filenumber ]...
}
```

partial_index_clause

```
INDEXING { PARTIAL | FULL }
```

partition_attributes

```
[ { physical_attributes_clause
  | logging_clause
  | allocate_extent_clause
  | deallocate_unused_clause
  | shrink_clause
  }...
]
[ OVERFLOW
  { physical_attributes_clause
  | logging_clause
  | allocate_extent_clause
  | deallocate_unused_clause
  }...
]
[ table_compression ]
[ inmemory_clause ]
[ { { LOB LOB_item | VARRAY varray } (modify_LOB_parameters) }...]
```

partition_extended_name

```
PARTITION partition
|
PARTITION FOR ( partition_key_value [, partition_key_value]... )
```

partition_extended_names

```
{ PARTITION | PARTITIONS }
partition | { FOR ( partition_key_value [, partition_key_value ]... ) }
  [, partition | { FOR ( partition_key_value [, partition_key_value ]... ) } ]...
```

partition_extension_clause

```
{ PARTITION (partition)
| PARTITION FOR (partition_key_value [, partition_key_value]...)
| SUBPARTITION (subpartition)
| SUBPARTITION FOR (subpartition_key_value [, subpartition_key_value]...)
}
```

partition_or_key_value

```
partition
|
FOR ( partition_key_value [, partition_key_value ]... )
```

partition_spec

```
PARTITION [ partition ] [ table_partition_description ]
```

partitioning_storage_clause

```
[ { TABLESPACE tablespace
  | OVERFLOW [TABLESPACE tablespace]
  | table_compression
  | index_compression
  | inmemory_clause
  | LOB_partitioning_storage
  | VARRAY varray_item STORE AS [SECUREFILE | BASICFILE] LOB LOB_segname
  }...
]
```

password_parameters

```
{ { FAILED_LOGIN_ATTEMPTS
  | PASSWORD_LIFE_TIME
  | PASSWORD_REUSE_TIME
  | PASSWORD_REUSE_MAX
  | PASSWORD_LOCK_TIME
  | PASSWORD_GRACE_TIME
  }
  { expr | UNLIMITED | DEFAULT }
| PASSWORD_VERIFY_FUNCTION
  { function | NULL | DEFAULT }
}
```

path_prefix_clause

```
PATH_PREFIX = { 'path_name' | NONE }
```

pdb_change_state

```
[ pdb_name ] { pdb_open | pdb_close | pdb_save_or_discard_state }
```

pdb_change_state_from_root

```
{ pdb_name [, pdb_name ]... | ALL [ EXCEPT pdb_name [, pdb_name ]... ] }
{ pdb_open | pdb_close | pdb_save_or_discard_state }
```

pdb_close

```
CLOSE [ IMMEDIATE ] [ instances_clause | relocate_clause ]
```

pdb_datafile_clause

```
[ pdb_name ] DATAFILE
  { { { 'filename' | filenumber } [, 'filename' | filenumber ]... } | ALL }
  { ONLINE | OFFLINE }
```

pdb_dba_roles

```
ROLES = ( role [, role ]... )
```

pdb_force_logging_clause

```
{ ENABLE | DISABLE } FORCE { LOGGING | NOLOGGING }
```

pdb_general_recovery

```
RECOVER [ AUTOMATIC ]  [ FROM 'location' ]
  [ DATABASE
  |
  TABLESPACE tablespace [, tablespace ]...
  |
  DATAFILE { 'filename' | filenumber }
             [, 'filename' | filenumber ]...
  |
  LOGFILE 'filename'
  |
  CONTINUE [ DEFAULT ]
  ]
```

pdb_logging_clauses

```
{ logging_clause
| pdb_force_logging_clause
}
```

pdb_open

```
OPEN
  { [ READ WRITE | READ ONLY ] [ RESTRICTED ] [ FORCE ]
  | [ READ WRITE ] UPGRADE [ RESTRICTED ]
  | RESETLOGS
  }
  [ instances_clause ]
```

pdb_recovery_clauses

```
[ pdb_name ] { pdb_general_recovery
             | { BEGIN | END } BACKUP
             | { ENABLE | DISABLE } RECOVERY
             }
```

pdb_save_or_discard_state

```
{ SAVE | DISCARD } STATE [ instances_clause ]
```

pdb_settings_clauses

```
[ pdb_name ]
{ DEFAULT EDITION = edition_name
| SET DEFAULT ( BIGFILE | SMALLFILE ) TABLESPACE
| DEFAULT TABLESPACE tablespace_name
| DEFAULT TEMPORARY TABLESPACE { tablespace | tablespace_group_name }
| RENAME GLOBAL_NAME TO database.domain [. domain ]...
| set_time_zone_clause
| database_file_clauses
| supplemental_db_logging
| pdb_storage_clause
| pdb_logging_clauses
}
```

pdb_storage_clause

```
STORAGE
  { ( MAXSIZE { UNLIMITED | size_clause }
    |
    MAX_SHARED_TEMP_SIZE { UNLIMITED | size_clause }... )
  |
  UNLIMITED
  }
```

pdb_unplug_clause

```
pdb_name UNPLUG INTO 'filename'
```

period_definition

```
PERIOD FOR valid_time_column [ ( start_time_column, end_time_column ) ]
```

permanent_tablespace_clause

```
TABLESPACE tablespace
  [ DATAFILE file_specification [, file_specification ]... ]
{ MINIMUM EXTENT size_clause
| BLOCKSIZE integer [ K ]
| logging_clause
| FORCE LOGGING
| ENCRYPTION tablespace_encryption_spec
| DEFAULT [ table_compression ] [ inmemory_clause ] [ storage_clause ]
| { ONLINE | OFFLINE }
| extent_management_clause
| segment_management_clause
| flashback_mode_clause
}...
```

physical_attributes_clause

```
[ { PCTFREE integer
  | PCTUSED integer
  | INITRANS integer
  | storage_clause
  }...
]
```

physical_properties

```
{ [ deferred_segment_creation ] segment_attributes_clause [ table_compression ]
    [ inmemory_table_clause ] [ ilm_clause ]
| [ deferred_segment_creation ] ORGANIZATION
  { HEAP [ segment_attributes_clause ] heap_org_table_clause
  | INDEX [ segment_attributes_clause ] index_org_table_clause
  | EXTERNAL external_table_clause
  }
| CLUSTER cluster (column [, column ]...)
}
```

pivot_clause

```
PIVOT [ XML ]
  ( aggregate_function ( expr ) [[AS] alias ]
      [, aggregate_function ( expr ) [[AS] alias ] ]...
    pivot_for_clause
    pivot_in_clause
  )
```

pivot_for_clause

```
FOR { column
    | ( column [, column]... )
    }
```

pivot_in_clause

```
IN ( { { { expr
         | ( expr [, expr]... )
         } [ [ AS] alias]
       }...
     | subquery
     | ANY [, ANY]...
     }
   )
```

plsql_declarations

```
{ function_declaration | procedure_declaration }...
```

prefix_compression

```
COMPRESS [ integer ] | NOCOMPRESS
```

privilege_audit_clause

```
PRIVILEGES system_privilege [, system_privilege ]...
```

program_unit

```
{ FUNCTION [ schema. ] function_name
|
PROCEDURE [ schema. ] procedure_name
|
PACKAGE [ schema. ] package_name }
```

proxy_clause

```
{ GRANT CONNECT THROUGH { ENTERPRISE USERS | db_user_proxy db_user_proxy_clauses }
| REVOKE CONNECT THROUGH { ENTERPRISE USERS | db_user_proxy }}
```

qualified_disk_clause

```
search_string
[ NAME disk_name ]
[ SIZE size_clause ]
[ FORCE | NOFORCE ]
```

qualified_template_clause

```
ATTRIBUTE
( redundancy_clause
  striping_clause
  disk_region_clause
)
```

query_block

```
  [ with_clause ]
SELECT [ hint ] [ { { DISTINCT | UNIQUE } | ALL } ] select_list
  FROM { table_reference | join_clause | ( join_clause ) }
         [ , { table_reference | join_clause | (join_clause) } ] ...
  [ where_clause ]
  [ hierarchical_query_clause ]
  [ group_by_clause ]
  [ model_clause ]
```

query_partition_clause

```
PARTITION BY
  { expr[, expr ]...
  | ( expr[, expr ]... )
  }
```

query_rewrite_clause

```
{ ENABLE | DISABLE } QUERY REWRITE [ unusable_editions_clause ]
```

query_table_expression

```
{ query_name
| [ schema. ]
  { table [ partition_extension_clause
          | @ dblink
          ]
  | { view | materialized view } [ @ dblink ]
  } [sample_clause]
| [ LATERAL ] (subquery [ subquery_restriction_clause ])
| table_collection_expression
}
```

quiesce_clauses

```
QUIESCE RESTRICTED | UNQUIESCE
```

range_partition_desc

```
PARTITION [partition]
range_values_clause
table_partition_description
[ ( { range_subpartition_desc [, range_subpartition_desc] ...
    | list_subpartition_desc [, list_subpartition_desc] ...
    | individual_hash_subparts [, individual_hash_subparts] ...
    }
  ) | hash_subparts_by_quantity ]
```

range_partitions

```
PARTITION BY RANGE (column[, column ]...)
  [ INTERVAL (expr) [ STORE IN ( tablespace [, tablespace]...) ]]
( PARTITION [ partition ]
    range_values_clause table_partition_description
      [, PARTITION [ partition ]
        range_values_clause table_partition_description
      ]...
)
```

range_subpartition_desc

```
SUBPARTITION [subpartition] range_values_clause
  [indexing_clause] [partitioning_storage_clause]
```

range_values_clause

```
VALUES LESS THAN
  ({ literal | MAXVALUE }
     [, { literal | MAXVALUE } ]...
  )
```

rebalance_diskgroup_clause

```
REBALANCE [POWER integer] [WAIT | NOWAIT]
```

rebuild_clause

```
REBUILD
  [ { PARTITION partition
    | SUBPARTITION subpartition
    }
  | { REVERSE | NOREVERSE }
  ]
  [ parallel_clause
  | TABLESPACE tablespace
  | PARAMETERS ( 'ODCI_parameters' )
  | XMLIndex_parameters_clause
  | ONLINE
  | physical_attributes_clause
  | index_compression
  | logging_clause
  | partial_index_clause
  ]...
```

records_per_block_clause

```
{ MINIMIZE | NOMINIMIZE } RECORDS_PER_BLOCK
```

recovery_clauses

```
{ general_recovery
| managed_standby_recovery
| BEGIN BACKUP
| END BACKUP
}
```

redo_log_file_spec

```
[ 'filename | ASM_filename'
| ('filename | ASM_filename'
   [, 'filename | ASM_filename' ]...)
]
[ SIZE size_clause ]
[ BLOCKSIZE size_clause
[ REUSE ]
```

redundancy_clause

```
[ MIRROR | HIGH | UNPROTECTED ]
```

reference_model

```
REFERENCE reference_model_name ON (subquery)
  model_column_clauses [ cell_reference_options ]
```

reference_partition_desc

```
PARTITION [partition] [table_partition_description] )
```

reference_partitioning

```
PARTITION BY REFERENCE ( constraint )
  [ (reference_partition_desc...) ]
```

references_clause

```
REFERENCES [ schema. ] object [ (column [, column ]...) ]
  [ON DELETE { CASCADE | SET NULL } ]
```

register_logfile_clause

```
REGISTER [ OR REPLACE ]
  [ PHYSICAL | LOGICAL ]
LOGFILE [ file_specification  [, file_specification ]...
  [ FOR logminer_session_name ]
```

relational_properties

```
{ column_definition
| virtual_column_definition
| period_definition
| { out_of_line_constraint | out_of_line_ref_constraint }
| supplemental_logging_props
}
  [, { column_definition
     | virtual_column_definition
     | period_definition
     | { out_of_line_constraint | out_of_line_ref_constraint }
     | supplemental_logging_props
     }
  ]...
```

relational_table

```
[ (relational_properties) ]
[ ON COMMIT { DELETE | PRESERVE } ROWS ]
[ physical_properties ]
[ table_properties ]
```

relocate_clause

```
RELOCATE [ TO 'instance_name' ]
| NORELOCATE
```

rename_column_clause

```
RENAME COLUMN old_name TO new_name
```

rename_disk_clause

```
RENAME
  { DISK old_disk_name TO new_disk_name [, old_disk_name TO new_disk_name ]...
  | DISKS ALL }
```

rename_index_partition

```
RENAME
  { PARTITION partition | SUBPARTITION subpartition }
TO new_name
```

rename_partition_subpart

```
RENAME { partition_extended_name
       | subpartition_extended_name
       } TO new_name
```

replace_disk_clause

```
REPLACE DISK disk_name WITH 'path_name' [ FORCE | NOFORCE ]
  [, disk_name WITH 'path_name' [ FORCE | NOFORCE ] ]...
[ POWER integer ] [ WAIT | NOWAIT ]
```

resize_disk_clause

```
RESIZE
{ ALL [ SIZE size_clause ]
| [QUORUM | REGULAR] DISK
   disk_name [ SIZE size_clause ]
   [, disk_name [ SIZE size_clause ] ]...
| DISKS IN [QUORUM | REGULAR] FAILGROUP
    failgroup_name [ SIZE size_clause ]
    [, failgroup_name [ SIZE size_clause ] ]...
}
```

resource_parameters

```
{ { SESSIONS_PER_USER
  | CPU_PER_SESSION
  | CPU_PER_CALL
  | CONNECT_TIME
  | IDLE_TIME
  | LOGICAL_READS_PER_SESSION
  | LOGICAL_READS_PER_CALL
  | COMPOSITE_LIMIT
  }
  { integer | UNLIMITED | DEFAULT }
| PRIVATE_SGA
  { size_clause | UNLIMITED | DEFAULT }
}
```

return_rows_clause

```
RETURN { UPDATED | ALL } ROWS
```

returning_clause

```
{ RETURN | RETURNING } expr [, expr ]...
INTO data_item [, data_item ]...
```

reverse_migrate_key

```
SET [ ENCRYPTION ] KEY IDENTIFIED BY software_keystore_password
  REVERSE MIGRATE USING HSM_auth_string
```

revoke_object_privileges

```
{ object_privilege | ALL [ PRIVILEGES ] }
  [, { object_privilege | ALL [ PRIVILEGES ] } ]...
on_object_clause
FROM revokee_clause
[ CASCADE CONSTRAINTS | FORCE ]
```

revoke_roles_from_programs

```
{ role [, role ]... | ALL } FROM program_unit [, program_unit ]...
```

revoke_system_privileges

```
{ system_privilege | role | ALL PRIVILEGES }
  [, { system_privilege | role | ALL PRIVILEGES } ]...
FROM revokee_clause
```

revokee_clause

```
{ user | role | PUBLIC }
  [, { user | role | PUBLIC } ]...
```

role_audit_clause

```
ROLES role [, role ]...
```

rolling_migration_clauses

```
{ START ROLLING MIGRATION TO 'ASM_version'
| STOP ROLLING MIGRATION
}
```

rolling_patch_clauses

```
{ START ROLLING PATCH
| STOP ROLLING PATCH
}
```

rollup_cube_clause

```
{ ROLLUP | CUBE } (grouping_expression_list)
```

routine_clause

```
[ schema. ] [ type. | package. ]
{ function | procedure | method }
[ @dblink_name ]
( [ argument [, argument ]... ] )
```

row_limiting_clause

```
[ OFFSET offset { ROW | ROWS } ]
[ FETCH { FIRST | NEXT } [ { rowcount | percent PERCENT } ]
    { ROW | ROWS } { ONLY | WITH TIES } ]
```

row_movement_clause

```
{ ENABLE | DISABLE } ROW MOVEMENT
```

row_pattern

```
[ row_pattern | ] row_pattern_term
```

```
Note: The vertical bar is part of the syntax rather than BNF notation.
```

row_pattern_aggregate_func

```
[ RUNNING | FINAL ] aggregate_function
```

row_pattern_classifier_func

```
CLASSIFIER()
```

row_pattern_clause

```
MATCH_RECOGNIZE {
  [ row_pattern_partition_by ]
  [ row_pattern_order_by ]
  [ row_pattern_measures ]
  [ row_pattern_rows_per_match ]
  [ row_pattern_skip_to ]
  PATTERN (row_pattern)
  [ row_pattern_subset_clause ]
  DEFINE row_pattern_definition_list
  }
```

row_pattern_definition

```
variable_name AS condition
```

row_pattern_definition_list

```
row_pattern_definition [, row_pattern_definition ]...
```

row_pattern_factor

```
row_pattern_primary [ row_pattern_quantifier ]
```

row_pattern_match_num_func

```
MATCH_NUMBER()
```

row_pattern_measure_column

```
expr AS c_alias
```

row_pattern_measures

```
MEASURES row_pattern_measure_column [, row_pattern_measure_column ]...
```

row_pattern_nav_compound

```
{ PREV | NEXT }
( [ RUNNING | FINAL ] { FIRST | LAST } ( expr [, offset ] ) [, offset] )
```

row_pattern_nav_logical

```
[ RUNNING | FINAL ] { FIRST | LAST } ( expr [, offset ] )
```

row_pattern_nav_physical

```
{ PREV | NEXT } ( expr [, offset ] )
```

row_pattern_navigation_func

```
row_pattern_nav_logical
| row_pattern_nav_physical
| row_pattern_nav_compound
```

row_pattern_order_by

```
ORDER BY column [, column ]...
```

row_pattern_partition_by

```
PARTITION BY column [, column ]...
```

row_pattern_permute

```
PERMUTE ( row_pattern [, row_pattern ]...)
```

row_pattern_primary

```
variable_name
| $
| ^
| ( [ row_pattern ] )
| {- row_pattern -}
| row_pattern_permute

Note: The curly brackets are part of the syntax rather than BNF notation.
```

row_pattern_quantifier

```
* [ ? ]
| + [ ? ]
| ? [ ? ]
| { [ unsigned_integer ] , [ unsigned_integer ] } [ ? ]
| { unsigned_integer }

Note: The curly brackets are part of the syntax rather than BNF notation.
```

row_pattern_rec_func

```
row_pattern_classifier_func
| row_pattern_match_num_func
| row_pattern_navigation_func
| row_pattern_aggregate_func
```

row_pattern_rows_per_match

```
ONE ROW PER MATCH
| ALL ROWS PER MATCH
```

row_pattern_skip_to

```
AFTER MATCH {
  SKIP TO NEXT ROW
  | SKIP PAST LAST ROW
  | SKIP TO FIRST variable_name
  | SKIP TO LAST variable_name
  | SKIP TO variable_name
  }
```

row_pattern_subset_clause

```
SUBSET row_pattern_subset_item [, row_pattern_subset_item ]...
```

row_pattern_subset_item

```
variable_name = ( variable_name [, variable_name ] )
```

row_pattern_term

```
[ row_pattern_term ] row_pattern_factor
```

sample_clause

```
SAMPLE [ BLOCK ]
       (sample_percent)
       [ SEED (seed_value) ]
```

scoped_table_ref_constraint

```
{ SCOPE FOR ({ ref_column | ref_attribute })
  IS [ schema. ] { scope_table_name | c_alias }
}
```

scrub_clause

```
SCRUB [ FILE 'ASM_filename' | DISK disk_name ]
  [ REPAIR | NOREPAIR ]
  [ POWER { AUTO | LOW | HIGH | MAX } ]
  [ WAIT | NOWAIT ]
  [ FORCE | NOFORCE ]
```

search_clause

```
{ SEARCH
        { DEPTH FIRST BY c_alias [, c_alias]...
            [ ASC | DESC ]
            [ NULLS FIRST | NULLS LAST ]
         | BREADTH FIRST BY c_alias [, c_alias]...
            [ ASC | DESC ]
            [ NULLS FIRST | NULLS LAST ]
        }
        SET ordering_column
}
```

searched_case_expression

```
{ WHEN condition THEN return_expr }...
```

secret_management_clauses

```
{ add_update_secret
| delete_secret
}
```

security_clause

```
GUARD { ALL | STANDBY | NONE }
```

security_clauses

```
{ { ENABLE | DISABLE } RESTRICTED SESSION
  | SET ENCRYPTION WALLET OPEN
    IDENTIFIED BY { "wallet_password" | "HSM_auth_string" }
  | SET ENCRYPTION WALLET CLOSE
    [ IDENTIFIED BY { "wallet_password" | "HSM_auth_string" } ]
  | set_encryption_key
}
```

segment_attributes_clause

```
{ physical_attributes_clause
| TABLESPACE tablespace
| logging_clause
}...
```

segment_management_clause

```
SEGMENT SPACE MANAGEMENT { AUTO | MANUAL }
```

select_list

```
{ [t_alias.] *
| { query_name.*
  | [ schema. ]
    { table | view | materialized view } .*
  | expr [ [ AS ] c_alias ]
  }
    [, { query_name.*
       | [ schema. ]
```

```
         { table | view | materialized view } .*
       | expr [ [ AS ] c_alias ]
       }
    ]...
}
```

set_encryption_key

```
{ SET ENCRYPTION KEY
  {
    [ "certificate_id" ] IDENTIFIED BY "wallet_password"
    |
    IDENTIFIED BY "HSM_auth_string" [ MIGRATE USING "wallet_password" ]
  }
}
```

set_key

```
SET [ ENCRYPTION ] KEY [ USING TAG 'tag' ]
  IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
  [ CONTAINER = { ALL | CURRENT } ]
```

set_key_tag

```
SET TAG 'tag' FOR 'key_id' IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

set_parameter_clause

```
parameter_name =
   parameter_value [, parameter_value ]...
   [ COMMENT = string ]
   [ DEFERRED ]
   [ { SCOPE = { MEMORY | SPFILE | BOTH }
     | SID = { 'sid' | '*' }
     }...
   ]
   [ CONTAINER = { CURRENT | ALL } ]
```

set_subpartition_template

```
SET SUBPARTITION TEMPLATE
   { ( range_subpartition_desc [, range_subpartition_desc]... )
   | ( list_subpartition_desc [, list_subpartition_desc]... )
   | ( individual_hash_subparts [, individual_hash_subparts]... )
   | ()
   | hash_subpartition_quantity
   }
```

set_time_zone_clause

```
SET TIME_ZONE =
   '{ { + | - } hh : mi | time_zone_region }'
```

shrink_clause

```
SHRINK SPACE [ COMPACT ] [ CASCADE ]
```

shutdown_dispatcher_clause

```
SHUTDOWN [ IMMEDIATE ] dispatcher_name
```

simple_case_expression

```
expr
  { WHEN comparison_expr THEN return_expr }...
```

single_column_for_loop

```
FOR dimension_column
  { IN ( { literal [, literal ]...
         | subquery
         }
       )
  | [ LIKE pattern ] FROM literal TO literal
      { INCREMENT | DECREMENT } literal
  }
```

single_table_insert

```
insert_into_clause
{ values_clause [ returning_clause ]
| subquery
} [ error_logging_clause ]
```

size_clause

```
integer [ K | M | G | T | P | E ]
```

source_file_name_convert

```
SOURCE_FILE_NAME_CONVERT =
  { ( 'filename_pattern', 'replacement_filename_pattern'
      [, 'filename_pattern', 'replacement_filename_pattern' ]... )
    |
    NONE
  }
```

split_index_partition

```
SPLIT PARTITION partition_name_old
   AT (literal [, literal ]...)
   [ INTO (index_partition_description,
           index_partition_description
          )
   ]
   [ parallel_clause ]
```

split_nested_table_part

```
NESTED TABLE column INTO
  ( PARTITION partition [segment_attributes_clause],
    PARTITION partition [segment_attributes_clause] [split_nested_table_part]
  ) [ split_nested_table_part ]
```

split_table_partition

```
SPLIT partition_extended_name
  { AT (literal [, literal]... )
    [ INTO ( range_partition_desc, range_partition_desc ) ]
  | VALUES (literal [, literal] ... )
    [ INTO (list_partition_desc, list_partition_desc ) ]
  | INTO ( { range_partition_desc [, range_partition_desc ]...
           | list_partition_desc [, list_partition_desc ]... }
         , partition_spec )
  } [ split_nested_table_part ]
    [ dependent_tables_clause ]
    [ update_index_clauses ]
    [ parallel_clause ]
    [ allow_disallow_clustering ]
```

split_table_subpartition

```
SPLIT subpartition_extended_name
  { AT ( literal [, literal]... )
```

```
    [ INTO (range_subpartition_desc, range_subpartition_desc) ]
  | VALUES ({ literal | NULL  [, literal | NULL ]...})
    [ INTO (list_subpartition_desc, list_subpartition_desc) ]
  | INTO ( { range_subpartition_desc [, range_subpartition_desc ]...
           | list_subpartition_desc [, list_subpartition_desc ]... }
         , subpartition_spec )
  } [ dependent_tables_clause ]
    [ update_index_clauses ]
    [ parallel_clause ]
    [ allow_disallow_clustering ]
```

sql_format

```
[+ | -] days hours : minutes : seconds [. frac_secs ]
```

standard_actions

```
ACTIONS
  { { object_action | ALL }
    ON { DIRECTORY directory_name
       | MINING MODEL [ schema. ] object_name
       | [ schema. ] object_name }
  | { system_action | ALL }
  }
    [ { object_action | ALL }
      ON { DIRECTORY directory_name
         | MINING MODEL [ schema. ] object_name
         | [ schema. ] object_name }
    | { system_action | ALL } ]...
```

standby_database_clauses

```
{ { activate_standby_db_clause
| maximize_standby_db_clause
| register_logfile_clause
| commit_switchover_clause
| start_standby_clause
| stop_standby_clause
| convert_database_clause
} [ parallel_clause ] }
|
{ switchover_clause | failover_clause }
```

standbys_clause

```
STANDBYS = { NONE | ALL }
```

start_standby_clause

```
START LOGICAL STANDBY APPLY
[ IMMEDIATE ]
[ NODELAY ]
[ NEW PRIMARY dblink
| INITIAL [ scn_value ]
| { SKIP FAILED TRANSACTION | FINISH }
]
```

startup_clauses

```
{ MOUNT [ { STANDBY | CLONE } DATABASE ]
| OPEN
  { [ READ WRITE ]
      [ RESETLOGS | NORESETLOGS ]
        [ UPGRADE | DOWNGRADE ]
  | READ ONLY
  }
}
```

still_image_object_types

```
{ SI_StillImage
| SI_AverageColor
| SI_PositionalColor
| SI_ColorHistogram
| SI_Texture
| SI_FeatureList
| SI_Color
}
```

stop_standby_clause

```
{ STOP | ABORT } LOGICAL STANDBY APPLY
```

storage_clause

```
STORAGE
({ INITIAL size_clause
 | NEXT size_clause
 | MINEXTENTS integer
 | MAXEXTENTS { integer | UNLIMITED }
 | maxsize_clause
 | PCTINCREASE integer
 | FREELISTS integer
 | FREELIST GROUPS integer
 | OPTIMAL [ size_clause | NULL ]
 | BUFFER_POOL { KEEP | RECYCLE | DEFAULT }
 | FLASH_CACHE { KEEP | NONE | DEFAULT }
 | ENCRYPT
 } ...
)
```

storage_table_clause

```
WITH {SYSTEM | USER} MANAGED STORAGE TABLES
```

string

```
[ {N | n} ]
{ '[ c ]...'
| { Q | q } 'quote_delimiter c [ c ]... quote_delimiter'
}
```

striping_clause

```
[ FINE | COARSE ]
```

subpartition_by_hash

```
SUBPARTITION BY HASH (column [, column ]...)
   [ SUBPARTITIONS integer
        [ STORE IN (tablespace [, tablespace ]...) ]
   | subpartition_template
   ]
```

subpartition_by_list

```
SUBPARTITION BY LIST (column) [ subpartition_template ]
```

subpartition_by_range

```
SUBPARTITION BY RANGE ( column [, column]... ) [subpartition_template]
```

subpartition_extended_name

```
SUBPARTITION subpartition
|
SUBPARTITION FOR ( subpartition_key_value [, subpartition_key_value]... )
```

subpartition_extended_names

```
{ SUBPARTITION | SUBPARTITIONS }
subpartition | { FOR ( subpartition_key_value [, subpartition_key_value ]... ) }
  [, subpartition | { FOR ( subpartition_key_value [, subpartition_key_value ]... ) } ]...
```

subpartition_or_key_value

```
subpartition
|
FOR ( subpartition_key_value [, subpartition_key_value ]... )
```

subpartition_spec

```
SUBPARTITION [ subpartition ] [ partitioning_storage_clause ]
```

subpartition_template

```
SUBPARTITION TEMPLATE
  ( { range_subpartition_desc [, range_subpartition_desc] ...
    | list_subpartition_desc [, list_subpartition_desc] ...
    | individual_hash_subparts [, individual_hash_subparts] ...
    }
  ) | hash_subpartition_quantity
```

subquery

```
{ query_block
| subquery { UNION [ALL] | INTERSECT | MINUS } subquery
    [ { UNION [ALL] | INTERSECT | MINUS } subquery ]...
| ( subquery )
} [ order_by_clause ] [ row_limiting_clause ]
```

subquery_factoring_clause

```
query_name ([c_alias [, c_alias]...]) AS (subquery) [search_clause] [cycle_clause]
[, query_name ([c_alias [, c_alias]...]) AS (subquery) [search_clause] [cycle_clause]]...
```

subquery_restriction_clause

```
WITH { READ ONLY
     | CHECK OPTION
     } [ CONSTRAINT constraint ]
```

substitutable_column_clause

```
{ [ ELEMENT ] IS OF [ TYPE ] ( [ONLY] type)
| [ NOT ] SUBSTITUTABLE AT ALL LEVELS
}
```

supplemental_db_logging

```
{ ADD | DROP } SUPPLEMENTAL LOG
{ DATA
| supplemental_id_key_clause
| supplemental_plsql_clause
}
```

supplemental_id_key_clause

```
DATA
( { ALL | PRIMARY KEY | UNIQUE | FOREIGN KEY }
    [, { ALL | PRIMARY KEY | UNIQUE | FOREIGN KEY } ]...
)
COLUMNS
```

supplemental_log_grp_clause

```
GROUP log_group
```

```
(column [ NO LOG ]
  [, column [ NO LOG ] ]...)
  [ ALWAYS ]
```

supplemental_logging_props

```
SUPPLEMENTAL LOG { supplemental_log_grp_clause
                 | supplemental_id_key_clause
                 }
```

supplemental_plsql_clause

```
DATA FOR PROCEDURAL REPLICATION
```

supplemental_table_logging

```
{ ADD SUPPLEMENTAL LOG
  { supplemental_log_grp_clause | supplemental_id_key_clause }
    [, SUPPLEMENTAL LOG
       { supplemental_log_grp_clause | supplemental_id_key_clause }
    ]...
| DROP SUPPLEMENTAL LOG
  { supplemental_id_key_clause | GROUP log_group }
    [, SUPPLEMENTAL LOG
       { supplemental_id_key_clause | GROUP log_group }
    ]...
}
```

switch_logfile_clause

```
SWITCH ALL LOGFILES TO BLOCKSIZE integer
```

switchover_clause

```
SWITCHOVER TO target_db_name [ VERIFY | FORCE ]
```

system_partitioning

```
PARTITION BY SYSTEM [ PARTITIONS integer
                    | reference_partition_desc
                       [, reference_partition_desc ...]
                    ]
```

table_collection_expression

```
TABLE (collection_expression) [ (+) ]
```

table_compression

```
COMPRESS
| ROW STORE COMPRESS [ BASIC | ADVANCED ]
| COLUMN STORE COMPRESS [  FOR { QUERY | ARCHIVE } [ LOW | HIGH ] ]
  [ [NO] ROW LEVEL LOCKING ]
| NOCOMPRESS
```

table_index_clause

```
[ schema. ] table [ t_alias ]
(index_expr [ ASC | DESC ]
  [, index_expr [ ASC | DESC ] ]...)
  [ index_properties ]
```

table_partition_description

```
[ deferred_segment_creation ]
[ indexing_clause ]
[ segment_attributes_clause ]
[ table_compression | prefix_compression ]
[ inmemory_clause ]
[ OVERFLOW [ segment_attributes_clause ] ]
[ { LOB_storage_clause
  | varray_col_properties
  | nested_table_col_properties
  }...
]
```

table_partitioning_clauses

```
{ range_partitions
| list_partitions
| hash_partitions
| composite_range_partitions
| composite_list_partitions
| composite_hash_partitions
| reference_partitioning
| system_partitioning
}
```

table_properties

```
[ column_properties ]
[ indexing_clause ]
[ table_partitioning_clauses ]
[ attribute_clustering_clause ]
[ CACHE | NOCACHE ]
[ RESULT_CACHE ( MODE {DEFAULT | FORCE } ) ]
[ parallel_clause ]
[ ROWDEPENDENCIES | NOROWDEPENDENCIES ]
[ enable_disable_clause ]...
[ row_movement_clause ]
[ flashback_archive_clause ]
[ ROW ARCHIVAL ]
[ AS subquery ]
```

table_reference

```
{ { { ONLY (query_table_expression) | query_table_expression }
  [ flashback_query_clause ]
  [ pivot_clause | unpivot_clause | row_pattern_clause ] }
| containers_clause
}
[ t_alias ]
```

tablespace_clauses

```
{ EXTENT MANAGEMENT LOCAL
| DATAFILE file_specification [, file_specification ]...
| SYSAUX DATAFILE file_specification [, file_specification ]...
| default_tablespace
| default_temp_tablespace
| undo_tablespace
}
```

tablespace_datafile_clauses

```
DATAFILES { SIZE size_clause | autoextend_clause }...
```

tablespace_encryption_spec

```
[ USING 'encrypt_algorithm' ]
```

tablespace_group_clause

```
TABLESPACE GROUP { tablespace_group_name | '' }
```

tablespace_logging_clauses

```
{ logging_clause
| [ NO ] FORCE LOGGING
}
```

tablespace_retention_clause

```
RETENTION { GUARANTEE | NOGUARANTEE }
```

tablespace_state_clauses

```
{ { ONLINE
  | OFFLINE [ NORMAL | TEMPORARY | IMMEDIATE ]
  }
  | READ { ONLY | WRITE }
  | { PERMANENT | TEMPORARY }
}
```

tempfile_reuse_clause

```
TEMPFILE REUSE
```

temporary_tablespace_clause

```
TEMPORARY TABLESPACE tablespace
  [ TEMPFILE file_specification [, file_specification ]... ]
  [ tablespace_group_clause ]
  [ extent_management_clause ]
```

tiering_clause

```
TIER TO tablespace [ READ ONLY ]
```

timeout_clause

```
DROP AFTER integer { M | H }
```

trace_file_clause

```
TRACE
  [ AS 'filename' [ REUSE ] ]
  [ RESETLOGS | NORESETLOGS ]
```

truncate_partition_subpart

```
TRUNCATE { partition_extended_names | subpartition_extended_names }
   [ { DROP [ ALL ] | REUSE } STORAGE ]
   [ update_index_clauses [ parallel_clause ] ] [ CASCADE ]
```

undo_tablespace

```
  [ BIGFILE | SMALLFILE ]
UNDO TABLESPACE tablespace
  [ TABLESPACE file_specification [, file_specification ]...]
```

undo_tablespace_clause

```
UNDO TABLESPACE tablespace
  [ DATAFILE file_specification [, file_specification ]... ]
  [ extent_management_clause ]
  [ tablespace_retention_clause ]
```

undrop_disk_clause

```
UNDROP DISKS
```

unpivot_clause

```
UNPIVOT [ {INCLUDE | EXCLUDE} NULLS ]
( { column | ( column [, column]... ) }
  pivot_for_clause
  unpivot_in_clause
)
```

unpivot_in_clause

```
IN
( { column | ( column [, column]... ) }
      [  AS { literal | ( literal [, literal]... ) } ]
        [, { column | ( column [, column]... ) }
          [  AS {literal | ( literal [, literal]... ) } ]
        ]...
)
```

unusable_editions_clause

```
[ UNUSABLE BEFORE { CURRENT EDITION | EDITION edition } ]
[ UNUSABLE BEGINNING WITH { CURRENT EDITION | EDITION edition | NULL EDITION } ]
```

update_all_indexes_clause

```
UPDATE INDEXES
   [ ( index ( update_index_partition
             | update_index_subpartition
             )
        [, index ( update_index_partition
                 | update_index_subpartition
                 )
        ]...
      )
   ]
```

update_global_index_clause

```
{ UPDATE | INVALIDATE } GLOBAL INDEXES
```

update_index_clauses

```
{ update_global_index_clause
| update_all_indexes_clause
}
```

update_index_partition

```
index_partition_description [ index_subpartition_clause ]
  [, index_partition_description [ index_subpartition_clause ] ]...
```

update_index_subpartition

```
SUBPARTITION [ subpartition ]
   [ TABLESPACE tablespace ]
[, SUBPARTITION [ subpartition ]
      [ TABLESPACE tablespace ]
]...
```

update_set_clause

```
SET
{ { (column [, column ]...) = (subquery)
  | column = { expr | (subquery) | DEFAULT }
  }
     [, { (column [, column]...) = (subquery)
        | column = { expr | (subquery) | DEFAULT }
        }
     ]...
| VALUE (t_alias) = { expr | (subquery) }
}
```

upgrade_table_clause

```
UPGRADE [ [NOT ] INCLUDING DATA ]
   [ column_properties ]
```

use_key

```
USE [ ENCRYPTION ] KEY 'key_id' [ USING TAG 'tag' ]
  IDENTIFIED BY keystore_password
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

user_clauses

```
{ ADD USER user [, user]...
| DROP USER user [, user]... [CASCADE]
| REPLACE USER old_user WITH new_user [, old_user WITH new_user]...
}
```

user_tablespaces_clause

```
USER_TABLESPACES =
  { ( 'tablespace' [, 'tablespace' ]... )
  | NONE
  | ALL [ EXCEPT ( 'tablespace' [, 'tablespace' ]... ) ]
  }
```

usergroup_clauses

```
{ ADD USERGROUP usergroup WITH MEMBER user [, user]...
| MODIFY USERGROUP usergroup { ADD | DROP } MEMBER user [, user]...
| DROP USERGROUP usergroup
}
```

using_function_clause

```
USING [ schema. ] [ package. | type. ] function_name
```

using_index_clause

```
USING INDEX
  { [ schema. ] index
  | (create_index_statement)
  | index_properties
  }
```

using_statistics_type

```
USING { [ schema. ] statistics_type | NULL }
```

using_type_clause

```
USING [ schema. ] implementation_type [ array_DML_clause ]
```

validation_clauses

```
{ VALIDATE REF UPDATE [ SET DANGLING TO NULL ]
| VALIDATE STRUCTURE
     [ CASCADE { FAST | COMPLETE { OFFLINE | ONLINE } [ into_clause ] } ]
}
```

values_clause

```
VALUES ({ expr | DEFAULT }
          [, { expr | DEFAULT } ]...
       )
```

varray_col_properties

```
VARRAY varray_item
{ [ substitutable_column_clause ] varray_storage_clause
| substitutable_column_clause
}
```

varray_storage_clause

```
STORE AS [SECUREFILE | BASICFILE] LOB
{ [LOB_segname] ( LOB_storage_parameters )
| LOB_segname
}
```

virtual_column_definition

```
column [ datatype ] [ VISIBLE | INVISIBLE ]
  [ GENERATED ALWAYS ] AS (column_expression) [ VIRTUAL ]
  [ evaluation_edition_clause ] [ unusable_editions_clause ]
  [ inline_constraint [ inline_constraint ]... ]
```

where_clause

```
WHERE condition
```

windowing_clause

```
{ ROWS | RANGE }
{ BETWEEN
  { UNBOUNDED PRECEDING
  | CURRENT ROW
  | value_expr { PRECEDING | FOLLOWING }
  }
  AND
  { UNBOUNDED FOLLOWING
  | CURRENT ROW
  | value_expr { PRECEDING | FOLLOWING }
  }
| { UNBOUNDED PRECEDING
  | CURRENT ROW
  | value_expr PRECEDING
  }
}
```

with_clause

```
WITH [ plsql_declarations ] [ subquery_factoring_clause ]
```

XML_attributes_clause

```
XMLATTRIBUTES
  ( [ ENTITYESCAPING | NOENTITYESCAPING ]
    [ SCHEMACHECK | NOSCHEMACHECK ]
    value_expr [ { [AS] c_alias } | { AS EVALNAME value_expr } ]
      [, value_expr [ { [AS] c_alias } | { AS EVALNAME value_expr } ] ]...
  )
```

XML_namespaces_clause

```
XMLNAMESPACES
  ( [ string AS identifier ]
      [ [, string AS identifier ]
      ]...
    [ DEFAULT string ]
  )
```

XML_passing_clause

```
PASSING [ BY VALUE ]
```

```
    expr [ AS identifier ]
      [, expr [ AS identifier ]
      ]...
```

XML_table_column

```
column
     { FOR ORDINALITY
     | { datatype | XMLTYPE [ (SEQUENCE) BY REF ] }
     [ PATH string ] [ DEFAULT expr ]
     }
```

XMLIndex_clause

```
[XDB.] XMLINDEX [ local_XMLIndex_clause ]
                [ parallel_clause ]
  [ XMLIndex_parameters_clause ]
```

XMLSchema_spec

```
  [ XMLSCHEMA XMLSchema_URL ]
ELEMENT { element | XMLSchema_URL # element }
  [ STORE ALL VARRAYS AS { LOBS | TABLES } ]
  [ { ALLOW | DISALLOW } NONSCHEMA ]
  [ { ALLOW | DISALLOW } ANYSCHEMA ]
```

XMLTABLE_options

```
[ XML_passing_clause ]
[ RETURNING SEQUENCE BY REF ]
[ COLUMNS XML_table_column [, XML_table_column]...]
```

XMLType_column_properties

```
XMLTYPE [ COLUMN ] column
   [ XMLType_storage ]
   [ XMLSchema_spec ]
```

XMLType_storage

```
STORE
{ AS
{ OBJECT RELATIONAL
| [SECUREFILE | BASICFILE]
  { CLOB | BINARY XML }
    [ { LOB_segname [ (LOB_parameters) ]
      | (LOB_parameters)
      }
    ]
}
| { ALL VARRAYS AS { LOBS | TABLES } }
}
```

XMLType_table

```
OF XMLTYPE
  [ (oject_properties) ]
  [ XMLTYPE XMLType_storage ]
  [ XMLSchema_spec ]
  [ XMLType_virtual_columns ]
  [ ON COMMIT { DELETE | PRESERVE } ROWS ]
  [ OID_clause ]
  [ OID_index_clause ]
  [ physical_properties ]
  [ table_properties ]
```

XMLType_view_clause

```
OF XMLTYPE [ XMLSchema_spec ]
WITH OBJECT { IDENTIFIER | ID }
  { DEFAULT | ( expr [, expr ]...) }
```

XMLType_virtual_columns

```
VIRTUAL COLUMNS ( column AS (expr) [, column AS (expr) ]... )
```

ym_iso_format

```
[-] P [ years Y ] [months M] [days D]
  [T [hours H] [minutes M] [seconds [. frac_secs] S ] ]
```

zonemap_attributes

```
{ TABLESPACE tablespace
| SCALE integer
| PCTFREE integer
| PCTUSED integer
| { CACHE | NOCACHE }
}...
```

zonemap_clause

```
{ WITH MATERIALIZED ZONEMAP [ ( zonemap_name ) ] }
|
{ WITHOUT MATERIALIZED ZONEMAP }
```

zonemap_refresh_clause

```
REFRESH
[ FAST | COMPLETE | FORCE ]
[ ON { DEMAND | COMMIT | LOAD | DATA MOVEMENT | LOAD DATA MOVEMENT } ]
```

6 Data Types

This chapter presents data types that are recognized by Oracle and available for use within SQL.

This chapter includes the following sections:

Overview of Data Types

A **data type** is a classification of a particular type of information or data. Each value manipulated by Oracle has a data type. The data type of a value associates a fixed set of properties with the value. These properties cause Oracle to treat values of one data type differently from values of another.

The data types recognized by Oracle are:

ANSI-supported data types

```
{ CHARACTER [VARYING] (size)
| { CHAR | NCHAR } VARYING (size)
| VARCHAR (size)
| NATIONAL { CHARACTER | CHAR }
     [VARYING] (size)
| { NUMERIC | DECIMAL | DEC }
     [ (precision [, scale ]) ]
| { INTEGER | INT | SMALLINT }
| FLOAT [ (size) ]
| DOUBLE PRECISION
| REAL
}
```

Oracle built-in data types

```
{ character_datatypes
| number_datatypes
| long_and_raw_datatypes
| datetime_datatypes
| large_object_datatypes
| rowid_datatypes
}
```

Oracle-supplied data types

```
{ any_types
| XML_types
| spatial_types
| media_types
}
```

User-defined data types

User-defined data types use Oracle built-in data types and other user-defined data types to model the structure and behavior of data in applications.

> **See Also:** *Oracle Database SQL Language Reference* for more information about data types

Oracle Built-In Data Types

This section describes the kinds of Oracle built-in data types.

character_datatypes

```
{ CHAR [ (size [ BYTE | CHAR ]) ]
| VARCHAR2 (size [ BYTE | CHAR ])
| NCHAR [ (size) ]
| NVARCHAR2 (size)
}
```

datetime_datatypes

```
{ DATE
| TIMESTAMP [ (fractional_seconds_precision) ]
     [ WITH [ LOCAL ] TIME ZONE ]
| INTERVAL YEAR [ (year_precision) ] TO MONTH
| INTERVAL DAY [ (day_precision) ] TO SECOND
     [ (fractional_seconds_precision) ]
}
```

large_object_datatypes

```
{ BLOB | CLOB | NCLOB | BFILE }
```

long_and_raw_datatypes

```
{ LONG | LONG RAW | RAW (size) }
```

number_datatypes

```
{ NUMBER [ (precision [, scale ]) ]
| FLOAT [ (precision) ]
| BINARY_FLOAT
| BINARY_DOUBLE
}
```

rowid_datatypes

```
{ ROWID | UROWID [ (size) ] }
```

The codes listed for the data types are used internally by Oracle Database. The data type code of a column or object attribute is returned by the `DUMP` function.

Table 6–1 Built-in Data Type Summary

Code	Data Type	Description
1	`VARCHAR2`(*`size`* [`BYTE` | `CHAR`])	Variable-length character string having maximum length *`size`* bytes or characters. You must specify *`size`* for `VARCHAR2`. Minimum *`size`* is 1 byte or 1 character. Maximum size is: ▪ 32767 bytes or characters if `MAX_STRING_SIZE = EXTENDED` ▪ 4000 bytes or characters if `MAX_STRING_SIZE = STANDARD` Refer to *Oracle Database SQL Language Reference* for more information on the `MAX_STRING_SIZE` initialization parameter. `BYTE` indicates that the column will have byte length semantics. `CHAR` indicates that the column will have character semantics.
1	`NVARCHAR2`(*`size`*)	Variable-length Unicode character string having maximum length *`size`* characters. You must specify *`size`* for `NVARCHAR2`. The number of bytes can be up to two times *`size`* for `AL16UTF16` encoding and three times *`size`* for `UTF8` encoding. Maximum *`size`* is determined by the national character set definition, with an upper limit of: ▪ 32767 bytes if `MAX_STRING_SIZE = EXTENDED` ▪ 4000 bytes if `MAX_STRING_SIZE = STANDARD` Refer to *Oracle Database SQL Language Reference* for more information on the `MAX_STRING_SIZE` initialization parameter.
2	`NUMBER` [(*p* [, *s*])]	Number having precision *`p`* and scale *`s`*. The precision *`p`* can range from 1 to 38. The scale *`s`* can range from -84 to 127. Both precision and scale are in decimal digits. A `NUMBER` value requires from 1 to 22 bytes.
2	`FLOAT` [(*p*)]	A subtype of the `NUMBER` data type having precision *p*. A `FLOAT` value is represented internally as `NUMBER`. The precision *`p`* can range from 1 to 126 binary digits. A `FLOAT` value requires from 1 to 22 bytes.
8	`LONG`	Character data of variable length up to 2 gigabytes, or 2^{31} -1 bytes. Provided for backward compatibility.
12	`DATE`	Valid date range from January 1, 4712 BC, to December 31, 9999 AD. The default format is determined explicitly by the `NLS_DATE_FORMAT` parameter or implicitly by the `NLS_TERRITORY` parameter. The size is fixed at 7 bytes. This data type contains the datetime fields `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, and `SECOND`. It does not have fractional seconds or a time zone.
100	`BINARY_FLOAT`	32-bit floating point number. This data type requires 4 bytes.
101	`BINARY_DOUBLE`	64-bit floating point number. This data type requires 8 bytes.
180	`TIMESTAMP` [(*`fractional_seconds_precision`*)]	Year, month, and day values of date, as well as hour, minute, and second values of time, where *`fractional_seconds_precision`* is the number of digits in the fractional part of the `SECOND` datetime field. Accepted values of *`fractional_seconds_precision`* are 0 to 9. The default is 6. The default format is determined explicitly by the `NLS_TIMESTAMP_FORMAT` parameter or implicitly by the `NLS_TERRITORY` parameter. The size is 7 or 11 bytes, depending on the precision. This data type contains the datetime fields `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, and `SECOND`. It contains fractional seconds but does not have a time zone.

Table 6–1 (Cont.) Built-in Data Type Summary

Code	Data Type	Description
181	TIMESTAMP [(*fractional_seconds_precision*)] WITH TIME ZONE	All values of TIMESTAMP as well as time zone displacement value, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values are 0 to 9. The default is 6. The default format is determined explicitly by the NLS_TIMESTAMP_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The size is fixed at 13 bytes. This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, and TIMEZONE_MINUTE. It has fractional seconds and an explicit time zone.
231	TIMESTAMP [(*fractional_seconds_precision*)] WITH LOCAL TIME ZONE	All values of TIMESTAMP WITH TIME ZONE, with the following exceptions: ▪ Data is normalized to the database time zone when it is stored in the database. ▪ When the data is retrieved, users see the data in the session time zone. The default format is determined explicitly by the NLS_TIMESTAMP_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The size is 7 or 11 bytes, depending on the precision.
182	INTERVAL YEAR [(*year_precision*)] TO MONTH	Stores a period of time in years and months, where *year_precision* is the number of digits in the YEAR datetime field. Accepted values are 0 to 9. The default is 2. The size is fixed at 5 bytes.
183	INTERVAL DAY [(*day_precision*)] TO SECOND [(*fractional_seconds_precision*)]	Stores a period of time in days, hours, minutes, and seconds, where ▪ *day_precision* is the maximum number of digits in the DAY datetime field. Accepted values are 0 to 9. The default is 2. ▪ *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND field. Accepted values are 0 to 9. The default is 6. The size is fixed at 11 bytes.
23	RAW(*size*)	Raw binary data of length *size* bytes. You must specify *size* for a RAW value. Maximum *size* is: ▪ 32767 bytes if MAX_STRING_SIZE = EXTENDED ▪ 2000 bytes if MAX_STRING_SIZE = STANDARD Refer to *Oracle Database SQL Language Reference* for more information on the MAX_STRING_SIZE initialization parameter.
24	LONG RAW	Raw binary data of variable length up to 2 gigabytes.
69	ROWID	Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn.
208	UROWID [(*size*)]	Base 64 string representing the logical address of a row of an index-organized table. The optional *size* is the size of a column of type UROWID. The maximum size and default is 4000 bytes.
96	CHAR [(*size* [BYTE | CHAR])]	Fixed-length character data of length *size* bytes or characters. Maximum *size* is 2000 bytes or characters. Default and minimum *size* is 1 byte. BYTE and CHAR have the same semantics as for VARCHAR2.

Table 6–1 (Cont.) Built-in Data Type Summary

Code	Data Type	Description
96	`NCHAR[(`*size*`)]`	Fixed-length character data of length *size* characters. The number of bytes can be up to two times *size* for `AL16UTF16` encoding and three times *size* for `UTF8` encoding. Maximum *size* is determined by the national character set definition, with an upper limit of 2000 bytes. Default and minimum *size* is 1 character.
112	`CLOB`	A character large object containing single-byte or multibyte characters. Both fixed-width and variable-width character sets are supported, both using the database character set. Maximum size is (4 gigabytes - 1) * (database block size).
112	`NCLOB`	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is (4 gigabytes - 1) * (database block size). Stores national character set data.
113	`BLOB`	A binary large object. Maximum size is (4 gigabytes - 1) * (database block size).
114	`BFILE`	Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.

See Also: *Oracle Database SQL Language Reference* for more information about built-in data types

Oracle-Supplied Data Types

This section shows the syntax for the Oracle-supplied data types.

any_types

```
{ SYS.AnyData | SYS.AnyType | SYS.AnyDataSet }
```

media_types

```
{ ORDAudio
| ORDImage
| ORDVideo
| ORDDoc
| ORDDicom
| still_image_object_types
}
```

spatial_types

```
{ SDO_Geometry | SDO_Topo_Geometry |SDO_GeoRaster }
```

XML_types

```
{ XMLType | URIType }
```

Converting to Oracle Data Types

SQL statements that create tables and clusters can also use ANSI data types and data types from the IBM products SQL/DS and DB2. Oracle recognizes the ANSI or IBM data type name that differs from the Oracle data type name, records it as the name of the data type of the column, and then stores the column data in an Oracle data type based on the conversions shown in the following table.

Table 6–2 ANSI Data Types Converted to Oracle Data Types

ANSI SQL Data Type	Oracle Data Type

`CHARACTER(n)`
`CHAR(n)`
	`CHAR(n)`

`CHARACTER VARYING(n)`
`CHAR VARYING(n)`
	`VARCHAR2(n)`

`NATIONAL CHARACTER(n)`
`NATIONAL CHAR(n)`
`NCHAR(n)`
	`NCHAR(n)`

`NATIONAL CHARACTER VARYING(n)`
`NATIONAL CHAR VARYING(n)`
`NCHAR VARYING(n)`
	`NVARCHAR2(n)`

`NUMERIC[(p,s)]` `DECIMAL[(p,s)]` (**Note 1**)	`NUMBER(p,s)`

`INTEGER`
`INT`
`SMALLINT`
	`NUMBER(p,0)`

`FLOAT` (**Note 2**)	`FLOAT(126)`
`DOUBLE PRECISION` (**Note 3**)	`FLOAT(126)`
`REAL` (**Note 4**)	`FLOAT(63)`

Notes:

1. The `NUMERIC` and `DECIMAL` data types can specify only fixed-point numbers. For those data types, the scale (s) defaults to 0.
2. The `FLOAT` data type is a floating-point number with a binary precision b. The default precision for this data type is 126 binary, or 38 decimal.
3. The `DOUBLE PRECISION` data type is a floating-point number with binary precision 126.
4. The `REAL` data type is a floating-point number with a binary precision of 63, or 18 decimal.

Do not define columns with the following SQL/DS and DB2 data types, because they have no corresponding Oracle data type:

- `GRAPHIC`
- `LONG VARGRAPHIC`
- `VARGRAPHIC`
- `TIME`

Note that data of type `TIME` can also be expressed as Oracle datetime data.

See Also: *Oracle Database SQL Language Reference* for more information on data types

7 Format Models

This chapter presents the format models for datetime and number data stored in character strings.

This chapter includes the following sections:

- Overview of Format Models
- Number Format Models
- Datetime Format Models

Overview of Format Models

A format model is a character literal that describes the format of `DATETIME` or `NUMBER` data stored in a character string. When you convert a character string into a datetime or number, a format model tells Oracle how to interpret the string.

See Also: *Oracle Database SQL Language Reference* for more information on format models

Number Format Models

You can use number format models:

- In the `TO_CHAR` function to translate a value of `NUMBER` data type to `VARCHAR2` data type
- In the `TO_NUMBER` function to translate a value of `CHAR` or `VARCHAR2` data type to `NUMBER` data type

Number Format Elements

A number format model is composed of one or more number format elements. The following table lists the elements of a number format model.

Table 7–1 Number Format Elements

Element	Example	Description
, (comma)	`9,999`	Returns a comma in the specified position. You can specify multiple commas in a number format model. **Restrictions:** ■ A comma element cannot begin a number format model. ■ A comma cannot appear to the right of a decimal character or period in a number format model.
. (period)	`99.99`	Returns a decimal point, which is a period (.) in the specified position. **Restriction:** You can specify only one period in a number format model.
$	`$9999`	Returns value with a leading dollar sign.

0	`0999`
	`9990`
		Returns leading zeros.
		Returns trailing zeros.

9	`9999`	Returns value with the specified number of digits with a leading space if positive or with a leading minus if negative. Leading zeros are blank, except for a zero value, which returns a zero for the integer part of the fixed-point number.
B	`B9999`	Returns blanks for the integer part of a fixed-point number when the integer part is zero (regardless of zeros in the format model).
C	`C999`	Returns in the specified position the ISO currency symbol (the current value of the `NLS_ISO_CURRENCY` parameter).
D	`99D99`	Returns in the specified position the decimal character, which is the current value of the `NLS_NUMERIC_CHARACTER` parameter. The default is a period (.). **Restriction:** You can specify only one decimal character in a number format model.
EEEE	`9.9EEEE`	Returns a value using in scientific notation.
G	`9G999`	Returns in the specified position the group separator (the current value of the `NLS_NUMERIC_CHARACTER` parameter). You can specify multiple group separators in a number format model. **Restriction:** A group separator cannot appear to the right of a decimal character or period in a number format model.
L	`L999`	Returns in the specified position the local currency symbol (the current value of the `NLS_CURRENCY` parameter).
MI	`9999MI`	Returns negative value with a trailing minus sign (-). Returns positive value with a trailing blank. **Restriction:** The MI format element can appear only in the last position of a number format model.
PR	`9999PR`	Returns negative value in <angle brackets>. Returns positive value with a leading and trailing blank. **Restriction:** The PR format element can appear only in the last position of a number format model.
RN rn	`RN` `rn`	Returns a value as Roman numerals in uppercase. Returns a value as Roman numerals in lowercase. Value can be an integer between 1 and 3999.

Table 7–1 (Cont.) Number Format Elements

Element	Example	Description
S	`S9999`	Returns negative value with a leading minus sign (-).
		Returns positive value with a leading plus sign (+).
	`9999S`	Returns negative value with a trailing minus sign (-).
		Returns positive value with a trailing plus sign (+).
		Restriction: The S format element can appear only in the first or last position of a number format model.
TM	`TM`	The text minimum number format model returns (in decimal output) the smallest number of characters possible. This element is case insensitive. The default is TM9, which returns the number in fixed notation unless the output exceeds 64 characters. If the output exceeds 64 characters, then Oracle Database automatically returns the number in scientific notation. **Restrictions:** ■ You cannot precede this element with any other element. ■ You can follow this element only with one 9 or one E (or e), but not with any combination of these. The following statement returns an error: `SELECT TO_CHAR(1234, 'TM9e') FROM DUAL;`
U	`U9999`	Returns in the specified position the Euro (or other) dual currency symbol, determined by the current value of the `NLS_DUAL_CURRENCY` parameter.
V	`999V99`	Returns a value multiplied by 10^n (and if necessary, round it up), where *n* is the number of 9's after the `V`.
X	`XXXX` `xxxx`	Returns the hexadecimal value of the specified number of digits. If the specified number is not an integer, then Oracle Database rounds it to an integer. **Restrictions:** ■ This element accepts only positive values or 0. Negative values return an error. ■ You can precede this element only with 0 (which returns leading zeroes) or FM. Any other elements return an error. If you specify neither 0 nor FM with X, then the return always has one leading blank. Refer to *Oracle Database SQL Language Reference* for information on the FM format model modifier.

See Also: *Oracle Database SQL Language Reference* for more information on number format models

Datetime Format Models

You can use datetime format models:

- In the `TO_CHAR`, `TO_DATE`, `TO_TIMESTAMP`, `TO_TIMESTAMP_TZ`, `TO_YMINTERVAL`, and `TO_DSINTERVAL` datetime functions to translate a character string that is in a format other than the default datetime format into a `DATETIME` value
- In the `TO_CHAR` function to translate a `DATETIME` value that is in a format other than the default datetime format into a character string

Datetime Format Elements

A datetime format model is composed of one or more datetime format elements. The following table lists the elements of a date format model.

Table 7–2 Datetime Format Elements

Element	TO_* datetime functions?	Description

`-`
`/`
`,`
`.`
`;`
`:`
`"text"`
	Yes	Punctuation and quoted text is reproduced in the result.

`AD` `A.D.`	Yes	AD indicator with or without periods.
`AM` `A.M.`	Yes	Meridian indicator with or without periods.
`BC` `B.C.`	Yes	BC indicator with or without periods.
`CC` `SCC`		Century. ■ If the last 2 digits of a 4-digit year are between 01 and 99 (inclusive), then the century is one greater than the first 2 digits of that year. ■ If the last 2 digits of a 4-digit year are 00, then the century is the same as the first 2 digits of that year. For example, 2002 returns 21; 2000 returns 20.
`D`	Yes	Day of week (1-7). This element depends on the NLS territory of the session.
`DAY`	Yes	Name of day.
`DD`	Yes	Day of month (1-31).
`DDD`	Yes	Day of year (1-366).
`DL`	Yes	Returns a value in the long date format, which is an extension of Oracle Database's `DATE` format, determined by the current value of the `NLS_DATE_FORMAT` parameter. Makes the appearance of the date components (day name, month number, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` parameters. For example, in the `AMERICAN_AMERICA` locale, this is equivalent to specifying the format `'fmDay, Month dd, yyyy'`. In the `GERMAN_GERMANY` locale, it is equivalent to specifying the format '`fmDay, dd. Month yyyy`'. **Restriction:** You can specify this format only with the `TS` element, separated by white space.
`DS`	Yes	Returns a value in the short date format. Makes the appearance of the date components (day name, month number, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` parameters. For example, in the `AMERICAN_AMERICA` locale, this is equivalent to specifying the format '`MM/DD/RRRR`'. In the `ENGLISH_UNITED_KINGDOM` locale, it is equivalent to specifying the format '`DD/MM/RRRR`'. **Restriction:** You can specify this format only with the `TS` element, separated by white space.
`DY`	Yes	Abbreviated name of day.
`E`	Yes	Abbreviated era name (Japanese Imperial, ROC Official, and Thai Buddha calendars).
`EE`	Yes	Full era name (Japanese Imperial, ROC Official, and Thai Buddha calendars).

Table 7–2 (Cont.) Datetime Format Elements

Element	TO_* datetime functions?	Description
`FF [1..9]`	Yes	Fractional seconds; no radix character is printed. Use the X format element to add the radix character. Use the numbers 1 to 9 after FF to specify the number of digits in the fractional second portion of the datetime value returned. If you do not specify a digit, then Oracle Database uses the precision specified for the datetime data type or the data type's default precision. Valid in timestamp and interval formats, but not in `DATE` formats. **Examples:** `'HH:MI:SS.FF'` `SELECT TO_CHAR(SYSTIMESTAMP, 'SS.FF3') from dual;`
`FM`	Yes	Returns a value with no leading or trailing blanks. **See Also**: *Oracle Database SQL Language Reference* for more information on the FM format model modifier
`FX`	Yes	Requires exact matching between the character data and the format model. **See Also**: *Oracle Database SQL Language Reference* for more information on the FX format model modifier
`HH` `HH12`	Yes	Hour of day (1-12).
`HH24`	Yes	Hour of day (0-23).
`IW`		Week of year (1-52 or 1-53) based on the ISO standard.

`IYY`
`IY`
`I`
		Last 3, 2, or 1 digit(s) of ISO year.

`IYYY`		4-digit year based on the ISO standard.
`J`	Yes	Julian day; the number of days since January 1, 4712 BC. Number specified with J must be integers.
`MI`	Yes	Minute (0-59).
`MM`	Yes	Month (01-12; January = 01).
`MON`	Yes	Abbreviated name of month.
`MONTH`	Yes	Name of month.
`PM` `P.M.`	Yes	Meridian indicator with or without periods.
`Q`		Quarter of year (1, 2, 3, 4; January - March = 1).
`RM`	Yes	Roman numeral month (I-XII; January = I).
`RR`	Yes	Lets you store 20th century dates in the 21st century using only two digits. **See Also:** *Oracle Database SQL Language Reference* for more information on the RR datetime format element
`RRRR`	Yes	Round year. Accepts either 4-digit or 2-digit input. If 2-digit, provides the same return as RR. If you do not want this functionality, then enter the 4-digit year.
`SS`	Yes	Second (0-59).
`SSSSS`	Yes	Seconds past midnight (0-86399).

Table 7–2 (Cont.) Datetime Format Elements

Element	TO_* datetime functions?	Description
TS	Yes	Returns a value in the short time format. Makes the appearance of the time components (hour, minutes, and so forth) depend on the NLS_TERRITORY and NLS_LANGUAGE initialization parameters. **Restriction:** You can specify this format only with the DL or DS element, separated by white space.
TZD	Yes	Daylight saving information. The TZD value is an abbreviated time zone string with daylight saving information. It must correspond with the region specified in TZR. Valid in timestamp and interval formats, but not in DATE formats. **Example:** PST (for US/Pacific standard time); PDT (for US/Pacific daylight time).
TZH	Yes	Time zone hour. (See TZM format element.) Valid in timestamp and interval formats, but not in DATE formats. **Example:** 'HH:MI:SS.FFTZH:TZM'.
TZM	Yes	Time zone minute. (See TZH format element.) Valid in timestamp and interval formats, but not in DATE formats. **Example:** 'HH:MI:SS.FFTZH:TZM'.
TZR	Yes	Time zone region information. The value must be one of the time zone regions supported in the database. Valid in timestamp and interval formats, but not in DATE formats. **Example:** US/Pacific
WW		Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W		Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
X	Yes	Local radix character. **Example:** 'HH:MI:SSXFF'.
Y,YYY	Yes	Year with comma in this position.
YEAR SYEAR		Year, spelled out; S prefixes BC dates with a minus sign (-).
YYYY SYYYY	Yes	4-digit year; S prefixes BC dates with a minus sign.
YYY YY Y	Yes	Last 3, 2, or 1 digit(s) of year.

See Also: *Oracle Database SQL Language Reference* for more information on datetime format models

A SQL*Plus Commands

This appendix presents many of the SQL*Plus commands.

This appendix includes the following section:

- SQL*Plus Commands

SQL*Plus Commands

SQL*Plus is a command-line tool that provides access to the Oracle RDBMS. SQL*Plus enables you to:

- Enter SQL*Plus commands to configure the SQL*Plus environment
- Startup and shutdown an Oracle database
- Connect to an Oracle database
- Enter and execute SQL commands and PL/SQL blocks
- Format and print query results

SQL*Plus is available on several platforms.

The commands shown in Table A–1 are SQL*Plus commands available in the command-line interface. Not all commands or command parameters are shown.

See Also:

- *SQL*Plus Quick Reference*
- *SQL*Plus User's Guide and Reference*

Table A–1 Basic SQL*Plus Commands

Database Operation	SQL*Plus Command
Log in to SQL*Plus	`SQLPLUS [ [{username[/password][@connect_identifier] | / } [AS {SYSASM|SYSBACKUP|SYSDBA|SYSDG|SYSOPER|SYSKM}] [edition=value]] | /NOLOG ]`
List help topics available in SQL*Plus	`HELP [ INDEX | topic ]`
Execute host commands	`HOST [ command ]`
Show SQL*Plus system variables or environment settings	`SHOW { ALL | ERRORS | USER | system_variable [, system_variable] ...}`

Table A–1 (Cont.) Basic SQL*Plus Commands

Database Operation	SQL*Plus Command
Alter SQL*Plus system variables or environment settings	`SET system_variable value`

Start up a database	`STARTUP { db_options | cdb_options | upgrade_options }`
	
	Where `db_options` has the following syntax:
	`[FORCE] [RESTRICT] [PFILE=filename] [QUIET]`
	`[ MOUNT [dbname] | [ OPEN [open_db_options] [dbname] ]`
	`| NOMOUNT ]`
	Where `open_db_options` has the following syntax:
	`READ {ONLY | WRITE [RECOVER]} | RECOVER`
	Where `cdb_options` has the following syntax:
	`root_connection_options | pdb_connection_options`
	Where `root_connection_options` has the following syntax:
	`PLUGGABLE DATABASE pdbname [FORCE] | [RESTRICT]`
	`[ OPEN {open_pdb_options} ]`
	Where `pdb_connection_options` has the following syntax:
	`[FORCE] | [RESTRICT] [ OPEN {open_pdb_options} ]`
	Where `open_pdb_options` has the following syntax:
	`READ WRITE | READ ONLY`
	Where `upgrade_options` has the following syntax:
	`[PFILE=filename] {UPGRADE | DOWNGRADE} [QUIET]`

Connect to a database	`CONNECT [{username[/password] [@connect_identifier] | /`
	`         | proxy_user **[** username **]** [/password]`
	`         [@connect_identifier]}`
	`         [AS {SYSASM|SYSBACKUP|SYSDBA|SYSDG|SYSOPER|SYSKM}]`
	`         [edition=value]`
	`        ]`
	
	Note: The square brackets shown in boldface type are part of the syntax and do not imply optionality.

List column definitions for a table, view, or synonym, or specifications for a function or procedure	`DESCRIBE [ schema. ] object`
Edit contents of the SQL buffer or a file	`EDIT [ filename [ .ext ] ]`
Get a file and load its contents into the SQL buffer	`GET filename [ .ext ] [ LIST | NOLLIST ]`
Save contents of the SQL buffer to a file	`SAVE filename [ .ext ] [ CREATE | REPLACE | APPEND ]`
List contents of the SQL buffer	`LIST [ n | n m | n LAST ]`
Delete contents of the SQL buffer	`DEL [ n | n m | n LAST ]`
Add new lines following current line in the SQL buffer	`INPUT [ text ]`

Table A–1 (Cont.) Basic SQL*Plus Commands

Database Operation	SQL*Plus Command
Append text to end of current line in the SQL buffer	`APPEND` *`text`*
Find and replace first occurrence of a text string in current line of the SQL buffer	`CHANGE` *`sepchar old`* `[` *`sepchar`* `[` *`new`* `[` *`sepchar`* `] ] ]` *sepchar* can be any nonalphanumeric ASCII character such as "/" or "!"

Capture query results in a file and, optionally, send contents of file to default printer	`SPOOL [` *`filename`* `[` *`.ext`* `]`
	`  [ CREATE | REPLACE | APPEND | OFF | OUT ]`

Run SQL*Plus statements stored in a file	`@ {` *`url`* `|` *`filename`* `[` *`.ext`* `] } [` *`arg`* `... ]`
	`START {` *`url`* `|` *`filename`* `[` *`.ext`* `] } [` *`arg`* `... ]`
	
	ext can be omitted if the filename extension is .sql

Execute commands stored in the SQL buffer	`/`
List and execute commands stored in the SQL buffer	`RUN`
Execute a single PL/SQL statement or run a stored procedure	`EXECUTE` *`statement`*
Disconnect from a database	`DISCONNECT`
Shut down a database	`SHUTDOWN [ ABORT | IMMEDIATE | NORMAL | TRANSACTIONAL [LOCAL] ]`

Log out of SQL*Plus	`{ EXIT | QUIT }`
	` [ SUCCESS | FAILURE | WARNING |` *`n`* `|` *`variable`* `| :`*`BindVariable`* `]`
	` [ COMMIT | ROLLBACK ]`

Index